Universidad Nacional Autónoma de Nicaragua
UNAN-León

Tesis de Maestría en Computación

Maestría impartida por la UAH-España en la UNAN-León

PLAN DOCENTE PARA LAS ASIGNATURAS

INTRODUCCIÓN A LAS BASES DE DATOS
INGENIERIA DE SOFTWARE

TITULACIÓN INGENIERÍA EN SISTEMA DE INFORMACIÓN

Autor: William Noel Martinez Orozco
Director: Óscar García Población

León, Julio de 2004

PLAN DOCENTE DE BASES DE DATOS

1 ÍNDICE GENERAL

1 ÍNDICE GENERAL 2
2 ÍNDICE DE FIGURAS 8
3 INTRODUCCIÓN 11
4 OBJETIVOS 12
5 SITUACIÓN DE LA ASIGNATURA EN EL PLAN DE ESTUDIOS DEL INTAE – LEON- INATEC 13
5.1 Relación del módulo Introducción a las Bases de Datos con otras asignaturas 14
5.1.1 Fundamentos de la Programación 14
5.1.2 Programación I 14
5.1.3 Análisis y diseño de sistemas 15
5.1.4 Programación II 15
5.1.5 Ingeniería del software 15
6 METODOLOGÍA Y MATERIAL DIDÁCTICO UTILIZADO 16
6.1 Metodologías 16
6.1.1 Características del Curso Directo 16
6.1.2 Característica de Taller Didáctico 16
6.1.3 Características del Laboratorio de Aprendizaje 17
6.1.4 Circulo de Estudios 17
6.2 Técnicas didácticas 18
6.2.1 Expositiva 18
6.2.2 Lectura Comentada 18
6.2.3 Demostrativa 19
6.2.4 Discusión en Grupos Pequeños 19
7 PROCESO DE EVALUACIÓN 20
8 TEMPORIZACIÓN 21
8.1 División de los laboratorios 22
9 DESARROLLO TEMÁTICO DE CADA CLASE TEÓRICA 23
9.1 Clase 1: Papel económico, social y cultural de la información 23
9.1.1 Papel económico, social y cultural de la información 23
9.1.2 Cualidades de la información 25
9.1.3 Bibliografía comentada 27

9.2 Clase 2: Introducción a base de datos 27
9.2.1 Concepto de sistema de información 27
9.2.2 Definición de sistema 27
9.2.3 Concepto de sistema de información 29
9.2.4 Las Características de un SI, según Bubenko (1980) 31
9.2.5 Bibliografía comentada 32
9.3 Clase 3: introducción a las bases de datos 33
9.3.1 1.7 Componentes de un sistema de información 33
9.3.2 Sistemas de información para la gestión y sistemas de información para la ayuda a la decisión 35
9.3.3 Niveles de gestión 35
9.3.4 Tipología de los procesos de gestión 37
9.3.5 Bibliografía comentada 38
9.4 Clase 4: introducción a las Base de Datos 39
9.4.1 De los sistemas tradicionales de ficheros a las base de datos 39
9.4.2 Ventajas e inconvenientes de las bases de datos frente a los ficheros clásicos 41
9.4.3 Las ventajas de los sistemas de bases de datos 41
9.4.4 Ventajas de las bases de datos 43
9.4.5 Inconvenientes en el cambio en la orientación del SI 43
9.4.6 Bibliografía comentada 44
9.5 Clase 5: Introducción a las bases de datos 45
9.5.1 Concepto de base de datos 45
9.5.2 Distintos niveles de abstracción de una base de datos 48
9.5.3 Estructura lógicas de usuarios: esquema externo 48
9.5.4 Estructura lógica global: esquema conceptual 49
9.5.5 Estructura física: esquema interno 49
9.5.6 Bibliografía comentada 49
9.6 Clase 6: Historia 50
9.6.1 Origen y finalidad de las Bases de datos (BBDD) 50
9.6.2 Conveniencia de la separación en SGBD 51
9.6.3 Abstracción y modelos de datos 52
9.6.4 Bibliografía comentada 52

9.7 Clase 7: Sistema de gestión de la base de datos 53
9.7.1 El sistema de gestión de la base de datos 53
9.7.2 El SGBD como interfaz entre el usuario y la base de datos 53
9.7.3 Clasificación de los distintos tipos de usuarios de una base de datos 54
9.8 Clase 8: Sistema de Gestión de la Base de Datos 57
9.8.1 Lenguajes de los SGBD 57
9.9 Clase 9: Historia 60
9.9.1 Conceptos y principales funciones de un SGBD 60
9.9.2 Operaciones típicas que deben realizar un SGBD 60
9.9.3 Funciones de definición o descripción 61
9.9.4 Función de manipulación 62
9.9.5 Consulta a la base de datos 62
9.9.6 Función de control 63
9.9.7 Los componentes funcionales de un sistema de base de datos 65
9.10 Clase 10: Historia 68
9.10.1 Perspectiva funcional 68
9.10.2 Bibliografía comentada 69
9.11 Clase 11: Historia 70
9.11.1 Perspectiva tecnológica 70
9.11.2 Las Tres etapas fundamentales en la evolución 70
9.11.3 Bibliografía comentada 74
9.12 Clase 12: DISEÑO DE BBDD 75
9.12.1 Terminología BD 75
9.12.2 Diseño de BBDD y Ciclo de vida de SI 76
9.12.3 Ciclo de Vida típico de SI: 77
9.12.4 Bibliografía comentada 77
9.13 Clase 13: DISEÑO DE BBDD 78
9.13.1 Fases de diseño de BBDD 78
9.13.2 Diseño D-Dato 78
9.13.3 Bibliografía comentada 79
9.14 Clase 14: Modelo de datos 80
9.14.1 Concepto de modelo de datos 80
9.14.2 Introducción 80

9.14.3 Definición de modelo de datos 83
9.14.4 Estática 83
9.14.5 Dinámica 84
9.14.6 Una operación tiene dos componentes: 84
9.14.7 Plano conceptual para expresar una sentencia 85
9.15 Clase 15: Modelo de datos 86
9.15.1 Las restricciones de integridad en los modelos de datos 86
9.15.2 Clasificación de los modelos de datos 88
9.15.3 Niveles de abstracción de la arquitectura ANSI 89
9.15.4 Características de los modelos convencionales y conceptuales. 90
9.16 Clase 16: Diseño de BBDD 92
9.16.1 Mecanismos de abstracción de los Modelos de Datos para trabajar: 92
9.16.2 Abstracción 92
9.16.3 Clasificación, agregación, generalización 93
9.16.4 Ejemplo de pertenencia múltiple: 94
9.17 Clase 17: DISEÑO DE BBDD 95
9.18 Clase 18: Modelo entidad – relación 99
9.18.1 Modelo entidad – relación 99
9.18.2 Entidades y conjunto de entidades 100
9.18.3 Atributos 102
9.18.4 Atributos simples y compuestos 103
9.19 Clase 19: Modelo entidad – relación 104
9.20 Clase 20: Modelo entidad – relación 108
9.20.1 Restricciones de asignación (MAPPING) 108
9.20.2 Dependencias de existencias 110
9.21 Clase 21: Modelo entidad – relación 111
9.21.1 Claves 111
9.22 Clase 22: DISEÑO DE BBDD 120
9.22.1 Modelo E/I 120
9.22.2 Elementos básicos del modelo 121
9.23 Clase 23: Modelo entidad – relación 123
9.23.1 Diagrama entidad-relación 123
9.23.2 Reducción de los diagramas E-R a tablas 125

9.24 Clase 24: Modelo entidad – relación 127
9.24.1 Relaciones 127
9.24.2 Ejemplo de un Modelo Entidad-Relación 130
9.24.3 Diseño de un esquema de bases de datos Entidad - Relación. 136
9.25 Clase 25: DISEÑO DE BBDD 137
9.25.1 Etapa de una metodología de diseño 137
9.25.2 Transformación del esquema Conceptual al Relacional 139
9.26 Clase 26 DISEÑO DE BBDD 144
9.26.1 Teoría de la normalización 144
9.26.2 Problemas que puede presentar un esquema relacional 144
9.26.3 Efectos que produce la redundancia produce 145
9.26.4 Nociones intuitiva de las formas normales 147
9.27 Clase 27 DISEÑO DE BBDD 150
9.27.1 Dependencia Funcionales 150
9.27.2 Dependencia funcional plena o completa 152
9.27.3 Dependencia funcional transitiva 153
9.28 Clase 28 DISEÑO DE BBDD 155
9.28.1 Definición formal de las tres primeras formas normales 155
9.28.2 Descomposición de relaciones 157
9.29 Clase 29 DISEÑO DE BBDD 159
9.29.1 Descomposición sin pérdida de dependencia funcional 161
10 Prácticas de laboratorio 163
10.1 Laboratorio I 164
10.1.1 Tema: Requerimientos: 164
10.1.2 Competencia o Habilidad 164
10.1.3 Reedición del enunciado del problema: 168
10.2 Laboratorio II 170
10.3 Laboratorio III 172
10.3.1 Tema: Refinamiento del esquema previo, usando las técnicas: 172
10.3.2 Competencia o habilidad 172
10.4 Laboratorio IV 174
10.4.1 Tema: Esquema Final del modelo Entidad / Relación 174
10.4.2 Competencia o habilidad 174

10.5 Laboratorio V 176
10.5.1 Tema: Documentación Adicional 176
10.5.2 Competencia o habilidad 176
10.6 Laboratorio VI 181
10.6.1 Tema: Paso a Modelo Relacional 181
10.6.2 Competencia o habilidad 181

2 ÍNDICE DE FIGURAS

Figura 1.1 límites del medio ambiente o entorno del sistema 29
Figura 1.2 Sistemas dinámicos, que están en interacción con el entorno 30
Figura 1.3 Sistemas eminentemente dinámicos 31
Figura 1.4 Componentes de un sistema de información 33
Figura 1.5 Transmisión de la información 36
Figura 1.6 Sistema de información único (nivel directivo y operacional) 38
Figura 1.7 Organización clásica: Sistema orientado al proceso 39
Figura 1.9 Organización en B.D.:Sistema orientados a los datos 40
Figura 1.9 Las tres estructuras de los sistemas de bases de datos 48
Figura 1.10. Separación entre proceso y datos 50
Figura 1.11. Separación entre proceso dato y gestión de datos 50
Figura 1.12. Segunda separación 51
Figura 1.13 El SGBD como interfaz entre la base de datos y los niveles de gestión de la organización 54
Figura 1.14 Tipología de los lenguajes de los SGBD 57
Figura 1.15 Operaciones típicas sobre una base de datos 61
Figura 1.16 Interacción Usuario / Base de Datos 63
Figura 1.17 Funciones esenciales de un SGBD 64
Figura 1.18 Separación entre proceso dato y gestión de datos 66
Figura 1.19 Conjunto de Modelo y Producto 72
Figura 1.20 Manipulación de datos 73
Figura 1.21 Ciclo de Vida de un SI 77
Figura 1.22 Diseño de Datos 79
Figura 1.23 Tipos de modelos de datos corresponden a cada nivel de abstracción de una arquitectura a tres niveles 81
Figura 124 Clasificación de los modelos de datos globales 82
Figura 1.25 Aplicación de un modelo de datos a un mundo real para obtener un esquema 82
Figura 1.26 Migración de la semántica de los datos 87
Figura 1.27 Semántica de los datos “dispersa” entre los diferentes programas de aplicación, frente a semántica integrada en la base de datos 88

Figura 1.28 Abstracción de un objeto....92
Figura 1.29 Ejemplo de Clasificación....93
Figura 1.30 Ejemplo de Pertenencia Múltiple....94
Figura 1.31 Ejemplo de Agregación Especial....94
Figura 1.33 Relación entre Clasificación y Agregación....96
Figura 1.34 Ejemplo de Introducción a la Cardinalidad....97
Figura 1.35 Atributos Simples y Compuestos....103
Figura 1.36 Ejemplo de Atributos de Profesor....105
Figura 1.37 Ejemplo de Atributos de Película....106
Figura 1.38 Ejemplo de Atributos de Automóvil....107
Figura 1.39 Relación de una a una....109
Figura 1.40 Relación de una a muchas....109
Figura 1.41 Relación de muchas a una.....109
Figura 1.42 Relación de muchas a muchas....110
Figura 1.43 Ejemplo de Atributos de Departamento....115
Figura 1.44 Ejemplo de Atributos de Alumno....116
Figura 1.45 Ejemplo de Película con super clave....117
Figura 1.46 Ejemplo de Atributos de Propietarios....119
Figura 1.47 D*i*agrama E-R....124
Figura 1.48 Relaciones una a muchas y muchas a una....124
Figura 1.49 Relaciones una a una.....125
Figura 1.50 Diagrama E-R con indicadores de papeles....125
Figura 1.51 Diagrama E-R con un conjunto de entidades débil.....125
Figura 1.52 Diagrama E-R con una relación ternaria.....126
Figura 1.53 Diagrama E-R....126
Figura 1.54 Diagrama E-R....128
Figura 1.55 Ejemplos de Relaciones....129
Figura 1.56 Diagrama E-R....130
Figura 1.57 Diagrama E-R....133
Figura 1.58 Diagrama E-R....135
Figura 1.59 Representación grafica de la metodología propuesta para el diseño de base de datos relacionales.....138
Figura 1.60 Ejemplo de paso de ME/R al modelo relacional....140

Figura 1.61 Ejemplo de opciones de clave ajena en una transformación por propagación de clave 141

Figura 1.62 Opción de una clave ajena en la transformación de una dependencia en existencia 142

Figura 1.63 Opciones de clave ajena en la transformación de una interrelación N:M 142

Figura 1.64 Ejemplo de grafo relacional para el esquema de una biblioteca 143

Figura 1.65 Ejemplo de diagrama de dependencias funcionales 151

Figura 1.66 Ejemplo de dependencia funcional transitiva 153

3 INTRODUCCIÓN

Este Proyecto Docente esta destinado para ser impartido en los Centros Técnicos a Nivel Nacional, siendo su órgano rector el Instituto Nacional Tecnológico INATEC en la modalidad de Instituto Técnico de Administración y Economía – INTAE, por lo que las siglas para identificar el centro es INTAE –León – INATEC.

La asignatura de Introducción a las Bases de Datos se imparte actualmente con una duración 30 horas. En este proyecto se propone que se realice en 60 horas. El objetivo es consolidar los contenidos y abarcar temas que actualmente no están incluidos Ej. Fases de requerimientos, Modelos Conceptuales entre otros.

Este proyecto incluye junto con el plan docente otros aspectos importantes, como la situación de la asignatura en el plan de estudios de la Especialidad y su relación con otras asignaturas, en donde se presentan los conocimientos previos que el alumno debe adquirir para poder aprovechar los conocimientos de la asignatura de Bases de Datos. Todas las clases están estructuradas para ser impartidas en un tiempo de 2 horas cada una.

El plan docente está claramente diferenciado en las dos partes siguientes:

- **PARTE TEÓRICA:** en esta parte se desarrollan todos los conceptos y definiciones abstractos, con numerosos ejemplos orientados a facilitar el aprendizaje por parte de los alumnos.
- **PARTE PRÁCTICA:** donde el alumno podrá poner en práctica los conocimientos adquiridos en la parte de teoría.

También se detalla al estudiante toda la bibliografía utilizada al final de cada tema, con el fin de ayudar en la búsqueda de información cuando se desee profundizar en algún aspecto concreto. Por otro lado se describe la distribución temporal con la que se debe impartir cada tema, con objeto de cumplir las restricciones de tiempo impuestas por el plan de estudios.

4 OBJETIVOS

Los objetivos de esta asignatura son:

- Definir los conceptos básicos de las Bases de Datos.
- Introducir al estudiante en los pasos necesarios para modelar problemas de forma que puedan resolverse utilizando herramientas informáticas.
- Aplicar los conocimientos adquiridos en planteamiento de problemas basados en la realidad.
- Formar profesionales capaces de interpretar análisis realizados usando herramientas de modelos en las bases de datos.

5 SITUACIÓN DE LA ASIGNATURA EN EL PLAN DE ESTUDIOS DEL INTAE – LEON- INATEC

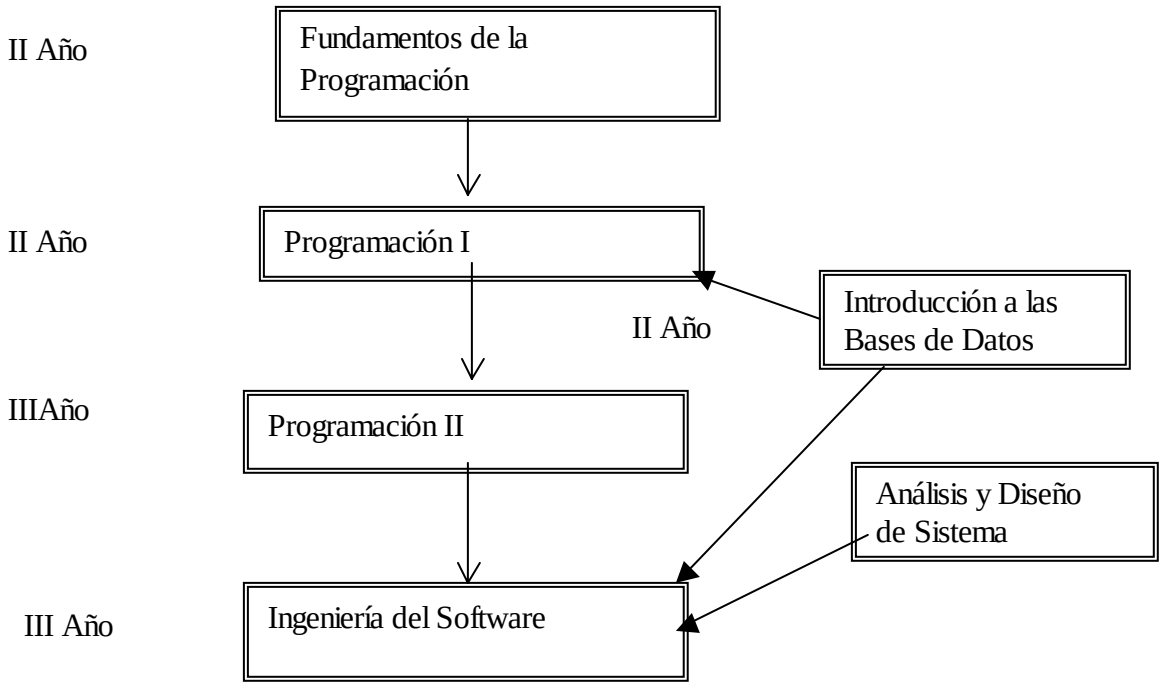
La asignatura de Base de Datos se imparte actualmente en la carrera de Técnico Medio en Programación, que ofrece el Instituto Nacional Tecnológico.

La carrera se desarrolla a lo largo de 2 años y medio de estudios, para la culminación de uno de sus graduandos. El módulo Introducción a las Bases de Datos se recomienda que imparta en el 2º año del plan actual de estudios de la carrera, con una duración de 60 horas, y no como esta actualmente con 20 horas. El período lectivo para esta asignatura constará de 2 horas diarias, de las cuales 1 hora se dedicará a la parte teórica y 1 hora a la parte práctica.

Un aspecto muy importante que hay que mencionar es lo relacionado a los conocimientos previos que debe haber adquirido los estudiantes en años anteriores. Dichos conocimientos serán necesarios para asimilar los conocimientos de la asignatura Introducción a la Bases de Datos. En el siguiente apartado se describen las asignaturas más relevantes en este aspecto.

5.1 Relación del módulo Introducción a las Bases de Datos con otras asignaturas

En el gráfico presentado a continuación, se muestra la relación de la asignatura Bases de Datos con otras asignaturas:



A continuación se describirá de forma breve, el contenido de las asignaturas presentadas en el gráfico anterior y cómo se relacionan con las demás:

5.1.1 Fundamentos de la Programación

La asignatura Fundamento de la Programación, y sus respectivos laboratorios, tienen una fuerte relación con Bases de Datos, ya que en esta asignatura se presentan los conocimientos básicos de la programación. En ella los estudiantes identifican la secuencia lógica de los problemas, usando para ejemplificar el lenguaje de Programación C, desarrollando habilidades y destrezas en el análisis y planteamiento de un problema.

Se introduce al estudiante en el aprendizaje del Lenguaje C, dándole elementos generales, como los tipos de datos, operadores, funciones de entrada/salida, sentencias de control, tipos estructurados de datos y punteros etc.

5.1.2 Programación I

En esta asignatura, se introduce al alumno en los conceptos de la programación estructurada, conceptos que se llevan a la práctica con ayuda del lenguaje de programación

Visual Foxpro. Se abordan temas tales como características de la programación estructurada, entrada y salida estándar, operadores.

El estudiante al haber recibido Programación I, desarrolla herramientas necesarias para abordar un problema valiéndose del análisis estructurado, lo que le facilitara la comprensión de temas que se abordan en la Introducción a la Bases de Datos.

5.1.3 Análisis y diseño de sistemas

En esta materia se abarcan temas tales como: la evolución del software, modelos de proceso de software, análisis de requisitos, especificación de requisitos software, modelado de datos (atributos, relaciones, cardinalidad), modelado funcional, diccionario de datos, diseño, etc., con un enfoque estructurado.

5.1.4 Programación II

En esta asignatura, se introduce al alumno en los conceptos de la programación orientada a objetos, conceptos que se llevan a la práctica con ayuda del lenguaje de programación Visual Foxpro. Se abordan temas tales como: clase, objetos, controles, propiedades, eventos, sucesos herencias, polimorfismos, Encapsulamiento etc.

5.1.5 Ingeniería del software

En este módulo se dota al estudiantes de herramientas para el análisis de un problema usando técnicas de modelado orientado a objeto tales como: casos de uso, diagrama de secuencia, modelo conceptual, contratos etc. Para ello es necesario que el alumno domine conceptos tales como clase objetos etc.

6 METODOLOGÍA Y MATERIAL DIDÁCTICO UTILIZADO

6.1 Metodologías

Las metodologías más afines para ser aplicadas en la impartición de la asignatura Bases de Datos son las siguientes:

- Curso Directo
- Taller Didácticos
- Laboratorio de Aprendizajes
- Círculos de Estudios o Corrillos

A continuación se detallarán las características particulares de cada una de ellas.

6.1.1 Características del Curso Directo

- Centrada en el docente
- Sin experiencia de los participantes. Toma como base las exposiciones del instructor para proporcionar la información.
- Utiliza cuestionarios, corrillos, casos e investigaciones dirigidas para trasmitir la información.
- Usa pocos materiales didácticos complejos. El principal material de apoyo es el retroproyector, rotafolio, o pizarrón como auxiliar de la exposición, las laminas de rotafolio, mapas, afiches y otros recursos para facilitar la comunicación.

6.1.2 Característica de Taller Didáctico

- Facilita el desarrollo de habilidades para realizar tareas o procesos (Destreza, algunas tareas de orden Ej. Operaciones administrativos, financieras etc.).
- Facilita adquirir dominio de lo enseñado por medio de la práctica.
- Los participantes deben concluir el modulo formativo demostrando saber realizar las tareas o proceso las tareas o procesos que nos referimos son las actividades de aprendizaje que conducen al logro de las competencias.
- Es una metodología centrada en el instructor en su primera parte, centrada en el alumno en la segunda.
- Aplicable en situaciones grupales o individuales.
- Se adapta a programas elaborados con criterios por compentencias.

  Competencia es el nivel de conocimiento adquirido hasta el punto de saber hacer correctamente una tarea.

- El instructor debe atender la dosificación y complejidad de las tareas así como las operaciones.

- El instructor es ante todo un supervisor de los avances en el proceso de aprendizaje de sus alumnos.
- Requiere del instructor un dominio completo de los contenidos.
- Los recursos didácticos que utilizan los alumnos y el instructor son los materiales y equipos reales propios de la tarea.
- Para proporcionar información teórica el instructor puede utilizar diversos materiales impresos.

6.1.3 Características del Laboratorio de Aprendizaje

- Esta metodología está sustentada en las experiencias del participante.
- Aplica principios de practica - teoría - práctica.
- El alumno es más protagonista, adquiera experiencia y experimenta consigo mismo.
- El participante descubre aportando experiencias.
- Los participantes aprenden viviendo una experiencia y sistematizando los elementos envueltos en la experiencia.
- Útil para desarrolla de habilidades congnoscitivas y afectivas, simples y complejas
- Es una metodología efectiva para la modificación de actitudes.
- Los participantes deben volver a practicar después de ser retroalimentados.
- Aprovecha una variedad amplia de ejercicios y técnicas.
- Requiere de instructores calificados para dirigir el análisis de los hechos significativos.
- El docente debe manejar eficientemente las técnicas.
- Facilita, a la larga, aprendizaje colateral.

6.1.4 Circulo de Estudios

- Requiere apoyos didácticos como documentación de trabajo, guías de estudio, cuadernos de prácticas y ejercicios. Estos materiales deben ser preferentemente personalizados.
- Propios para eventos no escolarizados y favorables para personas que no disponen de tiempo completo de estudio.
- Se requiere el apoyo de un asesor pendiente del grupo.
- los ejercicios previstos en los materiales deben tener autoevaluaciones.

- El asesor no necesariamente asiste a todas las reuniones, pero está atento a los avances del círculo.
- Los materiales deben contener toda la información y todas las situaciones de ejercitación que los objetivos requieren.
- El asesor supervisa y complementa, evita dar clases.
- Puede ser aplicado en programas duales o en programas basados en competencias.
- Exige persistencia, fuerza de voluntad y cooperación de los alumnos.
- Usado frecuentemente desarrolla habilidades para la comunicación y el análisis de documentación y las actitudes de respeto y tolerancia a las ideas ajenas.
- Puede aprovechar el tiempo libre de los participantes.
- Útil para aprendizaje cognoscitivos simples y medianamente complejos.
- Permite aprendizaje de proceso e información técnica básica.

6.2 Técnicas didácticas

Las técnicas didácticas consideradas en este Proyecto Docente son las siguientes:

- Expositivas
- Lectura Comentadas
- Demostrativas

A continuación se describirán las características propias de cada una de ellas.

6.2.1 Expositiva

- Craig y Bittel dicen que la exposición es la “la forma en que un instructor presenta una información a un grupo de participantes, quienes adoptan una actitud receptiva”
- Alfaro Castro la define como el procedimiento didáctico mediante el cual el instructor, valiéndose principalmente de un lenguaje adecuado presenta a los alumnos un nuevo tema para que definiéndolo y analizándolo, adquieran la compresión inicial necesaria para su aprendizaje.
- Las definiciones hacen evidente la presentación oral de un tema que, después de ser recibido y analizado por los alumnos, les permitirá asimilarlo y aplicarlo.

6.2.2 Lectura Comentada

- Esta técnica se llama Lectura Comentada porque el maestro suscita y mantiene conversación alrededor de un tema previamente leído, con el propósito de hallar las ideas fundamentales de los párrafos (análisis) y llegar finalmente a la redacción del resumen del texto (síntesis).

- Esta técnica se utiliza mucho porque además de proporcionar la información requerida para lograr una competencia o un objetivo de aprendizaje, desarrolla la ejercitación de la expresión oral, contribuyen a que el alumno utilice el vocabulario adecuado, practique la lectura y la correcta pronunciación.

6.2.3 Demostrativa

- Es la ejecución de una tarea por un docente frente a sus alumnos, con el propósito de que éstos la practiquen bajo su supervisión hasta llegar a dominarla.
- Para dar cumplimiento al desarrollo de una demostración el docente da inicio a la ejecución de la demostración de la tarea, bajo la observación de sus alumnos, explicando y aclarando a su vez los puntos clave a guardar durante su realización. Seguidamente los alumnos repiten la tarea tal y como fue enseñada, bajo la supervisión del docente, quien corrige errores y retroalimenta de forma oportuna. Para alcanzar el dominio continúa ejercitando las veces que considere necesarios hasta dominarla por completo.

6.2.4 Discusión en Grupos Pequeños

Esta técnica consiste en la conversación y el debate de un grupo acerca de un tema de interés común. En la discusión intervienen todos los miembros del grupo en una comunicación cara a cara.

Se utilizarán las Técnicas y Metodologías dependiendo del tema a desarrollar en las clases teóricas o prácticas, para cumplir fielmente con la temporización establecida. También se propondrá la realización de trabajos en grupo, según se vaya avanzando en el contenido para ayudar al estudiante a profundizar sobre el tema. Principalmente para despertar en él su capacidad de investigación, así como fomentar el compañerismo y respeto a las opiniones ajenas al trabajar en grupo.

En lo relacionado al material didáctico, se hará uso de pizarra para la explicación de las clases teóricas. El estudiante podrá tener acceso a la bibliografía existente con objeto de profundizar en un determinado tema.

7 PROCESO DE EVALUACIÓN

La asignatura de Introducción a las Base de Datos está compuesta por las clases teóricas y por la parte práctica, que corresponde al laboratorio. A lo largo del curso se realizan evaluaciones por cada unidad. La nota final de la asignatura, se obtiene sumando la ponderación de cada unidad.

En cuanto a la obtención de cada una de las notas por unidad, el desglose es el siguiente:

- Realización de un examen teórico: 60 puntos.
- Evaluación de laboratorio: 40 puntos.

El desglose para la obtención de la nota final es el siguiente:

Unidad	Ponderación
Unidad I	10 %
Unidad II	10 %
Unidad III	20 %
Unidad IV	30 %
Unidad V	30 %
TOTAL	**100 %**

Como se explicó anteriormente, la realización de los laboratorios tiene un valor de 40 puntos en cada unidad. Por cada práctica que se realice el estudiante, deberá entregar un informe que contendrá lo siguiente:

- Enunciado del ejercicio.
- Explicación de la solución dada al ejercicio.
- Solución del ejercicio.

Además del informe, el estudiante deberá el archivo con la solución del problema, usando algún soporte informático por Ej. Disco flexible, CD, etc.

La mayor parte de la evaluación consiste en la realización de un examen teórico, el cual podrá ser de los siguientes tipos:

- Opción Múltiple.
- Respuestas Alterna
- Correspondencia
- Ordenamiento
- Respuestas Breves
- Completación

La preferencia en los tipos de pregunta serán por Completación y Respuesta Alterna.

8 TEMPORIZACIÓN

La materia Base de Datos se imparte como final del II año de la carrera Técnico Medio en Programación, ofrecida por el INTAE – León - INATEC. Consta de 10 horas semanales, que se dividen en 6 horas de teoría y 4 de laboratorio. El número total de horas para impartir dicha esta asignatura se recomienda que sea de 60 horas. Las 60 horas reloj, deben ser cumplidas para terminar el modulo. Los días festivos no son contabilizados en la temporización

8.1 División de los laboratorios

La siguiente tabla muestra el resumen de las prácticas a realizar en el Laboratorio de Bases de Datos. Junto con el nombre de cada una de las prácticas aparece la semana en la que dicha práctica debería impartirse.

Semana	*No. Práctica*	*Contenido*
1	0	PRESENTACIÓN DEL LABORATORIO. INTRODUCCIÓN A BASES DE DATOS
2	0	SISTEMA DE GESTIÓN DE LA BASE DE DATOS.
3	1	REQUERIMIENTOS
4	2	REEDICIÓN DEL ENUNCIADO DEL PROBLEMA
5	3	DIAGRAMA E/R
6	4	ESQUEMA FINAL
7	5	DOCUMENTACIÓN ADICIONAL
8	6	PASO A MODELO RELACIONAL

9 DESARROLLO TEMÁTICO DE CADA CLASE TEÓRICA

En este apartado se desarrollará el contenido temático específico a impartir en cada una de las clases teóricas de la asignatura Bases de Datos. Para cada tema se incluirá además la bibliografía recomendada a los estudiantes, de forma que éstos puedan contrastar los contenidos y profundizar en aquellos aspectos que les resulte más interesantes.

9.1 Clase 1: Papel económico, social y cultural de la información

CONCEPTO DE SISTEMA DE INFORMACIÓN

DEFINICIÓN DE SISTEMA

CARACTERÍSTICAS DE UN SI, SEGÚN BUBENKO (1980)

OBJETIVOS:

- Definir el Concepto de Sistema de Información
- Definir el Concepto de Sistema
- Identificar las Características de un SI, según Bubenko

9.1.1 Papel económico, social y cultural de la información

Las necesidades de información de nuestra sociedad actual se dejan sentir de forma cada vez más imperiosa. La información, como soporte de la transferencia de conocimientos, es clave para el porvenir de la humanidad e indispensable para poder modelar bien este porvenir.

El problema de la información está estrechamente relacionado con el desarrollo económico y social. La investigación, la planificación y la toma de decisiones exigen una información precisa, oportuna y adaptada a las necesidades especificas de cada usuario y de cada circunstancia.

Si se analiza la evolución de los distintos sectores económicos en los últimos años, se puede comprobar la excepcional expansión que la información ha tenido en relación con otros sectores, llegándose a calificar esta expansión, y los profundos cambios a los que ha dado lugar, como segunda revolución industrial, marcando el comienzo de una nueva era en el desarrollo de la humanidad.

Nuestra infosfera, o esfera de la información, está siendo fuertemente alterada con el advenimiento de una tercera ola, la cual aparece como una tercera conmoción después de las revoluciones agrícolas e industriales, conmoción cuyos efectos han sido ampliamente analizados por distintos autores, como por ejemplo en el conodo libro de TOFFLER (1980) en donde señala: “La tercera ola no se limita acelerar nuestro flujo de información;

transforma la estructura profunda de la decisión de la que depende nuestras acciones diarias".

Nadie puede, en estos momentos, poner en tela de juicio la importancia de la información, importancia que provoca una fuerte demanda de este bien, siendo preciso analizar, además de los condicionamientos tecnológicos, el marco legal e institucional en el que se inscribe el derecho a la información esencial de nuestra época.

Son muchos los factores que han influido en la transformación que se ha operado en el papel que desempeña la información en los contextos económicos y social. Entre ellos es preciso destacar la elevación del nivel cultural; el afán de desmasificación, que eleva a una mayor diversidad, con el consiguiente crecimiento de las necesidades de información; el deseo de participar en las decisiones públicas; las exigencias de planificación y ordenación del territorio; las tendencias hacia una descentralización, que requiere datos cada vez más detallados para áreas más pequeñas; la aparición de nuevos métodos –más científicos – de toma de decisiones; etc.

La información es considerada, junto con la materia y con la energía, uno de los componentes fundamentales de la naturaleza, siendo vital para el desarrollo de los pueblos, al destacar la enorme importancia de la información.

Revisando de forma superficial el concepto de información, y que ha llevado a tres enfoques distintos, aunque no incompatible, ya que cada uno engloba los anteriores al ir dotando a este término de una mayor generalidad.

Una primera acepción está relacionada con la investigación, y considera la información como instrumento esencial y como elemento de entrada/salida en toda actividad del campo de la investigación, sea ésta fundamental o aplicada. Otra acepción, no tan limitada como la anterior, identifica información como conocimiento transmisible, lo que lleva a abrir el circulo de beneficiarios de la información, que ya no se circunscribe solamente a los científicos y a los técnicos, sino que se amplía a otro conjunto de actividades socioculturales como la medicina, la educación, los medios de comunicación de masas, etc. Por último, con una visión mucho más general, se puede considerar que la información, al igual que la materia y la energía, constituye un recurso fundamental, un bien en el sentido económico del término, que ha de ser utilizada en cualquier actividad humana. Se convierte así la información en elemento esencial para la producción. La igualdad de oportunidades exige no limitar arbitrariamente el acceso a este recurso que ha de ponerse al servicio de toda la sociedad.

Es bajo este amplio acepción de la información como recurso, cada vez más generalmente admitida, cuando cobra verdadero significado la exigencia de un derecho a la información por parte de las personas físicas y jurídicas, y donde se ve el importante papel que desempeñan los sistemas de información como instrumentos que aportan este recurso a las organizaciones y a los individuos.

A medida que se toma conciencia del valor de la información, se manifiesta la necesidad, que se llega a ser imperiosa, de impulsar la libre circulación de la información, tanto en un plano nacional como internacional, respondiendo así al mandato de las normas que establecen la libertad de información como uno de los grandes derechos de la civilización actual. En el artículo 19 de la Declaración Universal de los Derechos del

Hombre se señala: “Todo individuo tiene derecho a la libertad de opinión y de expresión. Esto implica recibir y difundir, sin consideración de fronteras, la información y las ideas, por cualquier medio de expresión que sea ...”.

Los gobiernos de los distintos países, al reconocer que la información circula para responder a unas necesidades, como son la necesidad de saber, la necesidad de conocer y la necesidad de elegir, no han podido por menos que ocuparse del tema, impulsando y planificando sistemas nacionales de información afin de asegurar un flujo de este recurso, imprescindible para la evolución científica, económica y social.

Es preciso, sin embargo reconocer que nos encontramos aún muy lejos de poder satisfacer las necesidades de información, ya que no sólo existen problemas tecnológicos relativos al almacenamiento ya acceso a la información, sino que existen también problemas económicos políticos, administrativos, etc, que impiden el desarrollo de sistemas de información eficientes capaces de atender debidamente las demandas de información.

9.1.2 Cualidades de la información

La explosión de la información –como se llama a veces a este enorme crecimiento de las necesidades de la información y a la mayor disponibilidad de este recurso- puede conducir, si no se ponen los medios para evitarlo, a una polución informativa. Fenómeno análogo al de la contaminación del aire, en el que la información, al perder sus cualidades, no puede cumplir sus objetivos, llegando incluso a ser más nociva que beneficiosa para sus destinatarios. Para evitar el peligro de polución informativa se debe exigir a la información un conjunto de cualidades que mantenga su valor comunicativo, ya que para hacer honor a su nombre debe ser capaz de informar, es decir, de aportar un conocimiento.

Las cualidades que debe poseer la información, y que hacen de ella un recurso fundamental de las organizaciones y de los individuos, son básicamente: precisión oportunidad, compleción significado e integridad. Todas ellas en el grado de exija cada sistema concreto.

La precisión se puede definir como el porcentaje de información correcta sobre la información total del sistema (ficheros, base de datos, etc.). De todas formas, el usuario ha de tener presente que el tratamiento por ordenador no puede mejorar la calidad de los datos que son elaborados, lo único que puede hacer la máquina es señalar ciertos errores o incompatibilidades, e incluso sustituir el dato detectado como erróneo por otro que no tenga error aparente, es decir, que sea preciso, debemos también suministrarle datos precisos, no pudiendo pretender en los resultados una precisión superior a la que tenían los datos de entrada. Una precisión baja lleva a una falta de credibilidad del usuario hacia la información que se le proponga.

La oportunidad se refiere al tiempo transcurrido desde el momento en que se produjo el hecho que originó el dato hasta el momento en el que la información se pone a disposición del usuario. Otras veces la oportunidad se mide en función del tiempo transcurrido desde que el dato tendría que estar disponible, o bien respecto al desfase que produce el proceso por ordenador. Al igual que ocurre con la precisión, también la oportunidad depende de cada aplicación. Por ejemplo, para un censo en el cual se manejan millones de datos de carácter bastante estable, un tiempo de proceso de meses no le resta oportunidad a la información, en cambio esta demora en la obtención de los indicadores de

coyuntura, como el IPC (índice de precios al consumo), sería inadmisible. En general, el valor de la información va disminuyendo con el transcurso del tiempo, e incluso, después de cierto momento, puede llegar a perder totalmente la relevancia que pudiera tener; la pérdida de valor será más o menos rápida dependiendo del tipo de información.

Otra cualidad que ha de tener la información es la compleción, lo que significa que ha de ser completa para poder cumplir sus fines. Por ejemplo, un informe que se emite con el objeto de que un directivo tome una decisión, ha de contener todos los elementos informativos necesarios para apoyar dicha decisión. La compleción absoluta es imposible de conseguir, y lo que se suele pretender en los sistemas de información es alcanzar un nivel que se considere suficiente, el cual dependerá de dos factores: de los datos existentes en el sistema de información y de los que el sistema es capaz de localizar ante una consulta concreta. En este segundo factor influirá la flexibilidad e idoneidad del lenguaje de recuperación y el acierto en la formulación de la consulta. Así pues, la compleción no es sólo función de la información en sí misma, sino también de otros factores, tanto técnicos como humanos.

La información que se suministra al usuario debe ser también significativa; es decir, ha de poseer el máximo contenido semántico posible, ya que sin él no constituiría verdadera información. Esto lleva a que ha de ser comprensible e interesante, lo que supone proporcionar a los usuarios grandes masas de información que por su volumen no puedan ser asimiladas. Un volumen de información justo es condición indispensable para que está sea significativa. Cuando se realiza el diseño ha de ser, además de fácilmente interpretación, sólo la necesaria y suficiente para que se cumpla los fines propuestos.

Asimismo, todo la información contenida en el sistema debe ser coherente en sí misma además de consistente con las reglas semánticas propias del mundo real al que ha de representar lo más fielmente posible; esta cualidad, que en las bases de datos se conoce a veces con el nombre de integridad, coincidente en parte con el concepto que hemos definido como precisión.

Es preciso también atender la seguridad de la información, ya que está ha de ser protegida tanto a su deterioro –por causas físicas o lógicas- como frente a acceso no autorizados. La seguridad de la información esta adquiriendo una gran relevancia, muy especialmente con la difusión de las nuevas posibilidades de comunicación y la enorme extensión de redes de conexión como Internet e Intranet. Actualmente el concepto de seguridad comprende confidencialidad, disponibilidad e integridad.

Cuando se están haciendo los estudios que nos llevarán a la implantación de un sistema de información, es preciso tener muy en cuenta todos estos requisitos de la información buscando el punto de equilibrio que permita alcanzar los objetivos del sistema a un costo aceptable, ya que cuantas más cualidades reúna la información más se incrementará su coste de obtención y tratamiento.

Por otra lado, unas cualidades pueden resultar incompatibles con otras; así, pretender una gran precisión lleva consigo generalmente una pérdida de oportunidad. Por ello, insistimos, es necesario llegar a una solución de compresión, encontrando el punto de equilibrio entre las diversas cualidades de la información, dentro de nos objetivos concretos de cada sistema y siempre a unos costes aceptables.

9.1.3 Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición. Capítulo 1: Sistemas de Información y Bases de Datos, Pags. 3

9.2 Clase 2: Introducción a base de datos

CONCEPTO DE SISTEMA DE INFORMACIÓN

DEFINICIÓN DE SISTEMA

CARACTERÍSTICAS DE UN SI, SEGÚN BUBENKO (1980)

OBJETIVOS:

- Definir el Concepto de Sistema de Información
- Definir el Concepto de Sistema
- Identificar las Características de un SI, según Bubenko

9.2.1 Concepto de sistema de información

Toda organización necesita para su funcionamiento un conjunto de informaciones que han de transmitir entre sus distintos elementos y, generalmente también, desde y hacia el exterior del sistema. Una parte de esta comunicación se realiza por medio de contactos interpersonales entre los empleados, es el sistema de información informal. Por este tipo de flujo de información, cuando se trata de organismos complejos, se muestra insuficiente y costoso, siendo preciso disponer de un sistema de información formal, también llamado organizacional que, integrado en el sistema de orden superior que es el organismo, aporte a éste la información necesaria de forma eficaz y eficiente.

Antes de estudiar el concepto de sistema de información analizaremos las características de los sistemas en general.

9.2.2 Definición de sistema

El concepto de sistema, término del cual puede que se haya abusado en los últimos tiempos, se aplica a los fenómenos más diversos y a veces sin demasiado rigor. Se trata de una noción difícil de precisar, posiblemente debido a la dosis de relatividad que este concepto lleva consigo.

El diccionario de la Real Academia Española (DRAE) define, en una segunda acepción, el vocablo sistema como " Conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a determinado objeto".

Podemos admitir como válida para nuestros fines esta definición de la Real Academia, con las siguientes precisiones:

El término "cosa", a diferencia de la acepción restrictiva –contrapuesta a algo viviente como persona y animal- que ha llegado a imponerse en el lenguaje común, es también, en la primera acepción del DRAE, todo lo que tiene entidad, ya sea corporal o espiritual, natural o artificial, real o abstracta. Por tanto, el sistema puede estar constituido por objetos físicos, actividades, formas de energía, seres vivientes, entes inanimados, conceptos, ideas, símbolos matemáticos, etc., sin que exija que todos pertenezcan a una misma clase. Llamaremos, desde ahora elementos a las cosas que integran el sistema.

Los elementos tiene que estar relacionadas entre sí con un orden, determinado por unas reglas que gocen de cierta estabilidad. Esta idea de relación ajustada a una normativa es fundamental.

El sistema típico es finalista; es decir, los elementos están relacionados para contribuir a determinado objetivo. Así, los sistemas debidos a los hombres, como los sistemas de información, son creados para el cumplimiento de unos fines; razón por la que algunos autores se refieren a la dinamicidad de los sistemas, considerando que éstos se mueven, en sentido real o figurado, hacia la consecución de un objetivo global.

La noción de sistema relativa, ya que, a excepción del universo, en la parte más superior de la jerarquía, cualquier sistema es siempre un subsistema de otro sistema más amplio que lo engloba. Para expresar este concepto de relatividad de los sistema KOESTLER (1979) introduce el término holón, con el que designa entidades de nivel intermedio que "están subordinados como partes a centros más altos en la jerarquía, pero al mismo tiempo funcionan como totales casi autónomos". Así, por ejemplo, la base de datos se puede considerar como un subsistema del sistema de información y éste, a su vez, es un subsitema de la organización.

En el enfoque sistémico, el todo, es decir, el sistema, es más que la simple suma o agregación de las partes componentes; porque, en general, su objetivo es distinto y presenta nuevas propiedades o características que no son explicables a partir de las características de sus elementos considerados de forma aislada.

Dentro de este concepto de sistema de información, como su nombre indica, será un sistema, y asimismo la base de datos es también un sistema.

Los sistema están natural o artificialmente limitados, llamándose a todo lo que está situado fuera de sus limites el medio ambiente o entorno del sistema. De este entorno toma el sistema los elementos o materias primas que constituyen las entradas, y al entorno se vierte los productos elaborados, que son las salidas.

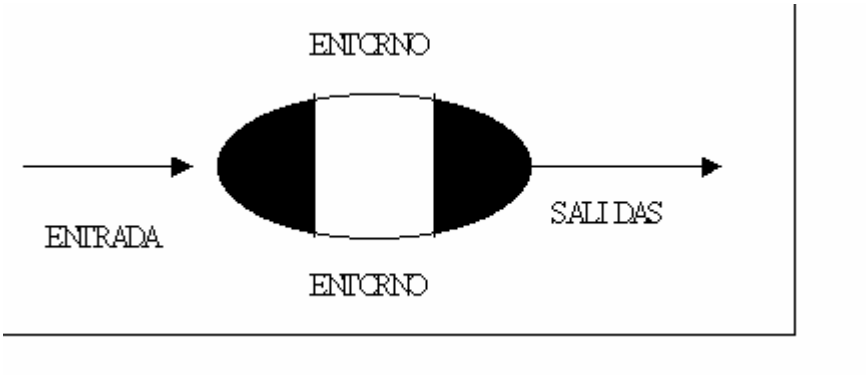


Figura 1.1 **límites del medio ambiente o entorno del sistema**

Los sistemas de pueden dividir en dos grandes grupos: los naturales y los artificiales. Entre estos últimos, que son debidos al hombre, se encuentran los sistema de información.

9.2.3 Concepto de sistema de información

Todo sistema de información formal, de ahora en adelante lo llamaremos simplemente sistema de información (SI), se diseña a fin de satisfacer las necesidades de información de una organización (empresa o cualquier tipo de institución pública o privada) y está inmerso en ella. El SI ha de tomar los datos del entorno (la propia organización así como fuentes externas) y sus resultados han de ser la información que dicha organización necesita para su gestión y toma de decisiones; por otra parte, los directivos de la organización tendrán que marcar los objetivos y directrices por lo que se regule el SI. Llamamos Sistema Objeto a la parte de la organización de la cual se nutre el SI y a la cual revierte sus resultados.

Sistema dinámico será aquel que controla su actuación en función de cómo las salidas cumplen los objetivos marcados; de esta forma el sistema se va adecuando dinámicamente a unas condiciones de entorno que, en el caso más general, son variables en el tiempo. El control del sistema puede realizarse por medio de mecanismos internos (sistema autorregulados), por mecanismo situados en el entorno, o por ambos; aunque esta distinción tiene un alto grado de subjetividad, ya que siempre se podrán ampliar los límites del sistema haciendo que los elementos que llevan a cabo la función reguladora estén comprendidos en el mismo.

Los sistemas dinámicos, que están en interacción con el entorno, de forma que las entradas y el proceso se van adaptando constantemente para obtener determinadas salidas, se pueden representar de acuerdo con el diagrama.

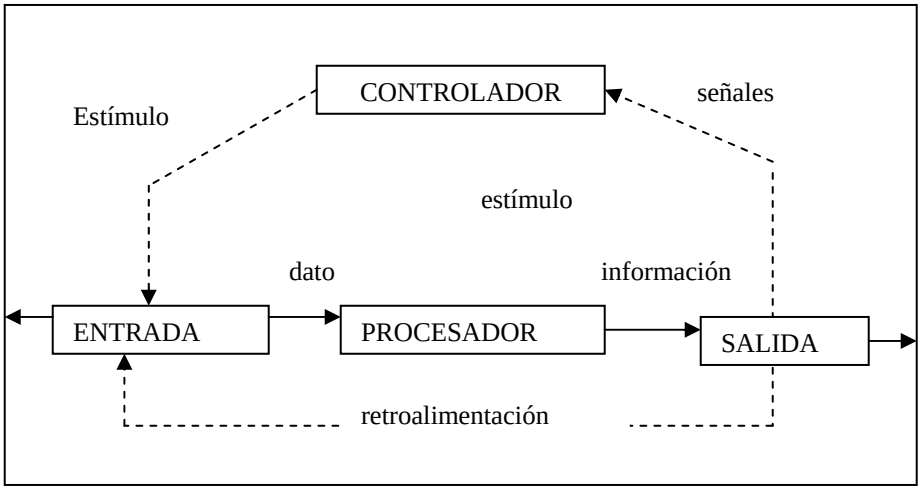


Figura 1.2 **Sistemas dinámicos, que están en interacción con el entorno**

El controlador del sistema, que ejerce funciones de planificación y de gobierno, actúa de acuerdo con la información que recoge de la salida, enviando estímulos a la unidad de entrada y al procesador, a fin de conseguir que las salidas respondan a los objetivos del sistema. Para ello, el controlador ha de ser capaz de recibir la información, interpretarla, compararla con los objetivos previstos y emitir los impulsos de control que exijan la regulación del sistema.

Las entradas del sistema son los elementos que se consumen o transforman en el proceso. Se corresponden con la materia prima en los procesos de fabricación; en el caso de los sistema de información, serán los datos. Los SI se diferencian de otros sistemas porque en ellos las entradas no se consumen, sólo se transforma sin destruirse, ya que quedan almacenadas en la base de datos del propio sistema.

Las salidas son los elementos que se crean en el proceso. Constituyen el producto terminado de los procesos de fabricación; en este caso la salida es la información.

El procesador es el lugar donde se efectúa el tratamiento y comprende todos los elementos que participan en él sin transformarse ni crearse; es decir, a excepción de las entadas y las salidas. El procesador suele ser, a su vez, un elemento sistémico situado en un orden más bajo de la jerarquía. Sus componentes son muchas veces nuevos elementos sistémicos de órdenes más bajos.

En muchos sistemas también existen realimentación, que va de la salida a la entrada sin pasar por el controlador.

En los SI (sistemas eminentemente dinámicos) existirá un control externo al propio SI, que son los órganos directivos de la organización que establecen el marco en el que el SI se desenvuelve; pero al mismo tiempo el SI tendrá que disponer en su interior de mecanismos autorreguladores más o menos desarrollados que interpreten y detallen las órdenes de los órganos directivos, e incluso las leyes y normas emanadas de órganos situados a niveles superiores, transmitiéndoselas a las unidades del SI que han de ser objeto de regulación. Podríamos decir que en los SI suele existir un control a dos o más niveles: el control externo, ejercido por los órganos directivos, y una autorregulación de tipo interno. La mayor o menor autonomía del SI estará en función del predominio del control interno sobre el externo.

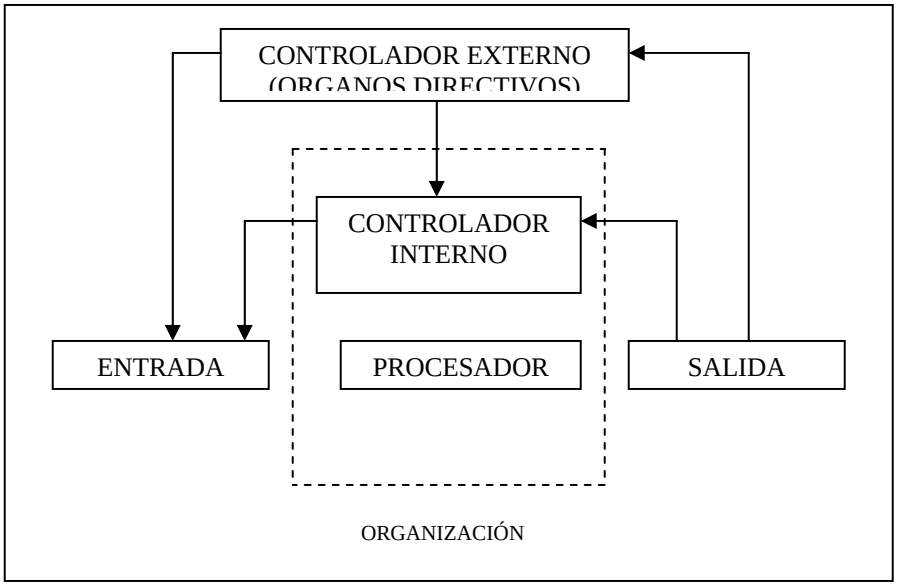


Figura 1.3 **Sistemas eminentemente dinámicos**

Al igual que en el caso de la definición de sistema, son también muy numerosas las existentes para SI. Así LANGESFORS (1977) da una definición tan breve como sencilla de este concepto: "Sistema de información son sistemas que suministran servicios de información" TEICHROEW (1976) dice: "Un sistema de información puede ser definido como una colección de personas, procedimientos y equipos diseñados, construidos, operados y mantenidos para recoger, registrar, procesar, almacenar recuperar y visualizar información".

Definiendo el sistema de información como un "conjunto de elementos, ordenadamente relacionados entre sí de acuerdo con unas ciertas reglas, que aporta al sistema objeto (es decir, a la organización a la cual sirve y que le marca las directrices de funcionamiento) la información necesaria para el cumplimiento de sus fines, para lo cual tendrá que recoger, procesar y almacenar datos, procedentes tanto de la misma organización como de fuentes externas, facilitando la recuperación, elaboración y presentación de los mismos". Uno de los instrumentos fundamentales para facilitar al SI el cumplimiento de estas funciones de recuperación, elaboración y presentación de la información es la base de datos.

9.2.4 Las Características de un SI, según Bubenko (1980)

Pueden agruparse en:

a) Tecnológicas que afectan al rendimiento y seguridad del sistema, desde el punto de vista del equipo.

b) Funcionales y semánticas, que se refiere a si el sistema hace lo que debe de una forma correcta (eficacia) y si es capaz de adaptarse a requisitos cambiantes

c) Económicas, que ponen el énfasis en el coste del sistema y en la eficacia con que responde a los objetivos.

d) Sociales, que son las que tienen un impacto sobre el entorno social (interno o externo) en que se desenvuelve el sistema.

El SI puede ser comparado con un motor que impulsa la información, haciéndola circular por el organismo, distribuyéndola y aportándola a aquellas áreas donde es necesaria. Para realizar esta función es preciso que el sistema recoja previamente los datos allí donde son generados y los procese para convertirlos en información útil.

Entre el SI y el organismo donde está inserto existe una mutua y estrecha interrelación; en realidad, el SI no es otra cosa que un subsistema de los varios que integran la organización. Es imprescindible tener esto muy presente, ya que si no existe la debida interacción y se produce un desfase entre ambos, el SI no podrá cumplir los objetivos para los que fue diseñado. La falta de adaptación entre el SI y el organismo es causa del fallo de muchos sistemas que prometían ser eficiente ente el SI y el organismo es causa del fallo de muchos sistemas que prometían ser eficaces, estando demostrado que las causas de estos fracasos se encuentran más frecuentemente en los aspectos sociales y humanos que en el diseño tecnológico.

Aun cuando los SI podrían no estar informatizados, siendo tratados manualmente, los SI se apoyan en técnicas informáticas, y los tratamientos y recuperación de la información se realiza, a menudo, por medio de sistema de gestión de bases de datos.

9.2.5 Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición. Capítulo 1: Sistema de Información y Bases de Datos Pag. 8 -14

9.3 Clase 3: introducción a las bases de datos

COMPONENTES DE UN SISTEMA DE INFORMACIÓN

SISTEMA DE INFORMACIÓN PARA LA GESTIÓN Y SISTEMAS DE INFORMACIÓN PARA AYUDA A LA DECISIÓN

NIVELES DE GESTIÓN

TIPOLOGÍA DE LOS PROCESOS DE GESTIÓN

OBJETIVOS:

Distinguir sistema de información para la gestión y sistemas de información para ayuda a la decisión

Mencionar los diferentes Niveles de Gestión

Distinguir las diferentes Tipología de los Procesos de Gestión

9.3.1 1.7 Componentes de un sistema de información

Un sistema de información está constituido por una serie de componentes que se resumen:

Figura 1.4 **Componentes de un sistema de información**

El contenido del SI es el conjunto de datos (ficheros o base de datos), con su correspondiente descripción, almacenados en un soporte de ordenador. Los datos habrán de adecuarse a los objetivos que se pretende alcanzar con el sistema de información, por lo

que, en general, al ser estos objetivos variados, también los datos contenidos en el sistema serán de distintos tipos.

Es importante distinguir entre dos tipos de datos: referenciales y factuales. Los sistema de información referencial contienen referencias bibliográficas de los documentos donde se puede encontrar la información, pero no la información en sí misma, de modo que una vez recuperado el dato (es decir, la referencia) es preciso conseguir el documento fuente. En cambio, los sistema de tipo factual devuelve la información buscada, la cual puede ser directamente utilizada sin necesidad de acudir a nuevos circuitos informativos.

Otra clasificación, más bien aplicable a los datos factuales, se refiere a su formato, según la cual los datos pueden ser estructurados o no estructurados (también llamados formateados y no formateados). Los primeros tienen una cierta estructura o formato en la que los distintos campos ocupan determinadas posiciones fijas (así, en un fichero de personal, el DNI del empleado se puede encontrar en primer lugar, ocupando las ochos primeras posiciones del fichero; el nombre y los apellidos a continuación, etc.).Existen sin embargo, otros datos cuyo formato no puede ser fijo, como los textos (propios de los sistemas documentales), o los datos multimedia (voz, imagen, etc.); son datos no estructurados. Se suele distinguir entre dos tipos de sistemas de gestión según se ocupen de tratamiento de datos estructurados o no estructurados.

Sistema de Gestión de Bases de Datos (SGBD): se ocupan del tratamiento (definición, actualización y recuperación) de datos estructurados.

Sistema de Recuperación de Información (SRI): se ocupan del tratamiento de datos no estructurados (documentos). Estos sistemas proporcionan facilidades de thesaurus, (búsqueda en texto libre), etc.

En la actualidad existe un clara tendencia hacia la convergencia de estos dos tipos de tratamiento, de modo que ambas funcionalidades sean ofrecidas por un único sistema.

El ordenador, que ha de soportar la función de tratamiento o proceso, está integrado por dos subsistemas: el equipo físico (hardware) y el soporte lógico (software).

El conjunto de programas, documentación, lenguaje, etc, es el software, el cual debe gestionar los datos mediante el Sistema de Gestión de Bases de Datos (SGBD), controlar las comunicaciones y dar respuesta a necesidades de tratamientos específicos (como gestión de personal, desestacionalización de series temporales, indicadores bibliométricos, estimación de un modelo econométrico, etc.), todo ello apoyándose en el sistema operativo.

Otro componente fundamental del sistema es el administrador, o más bien la unidad de administración (ya que se trata de una función y no de una persona), cuya misión es asegurar la calidad y permitir el uso correcto y permanente de los datos. El administrador no es el propietario de los datos, sino el gestor y custodio de los mismos, cuya responsabilidad se extiende tanto al contenido del sistema como al área informática, si bien estas dos funciones (administración del contenido y administración del SGBD) pueden estar encomendadas a unidades distintas de la organización

Por último, se considera como otro componente del sistema a los usuarios, es decir, a la persona o grupo de personas que han de acceder al SI. Estos usuarios pueden ser tantos informáticos (analista y programadores) como usuarios finales con pocos conocimientos de informática que necesitan consultar o actualizar los datos generalmente en modo

conversacional y mediante lenguaje muy sencillo o procedimientos preparados ex profeso. También pueden existir usuarios que no acceden directamente al sistema, pero que obtienen información del mismo.

9.3.2 Sistemas de información para la gestión y sistemas de información para la ayuda a la decisión

La aplicación de los ordenadores en las empresas e instituciones comenzó con el tratamiento administrativo de sus datos operacionales; es decir, los que son necesarios para llevar a cabo las tareas de rutina (nomina, contabilidad, etc.). sin embargo, la potencia de estas máquinas no podía permitir que se las confinase en este campo, excesivamente limitado para sus posibilidades reales, y el ordenador empezó a intervenir en otros niveles de la empresa, ayudando a la sistematización de las funciones de dirección y constituyen un elemento activo en el proceso de toma de decisiones. Surgen así sistemas de información basados en el ordenador, que tienen como principal objetivo mejorar el proceso de información de la empresa logrando su máxima eficacia.

En toda organización se suele distinguir tres niveles de gestión (operacional, táctico y estratégico), por lo que el SI estará compuesto por tres subsistemas estructurados jerárquicamente y que corresponden con cada uno de estos tres niveles. En el plano operacional, los usuarios necesitan datos puntuales (elementales) que describan los sucesos que, de una forma u otra, caracterizan las actividades de la organización, por lo que este subsistema de información será muy voluminoso. De él, mediante un proceso de elaboración adecuado (en general de agregación), se podrán obtener los datos necesarios (junto con los aportados desde el exterior) para el funcionamiento de los otros dos subsistemas, cuyos usuarios tiene unas exigencias muy distintas, y para los que tal volumen de información no solamente sería inadecuado, sino peor aún, inoperante y contraproducente.

9.3.3 Niveles de gestión

Se encuentran representados, donde se puede observar que, mientras la información se transmite en sentido ascendente, las órdenes y planes se mueven en sentido descendente.

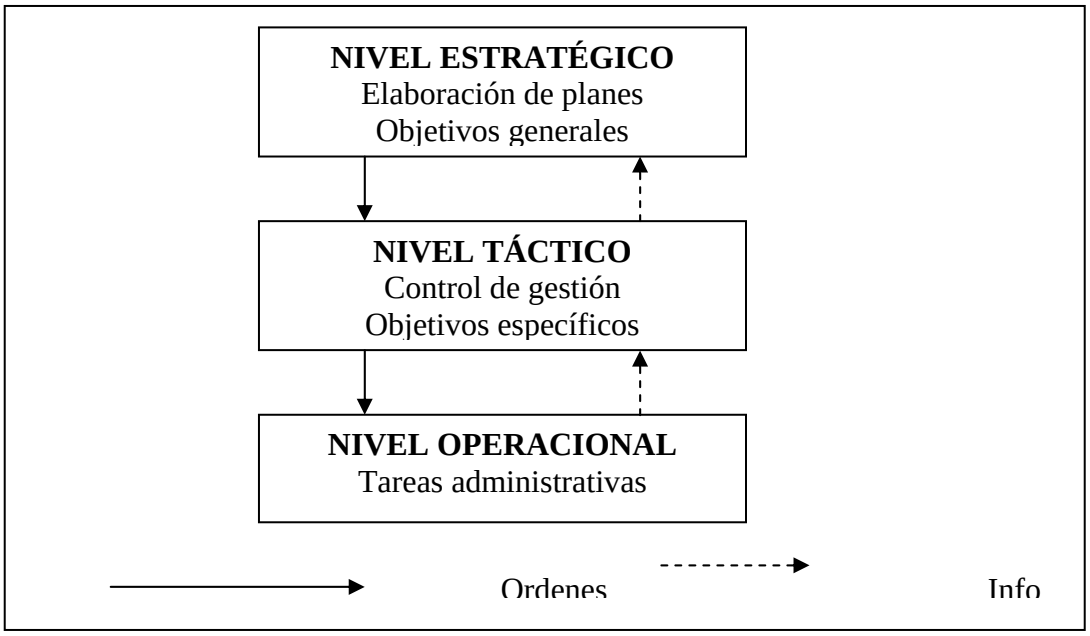


Figura 1.5 **Transmisión de la información**

En la figura 1.5 se muestran las características de la información que se necesitan en los distintos tipos de procesos que tienen lugar en las organizaciones. Se trata fundamentalmente de dos clases de información, una a nivel totalmente desagregado (microdatos), necesarios para los procesos que se suelen denominar administrativos, como son las tareas diarias y de rutinas que corresponden al plano operacional, y otra de ayuda a la decisión (tanto a nivel táctico como estratégico), que exige presentaciones muy diferentes, en la que muchos datos han de estar agregado (macrodatos) y cuya elaboración es bastante más compleja.

9.3.4 Tipología de los procesos de gestión

TIPOS DE PROCESOS	REPETITIVOS	EXCEPCIONALES
FORMALIZABLES	TAREAS ADMINISTRATIVAS Características: -datos voluminosos -propios, elementales y homogéneos pocas interrelaciones y simples muchas salidas normalizadas procesos sencillos y periódicos predomina el tratamiento secuencial y por lotes (ejemplo: nómina, facturas)	TAREAS ADMINISTRATIVAS DE EXCEPCIÓN Características: -datos no muy voluminosos -propios, elementales, así como agregados y homogéneos -muchas interrelaciones -pocas salidas normalizadas -procesos complejos pero estructurados -tratamiento no secuencial y, en general interactivo (ejemplo: estadísticas, modelos, gestión de personal, etc.)
NO FORMALIZABLES		AYUDA A LA DECISIÓN Características: -datos muy pocos voluminosos -propios y ajenos -agregados y muy heterogéneos -muchas interrelaciones complejas -pocas salidas, con informaciones significativa, oportunas y fácil de interpretar -procesos difíciles o imposible estructuración tratamiento no secuencial e interactivo (ejemplo: creación de una nueva unidad de producción)

Tabla 1.1 Tipología de los procesos de gestión

En un principio se atendieron las necesidades de información propias del nivel administrativo, desarrollando aplicaciones distintas y especificas que facilitaran cada una de las tareas de rutina. La información para la ayuda a la decisión, en esta primera etapa, se solía elaborar manualmente o, a veces, por programas diseñados ad hoc para resolver necesidades concretas y puntuales. Posteriormente, y ante los graves problemas a que daba lugar este planteamiento, se vio la necesidad de buscar nuevas soluciones, surgieron la idea de utilizar una base de datos común que incorporara, sin redundancia indeseable, la información necesaria para las distintas funciones. Con este enfoque se trata de disponer de un SI integrado capaz de dar respuesta tanto a las necesidades de gestión como de decisión (véase figura 1.7).

Posteriormente hemos asistido a la difusión de sistemas diseñados para servir de soporte a la toma de decisiones dirigidas a los directivos (conocidos con la siglas inglesas D.S.S., Decision Support System o EIS Executive Information Systems), uno de cuyos componentes principales es, precisamente, una base de datos. En estos momentos la extracción de información por medio de la búsqueda (o minería) de datos soportada en un almacén de datos ha venido a extender y a hacer más eficaces los anteriores sistemas.

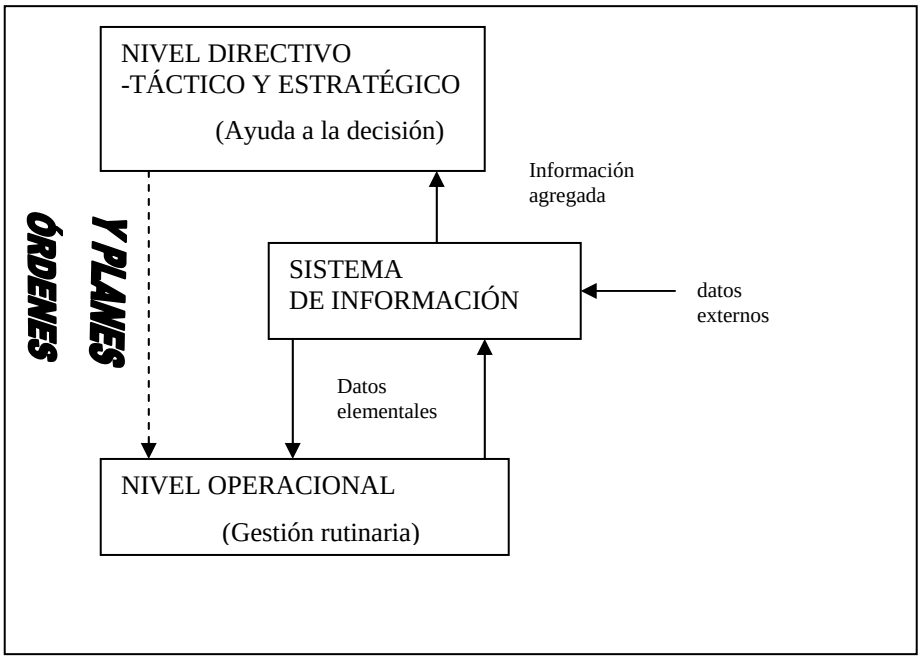


Figura 1.6 **Sistema de información único (nivel directivo y operacional)**

9.3.5 Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición. Capítulo 1: Sistema de Información y Bases de Datos Pag. 14-19

9.4 Clase 4: introducción a las Base de Datos

DE LOS SISTEMAS TRADICIONALES DE FICHEROS A LAS BASE DE DATOS

VENTAJAS E INCONVENIENTES DE LAS BASES DE DATOS FRENTE A LOS FICHEROS CLÁSICOS

VENTAJAS DE LAS BASES DE DATOS

OBJETIVOS:

- Mencionar la diferencia de los sistemas tradicionales de ficheros a las base de datos
- Identificar ventajas e inconvenientes de las bases de datos frente a los ficheros clásicos.
- Identificar las ventajas de las Bases de Datos

9.4.1 De los sistemas tradicionales de ficheros a las base de datos

Si analizamos la situación (especialmente la situación pasada) de algunos sistemas de información, podemos encontrar sistemas en los que existe una proliferación de ficheros, específicos cada uno de ellos de una determinada aplicación (véase figura 1.8). Los datos se recogen varias veces y se encuentra repetidos en los distintos archivos. Esta redundancia, además de malgastar recursos, origina a menudo divergencia en los resultados.

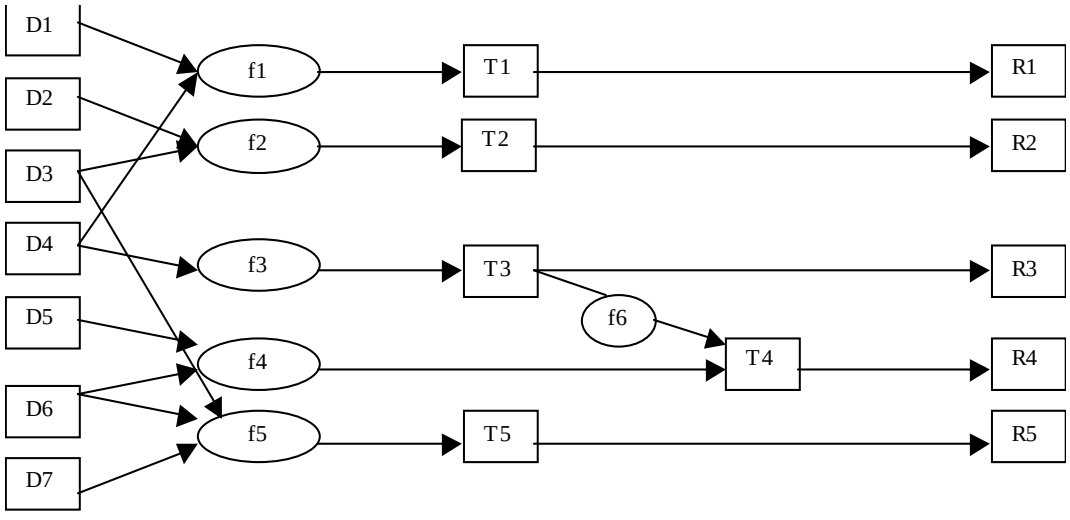


Figura 1.7 **Organización clásica: Sistema orientado al proceso**

Los sistemas informáticos de este tipo han sido llamados a veces sistemas orientados hacia el proceso, debido a que en ellos se pone el énfasis en los tratamientos que reciben los datos, los cuales se almacenan en ficheros diseñados para una determinada aplicación. Las aplicaciones se analizan e implantan con entera independencia unas de otras, y los datos no se suelen transferir entre ellas, sino que se duplican siempre que los correspondiente trabajos los necesitan.

Este planteamiento produce, además de una ocupación inútil de memoria secundaria, un aumento de los tiempos de proceso, al repetirse los mismos controles y operaciones en los distintos ficheros. Pero más graves todavía son las inconsistencias que a menudo se presentan en estos sistemas, debidos a que la actualización de los mismos datos, se encuentran en más de un fichero, no se suele realizar de forma simultánea en todos ellos.

Por otra parte, la dependencia de los datos respecto al soporte físico y a los programas da lugar a una falta de flexibilidad y de adaptabilidad frente a los cambios que repercute muy negativamente en el rendimiento de conjunto del sistema informático.

Los problemas son aún más acusados cuando se presentan demandas inesperadas de información o cuando los directivos pretenden tener un verdadero sistema de información orientado a la toma de decisiones, lo que es inalcanzable con estas aplicaciones diseñadas de forma estanca, que las hace totalmente inoperante fuera del contexto para el que fueron concebidas.

De este análisis se deduce claramente la necesidad de una gestión más racional del conjunto de datos, surgieron así un nuevo enfoque que se apoya sobre una base de datos, en la cual los datos son recogidos y almacenados una sola vez, con independencia de los tratamientos (véase figura 1.8)

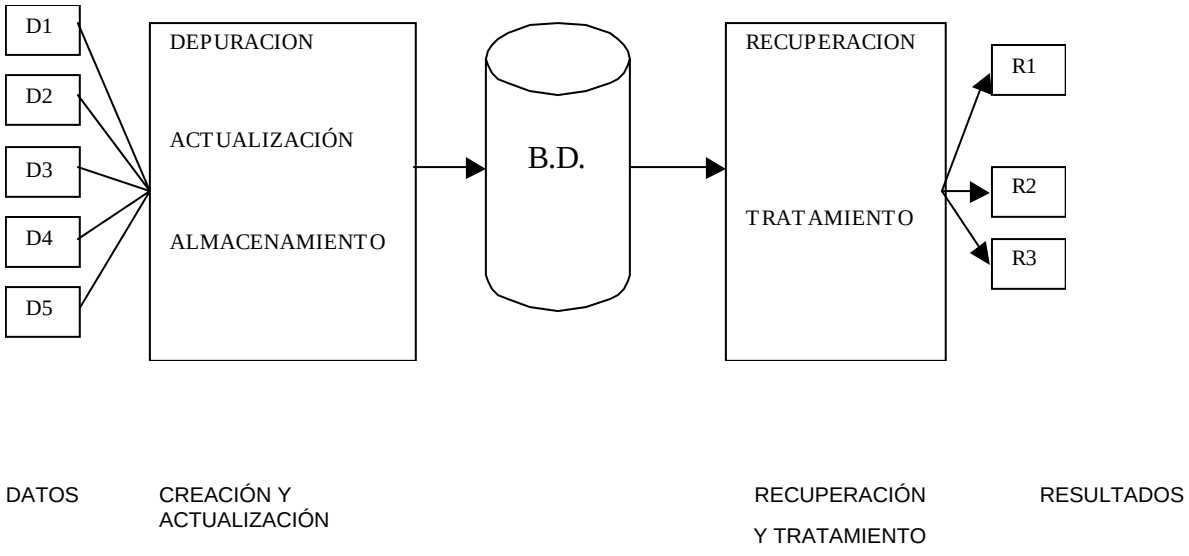


Figura 1.9 **Organización en B.D.:Sistema orientados a los datos**

Vemos por tanto, que la solución de los problemas asociados al tratamiento de los datos en los sistemas tradicionales lleva a un cambio radical en el enfoque del sistema de información, en el cual los datos se organizan y se mantienen en un conjunto estructurado que no está diseñado para una aplicación concreta, sino que, por el contrario, tiene a satisfacer las necesidades de información de toda la organización; necesidades cuya diversidad se ve acentuada en el transcurso del tiempo.

Estos sistema orientados hacia los datos van sustituyendo a los sistemas orientados hacia el proceso que, por su poca fiabilidad, falta de adecuación a la realidad y mal asegurada confidencialidad, han ido perdiendo de forma progresiva la confianza de los usuarios.

9.4.2 Ventajas e inconvenientes de las bases de datos frente a los ficheros clásicos

Las base de datos, surgieron como respuesta al nuevo planteamiento de los sistemas orientados hacia los datos para mejorar la calidad de las prestaciones de los sistemas informáticos y aumentar su rendimiento, presentan una multitud de ventajas frente a los sistemas clásicos de ficheros.

Debemos, sin embargo, señalar, para que nadie se sienta defraudado por el fracaso de algún proyecto de este tipo, que las bases de datos no son la panacea universal que soluciona todos los problemas que la información plantea a las empresas. Las bases de datos son un instrumento, que supone un distinto enfoque en la gestión de los datos, y su éxito o su fracaso estará condicionada por el uso que de ellas sepamos hacer, no sólo los técnicos, sino también los directivos.

9.4.3 Las ventajas de los sistemas de bases de datos

Independencia de los datos

La mutua independencia de datos y tratamientos lleva a que un cambio de estos últimos no imponga un nuevo diseño de la base de datos. Por otra parte, la inclusión de nuevas informaciones, desaparición de otras, cambios en la estructura física o en los cambios de acceso, etc., no deben obligar a alterar los programas. Esta independencia de los tratamientos frente a la estructura de la base de datos, supone una considerable ventaja, al evitar el importante esfuerzo que origina la reprogramación de las aplicaciones cuando se producen cambios en los datos.

La flexibilidad que proporciona la independencia de los datos y programas es muy importante para conseguir sin excesivo coses la continua adaptación del sistema de información a la evolución de las organizaciones. Aunque en la práctica dicha independencia nunca es absoluta, los sistemas de gestión de bases de datos actuales van respondiendo cada vez mejor a las exigencias de los usuarios relativas a las independencia y flexibilidad de los datos.

Coherencia de los resultados

Debido a que la información de la bases de datos se recoge y almacena una sola vez, en los tratamientos se utilizan los mismos datos, por lo que los resultados de todos ellos son coherente y perfectamente comparables. Además, al no existir (o al menos disminuir en gran medida) la redundancia en los datos, desaparece el problema que se presentaba en el enfoque clásico de que el cambio de un dato obligaba a actualizar una serie de ficheros. De

esta forma se elimina también el inconveniente de las divergencias en los resultados debidas a actualizaciones no simultáneas en todos los ficheros.

Mejor disponibilidad de los datos para el conjunto de los usuarios

Cuando se aplica la metodología de base de datos, cada usuario ya que no es propietario de los datos, puesto que estos se comparten entre el conjunto de aplicaciones, existiendo una mejor disponibilidad de los datos para todos los que tienen necesidad de ellos, siempre que estén autorizados para su acceso. Hay también una mayor transparencia respecto a la información existente, ya que todos los datos que se encuentran en la base se deben relacionar en un catálogo o diccionario que puede ser ampliamente difundido y accedido por medios informáticos

Mayor valor informativo

Puesto que la base de datos ha de ser reflejo del mundo real, en ella se recogen las interrelaciones entre los datos, por lo que el valor informativo del conjunto es superior a la suma del valor informativo de los elementos individuales que lo constituyen; es decir, actúa el efecto de sinergia.

Mejor y más normalizada la documentación de la información.

Mejor y más normalizada la documentación de la información la cual está integrada con los datos.

En el enfoque clásico los datos se encuentran separados de su contenido semántico; los primeros se almacenan en ficheros y su descripción se hace mediante un lenguaje de programación que se encuentra en los programas. La documentación de los datos, realizada por el analista o programador, es en general insuficiente, y a veces incluso inexistente. Además, por lo común, la estandarización brilla por su ausencia. Este problema se atenúa en gran medida en las bases de datos, ya que en la misma base se incluyen no sólo los datos, sino también la semántica de los mismos.

Mayor eficiencia en la recogida, validación e introducción de los datos en el sistema

Al no existir apenas redundancia, los datos se recogen y validan una sola vez, aumentando así el rendimiento de todo el proceso previo al almacenamiento.

Reducción del espacio de almacenamiento

La desaparición (o disminución) de las redundancia, así como la aplicación de técnicas de compactación, lleva en los sistemas de bases de datos a una menor ocupación de almacenamiento secundario –disco magnético-. Se ha de tener presente, sin embargo, que los elementos del sistema (diccionario, referencias, punteros, ficheros invertidos, etc.) ocupan bastante espacio.

En la tabla 1.2 se presenta un cuadro que resumen las ventajas de las bases de datos.

9.4.4 Ventajas de las bases de datos

REFERIDAS A:
LOS DATOS
-Independencia de estos respectos de los tratamientos y viceversa
-Mejor disponibilidad de los mismos
-Mayor eficiencia en la recogida, codificación y entrada en el sistema
LOS RESULTADOS
-Mayor coherencia
-Mayor valor informativo
-Mejor y mas normalizada documentación
LOS USUARIOS
-Acceso más rápido y sencillo de los usuarios finales
-Más facilidades para compartir los datos por el conjunto de los usuarios
-Mayor flexibilidad para atender a demandas cambiantes

Tabla 1.2 Resumen de las ventajas de las bases de datos

Como hemos señalado anteriormente, las bases de datos no sólo presentan ventajas, sino que también tienen posibles inconvenientes que es necesario valorar antes de tomar una decisión relativa a un cambio en la orientación del SI. Entre estos inconvenientes es preciso destacar:

9.4.5 Inconvenientes en el cambio en la orientación del SI

Los inconvenientes relacionados con la implantación de Sistemas Gestores de Bases de Datos en Sistemas de Información son los siguientes:

Instalación costosa

La implantación de un sistema de base de datos puede llevar consigo un coste elevado, tanto en equipo físico (nuevas instalaciones o ampliaciones), como en el lógico (sistema operativo, programas, compiladores, etc. necesarios para su uso), además del mismo coste de adquisición y mantenimiento del SGBD.

Personal especializado

Los conocimientos, que resultan imprescindible para una utilización correcta y eficaz y sobre todo para el diseño y administración de la bases de datos, implican una necesidad de personal especializado.

Implantación larga y difícil

Debido a las causas apuntadas anteriormente, la implantación de una base de datos puede convertirse en una tarea larga y laboriosa. Las dificultades que van apareciendo a lo largo de su desarrollo llevan en general a que se superen ampliamente los plazos inicialmente previstos.

Falta de rentabilidad a corto plazo

La implantación de un sistema de base de datos, tanto por su coste en personal y en equipo como por el tiempo que tarda en estar operativo, no resulta rentable a corto plazo, sino a medio o, incluso, a largo plazo.

Escasa estandarización

Un problema muy importante que se pone de manifiesto en el momento de la creación de una base de datos, es la falta de estandarización que facilite a los usuarios el manejo de los sistemas de bases de datos. Empieza, sin embargo, a observarse ya una preocupación por este tema, y van apareciendo estándares, tanto de iure como de facto, sobre todo en el campo de las bases de datos relacionales.

Desfase entre teoría y práctica

Al existir un considerable avance de la teoría en relación con la práctica en muchas ocasiones los usuarios, especialmente los directivos, se engañan respecto a las prestaciones reales que pueden proporcionarles los SGBD actualmente creyendo que constituyen ya una realidad ciertos aspectos que todavía son sólo teóricos.

Existe un evidente riesgo de frustración por parte de los directivos y usuarios de sistemas de bases de datos que pueden hacer olvidar las grandes posibilidades reales que estos sistemas nos ofrecen. Las bases de datos no constituyen únicamente una nueva tecnología, más o menos avanzada, pero tecnología al fin y al cabo, sino que nacen de una concepción distinta del SI, por lo que han de tener una influencia decisiva en las estructuras y organización de su entorno. Si esto no se tiene bien presente, muchas de las posibles ventajas de las bases de datos no se harán realidad y, en cambio se acentuará sus inconvenientes y problemas.

9.4.6 Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición. Capítulo 1: Sistema de Información y Bases de Datos Pag. 19 -25

9.5 Clase 5: Introducción a las bases de datos

CONCEPTO DE BASE DE DATOS

DISTINTOS NIVELES DE ABSTRACCIÓN DE UNA BASE DE DATOS

ESTRUCTURAS LÓGICAS DE USUARIOS: ESQUEMA EXTERNO

ESTRUCTURA LÓGICA GLOBAL: ESQUEMA CONCEPTUAL

ESTRUCTURA FÍSICA: ESQUEMA INTERNO

OBJETIVOS:

- Definir el Concepto de Base de Datos
- Distinguir los distintos niveles de abstracción de una base de datos
- Identificar las Estructura Lógica de Usuarios: Esquema Externo
- Identificar las Estructura Lógica Global: Esquema Conceptual
- Identificar la Estructura Física: Esquema Interno

9.5.1 Concepto de base de datos

Son muy numerosas las definiciones de bases de datos, y si se analizan detenidamente, se suele observar en casi todas ellas coincidencia en ciertos elementos; aunque también se detecta la falta de otros fundamentales, o al menos muy importantes, que son característicos, de las bases de datos y que marcan la diferencia entre este concepto y el de ficheros. En la tabla 1.3 se reproducen distintas definiciones de base de datos.

"Colección de datos interrelacionados almacenados en conjunto sin redundancia perjudicial o innecesaria; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinado para incluir nuevos datos y para modificar o extraer los datos almacenados." (Martin 1975)

"Colección o depósitos de datos, donde los datos están lógicamente relacionados entre si, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gran gama de usos y aplicaciones " (Conference des Statisticies Européens, 1977).

"Conjunto de datos de la empresa memorizado en un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos" (Flory, 1982)

"Conjunto estructurado de datos registrado sobre soporte accesible por ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y en

tiempo oportuno" (Delobel 1982)
"Colección no redundante de datos que son compartidos por diferentes sistemas de aplicaciones." (Howe, 1985)
"Conjunto de ficheros maestros, organizados y administrados de una manera flexible de modo que los ficheros puedan ser facilmente adaptados a nuevas tareas imprevisibles " (Frank, 1988)
"Colección de datos interrelacionados" (Elsmari y navathe, 1989).

Tabla 1.3 Diferentes definiciones de base de datos

La aparición de la expresión base de datos se produce a comienzo de los años sesenta. En 1963 tuvo lugar en Santa Mónica (EEUU) un simposio en cuyo título se encontraba la expresión Data Base. En una de sus sesiones, se propuso una definición de base de datos que, según las actas del simposio, no fue universalmente aceptada. Posteriormente, en 1967, el grupo de estandarización Codasyl decidió cambiar su primitiva denominación en la que no aparecía base de datos por el de Data Base Task Group. Poco a poco, el concepto y la expresión base de datos iba imponiéndose.

Vamos a empezar estudiando las características esenciales de una base de datos, que en algunos casos coinciden con las ventajas expuestas en el epígrafe anterior, para apoyar en ellas el concepto de base de datos.

En primer lugar, y esto coinciden todas las definiciones, una base de datos es un conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. Los datos están interrelacionado y estructurado de acuerdo con un modelo capaz de recoger el máximo contenido semántico. Dada la relevancia que tiene en el mundo real las interrelaciones entre los datos, es imprescindible que la base de datos sea capaz de almacenar estas interrelaciones. En el mundo real existen, además restricciones semánticas, a las que se está concediendo una importancia creciente y que, en los sistemas actuales, tienden a almacenarse junto con los datos, al igual que ocurre con las interrelaciones. La base de datos se describe y se manipula apoyándose en un modelo de datos.

La redundancia de los datos debe ser controlada, de forma que no existan duplicidades perjudiciales ni innecesarias, y que las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, sean tratadas por el mismo sistema, de modo que no puedan producirse inconsistencia. Esto podría resumirse diciendo que en las bases de datos no debe existir redundancia lógica, aunque sí se admite cierta redundancia física por motivo de eficiencia. Por tanto, un dato se actualizará lógicamente por el usuario de forma única, y el sistema se preocupará de cambiar físicamente todos aquellos campos en los que el dato estuviese repetido en caso de existir redundancia física; es lo que se denomina también redundancia controlada por el sistema.

Las bases de datos pretenden servir al conjunto de la organización, manejando los datos como otro recurso que viene a añadirse a los ya tradicionales. Por tanto, las bases de datos han de atender a múltiples usuarios y a diferentes aplicaciones, en contraposición a los sistemas de ficheros, en los que cada fichero está diseñado para responder a las necesidades de una determinada aplicación.

Otro aspecto importante de las bases de datos es la independencia, tanto física como lógica, entre datos y tratamientos. Esta independencia, objetivo fundamental de las bases de datos (a la que ya hemos hecho referencia al analizar las ventajas de las bases de datos, y en la cual seguiremos insistiendo), es una característica esencial esencial que distinguen las bases de datos de los ficheros y que ha tenido una enorme influencia en la arquitectura de los SGBD.

La definición o descripción del conjunto de datos contenidos en la base (lo que se denomina estructura o esquema de la base de datos) deben ser únicas y estar integradas con los mismos datos. En los sistemas basados en ficheros, los datos se encuentran almacenados en ficheros, mientras su descripción (muy somera) está separada de los mismos, formando parte de los programas, para lo cual se precisa que los lenguajes faciliten medios para la descripción de los datos. Suele haber, además, una documentación adicional, habitualmente en soporte de papel, en general insuficiente y desactualizada. Este tipo de organización da origen a infinidad de problemas, ya que a veces no se sabe cuál es la descripción de un determinado fichero, bien por pérdida de la misma, bien porque no se ha actualizado debidamente la correspondiente documentación y tampoco se conoce exactamente el programa que lo trataba. En las bases de datos, la descripción, y en algunos casos también una definición y documentación completas (metadatos), se almacena junto con los datos, de modo que éstos están auto documentados, y cualquier cambio que se produzca en dicha documentación se ha de reflejar y quedar en el sistema, con todas las ventajas que de este hecho se derivan.

La actualización y recuperación de los datos debe realizarse mediante procesos bien determinados, incluidos en el SGBD, el cual ha de proporcionar también instrumentos que faciliten el mantenimiento de la seguridad (confidencialidad, disponibilidad e integridad) del conjunto de datos.

El concepto de base de datos ha ido cambiando y configurándose a los largo del tiempo; en la actualidad, y de acuerdo con estas características que acabamos de analizar; podemos definir la base de datos como se muestra en la tabla 1.4

"Colección o depósito de datos integrados, almacenados en soporte secundario (no volatil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independiente de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de los datos."

Tabla 1.4 Definición de base de datos

El sistema de gestión de base de datos (SGBD) es el conjunto de programas que permiten la implementación, acceso y mantenimiento de la base de datos. El SGBD, junto con la base de datos y con los usuarios, constituyen el Sistema de Base de Datos.

9.5.2 Distintos niveles de abstracción de una base de datos

Se puede observar en los SI la existencia de dos estructuras distintas, la lógica (vista del usuario) y la física (forma que se encuentran los datos en el almacenamiento). En las bases de datos aparecen un nuevo nivel de abstracción que se ha denominado de diversas maneras; nivel conceptual, lógico global, etc. Esta estructura intermedia pretende una representación global de los datos que se interponga entre las estructuras lógicas y física de la arquitectura a dos niveles, siendo independiente, tanto del equipo como de cada usuario en particular (véase figura 1.9)

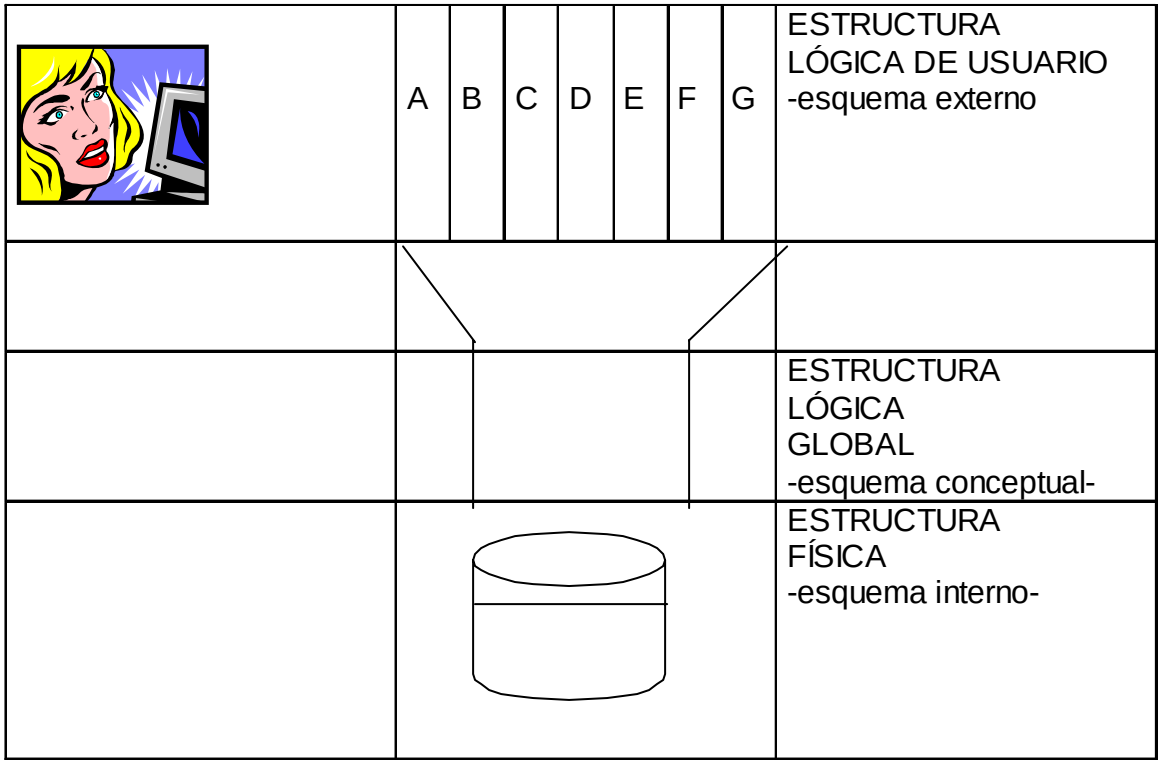


Figura 1.9 **Las tres estructuras de los sistemas de bases de datos**

La estructura lógica de usuario o esquema externo es la visión que tiene de la base de datos cada usuario en particular; la estructura lógica global (también denominada esquema conceptual) responde al enfoque del conjunto de la empresa y la estructura física (o esquema interno) es la forma en que se organizan los datos en el almacenamiento físico. La estructura de una base de datos en estos tres niveles de abstracción tiene como principal objetivo conseguir la independencia entre datos y aplicaciones.

9.5.3 Estructura lógicas de usuarios: esquema externo

Debido a que un esquema externo es la visión que de la base de datos tiene un usuario en particular, en él deberán encontrarse reflejados sólo aquellos datos e

interrelaciones que necesite el correspondiente usuario. También habrán de especificarse las restricciones de uso, como puede ser el derecho a insertar o a borrar determinados datos o el acceso a los mismos, etc. Asimismo, y aunque esto no sea lo más conveniente, ya que indica una dependencia física – lógica, puede que aparezca en este nivel los caminos de acceso a los datos, hecho que dependerá en gran medida del modelo de datos en el que se apoya el correspondiente SGBD; en el modelo relacional los caminos de acceso sólo se encuentran en el nivel interno, no siendo nunca visibles por los usuarios.

Habrán tantos esquemas externos como exija las diferentes aplicaciones. Un mismo esquema externo podrá ser utilizado por varias aplicaciones.

9.5.4 Estructura lógica global: esquema conceptual

En el esquema conceptual, por ser la visión global de los datos, deberá incluirse la descripción de todos los datos e interrelaciones entre éstos, así como las restricciones de integridad y de confidencialidad. La estabilidad de estos conceptos disminuyen en el orden en el que los hemos citado. Así, las restricciones de confidencialidad serán menos estables que las de integridad, y éstas, a su vez, serán menos estables que las interrelaciones o que los datos. Por esta razón, algunos autores proponen que este esquema se divida en varios, uno para cada concepto, de modo que, por ejemplo, un cambio en las restricciones no lleve consigo una nueva definición de todo el esquema.

9.5.5 Estructura física: esquema interno

Aunque el contenido del esquema interno depende mucho de cada SGBD, podemos distinguir tres clases de aspectos que deben especificarse en él.

Estrategia de almacenamiento

En este apartado se incluye la asignatura de espacios de almacenamiento para el conjunto de datos. También deberá indicarse la estrategia de emplazamiento de los datos que ha sido utilizada para optimizar tiempo de respuesta y espacio de memoria secundaria; por último deberán aparecer aspecto como el tratamiento de los desbordamientos, etc.

9.5.6 Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición. Capítulo 1: Sistema de Información y Bases de Datos Pag. 25 – 31

9.6 Clase 6: Historia

ORIGEN Y FINALIDAD DE LAS BASE DE DATOS

CONVENIENCIA DE LA SEPARACIÓN EN SGBD

ABSTRACCIÓN Y MODELO DE DATOS

OBJETIVOS:

- Identificar el origen de las Bases de Datos
- Definir el concepto de abstracción

9.6.1 Origen y finalidad de las Bases de datos (BBDD)

- Primera separación entre proceso y datos en la historia de los ordenadores

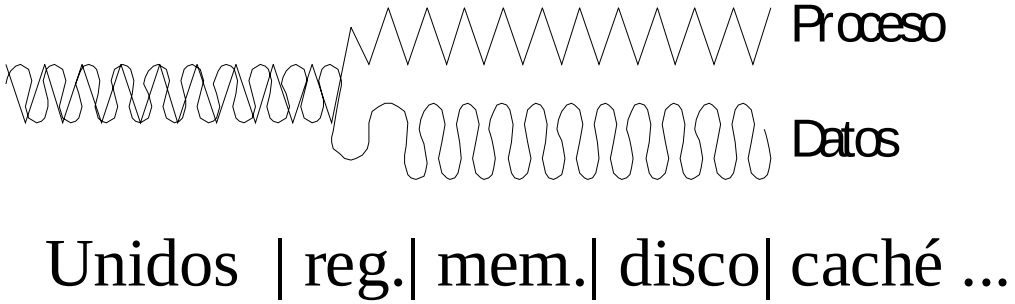


Figura 1.10. **Separación entre proceso y datos**

Jerarquía de memoria

Disco:

Almacén de ficheros

Tamaño creciente (128 Kb ->> 100 x 1024 x 1024 Kb)

Segunda separación entre proceso de datos y gestión de datos

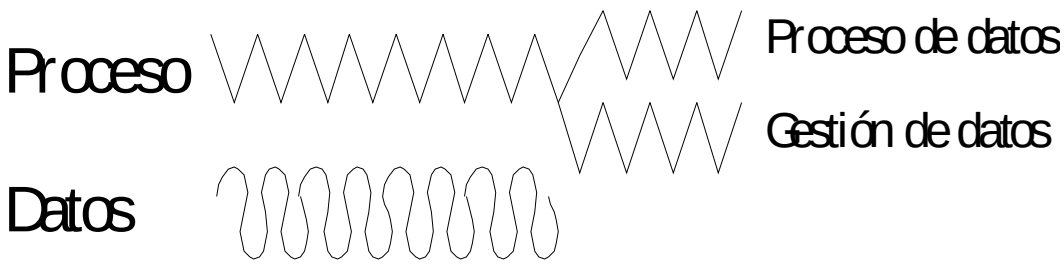


Figura 1.11. **Separación entre proceso dato y gestión de datos**

El control de ficheros era diseñado en cada proyecto

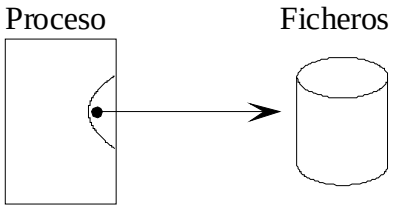
Se fue haciendo más complejo

Muy similar en cada proyecto

Se crea un desarrollo a parte que gestiona los ficheros

Resultado de la segunda separación: el SGBD

Antes



Ahora

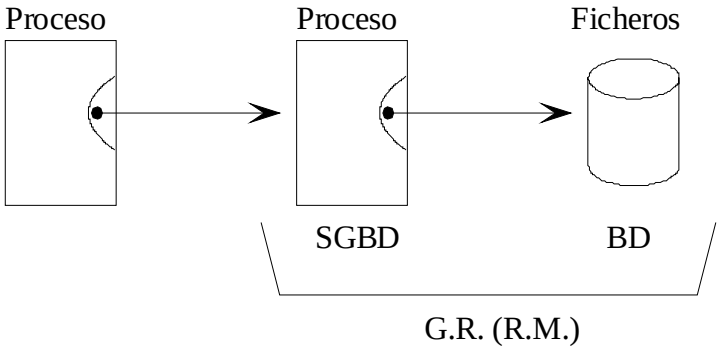


Figura 1.12. **Segunda separación**

9.6.2 Conveniencia de la separación en SGBD

- Beneficiosa
  - Tanto que el almacenamiento directo en ficheros es hoy pieza marginal de los SI
- Ventajas obtenidas
  - Evitar la dispersión de datos en ficheros y aplicaciones, todo se reúne en la BD
  - Mayor control
  - Mayor seguridad (integridad - seguridad)
  - Evitar las islas de información
  - Mayor velocidad
  - Mayor actualización del software

- Mayor sencillez en el uso
- Permite un mantenimiento más específico

9.6.3 Abstracción y modelos de datos

Todo Sistema de Información (en adelante SI) realiza una abstracción del mundo real en términos de los elementos que puede manejar

Un SGBD es un SI específico para la gestión del almacenamiento de información en ficheros

Un SGBD realiza abstracción

La abstracción en un SBGD tiene dos aspectos

- Necesidad: un SI no puede manejar objetos reales, sólo abstracciones de ellos

Lo trataremos más en detalle

Simplificar el manejo de los usuarios ocultar detalles técnicos no útiles

- Abstracción como simplificación

Usuarios de distintos niveles de conocimiento, usuarios con necesidades de productividad elevada

Varios niveles de abstracción

- Físico

  Administrador del SGBD

- Lógico

  Diseñador, programador, administrador de la BD

  Vistas parciales del nivel lógico

- Usuario final

Los niveles deben ser independientes de sus niveles inferiores

Cambios en niveles inferiores no deben influir en los niveles superiores

Especialmente los cambios en el nivel físico

9.6.4 Bibliografía comentada

Manual de: Curso de base de Datos 2003 Impartido por: José María Gutiérrez Martínez Tema 1Introducción e Historia Página 1 a 5

9.7 Clase 7: Sistema de gestión de la base de datos

EL SGBD COMO INTERFAZ ENTRE EL USUARIO Y LA BASE DE DATOS

CONCEPTOS Y PRINCIPALES FUNCIONES DE UN SGBD

OPERACIONES TÍPICAS QUE DEBEN REALIZAR UN SGBD

CONSULTA A LA BASE DE DATOS

FUNCIÓN DE CONTROL

OBJETIVOS:

Conceptuar El SGBD como Interfaz entre el Usuario y la Base de Datos

Definir el Conceptos y Principales Funciones de un SGBD

Identificar las Operaciones Típicas que deben realizar un SGBD

Citar Consulta a la Base de Datos

Identificar las Función de Control

9.7.1 El sistema de gestión de la base de datos

Una vez examinado el concepto de base de datos, analizaremos en esta unidad el Sistema de Gestión de Base de Datos, junto con sus funciones, lenguaje, facilidades, interfaces, etc. insistiremos en el objetivo de independencia, profundizando en la arquitectura a tres niveles de los SGBD y los modelos de referencia de ANSI, y analizaremos las actividades de estandarización en torno a los SGBD. Para finalizar presentaremos el conjunto de herramientas y facilidades que se ofrecen juntos con el núcleo del SGBD.

9.7.2 El SGBD como interfaz entre el usuario y la base de datos

En toda organización se suelen distinguir, como ya hemos indicado anteriormente, tres niveles de gestión: operacional, táctico y estratégico, de modo que el sistema de información estará integrado por tres subsistemas estructurados jerárquicamente y que se corresponden con cada uno de estos tres niveles.

La desconexión, que muchas veces se producen entre estos tres subsistemas de información (los cuales constituyen a menudo sistemas aislados o, incluso, grupos de aplicaciones sin relación alguna), aumenta el coste global de creación y mantenimiento del sistema de información y produce redundancias e incoherencia; es decir; impide una gestión racional de los datos, que debería estar encaminada a obtener el máximo rendimiento de los mismos, tal como se hace con cualquier otro recurso de la empresa.

La base de datos, como depósito único de datos para toda la organización, debe ser capaz de integrar los distintos subsistemas y aplicaciones atendiendo a las necesidades de

los usuarios en los tres niveles, siendo SGBD el que suministra la interfaz entre el conjunto de los datos y los usuarios (véase la figura 1.13)

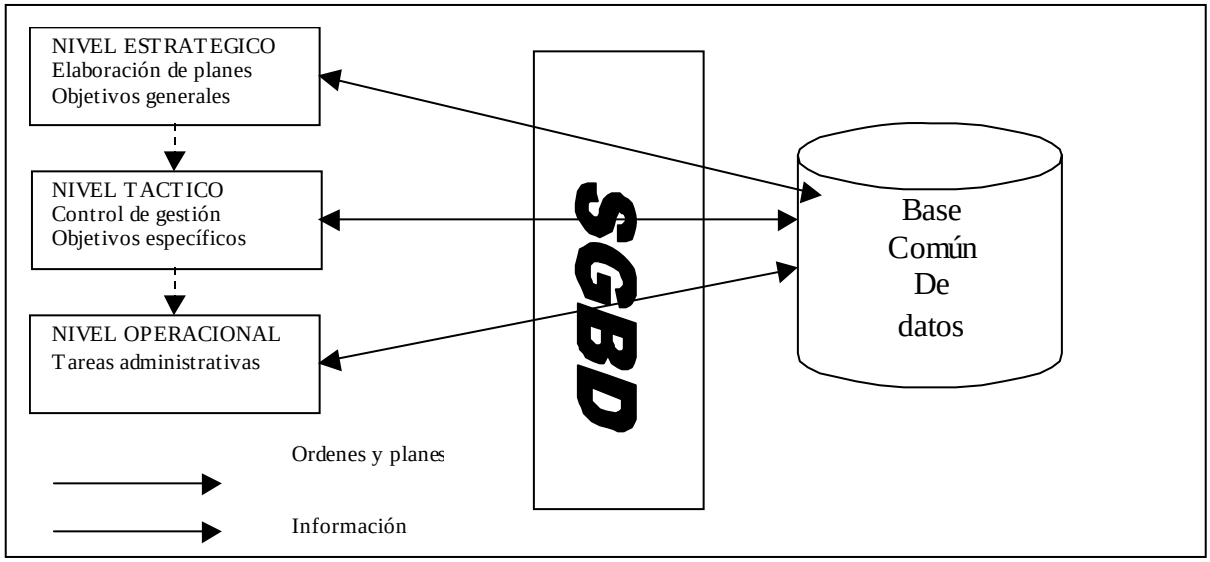


Figura 1.13 **El SGBD como interfaz entre la base de datos y los niveles de gestión de la organización**

Los distintos tipos de usuarios de una base de datos pueden clasificarse en usuarios informáticos y usuarios finales.

9.7.3 Clasificación de los distintos tipos de usuarios de una base de datos

Usuarios informáticos

Tienen a su cargo las tareas de creación y mantenimiento de la base de datos, así como la realización de los procedimientos y programas que necesiten los usuarios finales. Entre ellos se puede distinguir:

Diseñadores

Tiene la responsabilidad de identificar los datos que han de estar contenidos en la base de datos, de acuerdo con las necesidades que les manifiesten los usuarios, así como de determinar las estructuras más apropiadas para conseguir satisfacer estas necesidades.

Según la fase en que intervengan los diseñadores, es posible distinguir:

Diseñadores lógicos

Deben ser capaces de determinar qué tipos de datos han de estar contenidos en la base y conseguir plasmar el punto de vista del conjunto de usuarios en las estructuras lógicas más adecuadas para éstos. Para ello, deben mantener constante entrevistas con los usuarios (incluida la dirección) a fin de que la base de datos represente lo más fielmente posible del mundo real que se trata de recoger. Es decir, los diseñadores lógicos deben perseguir un objetivo de eficacia de la base de datos.

Diseñadores físicos

Su objetivo es transformar las estructuras lógica en estructuras físicas que proporcionen la mayor eficiencia de cara a la máquina, minimizando el tiempo de respuesta y el consumo de recurso, es decir, optimizar el ratio coste/beneficio. Muchas veces esta función es asumida por el administrador.

Administradores

El uso compartido de recursos, propios de una base de datos, obliga a la existencia de un administrador, cuya misión es la vigilancia y gestión de los datos. El principal recurso en una base de datos son los datos, y el administrador debe velar para que éstos no se destruyan ni se contaminen, perdiendo su confidencialidad, disponibilidad e integridad. Por ello, el administrador debe impedir consultas o actualizaciones no autorizadas y proteger la base de datos contra fallos del equipo lógico o físico. Será el responsable, por tanto, de establecer el sistema de autorizaciones de acceso y deberá coordinar y controlar su uso.

También el administrador tendrá a su cargo la gestión de otros recursos distintos de los datos, como pueden ser el SGBD y otras herramientas relacionadas con el mismo. Deberá ocuparse del buen funcionamiento de todo el sistema, sin que se produzcan paradas y de modo que se proporcionen los tiempos adecuados de respuesta.

En muchas organizaciones es la misma persona o grupos de personas las que tienen a su cargo las funciones de diseño y de administración, aunque se trata de papeles muy distintos que convendría diferenciar claramente.

Analista programadores

Los analistas y programadores tienen a su cargo el análisis y la programación de las tareas que no pueden ser llevadas a cabo por los usuarios finales, para lo cual han de desarrollar distintos procedimientos y programas que ponen a disposición de los usuarios finales a fin de facilitarles su trabajo.

Usuarios finales

Los usuarios finales son aquellos que tienen que acceder a los datos porque los necesitan para llevar a cabo su actividad. A diferencia de los usuarios informáticos, su interés suele estar centrado en el contenido de la base de datos, es decir, en los datos. Existen también distintas clases de usuarios finales.

Habituales

Suelen hacer consultas y/o actualizaciones en la base de datos como parte habitual de su trabajo. Utilizando en general menús previamente preparados por analistas y/o programadores, de forma que se facilite su interrelación con el ordenador. En otros casos pueden usar lenguajes sencillos para el acceso a la base de datos, así como paquetes de programas.

Es preciso distinguir, dentro de este grupo, a los operadores de entrada de datos, cuya labor consiste en actualizar la base de datos para lo que se les prepara menús de actualización, aunque con la diferencia respecto a otros usuarios habituales de que tienen

exigencias muy estrictas respecto a los tiempos de respuesta, además de no estar interesados en la información contenida en la base de datos.

Esporádicos

Es un tipo de usuarios muy parecido al anterior en la medida en que necesitan el ordenador a fin de que les preste una ayuda en su trabajo, pero en cambio no lo utilizan habitualmente porque el tipo de actividad que realizan no lo exige. Son por tanto, usuarios a los que se les tiene que suministrar herramientas sencillas y, en general, potente, ya que en bastante casos así lo exige la clase de tareas que han de llevar a cabo; tareas que, en muchas ocasiones, no son fácilmente formalizables, por lo que no pueden ser atendidos por medio de menús.

En resumen, se puede decir que la finalidad del SGBD es establecer las adecuadas interfaces entre los distintos tipos de usuarios y la base de datos.

9.7.4 Bibliografía comentada

I Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini ,Editorial Rama, segunda edición,Capitulo 1: Sistema de Información y Bases de Datos Pag. 25 – 31

9.8 Clase 8: Sistema de Gestión de la Base de Datos

LENGUAJES DE LOS SGBD

OBJETIVOS:

Identificar los procedimientos de los procedimientos de comunicación de las bases de datos.

9.8.1 Lenguajes de los SGBD

Las distintas funciones que ha de cumplir un SGBD hacen necesario disponer de diferentes tipos de lenguajes y procedimientos que permitan la comunicación con la base de datos; unos están orientados hacia la función (definición o manipulación), y otros dirigidos a diferentes tipos de usuarios o de aplicaciones (véase figura 1.14)

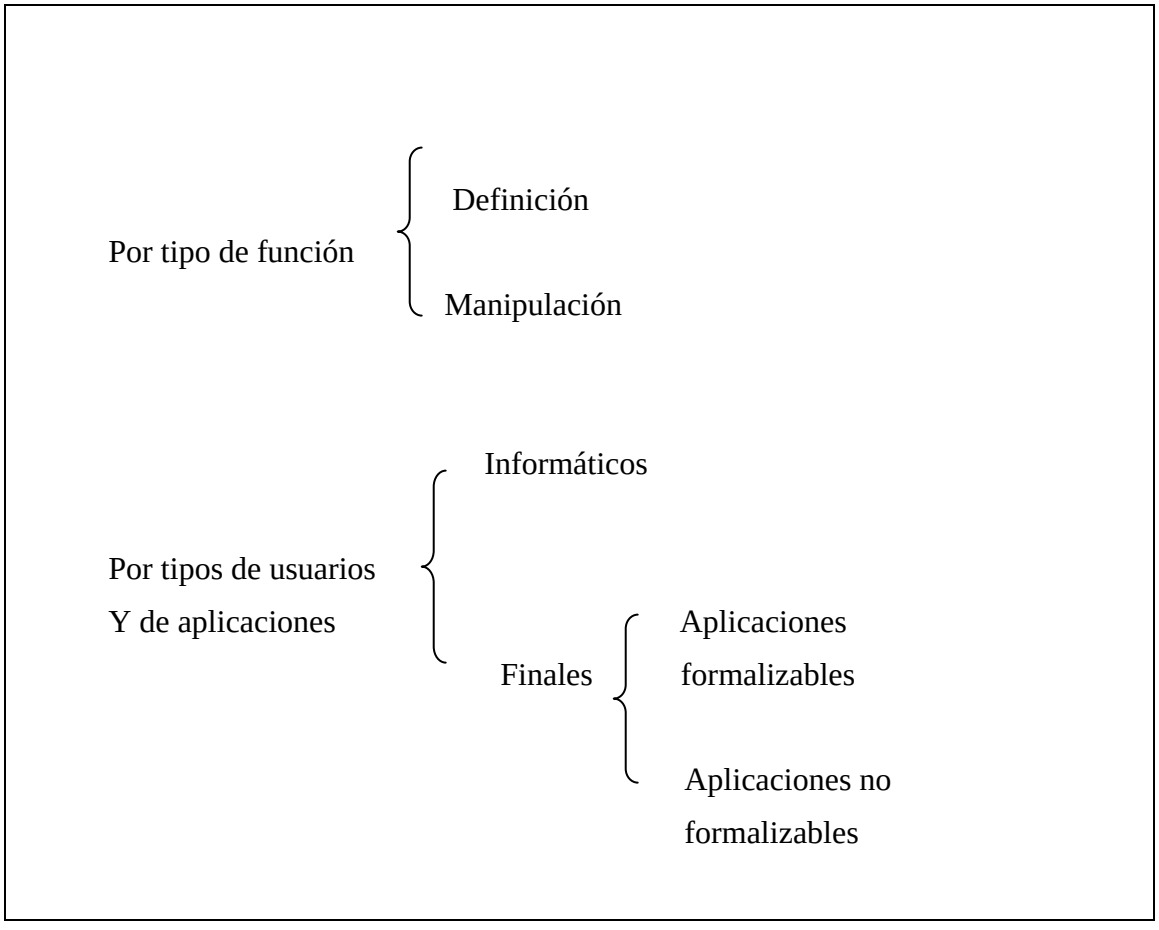


Figura 1.14 **Tipología de los lenguajes de los SGBD**

Si atendemos al tipo de función, tendremos lenguajes de definición y lenguajes de manipulación; y si atendemos al tipo de usuarios, el lenguaje puede estar destinado a usuarios informáticos o a usuarios finales cuyas aplicaciones pueden ser formalizables (como la gestión de personal) o poco fromalizables (como los procesos de toma de decisiones).

Cuando se trata de procesos formalizables y muy repetitivos, el programador se encarga en general de escribir los correspondientes programas, que muchas veces son sometidos a un tratamiento por lotes, con periodicidad fija (emisión diaria de recibir, obtención mensual de la nómina, etc.), o a un tratamiento interactivo (consulta). También es habitual utilizar paquetes que sean soportados por el SGBD.

Por el contrario, si el proceso es difícilmente formalizable o no es lo suficiente repetitivo, escribir un programa puede no resultar rentable, y suele ser más conveniente que el mismo usuario final resuelva directamente la consulta mediante los instrumentos que el SGBD pone a su alcance. Incluso en este caso las necesidades y el nivel de conocimiento de los usuarios serán diferentes, existiendo algunos que, por su formación, podrán dialogar con el ordenador de una forma más libre utilizando lenguajes enfocados al usuarios final; aunque la mayoría precisarán de procedimientos o programas parametrizados (en general de tipo menú) que, aunque menos flexibles, están orientados a unos objetivos más concretos y son más fáciles de usar.

También el SGBD debe proporcionar al administrador de la base de datos un conjunto de procedimientos que le faciliten sus tareas.

Los lenguajes de los SGBD habrán de tener, por tanto, un enfoque distinto según el tipo de usuarios, ya que, indudablemente, las exigencias de un programador o del administrador no tiene nada que ver con lo que puede necesitar un usuario no informático.

En general, los usuarios informáticos, como el administrador de la base de datos, analistas, programadores, etc., requerirán medios potentes y flexibles mediante los cuales consigan definir, administrar, extraer o manipular los datos de la base. En muchos casos desearán apoyarse en el lenguaje de programación que están habituados a manejar (lenguaje anfitrión), para lo cual deberá permitir hacer llamadas desde un programa de aplicación al SGBD. El conjunto de sentencias de manipulación del SGBD que pueden ser llamadas desde un lenguaje de programación, permitiendo así el acceso a la base de datos, se suele denominar sublenguaje de datos y también lenguaje huésped embebido.

Los SGBD admiten varios lenguajes de tipo anfitrión para manipulación de los datos, como Cobol, Ensamblador, Fortran, PL/I, Basic, Pascal, C, etc. Prácticamente la totalidad de los SGBD disponen también de lenguaje de cuarta generación (nombre confuso y controvertido, pero muy extendido), habitualmente poco procedimentales, que permiten el acceso a la base de datos, en general, mediante sentencias embebidas en el lenguaje de cuarta generación y escritas en un lenguaje de datos como el SQL. Los distintos productos comerciales proporcionan diferentes herramientas y lenguajes anfitriones.

El usuario final requerirá medios simples para comunicarse con la base de dato, lo que se puede conseguir mediante un lenguaje de manipulación autocontenido que tenga una sintaxis sencilla, pero lo suficientemente potente como para soportar demandas de información muy variadas (estos lenguajes en ciertos casos sólo permiten la consulta pero

no la actualización), o por medio de tratamiento parametrizados que suelen presentarse al usuario en forma de menús.

La estructura y sintaxis de todos estos tipos de lenguajes dependen de cada SGBD. Las normas Codasyl proponen especificaciones concretas de la sintaxis para los lenguajes de descripción y manipulación que responden a un modelo de datos en red. En el caso de los SGBD relacionales el SQL es un estándar muy extendido que proporciona estas facilidades. Para los modelos en red existe también el estándar de ANSI NDL/ASN, basado en las normas Codasyl, pero que, a diferencia del SQL, ha tenido una incidencia comercial prácticamente nula.

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini Editorial Rama, segunda edición, Capitulo 2: Sistema de Gestión de la Base de Datos Pag 43 - 44.

9.9 Clase 9: Historia

CONCEPTOS Y PRINCIPALES FUNCIONES DE UN SGBD

OPERACIONES TÍPICAS QUE DEBEN REALIZAR UN SGBD

FUNCIONES DE DEFINICIÓN O DESCRIPCIÓN

FUNCIÓN DE MANIPULACIÓN

CONSULTA A LA BASE DE DATOS

FUNCIÓN DE CONTROL

LOS COMPONENTES FUNCIONALES DE UN SISTEMA DE BASE DE DATOS

ESTRUCTURA GENERAL DE UN SGBD

OBJETIVOS:

Identificar las principales funciones de un SGBD

Identificar los componentes funcionales de SGBD

9.9.1 Conceptos y principales funciones de un SGBD

Se puede definir el Sistema de Gestión de Base de Datos (SGBD) como un conjunto coordinado de programas, procedimientos, lenguajes, etc, que suministran a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base, garantizando su seguridad.

Si se tiene en cuenta que, como ya hemos señalado, en una base de datos existe una gran variedad de usuarios, con necesidades diversas y variables a largo del tiempo, los cuales son susceptibles de trabajar simultáneamente con subconjunto de esta colección de datos, se pone de manifiesto que es imprescindible dotar al sistema de la adecuada flexibilidad para que pueda atender las exigencias de todos los usuarios y para que sea capaz de responder a los posibles cambios a un coste no excesivo. Es decir, el SGBD ha de estar diseñado de forma que las ventajas que se han señalados como propias de las bases de datos constituyan una realidad.

9.9.2 Operaciones típicas que deben realizar un SGBD

Las operaciones típicas que deben realizar un SGBD pueden resumirse en aquellas que afectan a la totalidad de los datos (o a todos los registros de un determinado tipo) y las que tienen lugar sobre registros concretos (véase figura 2.2)

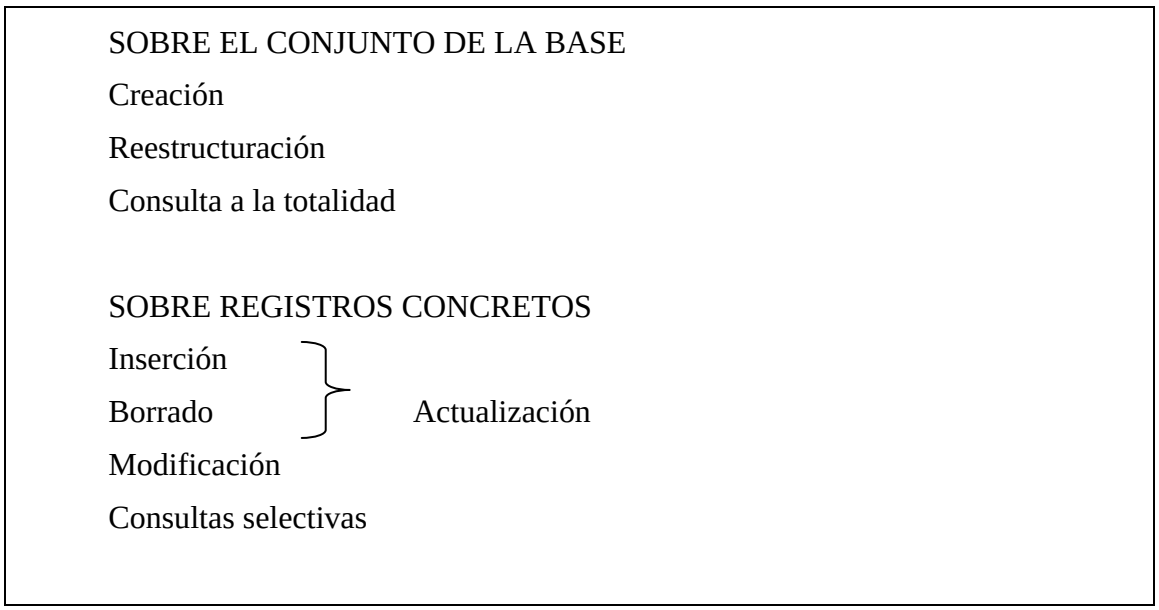


Figura 1.15 **Operaciones típicas sobre una base de datos**

Las funciones esenciales de un SGBD son las descripción, manipulación y control (o utilización).

9.9.3 Funciones de definición o descripción

La función de definición (también llamada de descripción) debe permitir al diseñador de la base de datos especificar los elementos de datos que la integran, su estructura y las relaciones que existen entre ellos, las reglas de integridad semántica, etc., así como las características de tipo físico y las vistas lógicas de los usuarios.

Esta función, realizada por el lenguaje de descripción o definición de datos (LDD) propio de cada SGBD, debe suministrar los medios para definir las tres estructuras de datos (externas, lógica global e interna), especificando las características de los datos a cada uno de estos niveles.

A nivel interno, se ha de indicar el espacio (volumen, cilindros y pistas) reservado para la base, la longitud de los campos o elementos de datos, su modo de representación (binario, decimal, alfanumérico, punto fijo o flotante,etc.). Además, se deben poder definir caminos de acceso, como punto, índices, etc.

Para las estructuras externas y lógicas global, la función de descripción ha de proporcionar los instrumentos para la definición de los objetos (entidades, tablas, registros, etc.), así como su identificación, atributos de los mismos, interrelaciones entre ellos, autorizaciones de acceso, restricciones de integridad, etc. Las descripciones de las estructuras lógicas de los usuarios han de estar referida a la estructura lógica global. El SGBD, además de suministrar facilidades de descripción, se ocupará de la función de correspondencia o transformación (Mapping) de la estructura lógica global a las estructuras externas, y entre aquellas y la estructura física.

9.9.4 Función de manipulación

Una vez descrita la base de datos, es preciso cargar los datos en las estructuras previamente creadas, con lo que la base de datos estará ya dispuesta para su utilización. Los usuarios tendrán necesidades de recuperar la información (consultar la base de datos), o bien de actualizar porque se hayan producido cambios en los datos.

9.9.5 Consulta a la base de datos

Las consulta a la base de datos puede ser de dos tipos:

Totalidad de los datos, en la que se recuperan todos los datos de la base de datos o todos los de un determinado tipo; por ejemplo, para la confección de la nómina será preciso recuperar todos los registros de los empleados de la empresa.

Consulta selectiva, en la que se tendrá que localizar los registros que cumplan una determinada condición (criterio de selección); por ejemplo, obtener los empleados que sean informáticos y sepan inglés.

En ambos casos será preciso especificar la estructura lógica externa que se desea recuperar; así, nombre, datos bancarios y salarios en el ejemplo de la nómina; nombre, departamento y categoría profesional en el ejemplo de la consulta selectiva. El SGBD deberá, con estos datos, acceder a la estructura física de la base de datos donde se encuentran almacenados los datos, localizar aquellos registros indicados y ponerlos a disposición del usuario.

La actualización o puesta al día de una base de datos supondrá tres tipos de operaciones distintas:

Inserción, cuando aparezcan nuevos elementos; por ejemplo, en un fichero de personal es preciso dar de alta a los nuevos empleados.

Borrado, porque hayan desaparecido algunos elementos; por ejemplo en el fichero de personal es preciso dar de baja a los empleados que ya no están en la empresa.

Modificación de los datos de aquellos registros en los cuales se hayan producido cambios; cuando se ha alterado se ha alterado la categoría profesional de un empleado.

En la figura 1.16. se representa la interacción del usuario con la base de datos.

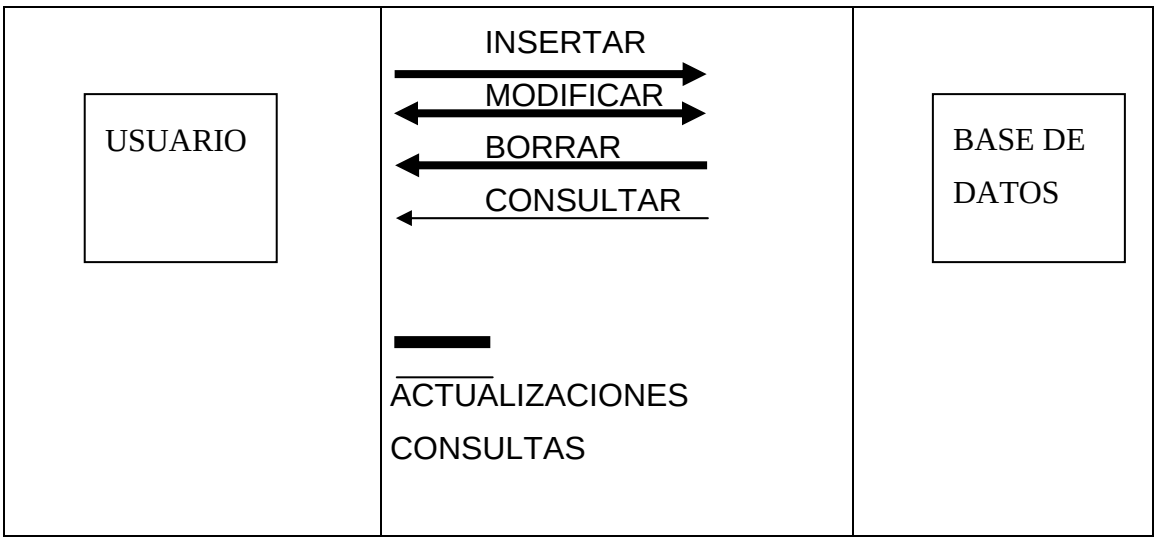


Figura 1.16 **Interacción Usuario / Base de Datos**

La función de manipulación permite a los usuarios de la base, informáticos o no, buscar, añadir, suprimir o modificar los datos de la misma, siempre de acuerdo con las especificaciones y normas de seguridad dictadas por el administrador.

La función de manipulación se llevará a cabo por medio de un lenguaje de manipulación de datos (LMD) que facilita los instrumentos necesarios para la realización de estas tareas. Muchas veces se trata de un conjunto de mandatos (lenguaje huésped) que se escriben en un lenguaje de programación (lenguaje anfitrión); mientras que otras veces se trata de un lenguaje autocontenido que no precisa apoyarse en ningún otro lenguaje, ya que dispone en sí mismo del conjunto de instrucciones necesarias para llevar a cabo tanto la recuperación como la actualización de los datos. La mayoría de los SGBD actuales atienden la función de manipulación mediante ambos tipos de lenguajes, huéspedes y autocontenidos; estos últimos, orientados a los usuarios no informáticos, suelen usarse de forma interactiva.

9.9.6 Función de control

Esta función reúne todas las interfaces que necesitan los diferentes usuarios para comunicarse con la base y proporciona un conjunto de procedimientos para el administrador.

Las exigencias respecto a la forma de utilizar la base de datos son muy diferentes, según los tipos de procesos y según los usuarios, siendo preciso que la función de utilización responda a todas ellas. En especial, esta función debe integrar una serie de instrumentos que faciliten las tareas del administrador. En la mayoría de los SGBD existen funciones de servicios, como cambiar la capacidad de los ficheros, obtener estadísticas de utilización, carga, archivo, etc., y principalmente las relacionadas con la seguridad física (copia de seguridad, recuperación rearranque en caso de caída del sistema, etc.) y de protección frente a acceso no autorizados. Todas ellas se consideran comprendidas.

Se presenta un cuadro con las funciones esenciales de un SGBD

DESCRIPCIÓN

Permite describir

Los elementos de datos con

- Su estructura
- Sus interrelaciones
- Sus validaciones

A tres niveles:

Externo

Lógico Global

Interno

Mediante un LDD

MANIPULACIÓN

Permite:
Buscar
Añadir
Suprimir
} datos de la base

Modificar

Mediante un LMD

Lo cual supone:

Definir un criterio de selección (responsabilidad del usuario)

Definir las estructuras externas a recuperar (responsabilidad del usuario)

Acceder a la estructura física (responsabilidad del sistema)

CONTROL

Reúne las interfaces de los usuarios

Suministra procedimientos para el administrador

Figura 1.17 **Funciones esenciales de un SGBD**

Tomado de: “Fundamentos de Bases de Datos” 4ª edición. 2002. Silberschatz define

Un sistema de base de datos se encuentra dividido en módulos cada uno de los cuales controla una parte de la responsabilidad total de sistema. En la mayoría de los casos, el sistema operativo proporciona únicamente los servicios más básicos y el sistema de la base de datos debe partir de esa base y controlar además el manejo correcto de los datos. Así el diseño de un sistema de base de datos debe incluir la interfaz entre el sistema de base de datos y el sistema operativo.

9.9.7 Los componentes funcionales de un sistema de base de datos

Gestor de archivos.

Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar información.

Manejador de base de datos.

Sirve de interfaz entre los datos y los programas de aplicación.

Procesador de consultas.

Traduce las proposiciones en lenguajes de consulta a instrucciones de bajo nivel. Además convierte la solicitud del usuario en una forma más eficiente.

Compilador de DDL.

Convierte las proposiciones DDL en un conjunto de tablas que contienen metadatos, estas se almacenan en el diccionario de datos.

Archivo de datos.

En él se encuentran almacenados físicamente los datos de una organización.

Diccionario de datos.

Contiene la información referente a la estructura de la base de datos.

Índices.

Permiten un rápido acceso a registros que contienen valores específicos.

Una forma gráfica de representar los componentes antes mencionados y la relación que existe entre ellos sería la siguiente.

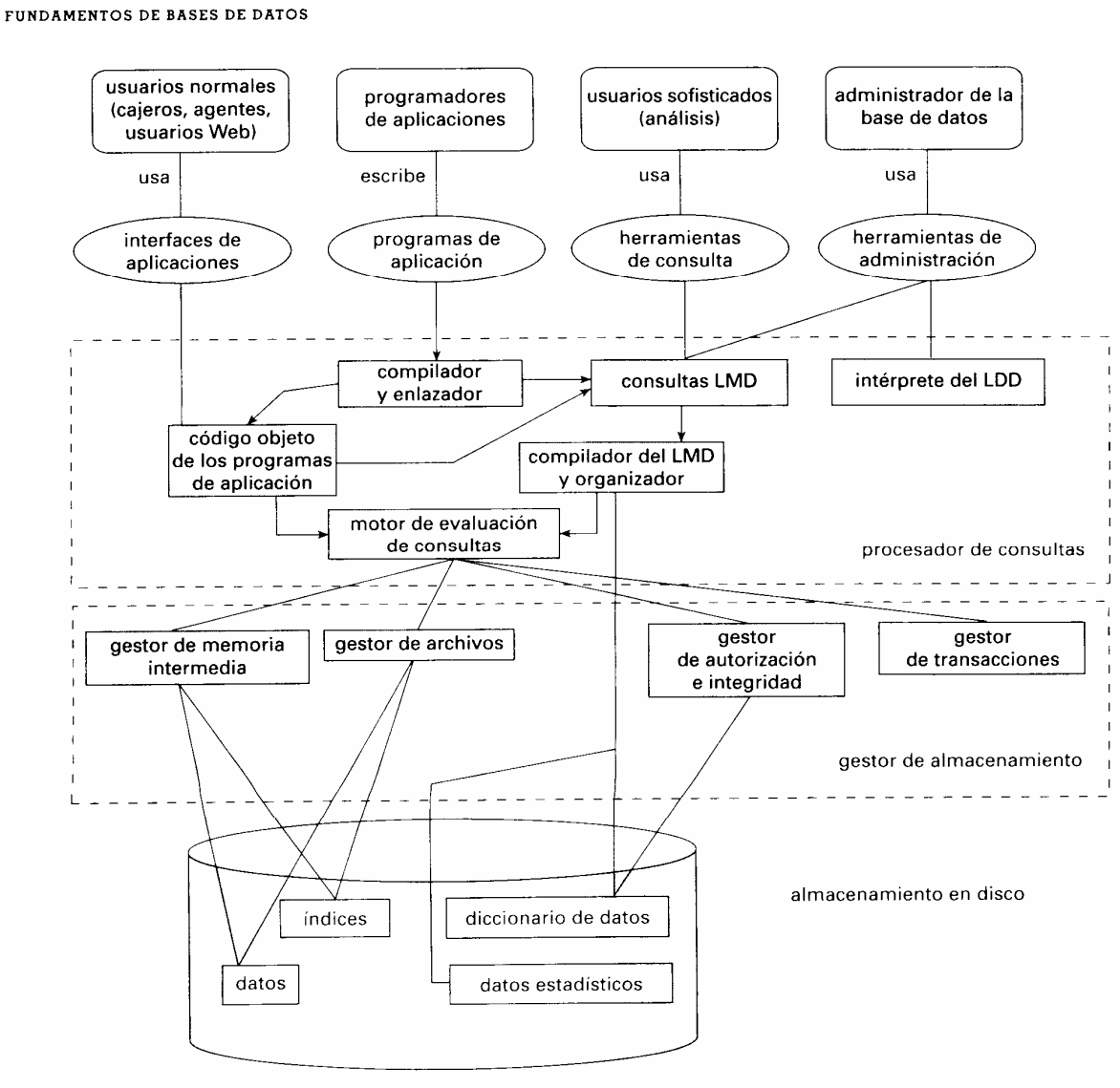


Figura 1.18 **Separación entre proceso dato y gestión de datos**

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS. Adoración de Miguel, Mario Piattini, Editorial Rama, segunda edición, Capitulo 2: Sistema de Gestión de la Base de Datos Pag 43 - 44. Capitulo 2: Sistema de Gestión de la Base de Datos Pag. 36 – 42,

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez, Tema I Introducción e Historia Página 6

9.10 Clase 10: Historia

PERSPECTIVA FUNCIONAL

OBJETIVOS:

Visualizar las perspectivas funcionales de las bases de datos

9.10.1 Perspectiva funcional

Desde este punto de vista veremos los avances en el uso de las BBDD y su evolución en características

La evolución sigue la línea temporal

Fases

Anterior a los computadores

Tarjetas perforadas, sistemas mecánicos

Década 1950, principios década 1960

Cintas perforadas, paquetes de tarjetas perforadas, cintas magnéticas

Acceso secuencial

Maestro, movimientos

Finales década 1960 y década 1970

Discos fijos

Acceso directo, todo dato a decenas de milisegundos

Bases de datos en red y jerárquicas

Posibilidad de almacenamiento de árboles y redes en los discos fijos

1970 Codd publica un artículo con la definición del modelo relacional

Década 1980

El modelo relacional no puede competir en rendimiento con los modelos en red y jerárquico

IBM crea System R. 1976

Primeros sistemas relacionales eficientes

Competencia con sistemas red y jerárquico

Supremacía total de sistemas relacionales

Trabajos iniciales de BBDD orientadas a objetos

Principios década 1990

Cambios en las necesidades de los usuarios

Antes grandes consultas

Ahora muchas actualizaciones

Nuevos tipos de BBDD

Primeras implementaciones funcionales

Ámbitos de aplicación específicos

Resurgen las necesidades de grandes consultas

Productos específicos

Finales década 1990

Crecimiento explosivo de WWW

Bases de datos para cualquier tarea

Desbancando finalmente a los ficheros

Bases de datos fiables, 24x7

Rendimientos elevados sobre el modelo relacional

9.10.2 Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez, Tema 1Introducción e Historia Página 7 a 8

9.11 Clase 11: Historia

PERPECTIVA TECNOLOGICA

LAS TRES ETAPAS FUNDAMENTALES EN LA EVOLUCIÓN

PRRELACIONALES

RELACIONES

POSTRELACIONALES

OBJETIVOS:

Identificar las evoluciones fundamentales de las bases de datos

9.11.1 Perspectiva tecnológica

Primeros pasos no son basados en modelos matemáticos o teóricos

Basados en resolución de problemas observados

Esta forma de trabajar que da lugar a la aparición de los sistemas

Condiciona la evolución durante muchos años

Los sistemas creados en las primeras etapas son específicos y dependientes del HW

Condiciona las posibilidades de los sistemas

Nivel de optimización elevado

9.11.2 Las Tres etapas fundamentales en la evolución

Prerelacional

Relacional

Postrelacional

Ordenadas cronológicamente

Sin límites definidos

Actualmente, Sin acabar Prerelacional

Estamos en Relacional

Comenzando Postrelacional

La etapa Prerelacional

- Aparición de las BBDD
- Evolución antes de la aparición del Modelo Relacional

El Modelo Relacional es el primero con modelo formal definido antes de implementar

Las tres etapas presentan nivel de abstracción creciente

Prerrelacionales

Como presentación general veremos características comunes

1. Estos sistemas han tenido una vida más larga que cualquier otro modelo de BBDD

2. Estos sistemas presentan un nivel de abstracción muy bajo

3. Manejo de los datos en la BD registro a registro

4. Los sistemas son meras abstracciones del sistema de almacenamiento subyacente y conservan muchas características del mismo

5. Según C. J. Date, la equivalencia entre el nivel de abstracción de los sistemas de BD y los lenguajes de programación es:

Ada ----------> Cobol ----------> Ensamblador

| | |

relacional----------> no relacional ----------> sistema de ficheros

6. No se define un modelo a priori, el modelo se infiere a posteriori a partir de las implementaciones

7. Son sistemas orientados a la programación, tipo COBOL. La optimización del espacio en BD y del acceso a la información, la realiza el programador. El programador es el único tipo de usuario previsto en este tipo de sistemas.

8. Los sistemas que aún funcionan intentan ofrecer una capa de acceso relacional al substrato no relacional, con las siguientes deficiencias:

Elevada complejidad y muchos problemas técnicos

No se consigue totalmente

Si el acceso se permite no relacionalmente, se pueden violar las reglas de integridad del modelo relacional, haciendo imposible el mantenimiento de la apariencia relacional.

Los modelos no existían a priori

Los modelos son extracciones de características comunes de productos

Ningún producto cumple el modelo del todo

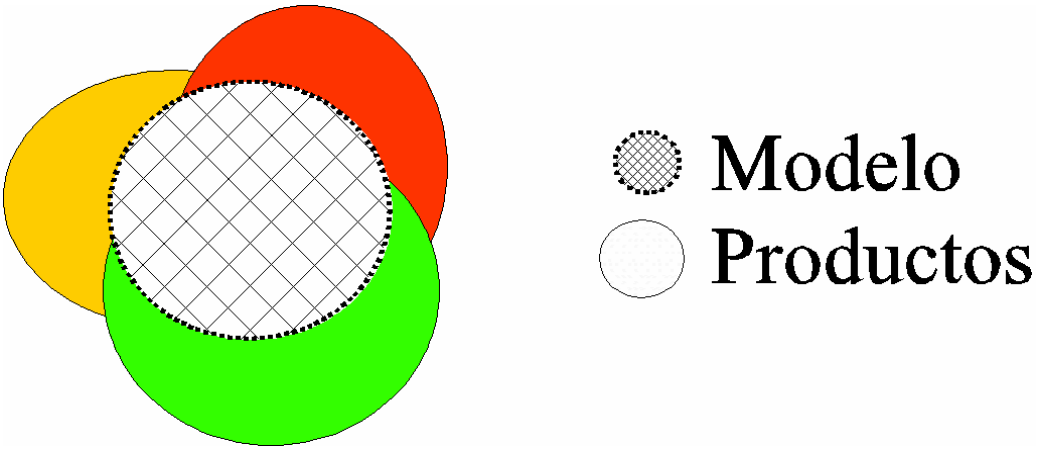


Figura 1.19 **Conjunto de Modelo y Producto**

Modelo jerárquico

BD más significativa: IMS (1968)

Plataforma: computadores IBM VS

Integridad de datos:

Sólo las reglas de mantenimiento de la propia estructura del árbol:

No se tienen hijos sin padre, salvo el nodo raíz

no se permite la creación o se borran automáticamente al borrar el nodo padre

Modelo de red (Codasyl)

BD más significativa: IDMS

Plataforma: computadores IBM (MVS, VSE)

Está basado y casi identificado con el modelo Codasyl

Este modelo no es conceptual sino de estructura interna de la BD

Modelo de datos:

Se puede decir que es un modelo jerárquico extendido. Sustituye el árbol por la liga que es más flexible

Otros conceptos:

La pieza base está formada por 1 nodo padre, 1 liga y n nodos hijo

Un hijo puede ser padre en otras ligas

Un nodo puede ser hijo en distintas ligas

Existe un orden establecido sobre los nodos padre

Existe un orden sobre los nodos hijo de una liga

Manipulación de datos:

Se realiza nodo a nodo.

La operativa consiste en localizar un nodo padre, siguiendo el orden establecido de padres o mediante una búsqueda por contenido de un campo del nodo. Después se siguen las ligas de este nodo hasta un nodo hijo y de este a otro hasta el que buscamos

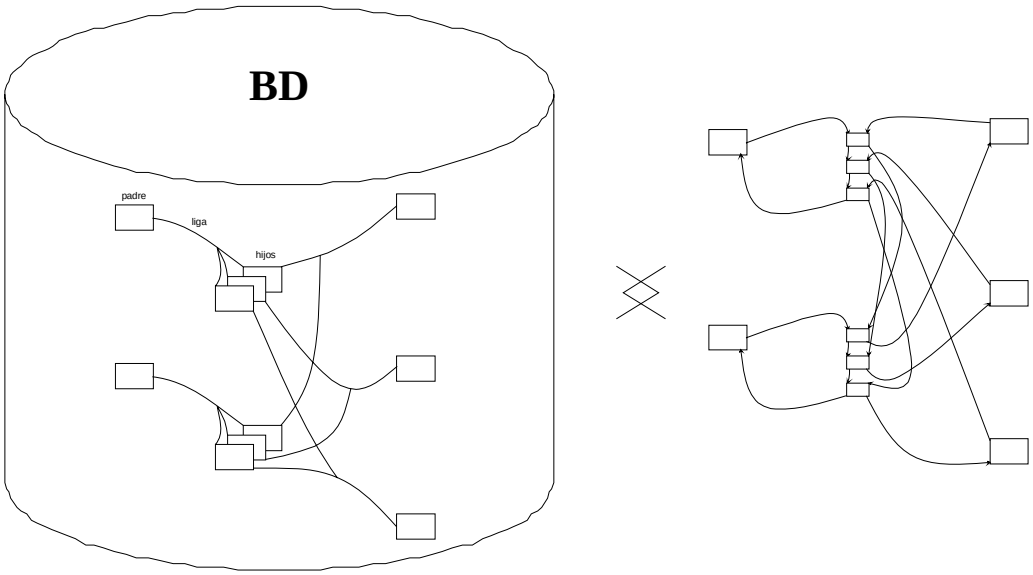


Figura 1.20 **Manipulación de datos**

Integridad de datos:

Existe un control de ligas con normas de creación, destrucción e integridad de las mismas, como puede ser evitar tener hijos sin padres y otros

Existe mayor posibilidad de ejercer control que en modelo jerárquico

Relacionales

Este modelo será estudiado en mayor profundidad

Únicamente plantearemos su situación respecto de otros modelos

Es el modelo más extendido actualmente, el más estudiado, implementado y utilizado por el público

Convive con los modelos prerelacionales y su tiempo de vida, comparado con estos es mucho menor.

Sus características diferenciadoras son:

El modelo teórico se plantea antes que las implementaciones

Su organización es independiente del modelo físico

Está orientado a las consultas del usuario, no al desarrollo de aplicaciones por parte de programadores

Opera sobre conjuntos de registros como unidad de trabajo habitual

Posee un mayor nivel de abstracción que los modelos precedentes y, por tanto, se encuentra más cerca del ser humano

Postrelacionales

Son modelos en evolución desarrollo o aceptación, planteados muchas veces como avances sobre el modelo relacional

Varios de ellos son extensiones sobre el modelo relacional más que conceptos novedosos

Varios son particularizaciones del modelo relacional para optimizar el rendimiento de la BD frente a cierto tipo de datos especiales o para ciertas aplicaciones concretas

9.11.3 Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez. Tema I Introducción e Historia Página 8 a 16

9.12 Clase 12: DISEÑO DE BBDD

INTRODUCCION

TERMINOLOGIA DE BD

DISEÑO DE BBDD Y CICLO DE VIDA DE SI

CICLO DE VIDA TÍPICO DE SI

OBJETIVOS:

Definir terminologías básicas de las bases de datos

Identificar los ciclos de vida de un Sistema de Información

9.12.1 Terminología BD

Conjunto de información en memoria masiva de un ordenador

Sistema Gestor de BD

Sistema software que permite gestionar toda esa información

DISEÑAR BD

Proceso que permite determinar la organización, estructura y contenido de la BD y aplicaciones para las que la BD funcionará

Punto de partida

Año 1970

Confusión entre diseño de BD e implementación

Se entendía el desarrollo de la BD como diseño

En 1990 se produce evolución de tecnologías en Gestión de BD

Evolución paralela del diseño y modelado de BD

Creación de un marco teórico

Las metodologías de diseño para BD no están muy extendidas

Es causa de inconsistencias en sistemas de información

Hay presión por terminar pronto, desarrollos parciales sin visión global, documentación deficiente por no tener un diseño adecuado

Eleva los costos de mantenimiento

Actualmente los diseños tienen los siguiente problemas:

No capturan la naturaleza abstracta de los datos, se trabaja a un nivel concreto

Se describen los datos en función del almacenamiento donde van a residir

9.12.2 Diseño de BBDD y Ciclo de vida de SI

¿Qué es Sistema de Información?

Colección de actividades que regulan la compartición y distribución de la información y el almacenamiento de los datos relevantes para la gestión de la organización

BD es componente del SI

El diseño de BD forma parte del ciclo de vida de un SI

Fases del ciclo de vida de un SI

Estudio de viabilidad

Permite evaluar si se puede abordar el proyecto

Se estudian alternativas

Se plantean prioridades entre las distintas líneas de actuación

Requerimientos y análisis

Se establecen las metas del proyecto de forma detallada

Se analiza el problema

Interacción con usuario, arquitectos informáticos, lenguaje semi-natural

Diseño

Diseño de BD y Diseño de aplicación

Prototipado

Basado en 4GL y Herramientas case

Es el equivalente a una maqueta de un edificio

Migración Inicial de datos.

Es fundamental reutilizar los datos que ya existen en la organización

Hay que tener en cuenta la migración de datos en la fase de diseño, por el diseño de la nueva BD

La migración se realiza justo antes de entrar en Operación

La separación de las fases suele ser borrosa

9.12.3 Ciclo de Vida típico de SI:

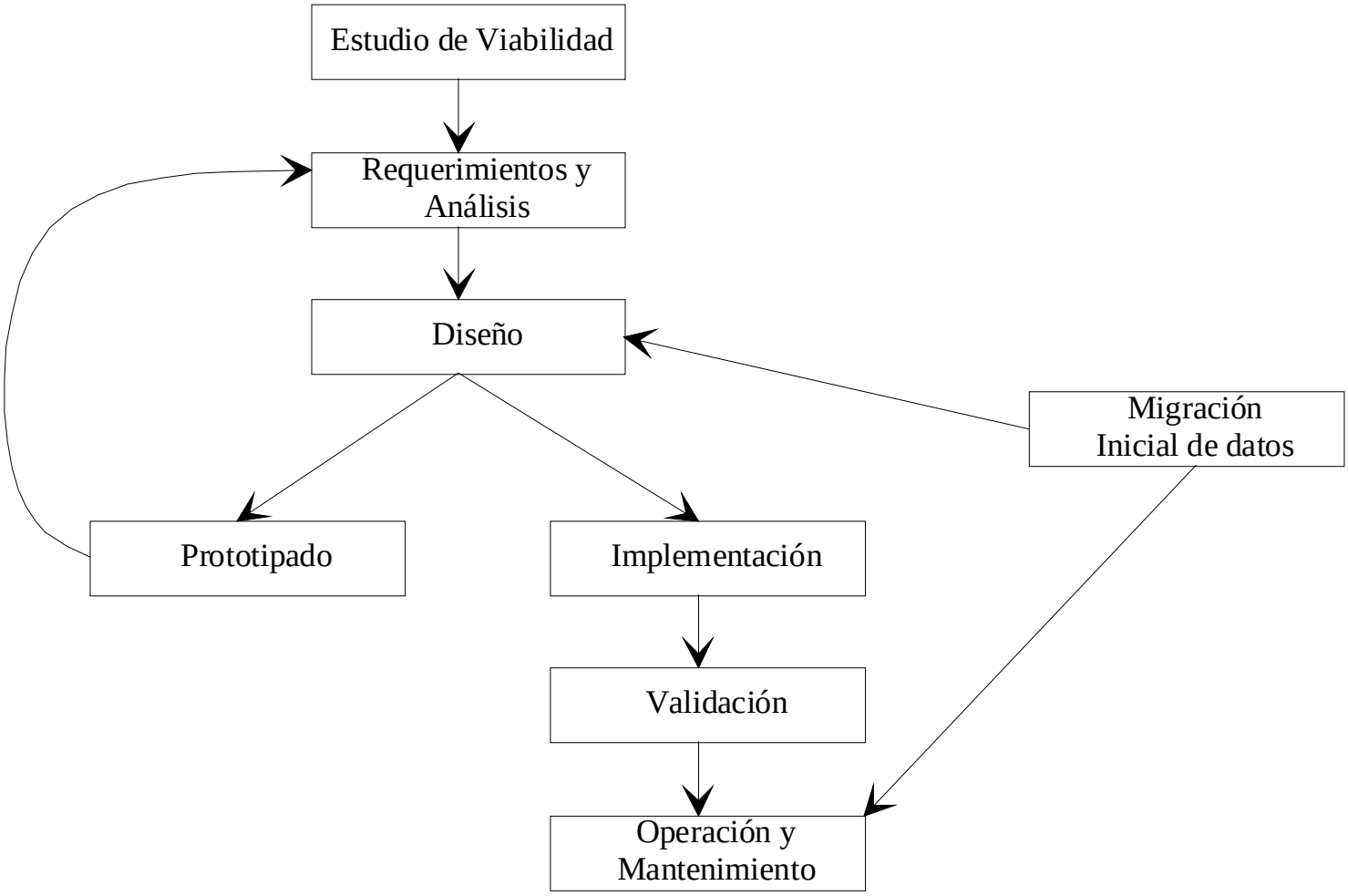


Figura 1.21 **Ciclo de Vida de un SI**

9.12.4 Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez. Tema 2 Diseño de Bases de Datos paginas 1 a 5

9.13 Clase 13: DISEÑO DE BBDD

FASES DE DISEÑO DE BBDD

DISEÑO D-DATO

DISEÑO DE D-DATOS, D-FUNCION

OBJETIVOS:

Definir las fases de diseño de bases de datos

Identificar las características del diseño de datos y el diseño de la función

9.13.1 Fases de diseño de BBDD

En la fase de Diseño del SI, hay dos líneas paralelas de trabajo: D-Dato, D-Función

La línea D-Dato (Dirigida al dato) establece el diseño de la parte estática de la aplicación

Es el diseño de la BD

La línea D-Función (Dirigida a la función) diseña los procesos sobre los datos

Ambas líneas deben realizarse en paralelo en el tiempo

9.13.2 Diseño D-Dato

Tres etapas en el diseño:

Conceptual Captar informaciones para llegar a conceptos relacionados y concretos de una serie de información se abstraen entidades (alumnos, asignaturas ...)

Lógico Pasar de las abstracciones a unos entes ya implementables

Físico Implementación, detalles de espacio, representación física ...

El paso 3) depende el SGBD con el que trabajaremos.

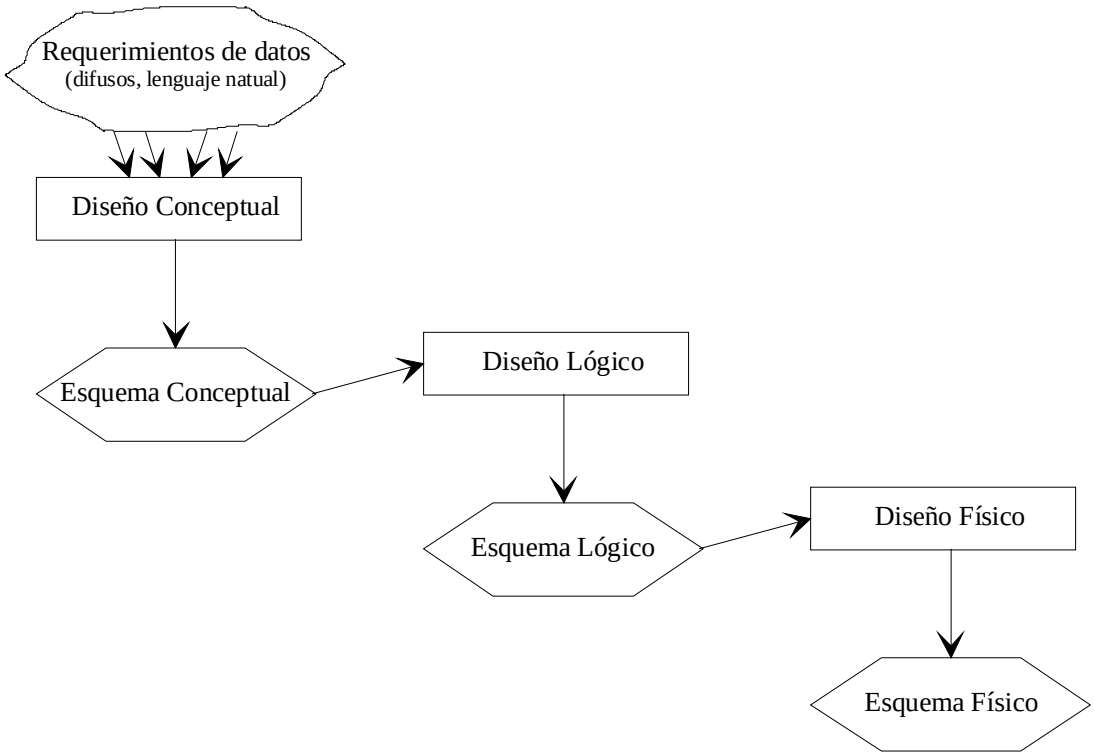


Figura 1.22 **Diseño de Datos**

9.13.3 Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez. Tema 2 Diseño de Bases de Datos paginas 1 a 5

9.14 Clase 14: Modelo de datos

CONCEPTO DE MODELO DE DATOS

DEFINICIÓN DE MODELO DE DATOS

ESTÁTICA

ELEMENTOS NO PERMITIDOS O RESTRICCIONES

DINÁMICA

PLANO CONCEPTUAL PARA EXPRESAR UNA SENTENCIA

OBJETIVOS:

Definir el Concepto de Modelo de Datos

Identificar la diferencia entre el Concepto de Modelo de Datos y la Definición de Modelo de Dato

Identificar los elementos no permitidos del modelo conceptual

Identificar el modelo conceptual para expresar una sentencia

9.14.1 Concepto de modelo de datos

Como hemos señalado en la primera parte, los datos, en un principio, se organización a fin de atender las necesidades de cada proceso; posteriormente se intenta que respondan a los requisitos de un conjunto de proceso; y, por último, se busca una interpretación de la realidad con el fin de captar su semántica. En este capítulo examinaremos el concepto de modelo de datos, que sirve de soporte a esta interpretación, insistiendo en la importancia de las restricciones, estudiaremos los distintos tipos de modelos y, por último, analizaremos el papel de los modelos en el diseño de base de datos.

9.14.2 Introducción

En una primera aproximación podemos decir que un Modelo de Datos (MD) es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos, a la cual denominaremos esquema. Según el nivel de abstracción de la arquitectura ANSI a tres niveles en el que se encuentre la estructura descrita, el modelo que permite su descripción será un modelo externo, global o interno (véase la figura 3.1), cada uno de los cuales ofrece distintos elementos de descripción. Los modelos externos nos permiten representar los datos que necesita cada usuario en particular con las estructuras propias del lenguaje de programación que va a emplear. Los modelos globales ayudan a describir los datos para el conjunto de usuarios, podríamos decir que es la información a nivel de empresa; y, por último, los modelos internos (también llamados físicos) están orientados a la máquina, siendo sus elementos de descripción punteros, índices, agrupamientos, etc.

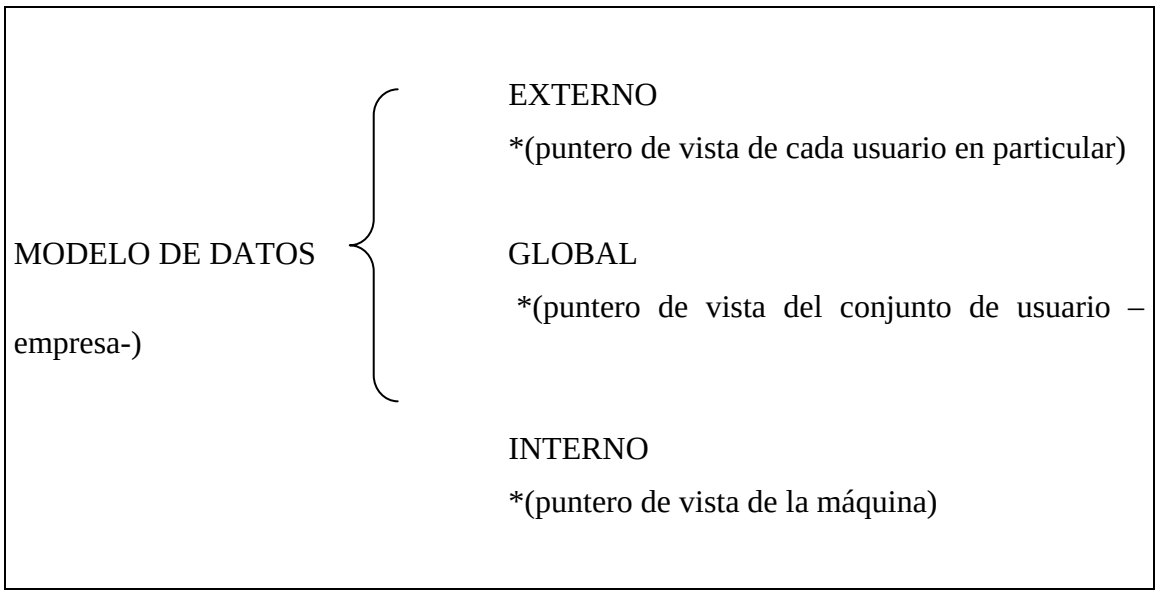


Figura 1.23 **Tipos de modelos de datos corresponden a cada nivel de abstracción de una arquitectura a tres niveles**

De entre los distintos tipos de modelos, es en los globales en los que vamos centrar nuestra atención, ya que los extremos suelen utilizar los mismos conceptos que los correspondientes globales y los internos no están estandarizados ni existen en realidad como tales modelos, sino que son propios de cada uno de los productos comerciales. Los modelos globales se clasifican, a su vez, en conceptuales y convencionales.

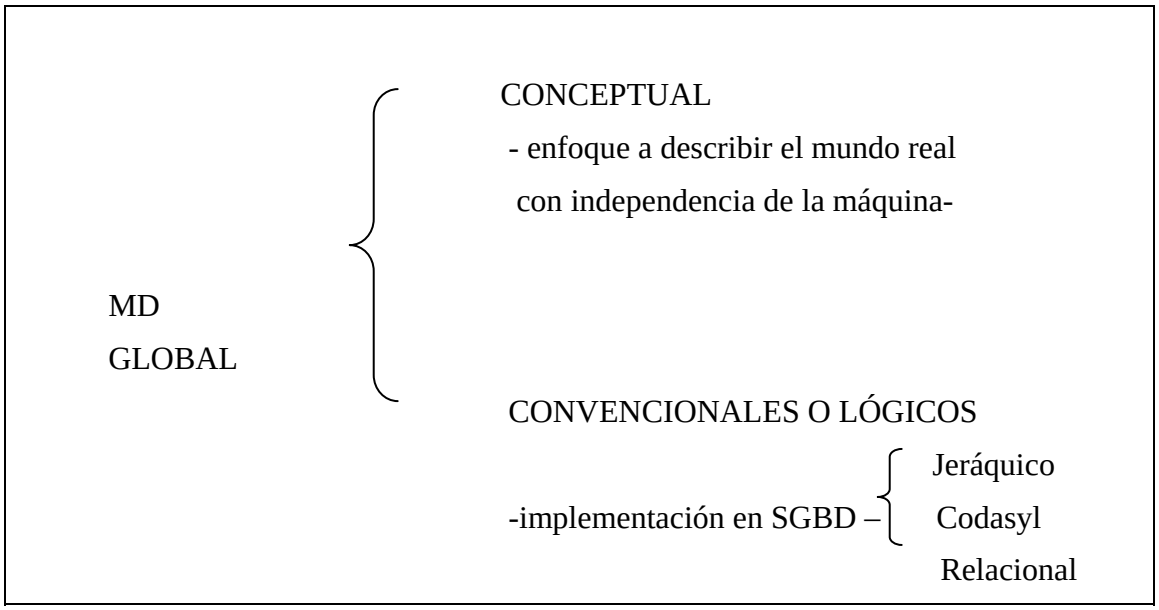


Figura 124 **Clasificación de los modelos de datos globales**

Los modelos conceptuales (también denominados de alto nivel) facilitan la descripción global de información de la empresa con independencia de la máquina (tanto del hardware como del SGBD concreto), por lo que sus conceptos son cercanos al mundo real (entidades, atributos, interrelaciones, etc.); son modelos de análisis, no de implementación. Los modelos convencionales se encuentran soportados por los SGBD (de ahí que también reciben el nombre de modelos de bases de datos), por lo que sus conceptos son propios de cada SGBD (tablas o relaciones en el caso del modelo Relacional, redes, Codasyl, árboles en el Jerárquico, etc.).

El modelo de datos (sea lógico o físico) es el instrumentos que se aplica a los datos para obtener el esquema (véase figura 3.3).

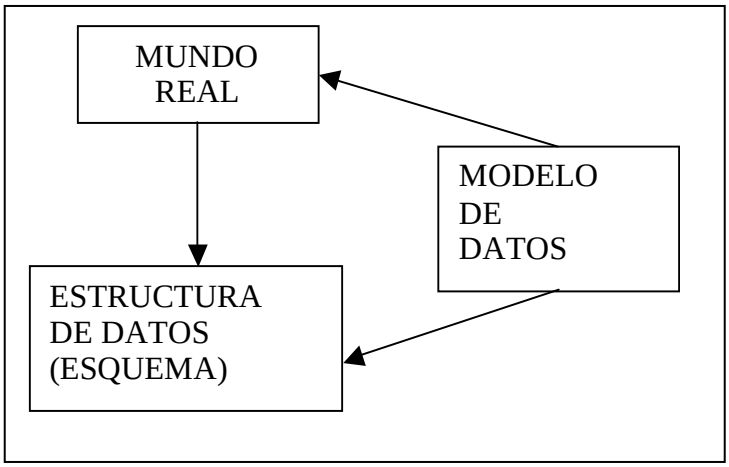


Figura 1.25 **Aplicación de un modelo de datos a un mundo real para obtener un esquema**

Es preciso distinguir entre esquema, como descripción de la estructura de la base de datos, y ocurrencia del esquema, que son los datos que se encuentran almacenados en el esquema en un determinado momento. El esquema no varía mientras no varíe el mundo real que éste descrie; en tanto que una ocurrencia del esquema, es decir, los datos contenidos en el transcurso del tiempo. Según observar DATE (1995), al igual que en los lenguajes de programación existen variables (constituidas por un tipo y un contenido), las cuales tienen en un momento determinado cierto valor, del mismo modo, en las bases de datos se debería hablar de variables de bases de datos, cuyo tipo sería el esquema. A pesar de que nos parece muy acertada esta precisión, y volveremos a comentarla al exponer el modelo relacional (en cuyo contexto la introduce Date), en general, nos mantendremos en la terminología clásica, refiriéndonos a esquema y ocurrencia.

Por lo que respecta a la relación entre los modelos y los lenguajes de datos, hay que destacar que los modelos son la base para los lenguajes, aunque el nivel de abstracción de estos últimos es menor, ya que el lenguaje es el modelo más una sintaxis; por ejemplo, el lenguaje SQL es el resultado de aplicar una determinada sintaxis al modelo relacional.

9.14.3 Definición de modelo de datos

Las propiedades del UD son de dos tipos; estáticos, o relativamente invariantes en el tiempo, que responden a lo que se suele entender como estructura; y dinámicas, que son las operaciones que se aplican a los datos o valores almacenados en las estructuras, los cuales varían en el transcurso del tiempo al aplicárseles dichas operaciones.

Analicemos a continuación, con mayor detalle, cada uno de estos componentes.

9.14.4 Estática

La estática de un modelo está compuesta por:

Elementos permitidos

No son los mismos para todos los modelos de datos (varían especialmente en su terminología), pero en general son:

Objetos (entidades, relaciones, registros, etc.)

Asociaciones entre objetos (interrelaciones, “set”, etc.)

Propiedades o características de los objetos o de las asociaciones (atributos, campos, elementos de datos, etc.)

Dominios, conjuntos nominadas de valores sobre los que se definen las propiedades.

La representación de estos elementos depende de cada modelo de datos, pudiendo hacerse en forma de grafos (como en el modelo Jerárquico o Codasyl) o de tablas (como en el modelo Relacional), o bien en ambos (como en el modelo Entidad/Interrelación).

Elementos no permitidos o restricciones

No todos los valores, cambios de valor o estructuras están permitidos en el mundo real; por ejemplo, un niño de tres años no puede estar casado, ni una persona puede pasar directamente de soltera a viuda, etc. además, cada modelo de datos también impone por sí mismo limitaciones a las estructuras que admite; así, el modelo relacional no permite que dos filas de una tabla sean iguales. Estas limitaciones, que unas veces vienen impuestas por el mismo modelo de datos y que otras nos las impone el universo del discurso que estamos modelando, se denomina restricciones inherentes, y las que responden al deseo de que el sistema de información sea un reflejo la más fiel posible del mundo real son las restricciones de integridad o semánticas. Las restricciones inherentes son propias del modelo y, por tanto, varían de un modelo a otro; imponen rigidez a la hora de modelar, ya que no permiten describir ciertas estructuras. Por el contrario, las restricciones de integridad son facilidades que se ofrecen al diseñador a fin de que pueda representar en el esquema, lo más fielmente posible, la semántica de los datos. Las restricciones de integridad son fundamentales en el diseño de base de datos y las analizaremos más a fondo en este mismo unidad.

9.14.5 Dinámica

Los valores que toman los distintos de un esquema en un momento determinado t_i recibe el nombre de ocurrencia del esquema o estado de la base de datos en el tiempo tj (BDi); en otro momento tj la ocurrencia del esquema será (BDj). Si entre ti y tj se ha producido un cambio en algún valor de la base de datos (alta, baja o modficiación) BD i BD j . Tanto BD i como BD j deben ser ocurrencias válidas de la base de datos, es decir, ambas deben cumplir las restricciones de integridad así como las posibles restricciones asociadas al cambio de estado.

La componente dinámica del modelo consta de un conjunto de operadores que se definen sobre la estructura del correspondiente modelo de datos, ya que no todas las estructuras admiten el mismo tipo de operaciones. La aplicación de una operación a una ocurrencia de un esquema transforma a ésta en otra ocurrencia:

O (BDi) = BD j

(Pudiendo ser BD i = BD j), por ejemplo, en caso de consulta o cuando falla una operación por haberse producido un error).

9.14.6 Una operación tiene dos componentes:

Localización o “enfoque“ (también llamado selección), en localizar una ocurrencia de un objeto indicando un camino, o bien un conjunto de ocurrencias especificando una condición. En el primer caso se trata de un sistema navegacional, mientras que el segundo se dice que es de especificación.

Acción, que se realiza sobre la(s) ocurrencia(s) previamente localizada(s) mediante una operación de localización, y puede consistir en una recuperación o en una actualización (inserción, borrado o modificación).

Las distinción entre localización y acción es de tipo formal, y si bien algunos lenguajes, como el LMD de Codasyl, tienen dos mandatos distintos, uno para expresar la

selección y otro para la acción, distinguiendo claramente entre ambos tipos de operaciones, otros lenguajes como el SQL reúnen ambas operaciones de un único operador.

9.14.7 Plano conceptual para expresar una sentencia

Sin seguir una sintaxis concretas, sino más bien en un plano conceptual, podemos expresar una sentencia del LMD de la siguiente forma:

LOCALIZACION <condición>

ACCION <objetivo>

Donde LOCALIZACION y ACCION son mandatos del LMD; <condición> representa una expresión lógica proporcional por el usuario que deben cumplir los objetos que se desea localizar, o bien especifica el camino que indica el usuario para llegar a esos objetos, mientras que <objetivo> son los objetos (o las propiedades de éstos) sobre los que el usuario desea que se aplique la acción. Si el SGBD se adaptase estrictamente a la arquitectura a tres niveles de ANSI, el <objetivo> debería ser el nombre de un esquema externo previamente definido; sin embargo, algunos SGBD, especialmente los basados en el modelo relacional, no obligan a definir previamente el esquema externo, permitiendo describir el objetivo dentro de la misma sentencia de manipulación. Así, en la siguiente sentencia SQL:

SELECT Titulo, Autor

FROM LIBRO

WHERE Fecha_Edicion = “1996”

La localización y la acción –en este caso, recuperar- se expresan mediante un único mandato con el verbo inglés SELECT, el objetivo son las propiedades (atributos en el modelo relacional) Título y Autor del objeto (relación) LIBRO, y la condición es que la fecha de edición del libro sea igual a 1996. Con esta sentencia se recuperan (no sólo se localizan) el título y el autor de todos los libros editados en 1996; para lo cual no se ha hecho referencia a ningún esquema externo (vista), ya que las estructura objetivo que se desea recuperar (Título y Autor del LIBRO) se incluye en la misma sentencia.

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini

Editorial Rama, segunda edición. Capitulo 2: Concepto de Modelo de Datos Pag 83 -89.

9.15 Clase 15: Modelo de datos

LAS RESTRICCIONES DE INTEGRIDAD EN LOS MODELOS DE DATOS

CLASIFICACIÓN DE LOS MODELOS DE DATOS

NIVELES DE ABSTRACCIÓN DE LA ARQUITECTURA ANSI

CARACTERÍSTICAS DE LOS MODELOS CONVENCIONALES Y CONCEPTUALES

OBJETIVOS:

Identificar las Restricciones de Integridad en los Modelos de Datos

Clasificar los Modelos de Datos

Identificar las características de la arquitectura ANSI

9.15.1 Las restricciones de integridad en los modelos de datos

En el mundo real existen ciertas reglas que deben cumplir los elementos en él existentes; por ejemplo, un libro sólo puede tener un título, ser publicado por una única editorial y en una cierta fecha de edición; además, un libro sólo tiene un ISBN y un determinado ISBN correspondiente a un único libro, etc. cuando diseñamos una base deseamos que ésta refleje lo más fielmente posible el universo del discurso (una determinada biblioteca, una cierta universidad, una empresa específica, etc.), que estamos tratando de recoger en nuestro sistema de información, por lo que en el esquema de la base de datos, junto con los objetos, las asociaciones y las propiedades de los mismos, deberíamos describir también estas reglas, llamadas restricciones semánticas o de integridad, las cuales pueden ser definidas como condiciones que limitan el conjunto de ocurrencias válidas de un esquema.

Algunos autores, ver por ejemplo DATE (1993), prefieren emplear el término “reglas” en lugar de restricciones, reservando “restricción“ para la condición que se debe satisfacer; sin embargo, la denominación más habitual es la de restricción semántica, ya que la semántica y la integridad son conceptos que en las bases de datos suelen ir asociados, aunque no son idénticos. Con semántica nos referimos al significado de los datos y con integridad a la corrección de los mismos y a su consistencia respecto al mundo real del cual proceden. Cuando en el esquema de una base de datos se encuentra descrita la semántica del mundo real, será posible comprobar si los valores de los datos se atiende o no esta semántica del mundo real, será posible comprobar si los valores de los datos se atienen o no esta semántica previamente definida, comprobándose así la integridad de los datos; de ahí que digamos que ambos conceptos suelen ir unidos. Por tanto, nosotros utilizaremos indistintamente las expresiones restricciones semánticas o restricciones de integridad o, a veces, diremos simplemente restricciones cuando por el contexto se comprenda que nos estamos refiriendo a las restricciones inherentes.

La semántica de los datos, que en un principio se encontraba en la mente del usuario que era el que comprobaba manualmente si los datos cumplían o no las reglas a ellos asociados, fue migrando a los programas, y desde éstos a la base de datos, como se muestra en la figura 1.26

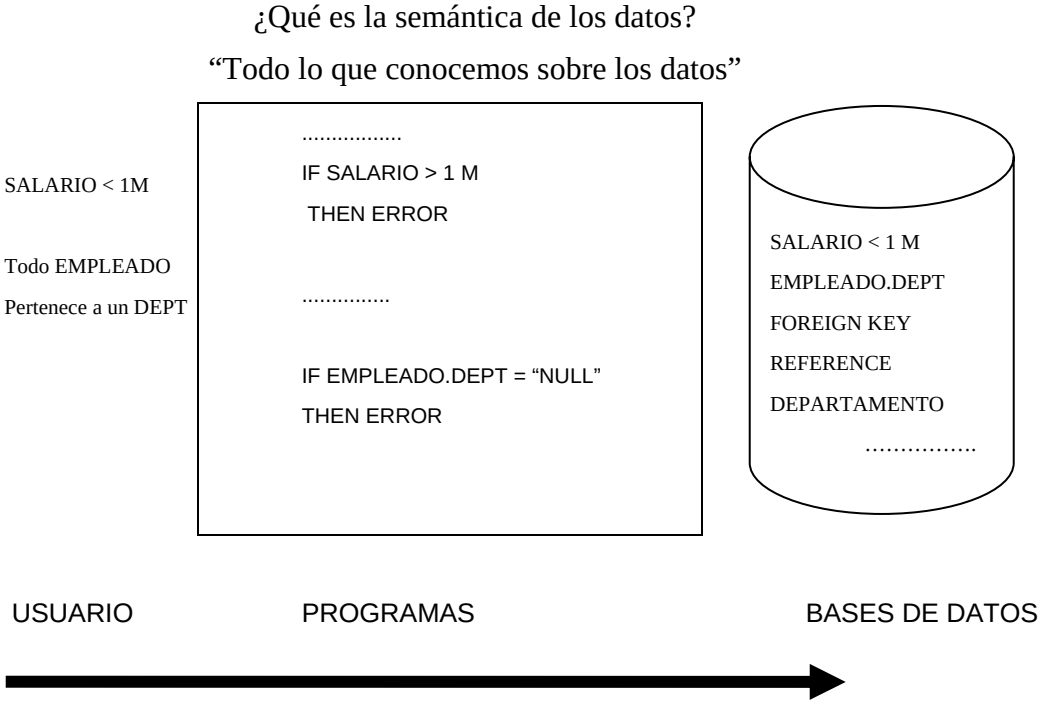


Figura 1.26 **Migración de la semántica de los datos**

Son muchas las ventajas de tener integridad la descripción de las restricciones con los datos en el esquema de la base de datos en lugar de que esté dispersa entre los diferentes programas de aplicación. Por un lado, tiene ventajas relativas a la integridad, ya que al ser única la descripción de las restricciones no se pueden producir inconsistencia debidas a que los distintos programadores hayan definido, cada uno en su programa y no de manera uniforme, las restricciones de integridad –por ejemplo, un programador se puede olvidar de incluir en su aplicación una determinada comprobación que otros sí han incluido- ;de esta forma puede, además, disminuirse drásticamente las cargas de programación (se considera que la programación de las sentencias necesarias para controlar la integridad puede llegar a suponer hasta un 90% del total de una aplicación). Por otro lado tiene también ventajas semánticas, ya que es importante que el significado de los datos, como parte fundamental de los mismos, se encuentre descrito junto con el resto de sus características, y que sea únicamente el diseñador el que se ocupe, por una sola vez, de definir la semántica, no dejando esta responsabilidad en manos de los programadores de aplicaciones; lo cual evita redundancia y permite a los usuarios, siempre que tengan la debida autorización, conocer el significado de los datos sólo con consultar el esquema de la base de datos en lugar de tener que indagar en las diferentes aplicaciones, tal como se muestra en la figura

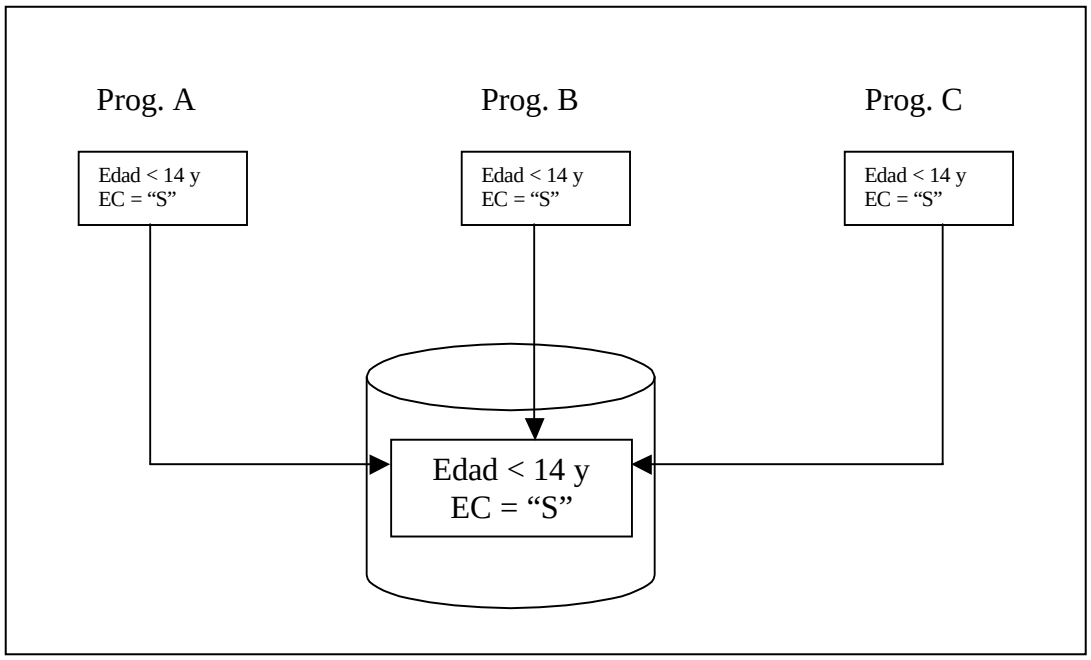


Figura 1.27 **Semántica de los datos "dispersa" entre los diferentes programas de aplicación, frente a semántica integrada en la base de datos**

Todas estas razones no muestran la necesidad de que la semántica del mundo real se encuentra descrita en el esquema, es decir, esté centralizada en lugar de dispersa en los diferentes programas de aplicación que actualizan la base de datos; pero para conseguir este objetivo, los modelos de datos han de permitir representar las restricciones semánticas dando instrumentos para ello, y los SGBD en los que están soportados los modelos tienen que reconocer y gestionar estas restricciones. Las restricciones semánticas que pueden ser especificadas en un determinado modelo de datos y representada en sus esquemas decimos que son restricciones semánticas reconocidas por el modelo. En el caso de que un determinado modelo (o el SGBD que lo implementa) no sea capaz de soportar cierto tipo de restricciones, será preciso, si se desea impedir que la base de datos pueda alcanzar estado inconsistentes con el mundo real que trata de representar, incluir estas restricciones en los programas de aplicación, con los inconvenientes que ello conlleva.

9.15.2 Clasificación de los modelos de datos

Durante la década de los sesenta y a principios de los ochenta, uno de los principales temas de discusión en el campo de las bases de datos giraba en tormo a los modelos de datos. El debate no se restringía en exclusiva a las ventajas de los distintos modelos de datos de cara a los usuarios y al sistema de información, así como a la eficiencia de los mismos en que lo que se refiere a la optimización de recursos físicos, sino que se extendía también a las posibilidades y conveniencias de una estandarización que definiera un único modelo capaz de satisfacer las necesidades de todos los usuarios.

9.15.3 Niveles de abstracción de la arquitectura ANSI

Con la aparición de los niveles de abstracción de la arquitectura ANSI, el tema se clarificó en parte al distinguirse los tres tipos de modelos a los que ya nos hemos referido anteriormente.

Modelos externos

Utilizados para la construcción de los esquemas externos (de cada usuario en particular), que persiguen satisfacer las necesidades de los usuarios (eficiencia humana).

Modelos conceptuales.

Utilizados en la elaboración del esquema conceptual los cuales buscan optimizar los recursos de información de la organización en su conjunto (eficiencia informativa).

Modelos internos.

Que sirven para construir el esquema físico o interno (eficiencia de los recursos informáticos).

Es el grupo ANSI / X3 /SPARC quien introdujo por primera vez en sus informes la expresión modelo conceptual pero sin llegar a definirlo, lo que ha dado lugar a distintas interpretaciones.

Basándonos en la arquitectura a tres niveles de ANSI, vamos a tratar de precisar una tipología de los modelos de datos reconociendo sus dificultades, dadas por las diferencias existentes entre los distintos autores, la falta de rigor con que, en general, se aplican los conceptos y la imprecisión y ambigüedad en su utilización y en la terminología.

En una primera clasificación, dividimos los modelos de datos en lógicos y físicos. La finalidad de los modelos lógicos es la representación de los aspectos lógicos de los diferentes tipos de datos existentes en el Universo del Discurso, reflejando su sentido semántico pero no las propiedades que respondan a características de tipo físico, a cuya definición están orientados los modelos físicos, llamados comúnmente internos. Los modelos lógicos, a su vez, los dividimos en externos y globales, según estén enfocados a describir los datos que necesitan cada usuario o aplicación en particular, o bien tengan como objetivo representar los datos de la organización en su conjunto. Dentro de los modelos globales se pueden distinguir los conceptuales y los convencionales. Aunque existe una gran disparidad de criterios respecto a la caracterización de ambos, en nuestra opinión, los modelos conceptuales poseen un mayor nivel de abstracción que los convencionales (sin embargo, en ocasiones, el modelo relacional se ha considerado un modelo conceptual, e incluso hace unos años algunos autores sostenían que el Codasyl podría ser un modelo conceptual adecuado). Por otro lado, mientras los modelos convencionales son modelos destinados a ser soportados por los SGBD, los modelos conceptuales tienen como finalidad la ayuda al diseño de base de datos en la fase de representación del universo del discurso, por lo que no suele ser habitual que se encuentren implementaciones en los SGBD. Existen, sin embargo, excepciones, y modelos conceptuales como el entidad/relación, CHEN (1976), o el modelo semántico de datos, HAMMER y McLEOD (1981) están soportados por algunos prototipos, e incluso empiezan a formar parte de algunos SGBD, por lo que al igual que los convencionales sirve, en este caso, de conexión (nivel de “mediación” en palabras de DATE (1993)) entre los esquemas

externos e internos. En cambio, los modelos conceptuales, y muy especialmente el entidad/interrelación, son la base de las herramientas de ayuda al diseño asistido por ordenadores (CASE).

En general, los modelos conceptuales, por su nivel de abstracción y riqueza semántica, constituyen una interfaz útil entre el informático y los usuarios finales en las primeras etapas del proceso de diseño de base de datos; mientras que los modelos convencionales se pueden considerar como interfaz entre el informático y el ordenador, apoyando al diseñador en etapas posteriores del proceso de diseño.

9.15.4 Características de los modelos convencionales y conceptuales.

CONVENCIONALES	CONCEPTUALES
-Implementados en SGBD comerciales	-No suele estar implementados en SGBD
-Depende del SGBD	-Independiente del SGBD
-Más próximo al ordenador	
-Poca capacidad semántica	-Mayor nivel de abstracción
-Más enfocados a la implementación	-Mayor capacidad semántica
-Interfaz informático / sistema	-Más enfocados al diseño de alto nivel (modelo conceptual)-
-Nivel de “mediación” entre el nivel externo e interno	-Interfaz usuario/informático

Tabla 1.5. Diferencia entre modelos convencionales y conceptuales

Los tres modelos de datos convencionales (Jerarquico, Codasyl, Relacional) difieren esencialmente en el modelo de representar las asociaciones entre entidades y en la forma de acceso a la base de datos.

La sencillez es una característica del modelo relacional que lo hace muy asequible a los usuarios; esta sencillez en la estructura también se pone de manifiesto en las operaciones de consulta y de actualización. Ahora bien, en contrapartida, el modelo Relacional ha tenido, hasta el momento y en ciertos aspectos, un menor contenido semántico que el modelo Codasyl, principalmente porque no permite distinguir entre objetos y asociaciones entre ellos; ambos se representan en el modelo relacional mediante una estructura única: la relación o tabla. El último estándar de ISO, el SGL92 (ISO (1992)), ha introducido importantes aspectos semánticos de los que carecía el anterior, y el que se está elaborando actualmente, el SQL3. ofrecerá interesantes mejoras semánticas.

En cuanto al tema de la independencia física/lógica, el modelo Codasyl, y aún más el Jerárquico, presenta una correspondencia directa entre las relaciones lógicas y los caminos de acceso físicos, lo que dificulta la independencia entre los niveles lógicos y

físicos y, además, trae como consecuencia que los lenguajes asociados sean de tipo navegacional. En el modelo Relacional, sin embargo, el acceso a los datos se realiza en función de las propiedades de éstos, y el usuario no conoce los caminos de acceso, ya que es el sistema el que se ocupa de seleccionar el camino físico optimizando los recursos y el tiempo de respuesta.

Cabe destacar que el rendimiento ha sido un argumento fundamental para los que defendían la supremacía de los sistemas tipo red. Sin embargo, CODD (1990) y DATE (1986) siempre han sostenido que no había razones objetivas para que los sistemas relacionales no fuesen tan eficientes como los basados en otros modelos; el rendimiento de las actuales versiones de algunos SGBDR parece que viene a dar la razón a estos autores.

Cada uno de estos modelos tiene ventajas e inconveniente, lo que originó en ciertos momentos posturas irreconciliables entre los que defienden uno u otro. Vistas estas posturas desde la perspectiva actual, nos encontramos con que, aun cuando no se ha llegado a una solución unánimemente aceptada, varios de los SGBD comerciales no relacionales (como IDMS, Adabas, Datacom/DB) ofrecen algún tipo de interfaz relacional, y que otros constructores (como IBM o Unisys) que tenían sistemas Jerárquico o Codasyl ofrecen también hoy productos relacionales como algún tipo de interfaz con el sistema antiguo a fin de paliar los costes y trastornos que supone el paso de SGBD a otro basado en un modelo de datos distintos. De todos modos, en la actualidad, el modelo Relacional se ha impuesto y las pasarelas entre los sistemas anteriores y los relacionales ya no es un tema crucial como podía serlo, para algunos hace unos años.

Posiblemente una de las razones por la que la discusión sobre los modelos de datos, tal como estaban planteada en un principio, ha perdido todo interés práctico, es la aceptación, bastante extendida, de que distintos usuarios necesitan distintas formas de ver los datos. También ha sido determinante en el cambio de actitud la difusión comercial de muchos SGBD relacionales que han mejorado drásticamente la eficiencia de sus instrumentaciones (los avances en los optimizadores han tenido mucho que ver en el tema). El problema de la eficiencia era crucial en las primeras versiones de los sistemas relacionales y constituía el principal origen de todas las críticas que se vertían sobre él.

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición, Capitulo 3: Concepto de Modelo de Datos Pag 89- 95

9.16 Clase 16: Diseño de BBDD

MECANISMOS DE ABSTRACCIÓN DE LOS MODELOS DE DATOS PARA TRABAJAR

ABSTRACCIÓN

CLASIFICACIÓN, AGREGACIÓN, GENERALIZACIÓN

OBJETIVOS:

Definir los mecanismos de abstracción de los modelos de datos

9.16.1 Mecanismos de abstracción de los Modelos de Datos para trabajar:

- Clasificación
- Agregación
- Generalización

9.16.2 Abstracción

Mecanismos de Abstracción

- Sirven para convertir la riqueza del mundo real en un representación de la que admiten los ordenadores; Más pobre y más organizada
- Definición: Proceso mental que no permite seleccionar propiedades comunes y/o diferenciadoras entre un conjunto de objetos para relacionarlos

Ejemplo:

Pintar un bicicleta

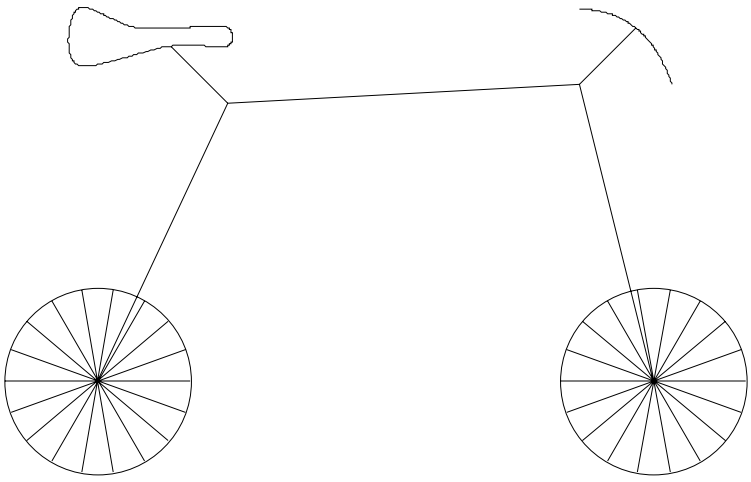


Figura 1.28 **Abstracción de un objeto**

Pero:

- El dibujo no es una bicicleta, pero la representa, representa el concepto de bicicleta
- Hacemos abstracción al fijarnos en ruedas, cuadro, sillín, manillar
- Hacemos abstracción al olvidar cadena, pedales, proporciones, etc

Los tres tipos de abstracción, estás fuertemente influenciados por la OO y son:

Clasificación. (ES_MIEMBRO_DE) no confundir con (ES_UN)

Significa agrupar una serie de elementos individuales en una clase

Agregación. (ES_PARTE_DE) es una composición

Formación de una clase a partir de otras clases

Generalización. (ES_UN)

Una clase más abstracta engloba otras que son especializaciones de esta

9.16.3 Clasificación, agregación, generalización

Clasificación

Sirve para definir un concepto que agrupa las propiedades comunes de una serie de objetos del mundo real

Ejemplo:

(Enero, Febrero, ... , Diciembre) son MESES

Olvidamos alguna característica como el número de días distintos, capturamos el concepto que los une, "ser meses"

Representación: la clase subrayada y los objetos unidos con flechas discontinuas a ella

Ejemplo:

Figura 1.29 **Ejemplo de Clasificación**

9.16.4 Ejemplo de pertenencia múltiple:

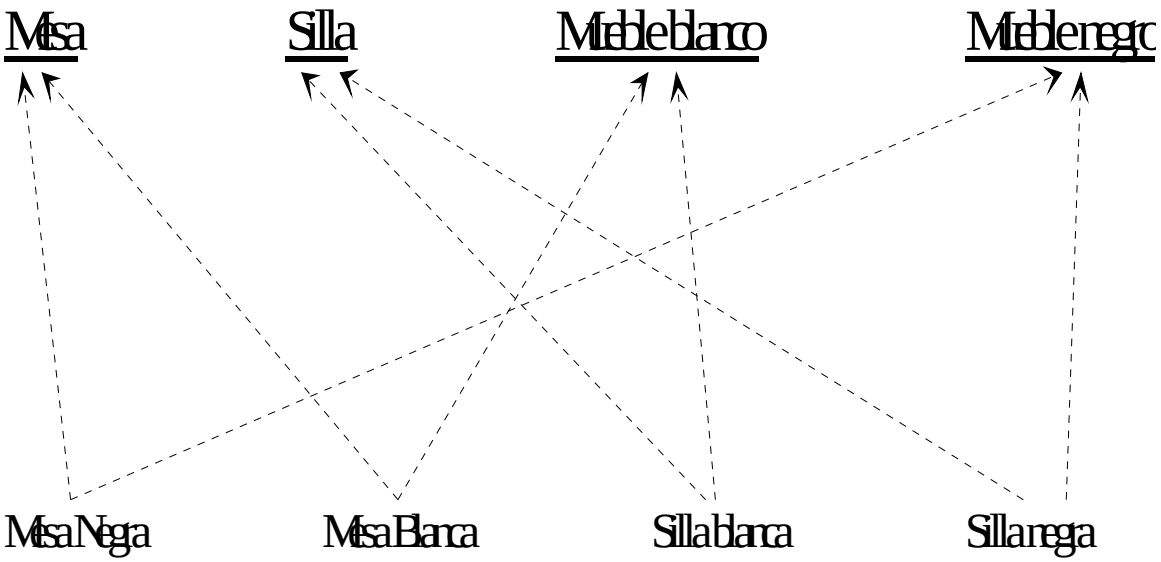


Figura 1.30 **Ejemplo de Pertenencia Múltiple**

Agregación

Permite definir una clase a partir de un conjunto de clases que representan sus partes componentes

Ejemplo:

Tenemos las clases Pedal, sillín, rueda y las unimos en la clase Bicicleta

Tenemos las clases Nombre, sexo, salario y las unimos en la clase Persona

Ejemplo: Ejemplo de agregación especial:

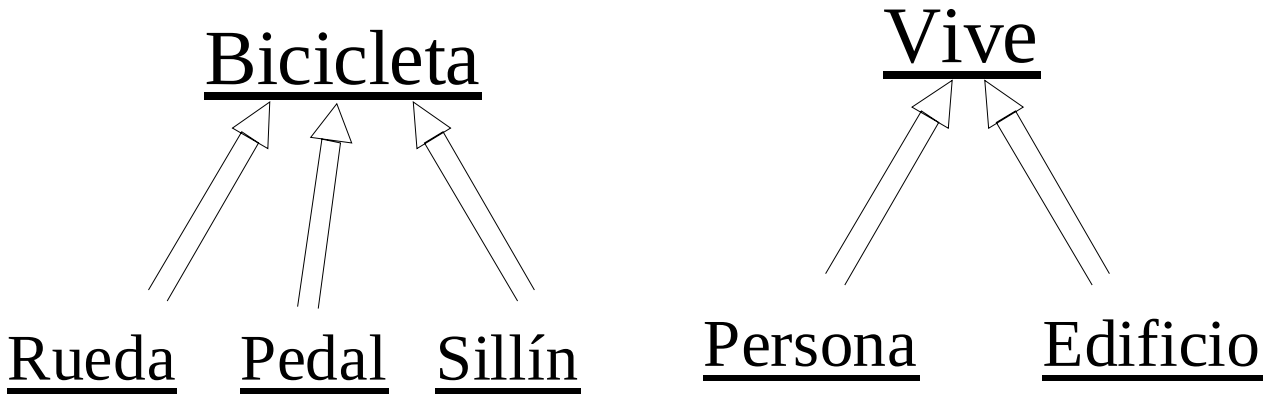


Figura 1.31 **Ejemplo de Agregación Especial**

Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez, Tema 2 Diseño de Bases de Datos paginas 9 a 12

9.17 Clase 17: DISEÑO DE BBDD

GENERALIZACIÓN

CARDINALIDAD MÍNIMA **Y** MÁXIMA

OBJETIVOS:

Identificar la generalización en las entidades

Identificar la cardinalidad mínima y máxima en los modelos de datos

Generalización

Crea una relación de subordinación del tipo "es un subconjunto" entre varias clases. Es la abstracción más intuitiva

Representación:

clases subrayadas, unidas con flechas continuas simples haciendo ángulos rectos cuando sea necesario

Ejemplo:

Figura 1.32 Ejemplo de Generalización

Esta abstracción se puede leer en dos direcciones

De las abstracciones anteriores teníamos que Persona es una agregación de nombre, sexo y edad que son clasificaciones de conceptos

Ahora vemos que Mujer y Hombre son especializaciones de Persona, y por tanto están formadas por los conceptos de Persona y algunos otros conceptos, o solamente son una partición de las personas en dos grupos

¿Son las siguientes abstracciones iguales?

Persona Sexo

Hombre Mujer Hombre Mujer

Resumen

- Clasificación identifica los tipos de los registros
- Agregación establece atributos o campos

- Ejemplo que ilustra esta relación entre ambos:

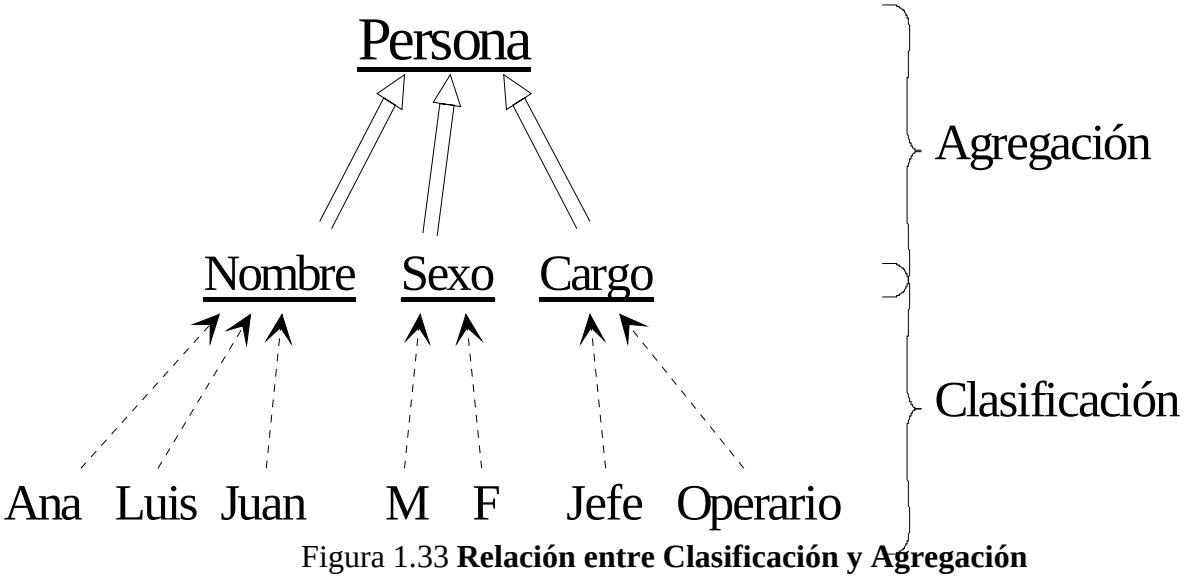


Figura 1.33 **Relación entre Clasificación y Agregación**

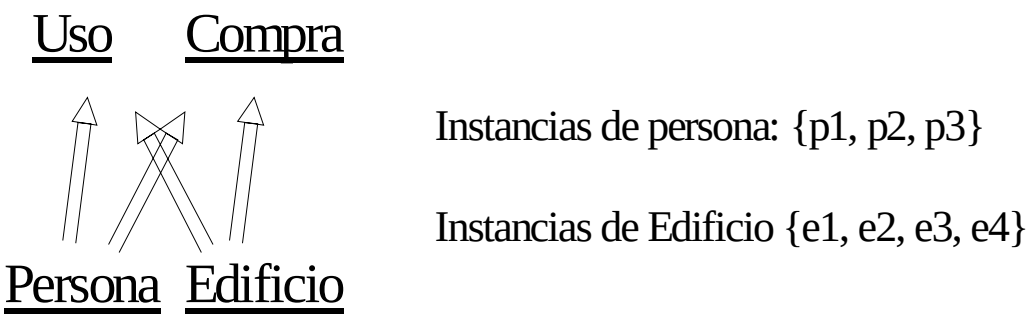
Las tres abstracciones son independiente

Ninguna puede ser definida en base a las otra

Son tres conceptos matemáticos distinto

- Clasificación = Pertenencia a conjunto = (ES_MIEMBRO_DE)
- Agregación = Composición de conjuntos = (ES_PARTE_DE)
- Generalización = Subconjuntos = (ES_UN)
- Agregación y generalización establecen relaciones entre clases y la clasificación entre instancias y clases
- Estas relaciones pueden tener una cierta cardinalidad
- Veamos un ejemplo

Ejemplo:



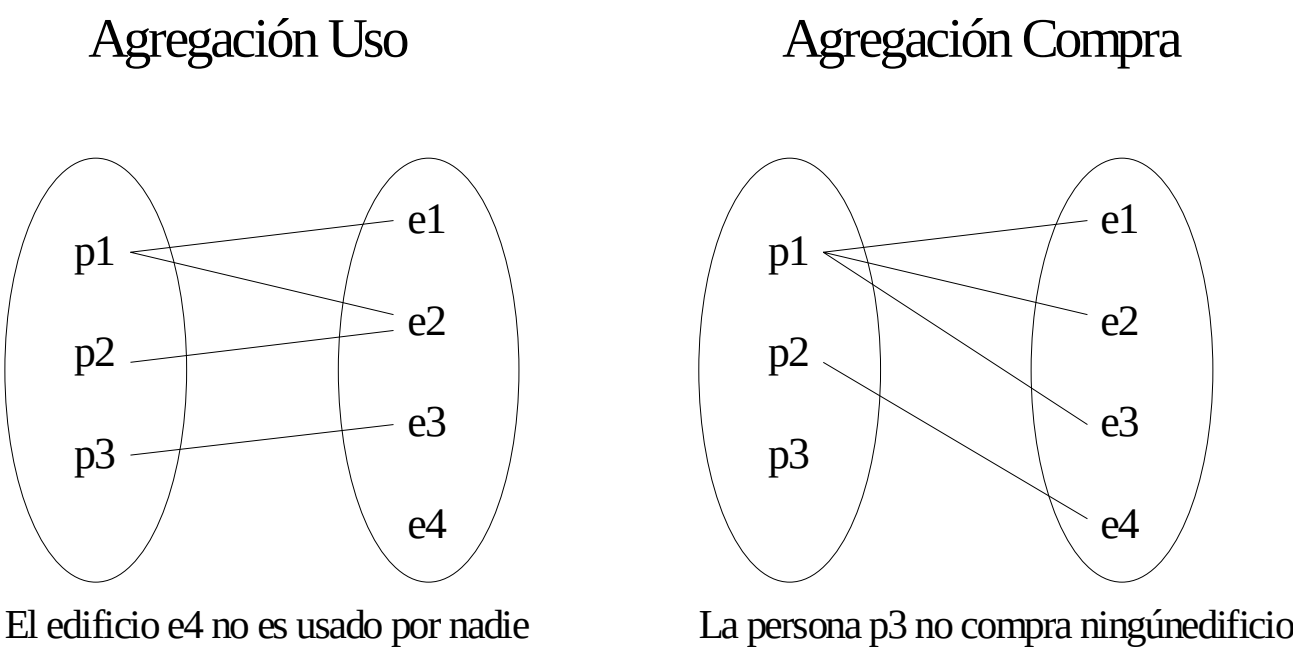


Figura 1.34 **Ejemplo de Introducción a la Cardinalidad**

Para este ejemplo son los siguientes:

Cardinalidad mínima:

- Cada persona como mínimo usa un edificio 1
- Un edificio puede estar deshabitado 0
- Existen personas que no compran edificios 0
- Todo edificio lo compró alguien 1

Cardinalidad máxima:

- Una persona puede usar varios edificios n
- Un edificio puede ser usado por varias personas n

- Una persona puede comprar varios edificios n
- Una edificio solo puede ser comprado una vez 1. (en este caso)

La cardinalidad mínima 0 indica que es optativo y 1 obligatorio

Esto resulta en las siguientes cardinalidades:

- PERSONA-COMPRA: (0, n)
- PERSONA-USA: (1, n)
- EDIFICIO-USA: (0, n)
- EDIFICIO-COMPRA: (1, 1)
- Esto será muy útil en el modelo E/R

Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez. Tema 2 Diseño de Bases de Datos paginas 12 a 15

9.18 Clase 18: Modelo entidad – relación

MODELO ENTIDAD – RELACIÓN

ENTIDADES Y CONJUNTO DE ENTIDADES

ATRIBUTOS

OBJETIVOS:

Definir el Concepto de Modelo Entidad – Relación

Definir el Concepto de Entidades y Conjunto de Entidades

Identificar las Relaciones y Conjuntos de Relaciones

Definir el Concepto de Atributos

9.18.1 Modelo entidad – relación

El modelo de datos entidad – relación (E-R) se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificaciones de un esquema empresarial. Este esquema representa la estructura lógica global de la base de datos.

Este modelo surge por la necesidad de plasmar la información que debe ir en la base de datos y para plasmar la especificación de información que necesita una empresa.

El modelo Entidad - Relación pertenece al grupo de modelado semántico que pretende representar el significado de los datos. El modelo Entidad - Relación sirve para crear el esquema conceptual.

Es el más extendido de los modelos de datos para describir esquemas conceptuales. El hecho de poder establecer relaciones simultáneas entre varias entidades por medio de herramientas gráficas permite la representación de esquemas de fácil lectura incluso para aquellos sistemas con relaciones complejas. El Modelo Entidad-Relación es una técnica especial de representación gráfica que incorpora información relativa a los datos y la relación existente entre ellos, para darnos una visión del mundo real. Las características del modelo E-R son:

- Reflejan tan sólo la existencia de los datos, no lo que se hace con ellos.
- Se incluyen todos los datos del sistema en estudio y, por tanto no es orientado a aplicaciones particulares.
- Es independiente de las base de datos y sistemas operativos concretos.
- No se tienen en cuenta restricciones de espacio, almacenamiento ni tiempo de ejecución.
- Está abierto a la evolución del sistema.

Por tanto, en el modelo E-R se da una visión del mundo real con la mayor naturalidad mediante los objetos y sus relaciones.

Como ya se indicó anteriormente, la representación de datos implica la construcción de un modelo que pueda reflejar el mundo de la realidad dentro del mundo de la máquina a través de nuestras ideas y de los conocimientos del mundo de la información. El modelo E-R, por tanto, se basa en la percepción de un mundo real que consiste en un conjunto de objetos básicos denominados Entidades, así como las Relaciones existentes entre ellos.

9.18.2 Entidades y conjunto de entidades

Una entidad es un objeto que existe y es distinguible de otros objetos. Por ejemplo Juan Perez, con número de seguro social 890-12-3456, es una entidad, ya que identifica únicamente una persona específica en el universo. Análogamente, el número de cuenta 104 en la sucursal del Banco de América es una entidad que identifica únicamente una cuenta determinada. Una entidad puede ser concreta, tal como una persona o un libro, o puede ser abstracta, como un día festivo o un concepto.

Un conjunto de entidades es un conjunto de entidades del mismo tipo. El conjunto de todas las personas que tienen una cuenta en un banco, por ejemplo, puede definirse como el conjunto de entidades cliente. Análogamente, el conjunto de entidades cuenta podría representar el conjunto de todas las cuentas en un banco determinado.

Los conjuntos de entidades no necesitan ser disjuntos. Por ejemplo, es posible definir el conjunto de entidades de todos los clientes del banco (cliente). Una entidad persona puede ser una entidad empleado, una entidad cliente, ambas o ninguna de las dos.

Una entidad está representada por un conjunto de atributos. Posiblemente atributos del conjunto de entidades cliente son nombre-cliente, seguridad-social y ciudad-cliente. Posibles atributos del conjunto de entidades cuenta son número-cuenta y saldo. Para cada atributo hay un conjunto de valores permitidos, llamados dominio de ese atributo. El dominio del atributo nombre-cliente podría ser el conjunto de todas las cadenas de texto de una determinada longitud. Análogamente, el dominio del atributo número-cuenta podría ser el conjunto de todos los enteros positivos.

Formalmente, un atributo es una función que asigna un conjunto de entidades a un dominio. Así, cada entidad se describe por medio de un conjunto de pares (atributos, valor del dato), un par para cada atributo del conjunto de entidades. Una entidad cliente determinada se describe por medio del conjunto [(nombre, Juan), (seguridad-social, 890-12-3456), (calle, del encanto), (ciudad, León)], lo cual significa que la entidad describe una persona llamada Juan con número de seguridad social 890-12-3456, que reside en la calle del encanto.

El concepto de un conjunto de entidades correspondiente a la noción de definición de tipo en un lenguaje de programación. Una variable de un tipo dado tiene un valor determinado en un instante de tiempo dado. Así, una variable en los lenguajes de programación corresponde al concepto de una entidad en el modelo E-R.

Por tanto, una base de datos incluye una colección de conjuntos de entidades cada uno de los cuales contiene un número cualquiera de entidades del mismo tipo ejemplos de nombres de entidades y su descripción.

Sucursal, el conjunto de todas las sucursales de un banco determinado. Cada sucursal se describe por los atributos nombre-sucursal, ciudad-sucursal y activo.

Cliente, el conjunto de todas las personas que tienen una cuenta en el banco. Cada cliente se describe por los atributos nombre-cliente, seguridad-social, calle y ciudad-cliente.

Empleado, el conjunto de todas las personas que trabajan en el banco. Cada empleado se describe por los atributos nombre-empelado, número-teléfono.

Cuenta, el conjunto de todas las cuentas que mantiene en el banco. Cada cuenta se describe por los atributos número-cuenta y saldo.

Transacción, el conjunto de todas las transacciones de cuentas ejecutadas en el banco. Cada transacción se describe por los atributos número-transacción, fecha, y cantidad.

Ejemplo: Conjunto de entidades “Cliente” y “Prestamo_Bancario”

Santos	3569852	C/ Amargura	C
Gómez	4251589	Rúa Urzáiz	VIGO
López	1475842	Beiramar	PO
Ruiz	3656289	Rúa da Paz	OU

P-15	2.000.000
P-55	150.000
P-07	125.000
P-13	800.000

9.18.3 Atributos

Una entidad es un conjunto de ítems de datos o Atributos, cada uno de los cuales juega un papel en la definición de una entidad. Un atributo es una unidad básica e indivisible de información acerca de una entidad que sirve para identificarla o describirla.

Puesto que la noción de conjunto de entidades y conjunto de relaciones no es precisa, es posible definir un conjunto de entidades y sus relaciones de varias formas diferentes. La principal diferencia está en la forma en que tratamos los diversos atributos. Considérese el conjunto de entidades. Empleado con atributos nombre-empleado y número-teléfono. Se puede argumentar fácilmente que un teléfono es una entidad en si mismo con atributos número-teléfono y situación (la oficina donde está colocado el teléfono). Si tomamos este punto de vista, el conjunto de entidades empleado debe redefinirse como sigue:

-El conjunto de entidades empleado con atributos nombre-empleado.

-El conjunto de entidades teléfono con atributo número-teléfono y situación.

-El conjunto de relaciones EmplTelf, que indica la asociación entre los empleados y los teléfonos que tienen.

¿Cuál es, entonces, la principal diferencia entre estas dos definiciones de empleados? En el primer caso, la definición implica que cada empleado tiene exactamente un número de teléfono asociado. En el segundo caso, sin embargo, la definición afirma que los empleados pueden tener varios números de teléfonos asociados (incluyendo el cero). Así, la segunda definición es más general que la primera, y puede reflejar con más precisión la situación del mundo real.

Aun si nos dicen que cada empleado tiene exactamente un número de teléfono asociado, la segunda definición puede ser todavía más apropiada si el teléfono se comparte entre varios empleados.

No sería apropiado, sin embargo, aplicar la misma técnica al atributo nombre-empleado. Esto se debe a que es difícil argumentar que nombre-empleado es una entidad en si misma (a diferencia de teléfono). Así es adecuado tener nombre-empleado como un atributo del conjunto de entidades empleado.

Surge entonces una pregunta natural: ¿Qué constituye un atributo, y qué constituye un conjunto de entidades? Desafortunadamente, no hay una respuesta sencilla. La distinción depende principalmente de la estructura de la empresa que se esté modelando y de la semántica asociada con el atributo en cuestión.

Formalmente un atributo de un conjunto de entidades es una función que asigna al conjunto de entidades un dominio. Por tanto, una entidad se puede describir como un conjunto de pares atributo – valor, uno por cada atributo.

Ejemplo:

CLIENTE: {(Nombre, Santos), (DNI, 3569852), (Calle, El Calvario), (Ciudad, Telica)

En el modelo Entidad - Relación existen distintos modelos de atributos:

- Simples y compuestos.

- Univalorados y multivalorados.
- Nulos.
- Derivados.

9.18.4 Atributos simples y compuestos

.- Los atributos simples son aquellos que no tienen capacidad de ser descompuestos, mientras que los atributos compuestos son aquellos que permiten descomponerse en otros atributos conformando lo que se denominan jerarquías. Un ejemplo de atributo simple podría ser la edad, mientras que uno compuesto podría ser la dirección de un cliente:

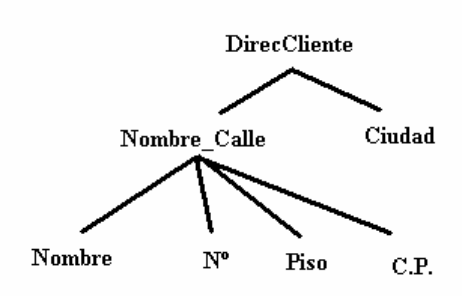


Figura 1.35 **Atributos Simples y Compuestos**

Es conveniente usar atributos compuestos porque facilitan el diseño haciéndolo más claro y sencillo.

Atributos univalorados y multivalorados.

Los atributos univalorados son aquellos que sólo pueden tomar un valor (NombreCliente), mientras que los atributos multivalorados pueden tomar más de un valor para un mismo atributo (DirecCliente, NumTlf). Hay que marcar siempre un límite inferior y un límite superior.

Atributos nulos

.- Un atributo es nulo cuando para una determinada entidad ese atributo no tiene valor. Se entiende nulo como desconocido. Un atributo nulo es no conocido cuando no se ha introducido un valor en el campo correspondiente, y está perdido cuando se produce un error en la base de datos y el valor se pierde o se corrompe.

Atributos derivados

El valor para este atributo se puede derivar de los valores de otros atributos ó entidades. Por ejemplo, si existe un campo fecha_nacimiento, la edad sería un atributo derivado

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición, Capitulo 3: Concepto de Modelo de Datos Pag 96- 98

9.19 Clase 19: Modelo entidad – relación

EJEMPLO DE ENTIDADES Y ATRIBUTOS

OBJETIVOS:

Identificar la representación de las entidades y atributos

Ejemplo 4.1

Supongamos que queremos informatizar los datos de la Universidad UNAN - LEON. En este caso, algunas de las entidades a tener en cuenta serían las siguientes:

PROFESOR	Almacenará información sobre los profesores que imparten docencia en la Universidad
ALUMNO	Contendrá datos de los alumnos que se han matriculado en la Universidad
P.A.S.	Los datos de personal de administración y servicios se almacenarán en esta entidad
MATRICULA	Contendrá información de las matrículas realizadas por los alumnos
NOMINA	Almacenará las nóminas del personal contratado por la Universidad

Éstas son únicamente algunas de las entidades que formarían parte del sistema de información de la Universidad. En la realidad, son muchas más: horarios, actividades, deportes etc.

Ejemplo 4.2

Si se desea diseñar un sistema de gestión de información en un vídeoclub, las entidades principales a considerar serían:

SOCIO	Los datos de los socios del vídeoclub quedaría reflejada en esta entidad
PELICULA	Almacenaría la información de las películas disponibles en el vídeoclub

Ejemplo 4.3

Entre las entidades utilizadas para informatizar un taller de automóviles podemos destacar:

AUTOMOVIL	Almacenará información sobre los vehículos que son reparados en el taller
MECANICO	Contendrá datos de los mecánicos de los que dispone el taller

Ejemplo 4.4

Para la entidad PROFESOR, algunos atributos que identifiquen a la entidad podrían ser los siguientes:

INSS	Número de la seguridad social
DNI	DNI
NOMBRE	Nombre completo
DPTO	Departamento al que pertenece
AREA	Área de conocimiento a la que está inscrito
CATEGORIA	Categoría (Catedrático, Titular, Ayudante, Asociado)
DESPACHO	Número del despacho en el que habitualmente trabaja

Entidad PROFESOR

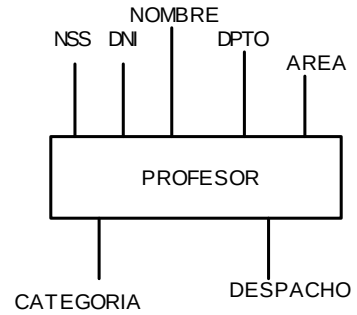


Figura 1.36 **Ejemplo de Atributos de Profesor**

Ejemplo 4.5

Atributos para la entidad PELICULA:

COD	Código asignado a la película por el videoclub
TITULO	Título
DURACIÓN	Duración en minutos
PRECIO	Precio por el que se alquila
TIPO	Tipo (Drama, Comedia, Wester, Intriga,...)
PAIS	Nacionalidad de la película
AÑO	Año en el que apareció

Entidad PELICULA

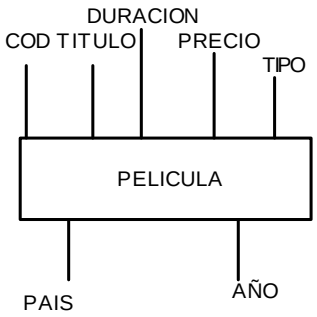


Figura 1.37 **Ejemplo de Atributos de Película**

Ejemplo 4.6

Atributos para la entidad AUTOMOVIL:

MATRICULA	Número de matrícula
MARCA	Marca (Seat, Opel, Renault,...)
MODELO	Modelo (Marbella, Corsa, Clio,...)
COLOR	Color
CV	Caballos / Potencia
KM	Kilómetros
SEGURO	Número de la póliza de seguros
DNI	DNI del dueño del automóvil
F_COMPRA	Fecha de compra

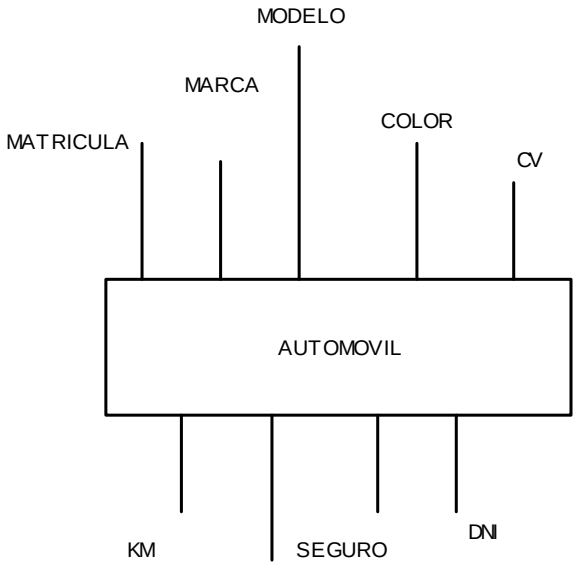


Figura 1.38 **Ejemplo de Atributos de Automóvil**

9.20 Clase 20: Modelo entidad – relación

RESTRICCIONES DE ASIGNACIÓN (MAPPING)

DEPENDENCIAS DE EXISTENCIAS

OBJETIVOS:

Identificar las Restricciones de Asignación (Mapping)

Definir el concepto de dependencia de existencias

9.20.1 Restricciones de asignación (MAPPING)

Una planificación E-R de una empresa puede definir ciertas restricciones a las cuales deben ajustarse los contenidos de una base de datos. Una restricción importante es la de las cardinalidades de asignación, que expresan el número de entidades con las que puede asociarse otra entidad mediante un conjunto de relaciones.

Las cardinalidades de asignación son más útiles al describir conjuntos binarios de relaciones, aunque ocasionalmente contribuyen a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades. En esta unidad nos concentraremos sólo en conjunto binarios de relaciones.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de asignación debe ser una de las siguientes:

-**Una a una**. Una entidad en A está asociada a lo sumo con una entidad en B, y una entidad en B está asociada a lo sumo con una entidad en A. (véase figura 1.39)

-**Una a muchas**. Una entidad en A está asociada con un número cualquiera de entidades en B. Una entidad en B, sin embargo, puede estar asociada a lo sumo con una entidad en A. (Véase la figura 1.40)

-**Muchas a muchas**. Una entidad en A está asociada con un número cualquiera de entidades en B. Y una entidad en B, esta asociada con un número de cualquiera de entidad en A. (Véase la figura 1.41)

La cardinalidad de asignación adecuada para un conjunto de relaciones determinado obviamente es dependiente del mundo real que el conjunto de relaciones está modelando.

Para ilustrar lo anterior, considérese el conjunto de relaciones CtaCli. Sí, en un banco dado una cuenta puede pertenecer únicamente

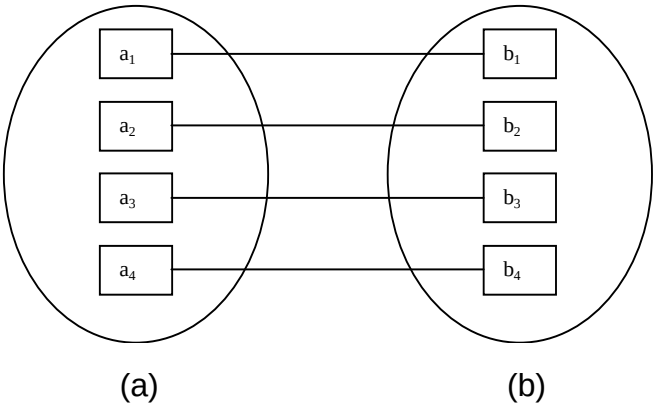


Figura 1.39 **Relación de una a una**

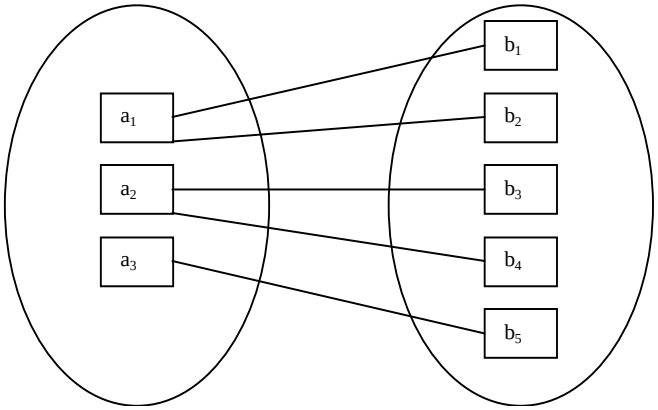


Figura 1.40 **Relación de una a muchas**

A un cliente, y un cliente puede tener varias cuentas, entonces el conjunto de relaciones es de una a muchas de clientes a cuentas. Si una cuenta puede pertenecer a varios clientes (como en cuentas comunes que tienen varios miembros de la familia), el conjunto de relaciones es de muchas a muchas.

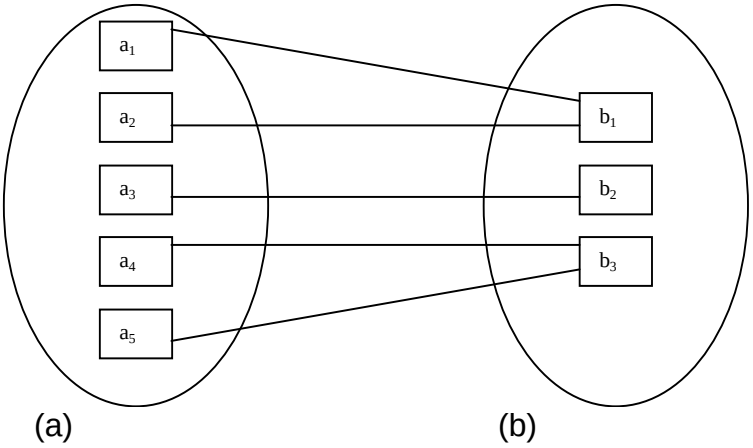


Figura 1.41 **Relación de muchas a una.**

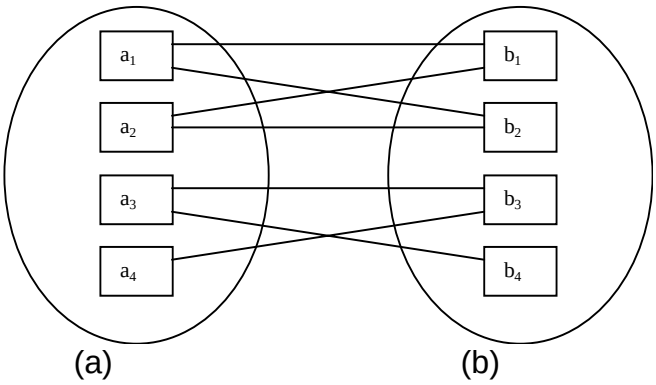


Figura 1.42 **Relación de muchas a muchas.**

9.20.2 Dependencias de existencias

Constituyen otra clase importante de restricciones. Especificaciones, si la existencia de entidades “x” depende de la existencia de la entidad “y”, entonces se dice que es dependiente por existencia de “y”. Operativamente, esto significa que si se suprime “y”, también se suprime “x”. La entidad “y” se dice que es una entidad dominante y “x” se dice que es una entidad subordinada.

Para ilustrar lo anterior, considérense los conjuntos de entidades cuenta y transacción. Formamos un conjunto de relaciones bitácora entre estos dos conjuntos que especifican que para una cuenta determinada puede haber varias transacciones. Este conjunto de relaciones es de una a muchas de cuenta a transacción. Cada entidad transacción debe estar asociada con una entidad cuenta. Si se suprime una entidad cuenta, entonces todas sus entidades transacción asociada también deben suprimirse. Por el contrario, las entidades transacción pueden suprimirse de la base de datos sin afectar a ninguna cuenta. El conjunto de entidades cuenta, por tanto, es dominante y transacción es subordinado en el conjunto de relaciones bitácora.

Bibliografía comentada

Fundamentos de BASES de DATOS, Henry F. Kortth, Abraham Silberschatz. Editorial McGrawHill, segunda edición, Capitulo 2: Modelo entidad - relación Pag 30 – 36

9.21 Clase 21: Modelo entidad – relación

CLAVES

SUPERCLAVE

CLAVES CANDIDATAS.

CLAVE PRIMARIA

EJEMPLOS DE CLAVES

OBJETIVOS:

Definir el concepto de claves

Diferenciar una superclave de una claves candidatas y de claves primarias del conjunto de atributos

9.21.1 Claves

Es importante poder especificar cómo se distinguen las entidades y las relaciones por lo tanto recordemos. Conceptualmente, las entidades individuales y las relaciones son distintas, pero desde la perspectiva de una base de datos, la diferencia entre ellas debe expresarse en términos de sus atributos. El concepto de superclave nos permite hacer tales distinciones.

Superclave

Una es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única a una entidad en el conjunto de entidades. Por ejemplo, el atributo seguridad-social, del conjunto de entidades cliente, es suficiente para distinguir una entidad cliente de otra. Así, seguridad-social es una superclave. Análogamente, la combinación de nombre-cliente y seguridad social es una superclave para el conjunto de entidades cliente. El atributo nombre-cliente de cliente no es una superclave, ya que varias personas podrían tener el mismo nombre.

El concepto de superclave no es suficiente para nuestros propósitos, ya que, como vivimos arriba, una superclave puede contener atributos ajenos. Si k es una superclave, entonces también lo será cualquier super conjunto K. A menudo estamos interesados en superclaves para las cuales ningún subconjunto propio es superclave. Dichas superclaves mínimas se llaman claves candidatas.

Es posible que varios conjuntos de atributos distintos pudieran servir como claves candidatas. Supóngase que una combinación de nombre-cliente y calle es suficiente para distinguir entre los miembros del conjunto de entidades clientes. Entonces {seguridad-social} y {nombre-cliente, calle} son claves candidatas. Aunque los atributos seguridad-social y nombre-cliente juntos pueden distinguir entidades cliente, sus combinaciones no forman una clave candidata, ya que el atributo seguridad-social por si solo es una clave candidata.

Clave primaria

Usaremos el término clave primaria para denotar una clave candidata que elige el diseñador de la base de datos como el medio principal de identificar entidades dentro de un conjunto de entidades.

Es posible que un conjunto de entidades no tenga atributos suficientes para formar una clave primaria. Un conjunto de entidades de este tipo se denomina conjunto de entidades débil. Un conjunto de entidades que tiene una clave primaria se denomina conjunto de entidades fuertes. Para ilustrarlo, considérese el conjunto de entidades transacción, que tiene tres atributos número-transacción, fecha y cuenta. Aunque cada entidad transacción es distinta, las transacciones en cuenta diferentes pueden compartir el mismo número de transacción. Así, este conjunto de entidades no tiene una clave primaria y, por tanto, es un conjunto de entidades débil. Para que un conjunto de entidades débil sea significativo, debe ser parte de un conjunto de relaciones una a muchas (mas adelante explicaremos en detalles). Este conjunto de relaciones no debe tener atributos descriptivos, ya que cualquier atributo que se necesite puede estar asociado con el conjunto de entidades débil.

Los conceptos de conjuntos de entidades débiles y fuertes están relacionados con las dependencias por existencia. Un miembro de un conjunto de entidades fuerte es, por definición, una entidad dominante, mientras que un miembro de un conjunto de entidades débil es una entidad subordinada.

Aunque un conjunto de entidades débil no tiene una clave primaria, sin embargo debe haber un medio de distinguir entre todas aquellas entidades en el conjunto de entidades que dependen de una entidad fuerte determinada.

El discriminante de un conjunto de entidades débil es un conjunto de atributos que permiten que se haga esta distinción. Por ejemplo, el discriminante del conjunto de entidades débil transacción es el atributo número-transacción, ya que para cada cuenta un número de transacción identifica de forma única una transacción.

La clave primaria de un conjunto de entidades débil está formada por la clave primaria del conjunto de entidades fuerte de la que dependen su existencia y su discriminador. En el caso del conjunto de entidades transacción, su clave primaria es {número-cuenta, número-transacción}, donde número-cuenta identifica la entidad dominante de una transacción y número-transacción distingue entidades transacciones dentro de la misma cuenta.

La clave primaria de un conjunto de entidades nos permite distinguir entre las diversas entidades del conjunto. Necesitamos un mecanismo parecido para distinguir entre las diversas relaciones de un conjunto de relaciones. Para hacer esto, se debe explicar primero cómo se describen las relaciones individuales. Una vez conseguido, podemos explicar cómo se define una clave primaria para un conjunto de relaciones.

Las claves tienen que ser definidas respecto al conjunto de entidades y con respecto al conjunto de relaciones.

Con respecto al conjunto de entidades se definen:

Superclave.- Conjunto de atributos que permiten identificar de manera unívoca a cada una de las entidades del conjunto de entidades. Un ejemplo de esto sería un campo DNI.

Clave Primaria.- Es la clave candidata que es elegida por el diseñador de la base de datos para identificar a cada una de las entidades de un conjunto de entidades.

Sea R un conjunto de relaciones que implican a los conjuntos de entidades E_1, E_2,.....E_n. Sea (E_i) la clave primaria que denota el conjunto de atributos que forma la clave primaria para el conjunto de entidades E_i. Supóngase que los nombres de atributos de todas las claves primarias son únicas (si no es así, utilícese un esquema adecuado que los renombre) . Supóngase que R no tiene atributos. Entonces los atributos que describen las relaciones individuales del conjunto R, denotadas por el atributo (R), son

Clave-primaria (E_1) U clave-primaria(E_2)U.....Uclave-primria(E_n)

En el caso de que R tenga descriptivos, digamos {a_1,a_2,.....a_n}, entonces el conjunto de atributo(R) consta de

Clave-primaria(E_1)U.....U Clave-primaria(E_n)U{a_1,a_2,.....a_n}.

Para ilustrar lo anterior, considérese el conjunto de relaciones CtaCli (Cuenta Cliente), implica los siguientes conjuntos de entidades:

-cliente, con la clave primaria seguridad-social

-cuenta, con la clave primaria número-cuenta.

Puesto que el conjunto de relaciones tiene el atributo fecha, el conjunto atributo (CtaCli) se compone de tres atributos seguridad-social, número-cuenta, fecha.

Ahora ya podemos explicar qué constituye la clave primaria de un conjunto de relaciones R. La composición de la clave primaria depende de la cardinalidad de asignación y de la estructura de los atributos asociado con el conjunto de relaciones R.

Si el conjunto de relaciones R no tiene atributos asociados, entonces el conjunto atributo (R) forma una superclave. Esta superclave es una clave primaria si la cardinalidad de asignación es de mucha a muchas. Considérese, una vez más, el conjunto de relaciones CtaCli. Si el conjunto de relaciones es muchas a muchas, entonces su clave primaria es {seguridad-social, número-cuenta}. Si el conjunto de relaciones es muchas a una de cliente a cuenta, entonces su clave primaria es {seguridad-social}, ya que una persona puede tener una cuenta asociada con ella misma.

Por ejemplo, una superclave de la entidad PROFESOR sería el conjunto formado por {NSS, DNI}. Dados el número de la seguridad social y el DNI del profesor, se conocerían sin ambigüedad el resto de los atributos (nombre, departamento, área de conocimiento, categoría y despacho). Esto quiere decir que, para un valor determinado de NSS y DNI, no pueden existir diferentes valores para el conjunto de atributos restantes, o lo que es lo mismo, que para un mismo valor de la pareja {NSS, DNI}, el valor del conjunto {NOMBRE, DPTO, AREA, CATEGORIA, DESPACHO} debe ser siempre el mismo.

Otra superclave de la entidad profesor sería el conjunto formado por el único atributo {NSS}, ya que el número de la seguridad social es único por profesor, y por tanto,

una vez conocido dicho número se podría acceder al resto de la información del profesor. De igual manera, se puede considerar como superclave al atributo {DNI}.

Toda entidad debe tener al menos una superclave que estará formada por todos los atributos de la entidad. Sin embargo, lo más normal es que para una entidad se puedan encontrar muchas superclaves. De hecho, cualquier superclave a la que se le añada uno o más atributos forma una nueva superclave. Así por ejemplo, si para la entidad PROFESOR teníamos como superclave {NSS, DNI}, también lo serán los siguientes conjuntos de atributos:

{NSS, DNI, NOMBRE}

{NSS, DNI, DPTO}

{NSS, DNI, AREA}

{NSS, DNI, CATEGORIA}

{NSS, DNI, DESPACHO}

{NSS, DNI, NOMBRE, DPTO}

{NSS, DNI, NOMBRE, AREA}

{NSS, DNI, NOMBRE, CATEGORIA}

{NSS, DNI, NOMBRE, DESPACHO}

{NSS, DNI, NOMBRE, DPTO, AREA}

{NSS, DNI, NOMBRE, DPTO, CATEGORIA}

{NSS, DNI, NOMBRE, DPTO, DESPACHO}

{NSS, DNI, NOMBRE, DPTO, AREA, CATEGORIA}

{NSS, DNI, NOMBRE, DPTO, AREA, DESPACHO}

{NSS, DNI, NOMBRE, DPTO, AREA, CATEGORIA, DESPACHO}

Estas son sólo algunas de la superclaves de la entidad PROFESOR, pero todavía podemos encontrar más. Por ejemplo, si suponemos que un despacho no va a ser compartido por profesores que se llamen exactamente igual, el conjunto {NOMBRE, DESPACHO} forma otra superclave.

Cuando se está diseñando un sistema de gestión de datos, se pueden realizar todas las suposiciones que se crean convenientes, siempre y cuando, éstas sean coherentes con la realidad que se desea representar. Es decir, la suposición de que un despacho no va a ser compartido por profesores con idéntico nombre es bastante razonable. Sería algo insólito encontrar un despacho en el que dos profesores se llamen exactamente igual.

Ejemplo 4.7

Entidad DPTO:

COD	Código del departamento
DESC	Descripción o nombre del departamento
DIRECTOR	Nombre del director
N_PROF	Número de profesores del departamento

Entidad DPTO

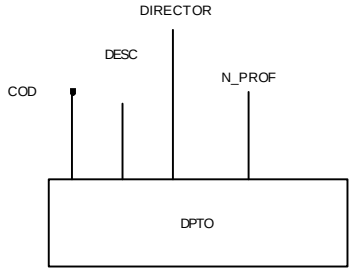


Figura 1.43 **Ejemplo de Atributos de Departamento**

Superclaves:

{COD}, {COD, DESC}

{COD, DIRECTOR}

{COD, N_PROF}

{COD, DESC, DIRECTOR}

{COD, DESC, N_PROF}

{COD, DIRECTOR, N_PROF}

{COD, DESC, DIRECTOR, N_PROF}

{DESC, DIRECTOR}

{DESC, DIRECTOR, N_PROF}

Claves Candidatas:

{COD}, {DESC, DIRECTOR}

Clave Primaria: {COD}

Claves Alternativas: {DESC, DIRECTOR}

Ejemplo 4.8

Entidad ALUMNO:

N_MAT	Número de matrícula asignado al alumno
DNI	DNI del alumno
NOMBRE	Nombre completo
F_NAC	Fecha de nacimiento
DIRECCION	Dirección
LOCALIDAD	Localidad
PROVINCIA	Provincia
TLF	Teléfono

Entidad ALUMNO

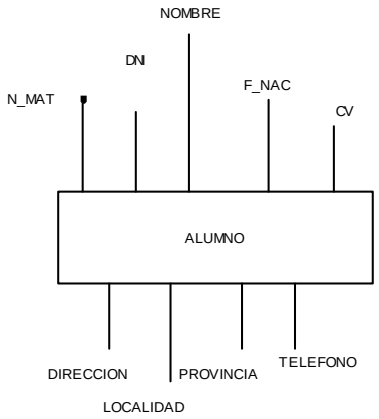


Figura 1.44 **Ejemplo de Atributos de Alumno**

Superclaves:

{MATRICULA}

{MATRICULA, DNI}

{MATRICULA, NOMBRE}

{MATRICULA, DIRECCION}

{MATRICULA, PROVINCIA}

{MATRICULA, DNI, NOMBRE}

{MATRICULA, NOMBRE, TLF}

{MATRICULA, F_NAC, TLF}

{MATRICULA, DIRECCION, LOCALIDAD}

{MATRICULA, DNI, NOMBRE, F_NAC}

{MATRICULA, DNI, NOMBRE, DIRECCION}

{MATRICULA, NOMBRE, PROVINCIA, TLF}

{MATRICULA, F_NAC, LOCALIDAD, PROVINCIA}

{MATRICULA, DIRECCION, LOCALIDAD, PROVINCIA}

{MATRICULA, DNI, NOMBRE, F_NAC, DIRECCION}

{MATRICULA, DIRECCION, LOCALIDAD, PROVINCIA, TLF}

{MATRICULA, DNI, NOMBRE, F_NAC, LOCALIDAD, PROVINCIA, TLF}

{MATRICULA, DNI, NOMBRE, F_NAC, DIRECCION, LOCALIDAD, PROVINCIA, TLF}

{DNI}

{DNI, F_NAC}

{DNI, F_NAC, DIRECCION, LOCALIDAD, PROVINCIA, TLF}

Claves Candidatas:

{MATRICULA}

{DNI}

Clave Primaria:

{MATRICULA}

Claves Alternativas:

{DNI}

Ejemplo 4.9

Entidad PELICULA

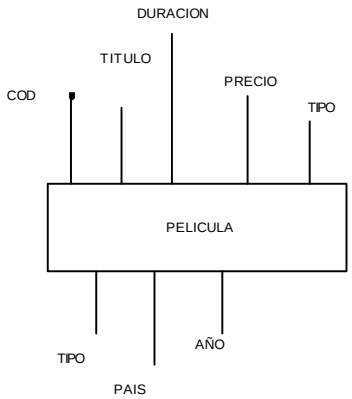


Figura 1.45 **Ejemplo de Película con super clave**

Superclaves:

{COD}

{COD, TITULO}

{COD, DURACION}

{COD, PRECIO}

{COD, TIPO}

{COD, PAIS}

{COD, AÑO}

{COD, TITULO, DURACION}

{COD, TITULO, PRECIO}

{COD, TITULO, TIPO}

{COD, TITULO, PAIS}

{COD, TITULO, AÑO}

{COD, TITULO, DURACION, PRECIO}

{COD, TITULO, DURACION, TIPO}

{COD, TITULO, DURACION, PAIS}

{COD, TITULO, DURACION, AÑO}

{COD, TITULO, DURACION, PRECIO, TIPO}

{COD, TITULO, DURACION, PRECIO, PAIS}

{COD, TITULO, DURACION, PRECIO, AÑO}

{COD, TITULO, DURACION, PRECIO, TIPO, PAIS}

{COD, TITULO, DURACION, PRECIO, TIPO, AÑO}

{COD, TITULO, DURACION, PRECIO, TIPO, PAIS, AÑO}

{TITULO, DURACION}

{TITULO, DURACION, PRECIO}

{TITULO, DURACION, TIPO}

{TITULO, DURACION, PAIS}

{TITULO, DURACION, AÑO}

{TITULO, DURACION, PRECIO, TIPO}

{TITULO, DURACION, PRECIO, PAIS}

{TITULO, DURACION, PRECIO, AÑO}

{TITULO, DURACION, PRECIO, TIPO, PAIS}

{TITULO, DURACION, PRECIO, TIPO, AÑO}

{TITULO, DURACION, PRECIO, TIPO, PAIS, AÑO}

Claves Candidatas:

{COD}

{TITULO, DURACION}

Clave Primaria:

{COD}

Claves Alternativas:

{TITULO, DURACION}

Si un atributo o conjunto de atributos de una entidad es clave primaria de otra entidad, a dicho conjunto de atributos se le denomina Clave Foránea o Clave Ajena de la entidad. Volvamos al ejemplo de la entidad AUTOMOVIL, y supongamos que tenemos otra entidad denominada PROPIETARIO para almacenar los datos del dueño del automóvil:

DNI	DNI del dueño del automóvil
NOMBRE	Nombre del propietario
F_CARNET	Fecha en la que consiguió el carnet de conducir
TLF	Teléfono de contacto

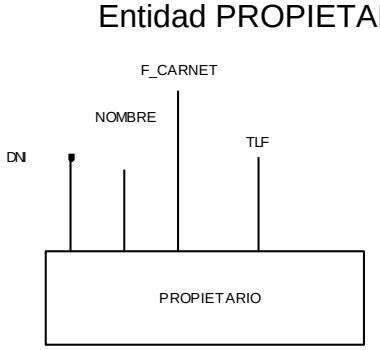


Figura 1.46 **Ejemplo de Atributos de Propietarios**

Si nos fijamos en el modelo de la entidad AUTOMOVIL, existe un atributo {DNI} que en la entidad PROPIETARIO es clave primaria. Por tanto, al atributo {DNI} de la entidad AUTOMOVIL se le denomina clave foránea de la entidad PROPIETARIO.

Bibliografía comentada

9.22 Clase 22: DISEÑO DE BBDD

MODELO E/I

ELEMENTOS BASICOS DEL MODELO

CARDINALIDADES DE RELACIONES Y ATRIBUTOS

OBJETIVOS:

Identificar las características de un modelo E/I

Identificar los elementos básicos del modelo E/I

Identificar las cardinalidades de relaciones y de atributos

9.22.1 Modelo E/I

- Múltiples modelos de Bases de Datos

  OO, Relacional, Semántico y otros

El modelo E/R es un modelo semántico

- Orientado al diseño de BD
- Con un nivel superior de modelado
- Independiente del modelo de BD
- Permite tener autonomía de la elección de modelo de BD en los primeros estados del diseño de una BD

- E/R significa Entity-Relationship
  - Entidad-Interrelación o Entidad-Relación
  - Se suele usar la segunda forma aunque la primera es más correcta

- El modelo E/R se debe a P. Chen, en 1976
  - Estándar ANSI en 1988
  - Posteriormente, Huges, creo el modelo E/R extendido
  - En la actualidad existen muchas variantes muy similares

- La versión inicial tenía los conceptos:
  - Entidad
  - Interrelación
  - Atributo

- La versión extendida, añadía:
  - Composición de atributos
  - Jerarquías de generalización

- Subtipos
- Identificadores

- E/R = modelo + técnica de diagramación
  - Es más conocido y utilizado por la técnica de diagramación que por el modelo en sí
  - Desde un punto de vista crítico, podemos decir sobre el modelo:
    - (-) no es demasiado flexible (hay aspectos semánticos difíciles de representar)
    - (-) no permite representar la evolución del modelo, un cambio obliga a empezar de cero
    - (-) resulta aconsejable y casi necesario usar una de las herramientas gráficas existentes
    - (+) buena captura de la semántica de los problemas
    - (+) resulta sencilla la lectura y comprensión de los modelos
    - (+) es muy fácil la implementación relacional (automática en algunos casos)

9.22.2 Elementos básicos del modelo

Con estos elementos se pueden crear modelos avanzados que capturen la semántica de problemas complejos

Entidades

- Función: representar conceptos del mundo real
- Diagrama: rectángulo con el nombre de clase (en singular)
- Ejemplo: persona, hombre, mujer, empleado. son entidades para una BD de personal

Interrelaciones

- Función: representan vinculaciones entre dos o más entidades.
- Diagrama: un rombo unido por líneas a las entidades que conecta, en su interior se escribe el nombre que representa la relación que se establece
- Ejemplo: Nació_en relaciona la entidad persona con la entidad ciudad
- Nota: las relaciones pueden ser binarias, n-arias o recursivas (de una entidad consigo misma)

Atributos

- Función: representar propiedades elementales de las entidades o interrelaciones.
- Diagrama: un círculo pequeño unido por una línea a la entidad o interrelación más el nombre del atributo a lado del mismo. También se

pueden representar por una elipse unida a la entidad o interrelación con el nombre en su interior

- Ejemplo: para la entidad ciudad, el nombre y número de habitantes son atributos. Para la relación vive_en, fecha de traslado es un atributo.

Cardinalidades de relaciones y atributos

Elementos del modelo extendido

- Jerarquías
- Subtipos
- Atributos Compuestos
- Identificadores

Bibliografía comentada

Manual de: Curso de base de Datos 2003, Impartido por: José María Gutiérrez Martínez. Tema 2 Diseño de Bases de Datos paginas 16 a 18

9.23 Clase 23: Modelo entidad – relación

DIAGRAMA ENTIDAD-RELACION

REDUCCIÓN DE LOS DIAGRAMAS E-R A TABLAS

OBJETIVOS:

Identificar los elementos de un Diagrama Entidad-Relación

Identificar los procesos para transformar los Diagramas E-R a Tablas

9.23.1 Diagrama entidad-relación

La estructura lógica global de una base de datos puede representarse gráficamente por medio de un diagrama E-R. Recuérdese que un diagrama de ese tipo consta de las siguientes componentes.

-**Rectángulo**, que representan conjuntos de entidades

-**Elipse**, que representa atributos

-**Rombos**, que representan conjuntos de relaciones

-**Líneas**, que enlazan atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.

Considérese el diagrama entidad-relación de la figura 1.47, que consta de dos conjuntos de entidades, cliente y cuenta, relacionados mediante un conjunto binario de relaciones CtaCli. Los atributos asociados con clientes son nombre-cliente, seguridad-social, calle y ciudad-cliente. Los atributos asociados con cuenta son número-cuenta y saldo.

El conjunto de relaciones CtaCli puede ser muchas a muchas, unas a muchas, muchas a una o una a una. Para distinguir entre éstas. Dibujaremos bien una línea con dirección del conjunto de relaciones ***CtaCli*** al conjunto de entidades ***cuenta*** especifica que el conjunto de entidades cuenta participa en una relación una a una o en una a muchas a una con el conjunto de entidades cliente. No puede participar en una relación muchas a muchas o en una a muchas con el conjunto de entidades cliente. Una línea sin dirección del conjunto de relaciones CtaCli al conjunto de entidades cuenta especifica que el conjunto de entidades cuenta participa en una relación muchas a muchas o en una muchas a una con el conjunto de entidades cliente.

Volviendo al diagrama E-R de la figura 1.47. vemos que el conjunto de relaciones CtaCli es muchas a muchas. Si el conjunto de relaciones CtaCli fuera una a mucha, de cliente a cuenta, entonces la línea de CtaCli a cuenta tendría dirección, con una flecha apuntada al conjunto de entidades (Figura 1.48.a). De manera similar, si el conjunto de relaciones CtaCli fuera muchas a una de cliente a cuenta, entonces la línea de CtaCli a cuenta tendría una flecha apuntando al conjunto de entidades cuenta (Figura 1.48.b). Finalmente, si el conjunto de relaciones CtaCli fuera una a una, entonces las dos líneas de CtaCli tendrían flechas, una apuntando al conjunto de entidades cuenta y otra apuntando al

conjunto de entidades cliente (Figura 1.49). Veremos otra forma de representar los atributos que podrás encontrar en los libros.

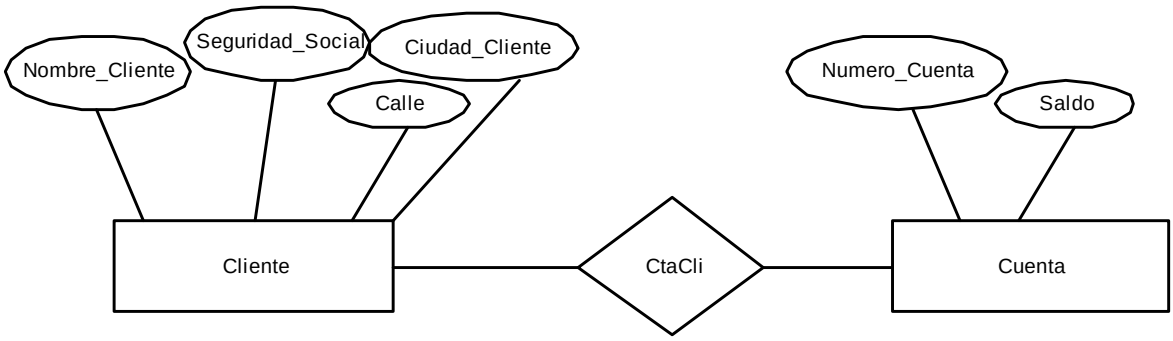
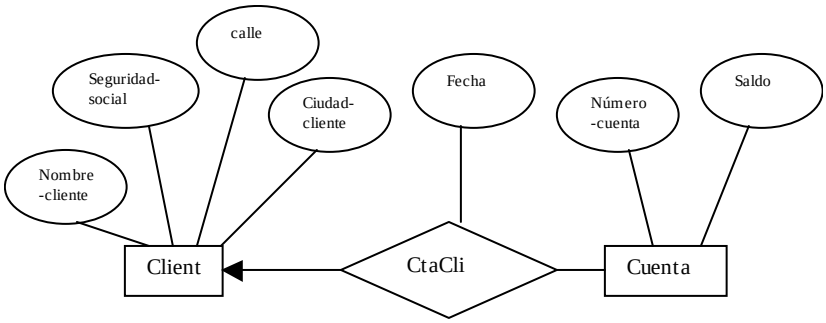
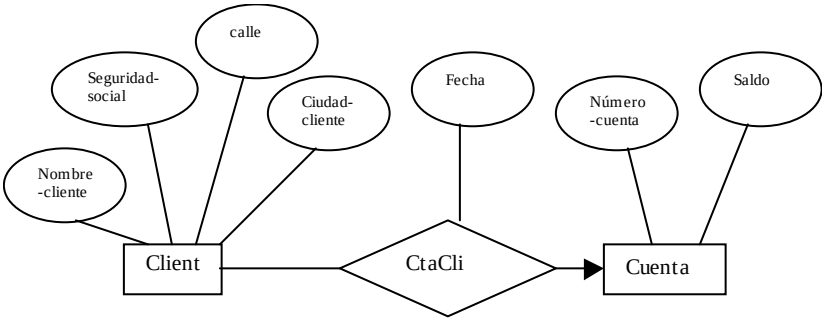


Figura 1.47 ***Diagrama* E-R**



(a)



(b)

Figura 1.48 **Relaciones una a muchas y muchas a una.**

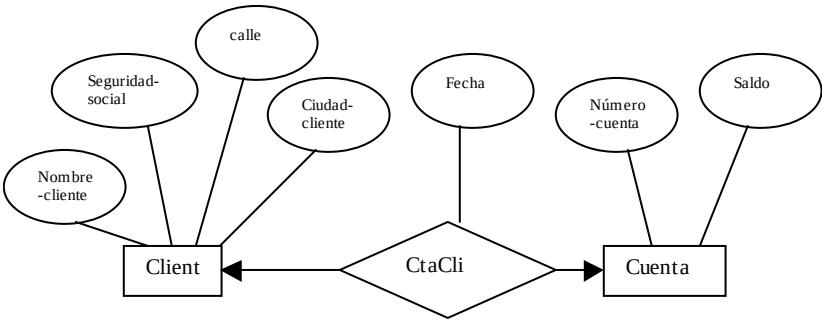


Figura 1.49 **Relaciones una a una.**

Un conjunto de entidades débil se indica en los diagramas E-R por medio de un rectángulo de doble contorno. En la figura 1.51, el conjunto de entidades débil transacción es dependiente del conjunto de entidades fuerte cuenta a través del conjunto de relaciones bitácora.

Los conjuntos de relaciones no binarias pueden especificarse fácilmente en un diagrama E-R. La figura 1.52 consta de tres conjuntos de entidades, cliente, cuenta y sucursal, relacionados por medio del conjunto de relaciones CAB. Este diagrama especifica que un cliente puede tener varias cuentas, cada una situada en una sucursal especifica del banco, y que una cuenta puede pertenecer a varios clientes distintos.

9.23.2 Reducción de los diagramas E-R a tablas

Una base de datos que se ajusta a un diagrama E-R puede representarse por medio de una colección de tablas. Para cada conjunto de entidades,

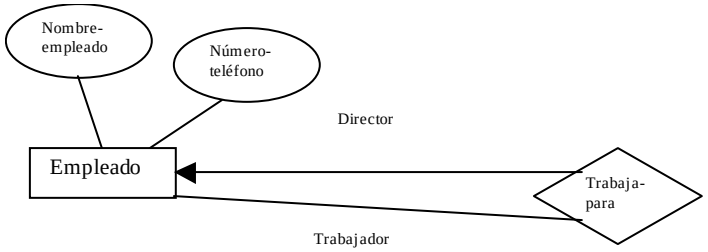


Figura 1.50 **Diagrama E-R con indicadores de papeles**

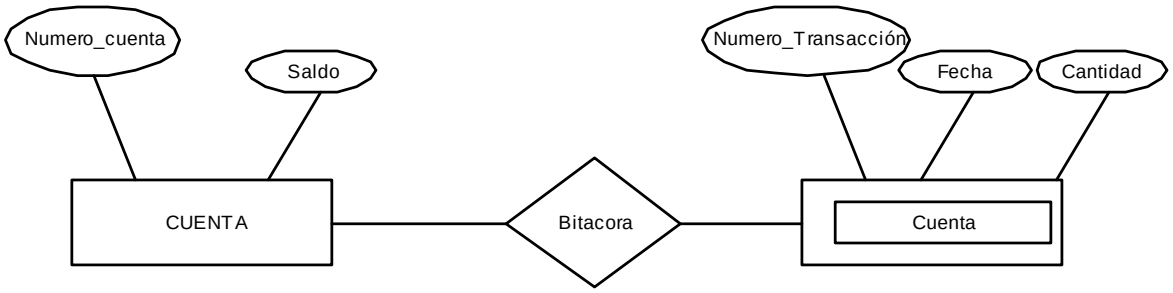


Figura 1.51 **Diagrama E-R con un conjunto de entidades débil.**

Y para conjunto de relaciones en la base de datos, existen una tabla única a la que se le asigna el nombre del conjunto de entidades o del conjunto de relaciones correspondiente. Cada tabla tiene un número de columnas que, a su vez, tienen nombres únicos.

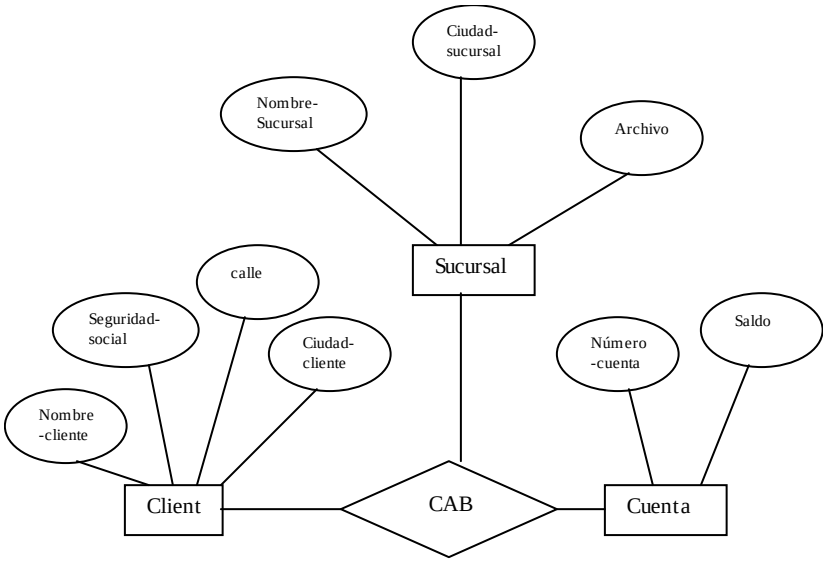


Figura 1.52 **Diagrama E-R con una relación ternaria.**

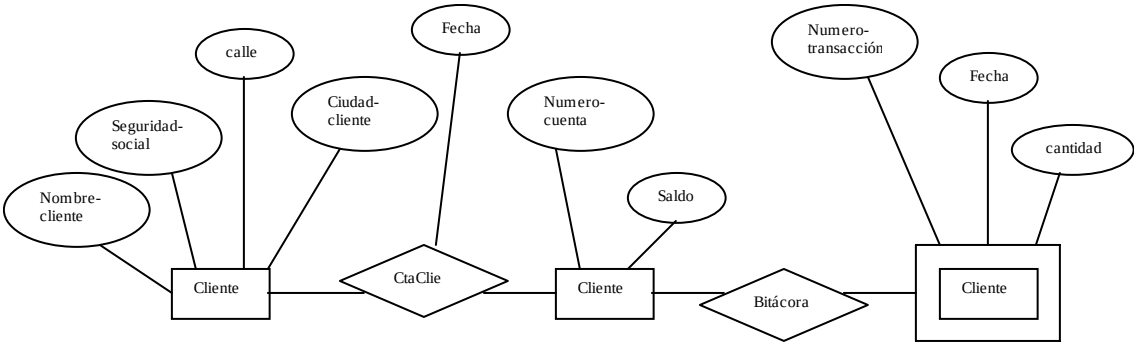


Figura 1.53 **Diagrama E-R**

9.24 Clase 24: Modelo entidad – relación

RELACIONES

EJEMPLO DE UN MODELO ENTIDAD-RELACIÓN

DISEÑO DE UN ESQUEMA DE BASES DE DATOS ENTIDAD - RELACIÓN

OBJETIVOS:

Identificar los elementos de un Diagrama Entidad-Relación

Identificar los procesos para transformar los Diagramas E-R a Tablas

9.24.1 Relaciones

Las entidades por sí solas no describen la realidad de un sistema de información. No basta con identificar objetos, hay además que establecer las asociaciones existentes entre los mismos. Para ello, se introduce el concepto de Relación. Relación significa la existencia de algo común entre entidades. Las relaciones se representan mediante rombos y el nombre de la relación en su interior.

El Grado de una Relación representa la participación en la relación de cada una de las entidades afectadas y existen 3 tipos posibles:

- 1:1 (una a una). A cada ocurrencia de una entidad le corresponde no más de una ocurrencia de la otra y a la inversa.
- 1:n (una a muchas). A cada ocurrencia de la primera entidad le pueden corresponder varias ocurrencias de la segunda y a cada ocurrencia de la segunda le corresponde no más de una ocurrencia de la primera.
- n:m (muchas a muchas). A cada ocurrencia de la primera entidad le pueden corresponder varias ocurrencias de la segunda y viceversa.

El grado de las relaciones se puede representar de dos formas diferentes (aunque a menudo se utilizan ambos métodos en el mismo diagrama para que resulte más claro):

- Indicando el grado en los segmentos que asocian las entidades de la relación.
- Incluyendo en los segmentos que asocian las entidades una flecha para indicar el grado 1, y dos flechas para mostrar el grado n o m

Ejemplo 4.10

Vamos a representar el modelo Entidad-Relación para almacenar los datos de propietarios de vehículos. Para ello, necesitamos dos entidades: una que almacene los datos de dueños de automóviles (PROPIETARIOS) y otra que contenga la información de los vehículos (AUTOMOVIL).

La relación que existe entre ambas entidades es una relación de pertenencia (PERTENECE). Esta relación tendrá como atributos los siguientes:

MATRICULA	Matrícula del vehículo
DNI	DNI del dueño del automóvil
F_COMPRA	Fecha en la que el propietario compró el automóvil
KM	Kilómetros del vehículo cuando lo adquirió el propietario

El grado de la relación es 1:n, un propietario puede tener varios vehículos, pero un automóvil sólo está registrado a nombre de un único propietario. Esto es así, porque estamos almacenando información actual. Si quisiéramos registrar el movimiento de vehículos entre propietarios para un determinado período de tiempo, la relación sería de muchos propietarios a muchos automóviles, y la clave de la relación pasaría a ser {MATRICULA, DNI}. Vamos a explicar esto más detenidamente. Supongamos que queremos almacenar los datos de todos los coches de Jaén en un período de 5 años junto con sus propietarios y los datos de la compra (F_COMPRA y KM). Está claro que un propietario puede tener más de un coche. En principio, si almacenamos únicamente los datos actuales (sin tener en cuenta el paso del tiempo), no es posible que un coche esté registrado a nombre de dos personas. Sin embargo, si consideramos un período de 5 años, con toda probabilidad habrá alguien que venda su coche y ese mismo vehículo sea adquirido por otra persona. Por lo tanto, el mismo coche ha pertenecido a dos propietarios. Ese vehículo puede volver a venderse una y otra vez, y pertenecer a muchos propietarios. Así, la relación también sería de un automóvil a muchos propietarios. En definitiva tenemos que la relación es de muchos a muchos, ya que un propietario puede tener varios coches y un coche puede pertenecer a varios propietarios (siempre teniendo en cuenta que estamos hablando de un intervalo de tiempo y no de la situación actual

Modelo Entidad-Relación para almacenar propietarios de vehículo

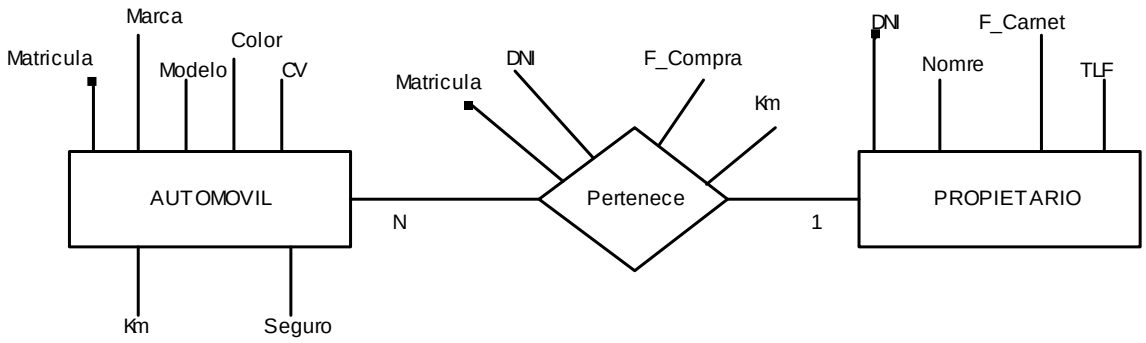
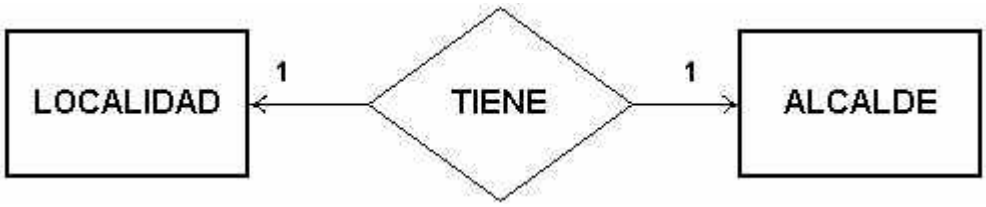


Figura 1.54 **Diagrama E-R**

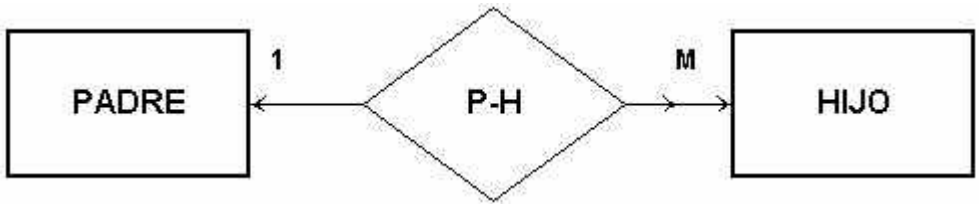
Ejemplo 4.11

Relación LOCALIDAD-ALCALDE



Ejemplo 4.12

Relación PADRE-HIJO



Ejemplo 4.13

Relación SECCION-EMPLEADO

Ejemplo 4.14

Relación SOCIO-PELICULA

Figura 1.55 **Ejemplos de Relaciones**

Ejemplo 2.15

Relaciones entre las entidades de un departamento de la Universidad

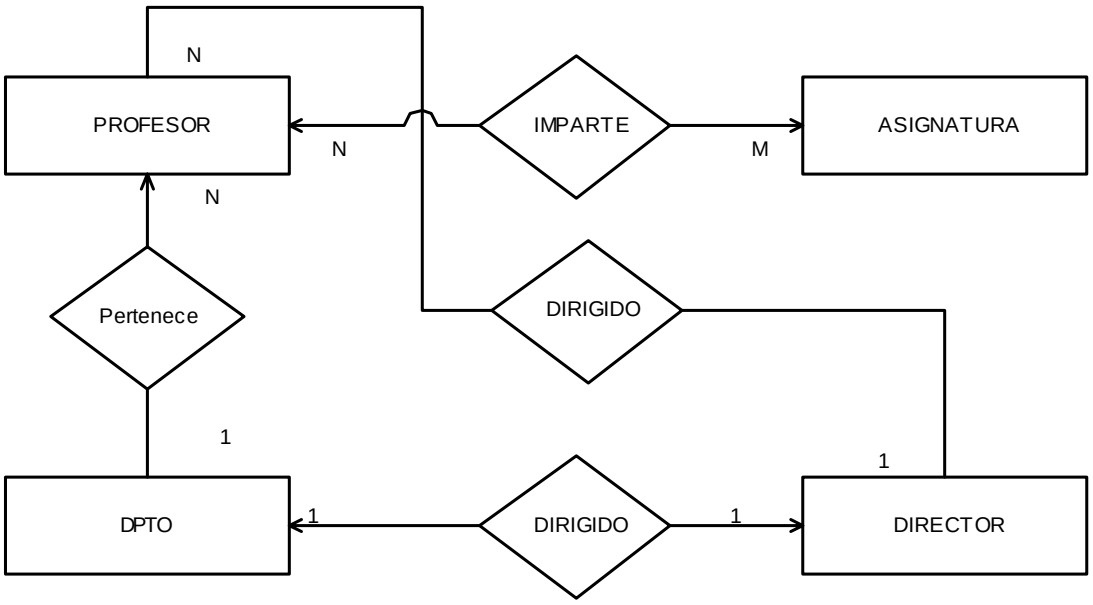


Figura 1.56 **Diagrama E-R**

9.24.2 Ejemplo de un Modelo Entidad-Relación

Se desean informatizar los datos de profesores y proyectos de investigación, así como la relación que indica los proyectos en los que interviene cada profesor. Se tienen en cuenta las siguiente suposiciones:

- Un profesor puede estar trabajando en varios proyectos
- En un proyecto pueden estar trabajando varios profesores
- Cuando un profesor termina de trabajar en un proyecto no podrá volver a trabajar en el mismo proyecto

Para realizar el modelo E-R, lo primero que hay que hacer es identificar las entidades que van intervenir. En este caso, se distinguen al menos dos entidades: PROFESOR y PROYECTO. El modelo E-R más simple incluiría únicamente estas dos entidades, junto con la relación P-P que indica los profesores que intervienen en los proyectos. El grado de la relación será n:m (de muchos a muchos), ya que un profesor puede trabajar en varios proyectos y en un proyecto pueden intervenir varios profesores. Ahora, sólo falta indicar los atributos de cada entidad y de la relación:

Entidad PROFESOR

NSS	Número de la seguridad social
DNI	DNI
NOMBRE	Nombre completo
DPTO	Departamento al que pertenece
AREA	Área de conocimiento a la que está adscrito
CATEGORÍA	Categoría (Catedrático, Titular, Ayudante, Asociado)
DESPACHO	Número del despacho en el que habitualmente trabaja
TIEMPO	Indica si el profesor está contratado a tiempo completo o tiempo parcial

Superclaves:

{NSS}

{NSS, DNI}

{NSS, NOMBRE, TIEMPO}

{NSS, DNI, NOMBRE, DPTO, AREA, CATEGORIA, DESPACHO, TIEMPO}

{DNI}

{DNI, CATEGORIA}

{NOMBRE, DESPACHO}

{NOMBRE, DESPACHO, DPTO, AREA}

Claves candidatas:

{NSS}

{DNI}

{NOMBRE, DESPACHO}

Clave primaria:

{NSS}

Claves alternativas:

{DNI}

{NOMBRE, DESPACHO}

Entidad PROYECTO

COD	Código asignado al proyecto
DESC	Descripción / Denominación del proyecto
PRESUP.	Presupuesto asignado al proyecto
AMBITO	Ámbito del proyecto (Internacional, Nacional, Local)
DIRECTOR	Nombre del director del proyecto
F_INICIO	Fecha en la que comienza el proyecto
F_FIN	Fecha de finalización del proyecto

Superclaves:

{COD}

{COD, DESCP}

{COD, DIRECTOR}

{COD, F_INICIO, F_FIN}

{COD, DESC, PRESUP, AMBITO, DIRECTOR, F_INICIO, F_FIN}

{DESC, F_INICIO}

{DESC, F_INICIO, F_FIN}

{DESC, F_INICIO, DIRECTOR}

{DIRECTOR, F_INICIO}

{DIRECTOR, F_INICIO, PRESUP}

{DIRECTOR, F_INICIO, AMBITO}

Claves candidatas:

{COD}

{DESC, F_INICIO}

{DIRECTOR, F_INICIO}

Modelo Entidad-Relación de los proyectos en los que intervienen profesores

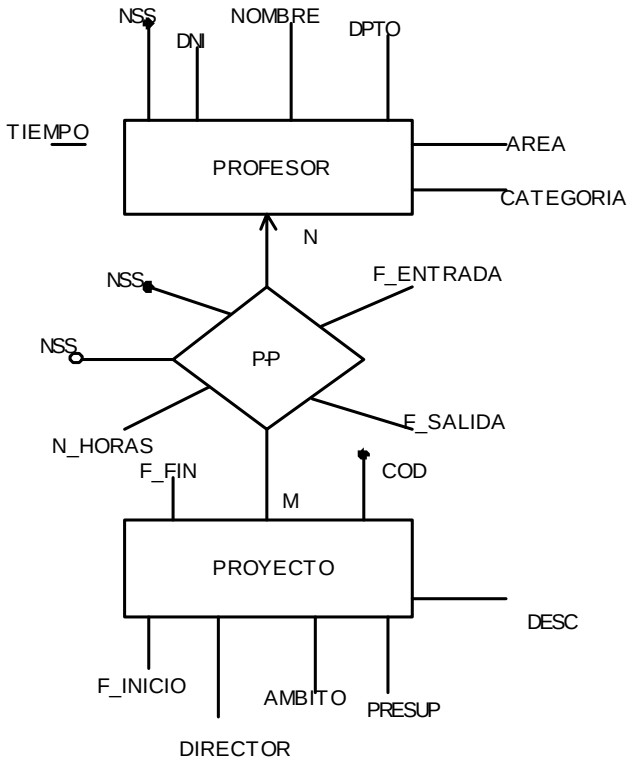


Figura 1.57 **Diagrama E-R**

Clave primaria:

{COD}

Claves alternativas:

{DESC, F_INICIO}

{DIRECTOR, F_INICIO}

Relación P-P

NSS	Número de la seguridad social del profesor que interviene en el proyecto
COD	Código del proyecto en el que trabajan profesores
N_HORAS	Número de horas semanales que se asigna a un profesor para que dedique a un proyecto
F_ENTRADA	Fecha en la que un profesor comienza a trabajar en un proyecto
F_SALIDA	Fecha en la que un profesor abandona un proyecto

Clave primaria:

{NSS, COD}

Si no consideramos la última suposición del ejercicio (cuando un profesor termina de trabajar en un proyecto no podrá volver a trabajar en el mismo proyecto) la clave primaria debería incluir también el atributo F_ENTRADA, ya que en este caso, no sería posible que un profesor comenzara a trabajar en un proyecto determinado dos veces en el mismo día. Con la clave {NSS, COD, F_ENTRADA} nos aseguramos de que no haya ninguna ocurrencia repetida para la relación P-P.

Como se comentó al principio del ejemplo, éste sería el modelo E-R más elemental. Sin embargo, lo podemos complicar cuanto queramos. Por ejemplo, se podría almacenar información acerca de los organismos que subvencionan los proyectos. Para ello, incluimos otra entidad EMPRESA y la unimos a la entidad PROYECTO a través de una relación P-E de grado n:m (una empresa puede colaborar en varios proyectos y un proyecto puede estar subvencionado por varias empresas). La entidad EMPRESA incluye como atributos el NIF, el nombre de la empresa, el número de empleados y la fecha de creación. La relación P-E tiene como atributos la clave de cada una de las entidades que relaciona junto con la cantidad subvencionada por la empresa al proyecto y el tipo de subvención (anual, trimestral, mensual,...).

Modelo Entidad-Relación para almacenar profesores, proyectos y empresas que subvencionan los proyectos otra forma de representar los modelos pero las claves son fáciles de identificar.

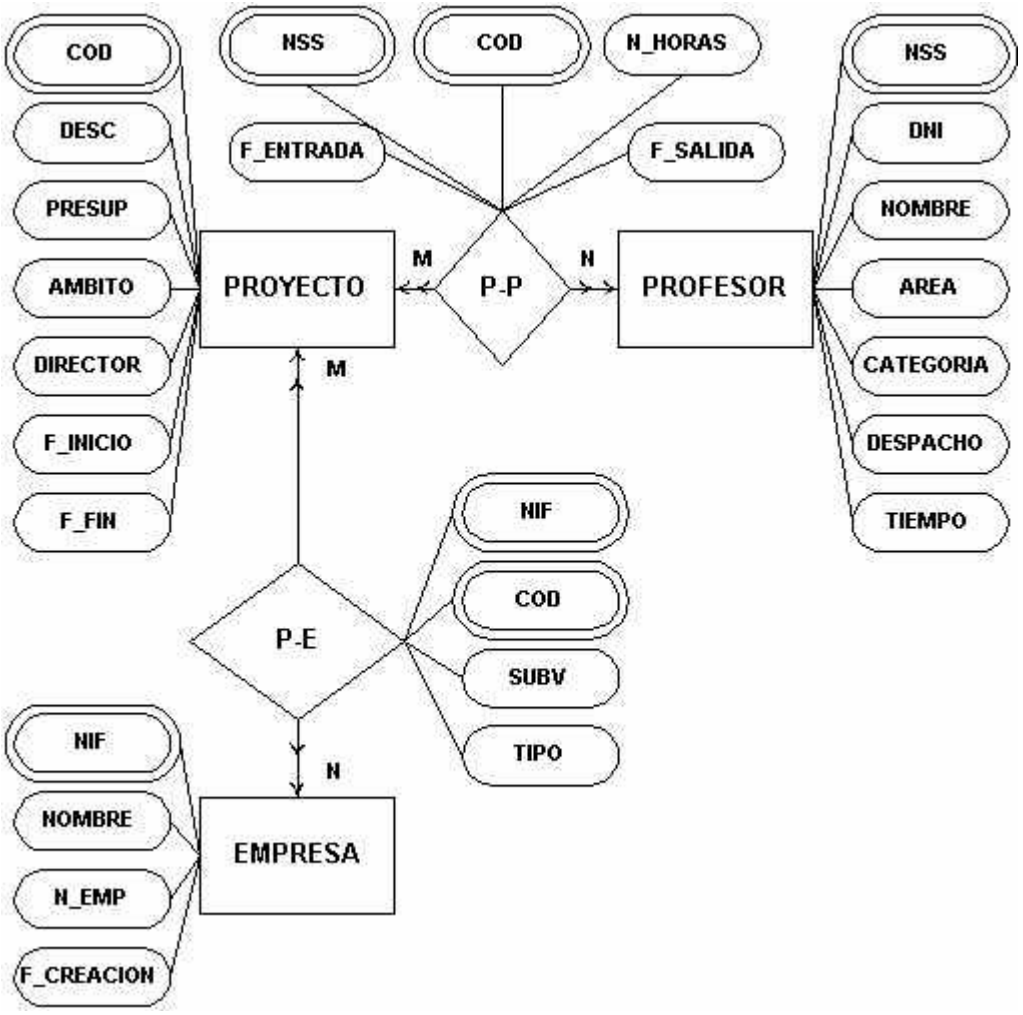


Figura 1.58 **Diagrama E-R**

Todavía podría complicarse más. En lugar de almacenar en la entidad PROFESOR el departamento al que pertenece, se podría considerar otra entidad DPTO, que incluyera la información referente al mismo. Esta nueva entidad debería unirse a la entidad PROFESOR a través de una relación D-P que indicará los profesores que pertenecen a un departamento. El grado de la relación sería 1:n, un departamento está formado por muchos profesores, pero un profesor pertenece a un único departamento.

Teniendo en cuenta que un departamento tiene varias áreas de conocimiento, podría incluirse otra entidad AREA relacionada con DPTO a través de una relación m:1 (un área pertenece a un único departamento y un departamento incluye varias áreas) y relacionada también con PROFESOR con grado 1:m (a un mismo área de conocimiento pertenecen varios profesores y un profesor sólo está adscrito a un área). En caso de incluir esta entidad, debemos eliminar el atributo AREA de la entidad profesor.

En la práctica, la complejidad del modelo E-R depende de la información que queramos almacenar. Cuanto más amplia sea la parcela del mundo real que queremos representar, tanto más complejo será el modelo E-R.

9.24.3 Diseño de un esquema de bases de datos Entidad - Relación.

Para un diseño de un esquema de base de datos hay cuatro fases:

1. Especificación de requisitos del usuario.- Consiste en obtener las necesidades de datos de los usuarios de la base de datos, esto es, sonsacarle al usuario toda la información que se desea plasmar en la base de datos. Esta es la fase que se dará en el examen.
2. Diseño conceptual (Entidad - Relación).
3. Especificación de requisitos funcionales.- Vamos a definir las operaciones que se harán sobre la base de datos (operaciones permitidas sobre la base de datos)
4. Especificación de requisitos funcionales.- Primero se procede a realizar el diseño lógico, que consiste en adaptar el diseño conceptual al sistema de gestión de la base de datos, y a continuación se realiza el diseño físico, que consiste en dar todas las características de almacenamiento de la base de datos.

Bibliografía comentada

9.25 Clase 25: DISEÑO DE BBDD

TRANSFORMACIÓN DEL ESQUEMA CONCEPTUAL AL RELACIONAL

LAS OPCIONES DE CLAVE AJENA EN LA TRANSFORMACIÓN DEL ME/R AL RELACIONAL

GRAFO RELACIONAL

OBJETIVOS:

Transformar del esquema conceptual al relacional

Características de las claves ajena en la transformación del ME/R al relacional

Identificar las características del grafo relacional

Una vez expuesto los fundamentos de la bases de datos y analizadas las características de los modelos entidades/interrelación y relacional, podemos abordar el problema práctico del diseño de una base de datos. La gran difusión de los SGBDR, que existen para todo tipo de equipos y que se han impuesto en el mercado, nos lleva a tratar más a fondo el diseño de bases de datos relaciónales. La metodología de diseño que se propone en esta unidad comienza en una primera fase describiendo el mundo real en el modelo E/R, para transformar posteriormente el esquema E/R resultante al modelo relacional. La teoría de la normalización, que también se aborda (aunque sólo es sus tres primeras formas normales) en este unidad nos permitirá comprobar si un esquema relacional cumple una serie de requisitos, y sino fuese así llevarlo a la forma normal que se considere más adecuada.

9.25.1 Etapa de una metodología de diseño

En los últimos años venimos asistiendo, gracias al avance tecnológico, a una gran difusión de los SGBDR, cuyos productos se soportan en cualquier plataforma, desde los superordenadores a los ordenadores personales. Sin embargo, y a pesar del esfuerzo realizada por numerosos investigadores y estudioso del tema, la concepción de una BD sigue siendo una tarea larga y costosa.

Estas dificultades inherente al diseño de una base de datos han de afrontarse con procedimientos ordenados y metódicos en el marco de una metodología de diseño de base de datos.

En el proceso de diseño de una base de datos hemos de distinguir tres grandes fases:

Diseño conceptual: cuyo objetivo es obtener una buena representación de los recursos de información de la empresa, con independencia de usuarios o aplicaciones en particular, y fuera de consideraciones sobre eficiencias del ordenador.

Diseño lógico: cuyo objetivo es transformar el esquema conceptual obteniendo en la etapa anterior, adaptándolo al modelo de datos en el que se apoya el SGBD que se va a utilizar. Nosotros nos vamos a referir al modelo relacional, pero de forma análogo se podría

adaptar esta etapa de diseño lógico a otros modelos de datos, como el jerárquico o el Codasyl.

Diseño físico; cuyo objetivo es conseguir una instrucción, lo más eficiente posible, del esquema lógico.

Cuando estudiamos el papel de los modelos de datos en el díseño de las bases de datos, ya señalamos las ventajas de obtener, en una primera fase del diseño, un esquema conceptual independiente de las características de los modelos convencionales como el modelo relacional, y que recogiese la semántica del mundo real.

Un problema que se suele presentar en el diseño de bases de datos es la comunicación entre el diseñador y el usuario, este último conoce bien el dominio de aplicación, lo que en general no ocurre con el diseñador, pero muchas veces no sabe expresarlo de forma correcta y, menos aún, precisa la utilización del modelo E/R, en una primera fase del diseño, facilita el diálogo entre el informático, conocedor de las técnicas de estructuración de los datos pero ajeno al domino de la aplicación, y el usuario, que suele conocer bien su mundo real pero no es capaz de describirlo bajo las premisas que impone un modelo convencional.

El modelo E/R sencillo, pero a la vez suficientemente potente, permite entablar un dialogo entre el usuario y el diseñador que facilitará que se despejen dudas y aclaren aspectos del universo del discurso a modelar. Se facilitar así la colaboración de los especialistas con los usuarios, de manera que estos últimos pueden participar activamente, e incluso ser protagonistas, en el diseño.

Podemos representar esquemáticamente las dos primeras fases (diseño conceptual y diseño lógico) de la metodología como se indica en la figura 5.1, en la cual aparece el proceso de diseño de una biblioteca; el diseñador observa el mundo real bajo unos ciertos objetivos (Universo del Discurso) y, apoyándose en una primera etapa en el modelo E/R, llega a un esquema conceptual, al cual se le aplicará un conjunto de reglas a fin de transformarlo en una estructura relacional (conjunto de tablas).

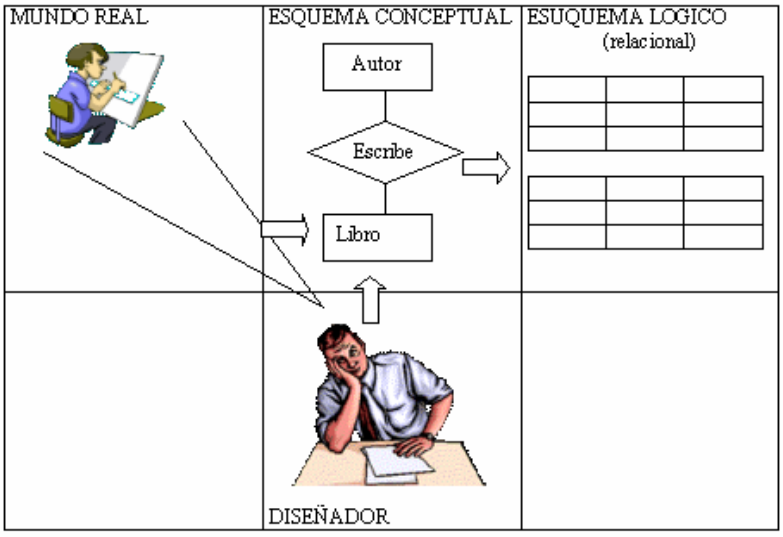


Figura 1.59 **Representación grafica de la metodología propuesta para el diseño de base de datos relacionales.**

Una vez adaptada la estructura relacional a las restricciones impuestas por el producto concreto y a las exigencias relativas a su eficiencia, será posible almacenar en la estructura resultante los datos necesarios para la gestión de dicha biblioteca: préstamo de libro, consulta sobre artículos de una determinada temática, sobre autores, etc.

9.25.2 Transformación del esquema Conceptual al Relacional.

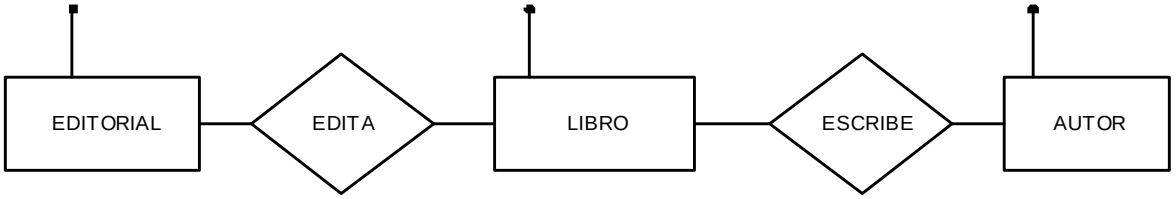
El paso de un esquema en el modelo E/R al relacional está basado en los tres principios siguientes:

- Todo tipo de entidad se convierte en una relación.
- Todo tipo de interrelación N:M se transforma en una relación
- Todo tipo de interrelación 1:N se traduce en el fenómeno de propagación de clave o bien se crea una nueva relación.

A primera vista se puede observar que en el paso del modelo E/R al relacional se pierde semántica, puesto que tanto las entidades como las interrelaciones se transforma en relaciones, de forma que ya no es posible distinguir entre unas y otras (en el modelo relacional sólo existe la relación para presentar ambos tipos de objetos).

También se constata que la pérdida de semántica es aún mayor en el caso de la propagación de clave, donde desaparece incluso el nombre de la interrelación. Es preciso destacar que la pérdida de semántica no implica, necesariamente, un peligro para la integridad de la base de datos, ya que, si la transformación se ha realizado correctamente, se habrán definido las necesarias restricciones (muy en especial las claves ajenas con sus opciones) que asegurarán las consistencia de los datos.

En el ejemplo de la figura 1.59 puede observarse que las tres entidades EDITORIAL. LIBRO y AUTOR se transforman en otras tantas relaciones. La interrelación N:M Escribe da lugar a una nueva relación ESCRIBE cuya clave primaria es la concatenación de los atributos identificadores de las entidades que participan en ella (Nombre_a de AUTOR y Código del LIBRO), siendo además éstos claves ajenas de ESCRIBE que referencia a las relaciones AUTOR y LIBRO, respectivamente. La interrelación 1:N Edita se transforma mediante el mecanismo de propagación de clave, por el que se incluye en la relación LIBRO la clave de la relación EDITORIAL (a la que llamamos Editorial); atributos que será clave ajena de la relación LIBRO referencia a EDITORIAL.



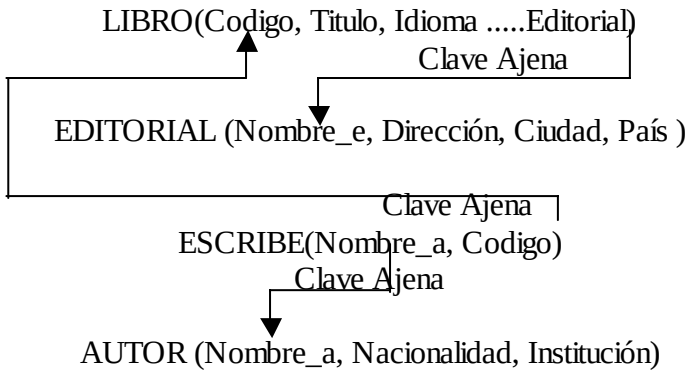


Figura 1.60 **Ejemplo de paso de ME/R al modelo relacional**

Las opciones de clave ajena en la transformación del ME/R al relacional

Si bien se insiste, que el paso del ME/R al relacional implica una cierta pérdida de semántica, se debe procurar que esta pérdida sea lo menor posible, además de asegurar al máximo la consistencia de los datos, es decir, la integridad de la base de datos en las operaciones de actualización.

Analizaremos las posibles opciones de borrado y modificación asociada a las claves ajenas en los casos de propagación de clave y de creación de una nueva relación.

A) Transformación por propagación de clave (interrelación 1:N)

Sea la interrelación Edita de la figura 1.60, donde suponemos que todo libro de nuestra base de datos está siempre editada por una única editorial, es decir, las cardinalidades de EDITORIAL en la interrelación Edita serían las que aparecen en la figura 1.61 a la derecha de la figura se puede ver el esquema relacional resultante de la transformación.

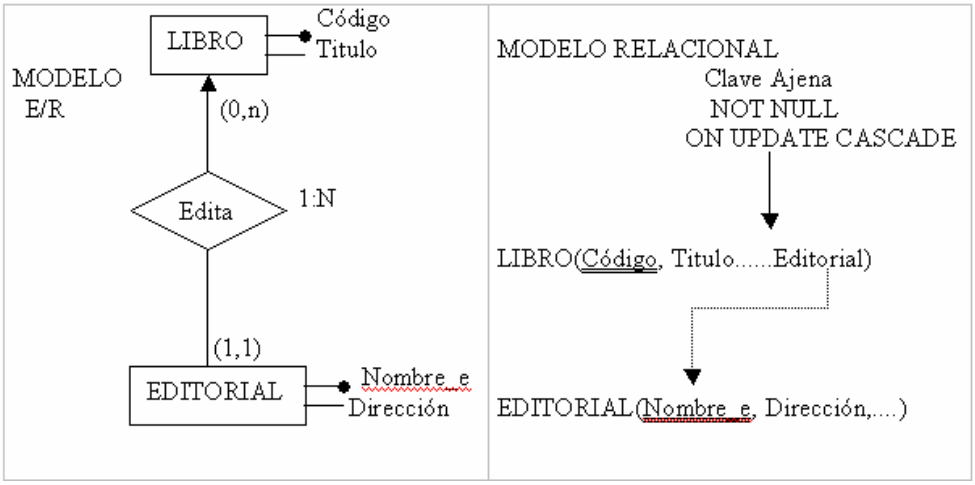


Figura 1.61 **Ejemplo de opciones de clave ajena en una transformación por propagación de clave**

Las posibles opciones de borrado que se podrían aplicar en este caso serían:

Restringida ("NO ACTION"): impiden el borrado (o actualización) de una ocurrencia de EDITORIAL, en tanto existan en la base de datos libros editados por dicha editorial (es la opción que el sistema toma por defecto en el caso de no explicitar ninguna, como ocurre en el ejemplo de la figura 5.3)

Cascada ("CASCADE"): se utilizará esta opción si se desea que, al borrar una ocurrencia de EDITORIAL, se borren en la relación LIBRO todos los libros editado por ella.

Valor por defecto("SET DEFAULT"): pondríamos el valor definido por defecto, para el atributo Editorial en la tabla LIBRO en todas aquellas ocurrencia asociadas a una editorial borrada en la relación EDITORIAL.

Como podemos observar, las opciones de borrado y modificación de la clave ajena ayuda a mantener la integridad de los datos.

Cuando la interrelación es una dependencia en existencia, la transformación se realiza tal como aparece en el ejemplo de la figura 1.62

Como en este caso las ocurrencias de las entidades débil (FAMILIAR) tienen que ser eliminada cuando se borra la ocurrencia de la entidad regular (EMPLEADO) de la cual depende, la opción de borrado será cascada.

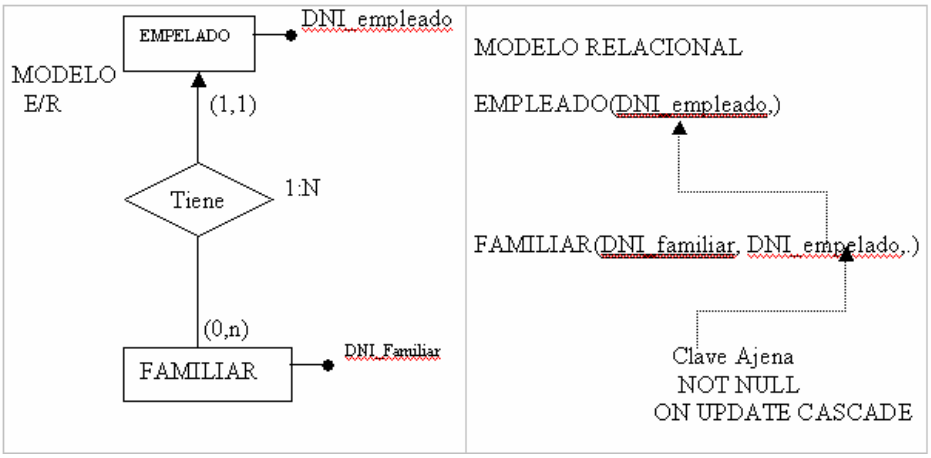


Figura 1.62 **Opción de una clave ajena en la transformación de una dependencia en existencia**

B) Transformación creando una nueva relación (Interrelación N:M)

Sea la interrelación Escribe de la figura 1.63, donde suponemos las cardinalidades que aparecen en la figura, es decir, existen libros anónimos (un libro pude no ser escrito por ningún autor) y, en cambio, todo autor tiene que haber escrito al menos un libro. El esquema relacional resultante de la transformación, con la claves ajenas y sus opciones, aparecen en la parte derecha de la figura.

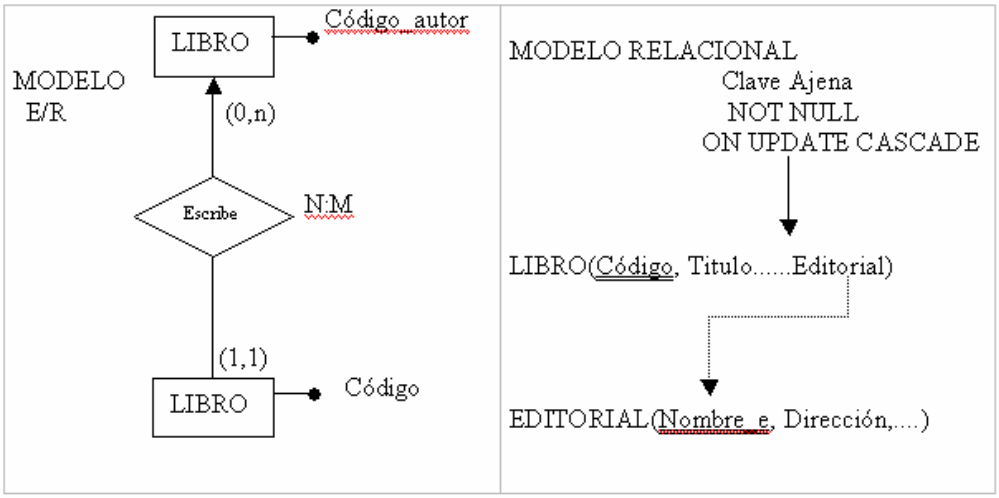


Figura 1.63 **Opciones de clave ajena en la transformación de una interrelación N:M**

Las opciones de borrado y de modificación, en el caso de interrelación N:M, suelen ser cascada, pero hay casos, en los que está justificado elegir otra opción; así, en el ejemplo de la figura, para la clave ajena Cod_autor que referencia a AUTOR no se ha puesto opción de borrado (toma, por tanto, NO ACTION), ya que se supone que no se desea que se borre un autor en tanto exista una ocurrencia de ESCRIBE que asocie a ese autor con algún libro. Las cardinalidades mínima de uno no pueden recogerse mediante las opciones de clave ajena.

Cuando la interrelación tiene un atributo, éste pasa a ser un atributo de la tabla en la que se ha transformado la interrelación.

Grafo relacional

Una forma sencilla de representar el esquema relacional es el denominador grafo relacional. Es un grafo compuesto de un conjunto de nodos multiparticionados, donde cada nodo representa un esquema de relación, es decir, una tabla de la BD. Para cada esquema de relación ha de aparecer, como mínimo, su nombre y sus atributos, indicando su clave primaria (subrayando los atributos con trazo discontinuo) y las claves ajenas (de las cuales parten arcos que señalan la tabla referenciada por la correspondiente clave ajena), tal como aparece en el ejemplo de la figura 1.64 las opciones de borrados y modificación de la clave ajena se pueden poner en el arco que une la clave ajena con la tabla referenciada.

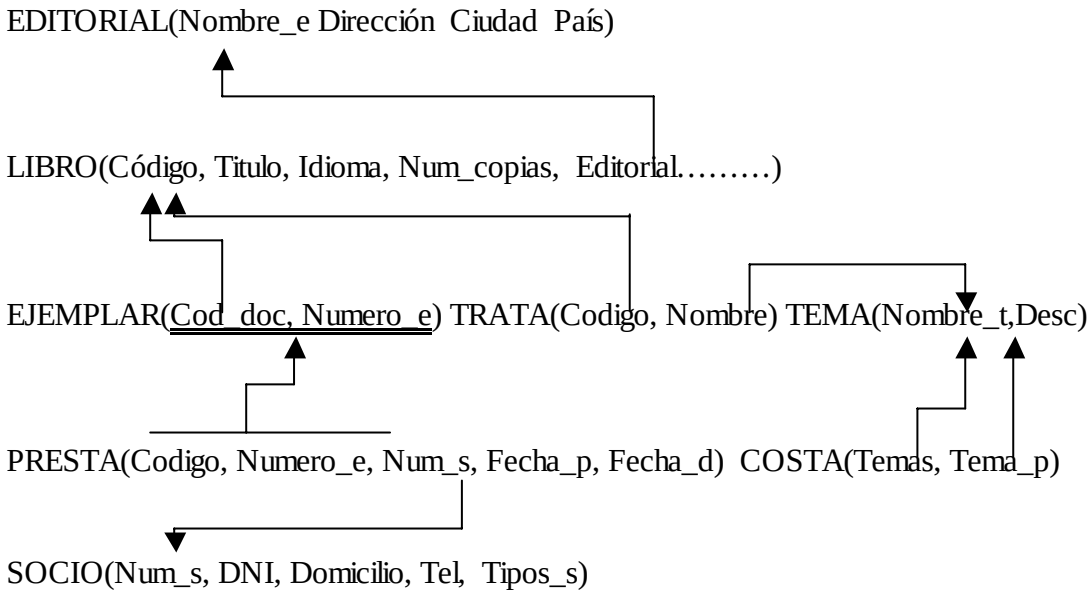


Figura 1.64 **Ejemplo de grafo relacional para el esquema de una biblioteca**

Bibliografía comentada

Fundamentos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Editorial Rama, segunda edición, Capitulo 8: Diseño lógico de las bases de datos en el modelo relacional Pag 265 - 273

9.26 Clase 26 DISEÑO DE BBDD

TEORÍA DE LA NORMALIZACIÓN

PROBLEMAS QUE PUEDE PRESENTAR UN ESQUEMA RELACIONAL

EFECTOS QUE PRODUCE LA REDUNDANCIA

NOCIONES INTUITIVA DE LAS FORMAS NORMALES

OBJETIVOS:

Identificar las ventajas que se consigue con la teoría de normalización en los modelos de datos

Identificar los problemas que puede representar un esquema relacional

9.26.1 Teoría de la normalización

El diseño de una base de datos relacional se puede realizar mediante la metodología que acabamos de exponer, aplicando al mundo real, en una primera fase, un modelo semántico como el ME/R, a fin de obtener un esquema conceptual; en una segunda fase, se transforma dicho esquema al modelo relacional mediante las correspondientes reglas de transformación. Si bien es cierto que se insiste en las ventajas de este enfoque, existen otra posibilidad que es plasmar directamente el esquema relacional sin realizar ese paso intermedio que es el esquema conceptual.

Aunque, en general, la primera aproximación produce un esquema relacional estructurado y con poca redundancia, por lo que no es imprescindible verificar la "bondad" del esquema obtenido, siempre es conveniente aplicar un conjunto de reglas, conocidas como **teoría de la normalización**, que nos permiten asegurar que un esquema relacional cumple unas ciertas propiedades. En el segundo enfoque, es decir, cuando no se ha aplicado la metodología de diseño anteriormente expuesta, la teoría de normalización resulta imprescindible.

9.26.2 Problemas que puede presentar un esquema relacional

Entre los problemas que puede presentar un esquema relacional cuando el diseño es inadecuado cabe destacar:

- Incapacidad para almacenar ciertos hechos.
- Redundancia y, por lo tanto, posibilidad de inconsistencias.
- Ambigüedades.
- Pérdida de información (aparición de tuplas espurias).
- Pérdida de ciertas restricciones de integridad que dan lugar a interdependencias entre los datos (dependencias funcionales).

Aparición de bases de datos, como consecuencia de las redundancias, de estados que no son válidos en el mundo real; es lo que se llama anomalías de inserción, borrado y modificación.

El esquema relacional debe ser, por tanto, analizado para comprobar que no presenta los problemas anteriormente citados, evitando así la pérdida de información y la aparición de inconsistencias.

Veamos un ejemplo de estos problemas derivados de un diseño inadecuado. En Tabla 1.6 se muestra la relación ESCRIBE que almacena datos sobre los libros (Cod_Libro, Título, Editorial, Año) y sobre los autores que los han escrito (Autor y Nacionalidad). Si observamos esta relación, vemos que presenta varios de los problemas enumerados anteriormente.

ESCRIBE

Autor	Nacionalidad	Cod_Libro	Título	Editorial	Año
Date, C.	Norteamericano	23433	Databases	Adisson-W	1990
Date, C.	Norteamericano	54664	SQL Standard	Adisson-W	1986
Date, C.	Norteamericano	53235	Guide to Ingres	Adisson-W	1988
Codd, E.	Norteamericano	97875	Relational M.	Adisson-W	1990
Gardarin	Frances	34245	Base de Datos	Paraninfo	1986
Gardarin	Frances	55366	Comparación de BD	Eyrolles	1984
Valduriz	Frances	86754	Comparación de BD	Eyrolles	1984
Kim, W	Norteamericano	32176	OO Databases	ACM Press	1989
Lochosvksy	Canadiense	23456	OO Databases	ACM Press	1989

Tabla 1.5 Ejemplo de diseño inadecuado

Los principales problemas de esta relación se deriva de la gran cantidad de redundancia que presenta. Por ejemplo, la nacionalidad del autor se repite por cada libro que ha escrito; algo análogo sucede, cuando un libro tiene más de una autor, con la editorial y el año publicado.

9.26.3 Efectos que produce la redundancia produce

- Anomalías de inserción, ya que dar de alta un libro obliga a insertar en la base de datos tantas tuplas como autores tenga el libro.
- Anomalías de modificación, ya que cambiar la editorial de un libro obliga a modificar todas las tuplas que corresponden a ese libro.
- Anomalías de borrado, ya que el borrado de un libro obliga a borrar varias tuplas, tantas como autores tenga ese libro y, viceversa, el borrado de un autor nos lleva a borrar tantas tuplas como libros ha escrito ese autor.

Por tanto, que la actualización (alta, baja o modificación) de un solo libro o de un solo autor nos puede obligar a actualizar más de una tupla, dejándose la integridad de la base de datos en manos del usuario. Al riesgo de la posibilidad pérdida de integridad hay que añadir la falta de eficiencia asociada a estas múltiples actualizaciones.

Además de estas anomalías de inserción, borrado y modificación, existen otros problemas adicionales, como la imposibilidad de almacenar ciertos hechos o la desaparición de información que desearíamos mantener en la base de datos. Por ejemplo si se quisiera incluir información sobre un autor del que no existiera ningún libro en la base de datos, no sería posible, ya que el atributo cod_libro forma parte de la clave primaria de la relación; ni tampoco podríamos introducir obras anónimas (recuerde la regla de integridad

que no permite los nulos en ningún atributo que forme parte de una clave primaria). Por otro lado, al dar de baja un libro, se pierden también los datos de sus autores (si esto sólo tuviesen ese libro en la base de datos) y, viceversa, si borramos un autor desaparecen de nuestra base de datos los libros escritos por él (a no ser que el libro tuviera más de un autor).

Esta relación presenta todos estos problemas debido a que atenta contra un principio básico en todo diseño:

"Hechos distinto se deben almacenar en objetos distintos"

En este caso, en relaciones distintas, con lo que se habrían evitado redundancias y, por tanto, los problemas que acabamos de describir.

Si se hubiera llevado a cabo un diseño riguroso no se habría presentado una relación de este tipo. El problema es que a menudo no se llega a comprender completamente o a representar de forma precisa el Universo del Discurso, debido a una excesiva premura al realizar el análisis o a carecer el analista de conocimientos sobre metodología de diseño de bases de datos o de experiencia para aplicarlas adecuadamente; también el problema deriva muchas veces de una falta de comunicación entre el analista y el usuario.

Si se siguiera la metodología de diseño propuesta, realizando un buen diseño conceptual en el modelo E/R, seguido de una cuidadosa transformación al modelo relacional, se evitarían en gran parte estas anomalías, obteniéndose en general un esquema exento de errores. Sin embargo, ante las posibles dudas respecto a si un determinado esquema relacional es o no correcto, será preferible aplicar siempre a dicho esquema un método formal de análisis que determine lo que pueda estar equivocado en el mismo y no permitir llegar a otro esquema en el que se asegure el cumplimiento de ciertos requisitos; este método formal, como ya hemos indicado, es la teoría de la normalización.

La teoría de la normalización evita las redundancias y las anomalías de actualización, obteniendo relaciones más estructuradas que no presenten los problemas que comentábamos anteriormente. Así, en lugar de la relación del ejemplo que aparecería en la figura 1.1 podría haber diseñado el siguiente esquema relacional:

LIBRO(Cod_libro, Título, Editorial, Año)

AUTOR(Nombre, Nacionalidad)

ESCRIBE (Cod_libro, Nombre)

Donde se ha seguido el principio básico anteriormente enunciado, separando hechos distintos en relaciones distintas, de forma que cada uno de estos esquemas de relación recogen un hecho bien determinado y concreto del mundo real con sus correspondientes atributos; esto es, evidentemente, lo que habría hecho cualquier diseñador medianamente experimentando. Pero ¿cuáles son las razones para haberlo hecho de esta manera?, ¿en qué nos podemos apoyar para afirmar que este diseño es mejor que el anterior?, ¿existen métodos matemáticos y los correspondientes algoritmos que permitan llegar a este diseño?, ¿cómo podríamos conseguir, sin basarnos en la experiencia, que un diseñador advierta los errores del primer diseño y, en lugar de la primera relación ESCRIBE, obtuviese el segundo esquema relacional con las tres relaciones LIBRO, AUTOR, ESCRIBE?

La respuesta a estas y a otras preguntas análogas se encuentran en la teoría de la normalización, la cual puede definirse como una técnica formal para organizar datos, la cual nos ayuda a determinar qué es lo que está equivocado en un diseño y nos enseña la manera de corregirlo. Por tanto, la teoría de la normalización introduce una formalización en el diseño lógico de bases de datos relacionales, lo que, además, permite mecanizar parte del proceso al poder disponer de instrumentos algorítmicos de ayuda al diseño.

Se advierte que las anomalías a las que da lugar el diseño inadecuado de una base de datos se producen sólo en proceso de actualización y nunca en los de consultas. La aplicación de la teoría de la normalización consigue una disminución de dichas anomalías, evitando muchos de los problemas que se pueden plantear en las actualizaciones. Sin embargo, al mismo tiempo penaliza las consultas al disminuir la eficiencia de las mismas, ya que cuando se aplica el proceso de normalización a una base de datos aumenta el número de relaciones, por lo que una determinada consulta puede llevar consigo el acceso a varias tablas realizando combinaciones entre ellas, lo que, indudablemente, eleva el coste de la consulta

9.26.4 Nociones intuitiva de las formas normales

La teoría de la normalización se centra en lo que se conoce como formas normales. Se dice que un esquema de relación está en una determinada forma normal si satisface un conjunto específico de restricciones.

La primera forma normal (1FN) fue introducida por Codd en su primer trabajo, es una restricción inherente al modelo relacional, por lo que su cumplimiento es obligatorio. Consiste en la prohibición de que en una relación existan grupos repetitivos, esto es, de que un atributo pueda tomar más de un valor del dominio subyacente. En realidad, se debe decir que una tabla está normalizada sólo con que se encuentre en 1FN, aunque, a menudo, esta expresión no se aplica con la debida precisión y se afirma que una tabla no está normalizada porque no está en formas normales superiores a la primera.

Codd, junto con la 1FN, definió también la segunda forma normal (2FN) y la tercera (3FN). Posteriormente, otros autores propusieron nuevas formas normales, como la Forma Normal de BOYCE y CODD (FNBC), CODD (1974), y la cuarta y quinta forma normal (4FN, 5FN) debidas a FAGFIN (1977 y 1979).

Tal como se ilustra en la Figura 1.2, de todo el universo de relaciones en 1FN, sólo algunas se encuentran en 2FN, y de éstas únicamente una parte está en 3FN, y así sucesivamente; es decir, la 2FN impone más que la 1FN, la 3FN más que la 2FN, etc., siendo la 5FN la que impone restricciones más fuerte. A fin de evitar las anomalías que describimos anteriormente es preferible la 5FN, después la 4FN, la 3FN, la 2FN y, por último la 1FN.

La teoría de la normalización se basa en ciertas restricciones definidas sobre los atributos de una relación, las cuales son conocidas con el nombre de dependencias. Existen varios tipos de dependencias, encontrándose relacionadas la 2ª , 3ª . y FNBC con las dependencia funcionales, la 4ª , con la dependencia multivaluadas y la 5ª con las depedencias de proyección-combinación.

Dentro de este enfoque intuitivo podemos decir que un esquema de relación se encuentra en 2FN si, además de estar en 1FN, todos los atributos que no forman parte de

ninguna clave candidata suministran información acerca de la clave completa, no de una parte de la clave.

Esta es la causa por la cual la relación ESCRIBE de la figura 1.2 presenta tantas anomalías, y es también el origen de los mismos problemas en la relación PRESTAMO del ejemplo que exponemos a continuación. Sea el esquema de relación:

PRESTAMO(Num_socio, Dni_socio, Cod_libro, Fecha_préstamo, Editorial, País)

Donde las claves candidatas son:

(Num_socio, Cod_libro)

(Dni_socio, Cod_libro)

Se puede observar que ciertos atributos que no forman parte de ninguna clave candidata, como Editorial, nos dan información acerca del libro pero no tienen nada que ver con el socio, por lo no constituyen una información acerca de la clave completa sino únicamente de una parte de la clave, luego la relación PRESTAMO no se encuentra en 2FN

La solución a los problemas ya expuestos que presentan esta relación es descomponer en dos:

PRESTAMO1(Num_socio, Dni_socio, Cod_libro,Fecha_préstamo)

LIBROS (Cod_libro, Editorial, País)

En el primer esquema existen dos claves (Num_socio, Cod_libro y DNI_socio, Cod_libro), y el único atributo (Fecha_préstamo) que no forma parte de ninguna clave suministra información acerca de la totalidad de ambas claves candidatas; por otra parte, en LIBROS la clave es Cod_libro, y los dos atributos que no son clave también suministran información acerca de la clave completa, ya que en este caso es simple. Por tanto, ambas relaciones se encuentran en 2FN.

De lo que acabamos de exponer se deduce que toda relación cuya clave está formada por un único atributo está, al menos, en 2FN.

Una relación se encuentra en 3FN si, además de las dos restricciones anteriores (1FN y 2FN), se cumple que los atributos que no forman parte de ninguna clave candidata facilitan información sólo acerca de la(s) clave(s) y no acerca de otros atributos.

Así por ejemplo PRESTAMO1 el atributo Fecha_préstamo sólo facilita información acerca de las claves, ya que no existe ningún otro atributo no clave, por lo que dicha relación está en 3FN. Sin embargo, en la relación LIBROS el atributo País facilitan información acerca de la Editorial, no encontrándose por tanto esta relación en 3FN.

Para resolver los problemas (redundancia, inconsistencia, etc.) que presenta esta relación, conviene descomponer en:

LIBROS1 (Cod_libro, Editorial)

EDITORIALES(Editorial, País)

Que están en 3FN, ya que todo atributo no clave sólo facilita información acerca de la clave.

En resumen, podemos decir que, para que una relación se encuentre en 3FN, ha de estar en 1FN y, además todos sus atributos no formar parte de ninguna clave candidata "deben ser información referida a la clave completa y nada más que la clave" KENT (1983).

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Capitulo 8 Diseño Lógico de las Bases de Datos en Modelo Relacional páginas273 – 280

9.27 Clase 27 DISEÑO DE BBDD

DEPENDENCIA FUNCIONALES

DEPENDENCIA FUNCIONAL PLENA O COMPLETA

DEPENDENCIA FUNCIONAL TRANSITIVA

OBJETIVOS:

Definir el concepto de dependencia funcionales plena, completa y transitiva

9.27.1 Dependencia Funcionales

Recordando que la teoría de normalización se base en el concepto de dependencia, hasta el punto de que se la conoce también como teoría de las dependencias.

Las dependencias son propiedades inherente al contenido semántico de los datos que se han de cumplir para cualquier extensión del esquema de relación y forman parte de las restricciones de usuarios del modelo relacional. La existencia de una dependencia no se puede demostrar, pero si afirmar por observación del mundo real que se trata de modelar; por tanto, del análisis de una extensión válida de un esquema de relación nunca podremos deducir la existencia de una dependencia, pero si podremos, en cambio, llegar a la conclusión de que no existe una dependencia, pero sí podremos, en cambio, llegar a la conclusión de que no existe una determinada dependencia. Por otra parte, si conocemos que una dependencia es cierta para un esquema de relación, podremos asegurar que una extensión de dicho esquema no es válida sin la cumple.

Las dependencias nos muestran algunas importantes interrelaciones existentes entre los atributos del mundo real, cuya semántica tratamos de incorporar a nuestra base de datos; son, por tanto, invariante en el tiempo, siempre que no cambie el mundo real del cual proceden.

Las dependencias funcionales son un tipo especial de dependencias, el más extendido en la práctica, en el cual se basan las primeras formas normales. A continuación se definen dichos conceptos, procurando simplificar al máximo el formalismo matemático a él asociado.

Sea el esquema de relación R definido sobre el conjunto de atributos A y sean X e Y subconjunto de A llamados descriptores. Se dice que Y depende funcionalmente de X, o lo que es igual, que X determina o implica a Y si, y sólo si, cada valor de X tiene asociado en todo momento un único valor de Y. Representamos esta dependencia funcional de la siguiente forma:

$X \rightarrow Y$

Se llama determinante o implicante al descriptor que se encuentra a la izquierda del símbolo de implicación, e implicación al descriptor que se encuentra a la derecha.

Sea, por ejemplo, la relación:

LIBRO (Cod_libro, Título, Idioma…..)

Se pude decir que el código de un libro determina el título del mismo:

Cod_libro → Título

El código del libro (Cod_libro) es el implicante (o determinante) en la anterior dependencia y Título es el implicante. Esta dependencia también nos dice que el título es una información acerca del libro, ya que una dependencia funcional se puede interpretar diciendo que el implica es un hecho (en realidad una información) acerca del implicante.

Se advierte que la afirmación de que Cod_libro determina Título no significa que conocido el código de un libro podamos deducir, a partir de él cuál es su título, a no ser que tengamos una extensión r(R) del esquema de relación que contiene la correspondiente dependencia funcional. Es decir, si para un esquema R tenemos la dependencia funcional:

X → Y

Dado el valor de X no podemos, en general conocer el valor de Y; solamente nos limitaremos a afirmar que para dos tuplas de cualquier extensión r(R) que tengan el mismo valor de X, el valor de Y será también en ambas.

Una herramienta muy útil a la hora de explicitar las dependencias funcionales es el grafo o diagrama de dependencias funcionales, mediante el cual se representa un conjunto de atributos y las dependencias funcionales existentes entre ellos. En el grafo aparecen los nombres de los atributos unidos por flechas, las cuales indican las dependencias funcionales y parten del implicante hacia el implicado. Cuando el implicante de un dependencia no es un único atributo, es decir, se trata de un implicante compuesto, los atributos que lo componen se encierran en un recuadro y la flecha parte de éste, no de cada atributo.

En la figura 1.65 se representa un ejemplo de cómo se visualizan las dependencias; podemos observar que Cod_libro determina funcionalmente el Título del libro y la Editorial, como indica la correspondencia flecha; de forma análoga, Num_socio determina el Nombre, Domicilio y Teléfono del socio (se supone que sólo nos interesa un teléfono); mientras que ambos atributos en conjunto Cod_libro y Num_socio (lo que se indica haciendo que la flecha parte del recuadro que los incluye) determinan Fecha_préstamo y Fecha_dev.

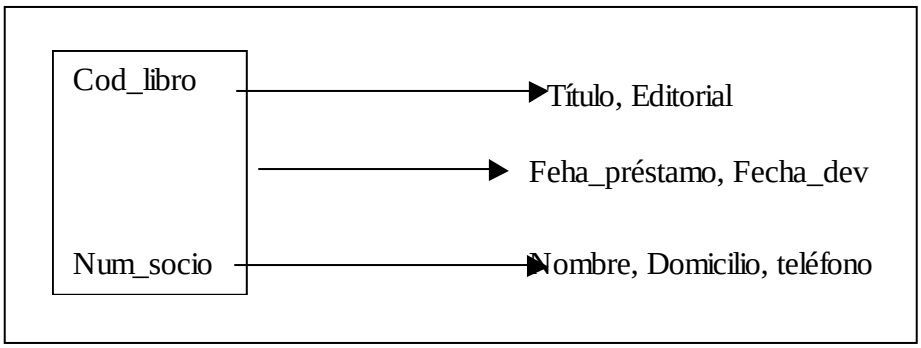


Figura 1.65 **Ejemplo de diagrama de dependencias funcionales**

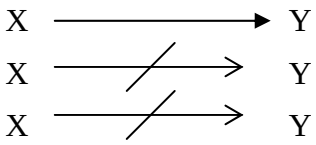
Existen otros conceptos, como el de dependencia plena o completa y el de dependencia transitiva, fundamentales en la teoría de la normalización, que veremos a continuación.

9.27.2 Dependencia funcional plena o completa

Sea un descriptor compuesto X:

X(X1,X2)

Se dice que Y tiene dependencia funcional completa o plena de X si depende funcionalmente de X pero no depende de ningún subconjunto del mismo, este es:



Lo que se representa por:

X => Y

Supongamos la relación:

PRESTAMO (Cod_libro, Título, Editorial, Num_socio, Nombre, Domicilio, Teléfono, Fecha_préstamo, Fecha_dev)

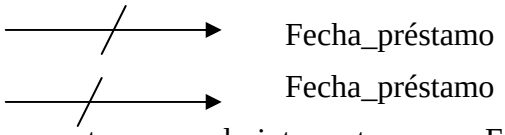
Cuyas dependencias funcionales aparecen en la figura 2.2, la dependencia funcional:

Cod_libro, Num_socio ⟶ Fecha_préstamo.

Indica que, dado un determinado código de libro y un número de socio, existe una única fecha de préstamo (para simplificar se ha supuesto que un mismo libro no se presta a un mismo socio en varias ocasiones). Ni código de libro, ni tampoco número de socio, implican, por sí solos, la fecha de préstamo, ya que tanto un libro se puede prestar en varias fechas como un socio puede recibir libros prestados en varias fechas. Por tanto, la dependencia funcional anterior es completa, lo que se representa:

Cod_libro, Num_socio => Fecha_préstamo

Ya que:



Lo que, intuitivamente, se puede interpretar como Fecha_préstamo constituye una información sobre el conjunto de libro y socio, pero esta información no atañe a un libro o a un socio por separado.

Por el contrario, la dependencia:

Cod_libro, Num_socio ⟶ Editorial

No es plena ya que:

Cod_libro, ⟶ Editorial

Se dice que, en esa dependencia, Num_socio es un atributo redundante o ajeno a la dependencia (también llamado extraño).

9.27.3 Dependencia funcional transitiva

Sea la relación

R(X,Y,Z)

En la que existen las siguientes dependencias funcionales:

X ⟶ Y

X ⟶ Y

X ⟶ Y

Se dice entonces que Z tiene una dependencia transitiva respecto de X a través de Y. lo que se representa por:

X --- --> Z

Es transitiva.

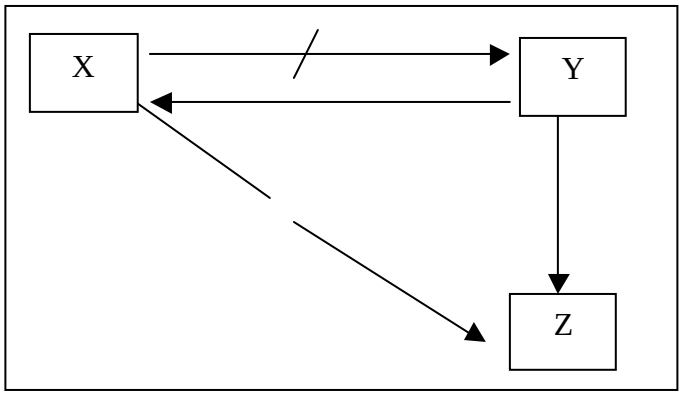


Figura 1.66 **Ejemplo de dependencia funcional transitiva**

Si consideramos la relación

LIBROS(Cod_libro, Editorial, País)

En donde, para cada libro, su código, la editorial que lo publica y el país de la editorial (se supone que una editorial tiene su sede en un único país), se tendrá las siguientes dependencias:

Cod_libro ⟶ Editorial

Editorial ⟶ País

Cod_libro ⟶ País

Además, Editorial $\not\longrightarrow$ Cod_libro (ya que una editorial puede publicar varios libros), la dependencia funcional entre Cod_libro y País es una dependencia transitiva a través de Editorial, representándola por:

Cod_libro $\longrightarrow$ País

(lo que puede interpretarse intuitivamente como que País es una información sobre el libro, pero indirectamente, ya que constituye una información sobre la editorial y ésta a su vez, sobre libro).

Además, como País $\not\longrightarrow$ Editorial, la dependencia transitiva es estricta.

En cambio si estuviera la relación:

LIBRO (Cod_libro, Título, Editorial)

Con las siguientes dependencias:

Cod_libro $\longrightarrow$ Título

Título $\longrightarrow$ Cod_libro

Cod_libro $\longrightarrow$ Editorial

Título $\longrightarrow$ Editorial

Ninguna de las dependencias

Cod_libro $\longrightarrow$ Editorial y Título $\longrightarrow$ Editorial son transitivas, al ser los atributos Cod_libro y Título equivalentes (ambos se implican mutuamente).

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Capitulo 8 Diseño Lógico de las Bases de Datos en Modelo Relacional páginas280 - 285

9.28 Clase 28 DISEÑO DE BBDD

DEFINICIÓN FORMAL DE LAS TRES PRIMERAS FORMAS NORMALES

DESCOMPOSICIÓN DE RELACIONES

DESCOMPOSICIÓN SIN PÉRDIDAS DE INFORMACIÓN

OBJETIVOS:

Definir las tres primeras formas normales

Descomponer las relaciones

Descomponer sin perdida de información

9.28.1 Definición formal de las tres primeras formas normales

Expuestos ya el concepto de dependencia funcional, podemos formalizar la definición de segunda y tercera forma normal apoyándose en la dependencia funcional plena y transitiva.

Primera forma normal (1FN).

Recordemos que para que una tabla pueda ser considera una relación no debe admitir grupos repetitivos, esto es, debe estar en primera forma normal. En el ejemplo de la figura 2.4 se observa, en la parte a), grupos repetitivos en aquellas libros más de un autor; para pasar esta tabla (no es una relación puesto que no está en 1FN) a 1FN, habrá que repetir el resto de atributos de la tupla para cada uno de los valores del grupo repetitivo, tal como aparece en la parte b) de la figura. Se dice que una relación está en 1FN si cada atributo toma un único valor del dominio subyacente.

a)

LIBRO(Código, Título, Autor)

CODIGO	TITULO	AUTOR
02154	Data models	Tsichiritzis Lochovsky
87654	A guide to DB2	Date
65465	Bases de datos	Gardiarin Valduriez

NO ESTA EN IF

Hay grupos repetitivo

b)

LIBRO (Código, Título, Autor)

CODIGO	TITULO	AUTOR
02154	Data models	Tsichritzis
02155	Data models	Lochovsky
87654	A guide to DB	Date
87655	Base de datos	Gardarin
87656	Base de datos	Valduriz

ESTA EN IF

Tabla 1.7 Tabla con grupos repetitivos y relación en 1FN

Segunda forma normal (2FN)

Se dice que una relación está en 2 FN si:

Está en 1FN.

Cada atributo no principal tiene dependencia funcional completa respecto de cada una de las claves.

Por ejemplo, si tenemos la relación PRESTAMO (Cod_libro, Num_socio, Editorial), en la que existe la siguiente dependencia funcional:

Cod_libro ⟶ Editorial

Y cuya clave está constituida por el descriptor Cod_libro, Num_socio, esta relación no se encuentra en 2FN al venir la editorial determinada sólo por el código de libro y no por la clave completa.

Tercera Formal (3FN)

se dice que una relación está en 3FN si:

Está en 2FN.

No existe ningún atributo no principal que dependa transitivamente de alguna de las claves de relación.

Sea, por ejemplo, la relación:

SOCIO(Num_socio, Ciudad, País)

Que presenta las siguientes dependencias funcionales:

Num_socio ⟶ Ciudad

Ciudad ⟶ País

Y cuya clave es, evidentemente, Num_socio. Dicha relación no se encuentra en 3FN, al depender País transitivamente de la clave a través de Ciudad.

Tanto la relación PRESTA como la relación SOCIO presentan los problemas citados anteriormente. Si estas relaciones se llevasen a formas normales más avanzadas se eliminarían, o al menos se atenuarían, estos problemas.

9.28.2 Descomposición de relaciones

La transformación de una relación que se encuentra en una determinada forma normal, en otra relación cuya forma normal es superior se realiza por medio del operador de proyección del álgebra relacional.

Sea, por ejemplo, la relación:

PRESTA(Cod_libro, Num_socio, Editorial)

Que, como hemos visto se encuentra sólo en 1Fn (ya que su único atributo no Principal, Editorial, no depende totalmente de la clave). Si quisiéramos llevar esta relación a una forma normal más avanzada, sería preciso descomponerlas, mediante proyección, obteniendo varias relaciones. Así, proyectando sobre Cod_libro, Num_socio y sobre Cod_libro, Editorial, tendríamos:

PRESTA1(Cod_libro, Num_socio)

PRESTA2(Cod_libro, Editorial)

Ambas relaciones están en 3FN y han desaparecido las redundancias y las inconsistencias a las que antes aludíamos.

Además, la combinación natural de PRESTA1*PRESTA2 (por el atributo común Cod_libro) nos devuelven la relación original.

De manera análoga, la relación:

SOCIO(Num_socio, Ciudad, País)

Que se encuentra en 2FN, pero no en tercera, podría descomponerse en dos proyecciones.

SOCIO1(Num_socio, Ciudad)

CIUDAD(Ciudad, País)

También, al igual que en el caso anterior:

SOCIO * CIUDAD = SOCIO

En el proceso de descomposición de relaciones, para llevarlos a formas normales más avanzadas es preciso cumplir unas determinadas reglas:

- Descomposición sin pérdidas de información

Se dice que una descomposición se ha realizado sin pérdida de información cuando la combinación natural de las proyecciones resultantes no devuelve la relación original.

Sea la relación:

LIBRO(Cod_libro, Editorial, País)

En la cual tienen lugar las siguientes dependencias funcionales:

Cod_libro ⟶ Editorial

Editorial ⟶ País

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Capitulo 8 Diseño Lógico de las Bases de Datos en Modelo Relacional páginas285 - 288

9.29 Clase 29 DISEÑO DE BBDD

EJEMPLO DE NORMALIZACIÓN

OBJETIVOS:

Aplicar la normalización a entidades propuestas

Una extensión de esta relación aparece en la a) de la figura 2.5

A) LIBRO

COD_LIBRO	EDITORIAL	PAIS
654654	Ra-Ma	España
665465	Ra-Ma	España
876545	Paraninfo	España
987456	Anaya	España
965842	Addison-w	E.E.U.U

B) LIBRO1

COD_LIBRO	PAIS
654654	España
665465	España
876545	España
987456	España
965842	E.E.U.U

EDITORIAL1

EDITORIAL	PAIS
Ra-Ma	España
Ra-Ma	España
Paraninfo	España
Anaya	España
Addison-w	E.E.U.U

C) LIBRO1 * EDITORIAL1

COD_LIBRO	EDITORIAL	PAIS
654654	Ra-ma	España
654654	Paraninfo	España
654654	Anaya	España
665465	Ra-ma	España
665465	Paraninfo	España
665465	Anaya	España
876545	Ra-ma	España
876545	Paraninfo	España
876545	Anaya	España
987456	Ra-ma	España
987456	Paraninfo	España
987456	Anaya	España
965842	Addison-w	E.E.U.U

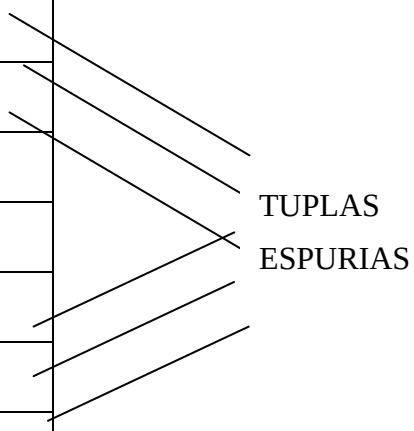


Tabla 1.8 Descomposición con pérdida de información (aparición de tuplas espurias)

Supongamos que descomponemos esta relación en:

LIBRO(Cod_libro, País)

EDITORIAL(Editorial, País)

Cuya extensión aparecen en la parte b) de la figura. La combinación de estas dos relaciones (ver parte c) de la figura) da lugar a la aparición de tuplas espurias que no se encontraban en la extensión original. Se dice que la descomposición de LIBRO ha dado lugar a pérdida de información.

Si en lugar de esta descomposición hubiéramos obtenidos:

LIBRO2(Cod_libro, País)

EDITORIAL2(Cod_libro, Editorial)

Es fácil comprobar que la combinación natural.

LIBRO2 * EDITORIAL2

Cod_libro

Daría la relación original (sin tuplas espurias).

La condición necesaria y suficiente para que una descomposición se produzca sin pérdida de información es que el atributo común de las dos relaciones sea clave, al menos, en una de ellas; condición que no se cumple en la primera descomposición, pero si en la segunda.

9.29.1 Descomposición sin pérdida de dependencia funcional

Las dependencias funcionales recogen la semántica del mundo real, por lo que es conveniente conservarlas en el proceso de descomposición.

Sea la misma relación LIBRO del ejemplo anterior que hemos descompuesto en LIBRO2 y EDITORIAL2, donde no ha habido pérdida de información, dado que el atributo común (Cod_libro) es clave en ambas relaciones.

Sin embargo, en esta descomposición se ha perdido una dependencia funcional, ya que en la relación LIBRO la única dependencia funcional es:

Cod_libro $\longrightarrow$ País

y en la EDITORIAL2, la única dependencia es:

Cod_libro $\longrightarrow$ Editorial

Luego la dependencia Editorial $\longrightarrow$ País se ha perdido, y con ella también ha desaparecido en nuestro esquema parte de la semántica del mundo real, que nos dice que, dada una editorial, ésta se encuentra en único país.

- Descomposición en proyección independientes

La descomposición de una relación R en un conjunto de relaciones (R) se dice que se ha realizado en proyección independiente si no ha habido pérdida de información ni de dependencias funcionales.

Si la relación LIBRO de nuestro ejemplo la hubiésemos descompuesto en:

LIBRO3(Cod_libro, Editorial)

EDITORIAL3(Editorial, País)

Estas dos proyecciones serían independiente, ya que el atributo común es clave de una de las relaciones (la segunda), por lo que no hay pérdida de información; además, tampoco hay pérdida de dependencia funcionales, dado que en LIBRO3 tenemos la dependencia:

Cod_libro $\longrightarrow$ Editorial

Y en EDITORIAL3, tenemos:

Editorial $\longrightarrow$ País

Se trata de la mejor descomposición: la relación resultante son equivalente a la relación original y, en ellas, se han eliminado las anomalías de inserción, borrado y

modificación (dado que sólo se ha llegado a 3FN no se asegura que, en todos los casos, se eliminen por completo las anomalías).

Se puede demostrar que es posible descomponer cualquier relación, llevándola a 3FN, sin pérdida de información ni de dependencias funcionales (cosa que no siempre ocurre si se quisiese llevarla a formas normales más avanzadas)

A veces, el proceso de normalización se aplica a la denominada relación universal, constituida por el conjunto de atributos de universo del discurso que se desea modelar y por el conjunto de sus dependencias funcionales. Además del método de descomposición, también llamado análisis, cuyos fundamentos acabamos de exponer, existe otro método para normalizar una relación que es el método de síntesis (el cual no abarca este trabajo).

Ejemplo

Bibliografía comentada

Fundamentos y modelos de BASES de DATOS, Adoración de Miguel, Mario Piattini. Capitulo 8 Diseño Lógico de las Bases de Datos en Modelo Relacional páginas289 – 291

10 Prácticas de laboratorio

10.1 Laboratorio I

Guía de laboratorio

10.1.1 Tema: Requerimientos:

Fases de requerimientos

- F1. Nivel de abstracción apropiado para los términos escogidos
- F2. Evitar usar "instancias" (personalizaciones) en lugar de conceptos generales.
- F3. Control de Sinónimos y homónimos
- F4. Hacer explícitas las relaciones entre términos
- F5. Glosario de términos

Términos Ambiguos con correcciones

10.1.2 Competencia o Habilidad

Este Laboratorio pretende que usted adquiera la competencia necesaria para "Analizar los requerimientos de un problema utilizando las 5 Fases en el Análisis de Requerimiento"

Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

Tras la lectura de información alusiva y su experiencia personal, determinará los requerimientos necesarios para el diseño del modelo.

Dado el planteamiento de un problema, analizará usando las diferentes fases de requerimiento para identificar ambigüedades, sinónimos que puedan oscurecer el planteamiento del problema.

Fomentar la participación activa del grupo mediante la solución adecuada de las distintas fases del diseño de requerimiento.

Actividades de Aprendizaje

1. Lea los objetivos aquí propuestos, desvanezca dudas y no inicie el estudio del material si no tiene claro que competencia y objetivos se persiguen y como lograrlos.
2. Lea el enunciado del problema "Pesca Deportiva", el texto presenta un caso sobre la pesca deportiva con varios supuestos, después de leerlo y revisado el ejemplo de análisis de requerimiento, usted analizará la situación, primero de manera individual y después en grupo. Durante el análisis del caso irá resolviendo cada una de las fases de requerimiento con ayuda del grupo y del profesor.
3. Revise el ejemplo propuesto y analizar las respuestas dadas para cada una de las fases.

4. Resolverá cada una de la fases del análisis de requerimiento plasmando todas las situaciones que encuentre en el planteamiento del problema. Para lograr una mejor comprensión del mismo.
5. Tiempo aproximado 4 horas
6. Al final se realizará la evaluación del laboratorio, mediante la entrega de su Reedición del enunciado del problema.

Modelo conceptual usando E/R

Requerimientos:

Pesca Deportiva II

A.1 Enunciado del problema.

La asociación de pesca deportiva "Fishermen Team" a la vista del buen resultado que ha dado el sistema de información desarrollado y el interés de sus afiliados en tener más información disponible para la práctica y seguimiento de la pesca deportiva, desea ampliar el mismo.

En este sentido, Fishermen Team desea mantener información sobre los siguientes objetos:

Los tipos de licencias de pesca que se expiden en el territorio nacional y que son exigidos en los diferentes lugares de pesca para la práctica de la misma sea en un evento o no.

- Las casas suministradoras de los productos de pesca, algunas de las cuales hacen ofertas especiales a la asociación.
- Los cebos utilizados en la captura de los peces, tanto en los eventos deportivos como fuera de ellos.
- Los cebos permitidos para la práctica de la pesca en cada uno de los lugares de pesca.
- La ampliación que requiere Fishermen Team es bastante completa, aportando para ello la siguiente información sobre el problema planteado:

Supuesto 1: las licencias son expedidas por las comunidades autónomas y sólo dan permisos para la práctica de la pesca en cualquier cauce fluvial de esa comunidad.

Supuesto 2: una licencia da permiso para la pesca de cualquier especie piscícola en el territorio de la comunidad, aunque existen determinadas especies piscícolas como la trucha, salmón, etc., que requieren una licencia o suplemento especial que debe obtenerse además de la general.

Supuesto 3: No en todas las comunidades autónomas existen licencias, o recargos en las mismas, para la pesca de determinadas especies, debido a que en sus cauces fluviales no están presentes algunas de sus especies psicícolas.

Supuesto 4: Los cebos utilizados en la pesca se clasifican en naturales y artificiales. Los cebos naturales agrupan a cualquier animal o vegetal utilizado como señuelo para la

captura de un pez. Por otro lado, los cebos artificiales son engaños no vivos realizados para el mismo fin.

Supuesto 6*:* Algunos fabricantes ofrecen cada grupo de cebos artificiales en distinto tamaño y color, información que es interesante conocer.

Supuesto 5: Los cebos artificiales están clasificados en familias y dentro de éstas existen grupos de cebos que tienen un nombre genérico. Para cada uno de estos grupos de cebos existen muchos fabricantes de los mismos, los cuales, además del nombre del grupo le "bautizan" con un nombre genérico o del propio fabricante.

Supuesto 7: Los cebos naturales, a su vez, se clasifican en animales, vegetales y mixtos. Si bien los cebos animales consideran solo a una especia animal, los cebos vegetales están, generalmente, formados por un conjunto de vegetales en una proporción determinada. Los cebos mixtos están formados por uno o un conjunto de cebos vegetales a los cuales se les añaden uno o varios cebos animales.

Supuesto 8: A la asociación le interesa también mantener información de los cebos vegetales utilizados, o no, por sus afiliados. Su composición y modo de preparación son las recetas mágicas de todo pescador que se precie.

Supuesto 9: En la captura de un pez interesa conocer el cebo utilizado en la misma. Se considera que un pez puede ser capturado haciendo uso de uno o más cebos.

Supuesto 10: Existen una serie de comercios destinados a la venta de artículos para la practica de la pesca. Solo interesa conocer, en estos momentos(para este problema), la información correspondiente a los mismos, y si tienen descuentos especiales para los afiliados de la asociación.

Reescritura de requerimientos:

Fases del primer análisis de requerimientos

F1. Nivel de abstracción apropiado para los términos escogidos:

Los términos abstractos que se encuentra en nuestro problema, se te pide que encuentres los otros que hacen falta.

Términos	Deberían aparecer
Ofrecen	Fabrican
A la vista	Envista del
Objetos	Aspectos

Completación la Respuesta

Expiden	que se otorgan
Casas suministradoras	casas distribuidoras
Cebos	carnada

F2. Evitar usar "instancias" (personalizaciones) en lugar de conceptos generales:

Respuesta

Términos Quiere decir

- Fuera de ellos pesca libre

F3. Control de Sinónimos y homónimos:

Recordando sinónimo se refieren palabras diferentes con el mismo concepto ejemplo

Cebos y engañadores

Respuesta

- Suplemento especial y licencia especial
- Recargos en la misma y licencia especial
- Son las recetas y constituyen las recetas

Homónimos:

Licencia, para referirse a licencia general y licencia especial suplementaria

F4. Hacer explícitas las relaciones entre términos.

"El mismo", se refiere al sistema de información desarrollado o al seguimiento de pesca deportiva.

F5. Glosario de términos:

Ejemplo del glosario de termino encuentre los que hacen falta.

Precie: que se las da de buen pescador.

Evento: Concurso, competencia, acontecimiento, suceso imprevisto

Respuesta

Cauce: Lecho de los ríos y arrollos. Por donde corren las aguas

Fluvial: Relativo al agua de lluvia, el régimen pluvial de nuestra nación. 2/ Capa pluvial: ornamento que usan los sacerdotes en ciertas solemnidades religiosas.

Términos Ambiguos con correcciones:

Ejemplo de términos ambiguos

Términos	Nuevo Término	Razón para la corrección
Ofrecen	Fabrican	Ofrecen es demasiado genérico
A la vista	Envista del	Es mas especifico que el primero
Objetos	Aspectos	Es mas específico que el primero
Expedidas	Otorgadas	Es mas específico que el primero
Casas suministradoras	Casas distribuidoras	Es mas especifico que el primero

Respuesta

Cebos	Carnada, engaño	Son sinónimos
Fuera de ellos	Pesca libre	Es mas especifico que el primero
Suplemento especial	Licencia especial	Se refieren al mismo concepto
Recargos en la misma	Licencia especial	Se refieren al mismo concepto
El mismo	Al sistema de información	Más específico
Son las recetas	Constituyen las recetas	Son sinónimos

10.1.3 Reedición del enunciado del problema:

- La asociación de pesca deportiva “Fishermen Team” en vista del buen resultado que ha dado el sistema de información desarrollado y el interés de sus afiliados en tener más información disponible para la práctica y seguimiento de la pesca deportiva, desea ampliar el Sistema de información.
- En este sentido, Fishermen Team desea mantener información sobre los siguientes Aspectos:
- Los tipos de licencias de pesca que se expiden en el territorio nacional y que son exigidos en los diferentes lugares de pesca para la práctica de la misma sea en un evento o no.
- Las casas suministradoras o distribuidoras de los productos de pesca, algunas de las cuales hacen ofertas especiales a la asociación.
- Los cebos utilizados en la captura de los peces, tanto en los eventos deportivos como fuera de ellos (pesca libre).

- Los cebos permitidos para la práctica de la pesca en cada uno de los lugares de pesca.

- La ampliación que requiere Fishermen Team es bastante completa, aportando para ello la siguiente información sobre el problema planteado:

Supuesto 1: las licencias son otorgadas por las comunidades autónomas y sólo dan permisos para la práctica de la pesca en cualquier cauce fluvial de esa comunidad.

Supuesto 2: una licencia da permiso para la pesca de cualquier especie piscícola en el territorio de la comunidad, aunque existen determinadas especies piscícolas como la trucha, salmón, etc., que requieren una licencia especial o suplemento especial que debe obtenerse además de la general.

Supuesto 3: No en todas las comunidades autónomas existen licencias, o recargos (suplemento especial) en las mismas, para la pesca de determinadas especies, debido a que en sus cauces fluviales no están presentes algunas de sus especies piscícolas.

Supuesto 4: Los cebos utilizados en la pesca se clasifican en naturales y artificiales. Los cebos naturales agrupan a cualquier animal o vegetal utilizado como señuelo para la captura de un pez. Por otro lado, los cebos artificiales son engaños no vivos realizados para el mismo fin.

Supuesto 6*:* Algunos fabricantes fabrican cada grupo de cebos artificiales en distinto tamaño y color, información que es interesante conocer.

Supuesto 5: Los cebos artificiales están clasificados en familias y dentro de éstas existen grupos de cebos que tienen un nombre genérico. Para cada uno de estos grupos de cebos existen muchos fabricantes de los mismos, los cuales, además del nombre del grupo le "bautizan" con un nombre genérico o del propio fabricante.

Supuesto 7: Los cebos naturales, a su vez, se clasifican en animales, vegetales y mixtos. Si bien los cebos animales consideran solo a una especia animal, los cebos vegetales están, generalmente, formados por un conjunto de vegetales en una proporción determinada. Los cebos mixtos están formados por uno o un conjunto de cebos vegetales a los cuales se les añaden uno o varios cebos animales.

Supuesto 8: A la asociación le interesa también mantener información de los cebos vegetales utilizados, o no, por sus afiliados. Su composición y modo de preparación constituyen las recetas mágicas de todo pescador que se precie.

Supuesto 9: En la captura de un pez interesa conocer el cebo utilizado en la misma. Se considera que un pez puede ser capturado haciendo uso de uno o más cebos.

Supuesto 10: Existen una serie de comercios destinados a la venta de artículos para la practica de la pesca. Solo interesa conocer, en estos momentos(para este problema), la información correspondiente a los mismos, y si tienen descuentos especiales para los afiliados de la asociación.

10.2 Laboratorio II

Guía de laboratorio

Tema: Descomposición del texto en sentencias divididas por concepto.

Representar graficamente un diagrama las sentencias en un digrama E/R Previo.

Competencia o habilidad

Desarrolle su habilidad para descomponer un texto en sentencia dividas por conceptos, representar los conceptos en un diagrama de Entidad Relación Previo.

Objetivos de Aprendizaje

Como fruto de las actividades de aprendizaje prevista se pretende que usted podrá:

Descomponer un texto en sentencia dividas por conceptos.

Representar gráficamente los conceptos usando las técnicas del diagrama Entidad Relación previo

Actividades de Aprendizaje

- Iniciar el estudio de este laboratorio aclarándose sobre las competencias y objetivos de aprendiza si tiene dudas, consulte al profesor.
- Identificar en el texto planteado las sentencia y las dividirá por conceptos
- Revise el ejemplo propuesto y analice las divisiones de sentencias realizadas.
- Representar los conceptos encontrados en un diagrama Entidad Relación previo.
- Tiempo aproximado 4 horas
- Al final se realizará la evaluación del laboratorio, mediante la entrega de la división sentencias por conceptos y el modelo Entidad Relaciona para dichos conceptos.

F6. Descomponer el texto en sentencias divididas por concepto

Sentencias Generales

Sentencias sobre Afiliados (pescador):

El interés de lus afiliados en tener más información disponible para la práctica y seguimiento de la pesca deportiva, desea ampliar el Sistema de información.

2_____

3. _____

4._____.

Respuesta

2. Sentencias sobre **Licencias:**

Los tipos de licencias de pesca que se expiden en el territorio nacional y que son exigidos en los diferentes lugares de pesca para la práctica de la misma sea en un evento o no.

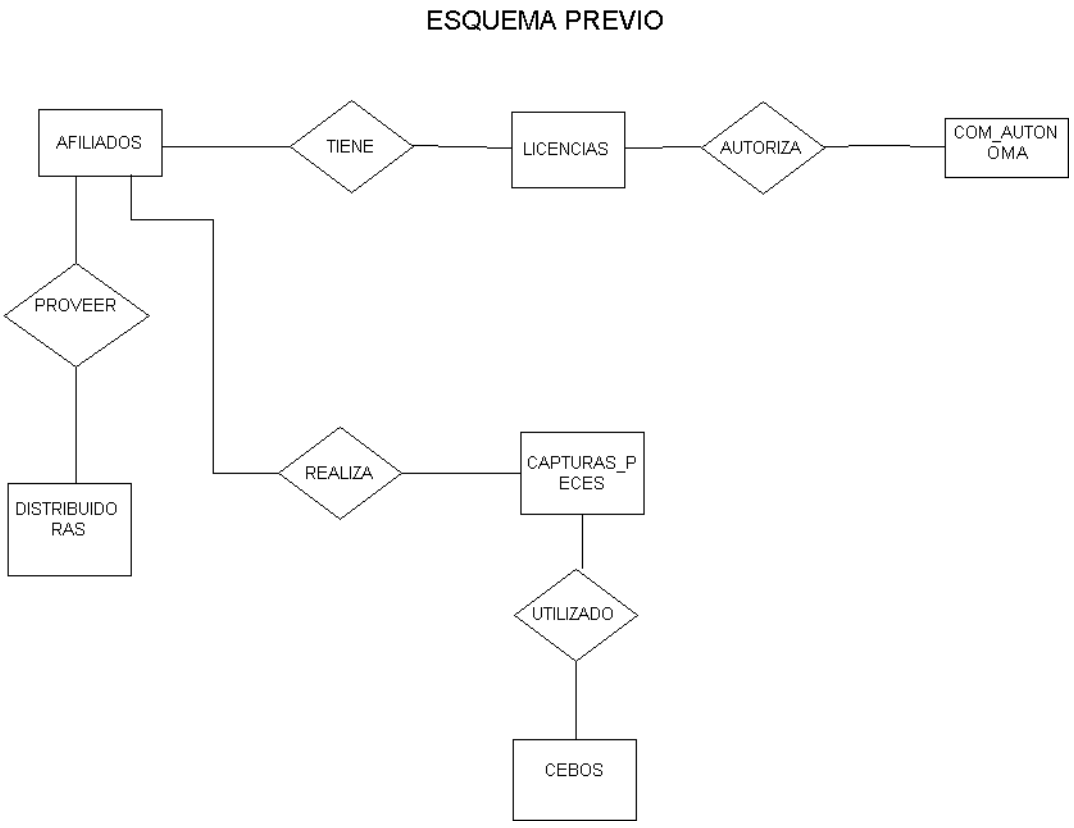
3.Sentencias sobre **Las distribuidoras**:

Las casas suministradoras o distribuidoras de los productos de pesca, algunas de las cuales hacen ofertas especiales a la asociación.

4. Sentencias sobre los **cebos o carnadas**:

Los cebos utilizados en la captura de los peces, tanto en los eventos deportivos como fuera de ellos (pesca libre).

Diagrama E/R:



10.3 Laboratorio III

Guía de laboratorio

10.3.1 Tema: Refinamiento del esquema previo, usando las técnicas:

Refinamiento Top – Down

Refinamiento Inicide -Out

10.3.2 Competencia o habilidad

Se pretende que adquiera la competencia siguiente.

Usar técnicas de refinamiento Top-Down y Inicide – Out atendiendo sus potenciales dentro del análisis de requerimientos.

Objetivos de Aprendizaje

Como resultado de las actividades previstas en el laboratorio usted.

- Aplicará técnicas de refinamiento al esquema previo garantizando una mejor compresión del problema.
- Actividades de Aprendizaje
- Iniciar el estudio de este laboratorio aclarándose sobre las competencias y objetivos de aprendiza si tiene dudas, consulte al profesor.
- Aplique técnicas de refinamiento a los conceptos que por sus características pueden ser refinado.
- Tiempo aproximado 2 horas
- Al final se realizará la evaluación del laboratorio, mediante la entrega del refinamiento de los conceptos.

Desarrollo

Para refinar el esquema previo y especificar todos los requerimientos, usaremos las técnicas:

Ejemplos de refinamiento Top-Down.

Refinamiento Top-Down.

a. La entidad LICENCIAS, se puede refinar en dos subentidades, LICENCIAS_GRAL y LICENCIA_ESPEC
b. La entidad CEBOS, se puede refinar en dos subentidades, CEBOS_NAT y CEBOS_ART.

Completación de la Respuesta

a) A su vez la entidad CEBOS_ART, se puede refinar en dos subentidades, FAMILIA_CEB y GRUPOS_CEB

b) De la misma forma, la entidad CEBOS_NAT, se puede refinar en tres subentidades, CEBOS_ANIMALES , CEBOS_VEGETALES y CEBOS_MIXTOS

Ejemplos de refinamiento Incide - Out

Refinamiento Incide-Out.

a. Agregamos la entidad FABRICANTE quien se relaciona con FABRICA con la entidad CEBOS.

b. Agregamos la entidad VARIEDAD_PECES quien se relaciona con CAPTURAS_PECES con la relación PERTENECE_A

Completación la Respuesta

a. Integramos una nueva relación entre la entidad COM_AUTONOMA y la entidad CAPTURAS_PECES, denominada REALIZADO_EN

b. Además integramos una nueva relación entre la entidad LICENCIA_ESPEC con la entidad VARIEDAD_PECES, con la relación AUTORIZADO_PARA, que permite conocer sobre que variedades de peces se expiden las Licencias Especiales o con recargo.

Una de las propiedades de COM_AUTONOMA, es LUGAR_PESCA y sobre ella se asocian muchos nombres, direcciones etc. Representaresmo LUGAR_PESCA como una entidad y luego representar los enlaces entre ambas entidades por medio de la relación PERTENECE_A

10.4 Laboratorio IV

Guía de laboratorio

10.4.1 Tema: Esquema Final del modelo Entidad / Relación

10.4.2 Competencia o habilidad

Desarrolle su habilidad para diseñar modelo Relacional, aplicando sus conocimientos sobre modelado de lo conceptos, atributos y relaciones.

Objetivos de Aprendizaje

Al terminar este laboratorio, se espera que usted será capaz de:

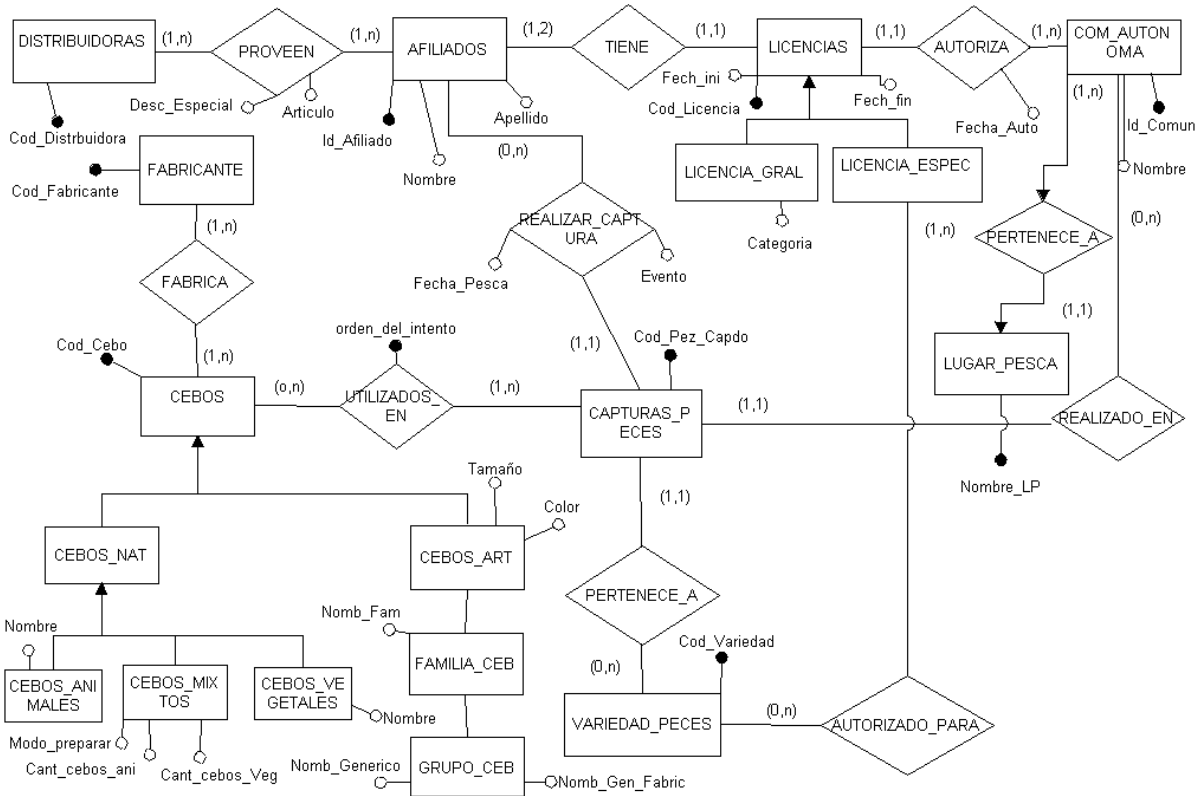
- Representar conceptos, atributos y relaciones en un modelo Entidad / Relación

Actividades de Aprendizaje

- Iniciar el estudio de este laboratorio aclarándose sobre las competencias y objetivos de aprendiza si tiene dudas, consulte al profesor.
- Represente los conceptos, atributos y relaciones en un modelo Entidad / Relación
- Tiempo aproximado 4 horas
- Al final se realizará la evaluación del laboratorio, mediante la entrega del esquema final.

Desarrollo.

ESQUEMA FINAL



10.5 Laboratorio V

Guía de laboratorio

10.5.1 Tema: Documentación Adicional

Descripción de Jerarquías

Documentación de entidades

Documentación de relaciones

10.5.2 Competencia o habilidad

Desarrolle su habilidad para describir las jerarquia de las entidades y documentar las entidades y relaciones.

Objetivos de Aprendizaje

Identificar las jerarquias de las entidades presentes en el modelo.

Documentar las entidades y las relaciones con su descripción y cardinalidades.

Actividades de Aprendizaje

Iniciar el estudio de este laboratorio aclacrándose sobre la competencia y objetivos de aprendizaje, si tiene dudas, consulte al profesor.

- Describir las jerarquías de las entidades.
- Documentar las entidades y relaciones presentes en el modelo.
- Tiempo Aproximado 6 horas.
- Al final se realizará la evaluación del laboratorio, mediante la entrega de las descripciones y jerarquías encontradas en las entidades, y la docuementación de estas.

Desarollo

1.- DISTRIBUIDORAS

Descripción: Empresas que comercializan productos para la practica de la pesca.

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Distribuidora	Es un código que permite identificar y diferenciar a cada uno de los centros de distribución que los afiliados adquieren productos de pesca. Se podría usar el nombre de las Distribuidora, también.	Sí	1,1

2.- FABRICANTE

Descripción: Son empresas que se dedican a producir productos para la practica de la pesca, los cuales son comercializados por las empresas Distribuidoras, donde los afiliados adquieren los artículos.

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Fabricante	Es un código que permite identificar y diferenciar a cada una de las fabricas que producen artículos para pesca. Se podría usar el nombre de los Fabricantes también.	Sí	1,1

3.- CEBOS

Descripción: Son los son los señuelos que se utilizan para la actividad de pescar, en este caso es una entidad que generaliza los tipos de cebos CEBOS_NAT y CEBOS_ART

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos que se utilizan en la actividad de la pesca.	Sí	1,1

4.- CEBOS_NAT:

Descripción: Son los son los señuelos que se utilizan para la actividad de pescar, pero los de este tipo son los de origen Naturales. En este caso es una entidad que generaliza los tipos de cebos naturales CEBOS_ANIMALES, CEBOS_VEGETALES y CEBOS_MIXTOS

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos naturales que se utilizan en la actividad de la pesca.	Sí	1,1

5.- CEBOS_ART

Descripción: Son los son los señuelos que se utilizan para la actividad de pescar, pero los de este tipo son los de origen Vegetal. En este caso es una entidad que generaliza los tipos de cebos FAMILIA_CEB y GRUPOS_CEB

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos vegetales que se utilizan en la actividad de la pesca.	Sí	1,1
Tamaño	Son las dimensiones o longitud de los cebos artificiales	No	1,1
Color	Es el color predominante que tienen los cebos artificiales.	No	1,1

Actividad a realizar describir las jerarquia de las entidades siguientes:

CEBOS_ANIMALES,CEBOS_VEGETALES, CEBOS_MIXTOS, FAMILIA_CEB

6.- CEBOS_ANIMALES

Descripción: Son los son los señuelos de origen animal que se utilizan para la actividad de pescar, en este caso es compuesto por un solo tipo de animal

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos de origen animal que se utilizan en la actividad de la pesca.	Sí	1,1
Nombre	Es el nombre con el cual del animal que se constituye como un cebo para la pesca.	No	1,1

7.- CEBOS_VEGETALES

Descripción: Son los son los señuelos de origen vegetal que se utilizan para la actividad de pescar, en este caso puede estar compuesto por un conjunto de vegetales en una proporción determinada.

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos de origen vegetal que se utilizan en la actividad de la pesca.	Sí	1,1
nombre	Es el nombre con el cual se conoce el tipo de cebo, el cual representa a a los vegetales que participan en mayor proporción.	No	1,1

8.- CEBOS_MIXTOS

Descripción: Son los son los señuelos mixtos de origen vegetal y animal que se utilizan para la actividad de pescar, en este caso puede estar compuesto por un conjunto de cebos vegetales y un conjunto de cebos animales.

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos de origen mixtol que se utilizan en la actividad de la pesca.	Sí	1,1
Cant_cebos_ani	Representa la cantidad de cebos animales por lo que esta compuesto.	No	1,1
Cant_Cebos_veg	Representa la cantidad de cebos vegetales por lo que esta compuesto.	No	1,1
Modo_preparar	Indica los procedimientos que se utiliza para la preparación del cebo mixto	No	1,1

9.- FAMILIA_CEB

Descripción: Son los son los señuelos de origen artificial que se clasifican en familias que se utilizan para la actividad de pescar.

Atributos:

Nombre	Descripción	Clave	Card.
Cod_Cebo	Es un código que permite identificar y diferenciar a cada una de los tipos de cebos de origen artificial que se utilizan en la actividad de la pesca.	Sí	1,1
Nom_Fam	Es el nombre que recibe cada tipo de familia	No	1,1

10.6 Laboratorio VI

Guía de laboratorio

10.6.1 Tema: Paso a Modelo Relacional

Inclusión de atributos

Esquema relacional resultado.

10.6.2 Competencia o habilidad

Representar graficamente los atributos en el esquema relacional

Objetivos de Aprendizaje

Representar el esquema relacional con los atributos incluidos usando una herramienta de representación.

Aplicar la primera, segunda y tercer forma normal a las entidades.

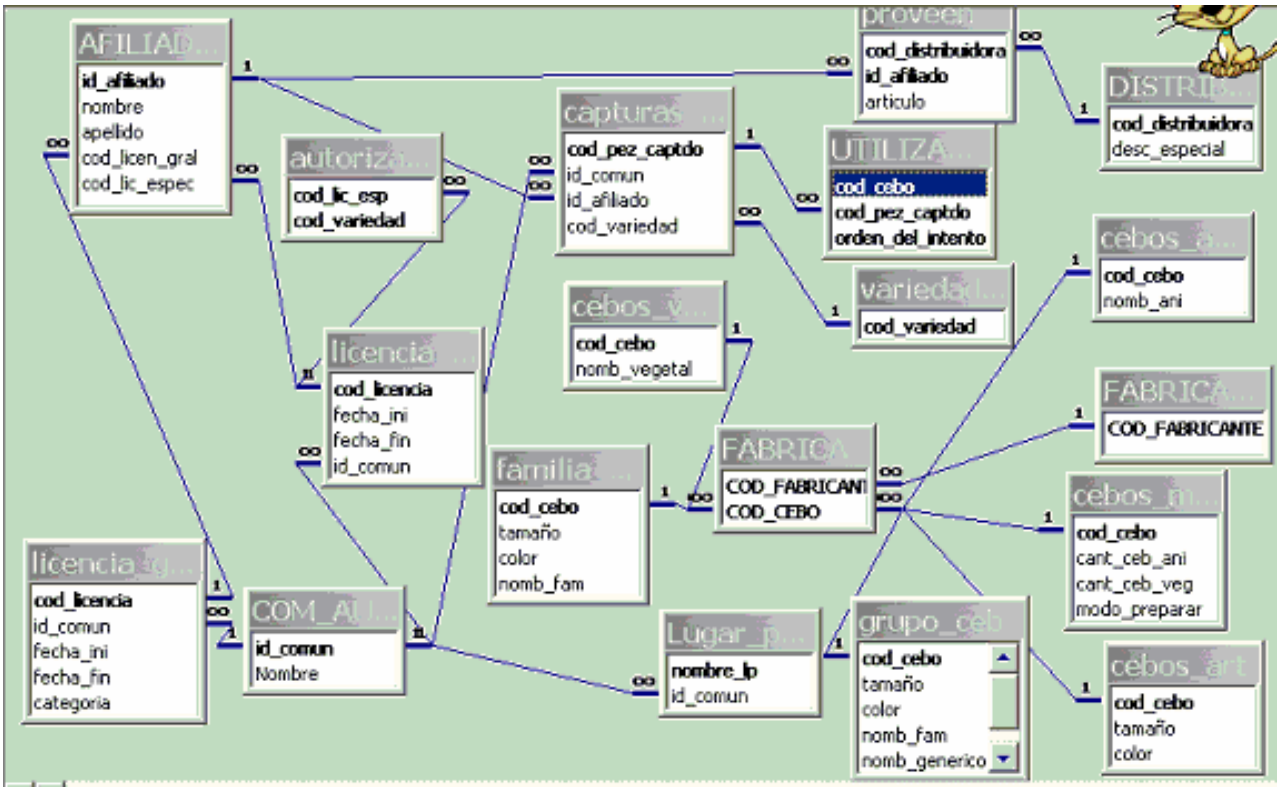
Actividades de Aprendizaje

Iniciar el estudio de este laboratorio aclacrándose sobre las competencia y objetivos de aprendizaje, si tiene dudas, consulte al profesor.

Representar graficamente el esquema relacional con los atributos incluidos

Normalilzar las entidades.

Desarrollo.


AFILIAD...
id_afiliado
nombre
apellido
cod_licen_gral
cod_lic_espec
autoriza...
cod_lic_esp
cod_variedad
capturas ...
cod_pez_captdo
id_comun
id_afiliado
cod_variedad
proveen
cod_distribuidora
id_afiliado
articulo
DISTRIB...
cod_distribuidora
desc_especial
UTILIZA...
cod_cebo
cod_pez_captdo
orden_del_intento
variedad...
cod_variedad
cebos_a...
cod_cebo
nomb_ani
cebos_v...
cod_cebo
nomb_vegetal
licencia ...
cod_licencia
fecha_ini
fecha_fin
id_comun
familia ...
cod_cebo
tamaño
color
nomb_fam
FABRICA
COD_FABRICANT
COD_CEBO
FABRICA...
COD_FABRICANTE
cebos_n...
cod_cebo
cant_ceb_ani
cant_ceb_veg
modo_preparar
licencia_g...
cod_licencia
id_comun
fecha_ini
fecha_fin
categoria
COM_AF...
id_comun
Nombre
Lugar_p...
nombre_lp
id_comun
grupo_ceb
cod_cebo
tamaño
color
nomb_fam
nomb_generico
cebos_art
cod_cebo
tamaño
color

Ejemplos de normalización:

1) DISTRIBUIDORA(Cod_distribuidora)

1FN Cumple con la 1FN

2FN Cumple con la 2FN

2) AFILIADOS(id_Afiliado, nombre, apellido, cod_Licencia_gral,

Cod_Licencia_Espec)

1FN Cumple con la 1FN

2FN Todos los atributos dependen de la llave primaria ID_AFILIADO

3FN Ninguno de los atributos depende de la llave primaria de forma transitiva, esta 3FN

Normalizar las siguientes entidades

PROVEEN, DESCUENTO

Solución de las entidades propuestas

3) PROVEEN(cod_Distribuidora, Id_Afiliado, Artículo, Desc.especial)

1FN Cumple con la 1FN

2FN No cumple porque Desc_especial, depende solo del Cod_distribuidora, o sea de la DISTRIBUIDORA.

3.1) Modificar las Relacion:

PROVEEN(cod_Distribuidora, Id_Afiliado, Artículo)

3.2) Crear la relacion DESCUENTO(cod_distribuidora, Desc_especial)

o bien el atributo Desc_especial, agregárselo a la relacion DISTRIBUIDORA

Ambas estan en 1FN y 2 FN

3FN Ninguno de los atributos depende de la llave primaria de forma transitiva, están 3FN

1 ÍNDICE GENERAL

1 ÍNDICE GENERAL 1
2 ÍNDICE DE FIGURAS 8
3 INTRODUCCIÓN 10
4 OBJETIVOS 11
5 SITUACIÓN DE LA ASIGNATURA EN EL PLAN DE ESTUDIOS DEL INTAE – LEON- INATEC 12
5.1 Relación de Ingeniería del Software I con otras asignaturas 13
5.1.1 Fundamentos de la Programación 13
5.1.2 Programación I 13
5.1.3 Introducción a las bases de datos 14
5.1.4 Análisis y diseño de sistemas y laboratorios 14
5.1.5 Programación II 14
5.1.6 Ingeniería del software I 14
6 METODOLOGÍA DIDÁCTICA Y MATERIAL DIDÁCTICO UTILIZADO 15
6.1 Metodologías a usar 15
6.1.1 Características de Curso Directo 15
6.1.2 Características del Taller Didáctico 15
6.1.3 Características del Laboratorio de Aprendizaje 16
6.1.4 Círculo de Estudios o Corrillos 16
6.2 Técnicas docentes 17
6.2.1 Expositiva 17
6.2.2 Lectura Comentada 17
6.2.3 Demostrativa 18
6.2.4 Discusión en Grupos Pequeños 18
7 PROCESO DE EVALUACIÓN 19
8 TEMPORIZACIÓN 20
8.1 División de los laboratorios 21
9 DESARROLLO DEL TEMARIO 22
10 UNIDAD I: INTRODUCCIÓN A UML 22
10.1 Temas a abordar 22
10.2 CLASE 2: INTRODUCCIÓN A UML 22
10.2.1 ¿Qué es el UML? 22
10.2.2 Visión General de UML 23

10.2.3 Bibliografía: 24
10.3 CLASE 3: INTRODUCCIÓN A UML 25
10.3.1 Notaciones y Metamodelo. 25
10.3.2 Modelado orientado a objetos 25
10.3.3 Bibliografía 26
11 UNIDAD II: FASE DE PLANEACIÓN, ANÁLISIS Y DISEÑO 27
11.1 Temas a abordar: 27
11.2 Fase de análisis: 27
11.3 Fase de diseño: 27
11.4 CLASE 4: ESPECIFICACION DE REQUISITOS 28
11.4.1 Los requerimientos. 28
11.4.2 Funciones del sistema. 29
11.4.3 Categorías de las funciones: 29
11.4.4 Bibliografía 31
11.5 CLASE 5: ESPECIFICACION DE REQUISITOS 32
11.5.1 Atributos del sistema. 32
11.5.2 Atributos del sistema en las especificaciones de funciones: 33
11.5.3 Otros Elementos importantes para atenuar el riesgo 34
11.5.4 Bibliografía 34
11.6 CLASE 6 : CASOS DE USO 35
11.6.1 Introducción a los casos de uso. 35
11.6.2 ¿Qué es un caso de uso? 35
11.6.3 Objetivos de usuario e interacciones con el sistema 36
11.6.4 Ejemplo de un caso expandido de uso: Preparar Producto 37
11.6.5 Actores. 39
11.6.6 Uses y extends 42
11.6.7 Identificación de los casos de uso 44
11.6.8 Casos de uso y procesos del dominio. 45
11.6.9 Caso de uso, funciones del sistema y rastreabilidad 45
11.6.10 Bibliografía: 46
11.7 CLASE 7: DIAGRAMAS DE CASOS DE USO 47
11.7.1 Diagramas de casos de uso. 47
11.7.2 Formatos de los casos de uso 48
11.7.3 Los sistemas y sus fronteras. 48
11.7.4 Casos de uso primario, secundarios y opcionales 50
11.7.5 Casos esenciales de uso comparados con los casos reales de uso. 50
11.7.6 Notación de los casos de uso. 51
11.7.7 Cuando emplear casos de uso 52
11.7.8 Bibliografía: 52
11.8 CLASE 8: DIAGRAMAS DE CASOS DE USO 53
11.8.1 Inicio de un caso expandido de uso: 53
11.8.2 Decisión de notación y ramificación: 53
11.8.3 Sección: principal 54
11.8.4 Sección: Pago de colegiatura en efectivo 55
11.8.5 Casos de uso dentro de un proceso de desarrollo. 55
11.8.6 Bibliografía 60

11.9 CLASE 9: CONSTRUCCION DEL MODELO CONCEPTUAL 61
11.9.1 Modelos conceptuales 61
11.9.2 Conceptos: 62
11.9.3 Los modelos conceptuales y la descomposición: 63
11.9.4 Bibliografía 64
11.10 CLASE 10: CONSTRUCCION DEL MODELO CONCEPTUAL 65
11.10.1 Estrategias para identificar los conceptos. 65
11.10.2 Obtención de conceptos a partir de una lista de categorías de conceptos: 66
11.10.3 Conceptos para el dominio del INTAE 67
11.10.4 Objetos del Informe: 67
11.10.5 Directrices para construir modelos conceptuales. 68
11.10.6 Asignación de nombres: 68
11.10.7 Un error que se comete al identificar los conceptos: 68
11.10.8 Especificación o descripción de conceptos. 69
11.10.9 Necesidad de las especificaciones: 69
11.10.10 Definición de términos en el lenguaje UML. 70
11.10.11 Bibliografía 71
11.11 CLASE 11: AGREGACION DE ASOCIACIONES Y ATRIBUTOS AL MODELO CONCEPTUAL 72
11.11.1 Asociaciones. 72
11.11.2 Notación de las asociaciones en UML. 73
11.11.3 Identificación de las asociaciones: lista de asociaciones comunes 74
11.11.4 Asociaciones de alta prioridad: 75
11.11.5 Directrices de las asociaciones. 75
11.11.6 Papeles. 75
11.11.7 Multiplicidad 75
11.11.8 Asignación de nombre a las asociaciones 76
11.11.9 Asociaciones múltiples entre dos tipos. 76
11.11.10 Asociaciones del dominio del problema. 77
11.11.11 Modelo conceptual del problema. 77
11.11.12 Bibliografía 78
11.12 CLASE 12: AGREGACION DE ASOCIACIONES Y ATRIBUTOS AL MODELO CONCEPTUAL 79
11.12.1 Agregacion de los atributos al modelo conceptual 79
11.12.2 Modelo conceptual 80
11.12.3 Asociaciones múltiples entre dos tipos 81
11.12.4 Notación de los atributos en el UML 83
11.12.5 Tipos de atributos válidos. 83
11.12.6 Deterioro del diseño: ningún atributo debe incluirse como llave foránea: 84
11.12.7 Atributos en el modelo del ejemplo. 85
11.12.8 Explicación de los atributos del problema: 86
11.12.9 Modelo conceptual (con atributos y asociaciones). 86
11.12.10 Bibliografía 86
11.13 CLASE 13: DIAGRAMAS DE SECUENCIA DEL SISTEMA 87
11.13.1 Introducción. 87
11.13.2 Diagramas de secuencia del sistema. 87

11.13.3 Ejemplo de un diagrama de la secuencia de un sistema. 88
11.13.4 Eventos y operaciones de un sistema. 89
11.13.5 Registro de las operaciones de un sistema: 89
11.13.6 Bibliografía 90
11.14 CLASE 14: DIAGRAMAS DE SECUENCIA DEL SISTEMA 91
11.14.1 Cómo elaborar un diagrama de secuencia de un sistema. 91
11.14.2 Eventos y fronteras de un sistema. 91
11.14.3 Presentación del texto del caso de uso 92
11.14.4 Bibliografía 93
11.15 CLASE 15: CONTRATOS 94
11.15.1 Introducción. 94
11.15.2 Contratos. 94
11.15.3 Diseño por Contrato 95
11.15.4 Cuándo utilizar el Diseño por contrato 96
11.15.5 Secciones del contrato. 97
11.15.6 Cómo preparar un contrato. 98
11.15.7 Poscondiciones. 98
11.15.8 Las poscondiciones se relacionan con el modelo conceptual: 99
11.15.9 Descripción de los detalles y algoritmos del diseño: notas. 99
11.15.10 La ventaja de las poscondiciones 99
11.15.11 El espíritu de las poscondiciones: el escenario y el telón. 100
11.15.12 Precondiciones. 100
11.15.13 Recomendación sobre cómo redactar contratos. 100
11.15.14 Bibliografía 101
11.16 CLASE 16: CONTRATOS 102
11.16.1 Contrato para el caso de uso Dar cambio 102
11.16.2 Contrato para el caso de uso efectuar Pago 103
11.16.3 Contrato para el caso de uso Inicio 103
11.16.4 Bibliografía 103
11.17 CLASE 17: DESCRIPCION DE LOS CASOS REALES DE USO 104
11.17.1 Casos reales de uso. 104
11.17.2 Bibliografía COMENTADA: 106
11.18 CLASE 18: DIAGRAMAS DE COLABORACION 107
11.18.1 Introducción. 107
11.18.2 Para realizar un diagrama de interacción, es necesario lo siguiente: 107
11.18.3 Diagramas de interacción. 107
11.18.4 Cómo preparar diagramas de colaboración. 108
11.18.5 Relación de los diagramas de colaboración y otros elementos. 109
11.18.6 Notación básica de los diagramas de colaboración. 109
11.18.7 Representación gráfica de los vínculos: 110
11.18.8 Bibliografía 110
11.19 CLASE 19: DIAGRAMAS DE COLABORACIÓN 111
11.19.1 Representación gráfica de los mensajes. 111
11.19.2 Representación gráfica de los parámetros. 112
11.19.3 Sintaxis de los mensajes: 113
11.19.4 Representación gráfica de los mensajes al “emisor” o a “esto”: 113
11.19.5 Representación gráfica de una iteración: 113

11.19.6 Bibliografía 114
11.20 CLASE 20: DIAGRAMAS DE COLABORACION 115
11.20.1 Representación gráfica de la creación de instancias: 116
11.20.2 Representación gráfica de la secuencia de número de los mensajes: 116
11.20.3 Bibliografía 118
11.21 CLASE 21: DIAGRAMAS DE COLABORACION 119
11.21.1 Representación gráfica de los mensajes condicionales: 119
11.21.2 Representación gráfica de trayectorias condicionales mutuamente excluyentes: 120
11.21.3 Representación gráfica de las colecciones: 121
11.21.4 Representación gráfica de los mensajes dirigidos a multiobjetos: 121
11.21.5 Representación gráfica de los mensajes dirigidos a un objeto clase: 121
11.21.6 Bibliografía 123
11.22 CLASE 22: CLASES Y RELACIONES. 124
11.22.1 Clases. 124
11.22.2 Nombre de una clase 124
11.22.3 Atributos de una clase 124
11.22.4 Perpectivas 126
11.22.5 Asociaciones 127
11.22.6 Bibliografía 131
11.23 CLASE 23: CLASES Y RELACIONES. 132
11.23.1 Operaciones 132
11.23.2 Operaciones de una clase. 133
11.23.3 Organización de atributos y operaciones. 133
11.23.4 Tarjetas de CRC. 133
11.23.5 Cuándo usar las tarjetas de CRC 134
11.23.6 Responsabilidades. 135
11.23.7 Relaciones. 136
11.23.8 Dependencia. 136
11.23.9 Generalización. 137
11.23.10 Bibliografía 138
11.24 CLASE 24: CLASES Y RELACIONES. 139
11.24.1 Asociación. 139
11.24.2 Agregación 140
11.24.3 Bibliografia 141
11.25 CLASES 25, 26 y 27: MECANISMOS COMUNES Y DIAGRAMAS. 142
11.25.1 Mecanismos comunes. 142
11.25.2 Notas. 142
11.25.3 Valores etiquetados 147
11.25.4 Restricciones. 148
11.25.5 Diagramas. 148
11.25.6 Diagramas estructurales. 149
11.25.7 Diagramas de comportamiento. 150
11.25.8 Bibliografía 150
11.26 CLASES 28 y 29: DIAGRAMAS DE CLASES Y DIAGRAMAS DE OBJETOS. 151
11.26.1 Diagramas de clases 151

11.26.2 Propiedades comunes. 151
11.26.3 Contenido..... 151
11.26.4 Diagrama de objetos. 152
11.26.5 Propiedades comunes. 152
11.26.6 Contenido..... 152
11.26.7 Bibliografía..... 153

12 UNIDAD IV: MODELADO ESTRUCTURAL AVANZADO..... 154

12.1 Temas a abordar..... 154
12.2 CLASES 30,31 y 32: CARACTERÍSTICAS AVANZADAS DE LAS CLASES 155
12.2.1 Introducción. 155
12.2.2 Clasificadores. 155
12.2.3 Visibilidad. 157
12.2.4 Elementos abstractos, raíces, hojas y polimórficos. 157
12.2.5 Multiplicidad. 159
12.2.6 Atributos. 159
12.2.7 Operaciones. 160
12.2.8 En su forma completa, la sintaxis de una operación en UML es: 160
12.2.9 Las siguientes son declaraciones legales de operaciones: 160
12.2.10 Bibliografía..... 161
12.3 Clase 33: Características avanzadas de las relaciones..... 162
12.3.1 Introducción. 162
12.3.2 Términos y conceptos. 162
12.3.3 Dependencia. 162
12.3.4 Un estereotipo que aparece en el contexto de las máquinas de estados 164
12.3.5 Generalización. 164
12.3.6 Asociación. 165
12.3.7 Navegación. 165
12.3.8 Visibilidad. 166
12.3.9 Composición. 166
12.3.10 4.4.3.4 Clases asociación. 167
12.3.11 Bibliografía comentada..... 168
12.4 Clase 34: Paquetes..... 169
12.4.1 Introducción. 169
12.4.2 Términos y conceptos. 170
12.4.3 Visibilidad. 172
12.4.4 Modelado de grupos de elementos. 173
12.4.5 Cuando se dibuje un paquete en UML: 174
12.4.6 Bibliografía..... 176
12.5 Clase 35 36 37: Instancias 177
12.5.1 Introducción. 177
12.5.2 Abstracciones e instancias. 178
12.5.3 Modelado de instancias concretas. 179
12.5.4 Bibliografía..... 180

13 UNIDAD V: MODELADO BASICO DEL COMPORTAMIENTO 181

13.1 CLASES 38, 39 y 40: DIAGRAMAS DE ACTIVIDADES. 182
13.1.1 Diagrama de actividades. 182
13.1.2 Estados de acción y estados de actividad. 183
13.1.3 Transiciones. 183
13.1.4 Bifurcación. 183
13.1.5 División y unión. 183
13.1.6 Calles (swimlanes). 184
13.1.7 Flujo de Objetos. 184
13.1.8 Bibliografía 187
13.2 ANEXO 188
13.2.1 Cursos Alternos: 216

2 ÍNDICE DE FIGURAS

Figura 1.1 Perspectivas de desarrollo orientada a objetos y estructurada 26
Figura 2.1 Representación de un caso de uso 37
Figura 2.2 Representación de un actor 40
Figura 2.2 Representación de un actor máquina 40
Figura 2.3 Diagrama de caso de uso 42
Figura 2.3 Diagrama parcial de casos de uso 48
Figura 2.4 Frontera de un caso de uso 49
Figura 2.5 Casos de uso y actores cuando las fronteras son distintas 50
Figura 2.7 Diagrama de casos de uso de Realizar Matricula 57
Figura 2.8 Un modelo conceptual muestra conceptos del mundo real, no muestra los artefactos del software 62
Figura 2.9 El concepto tiene un símbolo, intención y extensión 63
Figura 2.10 Modelo conceptual parcial del dominio del INTAE-LEON. 63
Figura 2.11 Modelo conceptual inicial del dominio del INTAE-LEON. 68
Figura 2.12 Ejemplo en el dominio de un hotel. 69
Figura 2.13 Especificaciones de otras cosas. El signo "*" significa una multiplicidad de "muchos". Indica que una EspecificacióndeProducto puede describir muchos (*) Productos 70
Figura 2.14 Asociaciones 72
Figura 2.15 Multiplicidad de una asociación 76
Figura 2.16 Valores de la multiplicidad 76
Figura 2.17 Asociaciones múltiples 77
Figura 2.18 Nombres de las asociaciones 80
Figura 2.19 Modelo Conceptual aplicado al INTAE – LEON-INATEC 82
Figura 2.20 Concepto y atributos 83
Figura 2.21 Relacione con asociaciones, no con atributos 84
Figura 2.22 Representación incorrecta de atributos 84
Figura 2.23 No utilice atributos como claves extrañas 85
Figura 2.24 Modelo conceptual que muestra los atributos 85
Figura 2.25 Modelo conceptual de la máquina de café 86
Figura 2.26 Diagrama de la secuencia del sistema para el caso de uso Registrar Notas 88
Figura 2.27 Notación de la operaciones en UML 89
Figura 2.28 Operaciones de un sistema registradas en el tipo Sistema 90
Figura 2.29 Definición de la frontera del sistema 92
Figura 2.30 Deducción de los diagramas de secuencia de un sistema de los casos de uso 93
Figura 2.31 Pantalla de agregar alumnos 105
Figura 2.32 Diagrama de colaboración 108
Figura 2.33 Diagrama de secuencia 108
Figura 2.34 Clase e instancias 110

Figura 2.35 Líneas de vínculo (enlace) 110
Figura 2.36 Mensajes..... 111
Figura 2.37 Parámetros..... 112
Figura 2.38 Devolver valores 112
Figura 2.39 Mensaje a “esto” 113
Figura 2.40 Iteración..... 114
Figura 2.41 Mensajes múltiples dentro de la misma cláusula de iteración 115
Figura 2.42 Creación de instancias..... 116

3 INTRODUCCIÓN

En el presente documento se aborda el desarrollo de un Proyecto Docente para la asignatura de INGENIERÍA DE SOFTWARE a impartirse en la carrera de Programador del INTAE – LEON – INATEC. Actualmente está asignatura no se imparte en el centro; la que se imparten en su lugar es Diseño de Sistema que se encuentra dividido en dos asignaturas Diseño de Sistema I. con 80 horas . y Diseño de Sistema II con 40 horas. El motivo de la propuesta de este cambio es que la Programación Orientada a Objetos es una realidad en el mundo y su dominio requiere mayor disposición horaria.

Este proyecto incluye todo el plan docente así como otros aspectos relacionados, como la situación de la asignatura en el plan de estudios de la carrera y su relación con otras asignaturas, en donde se describen los conocimientos previos que hay que tener para un mejor aprendizaje de la misma. Todas las clases están estructuras para ser impartidas en un tiempo de 2 horas cada una.

El plan docente está claramente diferenciado en las dos partes siguientes:

- **PARTE TEÓRICA:** en esta parte se desarrollan todos los conceptos y definiciones con numerosos ejemplos, que permitirán obtener un mejor entendimiento de dichos conceptos.
- **PARTE PRÁCTICA:** van a permitir al estudiante aplicar los conocimientos teóricos adquiridos en clase, mediante el manejo del software adecuado.

También se detalla al estudiante toda la bibliografía utilizada, la que se agregan al final de cada tema con el fin de ayudar en la búsqueda de información cuando se encuentre en dificultades en la comprensión de algún contenido. Por otro lado, se muestra la distribución temporal con la que se debe impartir cada tema, para cumplir con las restricciones de tiempo establecidas y a las cuales el proyecto deberá dar respuesta.

La asignatura pretende abordar de la manera más clara posible las etapas por las cuales debe pasar el desarrollo de un sistema, utilizando la metodología orientada a objetos y las herramientas actuales de desarrollo, y por eso, que se ha elegido el Lenguaje Unificado de Modelado (UML).

4 OBJETIVOS

Los objetivos de esta asignatura son:

- Introducir al estudiante en la Ingeniería del Software orientada a objetos.
- Facilitar los conceptos necesarios para el aprendizaje del Lenguaje Unificado de Modelado.
- Guiar al estudiante en el uso de una herramienta de diseño, para el análisis orientada a objetos.
- Formar profesionales capaces de interpretar modelos orientados a objetos, los cuales son usados actualmente en el diseño de los sistemas.

5 SITUACIÓN DE LA ASIGNATURA EN EL PLAN DE ESTUDIOS DEL INTAE – LEON- INATEC

La asignatura de Ingeniería de Software no se imparte actualmente en la carrera de Técnico Medio en Programación, que ofrece el INTAE– LEON – INATEC. La Ingeniería de Software se recomienda se imparta en el III año del plan actual de estudios de la carrera con una duración de 80 horas.

El período lectivo para esta asignatura contara de 2 horas diarias, de las cuales 1 horas se dedican a la parte teórica y 1 horas a la parte práctica.

Un aspecto muy importante que hay que mencionar es lo relacionado a los conocimientos previos que debe haber adquirido el estudiante en años anteriores para asimilar los conocimientos de la asignatura de Ingeniería de Software. Dichos conocimientos se deben abordaron específicamente en las siguientes asignaturas:

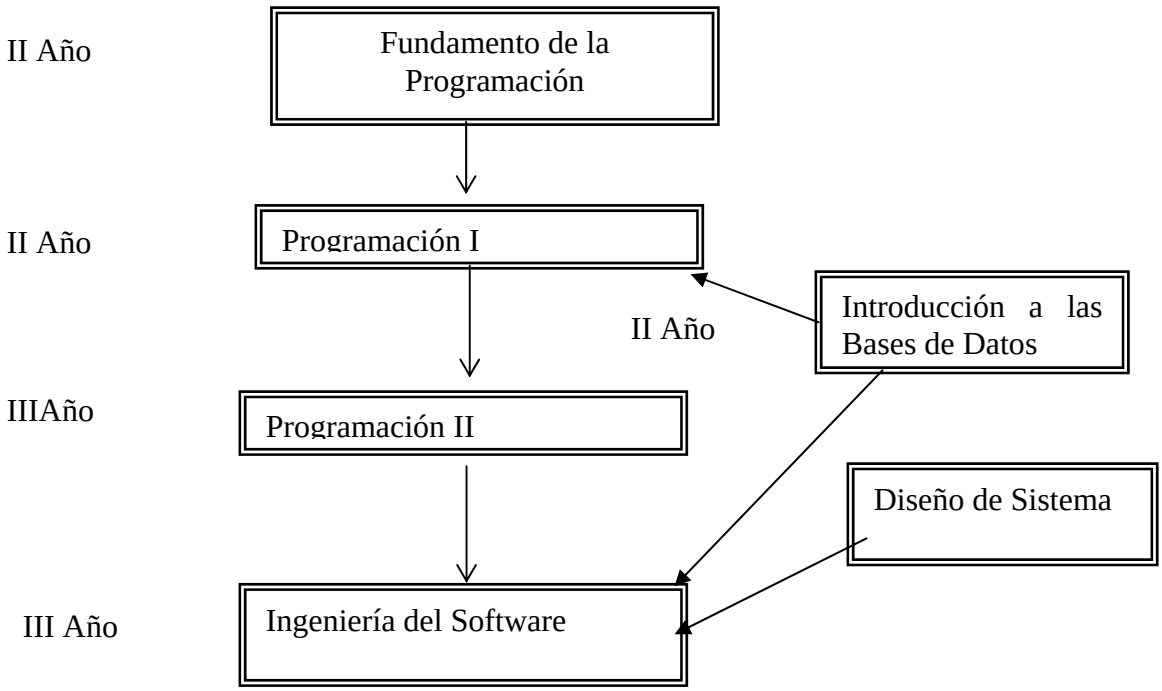
- Programación I y II (introducción a la programación estructuradas y la programación orientada a objeto).
- Base de Datos (introducción a las Bases de Datos).

Estas asignaturas se impartieron en el II año de la carrera, con sus respectivas clases prácticas, ya que forman la base para la ingeniería de Software.

Un último aspecto que hay que considerar es que en el modulo diseño de sistema se estudian los ciclos de vidas de los sistemas y técnicas de investigación entre otros.

5.1 Relación de Ingeniería del Software I con otras asignaturas

En el gráfico se muestra la relación de Ingeniería del Software con otras asignaturas:



A continuación se describirá de forma breve, el contenido de las asignaturas presentadas en el gráfico anterior y cómo se relacionan con las demás.

5.1.1 Fundamentos de la Programación

Las asignaturas fundamentos y sus respectivos laboratorios, tienen una fuerte relación, ya que son los conocimientos básicos de la programación en ella los estudiantes identifican la secuencia lógica de los problemas, usando para ejemplificar el lenguaje de Programación C. Desarrollando habilidades y destrezas en el análisis y planteamiento de un problema.

Se introduce al estudiante en el aprendizaje del Lenguaje C, dándole elementos generales, como los tipos de datos, operadores, funciones de entrada/salida, sentencias de control, tipos estructurados de datos y punteros etc.

5.1.2 Programación I

En esta asignatura, se introduce al alumno en los conceptos de la programación estructura, conceptos que se llevan a la práctica con ayuda del lenguaje de programación

Visual Foxpro. Se abordan temas tales como características de la programación estructurada, entrada y salida estándar, operadores, constantes, variables etc.

El estudiante al haber recibido Programación I, va adquiriendo habilidades en el planteamiento de los problemas usando el análisis estructurado, lo que le facilita la comprensión de temas que se abordan en Ingeniería del software.

5.1.3 Introducción a las bases de datos

Introducción a las bases de datos centra su estudio en el modelo entidad – relación y en el modelo relacional, respectivamente. Se familiariza al alumno con conceptos tales como: entidades, atributos, relaciones, cardinalidad de las asociaciones, tablas, llaves primarias y secundarias, normalización, etc, que de alguna manera, también son aplicados en las etapas de Análisis y Diseño de Sistema.

5.1.4 Análisis y diseño de sistemas y laboratorios

En esta materia se abarcan temas tales como: la evolución del software, modelos de proceso de software, análisis de requisitos, especificación de requisitos software, modelado de datos (atributos, relaciones, cardinalidad), modelado funcional, diccionario de datos, diseño, etc., con un enfoque estructurado.

5.1.5 Programación II

En esta asignatura, se introduce al alumno en los conceptos de la programación orientada a objetos, conceptos que se llevan a la práctica con ayuda del lenguaje de programación Visual Foxpro. Se abordan temas tales como: clase, objetos, controles, propiedades, eventos, sucesos herencias, polimorfismos, Encapsulamiento etc.

5.1.6 Ingeniería del software I

En este módulo se dota al estudiantes de herramientas para el análisis de un problema usando técnicas de modelado orientado a objeto tales como: casos de uso, diagrama de secuencia, modelo conceptual, contratos etc. Para ello es necesario que el alumno domine conceptos tales como clase objetos etc.

6 METODOLOGÍA DIDÁCTICA Y MATERIAL DIDÁCTICO UTILIZADO

6.1 Metodologías a usar

- Curso Directo
- Taller Didácticos
- Laboratorio de Aprendizajes
- Círculos de Estudios o Corrillos

6.1.1 Características de Curso Directo

- Centrada en el docente
- Sin experiencia de los participantes. Toma como base las exposiciones del instructor para proporcionar la información.
- Utiliza cuestionario, corrillos, casos e investigaciones dirigidas para ejercitar la información.
- Usa poco materiales didácticos complejos. El principal material de apoyo es el retroproyector, rotafolio, o pizarrón como auxiliar de la exposición, las laminas de rotafolio, mapas, afiches y otros recursos para facilitar la comunicación.

6.1.2 Características del Taller Didáctico

- Facilita el desarrollo de habilidades para realizar tareas o procesos (Destreza, algunas tareas de orden administrativos)
- Facilita el aprender destreza y adquirir dominio de lo enseñado por medio de la practica.
- Los participantes deben egresar del módulo demostrando saber realizar las tareas o proceso.
- Es una metodología centrada en el instructor en su primera parte y centrada en el alumno en la segunda.
- Aplicable en situaciones grupales o individuales.
- Se adapta a programas elaborados con criterios por competencias.
- El instructor debe atender la dosificación y complejidad de las tareas y operaciones.

- El instructor es ante todo, un supervisor de los avances en el proceso de aprendizaje de sus alumnos.
- Requiere del instructor un dominio completo de los contenidos
- Los recursos didácticos que utilizan son los materiales y equipos reales propios de la tarea.
- Para proporcionar información teórica el instructor puede utilizar materiales impresos diversos.

6.1.3 Características del Laboratorio de Aprendizaje

- Esta metodología está sustentada en las experiencias del participante.
- Aplica principios de practica - teoría – práctica - práctica.
- El alumno es más protagonista, vive experiencia, experimenta consigo mismo.
- El participante descubre aportando experiencias
- Los participantes aprenden viviendo una experiencia y sistematizando los elementos envueltos en la experiencia.
- Util para desarrolla de habilidades congnoscitivas y afectivas, simples y complejas
- Es una metodología efectiva para la modificación de actitudes.
- Los participantes deben volver a practicar después de ser retroalimentados.
- Aprovecha una variedad amplia de ejercicios y técnicas.
- Requiere de instructores calificados para dirigir el análisis de los hechos significativos.
- El docente debe manejar eficientemente las técnicas.
- Facilita, a la larga, aprendizaje colaterales.

6.1.4 Círculo de Estudios o Corrillos

- Requiere de apoyos didácticos como documentación de trabajo, guías de estudio, cuadernos de prácticas y ejercicios. Estos materiales deben ser de preferencia personalizados.
- Propios para eventos no escolarizados y favorables para personas que no disponen de tiempo completo de estudio.
- Se requiere de apoyo de un asesor pendiente del grupo.
- Deben tener autoevaluaciones los ejercicios previstos en los materiales.
- El asesor no necesariamente asiste a todas las reuniones, pero está atento de los avances del circulo.

- Los materiales deben contener toda la información y todas las situaciones de ejercitación que los objetivos requieren.
- El asesor supervisa y complementa, evita dar clases.
- Puede ser aplicado en programas duales o en programas basados en competencias.
- Exige persistencia, fuerza de voluntad y cooperación de los alumnos.
- Usado frecuentemente desarrolla habilidades para la comunicación y el análisis de documentación y las actitudes de respeto y tolerancia a las ideas ajenas.
- Puede aprovechar el tiempo libre de los participantes.
- Util para aprendizaje cognoscitivos simples y medianamente complejos
- Permite aprendizaje de proceso e información técnica básica

6.2 Técnicas docentes

- Expositivas
- Lectura Comentadas
- Demostrativas

6.2.1 Expositiva

- Craig y Bittel dicen que la exposición es la "la forma en que un instructor presenta una información a un grupo de participantes, quienes adoptan una actitud receptiva"
- Alfaro Castro la define como el procedimiento didáctico mediante el cual el instructor, valiéndose principalmente de un lenguaje adecuado presenta a los alumnos un nuevo tema para que definiéndolo y analizándolo, adquieran la compresión inicial necesaria para su aprendizaje.
- Las definiciones hacen evidente la presentación oral de un tema que, después de ser recibido y analizado por los alumnos, les permitirá asimilarlo y aplicarlo.

6.2.2 Lectura Comentada

- Esta técnica se llama Lectura Comentada porque el maestro suscita y mantiene conversación alrededor de un tema previamente leído, con el propósito de hallar las ideas fundamentales de los párrafos (análisis) y llegar finalmente a la redacción del resumen del texto (síntesis).
- Esta técnica se utiliza mucho porque además de proporcionar la información requerida para lograr una competencia o un objetivo de aprendizaje, desarrolla la ejercitación de la expresión oral, contribuyen a que el

muchacho(a) utilice el vocabulario adecuado, practique la lectura y la correcta pronunciación.

6.2.3 Demostrativa

Es la ejecución de una tarea de parte de un docente frente a sus alumnos(as), con el propósito de que éstos la practiquen bajo su supervisión hasta llegar a dominarla.

Para dar cumplimiento al desarrollo de una demostración el (la) docente da inicio a la ejecución de la demostración de la tarea, bajo la observación de sus alumnos, explicando y aclarando a su vez, los puntos clave y medidas de seguridad a guardar durante su realización. Seguidamente los alumnos repiten la tarea tal y como fue enseñada, bajo la supervisión del (la) docente, quien corrige errores y retroalimenta de forma oportuna para alcanzar el dominio continúa ejercitando las veces que considere hasta dominarla por completo.

6.2.4 Discusión en Grupos Pequeños

Esta técnica consiste en la conversación y el debate de un grupo acerca de un tópico de interés común. En la discusión intervienen todos los miembros del grupo en una comunicación cara a cara.

Dependiendo del tema a desarrollar por medio de clases teóricas o practicas que en conjunto son de dos horas de duración y tratando de cumplir fielmente con la distribución temporal establecida. También se propondrá la realización de trabajos en grupo, según se vaya avanzando en el contenido para ayudar al estudiante a profundizar sobre el tema y principalmente, para despertar en él su capacidad de investigación y trabajo en grupo.

En lo relacionado al material didáctico, básicamente se hará uso de pizarra para la explicación de las clases teóricas, manuales, laboratorios, referencias bibliografía etc. El estudiante podrá tener acceso a la bibliografía existente en la biblioteca.

7 PROCESO DE EVALUACIÓN

La asignatura de Ingeniería de software estará compuesta por clases teóricas y práctica, que corresponde al laboratorio. A lo largo del curso se realizan evaluaciones parciales por cada unidad. Las notas obtenidas en las evaluaciones parciales serán independientes cada una. La nota final de la asignatura, se obtiene promediando los resultados obtenidos en cada unidad.

En cuanto a la obtención de cada una de las notas parciales, el desglose es el siguiente:

- Realización de un examen teórico: 60 puntos.
- Evaluación de laboratorio: 40 puntos.

La nota del módulo es el promedio de los tres parciales

Como se explicó anteriormente, la realización de las prácticas tiene un valor de 40 puntos en cada nota parcial y se harán un total de 9 prácticas que se repartirán entre los tres parciales. Por cada práctica que se realice, el estudiante deberá entregar un reporte con lo siguiente:

- Enunciado del ejercicio.
- Explicación de la solución dada al ejercicio.
- Diagramas que se pidan en el enunciado del ejercicio.

Además del reporte, el estudiante deberá entregar un disquete conteniendo el archivo .mdl con la solución del problema.

La mayor parte de la evaluación consiste en la realización de un examen teórico, el cual estará compuesto de los siguientes tipos de preguntas:

- Opción Múltiple.
- Respuestas Alterna
- Correspondencia
- Ordenamiento
- Respuestas Breves

8 TEMPORIZACIÓN

La materia Ingeniería del Software se impartirá como uno de los módulos finales del tercer año de la carrera Técnico Medio en Programación, ofrecida por el INTAE – León - INATEC. Constará de 10 horas semanales, que se dividen en 5 horas de teoría y 5 de laboratorio. El número total de horas para impartir dicha asignatura es de 80 horas por modulo. A continuación se presenta el plan analítico de dicha asignatura(parte teórica):

UNIDAD	SEMANA	No. CLASE	CONTENIDO
0	1	-1	Presentación de la asignatura.
I.		2 3	Unidad I: Introducción a UML. ¿Qué es UML? Visión general de UML. Notaciones y metamodelo. Modelado Orientado a Objetos.
II		4-5	Unidad II: Fase de planeación, análisis y diseño. Especificación de requisitos.
	2	6	Casos de uso.
		7-8	Diagramas de casos de uso.
		9-10	Modelo conceptual. Construcción del modelo conceptual
	3	11-12	Agregación de las asociaciones. Agregación de los atributos.
		13-14	Diagrama de secuencia.
	4	15-16	Contratos.
		17-18 19-20	Descripción de los casos reales de uso. Diagramas de colaboración.
	5	21	Diagramas de colaboración(continuación).
III		22-23-24	Unidad III: Modelado estructural básico. Clases. Relaciones.
	6	25-26-27	Mecanismos comunes. Diagramas.
		28-29	Diagramas de clases. Diagramas de objetos.
IV	7	30-31-32	Unidad IV: Modelo estructural avanzado. Características avanzadas de las clases.
		33	Características avanzadas de las relaciones.
		34	Paquetes.
	8	35-36.37	Instancias.
V		38-39-40	Unidad V: Modelado básico del comportamiento. Diagramas de actividades.

Para los laboratorios, tomamos en cuenta únicamente 7 semanas, ya que en la primera semana, se introduce a los estudiantes en la parte teórica, para luego aplicar dichos conocimientos en las prácticas.

8.1 División de los laboratorios

Semana	No. Práctica	Contenido
2	0	Presentación del laboratorio.
3	1	Especificación de requisitos.
4	2	Diagrama de clase
4-5	3	Diagramas de secuencia y contratos.
6	4	Casos de uso, casos de uso múltiples y expandidos
7	5	Casos reales de uso
8	6	Diagrama de colaboración, clase y paquetes

9 DESARROLLO DEL TEMARIO

En los siguientes apartados se desarrolla el contenido de cada unidad didáctica, profundizando posteriormente con el contenido de temático correspondiente a cada una de las clases que componen la unidad.

10 UNIDAD I: INTRODUCCIÓN A UML

10.1 Temas a abordar

- ¿Qué es UML?.
- Visión general de UML.
- Notaciones y metamodelo.
- Modelado orientado a objetos.

Esta unidad se impartirá en una sesión.

10.2 CLASE 2: INTRODUCCIÓN A UML

¿Qué es UML?.

Visión general de UML.

OBJETIVOS:

Definir el concepto de UML.
Relatar una breve reseña histórica de las metodologías utilizadas para el análisis y diseño de software.
Clasificar la visión general de UML

10.2.1 ¿Qué es el UML?

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los elementos de un sistema con gran cantidad de software. UML proporciona una forma estándar de escribir los

planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases descritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables.

El lenguaje unificado de modelado o UML (Unified Modeling Language) es el sucesor de la oleada de métodos de análisis y diseño orientados a objetos (OOA&D) que surgió a finales de la década de 1980 y principios de la siguiente. El UML unifica, sobre todo, los métodos de Booch, Rumbaugh (OMT) y Jacobson, pero su alcance llagará a ser mucho más amplio. En estos momentos el UML está en pleno proceso de estandarización con el OMG (Object Management Group o Grupo de Administración de Objetos) y se convertirá en el lenguaje de modelado estándar del futuro.

Decimos, pues, que el UML es un lenguaje de modelado, y no un método. La mayor parte de los métodos consisten, al menos en principio, en un lenguaje y en un proceso para modelar. El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar los diseños. El proceso es la orientación que nos dan sobre los pasos a seguir para hacer el diseño.

10.2.2 Visión General de UML

UML es un lenguaje para:

- Visualizar.
- Especificar.
- Construir.
- Documentar los elementos de un sistema con gran cantidad de software.

UML es un lenguaje: un lenguaje que proporciona un vocabulario y las reglas para combinar palabras de ese vocabulario con el objetivo de posibilitar la comunicación. Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema. Un lenguaje de modelado como UML es por tanto un lenguaje estándar para los planos del software. Para sistemas con gran cantidad de software, se requiere un lenguaje que cubra las diferentes vistas de la arquitectura de un sistema mientras evoluciona a través del ciclo de vida del desarrollo del software. El vocabulario y las reglas de un lenguaje como UML indican cómo crear y leer modelos bien formados.

UML es un lenguaje para visualizar: Algunas cosas se modelan mejor textualmente; otras se modelan mejor de forma gráfica y en todos los sistemas interesantes hay estructuras que trascienden lo que puede ser representado gráficamente en un lenguaje de programación. UML es uno de estos lenguajes gráficos, es decir, detrás de cada símbolo en la notación UML hay una semántica bien definida. De esta manera, un desarrollador puede escribir un modelo en UML, y otro desarrollador, o incluso otra herramienta, puede interpretar ese modelo sin ambigüedad.

UML es un lenguaje para especificar: Especificar significa construir modelos precisos, no ambiguos y completos. UML cubre la especificación de todas las decisiones de

análisis, diseño e implementación que deben realizarse al desarrollar un sistema con gran cantidad de software.

UML es un lenguaje para construir: UML no es un lenguaje de programación visual, pero sus modelos pueden conectarse de forma directa a una gran variedad de lenguajes de programación. Esto significa que es posible establecer correspondencias desde un modelo UML a un lenguaje de programación como JAVA, C++ o Visual Basic, o incluso a tablas en una base de datos relacional. Las cosas que se expresan mejor gráficamente también se representan gráficamente en UML, mientras que las cosas que se expresan mejor textualmente se plasman con el lenguaje de programación.

UML es un lenguaje para documentar: UML cubre la documentación de la arquitectura de un sistema y todos sus detalles. UML proporciona un lenguaje para expresar requisitos y pruebas; y también proporciona un lenguaje para modelar las actividades de planificación de proyectos y gestión de inversiones.

10.2.3 Bibliografía:

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 2: Presentación de UML. Muestra una introducción a las principales características del lenguaje unificado de modelado.

10.3 CLASE 3: INTRODUCCIÓN A UML

Notaciones y Metamodelo.
Modelado Orientado a Objetos.

OBJETIVOS:

Definir el significado de Notación en el modelo UML
Definir el concepto de Metamodelos .
Distinguir las formas más comunes de enfocar un modelo usando: perspectiva orientada a objetos y la perspectiva algorítmica

10.3.1 Notaciones y Metamodelo.

El UML define una notación y un metamodelo. La *notación* es el material gráfico que se ve en los modelos; es la sintaxis del lenguaje de modelado. Por ejemplo, la denominación de un diagrama de clases define cómo se representan conceptos y temas como clase, asociación y multiplicidad.

Un *modelo* es una simplificación de la realidad; un modelo proporciona los planos de un sistema. Los modelos se construyen para comprender mejor el sistema que estamos desarrollando; y se construyen modelos de sistemas complejos porque no podemos comprender el sistema en su totalidad.

Un *metamodelo* se define como un diagrama, usualmente un diagrama de clases, que defina la notación. Un metamodelo ayuda a definir qué es un modelo bien formado, es decir, uno que sintácticamente está correcto. Por ejemplo, se podría tener un metamodelo que muestre la relación entre asociaciones y generalizaciones.

10.3.2 Modelado orientado a objetos.

En el desarrollo de software hay varias formas de enfocar un modelo. Las dos formas más comunes son la perspectiva orientada a objetos y la perspectiva algorítmica.

La visión tradicional del desarrollo de software toma una perspectiva algorítmica. En este enfoque, el bloque principal de construcción de todo el software es el *procedimiento o función.* Esta visión conduce a los desarrolladores a centrarse en cuestiones de control y de descomposición de algoritmos grandes en otros más pequeños. Los sistemas construidos con un enfoque algorítmico se vuelven muy difíciles de mantener ya que dichos sistemas son frágiles cuando los requisitos cambian o el sistema crece.

La visión actual del desarrollo de software toma una perspectiva orientada a objetos. En este enfoque, el principal bloque de construcción de todos los sistemas software es el *objeto o clase*. Un objeto es una cosa extraída del vocabulario del espacio del problema o del espacio de la solución; una clase es una descripción de un conjunto de objetos similares. Actualmente, el enfoque orientado a objetos forma parte de la tendencia principal para el desarrollo de software, simplemente porque ha demostrado ser válido en la construcción de sistemas de toda clase de dominios de problemas.

Además, la mayoría de los lenguajes actuales, sistemas operativos y herramientas son orientadas a objetos, lo que ofrece más motivos para ver el mundo en términos de objetos. Dicha perspectiva permite el desarrollo de sistemas más consistentes y menos frágiles a los cambios que estos puedan tener.

En la siguiente figura se puede ver una comparación de ambas perspectivas de desarrollo para un sistema de información de una biblioteca, en la perspectiva orientada a objetos se muestran las clases y en la perspectiva algorítmica se muestran las funciones que realiza el sistema:

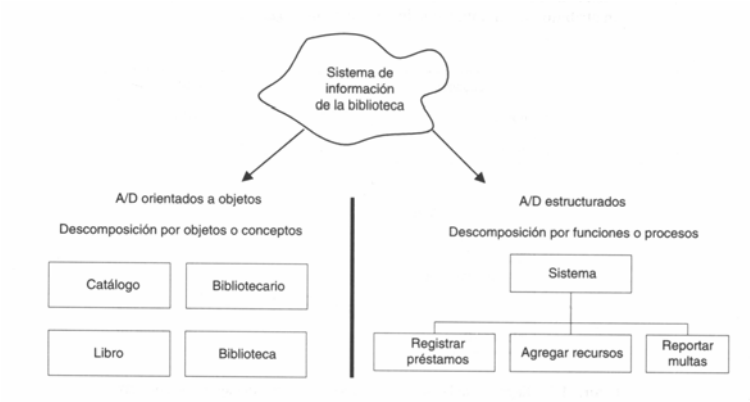


Figura 1.1 Perspectivas de desarrollo orientada a objetos y estructurada

10.3.3 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 2: Presentación de UML. Muestra una introducción a las principales características del lenguaje unificado de modelado.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 1: El génesis de UML. Páginas 9-11. Presenta los conceptos de modelo y metamodelo.

UML Distilled “Applyng the standard Object Modeling Language”. Martin Fowler. Editorial Addison Wesley. Capítulo 1: Introduction. Páginas 1-11. Expone de manera breve una introducción al uso de UML y los conceptos de análisis y diseño orientado a objetos.

11 UNIDAD II: FASE DE PLANEACIÓN, ANÁLISIS Y DISEÑO

11.1 Temas a abordar:

- Fase de planeación:
- Especificación de requisitos.
- Casos de uso.
- Diagramas de casos de uso.

11.2 Fase de análisis:

- Modelo conceptual.
- Construcción del modelo conceptual.
- Agregación de las asociaciones.
- Agregación de los atributos.
- Diagramas de secuencia.
- Contratos.

11.3 Fase de diseño:

- Descripción de los casos reales de uso.
- Diagramas de colaboración.

Esta unidad se impartirá en 38 horas .

11.4 CLASE 4: ESPECIFICACION DE REQUISITOS

Los requerimientos de Sistema.
OBJETIVOS:

Identificar las funciones del Sistema.
Clasificar las funciones del Sistema
Identificar los atributos del Sistema.
Clasificar los atributos del sistema y relacionarlos con las funciones

11.4.1 Los requerimientos.

Un proyecto no puede ser exitoso sin una especificación correcta y exhaustiva de los requerimientos. Los ***requerimientos*** *son una descripción de las necesidades de un producto.* La meta primaria de la fase de requerimiento es identificar y documentar lo que en realidad se necesita, en una forma que claramente se lo comunique al cliente y a los miembros del equipo de desarrollo. El reto que consiste en definirlos de manera inequívoca, de modo que se detecten los riesgos y no se presenten sorpresas al momento de entregar el producto.

Se recomiendan tomar en cuenta las siguientes fase de requerimientos:

- Panorama General
- Cliente
- Metas
- Funciones del Sistema
- Atributos del sistema

Para ejemplificar cada una de las partes que conforman la fase de requerimientos y a lo largo del documento, se agregaran ejemplos donde el tema de estudio se explique con mayor claridad iniciaremos analizando el siguiente ejemplo:

Sistema de Matriculas y Control de Notas
Supongamos que se nos ha pedido crear un programa para las matriculas y control de notas del INTAE – LEON - INATEC. Con una estrategia de incremento interactivo, vamos a realizar las fases de requerimientos, análisis y diseño orientado a objeto.

Presentación general.

Este proyecto tiene por objeto crear un sistema de "*Control de matriculas y Notas*" en el INTAE- LEON - INATEC.

Clientes.

INTAE- LEON - INATEC.

Metas.

En términos generales, la meta es agilizar el proceso de Matriculas y el Control de Notas, dar soporte a los servicios que da el centro agilizando la entrega de boletines, constancias, etc, Mas concretamente, la meta incluye:

- Matrículas ágiles de los alumnos
- Análisis rápido y exacto de su población estudiantil en periodos de matricula y en periodo lectivo
- Control automático de rendimientos finales y parciales de los alumnos

11.4.2 Funciones del sistema.

Las funciones del sistema *son lo que éste deberá hacer*, por ejemplo denegar la matricula de un alumno si la cantidad de matriculados sobrepasa la capacidad que el centro posee. Hay que identificarlas y listarlas en grupos cohesivos y lógicos.

Con el objeto de verificar que algún X es de verdad una función del sistema, la siguiente oración deberá tener sentido:

El sistema deberá hacer <X>.

Por ejemplo: El sistema deberá denegar la matricula de un alumno si la cantidad de matriculados sobrepasa la capacidad que el centro posee para brindar sus servicios..

Los **atributos del sistema** son cualidades no funcionales, entre ellas la facilidad de uso, que normalmente se confunde con las funciones. Nótese que "facilidad de uso" no encaja en la oración de verificación: El sistema deberá hacer la facilidad de uso. Los atributos no deben formar parte del documento de las especificaciones funcionales del sistema, sino que de un documento independiente que especifica sus atributos.

11.4.3 Categorías de las funciones:

Las funciones, deben clasificarse a fin de establecer prioridades entre ellas e identificar las que de lo contrario pasarían inadvertidas(pero que consumen tiempo y otros recursos). Las categorías son:

Categoría de la función	Significado
Evidente	Debe realizarse, y el usuario debería saber que se ha realizado.
Oculta	Debe realizarse, aunque no es visible para los usuarios. Esto se aplica a procesos como guardar información en un mecanismo persistente de almacenamiento. Las funciones ocultas a menudo se omiten erróneamente durante el proceso de obtención de los requerimientos.
Superflua	Opcionales, su inclusión no repercute significativamente en el costo ni en otras funciones.

Funciones básicas en el ejemplo anterior:

Ref. #	Función	Categoría
R1	Registra la matricula en proceso (actual): la especialidad año y turno	Evidente
R2	Reduce la capacidad del centro cuando se realiza una matricula	Oculta
R3	Formar los grupos para cada especialidad	Oculta
R4	Anula matricula si las matriculas actuales sobrepasa la capacidad del centro.	
R5	Solicita contraseña al responsable de registro	Evidente
R6	Ofrece mecanismo de comunicación entre los procesos y entre los sistemas	
R7	Ofrece un mecanismo de almacenamiento persistente	Oculta
R8	Mostrar los datos generales del alumno matriculado	Evidente
R9	Registra las notas de los alumnos	Evidente
R10	Entrega certificados, boletines y diplomas	Evidente
R11	Registra notas de reparación	Evidente
R12	Muestra lista de alumnos que deben ser retirados del centro por no cumplir con el rendimiento mínimo	Oculta
R13	Captura cancelación de pago de colegiatura	

Función de pago

Ref. #	Función	Categoría
R 13.1	Maneja los pagos de matricula, capturando la cantidad ofrecida y calculando el saldo deudor	Evidente
R13.2	Maneja los pagos por cheques, capturando el número de cedula mediante captura manual y autorizando el pago con el servicio de autorización .	Evidente
R 13.3	Registra los pagos en el sistema de cuentas por cobrar, pues el servicio de autorización de crédito debe a al centro el monto del pago.	Evidente
R13.4	Registra cancelación o abono de pago de colegiatura	

11.4.4 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 5: Conocimiento de los requerimientos. Páginas 39-46. Explica las partes que debe contener un documento de especificación de requisitos.

11.5 CLASE 5: ESPECIFICACION DE REQUISITOS

Requerimientos.

OBJETIVOS:

Identificar y clasificar las funciones del sistema.

Identificar y clasificar los atributos del sistema y relacionarlos con las funciones.

11.5.1 Atributos del sistema.

Los atributos del sistema son sus características o dimensiones, no son funciones. Por ejemplo:

- Facilidad de uso
- Tolerancia a fallos
- Tiempo de respuesta
- Interfaz de usuario
- Ejecución en múltiples plataformas

Los atributos del sistema pueden abarcar todas las funciones(por ejemplo, la plataforma del sistema operativo) o ser específicos de una función o de un grupo de funciones.

Los atributos tienen un conjunto de **detalles de atributos**, los cuales tienden a ser valores discretos, confusos o simbólicos, por ejemplo:

Tiempo de respuesta = (psicológicamente correcto)

Interfaz de usuario = (gráfico, colorido, basado en formas)

Algunos atributos del sistema también pueden tener **restricciones de frontera del atributo**, que son condiciones obligatorias de frontera, generalmente en un rango numérico de los valores de un atributo; por ejemplo:

Tiempo de respuesta = (cinco segundos como máximo)

Ejemplos:

Atributo	Detalles y restricciones de frontera
Tiempo de respuesta	*(restricción de frontera)* Cuando un usuario solicite un determinado producto, el tiempo máximo de respuesta para entregar el producto debe ser de 1 minuto.
Interfaz de usuario	*(detalle)* Botones de pulsación digital para controlar cada una de las operaciones de la máquina.
Facilidad de uso	*(detalle)* Fácil manejo mediante una pantalla que indique en todo momento el estado de la máquina.
Tolerancia a fallas	*(restricción de frontera)* Debe cancelar una operación en caso de que se produzca falta de energía, entregando al usuario el dinero en caso de que la entrega del producto no se haya completado.

11.5.2 Atributos del sistema en las especificaciones de funciones:

Es conveniente describir todos los atributos del sistema que se relacionen claramente con las funciones, dentro de la lista de especificación de funciones. Los detalles de los atributos y las restricciones de frontera pueden catalogarse como **obligatorios** u **opcionales.** Una restricción de frontera suele ser obligatoria, de lo contrario significaría que no era sólida.

Ref #	Función	Categoría	Atributo	Detalles y restricciones	Categoría
R4	Anula matricula si la nueva matricula sobre pasa la capacidad del centro.	Evidente	Tiempo de respuesta.	3 segundos como máximo.	Obligatorio
R1	Registra la matricula en proceso (actual): la especialidad año y turno	Evidente	Interfaz de Usuario	Pantalla basada en formas colorido	Obligatorio Opcional
R9	Registra las notas de los alumnos	Evidente	Tolerancia a fallas	Debe registrar en alumnos pendientes aquellos que no tienen nota	Obligatorio

11.5.3 Otros Elementos importantes para atenuar el riesgo

Para tener un mejor entendimiento del problema, además de las funciones y de los atributos del sistema, son necesarios otros elementos:

- **Requerimientos y equipos de enlace:** lista de las personas que deberían participar en la especificación de las funciones y atributos del sistema, en la realización de entrevistas, pruebas y otras actividades.
- **Grupos afectados:** los que reciben el impacto del desarrollo o aplicación del sistema.
- **Suposiciones:** las cosas cuya verdad se supone.
- **Riesgos:** cosas que pueden provocar el fracaso o retraso.
- **Dependencias:** otras personas, sistemas y productos de los cuales no puede prescindir el proyecto para su terminación.
- **Glosario:** definición de términos utilizados.
- **Casos de uso:** descripciones narrativas de los procesos del dominio.
- **Modelo conceptual preliminar:** modelo de conceptos importantes y de sus relaciones.

11.5.4 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 5: Conocimiento de los requerimientos. Páginas 39-46. Explica las partes que debe contener un documento de especificación de requisitos.

11.6 CLASE 6 : CASOS DE USO

Casos de uso

Actores

Identificación de los casos de uso

Casos de uso y procesos del dominio

OBJETIVOS:

Identificar y representar gráficamente los casos de uso.

11.6.1 Introducción a los casos de uso.

Para mejorar la comprensión de los requerimientos, es necesario la creación de los casos de uso, es decir, **descripciones narrativas de los procesos del dominio**.

Los casos de uso son un fenómeno interesante. Durante mucho tiempo, tanto en el desarrollo orientado a objeto como en el tradicional, las personas se auxiliaban de escenarios típicos que les ayudaban a comprender los requerimientos. Estos escenarios, sin embargo, se trataban de modo muy informal; siempre se construían, pero pocas veces se documentaban. Ivar Jacobson es ampliamente conocido por haber cambiado esto con su método Objectory y su primer libro sobre el tema.

Jacaobson elevó la visibilidad del caso de uso (su nombre para un escenario) a tal punto que lo convirtió en un elemento primario de la planificación y el desarrollo de proyectos. Desde la publicación de su libro en (1994), la comunidad de los objetos ha adoptado los casos de uso en grado notable.

11.6.2 ¿Qué es un caso de uso?

Un caso de uso es, en esencia, una interacción típica entre un usuario y un sistema de cómputo. Considérese el procesador de palabras con el que se escribieron estás líneas que usted lee. Dos casos de uso típico serían por una parte el “texto en negritas” y “crear un índice”. Por medio de estos ejemplos, se puede uno dar una idea de ciertas propiedades de los casos de uso.

- El caso de uso capta alguna función visible para el usuario
- El caso de uso puede ser pequeño o grande
- El caso de uso logra un objetivo discreto para el usuario.

En su forma más simple, el caso de uso se obtiene hablando con los usuarios habituales y analizando con ellos las distintas cosas que deseen hacer con el sistema. Se

debe abordar cada cosa discreta que quieran, darle un nombre y escribir un texto descriptivo breve (no más de unos cuantos párrafos).

Los casos de uso requieren tener un conocimiento parcial de los requerimientos del sistema, expresados en el documento donde se especifican.

Otro concepto el **caso de uso** es un documento narrativo que describe la secuencia de eventos de un actor(agente externo) que utiliza un sistema para completar un proceso. Los casos de uso son historias o casos de utilización de un sistema, no son exactamente los requerimientos ni las especificaciones funcionales, sino que ejemplifican e incluyen tácitamente los requerimientos en las historias que narran.

11.6.3 Objetivos de usuario e interacciones con el sistema

La diferencia entre lo que llaman objetivos del usuario e interacciones con el sistema. Se puede observar en el ejemplo siguiente, considere la funcionalidad del hoja de estilo con que cuentan la mayor parte de los procesadores de texto.

La interacciones con el sistema permite decir que los casos de uso incluirán cosas como: “define estilo”, “cambia estilo” y “mueve un estilo de un documento a otro”. Sin embargo, estos casos de uso reflejan más bien cosas que el usuario hace con el sistema, en lugar de los verdaderos objetivos que el usuario trata de conseguir. Los verdaderos objetivos del usuario se describirán con términos como “garantizar el formato consistente de un documento” y “hacer que el formato de un documento sea igual que el de otro”.

Esta dicotomía entre objetivo del usuario e interacción con el sistema no se presenta en todas las situaciones. Por ejemplo, el proceso de indización de un documento es muy parecido si se le considera como objetivo del usuario o como interacción con el sistema. No obstante, cuando los objetivos del usuario y las interacciones del sistema difieren, es muy importante tener clara la diferencia.

Ambos estilos de caso de uso tiene sus aplicaciones. Los casos de uso de interacción con el sistema sirven más para fines de planificación; conviene no perder de vista los objetivos del usuario, con el fin de poder considerar formas alternas para el cumplimiento de tales objetivos. Si se llega demasiado pronto a la interacción con el sistema, recurriendo a la primera alternativa obvia, se pasarán por alto otras maneras creativas de cumplir con mayor eficacia los objetivos del usuario. En todos los casos, es una buena idea preguntar **“¿Por qué hicimos esto?”** esta pregunta generalmente conduce a una mejor compresión del objetivo del usuario.

Se recomienda en los trabajos primero centrarse en los objetivos del usuario y después preocuparse en encontrar casos de uso que los cumplan. Hacia el final del periodo de elaboración, esperar tener por lo menos un conjunto de casos de uso de interacción con el sistema por cada objetivo de usuario que he identificado (como mínimo, para las metas del usuario que pretenda manejar en la primera entrega).

El icono utilizado en UML para un caso de uso, es un ovalo.

Figura 2.1 Representación de un caso de uso

Como ejemplo de un caso de uso de alto nivel, tenemos el siguiente: ***Solicitar Matricula.***

Caso de uso: Solicitar Matricula

Actores: Alumno

Tipo: Primario

Descripción: Un Alumno se presenta al centro con la intención de matricularse. La responsable de Registro registra la matricula.

Los encabezados y la estructura de este caso de uso son representativos. Sin embargo el UML no especifica un formato rígido. Cada quien puede definirse sus propios formatos, siempre y cuando utilice los mismos durante todo el proyecto y todos los miembros del equipo de desarrollo sepan interpretarlos.

11.6.4 Ejemplo de un caso expandido de uso: Preparar Producto

Un **caso expandido de uso** muestra más detalles que uno de alto nivel; suelen ser útiles para alcanzar un conocimiento más profundo de los procesos y de los requerimientos. A menudo se lleva acabo un estilo coloquial entre los actores y el sistema. Ejemplo:

Caso de Uso: Control de Acceso

Actores: Resp.. de Registro

Propósito: Captura la contraseña del Resp. de Registro para dar acceso al sistema

Resumen: El Resp. De registro llega al sistema para matricular, asignar notas, o revisar el rendimiento a los diferentes alumnos.

Tipo: Primario y esencial.

Referencias Cruzadas: *Funciones:* R1, R2, R3, R10, R11, R12, R13, R14

Curso normal de los eventos

Acción del actor	Respuesta del sistema
1- Este caso de uso comienza cuando el Resp. De Registro llega al sistema, con la intención de ingresar,	2- Muestra una interfaz solicitando su contraseña
3- El Resp. de Registro escribe su contraseña	4- Da acceso a la configuración del sistema
5- El Resp. De Registro asigna realiza las actualizaciones o modificaciones	6- Realiza las actualizaciones o modificaciones
7- Al terminar de realizar las actualizaciones o modificaciones indica al sistema que la captura finalizó	8- Muestra las actualizaciones o modificaciones realizadas
9- Solicita salir del Sistema	10 – Cierra el sistema

Cursos alternos:

Línea 2: introducción de la contraseña incorrectamente.

Explicación del formato expandido:

Caso de uso: Nombre del caso de uso.

Actores: Lista de actores(agentes externos), en la cual se indica quién inicia el caso de uso.

Propósito: Intención del caso de uso.

Resumen: Repetición del caso de uso de alto nivel o alguna síntesis similar.

Tipo: 1. Primario, secundario u opcional.

2. Esencial o real (a explicar).

Referencias cruzadas: Casos relacionados de uso y funciones también relacionadas del sistema.

La sección intermedia, ***curso normal de los eventos***, es la parte principal del formato expandido; describe los detalles de la conversión interactiva entre los actores y el sistema. Explica la secuencia más común de los eventos: la historia normal de las actividades y la terminación exitosa de un proceso. No incluye situaciones alternas.

Curso normal de los eventos:

Acción del actor	Respuesta del sistema
Acciones numeradas de los actores.	Descripciones numeradas de las respuestas del sistema.

La última sección, ***curso alterno de los eventos***, describe importantes opciones o excepciones que pueden presentarse en relación con el curso normal. Si son complejas, podemos expandirlas y convertirlas en nuestros casos de uso.

Cursos alternos:

Alternativas que pueden ocurrir en el número de línea. Descripción de excepciones.

11.6.5 Actores.

El actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Estimula al sistema con eventos de entrada o recibe algo de él. Los actores están representados por el papel que desempeñan en el caso: Alumnos, Registro etc.

Para facilitar su identificación en la narrativa del caso, es conveniente escribir su nombre con mayúscula en la narrativa del caso para facilitar la identificación.

Un usuario puede desempeñar varios papeles. Por ejemplo, un comerciante de edad madura podría desempañar el papel de gerente de comercio y además ser un comerciante normal. Por otra parte un comerciante puede ser también agente de ventas. Cuando se trata con actores, conviene pensar en los papeles, no en las personas ni en los títulos de sus puestos.

Los actores llevan a cabo casos de uso. Un mismo actor puede realizar muchos casos de uso; a la inversa, un caso de uso puede ser realizado por varios actores.

En la práctica, los actores son muy útiles cuando se trata de definir los casos de uso. Al enfrentarse con un sistema grande, puede ser difícil obtener una lista de casos de uso. Es mas fácil en tales situaciones definir la lista de los actores y después tratar de determinar los casos de uso de cada actor.

En UML, se representa un actor con una figura humana (monigote), pero algunos prefieren utilizar un icono con figura de computadora para designar los actores que son sistemas de cómputo y no seres humanos.

Figura 2.2 Representación de un actor

Figura 2.2 Representación de un actor máquina

Obsérvese que no es necesario que los actores sean seres humanos, a pesar de que los actores estén representados por monigote humanas en la figura 2.2 el actor puede ser también un sistema externo que necesite cierta información del sistema actual. Se aprecia la necesidad de actualizar las cifras del sistema de contabilidad.

El tema de las interacciones con sistemas externos produce mucha confusión y variaciones de estilo entre los usuarios de los diagramas de caso de uso.

1 Algunos sienten que todas las interacciones con sistemas remotos deben aparecer en el diagrama. Por ejemplo, si se necesita acceder a Reuter con el fin de cotizar un contrato, se deberá mostrar el vínculo entre el caso Negocia precio y Reuters.

2 Algunas personas consideran que sólo se deben mostrar los casos de uso con interacción externa, cuando quien inicia el contacto es otro sistema de contabilidad si dicho sistema invocara algún proceso que le dijera al sistema fuente que lo hiciera.

Algunas personas consideran que sólo se deben mostrar los actores del sistema cuando ellos sean los que necesiten el caso de uso. Por tanto, si el sistema genera cada noche un archivo que después es recogido por el sistema contabilidad, entonces éste es el actor que corresponde, debido a que es quien necesita el archivo generado.

Otros más sienten que constituye un enfoque equivocado considerar que un sistema es un actor. Por el contrario, consideran que un actor es un usuario que desea algo del sistema (por ejemplo, un archivo en particular).

Tomando en cuenta todos los factores, la recomendación es por la opción 3.

Todos los casos de uso tratan sobre funcionalidad requerida externamente. Si el sistema de contabilidad necesita un archivo, entonces ése es un requerimiento que debe satisfacer.

El acceso a Reuter es importante, pero no una necesidad del usuario. Si sigue la opción 4, se termina por analizar el sistema de contabilidad, cosa que probablemente usted no quisiera hacer. Dicho lo anterior, siempre se deben cuestionar los casos de uso con los actores del sistema, descubrir los objetivos reales del usuario y considerar formas alternas para lograrlos.

Cuando trabaje con actores y casos de uso, no se preocupe mucho por la relación exacta entre ellos. Lo que interesa casi siempre son los casos de uso; los actores son sólo un modo de llegar a ellos. Siempre y cuando obtenga todos los casos de uso, no se preocupe por los detalles acerca de los actores.

Sin embargo, una situación en la que los actores sí desempeñan un papel es en la configuración del sistema para varios tipos de usuarios. Si el sistema tiene casos de uso que corresponden a funciones de usuarios de alto nivel, usted puede emplear los vínculos entre casos de uso y actores para hacer los perfiles de los usuarios. Cada usuario tendría un lista asociada de nombres de actores con la que se determinaría los casos que puede ejecutar cada uno.

Otra buena razón para rastrear a los actores es la necesidad de saber cuáles son los casos de uso que quiere cada uno. Esto puede ser importante cuando usted esté evaluando las necesidades que compiten entre ellos. La Comprensión de los actores puede servir para negociar entre demandas de desarrollo que compiten entre sí. También pueden ser útiles para especificar una política de seguridad.

Algunos casos de uso no tienen vínculos claros con actores específicos. Considérese una compañía de servicios públicos. Por supuesto, uno de sus casos de uso es “envío de factura”. No es tan fácil, sin embargo, identificar un actor asociado. No existe un papel de usuario en particular que solicite una factura. La factura se envía al cliente, pero el cliente no protestaría si esto no se llevara a cabo. Lo que más se parece a un actor es el departamento de facturación, en el sentido de que obtiene un valor del caso de uso. Pero la facturación generalmente no interviene en la ejecución del caso de uso.

Los actores pueden tener varios papeles con respecto a un caso de uso. Pueden ser los que obtienen un valor del caso de uso, o tal vez sólo participen en él. Dependiendo de cómo se utilice la relación entre los actores será la importancia de los papeles que desempeñen los actores.

La clave es tener en cuenta que algunos casos de uso no saltarán a la vista como resultado de ponerse a pensar en los casos de uso de cada actor. Si esto sucede, no se preocupe demasiado. Lo importante es comprender los casos de uso y los objetivos del usuario que satisfacen.

Una buena fuente para identificar los casos de uso son los eventos externos. Piense en todos los eventos del mundo exterior ante los cuales usted quiere reaccionar. Un evento dado puede provocar una reacción en el sistema en la que no intervenga el usuario, o puede causar una reacción que provenga principalmente de los usuarios. La identificación de los

eventos ante los que se necesitará reaccionar será de ayuda en la identificación de los casos de uso.

11.6.6 Uses y extends

además de los vínculos entre los actores y los casos de uso, hay otros dos tipos de vínculos estos representa las relaciones de uses (usa) y extends (extiende) entre los casos de uso. Con frecuencia, tales relaciones son fuentes de confusión para quienes mezclan los significados de ambos conceptos, de modo que tómese el tiempo necesario para comprenderlos.

Analicemos otro ejemplo de caso de uso entre gerente, sistema de contabilidad, comerciante y agente de venta.

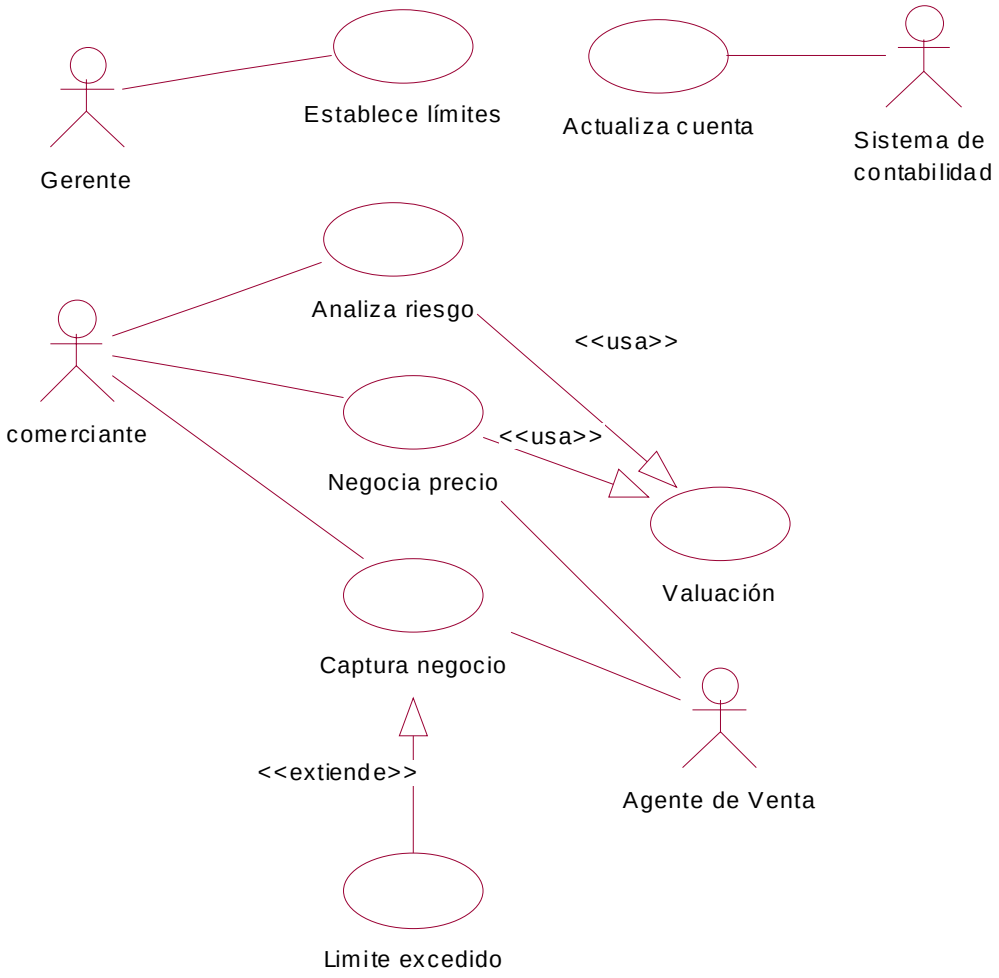


Figura 2.3 **Diagrama de caso de uso**

Se usa la relación extends cuando se tiene un caso de uso que es similar a otro, pero que hace un poco más.

En el ejemplo de la figura 2.3, el caso de uso es Captura negociación. Este es un caso en que todo sucede sin contratiempos. Sin embargo, hay situaciones que pueden estropear la captura de una negociación. Una de ellas surge cuando se excede algún límite, por ejemplo, la cantidad máxima que la organización de comercio ha establecido para un cliente en particular. En este ejemplo, dando que no efectuamos el comportamiento habitual asociado con dicho caso de uso, efectuamos una variación.

Podríamos poner esta variación dentro del caso de uso Captura negociación. Sin embargo, esto llenaría dicho caso de uso de una gran cantidad de lógica especial, la cual oscurecería el flujo "normal".

Otro modo de abordar la variación es poner la conducta normal en un caso y la conducta inusual en cualquier otro lado. A continuación se describe la esencia de la relación extends.

1 Primero obtenga el caso de uso simple y normal

2 En cada paso de ese caso de uso, pregúntese "¿qué puede fallar aquí?", "¿cómo podría funcionar esto de modo diferente?"

3 Dibuje todas las variaciones como extensiones. Si se listan por separados serán mucho más fáciles de entender.

Puede suceder que se tenga qué dividir un caso de uso tanto en la etapa de elaboración como en la de construcción. Durante la elaboración, se suele dividir los casos de uso que se estén volviendo muy complicados. No obstante, hay otros casos de uso cuya complejidad no se recomienda abordarlo a fondo, sino hasta llegar a la construcción.

En la etapa de construcción del proyecto, se realiza la división cuando no se puede construir un caso de uso completo en una sola iteración. Divida los casos de uso complejos en un caso normal y unas cuantas extensiones, y luego construya el caso normal en una sola iteración y las extensiones como parte de una o varias iteraciones posteriores. (Por supuesto, aquí sucederán cambios en el plan de compromiso, lo que deberá negociarse con los usuarios).

Las relaciones **uses** ocurren cuando se tiene una porción de comportamiento que es similar en más de un caso de uso y no se quiere copiar la descripción de tal conducta. Por ejemplo, tanto Analiza riesgo como Negociación precio requieren valuar la negociación. La descripción de la evaluación de la negociación requiere de bastante escritura.

Adviértase la diferencia entre extends y uses. Ambos implican la factorización de comportamiento comunes de varios casos de uso, dejando un solo caso de uso común que es empleado, o extendido, por otros varios casos de uso. No obstante, la intención es la que cambia.

En sus vínculos con los actores, estos dos tipos de relación implican asuntos diferentes. Tratándose de la relación extends, los actores tienen que ver con los casos de uso que se están extendiendo. Se supone que un actor dado se encargará tanto del caso de uso base como de todas las extensiones. En cuanto a las relaciones uses, es frecuente que no haya un actor asociado con el caso de uso común. Incluso si lo hay, no se considera que esté llevando a cabo los demás casos de uso.

Aplique las siguientes reglas.

1 Utilice extends cuando describa una variación de conducta normal.

2 Emplee uses para repetir cuando se trate de uno o varios casos de uso y desee evitar repeticiones.

Es probable que oiga usted el término escenario en conexión con los casos de uso. Cabe aclarar que este término se emplea de manera inconsistente. Algunas veces, escenario es usado como sinónimo de caso de uso. En el contexto del UML, la palabra escenario se refiere a una sola ruta a través de un caso de uso, una ruta que muestra una particular combinación de condiciones dentro de dichos caso de uso. Por ejemplo, para ordenar algunas mercancías, tendremos un solo caso de uso con varios escenarios asociados: uno en el cual todo va bien; otro donde no hay suficientes mercancías; otro en el que nuestro crédito es rechazado, y así por el estilo.

Conforme vaya realizando sus tareas de modelado, encontrará modelos que expresen la manera de hacer sus casos de uso, ya sea entre el software o entre la gente. Es evidente que hay más de una manera de llevar un caso de uso.

En un caso de uso hay un **actor iniciador** que produce la estimulación inicial, y otros **actores participantes**. Es conveniente indicar quién es el iniciador.

A como se menciono anteriormente actores también pueden ser los papeles representados por seres humanos pero puede ser cualquier tipo de sistema, como un sistema computarizado externo de bancos. Como ejemplo podemos mencionar:

- Papeles que desempeñan las personas.
- Sistemas de cómputo.
- Aparatos electrónicos o mecánicos.

Un error común en la identificación de los casos de uso es representar los pasos, las operaciones o las transacciones individuales como casos por ejemplo en el dominio de registro , podemos definir incorrectamente un caso denominado extender “recibo de matricula”, cuando en realidad esta operación no es más que un paso de un proceso más amplio del caso Realizar matricula. *Un caso de uso es una descripción amplia de un proceso de principio a fin, descripción que abarca muchos pasos o transacciones, no es un paso ni una actividad individual del proceso.*

Es posible dividir las actividades o parte del caso en subcasos, denominados casos ***abstractos de uso***, incluso en pasos individuales, pero esto no es lo habitual.

11.6.7 Identificación de los casos de uso.

Los pasos presentados a continuación para identificar los casos de uso, requieren la revisión del documento de especificación de los requerimientos.

Un método con que se identifican los casos de uso se basa en los actores:

- Se identifican los actores relacionados con un sistema o empresa.
- En cada actor, se identifican los procesos que inician o en que participan.

Un segundo método de identificación de casos de uso se basa en eventos:

- Se identifican los eventos externos a los que un sistema ha de responder.
- Se relacionan los eventos con los actores y con los casos de uso.

En nuestro ejemplo, algunos actores y procesos en que participan son:

Alumno.

Solicita matricula.

Cancela matricula

Cancelar colegiatura

Cajero.

Registra.

Entrega efectivo

11.6.8 Casos de uso y procesos del dominio.

Un caso de uso describe un proceso. Un proceso se describe de comienzo a fin, una secuencia de los eventos, de las acciones y las transacciones que se requieren para producir u obtener algo de valor para una empresa o actor. Ejemplos de procesos son los siguientes:

- Tramitar certificado de notas.
- Cancelar matrícula.
- Ordena listas de alumnos.
- Registra cursos que se imparten en una escuela.
- Verifica la ortografía de un documento con un procesador de palabras.
- Realiza una llamada telefónica.

11.6.9 Caso de uso, funciones del sistema y rastreabilidad

Las funciones del sistema identificadas durante la especificación previa de requerimientos deben asignarse a los casos de uso. Además debe ser posible verificar, mediante la sección *Referencias cruzadas*, que todas las funciones haya sido asignadas. Con ello se logra un vinculo importante respecto a la rasteabilidad entre los artefactos. En definitiva, todas las funciones y casos de uso del sistema deberían poder rastrearse hasta la implementación y la aplicación de pruebas.

11.6.10 Bibliografía:

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 6: Casos de uso, descripción de procesos. Páginas 47-54. Realiza una introducción al concepto de caso de uso y muestra ejemplos sobre los distintos formatos de representación que existen.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 117-129. Aborda de manera amplia el concepto de caso de uso.

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 16: Casos de uso. Páginas 191-202. Describe el concepto de caso de uso, relacionándolo con otros aspectos como flujo de eventos, escenarios, colaboraciones, etc.

UML Distilled “Applyng the standard Object Modeling Language”. Martin Fowler. Editorial Addison Wesley. Capítulo 3: Use Cases. Páginas 43-51. Explica de manera breve las principales características de los casos de uso.

UML Gota a Gota. Martin Fowler. Kendall Scott. Editorial Addison Wesley. Capítulo 3: Use Cases. Páginas 50..58

11.7 CLASE 7: DIAGRAMAS DE CASOS DE USO

Diagramas de casos de uso

Formatos de los casos de uso

Los sistemas y sus fronteras

Casos de uso primarios, secundarios y opcionales

Casos esenciales de uso comparados con los casos reales de uso

Notación de los casos de uso

Casos de uso dentro de un proceso de desarrollo

OBJETIVOS:

Diseñar diagramas de casos de uso.

Identificar los formatos de los casos de uso

Identificar los sistemas y sus fronteras

Contrastar los casos de uso de alto nivel con los expandidos.

Contrastar los casos de uso esenciales con los reales.

1

Jacosbson (1994), además de introducir los casos de uso cómo elementos primarios del desarrollo del Software, también diseño un diagrama para la representación gráfica de los casos de uso. El diagrama de casos de uso es ya también parte de UML.

Un **diagrama de casos de uso** explica gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre éstos.

Los casos de uso se muestran en óvalos y los actores en figuras estilizadas. Hay líneas de comunicaciones entre los casos y los actores; las flechas indican el flujo de la información o el estímulo.

El objetivo del diagrama es ofrecer una clase de diagrama contextual que nos permita conocer rápidamente los actores externos de un sistema y las formas básicas en que lo utilizan.

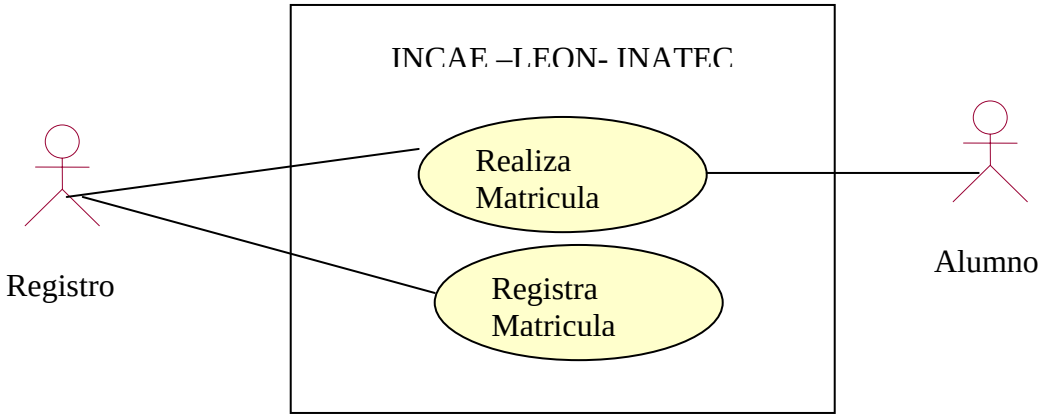


Figura 2.3 **Diagrama parcial de casos de uso**

11.7.2 Formatos de los casos de uso.

Un mismo caso de uso puede escribirse en diferentes formatos y con diversos niveles de detalle. Los podemos dividir en: casos con formato **de alto nivel** y **expandido**.

Formato de alto nivel:

Un **caso de uso de alto nivel** describe un proceso muy brevemente, casi siempre en dos o tres enunciados. Se aconseja su uso durante el examen inicial de los requerimientos y del proyecto, para entender de forma rápida el grado de complejidad y funcionalidad del sistema.

Formato expandido:

Un **caso de uso expandido** describe un proceso más a fondo que el de alto nivel. Se diferencia con éste, en que tiene una sección destinada al *curso normal de los eventos*, que los describe paso por paso. Durante la fase de especificación de requisitos, es conveniente escribir en el formato expandido los casos más importantes y de mayor influencia.

11.7.3 Los sistemas y sus fronteras.

Un caso de uso describe la interacción con un sistema. Las fronteras del sistema son:

- La frontera hardware/software de un dispositivo o sistema de cómputo.
- El departamento de una organización.
- La organización entera.

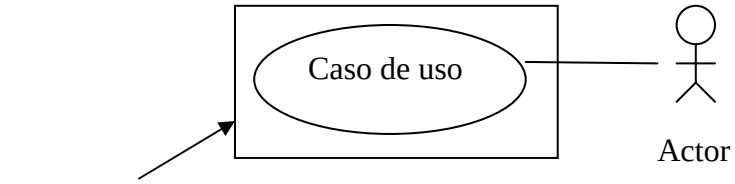
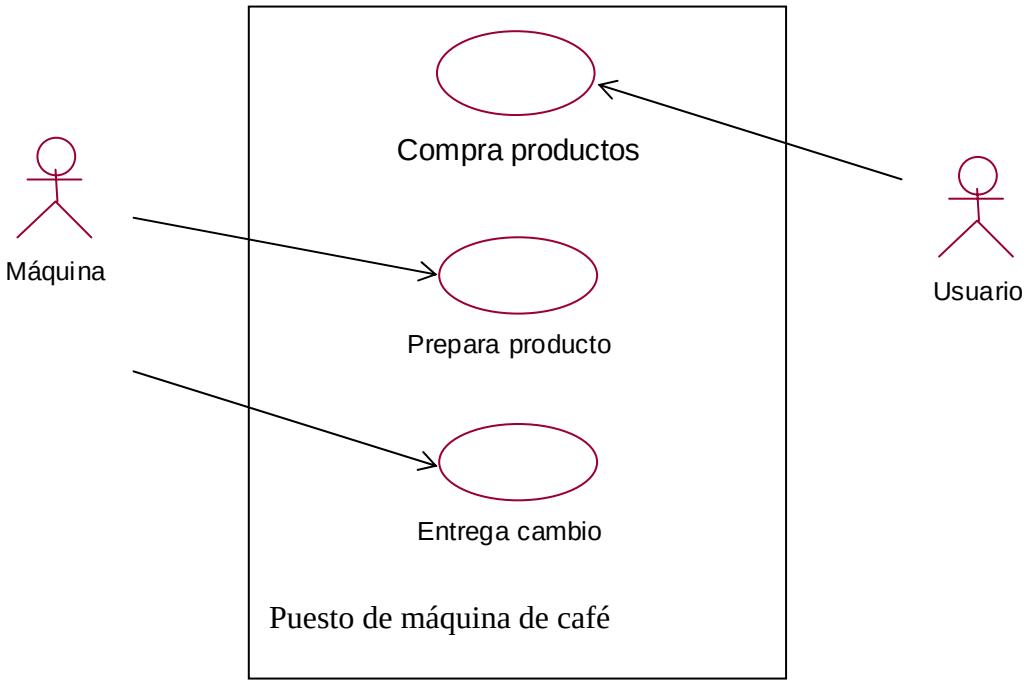


Figura 2.4 **Frontera de un caso de uso**

Es importante definir la frontera del sistema para identificar lo que es interno o externo, así como las responsabilidades del sistema. El ambiente externo está representado únicamente por actores.

Estudiaremos un ejemplo de la influencia que tiene seleccionar la frontera del sistema: la compra del producto en una máquina de café en el supermercado. Si elegimos como "el sistema" el supermercado, el único actor es el usuario y no la máquina, porque este último es un recurso del supermercado que realiza las funciones. Pero si escogemos como sistema el hardware y el software de la máquina de café, se trata como actores al usuario y a la máquina.

Ambas situaciones se ilustran mejor a continuación:



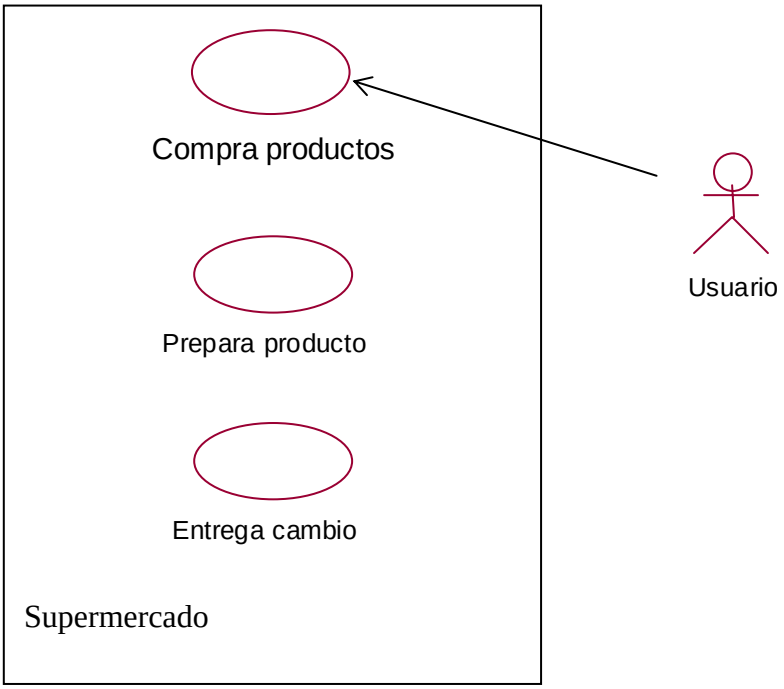


Figura 2.5 **Casos de uso y actores cuando las fronteras son distintas**

11.7.4 Casos de uso primario, secundarios y opcionales.

Los casos de uso se deberían clasificarse en primario, secundario u opcionales, mas adelante, a partir de estas designaciones, clasificaremos nuestro conjunto de casos de uso para establecer en su desarrollo.

- **Casos primarios de uso:** representan los procesos comunes más importantes, como ***Realizar matricula.***
- **Casos secundarios de uso:** representan procesos menores o raros, por ejemplo: ***Cancelar matricula.***
- **Casos opcionales:** representan procesos que pueden no abordarse.

11.7.5 Casos esenciales de uso comparados con los casos reales de uso.

Casos esenciales de uso:

Los **casos esenciales de uso** son casos expandidos que se expresan en una forma teórica que contiene pocos detalles de implementación; las decisiones de diseño se posponen, especialmente las de la interfaz para el usuario. Estos casos describen el proceso a partir de sus actividades.

Los casos de alto nivel siempre son de carácter esencial, debido a su brevedad y abstracción.

Como ejemplo podemos citar un caso de ***Retiro de efectivo*** en un cajero automático, que se expresa de forma esencial.

Acción de los actores	Respuesta del sistema
1- El cliente se identifica	2- Presenta opciones
3- y así sucesivamente	4- y así sucesivamente

La forma en que un cliente se identifica cambia con el tiempo(es una decisión de diseño), pero forma parte del proceso esencial de que la identificación se realice de alguna manera.

Conviene crear casos esenciales de uso al comenzar a investigar los requerimientos, para entender mejor el alcance del problema y las funciones necesarias. Este tipo de casos permiten captar la esencia del proceso y sus motivos fundamentales, sin ahondar en detalles de diseño.

Casos reales de uso:

Un **caso real de uso** describe el proceso a partir de su diseño concreto actual, sujeto a las tecnologías de entrada y de salida. Cuando se trata de la interfaz para el usuario, ofrece presentaciones de pantalla y explica la interacción con los artefactos. Por ejemplo, el caso ***Retiro de efectivo*** expresado en forma real:

Acción de los actores	Respuesta del sistema
1. El cliente introduce su tarjeta.	2. Pide el número de identificación personal(NIP).
3. Introduce el NIP con un teclado numérico.	4. Muestra el menú de opciones.
5. y así sucesivamente.	6. y así sucesivamente.

Observe que la acción esencial del “Cliente se identifica a sí mismo”, se realizó ahora en la serie de acciones comenzando con “El cliente introduce su tarjeta”.

Los casos reales de uso se crean durante la fase de diseño por ser un elemento de éste. Sin embargo, en algunos proyectos las primeras decisiones de diseño con respecto a la interfaz para el usuario, se discuten en la fase de elaboración, debido a esto, se establecen casos reales en dicha fase.

11.7.6 Notación de los casos de uso.

Asignación de nombres a los casos de uso:

El nombre de un caso de uso debe comenzar con un verbo para especificar que se trata de un proceso. Ejemplo:

- Preparar lista de alumnos.
- Introducir datos de alumnos.
- Recibir notas.

11.7.7 Cuando emplear casos de uso

Es difícil imaginarse una situación en la que no se emplee caso de uso. Son una herramienta esencial para la captura de requerimiento, la planificación, o el control de proyecto iterativos. La captura de los casos de uso es una de las tareas principales durante la fase de elaboración; de hecho, es lo primero que se debe hacer.

La mayoría de los casos de uso se generarán durante esa fase del proyecto, pero irá descubriendo otros a medida que avance. Esté siempre pendiente de ellos. Todo caso de uso es un requerimiento potencial y hasta que no haya usted capturado un requerimiento, no podrá planear manejarlo en el proyecto.

Algunos prefieren listar y analizar los casos de uso primero, y luego llevar a cabo un poco de modelado. El modelado conceptual con los usuarios ayuda a descubrir los casos de uso.

Los diseñadores emplean casos de uso con distintos grado de granularidad. Por ejemplo Ivar Jacobson dice que en un proyecto de 10 personas-año él esperaría unos 20 casos de uso (sin contar con las relaciones use y extends). En un proyecto reciente expuesto el autor. Martin Fowler. Trabajo con 100 casos de uso, pero prefiere los casos de uso con menor granularidad, pues con ellos es más fácil trabajar sin perder de vista el calendario de compromiso. Sin embargo, demasiados casos de uso pueden ser abrumadores, por el momento no hay una respuesta correcta, de modo que se recomienda que sea flexible y trabaje con lo que le resulte más cómodo.

11.7.8 Bibliografía:

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 6: Casos de uso, descripción de procesos. Páginas 55-71. Introduce los conceptos de diagramas de casos de uso y de casos reales de uso.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 117-129.

UML Gota a Gota. Martin Fowler. Kendall Scott. Editorial Addison Wesley. Capítulo 3: Use Cases. Páginas 58

11.8 CLASE 8: DIAGRAMAS DE CASOS DE USO

Diagramas de casos de uso

Formatos de los casos de uso

Los sistemas y sus fronteras

Casos de uso primarios, secundarios y opcionales

Casos esenciales de uso comparados con los casos reales de uso

Notación de los casos de uso

Casos de uso dentro de un proceso de desarrollo

OBJETIVOS:

Diseñar diagramas de casos de uso.

Contrastar los casos de uso de alto nivel con los expandidos.

Contrastar los casos de uso esenciales con los reales.

11.8.1 Inicio de un caso expandido de uso:

Comience un caso expandido con el siguiente esquema:

Este caso comienza cuando <Actor><inicia un evento>

Por ejemplo:

1. Este caso de uso comienza cuando el alumno solicita matricula.

De esta manera, se identifica claramente el actor y el evento iniciadores.

11.8.2 Decisión de notación y ramificación:

Un caso de uso puede contener puntos de decisión. Si una de las trayectorias de decisión es un caso muy representativo y las otras alternativas son raras, inusuales o excepcionales, el caso típico debe ser el único acerca del cual se escribe en el *Curso normal de los eventos* y las opciones deben de escribirse en la sección *Alternativas*.

Cuando el punto de decisión representa opciones cuya probabilidad es igual y normal, se usa la siguiente forma de notación:

En la sección principal *Curso normal de los eventos*, indique las ramas de las subsecciones.

Escriba una subsección en cada rama, usando otra vez *Curso normal de los eventos.* Inicie el evento numerando en 1 cada sección.

Si las subsecciones tienen opciones, escríbalas en una sección de alternativas de cada subsección.

11.8.3 Sección: principal

Curso normal de los eventos

Acción del actor	Respuesta del sistema
1.Este caso de uso comienza cuando el alumno solicita cancelar la matricula.	
2. El cajero introduce el código del alumno.	3. Muestra el estado de cuenta del alumno
	4. Calcula y presenta el total de las colegiaturas en mora.
5. El Cajero indica el total al alumno.	
6.El cliente escoge la forma de pago: a- Si paga en efectivo, véase las sección de pago en efectivo. b- Si paga con tarjeta de crédito, véase las sección de pagar con tarjeta de crédito. c- Si paga con cheque, véase la sección Pagar con cheque.	
	7. Registra la cancelación terminada.
	8. Actualiza el estado de cuenta
	9. Genera un recibo
10. El cajero entrega un recibo al alumno	
11. El alumno se marcha con el recibo	

11.8.4 Sección: Pago de colegiatura en efectivo

Curso normal de los eventos

Acción del actor	Respuesta del sistema
1. El alumno de un pago en efectivo –el "efectivo ofrecido", posiblemente mayor que el total del arancel de colegiatura	
2. El cajero registra el efectivo ofrecido.	3. Muestra al cliente la diferencia
4. El cajero deposita el efectivo recibido y extrae la diferencia. El cajero entrega al alumno en cambio del pago.	

Cursos alternativos:

Línea 4: efectivo insuficiente en la caja para pagar la diferencia. Se pide al supervisor o se pide al alumno un pago más cercano al total de la venta.

11.8.5 Casos de uso dentro de un proceso de desarrollo.

Pasos para la elaboración de las fase de planeación y elaboración:

Después de haber listado las funciones del sistema, defina la frontera e identifique los actores y casos de uso.

Escriba todos los casos de uso en el formato de *alto nivel*. Clasifíquelos en primarios, secundarios u opcionales.

Dibuje el diagrama de casos de uso.

Relacione los casos de uso y dé ejemplo de las relaciones en el diagrama correspondiente.

Escriba en el formato *esencial expandido* los casos de uso más importantes, para entender mejor el problema.

Los *casos reales* deberían posponerse hasta una fase de diseño, porque su creación conlleva decisiones de diseño. Sin embargo, a veces es necesario crear casos reales de uso durante la etapa de los requerimientos si:

Las descripciones facilitan la comprensión.

Los clientes exigen especificar sus procesos en esta forma.

Identifique los actores y casos de uso:

Defina la frontera del sistema que será el sistema de hardware/software. Un ejemplo de lista de los actores y procesos relevantes son los siguientes:

Cajera	Cobra el servicio de las matricula y pago de mensualidades.
Alumno	Realiza matricula, llena los espacios en blancos de la matricula
Resp. De Registro	Incorpora nuevos usuarios, realiza actualizaciones de notas
Administrador del Sistema	Asigna permisos al sistema

Escriba los casos de uso en el formato de alto nivel:
Una muestra de casos de uso de alto nivel comprende:

Caso de uso: Solicitar Matricula
Actores: Alumno, Cajero
Propósito: Capturar una matricula.
Tipo: Primario

Resumen: Antes de realizar cualquier otra actividad educativa, el alumno debe matricularse.

Se presenta a continuación el diagrama de nuestro ejemplo:

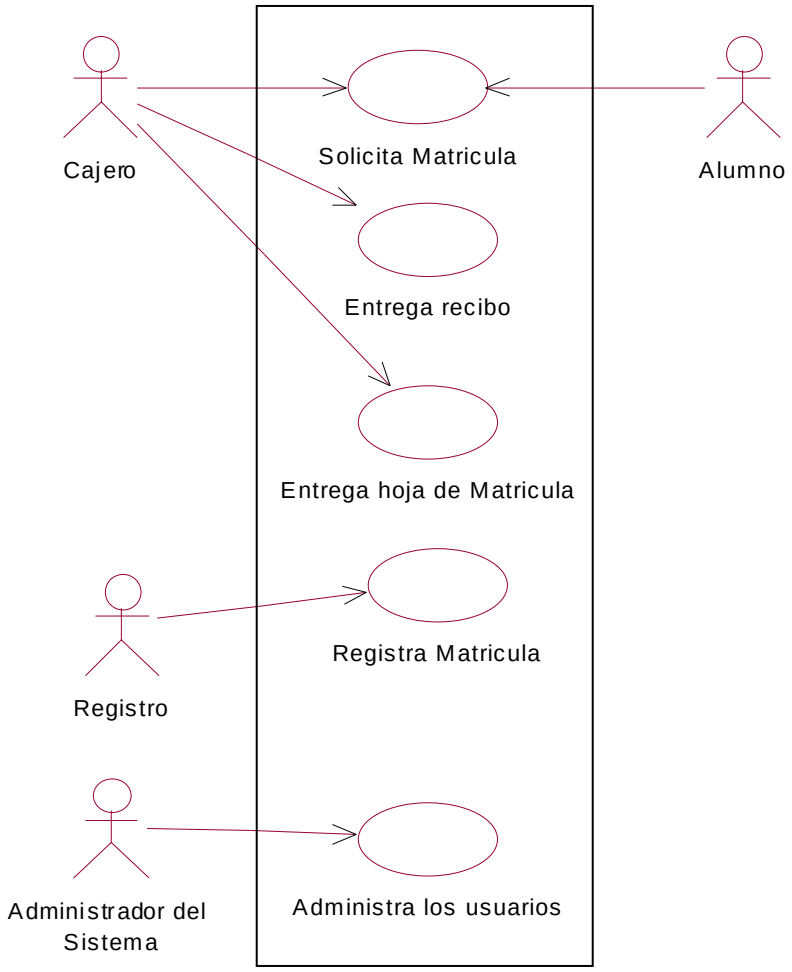


Figura 2.7 **Diagrama de casos de uso de Realizar Matricula**

Relacione los casos de uso:

Este tema se abordará posteriormente.

Escriba algunos casos esenciales expandidos de uso:

Entre los casos primarios de uso realmente significativos figuran:

Solicitar Matricula.

Escribirlo en una forma esencial expandida suministrará una mayor información y claridad de los requerimientos. A continuación se presenta el caso de uso *Solicitar Matricula* en su forma esencial expandida completa:

Caso de Uso: Cancelar colegiatura.

Actores: Alumno, Cajera

Propósito: Capturar una cancelación de colegiatura.

Resumen: Un alumno se presenta a caja con la intención de cancelar la colegiatura.. cancela el coste de la colegiatura entregan recibo, el alumno se marcha .

Tipo: Primario y esencial.

Referencias

Cruzadas: Funciones: R13,R13.1, R13.2, R13.3, R13.4

Curso normal de los eventos

Acción del actor	Respuesta del sistema
1.Este caso de uso comienza cuando el alumno solicita cancelar la matricula.	
2. El cajero introduce el código del alumno.	3. Muestra el estado de cuenta del alumno
	4. Calcula y presenta el total de las colegiaturas en mora.
5. El Cajero indica el total al alumno.	
6.El cliente escoge la forma de pago: d- Si paga en efectivo, véase las sección de pago en efectivo. e- Si paga con tarjeta de crédito, véase las sección de pagar con tarjeta de crédito. f- Si paga con cheque, véase la sección Pagar con cheque.	
	7. Registra la cancelación terminada.
	8. Actualiza el estado de cuenta
	9. Genera un recibo
10. El cajero entrega un recibo al alumno	
11. El alumno se marcha con el recibo	

Cursos alternos
Línea 2: se introduce un código invalido. Indique el error
Línea 6: El alumno no dispone de suficiente dinero para cancelar la matricula. Cancele la operación de actualización.

Sección: pagar en efectivo

Acción del actor	Respuesta del sistema
1. El alumno de un pago en efectivo –el "efectivo ofrecido", posiblemente mayor que el total del arancel de colegiatura	
2. El cajero registra el efectivo ofrecido.	3. Muestra al cliente la diferencia
4. El cajero deposita el efectivo recibido y extrae la diferencia. El cajero entrega al alumno en cambio del pago.	

Cursos alternativos:

Línea 4: efectivo insuficiente en la caja para pagar la diferencia. Se pide al supervisor o se pide al alumno un pago más cercano al total de la venta.

Sección: pago con tarjeta de crédito

Acción del actor	Respuesta del sistema
1.El alumno comunica su información de crédito para pagar con tarjeta.	2. Genera una solicitud de pago con tarjeta de crédito y la envía a un servicio externo de autorización de crédito.
3. El servicio de autorización de crédito autoriza el pago.	4. Recibe una respuesta aprobatoria de crédito del servicio de autorización de crédito. 5. En el sistema de cuentas por cobrar registra la información sobre el pago con tarjeta de crédito y la respuesta de aprobación. El servicio de autorización de crédito debe dinero al centro; por tanto, cuentas por cobrar debe darle seguimiento. 6. Muestra el mensaje aprobatorio de autorización.

11.8.6 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 6: Casos de uso, descripción de procesos. Páginas 55-71. Introduce los conceptos de diagramas de casos de uso y de casos reales de uso.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 117-129.Presenta las reglas para la construcción de los diagramas de casos de uso.

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 17: Diagramas de casos de uso. Páginas 203-210. Resume las principales características de los diagramas de casos de uso.

11.9 CLASE 9: CONSTRUCCION DEL MODELO CONCEPTUAL

Modelos conceptuales

Estrategias para identificar los conceptos

Directrices para construir modelos conceptuales

OBJETIVOS:

Crear un modelo conceptual preliminar.

Distinguir entre los atributos correctos y los incorrectos.

11.9.1 Modelos conceptuales.

El modelo conceptual es el elemento más importante a crear durante el análisis Orientado a Objetos1. El paso esencial de un análisis orientados a objetos es descomponer el problema en conceptos u objetos individuales. Un **modelo conceptual** es una representación de conceptos en un dominio del problema. En UML, se ilustra con un grupo de **diagramas de estructura estática** donde no se define ninguna operación. *Una cualidad esencial que debe tener un modelo conceptual es que representa cosas del mundo real, no componentes del software.*

Puede mostrarnos:

Conceptos.

Asociaciones entre conceptos.

Atributos de conceptos.

Además de descomponer el problema en conceptos, el modelo conceptual nos permite esclarecer la terminología o nomenclatura del dominio. Comunica cuáles son los términos importantes y cómo se relacionan entre sí.

Los modelos conceptuales no son modelos de diseño de software:

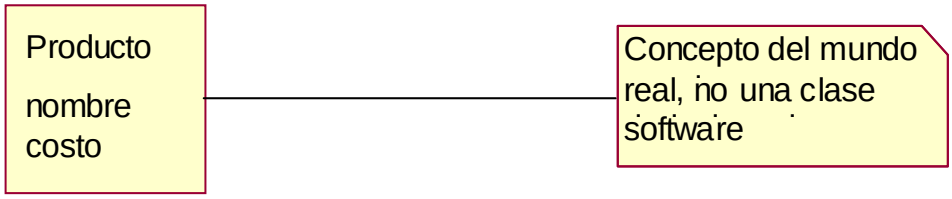
Como se mencionó anteriormente, el modelo conceptual no es una descripción del diseño del software, como una clase de C++. Debido a esto, no son adecuados en él los siguientes elementos:

---

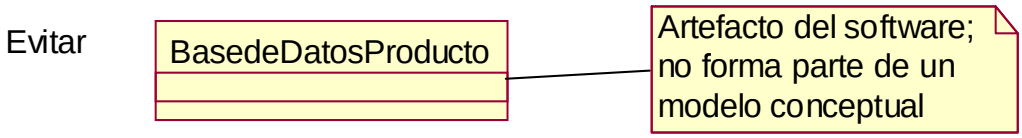
1 Los casos de uso son importantes en el análisis de requerimientos, pero no están orientados a objetos.

Los artefactos de software, como una ventana o una base de datos, excepto que el dominio a modelar se refiera a conceptos de software, por ejemplo, un modelo de interfaces gráficas.

Los métodos2 .



Un modelo conceptual muestra conceptos del mundo real



Un modelo conceptual no muestra los artefactos del software

Figura 2.8 **Un modelo conceptual muestra conceptos del mundo real, no muestra los artefactos del software**

11.9.2 Conceptos:

En términos informales el concepto es una idea, cosa u objeto. En un lenguaje más formal, podemos considerarlo a partir de su símbolo, intención3 y extensión.

Símbolo: palabras o imágenes que representan un concepto.

Intención: la definición del concepto.

Extensión: el conjunto de ejemplos a que se aplica el concepto.

2 Los métodos siempre se relacionan con entidades del software, por lo que no debe incluirse.

3 Intensión designa el grado de una cualidad.

Por ejemplo, a partir del caso de uso "Realizar Matricula", derivamos el símbolo *Matricula*.

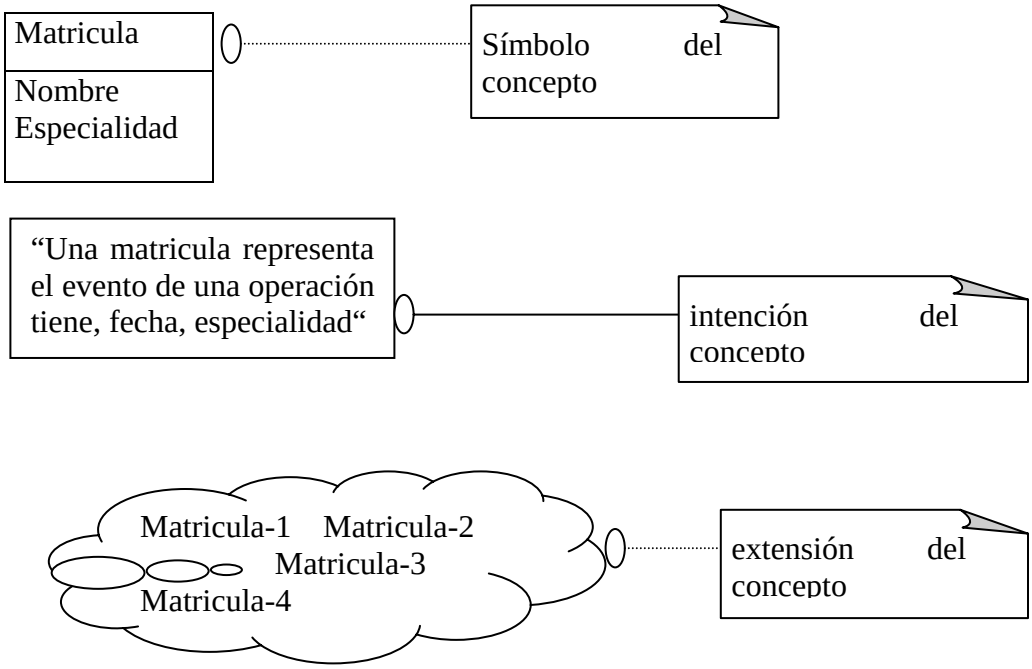


Figura 2.9 **El concepto tiene un símbolo, intención y extensión**

Cuando se crea un modelo conceptual, la vista del símbolo y de la intención de un concepto es el aspecto de mayor interés.

11.9.3 Los modelos conceptuales y la descomposición:

Una estrategia para resolver la complejidad de los problemas de software, es, dividir el problema en unidades comprensibles. En el ***análisis estructurado*** la descomposición se realiza mediante procesos o funciones. En el ***análisis orientado a objetos***, se lleva a cabo fundamentalmente con objetos.

Una tarea importante en la fase de análisis consiste en identificar varios conceptos en el dominio del problema y documentar los resultados en un modelo conceptual.

Figura 2.10 **Modelo conceptual parcial del dominio del INTAE-LEON.**

11.9.4 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 9: Construcción de un modelo conceptual. Páginas 85-103. Explica el concepto de modelo conceptual y las estrategias para su elaboración.

11.10 CLASE 10: CONSTRUCCION DEL MODELO CONCEPTUAL

Estrategias para identificar los conceptos

Obtención de conceptos a partir de una lista de categorías de conceptos

Obtención de conceptos a partir de la identificación de frases nominales

Objetos del Informe

Directrices para construir modelos conceptuales

Especificación o descripción de conceptos

OBJETIVOS:

Incorporar conceptos de especificación cuando convenga.

Comparar los términos: concepto, tipo, interfaz y clase.

Nuestra meta es crear un modelo conceptual de conceptos interesantes o significativo del dominio en cuestión. En este caso, ello significa conceptos relacionados con el caso de uso *Pagar Matricula versión 1*. La tarea fundamental será, pues, identificar los conceptos; se proponen dos estrategias.

La siguiente directriz es de gran utilidad en la identificación de conceptos:

Es mejor exagerar y especificar un modelo conceptual con muchos conceptos refinados que no especificarlo cabalmente. Es decir, que un modelo **no** es más adecuado si tiene menos conceptos.

Es frecuente omitir conceptos en la fase de identificación y descubrirlos cuando se examinen los atributos o asociaciones o durante la fase de diseño. Cuando se detecten, habrá que incorporarlos al modelo conceptual.

No se excluya un concepto simplemente porque los requerimientos no indiquen una necesidad evidente que permita recordar la información acerca de ella o porque el concepto carezca de atributos. Es válido tener conceptos sin atributos o conceptos con un papel puramente de comportamientos en el dominio.

11.10.2 Obtención de conceptos a partir de una lista de categorías de conceptos:

Al crear un modelo conceptual, debemos comenzar preparando una lista de conceptos idóneos a partir de la siguiente lista. Contiene muchas categorías comunes que se debe tener en cuenta, sin importar el orden de importancia.

Categoría del concepto	Ejemplos
Objetos físicos o tangibles	Producto
Especificaciones, diseño o descripciones de cosas	Catálogo de libros Especificación de Productos Descripción de vuelos
Lugares	Tienda, Aeropuerto,
Transacciones	Venta, Pago Reservaciones
Línea o renglón de elemento de transacciones	Descripción Venta Línea de Producto
Papel de las personas	Bibliotecario, Vendedor. Cajero, Piloto
Contenedores de otras cosas	Tienda, Catálogo Cesto, Avión
Cosas dentro de un contenedor	Producto Pasajero
Otros sistemas de cómputo o electromecánicos externos al sistema	Fax Sistema de Autorización de Tarjeta de Crédito Control de Trafico Aéreo
Conceptos de nombres abstractos	Hambre, Acrofobia
Organizaciones	Universidad, Departamento de venta Objeto Línea Aérea
Eventos	Robo, incendio, Venta, Robo, Junta, Vuelo, Accidente, Aterrizaje
Procesos(a menudo no están representados como conceptos, pero pueden estarlo)	SolicitarPedido, Venta de Producto Reservación de Asiento
Reglas y políticas	TramitarPedidoEspecial Política de Reembolso Política de Cancelación
Catálogos	CatalogodeProductos Catalogo de Parte
Registros de finanzas, de trabajo, de contratos de asuntos legales	ContratoProveedor, Recibo, Mayor, Bitácora de Mantenimiento
Instrumentos y servicios financieros	Línea de Crédito, Existencia
Manuales, libros	Manual del Personal Manual de Reparación

Obtención de conceptos a partir de la identificación de frases nominales:

Otra técnica muy útil consiste en identificar las frases nominales en las descripciones textuales del dominio de un problema y considerarlas conceptos o atributos. Los casos expandidos de uso son una excelente descripción que puede conseguirse con este análisis.

Algunas frases nominales son conceptos idóneos, pero algunas pueden ser atributos de conceptos. Una ambigüedad que podemos encontrar, es que varias frases nominales pueden designar el mismo concepto o atributo. Sin embargo, se recomienda usarlo en combinación con la *Lista de categorías de conceptos*.

11.10.3 Conceptos para el dominio del INTAE

A partir de la lista de categoría de conceptos y del análisis de frases nominales generamos una lista de conceptos adecuados para incluirlo en la aplicación del INTAE . La lista está sujeta a la restricción de los requerimientos y simplificaciones que se consideren en el momento: los casos simplificado de uso de **Matricula** son:

Matricula	Alumno
Caja	Cajera
Alumno	Matricula

11.10.4 Objetos del Informe:

El recibo es un registro de una venta o de un pago, así como un concepto relativamente prominente en el dominio de ventas; ¿debe, pues, mostrarse en el modelo? Hay que tener presente algunos factores:

El recibo es un informe de una matricula. En general, no conviene incluirlo en un modelo conceptual, ya que toda su información, proviene de otras fuentes. Este es un buen motivo para excluirlo.

El recibo cumple un papel especial respecto a las reglas de la empresa: al portador le confiere el derecho de presentar su recibo al momento de presentarse al centro. Esta es una razón para incorporarlo al modelo.

El recibo se excluirá, porque las devoluciones por matricula no se incluyen en este ciclo de desarrollo.

El modelo conceptual del INTAE – LEON – INATEC (sólo conceptos)

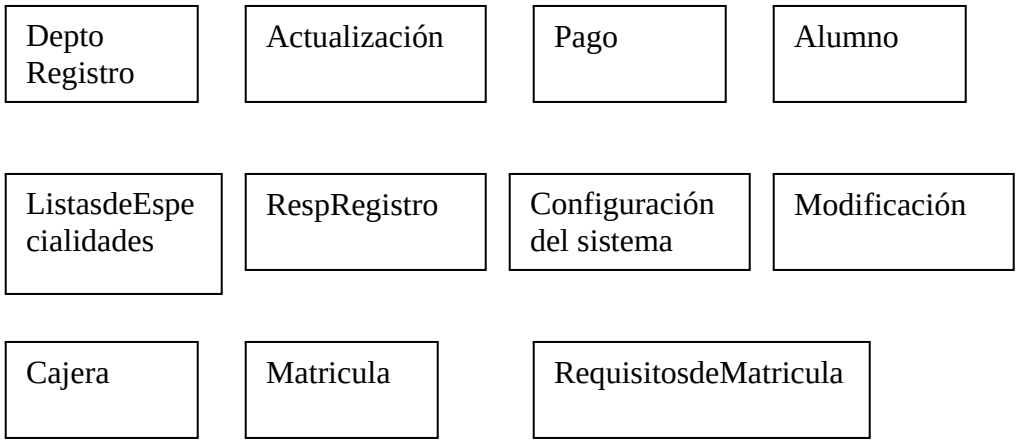


Figura 2.11 **Modelo conceptual inicial del dominio del INTAE-LEON.**

11.10.5 Directrices para construir modelos conceptuales.

Cómo construir un modelo conceptual:

Pasos para construir un modelo conceptual:

Liste los conceptos idóneos usando la *Lista de categorías de conceptos* y la identificación de la frase nominal relacionadas con los requerimientos en cuestión.

Dibújelos en un modelo conceptual.

Incorpore las asociaciones necesarias para registrar las relaciones para las cuales debe reservar un espacio en la memoria.

Agregue los atributos necesarios.

11.10.6 Asignación de nombres:

Para asignar nombres en un modelo conceptual se sugiere lo siguiente:

Utilice el vocabulario del dominio cuando se asignan nombres a los conceptos y a los atributos. Por ejemplo, al desarrollar el modelo de una biblioteca, al cliente se le llamará con nombres que use el personal: Visitante, lector u otro semejante.

Un modelo conceptual puede excluir en el dominio del problema los conceptos que no se relacionen con los requerimientos. Por ejemplo, podemos omitir Bolígrafo en el modelo conceptual de la biblioteca, por no tener una función importante que sea obvia.
El modelo conceptual ha de excluir cosas que no se encuentren en el dominio del problema.

11.10.7 Un error que se comete al identificar los conceptos:

El error más frecuente cuando se crea un modelo conceptual es el de representar algo como atributo, cuando debió de haber sido un concepto. Una regla para no caer en él,

es la siguiente: *Si en el mundo real no consideramos algún concepto X como número o texto, probablemente X sea un concepto y no un atributo.*

Supongamos el caso del dominio de las reservaciones en líneas en un hotel. Debería *Pieza* ser un atributo de *Hotel* o un concepto aparte *Habitación*?

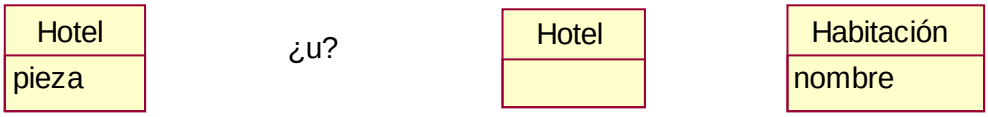


Figura 2.12 **Ejemplo en el dominio de un hotel.**

En el mundo real, una pieza no se considera número ni texto: es una cosa masiva que ocupa espacio. Por lo tanto, Habitación debería ser un concepto.

En caso de duda, convierta el atributo en un concepto independiente.

11.10.8 Especificación o descripción de conceptos.

Suponga lo siguiente:

La instancia de *Elemento* representa una entidad física de un almacén de muebles.

Un *Elemento* tiene una descripción, precio y código de producto, los cuales no están registrados en ninguna otra parte.

Cada vez que se vende un elemento físico, se elimina una instancia del software correspondiente a Elemento.

Tomando en cuenta lo anterior, ¿qué sucedería en el siguiente escenario?

Hay gran demanda de un nuevo mueble: Objeto Escritorio. El almacén vende todas sus existencias, lo cual significa que todas las instancias de Elemento de Objeto Escritorio se cancelan en la memoria de la computadora.

Si alguien pregunta: "¿Cuánto cuesta el Objeto Escritorio?", nadie podrá contestarle porque la memoria de su precio se anexó a las instancias inventariadas, que fueron eliminándose conforme se vendían.

Nótese que si esto se implementa en el software tal como se describe, posee datos duplicados y realiza un gasto innecesario de espacio, porque la descripción, el precio y el código del producto se duplica en cada instancia de Elemento del mismo producto.

11.10.9 Necesidad de las especificaciones:

El problema anterior demuestra la necesidad de un concepto de objetos que son especificaciones o descripciones de otras cosas. Para resolver el problema del *Elemento* lo que se necesita es una *EspecificacióndeProducto* (o DescripcióndeProducto), concepto que registra la información sobre los elementos. Una EspecificacióndeProducto no representa

un Elemento, sino una descripción acerca de ellos. Aunque todos los elementos inventariados se vendan y se eliminen sus instancias correspondientes de software, se conserva la EspecificacióndeProducto.

La descripción o especificación de objetos se relacionan bastante con aquella que describen. En un modelo conceptual, se acostumbra estipular que una EspecificaciónX describe una X.

Observe el siguiente ejemplo:

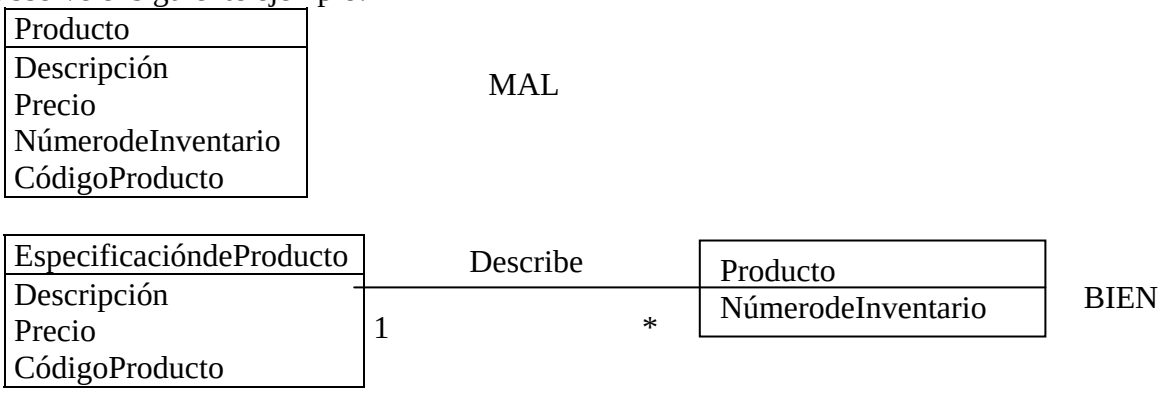


Figura 2.13 **Especificaciones de otras cosas. El signo "*" significa una multiplicidad de "muchos". Indica que una EspecificacióndeProducto puede describir muchos (*) Productos**

Como se puede observar, según la última figura, una EspecificacióndeProducto puede describir muchos Productos.

¿Cuándo se requiere especificar conceptos?

Incorpore una especificación o descripción de conceptos cuando:

La eliminación de las instancias de las cosas que describen Elemento, da por resultado una pérdida de la información que ha de conservarse, debido a la asociación incorrecta de la información con lo eliminado.

Reduce información redundante o duplicada.

11.10.10 Definición de términos en el lenguaje UML.

Con el término **concepto** se designarán cosas del mundo real.

La definición de **clase** en UML, es una "descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica".

En UML, una **operación** es un "servicio que puede solicitarse a un objeto para que realice un comportamiento" y **método** es la implementación de una operación que especifica el algoritmo o procedimiento de esta última.

La definición de **tipo** en UML se asemeja a la de clase, describe un conjunto de objetos parecidos con atributos y operaciones, pero no puede incluir métodos.

El término **interfaz** se define como un conjunto de operaciones visibles al exterior. En UML, puede estar asociada a tipos y clases. Aunque los conceptos reales pueden tener

una interfaz(la de un teléfono, por ejemplo), el término suele emplearse dentro del contexto de una interfaz para entidades de software.

11.10.11 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 9: Construcción de un modelo conceptual. Páginas 85-103. Explica el concepto de modelo conceptual y las estrategias para su elaboración.

11.11 CLASE 11: AGREGACION DE ASOCIACIONES Y ATRIBUTOS AL MODELO CONCEPTUAL

Asociaciones

Notación de las asociaciones en UML

Identificación de las asociaciones: lista de asociaciones comunes

Directrices de las asociaciones

Papeles

Asignación de nombre a las asociaciones

Asociaciones múltiples entre dos tipos

Asociaciones del dominio del problema

Modelo conceptual del problema

OBJETIVOS:

Identificar las asociaciones y papales en un modelo conceptual.

11.11.1 Asociaciones.

Es necesario identificar las asociaciones de los conceptos que se requieren para satisfacer los requerimientos de información de los casos de uso en cuestión y los que contribuyen a entender el modelo conceptual.

La **asociación** es una relación entre dos conceptos que indica alguna conexión significativa e interesante entre ellos.

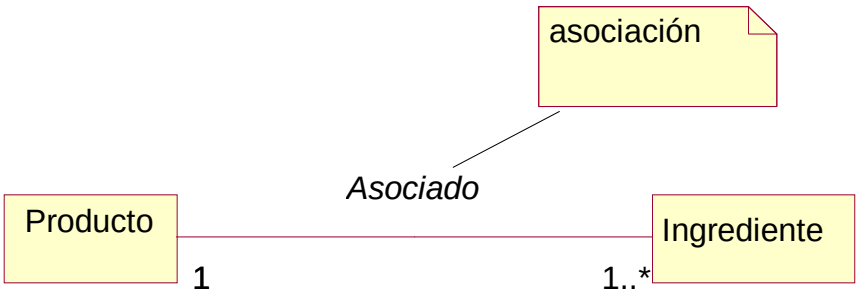


Figura 2.14 **Asociaciones**

Criterios de las asociaciones útiles:

En un modelo conceptual, se deben incluir las siguientes asociaciones:

Las asociaciones que vale la pena mencionar suelen incluir el conocimiento de una relación que ha de preservase durante algún tiempo: puede tratarse de milisegundos o años según el contexto. En otras palabras ¿entre que objetos hemos de tener algún recuerdo de una relación ?. Por ejemplo, en el caso de una cajera de un supermercado que vende productos ¿debemos recordar cuáles instancias de VentasLineasdeProductos están asociadas a la instancia Venta? Ya que de lo contrario no sería posible reconstruir la venta, imprimir el recibo ni calcular el total de la venta.

Examinemos la conveniencia de incluir las siguientes asociaciones en un modelo conceptual.

Las asociaciones en que el conocimiento de la relación ha de ser preservado durante algún tiempo (asociaciones que “deben conocerse”)

Las asociaciones provenientes de la Lista de asociaciones comunes.

11.11.2 Notación de las asociaciones en UML.

Una asociación se representa como una línea entre conceptos con el nombre de la asociación. Esta es intrínsecamente bidireccional, o sea, es posible un nexo lógico entre los objetos de un tipo y los del otro. Este vínculo es abstracto; no es una afirmación sobre las conexiones entre las entidades software.

Los extremos de una asociación pueden contener una expresión de multiplicidad que indique la relación numérica entre las instancias de los conceptos.

Una flecha opcional de la dirección de la lectura, indica la dirección en que debe leerse el nombre de la asociación. En su ausencia, por convención la asociación se lee de izquierda a derecha o de arriba hacia abajo, aunque el UML no hace de esto una regla.

11.11.3 Identificación de las asociaciones: lista de asociaciones comunes.

Agregue las asociaciones utilizando la lista que se presenta a continuación.

Categoría	Ejemplo
A es una parte física de B	Caja - Tienda Edificio – Universidad Ala avión
A es una parte lógica de B	VentasLineadeProducto-Venta TramodeVuelo-RutadeVuelo
A está físicamente contenido en B	Productos-Estantes Pasaje-Avión Estudiante – Edificio
A está contenido lógicamente en B	DescripcióndeProducto-Catalogo Vuelo-ProgramaVuelo Asignatura – Programa de asignatura
A es una descripción de B	DescripcióndeProducto-Producto DescripcióndeVuelo-Vuelo Objetivo de asignatura – Asignatura
A es un elemento de línea en una transacción o reporte B	VentaLilneaProducto-Venta TrabajodeMantenimiento-Mantenimiento NotadeEstudiante–Reportede Estudiante
A se conoce/introduce/registra/presenta/captura en B	Reservación-ListadePasajero Matrícula – ListadeEstudiantes
A es miembro de B	Cajero-Tienda Piloto-Avión Profesor – Departamento
A es una subunidad organizacional de B	Departamento-Tienda Mantenimiento-LineaAerea Secretaría – Departamento
A usa o dirige a B	Pioto-Avión
A se comunica con B	Alumno-Cajero AgentedeReservaciones-Pasajero Profesor – Estudiante
A se relaciona con una transacción B	Pago-Venta, Pago-Matricula Cajero-Boleto Estudiante – Inscripción de curso
A es una transacción relacionada con otra transacción de B	Pago-Venta Reservación-Cancelación Inscripción – Cancelación
A está contiguo a B	Ciudad-Ciudad
A es propiedad de B	Avión-LineaAerea Auditorio – Universidad

11.11.4 Asociaciones de alta prioridad:

Las siguientes, son algunas categorías de alta prioridad que siempre conviene incluir en un modelo conceptual:

A es una parte física o lógica de B.

A está física o lógicamente contenido en B.

A está registrado en B.

11.11.5 Directrices de las asociaciones.

Es necesario identificar las asociaciones de los conceptos que se requieren para satisfacer los requerimientos de información de los casos de uso en cuestión y los que contribuyen a entender el modelo conceptual. Para identificar dichas asociaciones se deben tomar en cuenta las siguientes directrices:

Concentrarse en las asociaciones en que el conocimiento de la relación ha de preservarse durante algún tiempo.

Es más importante identificar los conceptos que las asociaciones.

Muchas asociaciones tienden a confundir el modelo conceptual en vez de aclararlo. A veces se requiere mucho tiempo para descubrirlas, y los beneficios son escasos.

No incluir las asociaciones redundantes ni las derivables.

11.11.6 Papeles.

A los extremos de una asociación se les llama **papeles**. Estos pueden tener:

- Nombre
- Expresión de multiplicidad
- Navegabilidad

11.11.7 Multiplicidad

La **multiplicidad** define cuántas instancias de un tipo A pueden asociarse a una instancia del tipo B en determinado momento. En el UML, el valor de multiplicidad depende del contexto. Por ejemplo:

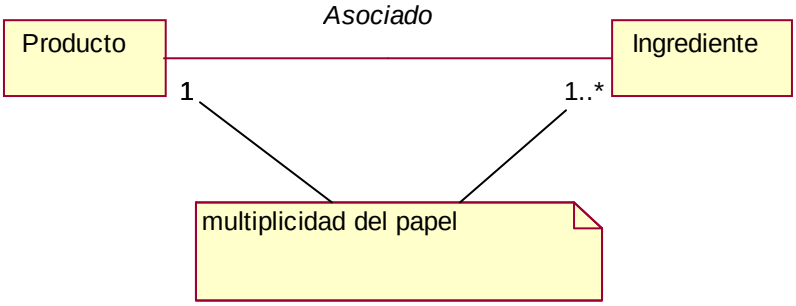


Figura 2.15 **Multiplicidad de una asociación**

A continuación se presentan algunos ejemplos de las expresiones de multiplicidad:

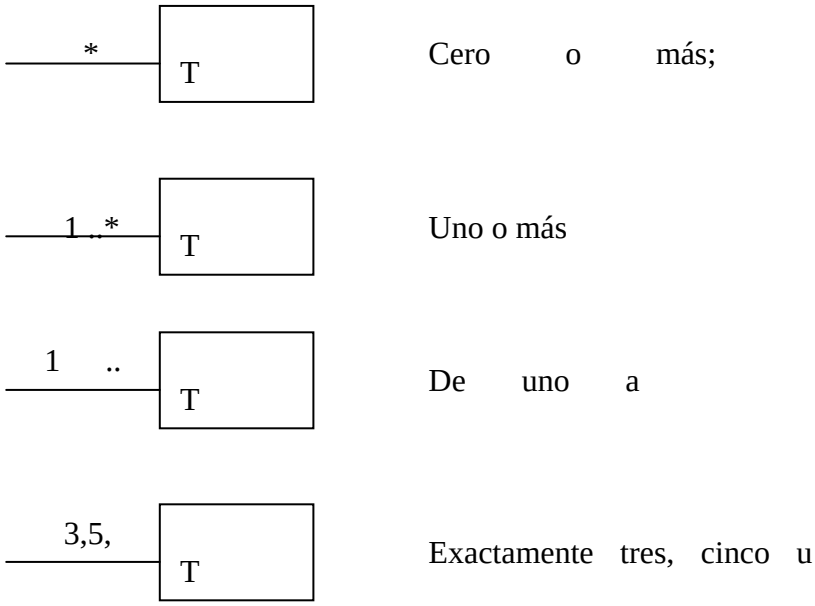


Figura 2.16 **Valores de la multiplicidad**

11.11.8 Asignación de nombre a las asociaciones.

Se asigna nombre a una asociación basándose en el formato NombredeTipo-FraseNominal-NombredeTipo, donde la frase nominal genera una secuencia que es legible y significativa dentro del contexto del modelo. Los nombres de las asociaciones comienzan con una mayúscula. Una frase nominal puede construirse con guiones.

11.11.9 Asociaciones múltiples entre dos tipos.

Dos tipos pueden tener varias asociaciones entre ellos. No hay un ejemplo sobresaliente en el problema que se construye, pero ilustraremos el caso con un ejemplo

diferente. Las asociaciones *Miembro_de* y *Presidido_por* entre Profesor y Departamento, son relaciones distintas que han de mostrarse por separado.

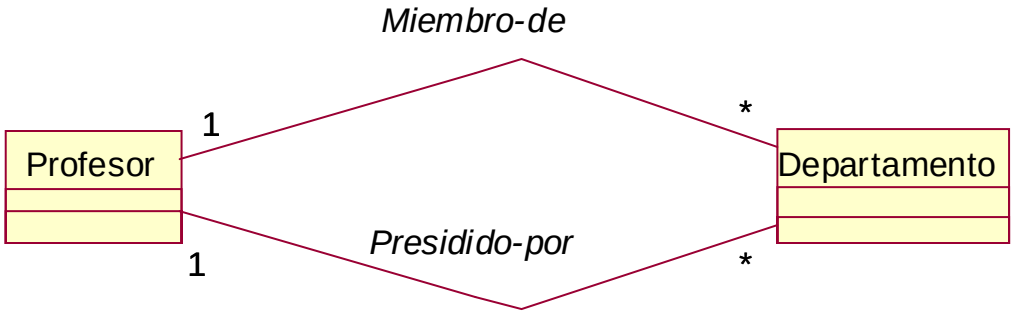


Figura 2.17 **Asociaciones múltiples**

11.11.10 Asociaciones del dominio del problema.

Se deben incorporar las asociaciones que indican los requerimientos(los casos de uso, por ejemplo), las que se necesitan recordar o que de alguna otra manera nos sugiere nuestra percepción del dominio del problema. Cuando encontremos un nuevo problema, hay que repasar y estudiar las categorías comunes de las asociaciones, ya que representan muchas asociaciones que generalmente es preciso registrar.

11.11.11 Modelo conceptual del problema.

En la construcción del modelo conceptual, es importante tener en cuenta lo siguiente:

Enfatice las asociaciones que deben conocerse, pero incorpore también las opcionales que se requieren sólo para la comprensión, con el fin de enriquecer el conocimiento básico del dominio.

11.11.12 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulos 10 y 11: Modelo conceptual, agregación de las asociaciones y Modelo conceptual, agregación de los atributos. Páginas 105-129. Presenta los conceptos de asociaciones y atributos así como las estrategias para su correcta agregación en el modelo conceptual.

11.12 CLASE 12: AGREGACION DE ASOCIACIONES Y ATRIBUTOS AL MODELO CONCEPTUAL

Atributos

Notación de los atributos en el UML

Tipos de atributos válidos

Atributos en el modelo del ejemplo

Modelo conceptual(con atributos y asociaciones)

OBJETIVOS:

Identificar las asociaciones de un modelo conceptual.

Identificar los atributos en un modelo conceptual.

11.12.1 Agregacion de los atributos al modelo conceptual

Atributos.

Un **atributo** es un valor lógico de un dato de un objeto. Incluya los siguientes atributos en un modelo conceptual:

Aquellos en que los requerimientos(por ejemplo, los casos de uso) indican o conllevan la necesidad de recordar información.

11.12.2 Modelo conceptual

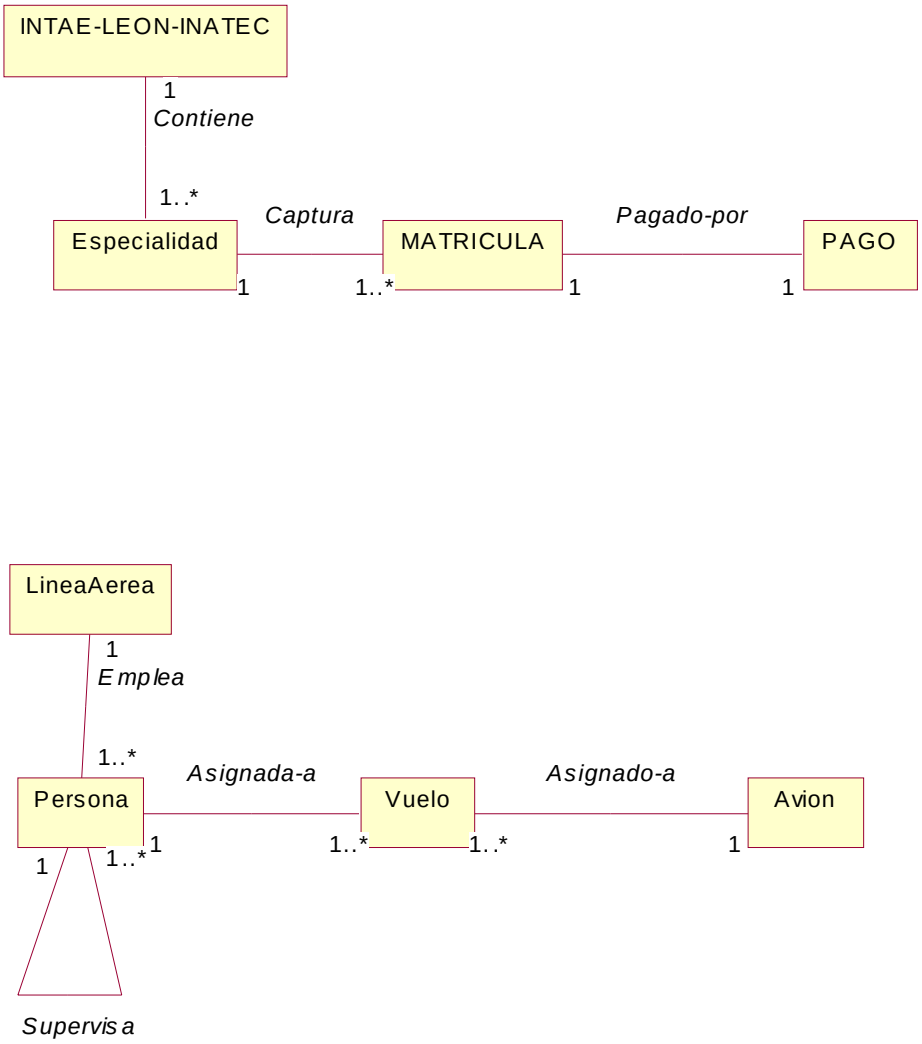
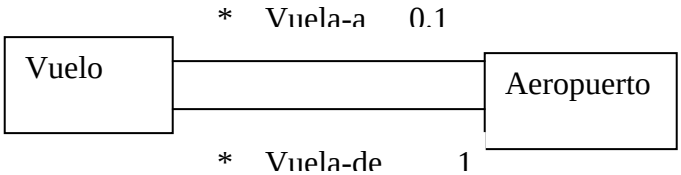


Figura 2.18 **Nombres de las asociaciones**

11.12.3 Asociaciones múltiples entre dos tipos

Dos tipos pueden tener varias asociaciones entre ellos; esto sucede con frecuencia. No hay un ejemplo sobresaliente en nuestro sistema INATE-LEON-INATEC, pero en el dominio de línea aérea encontramos uno en las relaciones entre Vuelo y Aeropuerto las asociaciones volar-hacia y volar-de son relaciones netamente diferentes que han de mostrarse por separado. Adviértase asimismo que ¡no se garantiza que todos los vuelos aterricen en un aeropuerto!



Aplicación de categoría de lista de comprobación de las asociaciones

Recorreremos la lista de comprobaciones, basándonos en los tipos anteriores identificados y teniendo presente los requerimientos actuales del caso de uso.

Categoría	Ejemplo
A es una parte física de B	No aplica
A está físicamente contenido en B	Registro-INTAE Especialidad – INTAE
A está contenido lógicamente en B	EspecificacióndeEspecialidades-CatalogoEspecialidades CatalogodeEspecialidades-INTAE
A es una descripción de B	EspecificacióndeEspecialidades-Especialidades Nota esto porque un alumno puede inscribir mas de una especialidad en diferente turno
A es un elemento de línea en una transacción o reporte B	Curso-Especialidad
A se conoce/introduce/registra/presen-ta/captura en B	Matrícula(Terminada)–INTAE Matricula(Actual)-Registro
A es miembro de B	Cajero-INTAE
A es una subunidad organizacional de B	Profesor – Departamento
A usa o dirige a B	Cajero-INTAE Registro-INTAE
A se comunica con B	Alumno-Cajero Profesor – Estudiante
A se relaciona con una transacción B	Alumno-Pago Cajero-Pago Estudiante – Inscripción de curso
A es una transacción relacionada con otra transacción de B	Pago-Matricula
A es propiedad de B	Registro-INTAE

El conjunto de las asociaciones que se incluyen en el modelo conceptual se obtuvo de manera bastante mecánica a partir de lista de comprobaciones. Pero tal vez hay que ser mas selectivo con las asociaciones que contendrá muestro modelo conceptual. Desde el punto de vista de la comunicación, no conviene saturar el modelo con asociaciones que no sean indispensables y que no mejoren nuestro conocimiento. El exceso de ellas no aclara sino que oscurece la situación.

Recordando los criterios para presentar las asociaciones:

Concentrarse en las asociaciones en que el conocimiento de la relación ha de ser durante algún tiempo (asociaciones "que deben conocerse").

No incluir las asociaciones redundantes ni las derivadas.

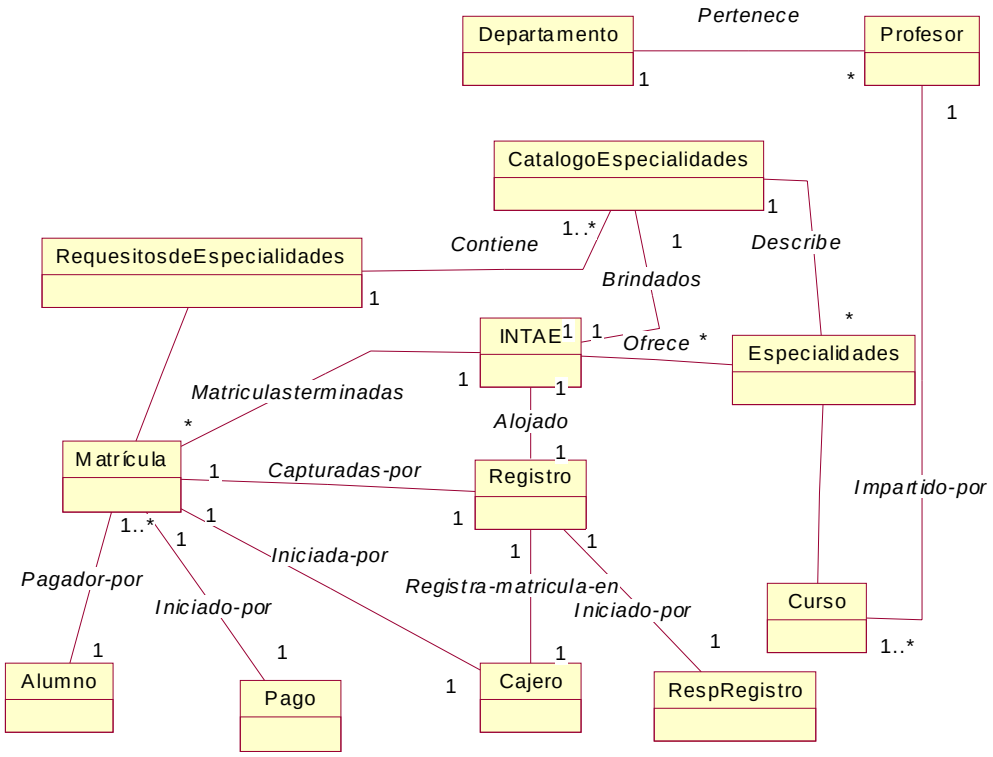


Figura 2.19 **Modelo Conceptual aplicado al INTAE – LEON-INATEC**

Conforme a las recomendaciones, no son indispensable todas las asociaciones que se muestran en cierto momento. Reflexione con cuidado la tabla anexa

Asociaciones	Explicación
Matricula capturada por cajero	Los requerimientos no indican la necesidad de conocer ni de registrar al cajero actual. Además, es derivada si existe la asociación Registro Usado-por Cajero
Registro	Los requerimientos no indican la necesidad de conocer ni de registrar al cajero actual

11.12.4 *Notación de los atributos en el UML.*

Los atributos se muestran en la segunda sección de la sección de conceptos. Es opcional indicar su tipo.

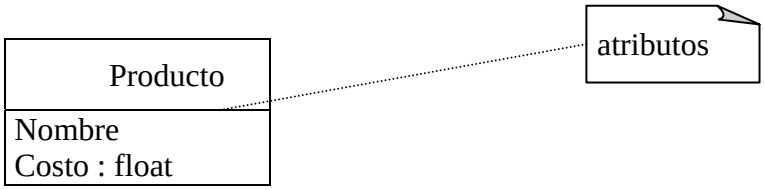


Figura 2.20 **Concepto y atributos**

11.12.5 Tipos de atributos válidos.

Hay algunas cosas que no deberían representarse como atributos, sino más bien como asociaciones. El tipo de un atributo no debe ser un concepto complejo del dominio. En un modelo conceptual, es preferible que los atributos sean **atributos simples** o **valores puros de datos.** Entre los tipos comunes de atributos simples más frecuentes se cuentan: Booleano, Fecha, Número, Cadena(Texto), Hora. Otros tipos comunes son: Dirección, Color, Número telefónico, Número del seguro social, Códigos postales, tipos enumerados.

Relaciones con asociaciones, no con atributos:

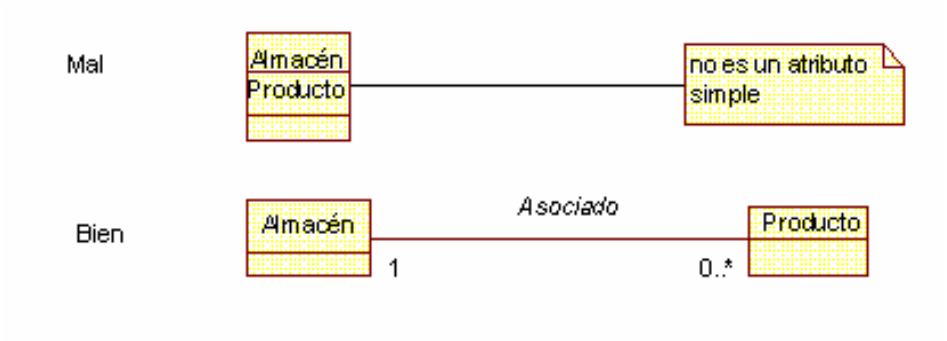


Figura 2.21 **Relacione con asociaciones, no con atributos**

No represente como atributos los conceptos complejos de dominio; use asociaciones

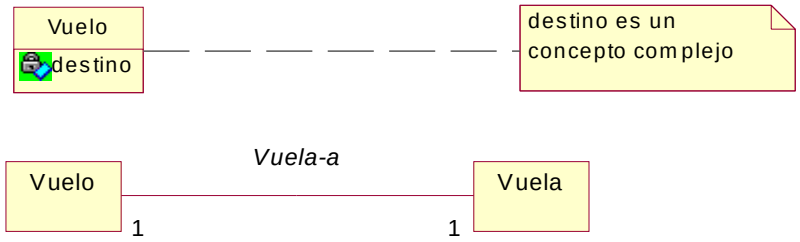


Figura 2.22 **Representación incorrecta de atributos**

11.12.6 Deterioro del diseño: ningún atributo debe incluirse como llave foránea:

Los atributos no deberían servir para relacionar conceptos en el modelo conceptual. Sin embargo, esta regla se viola con mucha frecuencia, ya que se agrega un tipo de atributo de llave foránea, lo cual suele hacerse con los diseños de bases de datos relacionales, a fin de asociar dos tipos. Por tanto, en un modelo conceptual, relacione los tipos a través de una asociación y no con un atributo.

Hay muchas formas de relacionar los objetos, las llaves foráneas son una de tantas; para evitar el deterioro del diseño, deberíamos posponer hasta la fase del diseño cómo vamos a implementar la relación por ejemplo en el caso que un cajero puede usar muchas cajas registradoras ejemplo en un supermercado.

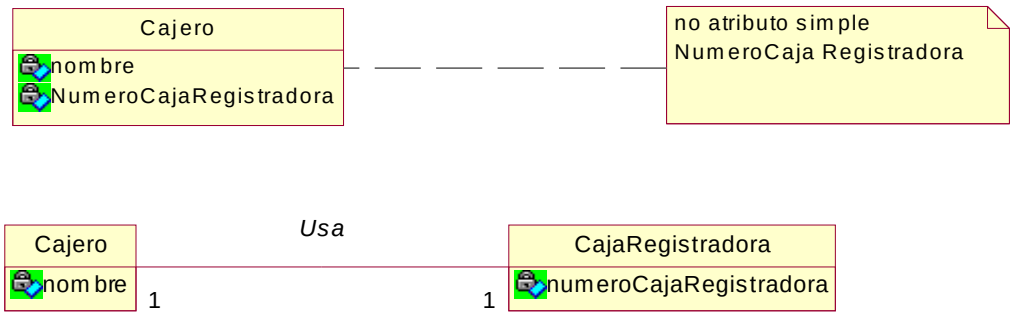


Figura 2.23 **No utilice atributos como claves extrañas**

11.12.7 Atributos en el modelo del ejemplo.

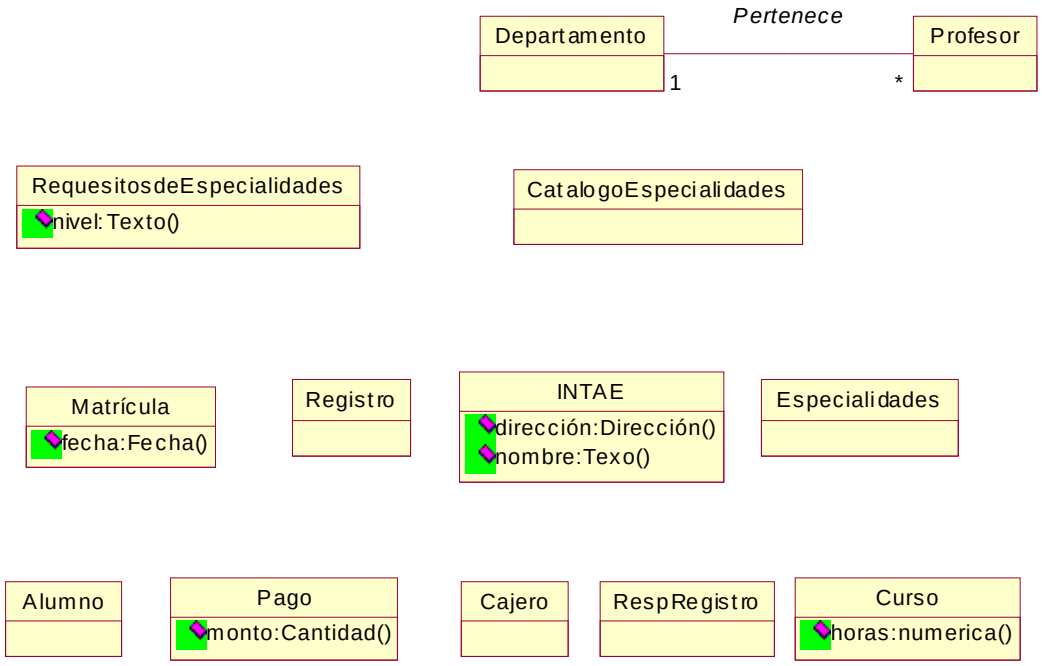


Figura 2.24 **Modelo conceptual que muestra los atributos**

11.12.8 Explicación de los atributos del problema:

RequisitosdeEspecialidad nivel: un identificador que determina el requisito mínimo para optar a un especialidad

Matricula fecha: el recibo es un informe escrito de una matricula, Normalmente contiene la fecha de la matricula

INTAE: dirección, nombre: el recibo requiere el nombre y la dirección de el INTAE

Pago monto: hay que capturar el importe (llamado también importe ofrecido) para determinar si se dio un pago suficiente y calcular el cambio.

Curso horas: indica el numero de horas que tiene el curso

11.12.9 Modelo conceptual (con atributos y asociaciones).

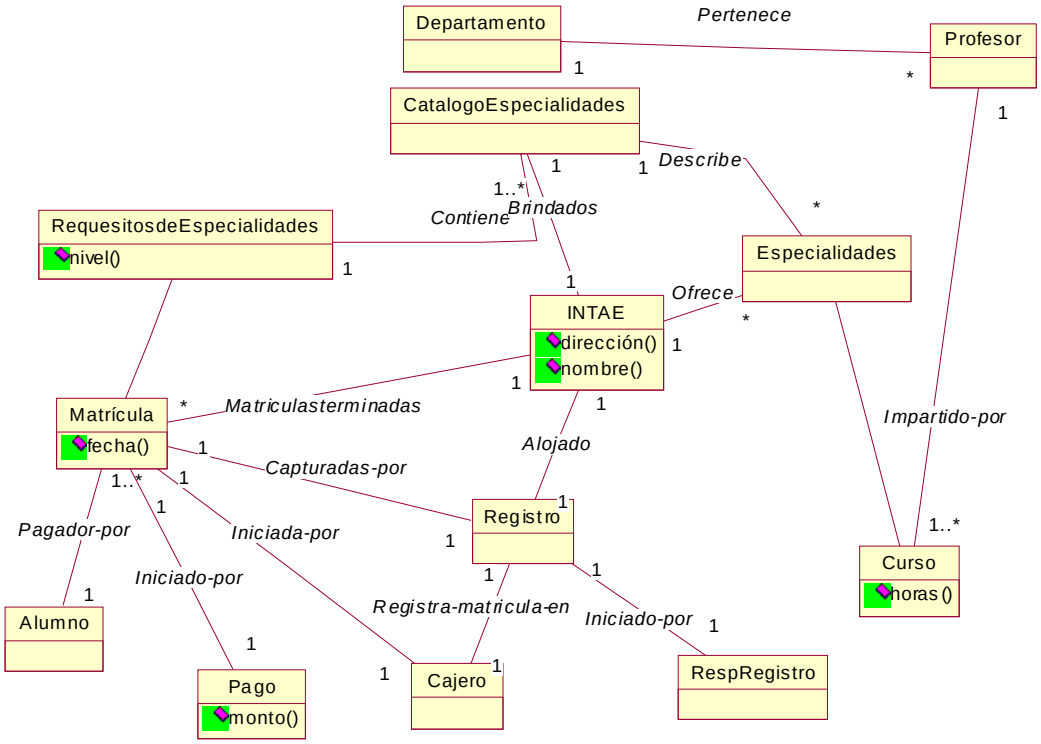


Figura 2.25 **Modelo conceptual de la máquina de café**

11.12.10 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulos 10 y 11: Modelo conceptual, agregación de las asociaciones y Modelo conceptual, agregación de los atributos. Páginas 105-129. Presenta los conceptos de asociaciones y atributos así como las estrategias para su correcta agregación en el modelo conceptual.

11.13 CLASE 13: DIAGRAMAS DE SECUENCIA DEL SISTEMA

Introducción

Diagramas de secuencia del sistema

Ejemplo de un diagrama de la secuencia de un sistema

Eventos y operaciones de un sistema

Cómo elaborar un diagrama de secuencia de un sistema

Eventos y fronteras de un sistema

Presentación del texto del caso de uso

OBJETIVOS:

Identificar los eventos y las operaciones del sistema.

Crear diagrama de la secuencia del sistema para los casos de uso.

11.13.1 Introducción.

La creación del diagrama de secuencias de un sistema depende de la formulación previa de los casos de uso.

Antes de iniciar el diseño de cómo funcionará una aplicación de software, es necesario investigar y definir su comportamiento. El **comportamiento** del sistema *es una descripción de lo **que** hace*, sin explicar la manera en que lo hace. Una parte de la descripción es un diagrama de la secuencia del sistema.

11.13.2 Diagramas de secuencia del sistema.

Los casos de uso indican cómo los actores interactúan con el sistema de software. Durante la interacción, un actor genera eventos dirigidos a un sistema, solicitando alguna operación a cambio. Con el evento de esa petición se inicia una operación del sistema. Por ejemplo, cuando un solicitante de pedido introduce un nuevo pedido, está pidiendo al sistema que registre dicho pedido.

UML incluye en su notación los diagramas de secuencia que dan una descripción gráfica de las interacciones del actor y de las operaciones a que da origen.

El **diagrama de secuencia de un sistema** es una representación que muestra, en determinado escenario de un caso de uso4, los eventos generados por actores externos, su

4 El escenario de un caso de uso es un ejemplo real de la ejecución del caso de uso.

orden y los eventos internos del sistema. Los diagramas se centran en los eventos que trascienden las fronteras del sistema y que fluyen de los actores a los sistemas.

11.13.3 Ejemplo de un diagrama de la secuencia de un sistema.

El diagrama de secuencia de un sistema describe, en el curso particular de los eventos de un caso de uso, los actores externos que interactúan directamente con el sistema y con los eventos del sistema generados por los actores. En el diagrama, el tiempo avanza hacia abajo y el orden de los eventos debe seguir el orden indicado en el caso de uso. Los eventos pueden tener parámetros.

El ejemplo presentado a continuación corresponde al curso normal de los eventos en el caso Registrar Notas, donde el responsable de registro es el único actor y genera los eventos nuevos.

Registrar Notas

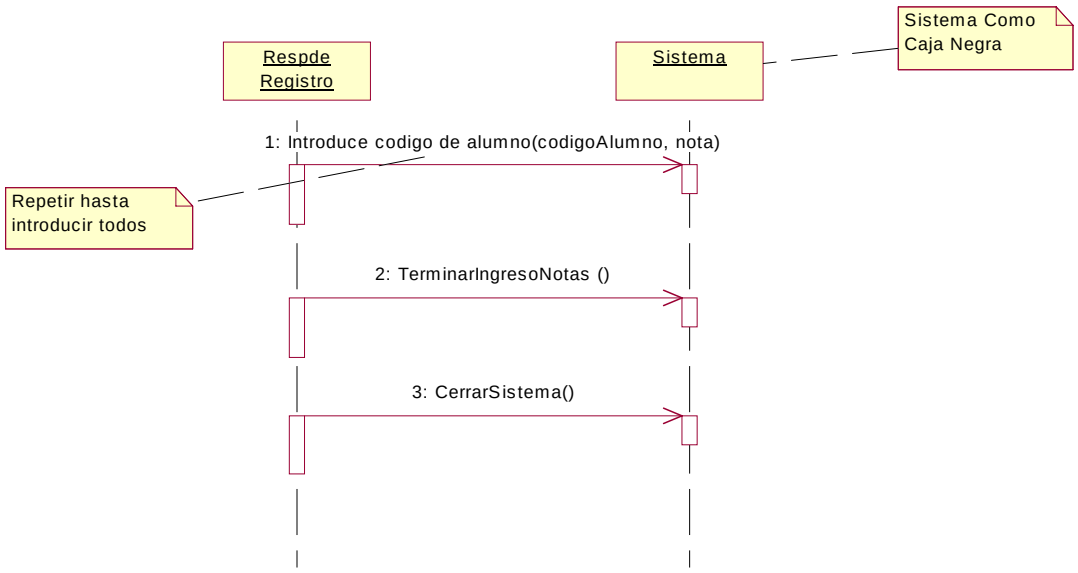


Figura 2.26 **Diagrama de la secuencia del sistema para el caso de uso Registrar Notas**

11.13.4 Eventos y operaciones de un sistema.

El **evento de un sistema** es un hecho externo de entrada que un actor produce en un sistema. El evento da origen a una operación de respuesta. La **operación de un sistema** es una acción que éste ejecuta en respuesta a un evento del sistema. Por ejemplo cuando el responsable de registro genera el evento *IntroducirNotas*, causa la ejecución de la operación *IntroducirNotas;* el nombre del evento y de la operación son idénticos; la diferencia reside en que el evento es el estimulo nombrado y la operación es la respuesta.

11.13.5 Registro de las operaciones de un sistema:

Para determinar el conjunto de las operaciones requeridas del sistema se identifican sus eventos. Cuando se usan parámetros, las operaciones son las siguientes:

IntroducirNota(CodigoAlumno,Nota)

TerminarIngresoNotas()

Para registrar las operaciones de un tipo, UML ofrece una notación como la que se muestra a continuación:

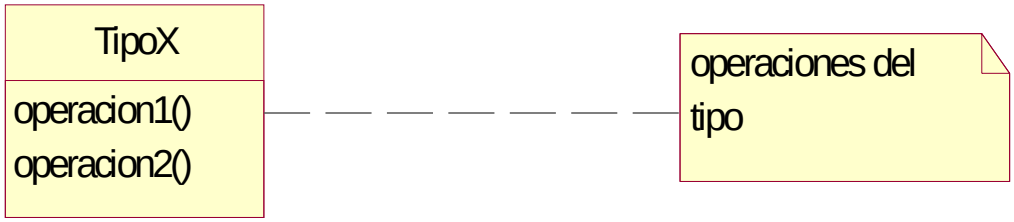


Figura 2.27 **Notación de la operaciones en UML**

Con esta notación, las operaciones del sistema pueden agruparse como operaciones del tipo Sistema (como en la figura siguiente). Los parámetros de las operaciones son opcionales: pueden incluirse o no.

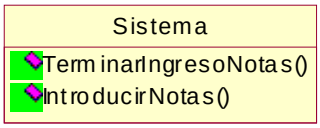


Figura 2.28 **Operaciones de un sistema registradas en el tipo Sistema**

Esta representación es muy diferente a la del modelo conceptual, los elementos de éste representan conceptos del mundo real, mientras que el tipo Sistema es un concepto artificial, que muestra además las operaciones. El modelo conceptual presenta información estática, mientras que aquí se describe el comportamiento del sistema, que es información dinámia.

11.13.6 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 13: Comportamiento de los sistemas, diagrama de la secuencia del sistema. Páginas 135-144. Aborda el concepto de diagrama de secuencia de un sistema y muestra su importancia mediante un ejemplo.

11.14 CLASE 14: DIAGRAMAS DE SECUENCIA DEL SISTEMA

Introducción

Diagramas de secuencia del sistema

Ejemplo de un diagrama de la secuencia de un sistema

Eventos y operaciones de un sistema

Cómo elaborar un diagrama de secuencia de un sistema

Eventos y fronteras de un sistema

Presentación del texto del caso de uso

OBJETIVOS:

Identificar los eventos y las operaciones del sistema.

Crear el diagrama de la secuencia del sistema para los casos de uso.

11.14.1 Cómo elaborar un diagrama de secuencia de un sistema.

Para elaborar diagramas de secuencia de un sistema que describan el curso normal de los eventos en un caso de uso:

Trace una línea que represente el sistema.

Identifique los actores que operan directamente sobre el sistema. Trace una línea para cada uno de ellos.

A partir del curso normal de los eventos del caso de uso identifique los eventos externos del sistema que son generados por los actores. Muéstrelos gráficamente en el diagrama.

A la izquierda del diagrama, puede incluir o no el texto del caso de uso.

La identificación y el orden de los eventos de un sistema para incluirlos en los diagramas de secuencia, se deducen de los casos de uso.

11.14.2 Eventos y fronteras de un sistema.

Para identificar los eventos de un sistema es necesario tener una idea clara de lo que se quiere al escoger su frontera. En lo que respecta al desarrollo del software, la frontera de

un sistema suele seleccionarse para que sea el sistema de software, por lo tanto, un evento del sistema es un hecho externo que estimula directamente al software.

Con el objetivo de identificar los eventos del sistema, parta del caso de uso correspondiente, y determine los actores que interactúan directamente con el sistema de software. Los nombres de los eventos(y sus operaciones asociadas) deben comenzar con un verbo, porque recalca que los eventos están orientados a comandos.

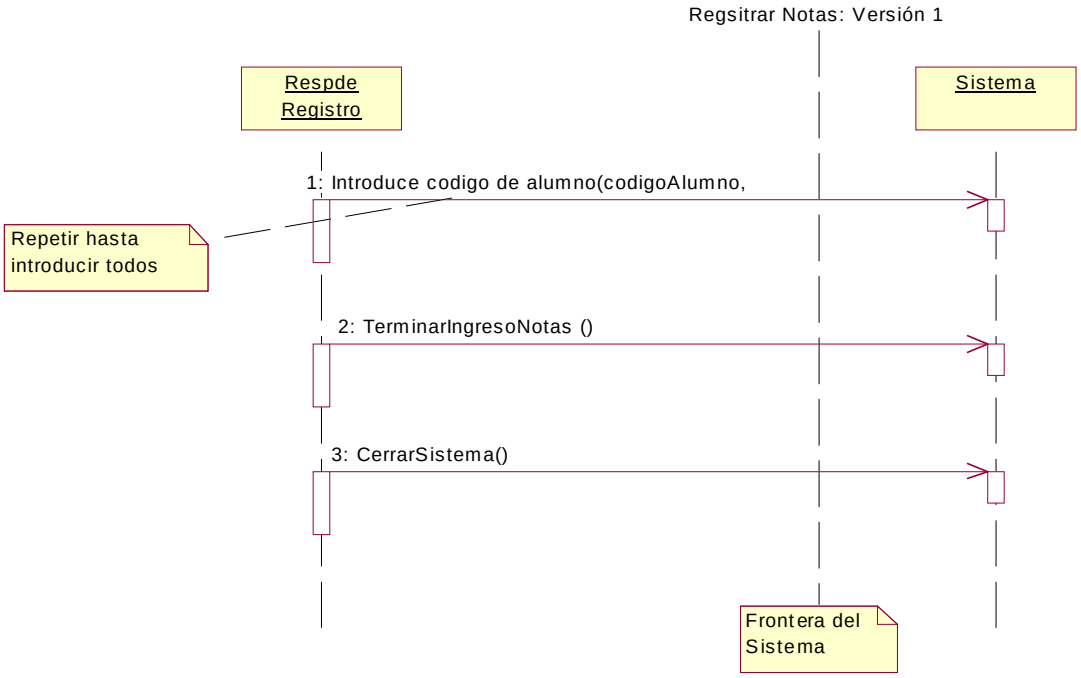


Figura 2.29 **Definición de la frontera del sistema**

11.14.3 Presentación del texto del caso de uso.

Algunas veces es aconsejable mostrar al menos una parte del texto del caso de uso dentro del diagrama de secuencia(a la izquierda de éste), para describir gráficamente su estrecha relación con el caso de uso.

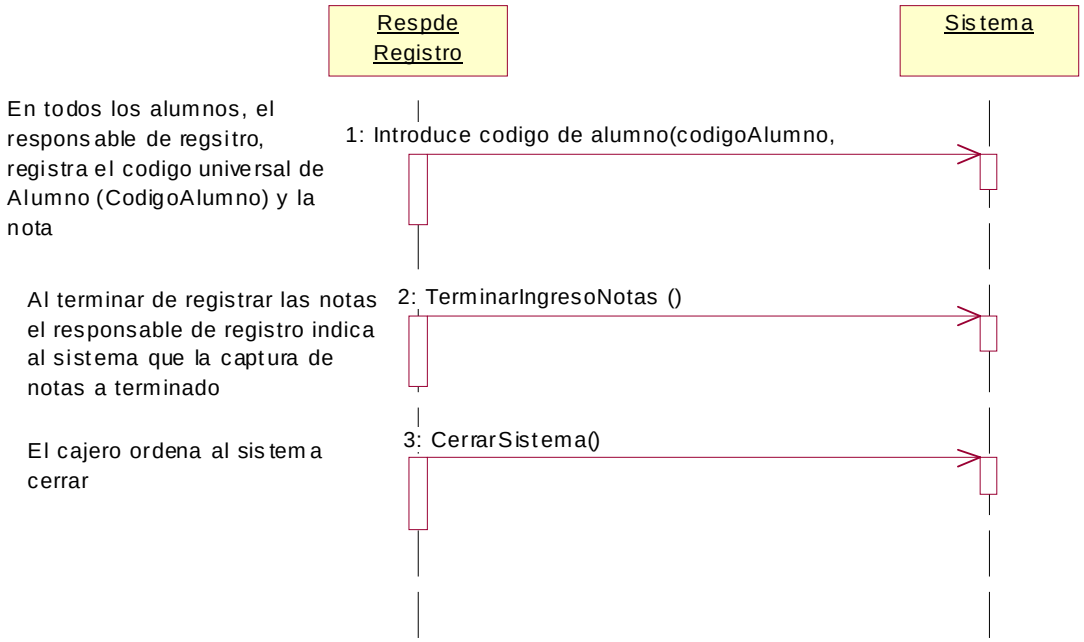


Figura 2.30 **Deducción de los diagramas de secuencia de un sistema de los casos de uso**

11.14.4 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 13: Comportamiento de los sistemas, diagrama de la secuencia del sistema. Páginas 135-144. Aborda el concepto de diagrama de secuencia de un sistema y muestra su importancia mediante un ejemplo.

11.15 CLASE 15: CONTRATOS

Introducción

Contratos

Secciones del contrato

Cómo preparar un contrato

Poscondiciones

Descripción de los detalles y algoritmos del diseño: notas

Precondiciones

Recomendación sobre cómo redactar contratos

OBJETIVOS:

Definir el concepto de Contrato

Identificar las partes que se compone un contrato

Identificar los pasos para preparar un contrato

Definir el concepto de Poscondición

Describir los detalle y algoritmo del diseño

Definir las precondiciones

Identificar las recomendaciones sobre la redacción de contratos

Los contratos describen el efecto que tienen las operaciones sobre el sistema. El lenguaje UML ofrece un soporte para definir los contratos, ya que permite definir las precondiciones y las poscondiciones de las operaciones[*].

Antes de empezar el diseño lógico de cómo funcionará una aplicación de software, es necesario investigar y definir su comportamiento como "caja negra". El comportamiento de un sistema es una descripción de lo que hace, sin explicar cómo lo hace. Los contratos son documentos muy útiles que describen el comportamiento de un sistema a partir de cómo cambia el estado de un sistema cuando se llama una operación suya.

11.15.2 Contratos.

Un **contrato** es un documento que describe lo que una operación se propone lograr. Suele redactarse en un estilo declarativo, enfatizando lo ***que sucederá y no cómo*** se conseguirá. Los contratos suelen expresarse a partir de los cambios de estado de las

[*] En la definición formal del UML, las operaciones tienen un conjunto de propiedades predefinidas que incluye sus precondiciones y poscondiciones.

precondiciones y de las poscondiciones. Puede elaborarse un contrato para un método de una clase de software o para una operación más global del sistema. Para cada operación del sistema se redactan contratos, con el propósito de describir su comportamiento.

Ejemplo de Contrato: RealizarMatricula

En el siguiente ejemplo se describe un contrato de la operación RealizarMatricula del Sistema

11.15.3 Diseño por Contrato

El Diseño por contrato es una técnica diseñada por Bertrand Meyer. Esta técnica es una característica central del lenguaje Eiffel, que el desarrolló. Sin embargo, el Diseño por contrato no es específico de Eiffel; es una técnica valiosa que se puede usar con cualquier lenguaje de programación.

En el corazón del Diseño por contrato se encuentra la afirmación. Una afirmación es un enunciado Booleano que nunca debe ser falso y, por tanto, sólo lo será debido a una falla. Por lo común, las afirmaciones se comprueban sólo durante la depuración y no durante la ejecución en producción. De hecho, un programa nunca debe suponer que se están comprobando las afirmaciones.

El diseño por contrato se vale de tres tipos de afirmaciones: poscondiciones, precondiciones e invariantes.

Las precondiciones y las poscondiciones se aplican a las operaciones. Una poscondición es un enunciado sobre cómo debería verse el mundo después de la ejecución de un operación. Por ejemplo, si definimos la operación “cuadrado” sobre un número, la poscondición adoptaría la forma de resultado = éste * éste, donde resultado es el producto y éste es el objeto sobre el cual se invocó la operación. Las poscondición es un forma útil de decir qué es lo que hacemos, sin decir cómo la hacemos, en otras palabras, de separar la interfaz de la implementación.

La precondición es un enunciado de cómo esperamos encontrar el mundo, antes de ejecutar una operación. Podemos definir una precondición para la operación “cuadrado” de éste >= 0. tal precondición dice que es un error invocar a “cuadrado” sobre un numero negativo y que las consecuencias de hacerlo son indefinidas.

A primera vista esto no parece ser una buena idea, pues deberíamos poner una comprobación en alguna parte para garantizar que se invoque correctamente a “cuadrado”. La cuestión importante es quién es responsable de hacerlo.

La precondición hace explicito que quien invoca es el responsable de la comprobación. Sin este enunciado explícito de responsabilidad, podemos tener muy poca comprobación (pues ambas partes suponen que la otra es la responsable) o tenerla en demasía (cuando ambas partes lo hacen). Demasiada comprobación es mala, ya que provoca que se duplique muchas veces el código de comprobación, lo cual puede complicar de manera importante un programa. El ser explícito sobre quien es el responsable contribuye a reducir esta complejidad. Se reduce el peligro de que el invocador, comprobar, por el hecho de que, por lo general, las afirmaciones se comprueban durante las pruebas y la depuración.

A partir de estas definiciones de precondición y poscondición, podemos ver una definición sólida del término excepción, que ocurre cuando se invoca una operación estando satisfecha su precondición y, sin embargo, no puede regresar con su poscondición satisfecha.

Una invariante es una afirmación acerca de una clase. Por ejemplo la clase Cuenta puede tener invariante que diga que balance = = sum(entradas.cantidad()). La invariante "siempre" es verdadera para todas las instancias de la clase. En este contexto, "siempre" significa "siempre que el objeto esté disponible para que se invoque una operación sobre ella".

En esencia, lo anterior significa que la invariante se agrega a las precondiciones y poscondiciones asociadas a todas las operaciones públicas de la clase dada. La invariante puede volverse falsa durante la ejecución de un método, pero debe haberse restablecido a verdadera para el momento en que cualquier otro objeto pueda hacerle algo al receptor.

Las afirmaciones pueden desempeñar un papel único en la subclasificación.

Uno de los peligros del polimorfismo es que se podrían redefinir las operaciones de una subclase, de tal modo que fueran inconsistentes con las operaciones de la superclase. Las afirmaciones impiden hacer esto. Las invariantes y poscondición de una clase deben aplicarse a todas las subclases. Las subclases pueden elegir el refuerzo de estas afirmaciones, pero no pueden debilitarlas. La precondición, por otra parte, no puede reforzarse, aunque sí debilitarse.

Esto parecerá extraño a primera vista, pero es importante para permitir vínculos dinámicos. Siempre debería ser posible tratar un objeto de subclase como si fuera una instancia de superclase (de acuerdo con el principio de la sustituibilidad). Si una subclase refuerza su precondición, entonces podría fallar una operación de superclase, cuando se aplicara a la subclase.

En esencia, las afirmaciones sólo pueden aumentar las responsabilidades de la subclase. Las precondiciones son un enunciado para trasladar una responsabilidad al invocador; se incrementan las responsabilidades de una clase cuando se debilita una precondición. En la práctica, todo esto permite un mucho mejor control de la subclasificación y ayuda a asegurar que las subclases se comporten de manera adecuada.

Idealmente, las afirmaciones se deben incluir en el código, como parte de la definición de la interfaz. Los compiladores deben poder encender la comprobación por afirmaciones durante las depuración y apagarla durante la operación en producción. Se pueden manejar varias etapas de comprobación de afirmaciones. Las precondiciones a menudo ofrecen las mejores posibilidades de atrapar errores con la menor cantidad de sobrecarga de proceso.

11.15.4 Cuándo utilizar el Diseño por contrato

El diseño por contrato es una técnica valiosa que siempre se debe emplear cuando se programa. Es particularmente útil para la construcción de interfaces claras.

El UML no habla mucho acerca de las afirmaciones, pero se pueden usar sin problema. Las invariantes son equivalentes a las reglas de condición en los diagramas de clase y se deberán usar tanto como sea posible. Las precondiciones y poscondiciones de operaciones se deben documentar con las definiciones de las operaciones.

Contrato

Nombre:	RealizarMatricula (Codigo_Alumno:texto,, , p: Producto):
Responsabilidades:	Capturar (registrar) la matricula de un alumno y agregar a la matricula. Desplegar los requisitos de la especialidad que desea estudiar.
Tipo:	Sistema.
Referencias Cruzadas:	Funciones del sistema: R1, R2, R3, R4, R5, R6, R10 Casos de uso: SolicitarMatricula
Notas:	Utilizar el acceso super rápido a la base de datos.
Excepciones:	Si la capacidad del centro esta saturada indicar que la matricula no se puede realizar Si los requisitos no son cumplidos emitir un mensaje que no puede ser matriculado en la especialidad que desea Salidas:
Precondiciones:	El sistema conoce la capacidad de las especialidades.

Poscondiciones:

- Si se trata de una nueva matricula se crea una matricula (Creación de instancia).
- Si se trata de una nueva matricula, la nueva matricula fue asociada a Registro (asociación formada).

-Se asoció una instancia RequisitosdeEspecialidad a la instancia CatalogodeEspecialidad, basado esto en la correspondencia del codigo (asociación formada)

11.15.5 Secciones del contrato.

A continuación presentamos la descripción de las secciones de un contrato se da en el siguiente esquema. No todas las secciones son necesarias; se recomienda las de Responsabilidades y Poscondiciones.

Contrato

Nombre:	Nombre de la operación y parámetros.
Responsabilidades:	Descripción informal de las responsabilidades que debe cumplir la operación.
Tipo:	Nombre del tipo(concepto, clase de software, interfaz)
Referencias Cruzadas:	Números de referencia de las funciones del sistema, casos de uso, etc.
Notas:	Notas de diseño, algoritmos e información afín.
Excepciones:	Casos excepcionales.
Salida:	No salidas de la Interfaz de Usuario; por ejemplo, mensajes o registros que se envían afuera del sistema.
Precondiciones:	Suposiciones acerca del estado del sistema antes de ejecutar la operación.
Poscondiciones:	El estado del sistema después de la operación.

11.15.6 Cómo preparar un contrato.

Aplique las siguientes sugerencias para elaborar contratos en los casos de uso:

1. Identifique las operaciones del sistema a partir de los diagramas de secuencia.
2. Elabore un contrato en cada operación del sistema.
3. Comience redactando la sección de *Responsabilidades*; después describa informalmente el propósito de la operación.
4. Complete la sección de Poscondiciones, describiendo en forma declarativa los cambios de estado de los objetos en el modelo conceptual.

Para describir las poscondiciones utilice las siguientes categorías:

- Creación y eliminación de las instancias.
- Modificación de los atributos.
- Asociaciones formadas y canceladas.

11.15.7 Poscondiciones.

La Poscondición en el ejemplo RealizarMatricula incluye una clasificación; por ejemplo creación de instancias o asociación formada. Después de la sección de *Responsabilidades*, la parte más importante del contrato son las poscondiciones, que estipulan cómo cambió el sistema tras esta operación. No son acciones que deben

efectuarse durante la operación; más bien son declaraciones sobre el estado del sistema que se aplican una vez concluida la operación.

UML no impone límites a la forma de expresar las poscondiciones, pero las siguientes categorías de cambio de estado han resultado de gran utilidad en la práctica:

Categorías útiles a las poscondiciones del contrato:

- Creación y eliminación de instancias.
- Modificación de los atributos.
- Asociaciones formadas y canceladas.

El UML no define cómo expresar las poscondiciones ; así que puede escoger el formato que más le agrade. Lo importante es que adopte una actitud declarativa, orientada al cambio de estado y no a la acción, porque las poscondiciones deberían ser declaraciones sobre el estado o resultado, no una declaración de acción a realizar.

Exprese las poscondiciones en tiempo pasado para enfatizar que se trata de declaraciones sobre un cambio de estado.

11.15.8 Las poscondiciones se relacionan con el modelo conceptual:

Las poscondiciones se expresan dentro del contexto del modelo conceptual. ¿Qué instancias es posible crear? La respuesta es: la provenientes de ese modelo. ¿Qué asociaciones es posible formar? La respuesta es: las que están en el modelo conceptual. Y así podríamos ir formulando y contestando más preguntas.

Cuando se realizan los contratos, el lector se percatará de la necesidad de registrar nuevos conceptos, atributos o asociaciones en el modelo conceptual. Mejore dicho modelo, conforme vaya haciendo más descubrimientos, mientras reflexiona sobre los contratos de las operaciones.

11.15.9 Descripción de los detalles y algoritmos del diseño: notas.

La sección del contrato correspondiente a Notas es el lugar donde pueden hacerse las declaraciones del diseño referentes a la operación. Por ejemplo, si se prefiere un algoritmo en particular para manejar la operación, esa sección es el sitio ideal para su documentación.

11.15.10 La ventaja de las poscondiciones

Expresado en una forma declarativa de cambios de estado, el contrato constituye excelente herramienta de investigación, pues permite describir los cambios necesarios para que el sistema funcione sin necesidad de describir cómo se logra. En otras palabras, podemos posponer el diseño y la solución del software y concentrarnos analíticamente en lo que debe suceder, no en la manera de conseguirlo.

11.15.11 El espíritu de las poscondiciones: el escenario y el telón.

Las poscondiciones debería describir el estado de un sistema, no las acciones a realizar. Exprésela en tiempo pasado para enfatizar que se trata de declaraciones sobre un cambio pretérito de estado. Por ejemplo:

Se creó la instancia matricula (mejor);

En lugar de

Crear una instancia matricula (pero).

Reflexione sobre las poscondiciones sirviéndose de la siguiente imagen:

Reflexione sobre las poscondiciones sirviéndose de la siguiente imagen:

El sistema y sus objetos se presentan en el escenario de un teatro.

1.- Tome una fotografía del escenario antes de la operación.

2.- Corra el telón del escenario y aplique la operación del sistema (ruido de fondo con sonidos metálicos, gritos, chillidos.....).

3.- Corra el telón y tome una segunda fotografía .

4.- Compare las fotografías de antes y después, y exprese como poscondición los cambios del estado del escenario (Se creó la instancia matricula)

11.15.12 Precondiciones.

Las precondiciones definen las suposiciones sobre el estado del sistema al iniciarse la operación. Hay muchas precondiciones que pueden declararse en una operación, pero las más relevantes son las siguientes:

- Cosas que son importantes probar en el software en algún momento de la ejecución de la operación.
- Cosas que no serán sometidas a prueba, pero de las cuales depende el éxito de la operación.

11.15.13 Recomendación sobre cómo redactar contratos.

- Una vez anotado el nombre de la operación, llene primero la sección de *Responsabilidades* y luego la de Poscondiciones, dejando al final la de Precondiciones. Son las secciones más importantes. Si no le es útil llenar una sección, no ocurre ningún problema.
- Haga uso de la sección de *Notas* para explicar los detalles del diseño, por ejemplo, los algoritmos y los pasos secuenciales.
- Use la sección de *Excepciones* para explicar la reacción ante situaciones raras o especiales.
- En las *poscondiciones*, utilice las siguientes categorías de cambios de estado:

- Creación y eliminación de instancias.
- Modificación de atributos.
- Asociaciones formadas y canceladas.

- Exprese las poscondiciones en forma pasiva declarativa(por ejemplo, fue registrado), para destacar la declaración de un cambio de estado en vez del diseño de cómo iba a obtenerse.
- No olvide establecer una memoria entre los objetos actuales y los de creación reciente, definiendo para ello la formación de una asociación.

El error más frecuente es olvidar incluir la *formación de asociaciones*, principalmente cuando se crean nuevas instancias y haber establecido asociaciones a varios objetos.

11.15.14 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 14: Comportamiento de los sistemas, contratos. Páginas 145-158. Explica el concepto de contrato y su importancia en las operaciones del sistema

UML Gota a Gota, Editorial Person, Capitulo 4: Diagrama de Clase Fundamentos Pagina 80 – 83.

11.16 CLASE 16: CONTRATOS

Ejemplos

OBJETIVOS:

Crear contratos para las operaciones de un sistema.

11.16.1 Contrato para el caso de uso Dar cambio

Contrato para la operación Dar cambio( )

Nombre: Dar cambio(cantidad : Entero)

Responsabilidades: deberá devolver el resto del dinero que sobra, restando la cantidad que el usuario ha ingresado y el precio del producto solicitado.

Tipo: Sistema.

Referencias cruzadas: Función del sistema: R4

Excepciones: Si la cantidad es menor que el precio del producto, deberá indicar mediante un mensaje.

Precondiciones: Debe haber monedas de distintas denominaciones en la máquina.

Poscondiciones:

-Disminuye la cantidad de monedas de la máquina: *DepositoMonedas.numMonedas*

- Captura monedas de otra denominación: *DepositoMonedasIguales.denominacion*

- Se recibe la nueva moneda introducida: *Maquina.RecibirMoneda()*

Se agrega la nueva moneda recibida: *DepositoMonedas.AgregarMoneda()*

Aumenta el valor recolectado de la máquina: Maquina.valorRecolectado

11.16.2 Contrato para el caso de uso efectuar Pago

Contrato

Nombre:	EfectuarPago (monto : Número o cantidad).
Responsabilidades:	Registrar el pago, calcular el saldo e imprimir el recibo..
Tipo:	Sistema.
Referencias cruzadas:	Función del sistema: R2.1 Caso de uso: Realizar Matricula
Excepciones:	Si la matricula no esta concluida, indicar que se cometió un error. Si el monto es menor que el pago de la matricula, indicar que se cometió un error.
Salida:	
Precondiciones:	
Poscondiciones:	

11.16.3 Contrato para el caso de uso Inicio

Contrato

Nombre:	Inicio ().
Responsabilidades:	Iniciar el sistema.
Tipo:	Sistema.
Referencias cruzadas:	
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Se creó la instancia Matricula, Registro, RequisitosdeEspecialidade y CatalogodeEspecialidad (Creación de Instancia)

Se asocio CatalogodeEspecialidad a RequisitosdeEspecialidad (asociación formada)

Se asocia registro a INTAE (asociación Formada)

Se asocia Matricula a RequisitosdeEspecialidad (asociación formada)

11.16.4 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 14: Comportamiento de los sistemas, contratos. Páginas 155-156. Explica el concepto de contrato y su importancia en las operaciones del sistema.

11.17 CLASE 17: DESCRIPCION DE LOS CASOS REALES DE USO

Casos reales de uso.

OBJETIVOS:

Crear casos reales de uso.

11.17.1 Casos reales de uso.

Un **caso real de uso** describe el diseño concreto del caso de uso a partir de una tecnología particular de entrada y salida, así como de su implementación global. Por ejemplo, si interviene una interfaz gráfica para el usuario, el caso de uso real incluirá diagramas de las ventanas en cuestión. No es necesario generarlos. Una alternativa podría consistir en que el diseñador realice secuencias de pantallas de la interfaz general para el usuario y que después fuera incorporando los detalles durante la implementación. Esta secuencia de pantallas son de mucha utilidad, si antes de la implementación los diseñadores o el cliente necesitan descripciones detalladas.

Caso de Uso: Solicitar Matricula.

Actores: Alumno, Cajera, Registro

Propósito: Capturar una matricular y su pago.

Resumen: Un alumno llega a la caja con intención de estudiar una carrera. El cajero recibe el pago y entrega la hoja de matricula. El alumno llena la hoja y de matricula . Si todo es correcto, el sistema se encarga de matricular al alumno, el alumno se marcha matriculado .

Tipo: Primario y esencial.

Referencias

Cruzadas: R1, R2, R3, R4, R5, R6, R10, R2.1

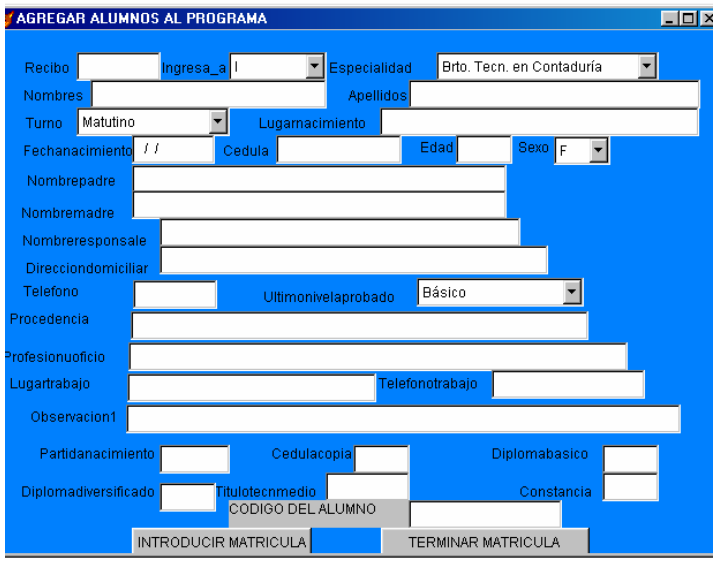


Figura 2.31 **Pantalla de agregar alumnos**

Curso normal de los eventos

Acción del actor	Respuesta del sistema
1. Este caso de uso comienza cuando un alumno llega con la intención de matricularse en una especialidad.	
	2. El sistema comprueba que existe disponibilidad de la carrera solicitada.
3. La Resp. De Reg. captura los datos del alumno	
	4. El sistema inicia el proceso de captura de los datos.
5. Al terminar la Captura de la matricula, el Resp de Reg. que termina la captura de datos	
	6. Calcula la disponibilidad del centro
7. La cajera recibe el dinero por la matricula	
8. El alumno se marcha con el recibo de matricula.	

Cursos alternos
Línea 2: No hay disponibilidad del centro para nueva matricula
Línea 8: El alumno no puede pagar. Cancela Matricula

11.17.2 Bibliografía COMENTADA:

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 16: Descripción de los casos reales de uso. Páginas 163-166. Presenta el concepto de caso real de uso y muestra un ejemplo.

11.18 CLASE 18: DIAGRAMAS DE COLABORACION

Introducción

Diagramas de interacción

Cómo preparar diagramas de colaboración

Relación de los diagramas de colaboración y otros elementos

Notación básica de los diagramas de colaboración

Representación gráfica de los mensajes

Representación gráfica de los parámetros

OBJETIVOS:

Definir los pasos para la elaboración de los diagramas de Interacción

Definir los pasos para la elaboración de los diagrama de colaboración

Identificar la relación de los diagrama de colaboración con otros elementos

Representar gráficamente los mensajes y los parámetros

UML contiene **diagramas de interacción** que explican gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas.

11.18.2 Para realizar un diagrama de interacción, es necesario lo siguiente:

- Un modelo conceptual: a partir de éste, se puede definir las clases del software correspondientes a los conceptos. Los objetos de las clases participan en las interacciones que se describen gráficamente en los diagramas.
- Contratos de operación del sistema: a partir de ellos se identifica las responsabilidades y las poscondiciones que han de llenar los diagramas de interacción.
- Casos de uso reales: a partir de estos, se reúne información sobre las tareas que realizan los diagramas de interacción, además de lo estipulado en los contratos.

11.18.3 Diagramas de interacción.

Un **diagrama de interacción** explica gráficamente las interacciones existentes entre las instancias(y las clases) del modelo. El punto de partida de las interacciones es el cumplimiento de las poscondiciones de los contratos de operación.

UML, define dos tipos de diagramas; ambos sirven para expresar interacciones semejantes o idénticas de mensaje:

1. Diagramas de colaboración.
2. Diagramas de secuencia.

Los **diagramas de colaboración** describen las interacciones entre los objetos en un formato de grafo o red.

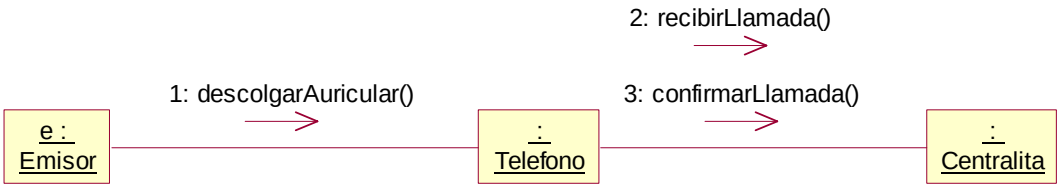


Figura 2.32 **Diagrama de colaboración**

Los **diagramas de secuencia** describen las interacciones en una especie de formato de cerca o muro:

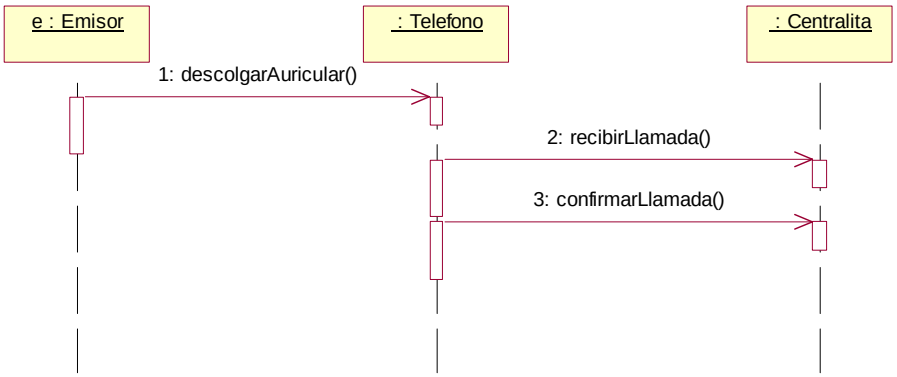


Figura 2.33 **Diagrama de secuencia**

11.18.4 Cómo preparar diagramas de colaboración.

Para preparar un diagrama de colaboración:

Elabore un diagrama por cada operación del sistema durante el ciclo actual de desarrollo.

En cada mensaje del sistema, dibuje un diagrama incluyéndolo como mensaje inicial.

Si el diagrama se torna complejo(por ejemplo, no cabe en una hoja de papel tamaño carta), divídalo en diagramas más pequeños.

Diseñe un sistema de objetos interactivos que realicen las tareas, usando como punto de partida las responsabilidades del contrato de operación, las poscondiciones y la descripción de casos de uso.

11.18.5 Relación de los diagramas de colaboración y otros elementos.

Debemos observar la relación existente entre los distintos elementos de UML, vistos hasta ahora:

- Los casos de uso indican los eventos del sistema que se muestran explícitamente en los diagramas de secuencia.
- En los contratos, se describen las operaciones del sistema.
- Las operaciones del sistema representan mensajes y éstos originan diagramas que explican gráficamente cómo los objetos interactúan para llevar a cabo las funciones requeridas.

11.18.6 Notación básica de los diagramas de colaboración.

Representación gráfica de las clases y de las instancias:

UML, ha adoptado un método para describir visualmente las instancias y distinguirlas de los tipos.

Con cada tipo de elemento del UML(clase, actor) una instancia utiliza el mismo símbolo gráfico usado para representar el tipo, pero se subraya el texto.

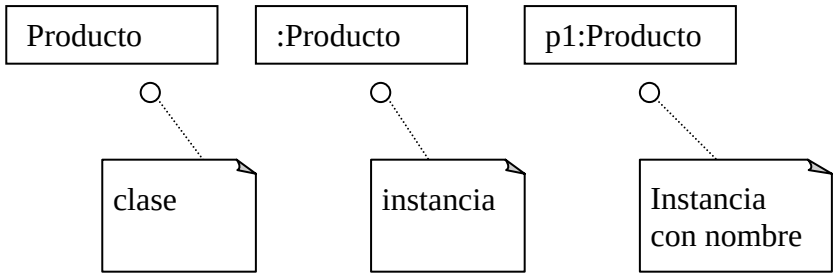


Figura 2.34 **Clase e instancias**

Esto quiere decir que, para incluir la instancia de una clase en un diagrama de interacción, se recurre al símbolo gráfico usual de la casilla de la clase, sólo que el nombre se subraya. ***En un diagrama de colaboración, al nombre de la clase siempre se le anteceden dos puntos.***

11.18.7 Representación gráfica de los vínculos:

Un **vínculo** (o enlace) es una trayectoria de conexión entre dos instancias. En un lenguaje más formal, el vínculo es una instancia de una asociación. Si vemos dos instancias en una relación, significa que los mensajes pueden enviarse de la primera instancia a la segunda.

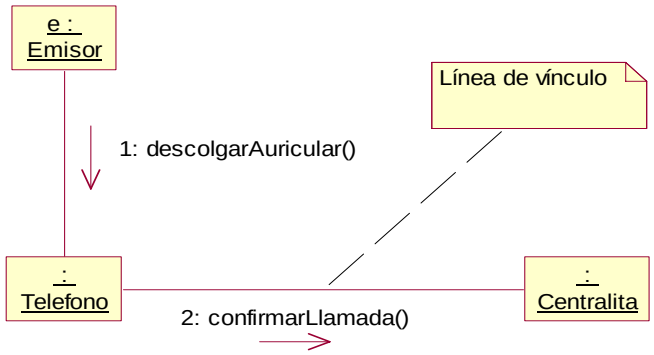


Figura 2.35 **Líneas de vínculo (enlace)**

11.18.8 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 16: Descripción de los casos reales de uso. Páginas 167-174. Presenta el concepto de caso real de uso y muestra un ejemplo.

11.19 CLASE 19: DIAGRAMAS DE COLABORACIÓN

Representación gráfica de los mensajes

Representación gráfica de los parámetros

OBJETIVOS:

Conocer la notación UML para representar gráficamente los mensajes y los parámetros en un diagrama de colaboración.

11.19.1 Representación gráfica de los mensajes.

Los mensajes entre objetos pueden representarse por medio de una flecha con un nombre y situada sobre una línea del vínculo. A través de éste puede fluir un número indefinido de mensajes. **Se agrega un número de secuencia que indique el orden consecutivo de los mensajes .**

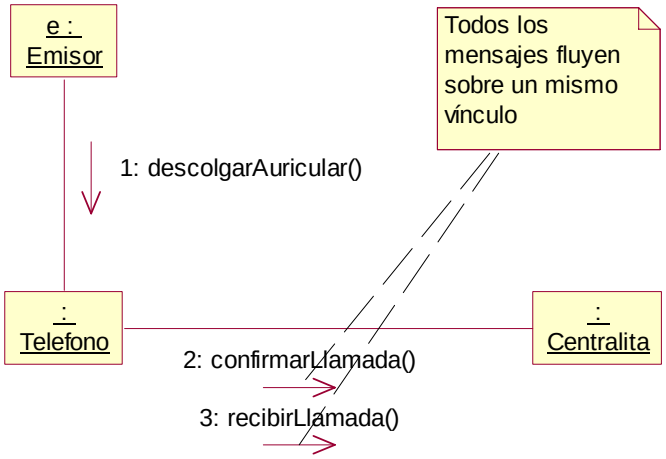


Figura 2.36 **Mensajes**

11.19.2 Representación gráfica de los parámetros.

Los parámetros de un mensaje pueden anotarse dentro de paréntesis después del nombre del mensaje. Es opcional incluir o no el tipo de parámetro.

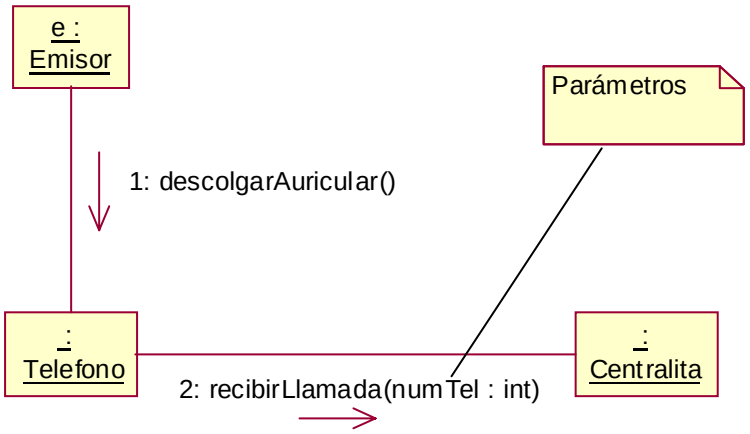


Figura 2.37 **Parámetros**

Representación gráfica del mensaje de devolver valor:

Puede incluirse un valor de retorno anteponiéndole al mensaje un nombre de variable de esa instrucción y un operador de asignación(':='). Es opcional mostrar el tipo del valor de retorno.

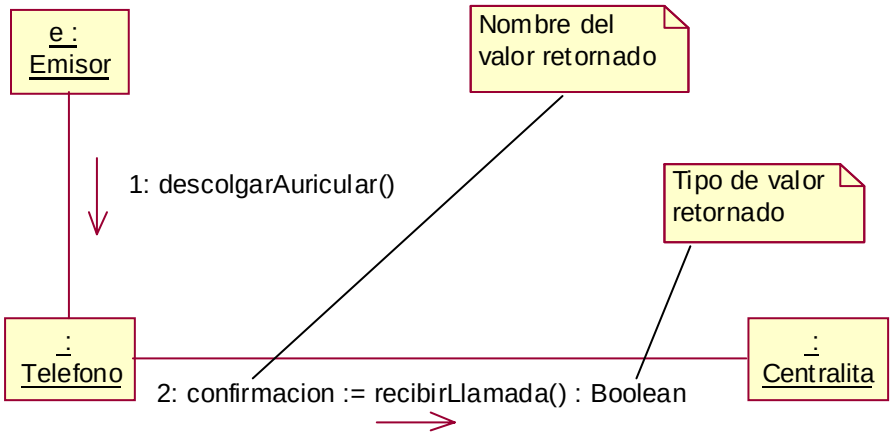


Figura 2.38 **Devolver valores**

11.19.3 Sintaxis de los mensajes:

UML cuenta con una sintaxis estándar para los mensajes:

retorno: mensaje(parametro : tipoParametro) : tipoRetorno

11.19.4 Representación gráfica de los mensajes al "emisor" o a "esto":

Un objeto puede enviarse un mensaje a sí mismo, a través de un vínculo, donde el mensaje fluye a lo largo del vínculo.

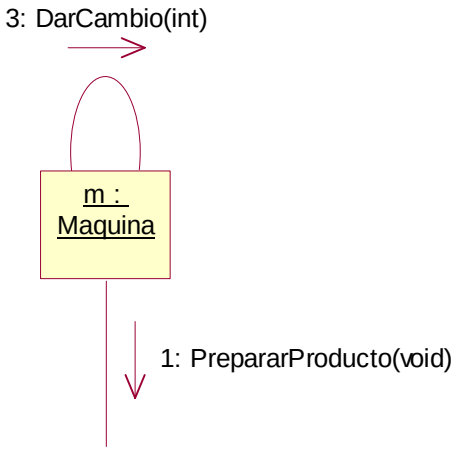


Figura 2.39 **Mensaje a "esto"**

11.19.5 Representación gráfica de una iteración:

La iteración se indica posponiendo un asterisco(*) al número de secuencia. El asterisco significa que el mensaje va a ser enviado repetidamente al receptor dentro de un ciclo. También se puede incluir una cláusula de iteración que indique los valores de recurrencia.

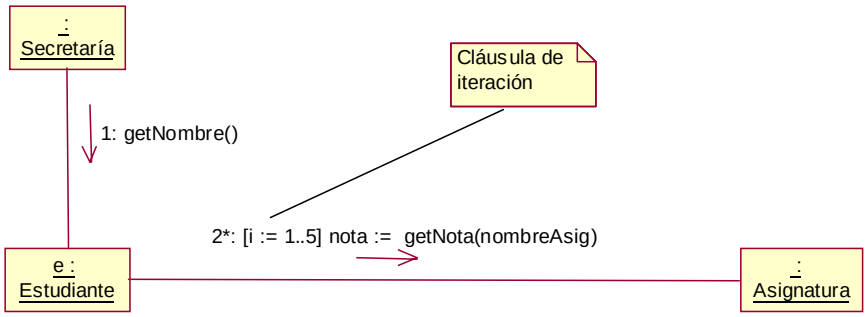


Figura 2.40 **Iteración**

11.19.6 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos. Craig Larman. Editorial Prentice Hall. Capítulo 16: Descripción de los casos reales de uso. Páginas 175-176. Presenta el concepto de caso real de uso y muestra un ejemplo.

11.20 CLASE 20: DIAGRAMAS DE COLABORACION

Representación gráfica de la creación de instancias

Representación gráfica de la secuencia de número de los mensajes

Representación gráfica de las colecciones

OBJETIVOS:

Representar gráficamente la creación de instancias y la secuencia de números de mensaje y colecciones

Si se expresa más de un mensaje dentro de la misma cláusula de iteración, se repetirá la cláusula con cada mensaje:

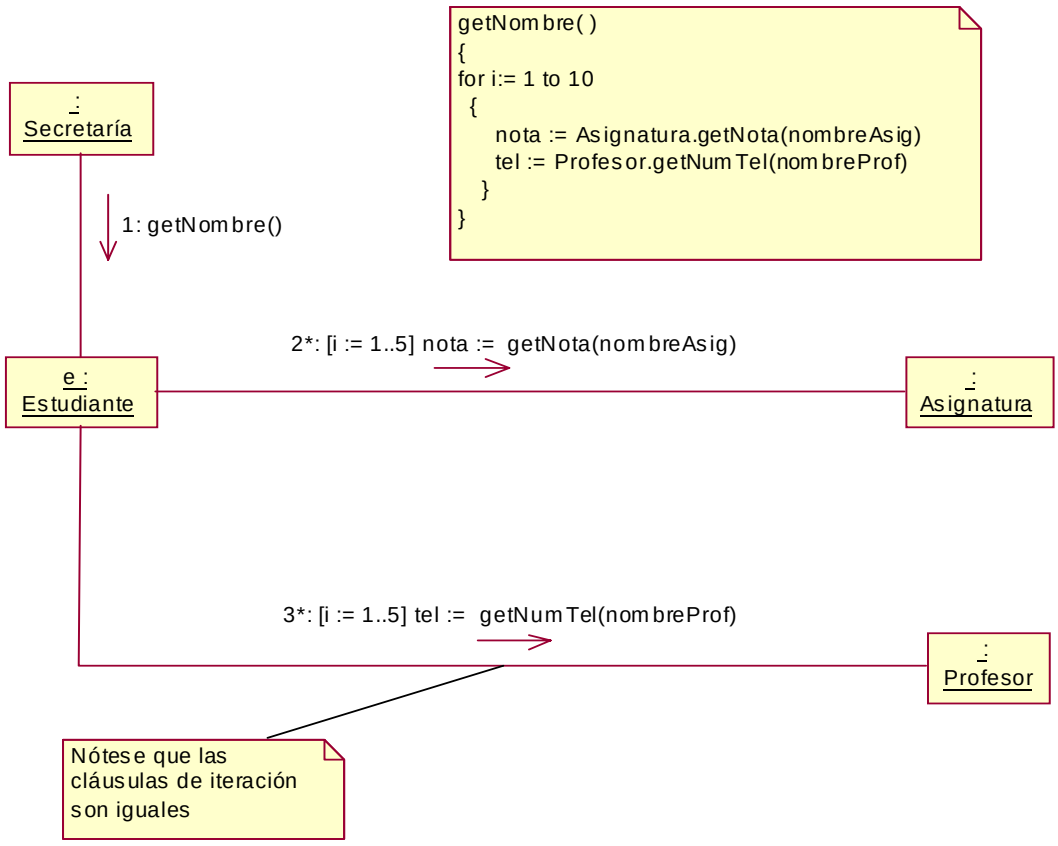


Figura 2.41 **Mensajes múltiples dentro de la misma cláusula de iteración**

11.20.1 Representación gráfica de la creación de instancias:

El mensaje de creación es **crear**, que se muestra en el momento de ser enviado a la instancia que vamos a generar.

En la mayoría de los lenguajes orientados a objetos, las instancias suelen generarse utilizando un mensaje *nuevo*(u operador) con la clase y no con una instancia, esta notación ocupa menos espacio, aunque no sea completamente exacta. Es opcional que la nueva instancia contenga o no un símbolo "nuevo".

El mensaje *crear* puede contener parámetros, lo cual indica la transferencia de los valores iniciales.

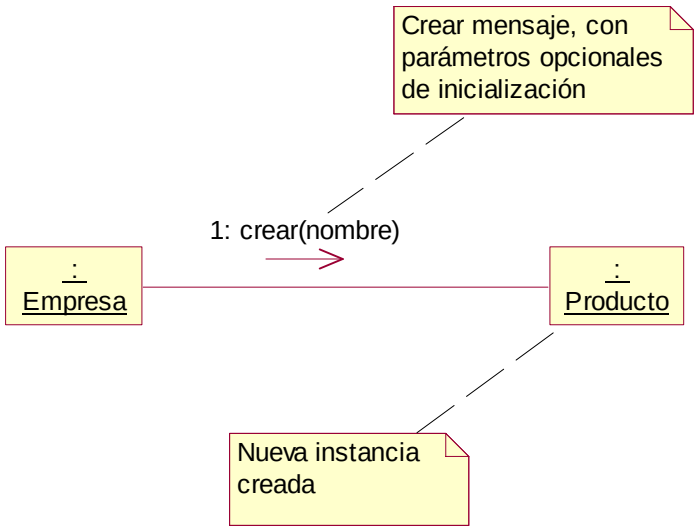


Figura 2.42 **Creación de instancias**

11.20.2 Representación gráfica de la secuencia de número de los mensajes:

El orden de los mensajes se indica con un **número de secuencia**. El esquema de numeración es:

El primer mensaje no se numera. Así, mens1( ) no lleva número.

El orden y el anidamiento de los mensajes siguientes se indican con un esquema legal de numeración, donde a los mensajes anidados se les ha antepuesto un número. La anidación se denota anteponiendo el número del mensaje de entrada al del mensaje de salida.

El siguiente es un ejemplo de la numeración de secuencia:

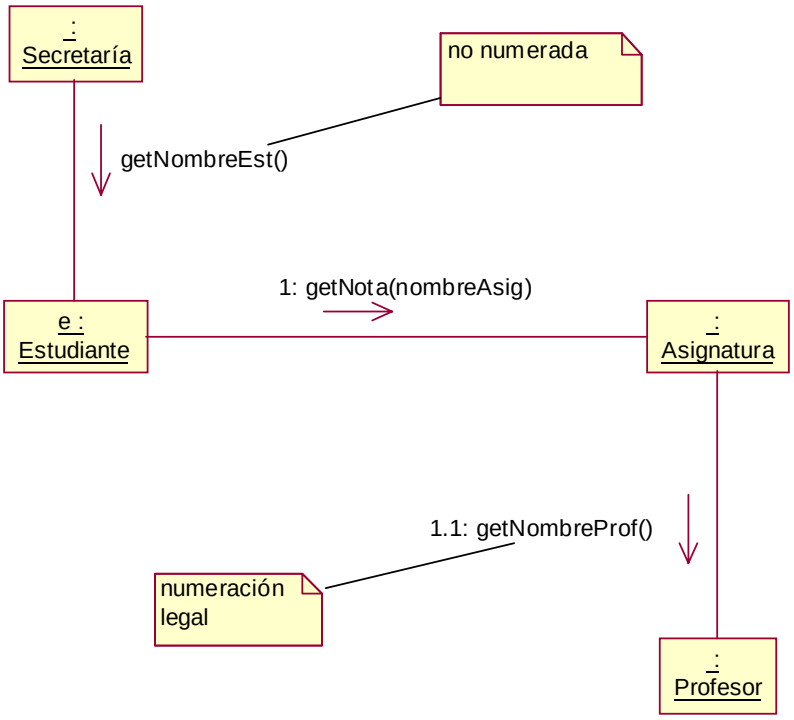


Figura 2.43 **Numeración de secuencias**

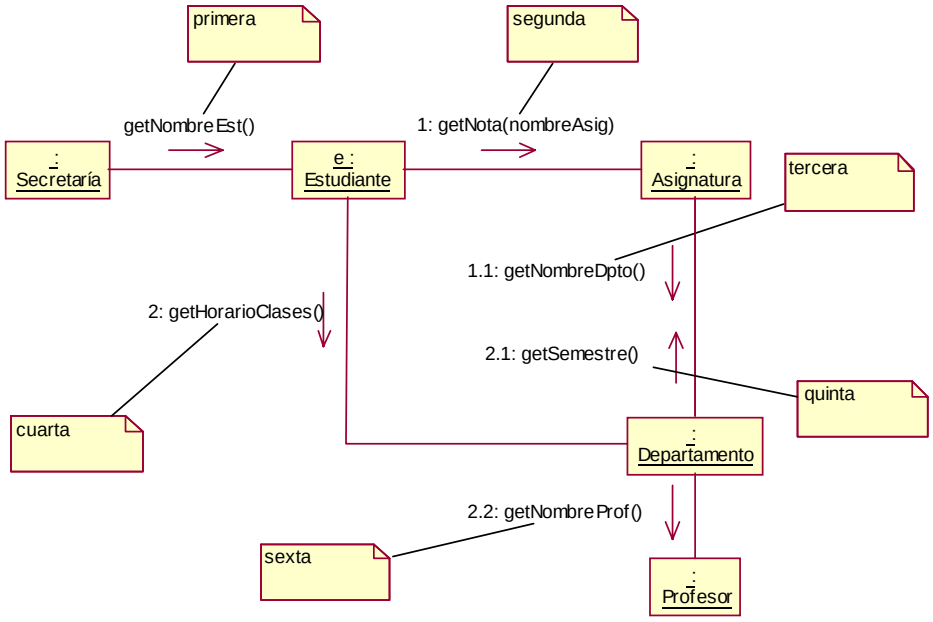


Figura 2.44 **Numeración compleja de secuencias**

11.20.3 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 16: Descripción de los casos reales de uso. Páginas 176-179. Presenta el concepto de caso real de uso y muestra un ejemplo.

11.21 CLASE 21: DIAGRAMAS DE COLABORACION

Representación gráfica de los mensajes condicionales

Representación gráfica de trayectorias condicionales mutuamente excluyentes

Representación gráfica de los parámetros

OBJETIVOS:

Representar gráficamente los mensajes condicionales y las trayectorias condicionales mutuamente excluyente.

11.21.1 Representación gráfica de los mensajes condicionales:

Un mensaje condicional se indica posponiendo al número de la secuencia una cláusula condicional entre corchetes. El mensaje se envía sólo si la cláusula se evalúa como verdadera.

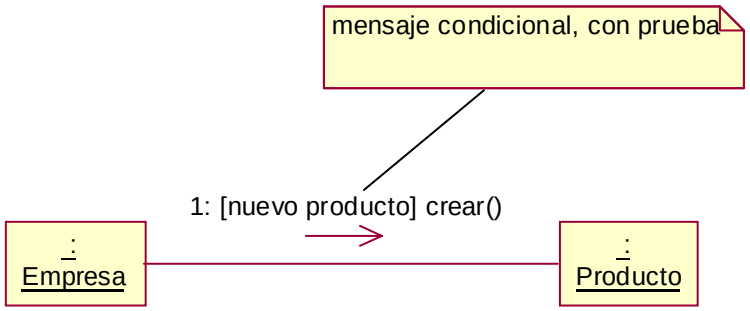


Figura 2.45 **Mensaje condicional**

11.21.2 Representación gráfica de trayectorias condicionales mutuamente excluyentes:

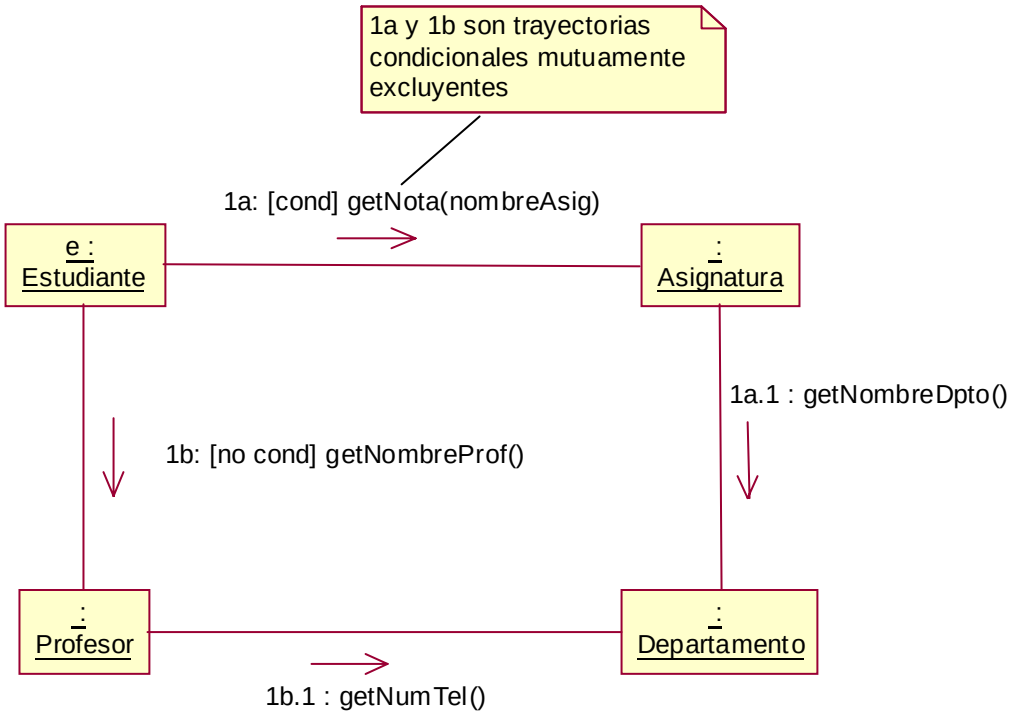


Figura 2.46 **Mensajes mutuamente excluyentes**

El diagrama anterior, contiene números de secuencia con trayectorias condicionales que se excluyen mutuamente. Es necesario modificar las expresiones de la secuencia con una letra de la trayectoria condicional. Por convención, la primera letra en usarse es **a**. Tanto 1a como 1b podrían ejecutarse después de mens1( ). Ambas son el número de la secuencia 1 porque pueden ser el primer mensaje interno. A los subsecuentes mensajes anidados se les sigue anteponiendo la secuencia de sus mensajes externos. Así 1b.1 es un mensaje anidado dentro de 1b.

11.21.3 Representación gráfica de las colecciones:

Un **multiobjeto** o conjunto de instancias, se representa como un icono de pila, tal como se muestra a continuación:

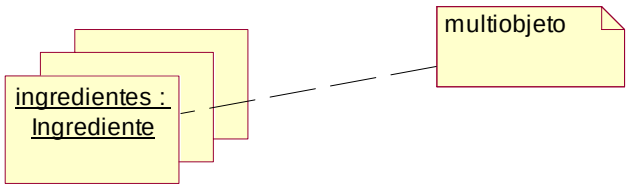


Figura 2.47 **Un multiobjeto**

Un multiobjeto suele implementarse como un grupo de instancias guardadas en un contenedor u objeto colección.

11.21.4 Representación gráfica de los mensajes dirigidos a multiobjetos:

Un mensaje dirigido a un icono multiobjeto indica que se envía al objeto colección. En UML, los mensajes dirigidos a un multiobjeto no se transmite a todos los elementos.

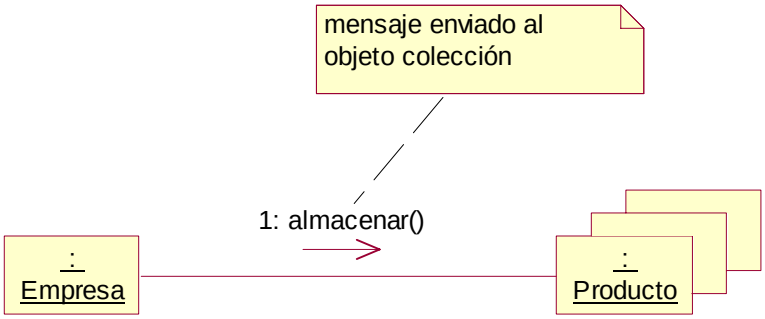


Figura 2.48 **Mensaje dirigido a un multiobjeto**

11.21.5 Representación gráfica de los mensajes dirigidos a un objeto clase:

Los mensajes pueden ser dirigidos a la propia clase y no a una instancia, con el fin de llamar los métodos de la clase.

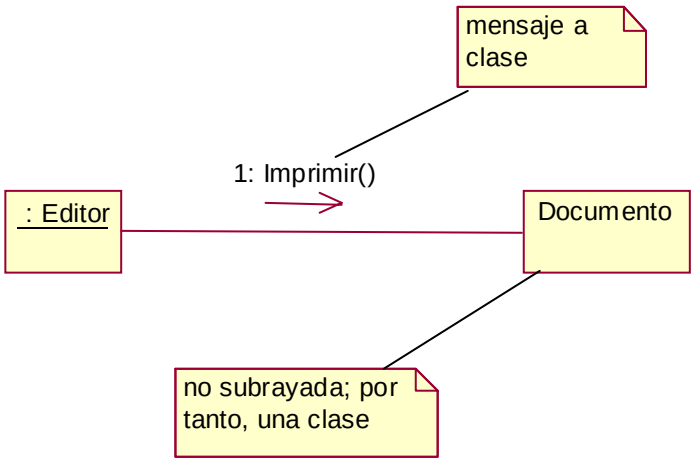


Figura 2.49 **Mensajes a un objeto clase (llamada a método estático)**

Para indicar que el mensaje va a ser enviado a una clase y no a una instancia, se incluyen dentro de una casilla de clases cuyo nombre no esté subrayado.

En el siguiente ejemplo se muestra el diagrama de colaboración para la operación .

Preparar producto( ) :

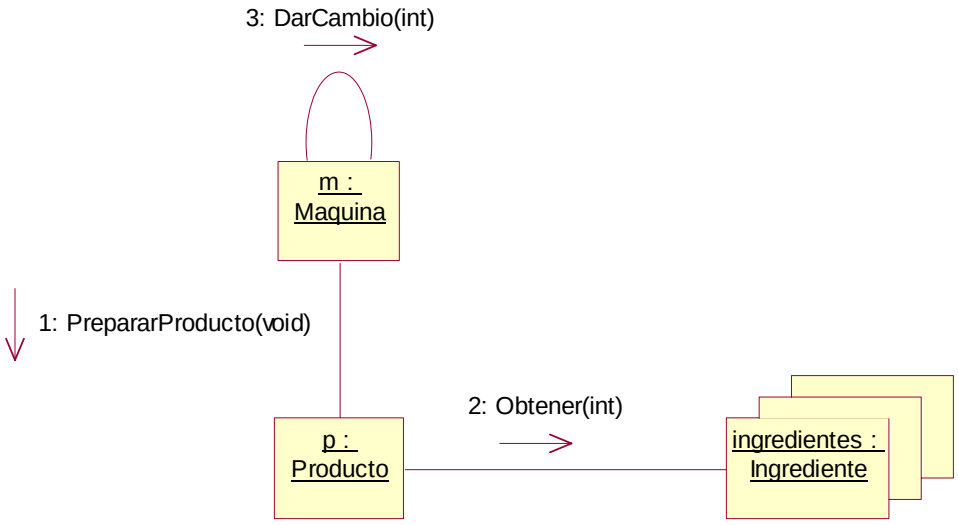


Figura 2.50 **Diagrama de colaboración para Preparar Producto**

11.21.6 Bibliografía

UML Y PATRONES, Introducción al análisis y diseño orientado a objetos.

Craig Larman. Editorial Prentice Hall. Capítulo 17: Diagramas de colaboración. Páginas 180-182. Explica el concepto de diagrama de colaboración y desarrolla todos los aspectos relacionados a su notación.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 135-143. Resume la notación de UML correspondiente a los diagramas de colaboración.

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 18: Diagramas de interacción. Páginas 215-223. Resume las principales características de un diagrama de colaboración.

UML Distilled “Applyng the standard Object Modeling Language”. Martin Fowler. Editorial Addison Wesley. Capítulo 6: Interaction Diagrams. Páginas 103-112. Presenta de manera breve las principales características de los diagramas de interacción.

11.22 CLASE 22: CLASES Y RELACIONES.

Clases.

Relaciones.

OBJETIVOS:

Definir el concepto de clase y explicar sus propiedades más importantes.

Definir el concepto de relación y dar una visión general de los diversos tipos de relaciones que se pueden presentar en la vida real.

La Técnica del Diagrama de clase se ha vuelto medular en los métodos orientados a objetos. Virtualmente, todos los métodos han incluido alguna variación de esta técnica.

El diagrama de clase, además de ser de uso extendido, también está sujeto a la más amplia gama de conceptos de modelado. Aunque los elementos básicos son necesarios para todos, los conceptos avanzados se usan con mucha frecuencia, por eso

11.22.1 Clases.

Una *clase* es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Las clases son los bloques de construcción más importantes de cualquier sistema orientado a objetos. Una clase es una abstracción de las cosas que forman parte del vocabulario del sistema que se está modelando. Una clase no es un objeto individual, sino que representa un conjunto de objetos.

11.22.2 Nombre de una clase.

Cada clase ha de tener un nombre que la distinga de otras clases. Un *nombre* es una cadena de texto. Ese nombre sólo se denomina *nombre simple*; un *nombre de camino* consta del nombre de la clase precedido por el nombre de un paquete en el que se encuentra. Una clase puede dibujarse mostrando sólo su nombre. Por ejemplo,

11.22.3 Atributos de una clase.

Desde un nivel conceptual, el atributo de nombre de un Cliente indica que los Clientes tienen nombres. Desde el nivel de especificación, este atributo indica que un objeto Cliente puede decir su nombre y tiene algún modo de establecer un nombre. En el nivel de implementación, un Cliente tiene un campo (también llamado variable de instancia o miembro de datos) para su nombre.

Dependiendo del detalle del diagrama, la notación de un atributo puede mostrar el nombre, el tipo y el valor predeterminado de un atributo (la sintaxis del UML es *visibilidad nombre: tipo = valor por omisión*, donde la *visibilidad* es igual que la de las operaciones).

¿Cuál es la diferencia entre un atributo y una asociación?

Desde la perspectiva conceptual, no hay diferencia; un atributo sólo lleva consigo otro tipo de notación, la cual puede servir si se estima conveniente. Los atributos siempre son de valor único. Por lo general, los diagramas no indican si los atributos son opcionales u obligatorios.

La diferencia se da en los niveles de especificación y de implementación. Los atributos implican navegabilidad sólo del tipo al atributo. Es más, queda implícito que el tipo contiene únicamente su propia copia del objeto de atributo, lo que implica que cualquier tipo que se emplee como atributo tendrá un valor más que unas semántica de referencia.

Lo mejor es considerar que los atributos son clases simples y pequeñas, tales como cadenas, fechas, objetos de moneda y valores no objetos, como int y real.

Un *atributo* es una propiedad de una clase identificada con un nombre, que describe un rango de valores que pueden tomar las instancias de la propiedad. Una clase puede tener cualquier número de atributos o no tener ninguno. Un atributo representa alguna propiedad del elemento que se está modelando que es compartida por todos los objetos de esa clase.

Por ejemplo, todo producto tiene un nombre y un costo. Un atributo, es por tanto, una abstracción de un tipo de dato o estado que puede incluir un objeto de la clase. En un momento dado, un objeto de una clase tendrá valores específicos para cada uno de los atributos de su clase.

Gráficamente, los atributos se listan en un compartimento justo debajo del nombre de la clase. Los atributos se pueden representar mostrando sólo sus nombres o se pueden especificar más indicando su tipo y quizás un valor inicial por defecto como se muestra a continuación:

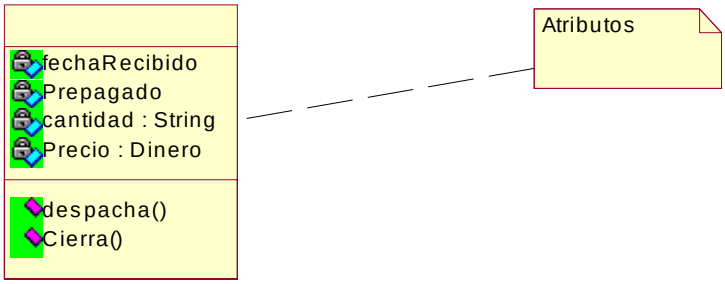


Figura 3.2 **Atributos**

Los diagramas de clase describen los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos.

Hay dos tipos principales de relaciones estáticas:

Asociaciones (por ejemplo, un cliente puede rentar diversas video-cintas).

Subtipos (una enfermera es un tipo de persona).

Los diagramas de clase también muestran los atributos y operaciones de una clase y las restricciones a que se ven sujetos, según la forma en que se conectan los objetos.

Diagrama Ejemplo de Proceso de Pedido

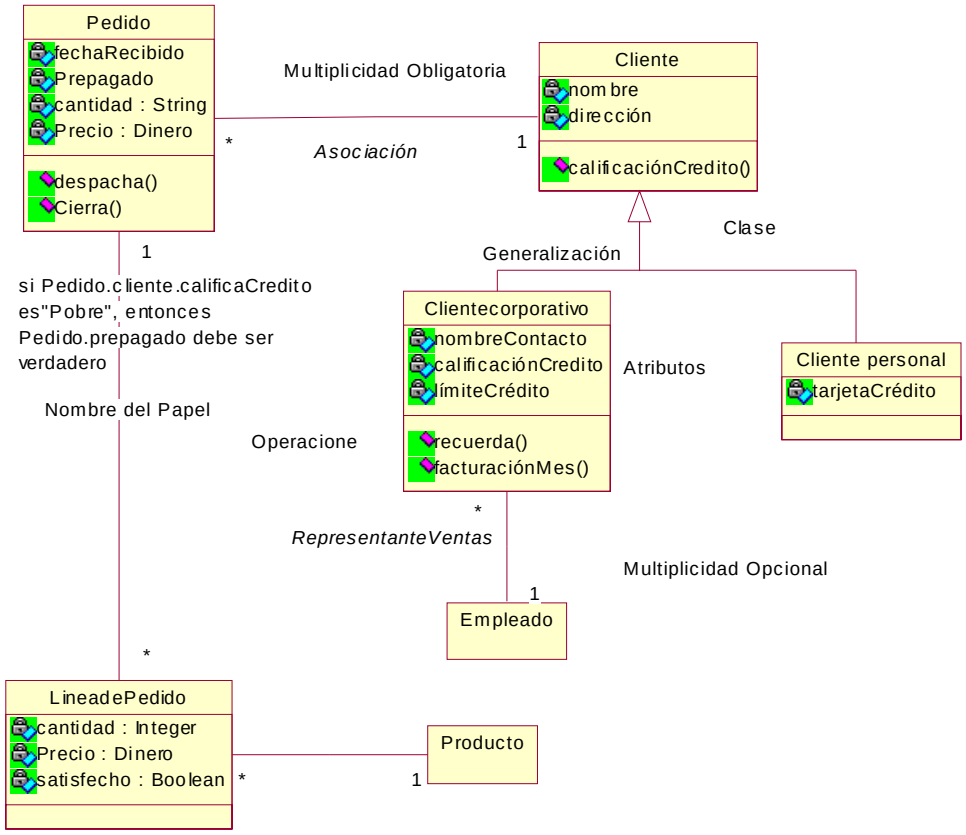


Figura 3.3 **Diagrama de clase de proceso de pedido**

11.22.4 Perpectivas

Antes de empezar a describir los diagramas de clase, quisiera señalar una importante sutileza sobre el modo en que se usan. Tal sutileza generalmente no está documentada, pero tiene sus repercusiones en el modo en que debe interpretarse un diagrama, ya que se refiere a lo que se va a describir con un modelo.

Recomiendan según Steve Cook y John Daniels (1994) consideran que hay tres perspectivas que se pueden manejar al dibujar diagramas de clase (o de hecho cualquier modelo, aunque esta división se advierte de modo especial en relación con los diagramas de clase).

Conceptual. Si se adopta la perspectiva conceptual, se dibuja un diagrama que represente los conceptos del dominio que se está estudiando. Estos conceptos se relacionan de manera natural con las clases que los implementan, pero con frecuencia no hay una correlación directa. De hecho, los modelos conceptuales se deben dibujar sin importar (o casi) el software con que se implementarán, por lo cual se puede considerar como independiente del lenguaje.

Especificaciones. Ahora estamos viendo el software, pero lo que observamos son las interfaces del software, no su implementación. Por tanto, en realidad vemos los tipos, no las clases. El desarrollo orientado a objeto pone un gran énfasis en la diferencia entre interfas e implementación, pero esto con frecuencia se pasa por alto en la práctica, ya que el concepto de clase en un lenguaje OO combina tanto la interfas como la implementación. Así, a menudo se hace referencia a las interfaces como tipos y a la implementación de esa interfas como clase. Influidos por este manejo de lenguaje, la mayor parte de los métodos han seguido este camino.

Implementación. Dentro de esta concepción, realmente tenemos clases y exponemos por completo la implementación. Esta es, probablemente, la perspectiva más empleada, pero en muchos sentidos es mejor adoptar la perspectiva de especificación.

La compresión de las perspectivas es crucial tanto para dibujar como para leer los diagramas de clase. Desafortunadamente, las divisiones entre las perspectivas no son tajante y la mayoría de los modeladores no se preocupan por definirlas con claridad cuando la dibujan.

Cuando dibuje un diagrama, hágalo desde el punto de vista de una sola perspectiva clara. Cuando lea un diagrama, asegúrese de saber desde qué perspectiva se dibujo. Dicho conocimiento es esencial si se quiere interpretar correctamente el diagrama.

11.22.5 Asociaciones

La figura 3.3 muestra un modelo de clase simple sobre un proceso de pedido. Se describirá sus partes y hablaremos sobre las maneras de interpretar, desde las distintas perspectivas.

Comenzaremos con las asociaciones. Las asociaciones representan relaciones instancias de clases (una persona trabaja para una empresa; una empresa tiene cierta cantidad de oficinas).

Desde la perspectiva conceptual, las asociaciones representan relaciones conceptuales entre clases. El diagrama indica que un pedido debe venir de un solo Cliente y que un Cliente puede hacer varios Pedidos en un periodo de tiempo. Cada uno de estos Pedidos tiene varias instancias de Línea de pedido, cada una de las cuales se refiere a un solo Producto.

Cada asociación tiene dos papales; cada papel es una dirección en la asociación. De este modo, la asociación entre Cliente y Pedido contiene dos papeles: uno del Cliente al Pedido; el segundo del Pedido al Cliente.

Se puede nombrar explícitamente un papel mediante una etiqueta. En este caso, el papel en sentido Pedido a Línea de orden se llama Artículo de línea. Si no hay etiqueta, el papel se puede nombrar de acuerdo con la etiqueta de la clase; de esta forma, el papel de Pedido a Cliente se llamará Cliente (en este trabajo se refiere a clase de donde parte el papel como origen y a la clase a donde va el papel como destino. Esto significa que hay un papel de Cliente cuyo origen es Pedido y cuyo destino es Cliente).

Un papel tiene también una multiplicidad, la cual es una indicación de la cantidad de objetos que participarán en la relación dada. En la figura 2.8 entre Cliente y Pedido

indica que el primero puede tener muchas Órdenes asociadas a él; el 1 indica que un Pedido viene de un solo Cliente.

En general, la multiplicidad indica los límites inferiores y superiores de los objetos participantes. El * representa de hecho el intervalo 0...infinito: el cliente no necesita haber colocado un Pedido, y no hay un tope superior (por lo menos en teoría) para la cantidad de pedidos que puede colocar. El 1 equivale a 1...1; cada Pedido debe haber sido solicitado por un solo Cliente.

En la práctica, las multiplicidad más comunes son 1, * , y 0...1 (se puede no tener ninguno o bien tener uno). Para una multiplicidad más general, se puede tener un solo número (por ejemplo, 11 jugadores en un equipo de fútbol), un intervalo (por ejemplo, 2,4 para los participantes de un juego de canasta) o combinaciones discretas de números e intervalos (por ejemplo, 2,4 para las puertas de un automóvil).

Dentro de la perspectiva de la especificación, las asociaciones representan responsabilidades.

En la figura 2.48 significa que hay uno o más métodos asociados con Cliente que me proporcionarán los pedidos que ha colocado un Cliente dado. Del mismo modo, existen métodos en Pedidos que permitirán saber qué Cliente colocó tal Pedido y cuales son los artículos de línea que contiene un Pedido.

La figura 2.48 implica también cierta responsabilidad al poner al día la relación. Debe haber algún modo de relacionar el Pedido con el Cliente. De nuevo, no se muestran los detalles; podría ser que se especifique el Cliente en el constructor del pedido, o tal vez, exista un método *agregarpedido* asociado al Cliente. Esto se puede hacer más explicito añadiendo operaciones al cuadro de clase.

Sin embargo, estas responsabilidades no implican una estructura de datos. A partir de un diagrama a nivel de especificación, no se puede hacer suposición alguna sobre la estructura de datos de las clases. No puede, ni debe poder decir si la clase Pedido contiene un apuntador a Cliente, o si la clase Pedido cumple su responsabilidad ejecutando algún código de selección que pregunte a cada Cliente si se refiere a un Pedido dado. El diagrama sólo indica la interfas, y nada más.

Si este fuera un modelo de implementación, diera a entender con ello que hay apuntadores en ambos sentidos entre las clases relacionadas. El diagrama diría entonces que Pedido tiene un campo que es un conjunto de apuntadores hacia Línea de pedido y también un apuntador a Cliente .

En la figura 3.4. es básicamente la misma que la figura 3.3, a excepción que se ha añadido un par de flechas a las líneas de asociación. Estas líneas indican navegabilidad.

En un modelo de especificación, esto indicaría que un Pedido tiene la responsabilidad de decir a qué Cliente corresponde, pero un Cliente no tiene la capacidad correspondiente para decir cuáles con los pedidos que tiene. En lugar de responsabilidad simétrica, aquí tenemos responsabilidad de un solo lado de la línea. En un diagrama de implementación se indicaría qué Pedido contiene un apuntador a Cliente pero Cliente no apuntaría a Pedido.

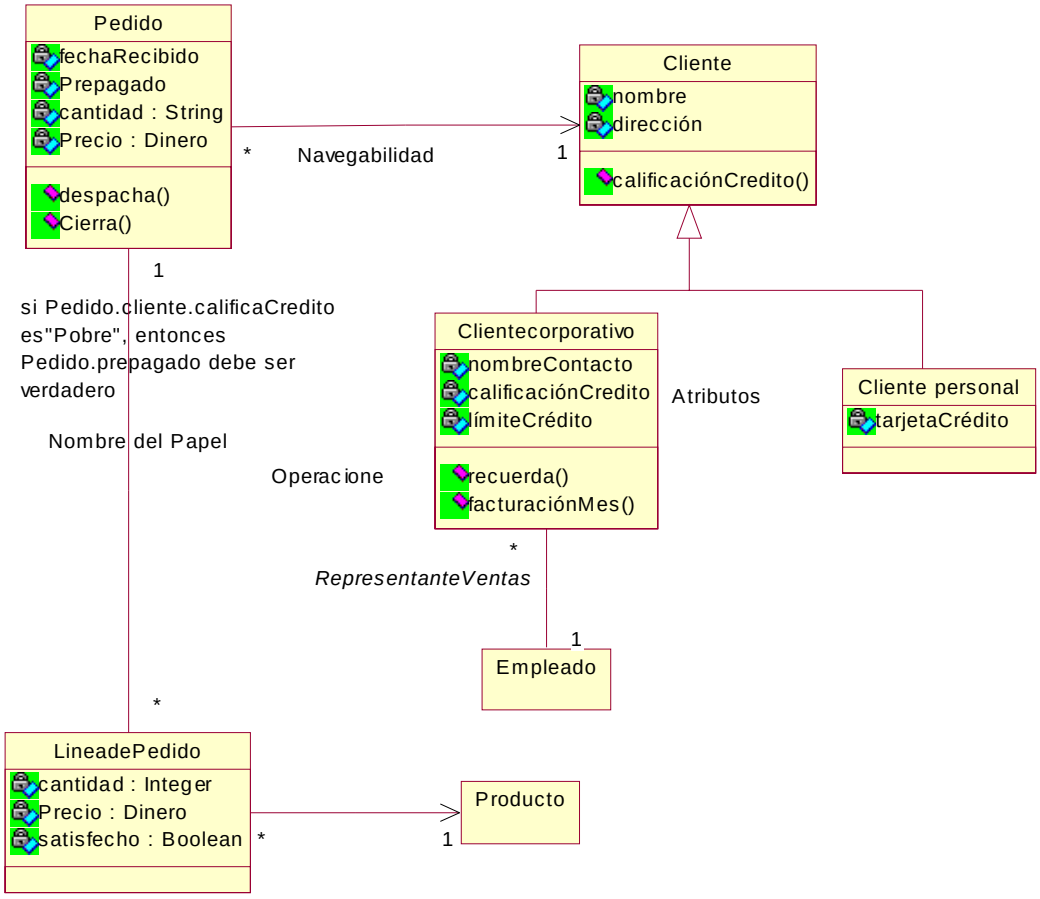


Figura 3.4 **Diagrama de clase con navegabilidad**

Como se podrá apreciar, la navegabilidad es una parte importante de los diagramas de implementación y especificación. Sin embargo la navegabilidad es cuestionada su utilidad en los diagramas conceptuales (Según UML Gota a Gota versión 1.2)

Si existe una navegabilidad en una sola dirección, a la asociación se le llama asociación unidireccional. Una asociación bidireccional contiene navegabilidad en ambas direcciones. El UML dice que las asociaciones sin flechas significan que la navegabilidad es desconocida o que la asociación es bidireccional. El proyecto debe definirse en el sentido de uno u otro de los significado.

Las asociaciones bidireccionales incluyen una restricción adicional, que consiste en que ambos papeles son inversos entre ellos. Esto es igual al concepto de funciones inversas en las matemáticas. En el contexto de la figura 2.49, esto significa que cada Artículo de la línea asociado con un Pedido se debe asociar con el Pedido original. De modo similar, si se toma una Línea de pedido y se busca en los artículos de línea el Pedido asociado, se deberá ver la Línea de pedido original en el conjunto.

Hay varias maneras de nombrar las asociaciones. A los modeladores de datos tradicionales les gusta denominar a una asociación con un verbo, de modo que se pueda emplear la relación en una oración. La mayoría de los modeladores de objetos prefieren valerse de sustantivos para denominar uno u otro papel, pues esta forma corresponde mejor a las responsabilidades y operaciones.

Gráficamente, en UML una clase se representa como un rectángulo con tres divisiones internas que corresponden al nombre, sus atributos y sus operaciones. Esta notación permite visualizar una abstracción independientemente de cualquier lenguaje de programación específico y de forma que permite resaltar las partes más importantes de una abstracción: su nombre, sus atributos y sus operaciones.

Para ejemplificar cada uno de los conceptos que se verán, vamos a tener como ejemplo el desarrollo de un sistema software que consiste en lo siguiente:

Continuando con el modelo de ejemplo de un proceso de Pedido

De acuerdo al problema anterior, haciendo una abstracción de las cosas más importantes que se utilizan en el vocabulario del sistema, algunas de las clases que podemos obtener son las siguientes:

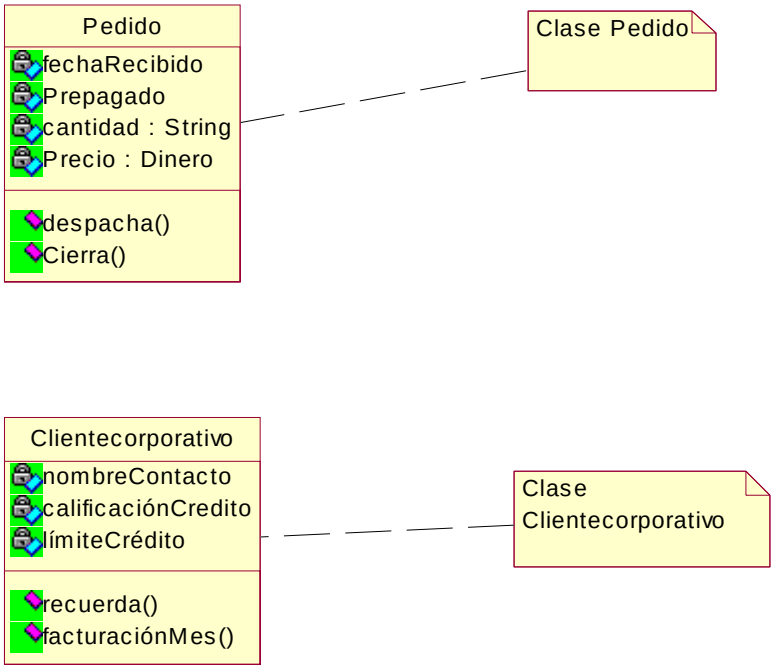


Figura 3.5 **Clases**

11.22.6 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulos 4 y 5: Clases y Relaciones. Páginas 41-64. Muestra de manera detallada las principales características de las clases y relaciones.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 2: La orientación a objetos. Páginas 33-75. Expone de manera amplia las características de las clases y relaciones.

UML GOTA A GOTA. Martin Fowler, Kendall Scott. Editorial Pearson Capítulos 4 : Diagrama de clase: fundamental. Páginas 61-72.

11.23 CLASE 23: CLASES Y RELACIONES.

Operaciones de una clase

Organizar atributos y operaciones

Tarjetas de CRC

Responsabilidades

Generalización

OBJETIVOS:

Definir el concepto de operación de clase.

Definir la organización de lo atributos y operaciones

Identificar las funciones de las tarjetas de CRC

Identificar las responsabilidades y generalización de las instancias

11.23.1 Operaciones

Las operaciones son los procesos que una clase sabe llevar a cabo. Evidentemente, corresponden a los métodos sobre una clase. En el nivel de especificación, las operaciones corresponden a los métodos públicos sobre un tipo. En general, no se muestran aquellas operaciones que simplemente manipulan atributos, ya que por lo común, se pueden inferir. Sin embargo, tal vez sea necesario indicar si un atributo dado es de sólo lectura o está inmutable congelado (esto es, que su valor nunca cambia). En el modelo de implementación, se podrían mostrar también las operaciones privadas y protegidas.

La sintaxis del UML completa para las operaciones es

Visibilidad nombre (lista-parámetros):expresiones-tipo-de-datos-a-regresar {cadena-de-propiedades}

Donde

Visibilidad es + (public), # (protecd), o-(private)

Nombre es una cadena de carácter (string)

Lista-de-parámetros contiene argumentos (opcionales) cuya sintaxis es la misma que la de los atributos

Expresión-tipo-de-dato-a-regresar es una especificación opcional dependiendo del lenguaje

Cadena-de-propiedades indica valores de propiedad que se aplican a la operación dada

Un ejemplo de operación sería: +últimaCantidadDe(valorTipoFenomeno):Cantidad

Dentro de los modelos conceptuales, las operaciones no deben tratar e especificar la interfas de una clase. En lugar de ello, deberán indicar las principales responsabilidades de dicha clase.

11.23.2 Operaciones de una clase.

Una *operación* es la implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento. Es decir, una operación es una abstracción de algo que se puede hacer a un objeto y que es compartido por todos los objetos de la clase. Una clase puede tener cualquier número de operaciones o ninguna. Por ejemplo, a todos los objetos de la clase *Ingrediente* se les puede aplicar la operación *agregar* que consistiría en aumentar la cantidad que exista de ese ingrediente en un momento determinado. A menudo (pero no siempre), la invocación de una operación sobre un objeto cambia los datos o el estado del objeto.

Gráficamente, las operaciones se listan en un compartimento justo debajo de los atributos de la clase. Las operaciones se pueden representar mostrando sólo sus nombres. Una operación se puede especificar indicando su *signatura*, la cual incluye el nombre, tipo y valores por defecto de todos los parámetros y en el caso de las funciones un tipo de retorno.

11.23.3 Organización de atributos y operaciones.

Cuando se dibuja una clase, no es necesario mostrar todos los atributos y las operaciones. De hecho, en la mayoría de los casos, no se puede (hay demasiados para ponerlos en una figura) y posiblemente no se debería hacer (probablemente sólo serán relevantes un subconjunto de esos atributos y operaciones para una vista específica. Por estas razones se puede abreviar una clase, es decir, se puede decidir mostrar sólo algunos, o incluso ninguno, de sus atributos y operaciones.

Un compartimiento vacío no significa necesariamente que no haya operaciones o atributos, sólo que se ha decidido no mostrarlos. Se puede especificar explícitamente que hay más atributos o propiedades que los mostrados acabando la lista con puntos suspensivos (“...”).

11.23.4 Tarjetas de CRC

A fines de la decada de 1980, uno de los centros más grandes de tecnologías de objetos era el laboratorio de investigación Tektronix, en Pórtland, Oregon, Estados Unidos. Este laboratorio tenia algunos de los principales usuarios de Smalltalk y muchas de las ideas clave de la tecnología de objetos se desarrollaron allí. Dos de sus programadores renombrados de Smalltalk eran Ward Cuningham y kent Beck.

En lugar de utilizar diagramas para desarrollar modelos, como lo hacían la mayoría de los metodólogos Cuningham y kent Beck representaron las clases en tarjetas 4 X 6 [pulgadas]. Y en lugar de indicar atributos y métodos en las tarjetas, escribieron responsabilidades.

Ahora bien, ¿qué es una responsabilidad? En realidad es una descripción de alto nivel del propósito de una clase. La idea es tratar de eliminar la descripción de pedazos de

datos y procesos y, en cambio, captar el propósito de la clase en unas cuantas frases. El que se haya seleccionado una tarjeta es deliberado. No se permite escribir más de lo que cabe en una tarjeta (véase la figura 3-4)

Nombre de la Clase

Responsabilidad	Pedido	Colaboración
Revisa si hay elementos en existencia		Línea de pedido
Determina precio		Línea de pedido
Revisa si el pago es válido		Cliente
Despacha a la dirección de entrega		

Tarjeta de Clase – Responsabilidad-Colaboración (CRC)

Con cada responsabilidad se indica cuales son las otras clases con las que se tiene que trabajar para cumplirla. Esto da cierta idea sobre los vínculos entre las clases, siempre a alto nivel.

Uno de los principales beneficios de las tarjetas CRC es que alienta la disertación animada entre los desarrolladores. Son especialmente eficaces cuando se está en medio de un caso de uso para ver cómo lo van a implementar las clases. Los desarrolladores escogen tarjetas a medida que cada clase colabora en el caso de uso. Conforme se van formando ideas sobre las responsabilidades, se pueden escribir en las tarjetas. Es importante pensar en las responsabilidades, ya que evita pensar en las clases como simple depositarias de datos, y ayuda a que el equipo se centre en comprender el comportamiento de alto nivel de cada clase.

Un error común que cometen la gente es generar largas listas de responsabilidades de bajo nivel. Este procedimiento se completamente fallido. Las responsabilidades deben caber sin dificultad en una tarjeta. Se cuestiona cualquier tarjeta que tenga más de tres responsabilidades. Plantéese la pregunta de si se deberá dividir la clase y si las responsabilidades se podrían indicar mejor integrándose en enunciados de un mayor nivel.

11.23.5 Cuándo usar las tarjetas de CRC

Algunos consideran maravillosas las tarjetas de CRC, en cambio, a otros, esta técnica los deja indiferentes.

Recomiendan definitivamente que se debería probar, a fin de saber si el equipo de trabajo le gusta trabajar con ellas. Se deben usar, en particular, si el equipo se ha empantanado en demasiados detalles o si parecen identificar clases apelmazadas y carentes de definiciones claras.

Se pueden emplear diagramas de clase y diagramas de interacciones para captar y formalizar los resultados del modelado CRC en un diseño con notación UML. Asegúrese que cada clase en su diagrama de clase tiene un enunciado de sus responsabilidades.

Es recomendable hacer la diferencia entre aquellas operaciones que cambian el estado de una clase y aquellas que no lo hacen. Una consulta es una operación que obtiene

un valor de una clase sin que cambie el estado observable de tal clase. El estado observable de un objeto es el estado que se puede determinar a partir de sus consultas asociadas.

Considérese un objeto de Cuenta que calcula su balance a partir de una lista de entrada. Para mejorar el desempeño, Cuenta puede poner en un campo caché el resultado de cálculo de balance, para consultas futuras. Por tanto, si el caché está vacío, la primera vez que se llama a la operación "balance", podrá el resultado en el campo caché. La operación "balance" cambia así el estado real del objeto Cuenta, pero no su estado observable, pues todas las consultas devuelven el mismo valor, esté o no lleno el campo caché. Las operaciones que si cambian el estado observable de un objeto se llaman modificadores.

Es necesario tener perfectamente clara la diferencia entre consulta y modificadores. Las consultas se pueden ejecutar en cualquier orden, pero la secuencia de los modificadores es más importante. Es recomendable como política evitar que los modificadores devuelvan valores, con el fin de mantenerlo separados.

Otra diferencia es la que se da entre operación y método. Una operación es algo que se invoca sobre un objeto (la llamada de procedimiento), mientras que un método es el cuerpo del procedimiento. Los dos son diferentes cuando se tiene polimorfismo. Si se tiene un supertipo con tres subtipos, cada uno de los cuales suplanta la operación del supertipo, entonces lo que hay es una operación y cuatro métodos que la implementan.

Es muy común que operación y método se empleen indistintamente, pero hay veces en que es necesario precisar la diferencia. En algunas ocasiones, la gente distingue entre una y otra mediante los términos llamada a un método, o declaración de método (en lugar de operación), y cuerpo del método.

11.23.6 Responsabilidades.

Una *responsabilidad* es un contrato o una obligación de una clase. Al crear una clase, se está expresando que todos los objetos de esa clase tienen el mismo tipo de estado y el mismo tipo de comportamiento. Los atributos y operaciones son simplemente las características por medio de las cuales se llevan a cabo las responsabilidades de la clase. Por ejemplo, la clase *producto* es responsable de saber sobre los nombres de productos y del costo que tienen.

Una clase puede tener cualquier número de responsabilidades, aunque, en la práctica, cada clase bien estructurada tiene al menos una responsabilidad y a lo sumo unas pocas. Gráficamente, las responsabilidades se pueden expresar en un compartimento separado al final del icono de la clase. Las responsabilidades son sólo texto libre, y en la práctica, una responsabilidad aislada se escribe como una frase, una sentencia o (como mucho) un párrafo cortos.

11.23.7 Relaciones.

Una relación es una conexión entre elementos. En el modelado orientado a objetos, las tres relaciones más importantes son las dependencias, las generalizaciones y las asociaciones. Gráficamente, una relación se representa como una línea, usándose diferentes tipos de línea para diferenciar los tipos de relaciones.

11.23.8 Dependencia.

Una *dependencia* es una relación de uso que declara que un cambio en la especificación de un elemento puede afectar a otro elemento que la utiliza, pero no necesariamente a la inversa. Gráficamente, una dependencia se representa como una línea discontinua dirigida hacia el elemento del cual se depende. Las dependencias se usarán cuando se quiera indicar que un elemento utiliza otro.

La mayoría de las veces, las dependencias se utilizarán en el contexto de las clases, para indicar que una clase utiliza a otra como argumento en la signatura de una operación. Por ejemplo, a continuación se puede ver que si la clase “canal” cambia, la operación “reproducir” de la clase “PelículaVídeo” también puede verse afectada:

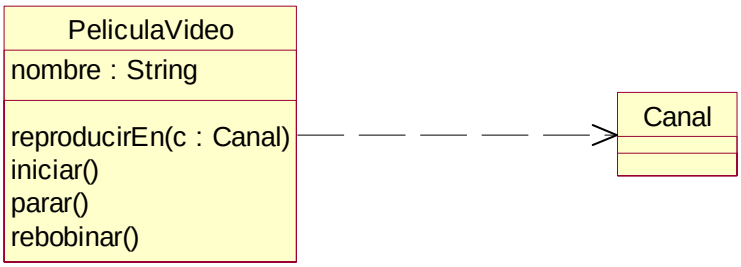


Figura 3.8 **Dependencias**

11.23.9 Generalización.

Una *generalización* es una relación entre un elemento general (llamado superclase o padre) y un caso más específico de ese elemento (llamado subclase o hijo). La generalización se llama a veces relación “es un tipo de”. La generalización significa que los objetos hijos se pueden emplear en cualquier lugar que pueda aparecer el padre, pero no a la inversa, es decir, la generalización significa que el hijo puede sustituir al padre.

Una clase hija hereda las propiedades de sus clases padres, especialmente sus atributos y operaciones. A menudo (no siempre) el hijo añade atributos y operaciones a los que hereda de sus padres. Una operación de un hijo con la misma signatura que una operación del padre redefine la operación del padre; esto se conoce como “polimorfismo”. Gráficamente, la generalización se representa como una línea dirigida continua, con una gran punta de flecha vacía, apuntando al padre, por ejemplo:

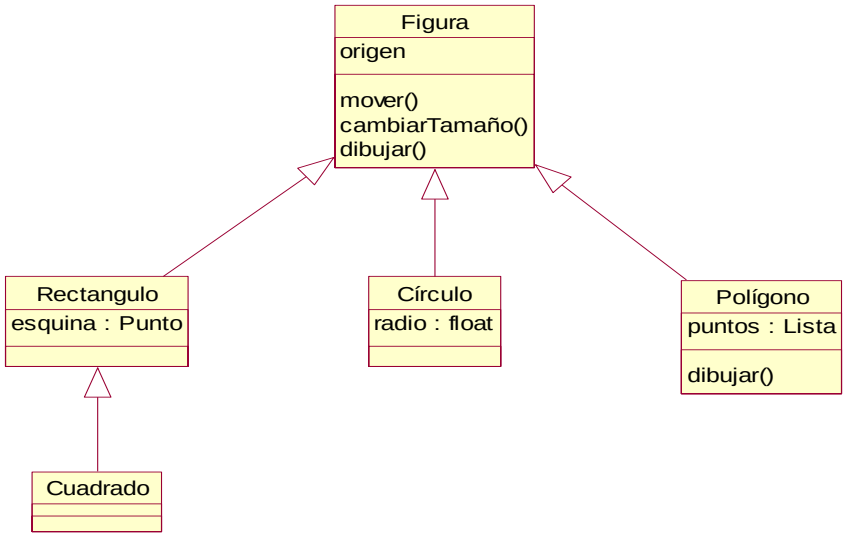


Figura 3.9 **Generalización**

Una clase puede tener ninguno, uno o más padres. Una clase sin padres y uno o más hijos se denomina clase raíz o clase base. Una clase sin hijos se llama clase hoja. Una clase con un único padre se dice que utiliza herencia simple; una clase con más de un padre se dice que utiliza herencia múltiple.

11.23.10 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulos 4 y 5: Clases y Relaciones. Páginas 41-64. Muestra de manera detallada las principales características de las clases y relaciones.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 2: La orientación a objetos. Páginas 33-75. Expone de manera amplia las características de las clases y relaciones.

11.24 CLASE 24: CLASES Y RELACIONES.

Asociaciones

Agregaciones

OBJETIVOS:

Identificar las asociaciones y agregaciones en las instancias

11.24.1 Asociación.

Una *asociación* es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro. Dada una asociación entre dos clases, se puede navegar desde un objeto de una clase hasta un objeto de la otra clase, y viceversa. Es legal que ambos extremos de una asociación estén conectados a la misma clase. Esto significa que, dado un objeto de la clase, se puede conectar con otros objetos de la misma clase.

Una asociación que conecta exactamente dos clases se dice "binaria". Aunque no es frecuente, se pueden tener asociaciones que conecten más de dos clases; estas son llamadas asociaciones n-arias. Gráficamente, una asociación se representa como una línea continua que conecta la misma o diferentes clases. Aparte de esta forma básica, hay cuatro adornos que se aplican a las asociaciones:

Nombre: una asociación puede tener un nombre, que se utiliza para describir la naturaleza de la relación. Para que no haya ambigüedad en su significado, se puede dar una dirección al nombre por medio de una flecha que apunte a la dirección en la que se pretende que se lea el nombre, por ejemplo:

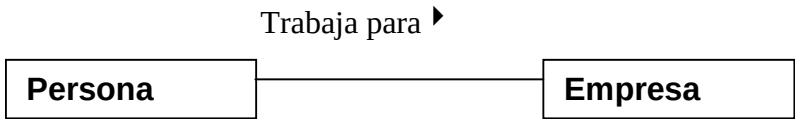


Figura 3.10 **Nombre de asociaciones**

Rol: cuando una clase participa en una asociación, tiene un rol específico que juega en la asociación; un rol es simplemente la cara que la clase de un extremo de la asociación presenta a la clase del otro extremo. Se puede nombrar explícitamente el rol que juega una clase en una asociación. Por ejemplo, una "Persona" que juega el rol de "empleado" está asociada con una empresa que juega el rol de "patrón":

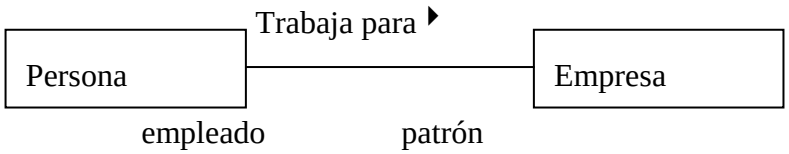


Figura **3.11 Roles**

Multiplicidad: una asociación representa una relación estructural entre objetos. En muchas situaciones de modelado, es importante señalar cuántos objetos pueden conectarse a través de una instancia de una asociación. Este "cuántos" se denomina multiplicidad del rol de la asociación, y se escribe como una expresión que se evalúa a un rango de valores o a un valor explícito. Por ejemplo:

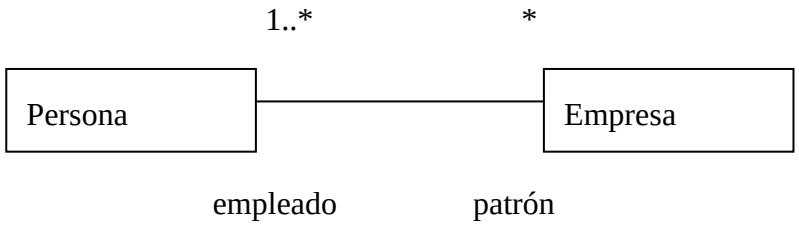


Figura 3.12 **Multiplicidad**

Cuando se indica una multiplicidad en un extremo de una asociación, se está especificando que, para cada objeto de la clase en el extremo opuesto debe haber tantos objetos en este extremo. Se puede indicar una multiplicidad de exactamente uno (1), cero o uno (0..1), muchos (0..*), o uno o más (1..*). Incluso se puede indicar un número exacto (por ejemplo, 3).

11.24.2 Agregación.

Una asociación normal entre dos clases representa una relación estructural entre iguales, es decir, ambas clases están conceptualmente en el mismo nivel, sin ser ninguna más importante que la otra. A veces, se desea modelar una relación "todo/parte", en la cual una clase representa una cosa grande "el todo", que consta de elementos más pequeños "las partes". Este tipo de relación se denomina *agregación*, la cual representa una relación del tipo "tiene un", o sea, un objeto del todo tiene objetos de la parte. En realidad, la agregación es sólo un tipo especial de asociación y se especifica añadiendo a una asociación normal un rombo vacío en la parte del todo. Por ejemplo:

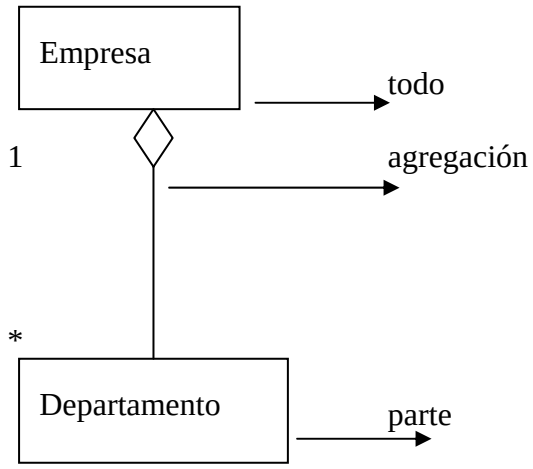


Figura 3.13 **Agregación**

11.24.3 Bibliografia

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulos 4 y 5: Clases y Relaciones. Páginas 41-64. Muestra de manera detallada las principales características de las clases y relaciones.

Modelado de objetos con UML.

Pierre-Alain Muller. Editorial Eyrolles. Capítulo 2: La orientación a objetos. Páginas 33-75. Expone de manera amplia las características de las clases y relaciones.

11.25 CLASES 25, 26 y 27: MECANISMOS COMUNES Y DIAGRAMAS.

Mecanismos comunes.

Estereotipos.

Clasificación dinámica

OBJETIVOS:

Definir el concepto de mecanismo comunes en el lenguaje UML

Definir el concepto de Estereotipos en el lenguaje UML

Identificara la Clasificación Dinámica

11.25.1 Mecanismos comunes.

UML se hace más sencillo por la presencia de cuatro mecanismos comunes que se aplican consistentemente en todas las partes del lenguaje: especificaciones, adornos, divisiones comunes y mecanismos de extensibilidad. Las notas son el tipo de adorno más importante que se utiliza. En los mecanismos de extensibilidad se encuentran los estereotipos, los valores etiquetados y las restricciones.

11.25.2 Notas.

Una *nota* es un símbolo gráfico para representar restricciones o comentarios asociados a un elemento o una colección de elementos. Gráficamente, una nota se representa como un rectángulo con una esquina doblada, junto a un comentario textual o gráfico.

Las notas se utilizan para añadir a un modelo información tal como requisitos, observaciones, revisiones y explicaciones; son el mecanismo que proporciona UML para capturar comentarios arbitrarios y restricciones que ayuden a clarificar los modelos que se han creado; pueden mostrar artefactos que juegan un papel importante en el ciclo de vida de desarrollo del software, como son los requisitos, o pueden simplemente mostrar observaciones, revisiones o explicaciones en formato libre. Por ejemplo:

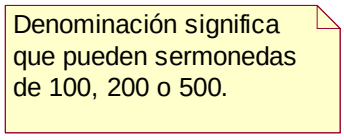


Figura 3.14 **Notas**

Una nota con un comentario no tiene un efecto semántico, es decir, su contenido no altera el significado del modelo al que está anexa. Por eso las notas se utilizan para especificar cosas como requisitos, observaciones, revisiones y explicaciones, además de

representar restricciones. Una nota puede contener cualquier combinación de texto y gráfico, se podría incluir un URL activo o incluso podría incluir o enlazar otro documento, por ejemplo:

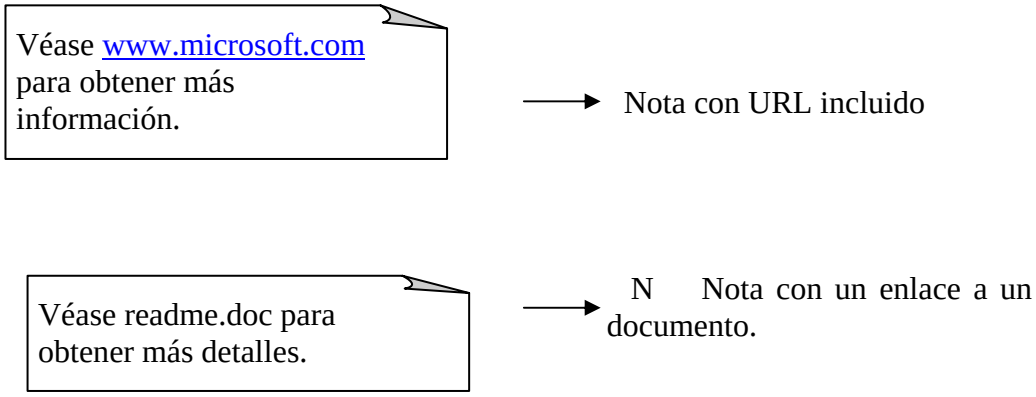


Figura 3.15 **Notas**

11.25.3 Estereotipos.

La idea de los estereotipos fue acuñada por Rebeca Wirf-Brock (Wirfs-Brock et al, 1990). El concepto ha sido adoptado con entusiasmo por los inventores de UML, aunque de un modo que en realidad no significa lo mismo. Sin embargo, ambas ideas son validas.

La idea original de un estereotipo se refería a una clasificación de alto nivel de un objeto que diera alguna indicación del tipo de objetos que era. Un ejemplo es la diferencia entre un “controlador” y un “coordinador”.

Con frecuencia se encuentran diseño OO en los que una clase parece hacer todo el trabajo, muchas veces por medio de una gran método *hazlo (dolt)*, mientras que las demás clases no hacen otra cosa más que encapsular datos. Este es un diseño pobre, pues significa que el controlador es muy complejo y difícil de manejar.

Para mejorar esta situación, se traslada el comportamiento del controlador a los objetos de datos relativamente tontos, de modo que éstos se vuelven. Así, el controlador se convierte en coordinador. El coordinador es el encargado de disparar tareas en una secuencia particular, pero otros objetos son los que saben cómo desempeñarlas.

La esencia del estereotipo consiste en que sugiere ciertas responsabilidades generales de una clase. El UML ha adoptado este concepto y lo ha convertido en un mecanismo general de extensión del lenguaje mismo.

En su trabajo original (1994), jacobson clasificaba todas las clases de un sistema en tres categorías: objetos de interfaz, objetos de control y objeto de entidad (sus objetos de

control, cuando están bien diseñados, son parecidos a los coordinadores de Wirfs-Brock). Jacobson sugirió reglas para la comunicación entre estos tipos de clases y les dio a cada una de ellas un icono diferente. Estas distinción no es una parte medular de UML. Al contrario, este tipo de clases contribuyen en realidad estereotipo de clases; de hecho, son estereotipo, en el sentido que Wirfs-Brock le da el término.

Los estereotipos generalmente se indican en el texto entre comillas francesas (<<objeto control>>), pero también se pueden mostrar definiendo un icono para el estereotipo. De lo que se trata es de que, si no se está empleando el enfoque de Jacobs, se pueden olvidar los estereotipo. Pero se quiere trabajar con dicho enfoque, se pueden definir los estereotipos y las reglas para su uso.

Muchas extensiones del núcleo de UML se pueden describir como una colección de estereotipos. En los diagramas de clase, pueden ser estereotipos de clases, asociaciones o generalizaciones. Pueden pensarse en los estereotipos como subtipos de los tipos Clase, Asociaciones y Generalización en el metamodelo.

Frecuentemente tienden a confundir las restricciones, es decir, las condiciones limitaciones con los estereotipos. Si se marca una clase como abstracta ¿se trata de una restricción o de un estereotipo? Los documentos oficiales actuales llaman a esto restricciones, pero hay que estar consciente de que se hacen un uso confuso de una y otro. Esto es de sorprender, ya que los subtipos son con frecuencia más limitados que los supertipos.

Un *estereotipo* es una extensión del vocabulario de UML que permite crear nuevos tipos de bloques de construcción similares a los existentes, pero específicos del problema que se está modelando. Gráficamente, un estereotipo se representa como un nombre entre comillas tipográficas y colocado sobre el nombre de otro elemento. Como señal visual, se puede definir un icono para el estereotipo y mostrar ese icono a la derecha del nombre o utilizar ese icono como símbolo básico para el elemento estereotipado. Ejemplos:

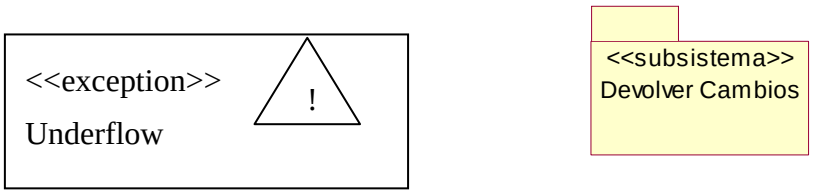


Figura 3.16 **Estereotipos**

11.25.4 Clasificación múltiple y dinámica

las clasificaciones se refiere a la relación entre un objeto y su tipo.

La mayor parte de los métodos hacen ciertas suposiciones sobre este tipo de relación (premisa que están presentes también en los principales lenguajes de programación OO). Estas premisas que están fueron cuestionadas por JIM Odell, quien las considero demasiado restrictivas para el modelo conceptual. Las suposiciones son para una clasificación simple

y estática de los obsjetos; Odell sugiere emplear una clasificación múltiple y dinámica de los objetos en los modelos conceptuales.

En la clasificación simple, un objeto pertenece a un solo tipo, que puede haber heredado de supertipos. En la clasificación múltiple, un objeto puede ser descrito por varios tipos que no están conectados necesariamente por medio de herencia.

Obsérvese que la clasificación múltiple es diferente de la herencia múltiple. La herencia múltiple plantea que un tipo puede tener muchos supertipos, pero que se debe definir un solo tipo por cada objeto sin definir un tipo especifico para tal fin.

Como ejemplo, considere una persona subtipificada como hombre o mujer, doctor o enfermera, paciente o no (vea la figura 3.14). La clasificación múltiple permite asignar cualquiera de estos tipos a un objeto en cualquier combinación posible, sin necesidad de que se definan tipos para todas las combinaciones legales.

Si se emplea la clasificación múltiple, habrá de asegurarse que haya quedado claro cuáles son las combinaciones legales. Esto se hace etiquetando una línea de generalización con un discriminador, el cual es una indicación de la base de subtipificación. Varios subtipos pueden compartir el mismo discriminador. Todos los subtipos con el mismo discriminador están desunidos, es decir, cualquier instancia del supertipo puede ser una instancia de sólo uno de los subtipos, dentro del discriminador. Una buena convención es que todas las subclases que emplean un discriminador desemboque en un triangulo, como se muestra en la figura 3.14. Se puede tener, como alternativa, varias flechas con la misma etiqueta.

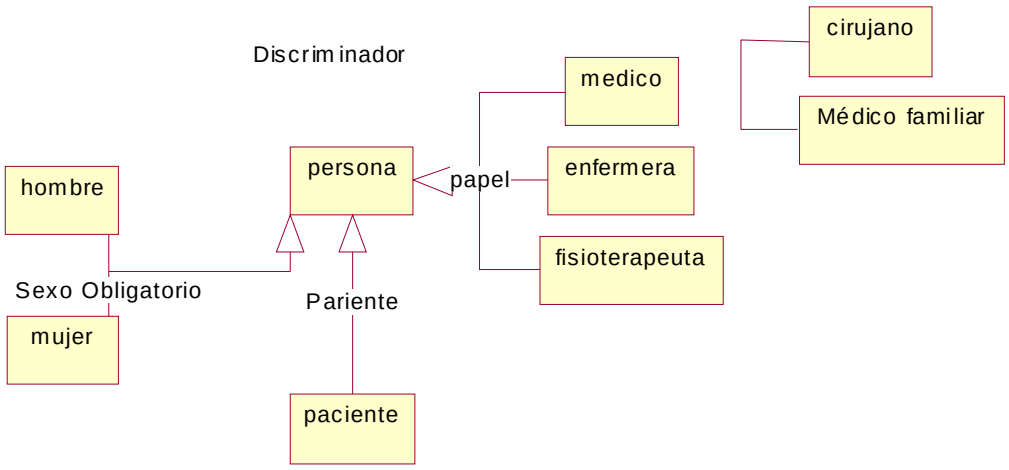


Figura 3.17 **Clasificación múltiple**

Las restricción útil (pero no estandar en UML) consiste en marcar al discriminador como obligatorio. Esto significa que cualquier instancia de superclase debe ser instancia de una de las subclases del grupo (por tanto la superclase es abstracta).

Para ilustrar obsérvense las siguientes combinaciones legales de subtipos en el diagrama: (Mujer, Paciente, Enfermera); (Hombre, Fisioterapeuta); (Mujer, Paciente); y

(Mujer, Médico, Cirujano). Obsérvense también que las combinaciones como (Paciente, Médico) y (Hombre, Médico, Enfermera) son ilegales: la primera, porque no incluyen un tipo del discriminador Sexo {Obligatorio}; la última, porque contiene dos tipos del discriminador Papel. La clasificación simple, por definición corresponde a un solo discriminador no etiquetado.

Otra pregunta que cabe plantearse es si un objeto puede cambiar de tipo. Un buen ejemplo para contestar es una cuenta bancaria. Cuando se sobregira la cuenta, cambia sustancialmente su conducta, específicamente se suplantan varias operaciones (incluyendo la de “retiro” y la de cierre).

La clasificación dinámica permite a los objetos cambiar de tipo dentro de la estructura de subtipificación; la clasificación estática, no En la clasificación estática se hace la distinción entre tipos y estados; en la clasificación dinámica estos dos conceptos.

Consideran algunos autores que es útil para el modelo conceptual. Se puede hacer con modelado por especificaciones; pero conviene sentirse cómodo con las técnicas necesarias para realizarlo. EL truco consiste en implementarlo de manera que parezca lo mismo que subclasificar desde la interfaz, de tal suerte que el usuario de una clase no sepa que implementación se está usando. Sin embargo, como en la mayoría de estas cosas, la elección depende de la circunstancias, así que se deberá aplicar lo que se juzgue mejor. La transformación de una interfaz múltiple y dinámica a un implementación estática única bien puede acarrear: más problemas de los que resolvería.

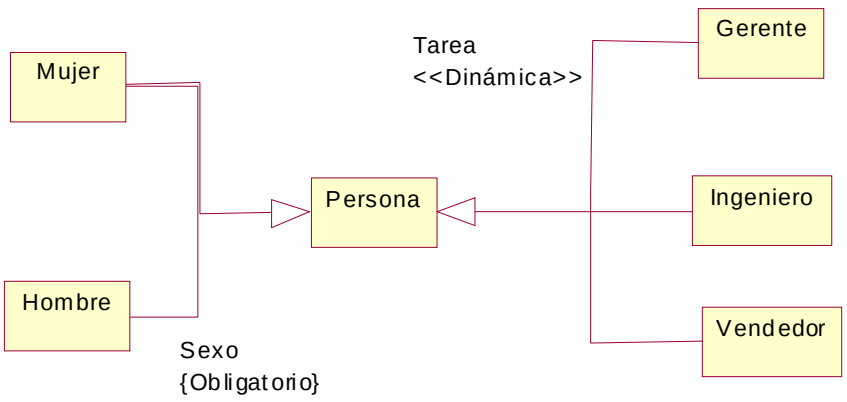


Figura 3.18 **Clasificación dinámica**

En la figura 3.18 muestra un ejemplo del empleo de la clasificación dinámica para el trabajo de una persona, que, por supuesto, puede cambiar. Esto puede ser lo apropiado, pero los subtipos necesitarían un comportamiento adicional, en lugar de ser sólo etiquetas. En estos casos, muchas veces vale la pena crear una clase separada para el trabajo y vincular a la persona con ella mediante una asociación.

11.25.5 Valores etiquetados.

Un *valor etiquetado* es una extensión de las propiedades de un elemento de UML, permitiendo añadir nueva información en la especificación de ese elemento. Gráficamente, un valor etiquetado se representa como una cadena de caracteres entre llaves colocada debajo del nombre de otro elemento. Esta cadena incluye un nombre (etiqueta), un separador (el símbolo =) y un valor de la etiqueta. Los valores etiquetados permiten añadir nuevas propiedades a un elemento de UML. Por ejemplo:

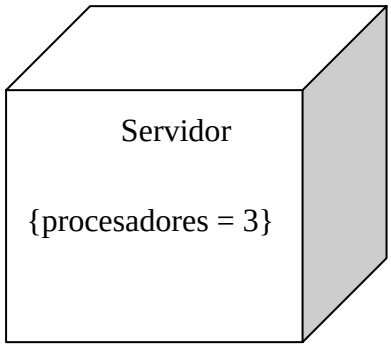


Figura 3.19 **Valores etiquetados**

11.25.6 Restricciones.

Una *restricción* es una extensión de la semántica de un elemento de UML, que permite añadir nuevas reglas o modificar las existentes. Gráficamente, una restricción se representa como una cadena de caracteres entre llaves colocada junto al elemento al que está asociada o conectada a ese elemento o elementos por relaciones de dependencia. Una restricción se puede representar en una nota.

Con las restricciones se puede añadir nueva semántica o modificar las reglas existentes. Una restricción especifica condiciones que deben cumplirse para que el modelo esté bien formado. Por ejemplo:

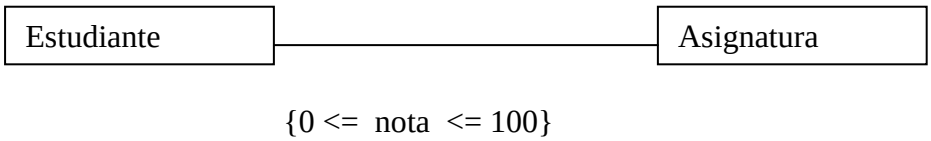


Figura 3.20 **Restricciones**

11.25.7 Diagramas.

Cuando se modela algo, se crea una simplificación de la realidad para comprender mejor el sistema que se está desarrollando. Con UML, se construyen modelos a partir de bloques de construcción básicos, tales como clases, interfaces, colaboraciones, componentes, nodos, dependencias, generalizaciones y asociaciones.

Los *diagramas* son los medios para ver estos bloques de construcción. Un diagrama es una representación gráfica de un conjunto de elementos, que la mayoría de las veces se dibuja como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se utilizan para visualizar un sistema desde diferentes perspectivas y dado que ningún sistema puede ser comprendido completamente desde una única perspectiva, UML define varios diagramas que permiten centrarse en diferentes aspectos del sistema.

Un sistema representa la cosa que se está desarrollando, vista desde diferentes perspectivas mediante diferentes modelos, y con esas vistas presentadas en forma de

diagramas. Normalmente, las partes estáticas de un sistema se representarán mediante uno de los cuatro diagramas siguientes:

Diagramas de clases.

Diagramas de objetos.

Diagramas de componentes.

Diagramas de despliegue.

Se emplearán cinco diagramas adicionales para ver las partes dinámicas de un sistema:

Diagramas de casos de uso.

Diagramas de secuencia.

Diagramas de colaboración.

Diagramas de estados.

Diagramas de actividades.

UML define estos nueve tipos de diagramas.

11.25.8 Diagramas estructurales.

Los cuatro diagramas estructurales de UML existen para visualizar, especificar, construir y documentar los aspectos estáticos de un sistema. Los aspectos estáticos de un sistema software incluyen la existencia y ubicación de clases, interfaces, colaboraciones, componentes y nodos (elementos estables del sistema). Estos cuatro diagramas son los siguientes:

Diagramas de clases: un diagrama de clases presenta un conjunto de clases, interfaces y colaboraciones, y las relaciones entre ellas. Los diagramas de clases son los diagramas más comunes en el modelado de sistemas orientado a objetos y se utilizan para describir la vista de diseño estática de un sistema.

Diagramas de objetos: un diagrama de objetos representa un conjunto de objetos y sus relaciones. Se utilizan para describir estructuras de datos, instantáneas de las instancias de los elementos encontrados en los diagramas de clases. Los diagramas de objetos, al igual que los diagramas de clases, cubren la vista de diseño estática del sistema, pero desde el punto de vista de casos reales.

Diagramas de componentes: un diagrama de componentes muestra un conjunto de componentes y sus relaciones, se utilizan para describir la vista de implementación estática de un sistema. Un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones.

Diagramas de despliegue: un diagrama de despliegue muestra un conjunto de nodos y sus relaciones, se utilizan para describir la vista de despliegue estática de una arquitectura. Un nodo normalmente incluye uno o más componentes.

11.25.9 Diagramas de comportamiento.

Los cinco diagramas de comportamiento de UML se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema. Se pueden ver los aspectos dinámicos de un sistema como aquellos que representan sus partes cambiantes. Estos cinco diagramas se organizan de acuerdo a las formas principales en que se puede modelar la dinámica de un sistema:

- Diagramas de casos de uso: organiza los comportamientos del sistema. Un diagrama de casos de uso representa un conjunto de casos de uso y actores (un tipo especial de clases) y sus relaciones.
- Diagramas de secuencia: centrados en la ordenación temporal de los mensajes. Un diagrama de secuencia es un diagrama de interacción que resalta la ordenación temporal de los mensajes, presenta un conjunto de objetos y los mensajes enviados y recibidos por ellos. Los diagramas de secuencia se utilizan para describir la vista dinámica de un sistema.
- Diagramas de colaboración: un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes. Un diagrama de colaboración muestra un conjunto de objetos, enlaces entre esos objetos y mensajes enviados y recibidos por esos objetos.
- Diagrama de estados: un diagrama de estados representa una máquina de estados, constituida por estados, transiciones, eventos y actividades. Son especialmente importantes para modelar el comportamiento de una interfaz, una clase o una colaboración. Los diagramas de estados resaltan el comportamiento dirigido por eventos de un objeto, lo que es especialmente útil al modelar sistemas reactivos.
- Diagramas de actividades: un diagrama de actividades muestra el flujo de actividades de un sistema. Una actividad muestra un conjunto de actividades, el flujo secuencial o ramificado de actividades, y los objetos que actúan y sobre los que se actúa. Son importantes para modelar la función de un sistema, así como para resaltar el flujo de control entre objetos.

11.25.10 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulos 6 y 7: Mecanismos comunes y diagramas. Páginas 65-91. Expone de manera detallada los principales mecanismos comunes y diagramas.

UML Gota a Gota. Martín Fowler con Kendall Scott, Editorial Person. Capítulo 5 Diagramas de Clase Conceptos Avanzados Pagina 86- 87

11.26 CLASES 28 y 29: DIAGRAMAS DE CLASES Y DIAGRAMAS DE OBJETOS.

Diagramas de clases.

Diagramas de objetos.

OBJETIVO:

Dar a conocer la importancia que juegan los diagramas de clases y de objetos en el modelado de sistemas orientados a objetos.

11.26.1 Diagramas de clases.

Un *diagrama de clases* es un diagrama que muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Son los más utilizados en el modelado de sistemas orientados a objetos. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema.

11.26.2 Propiedades comunes.

Un diagrama de clases es un tipo especial de diagrama y comparte las propiedades comunes al resto de los diagramas (un nombre y un contenido gráfico que es una proyección de un modelo). Lo que distingue a un diagrama de clases de los otros tipos es su contenido particular.

11.26.3 Contenido.

Los diagramas de clases contienen normalmente los siguientes elementos:

- Clases.
- Interfaces.
- Colaboraciones.
- Relaciones de dependencia, generalización y asociación.

Al igual que los demás diagramas, los diagramas de clases pueden contener notas y restricciones. También pueden contener paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes.

Un ejemplo de un diagrama de clases para nuestro problema del Intae – León – Inatec podría ser el siguiente:

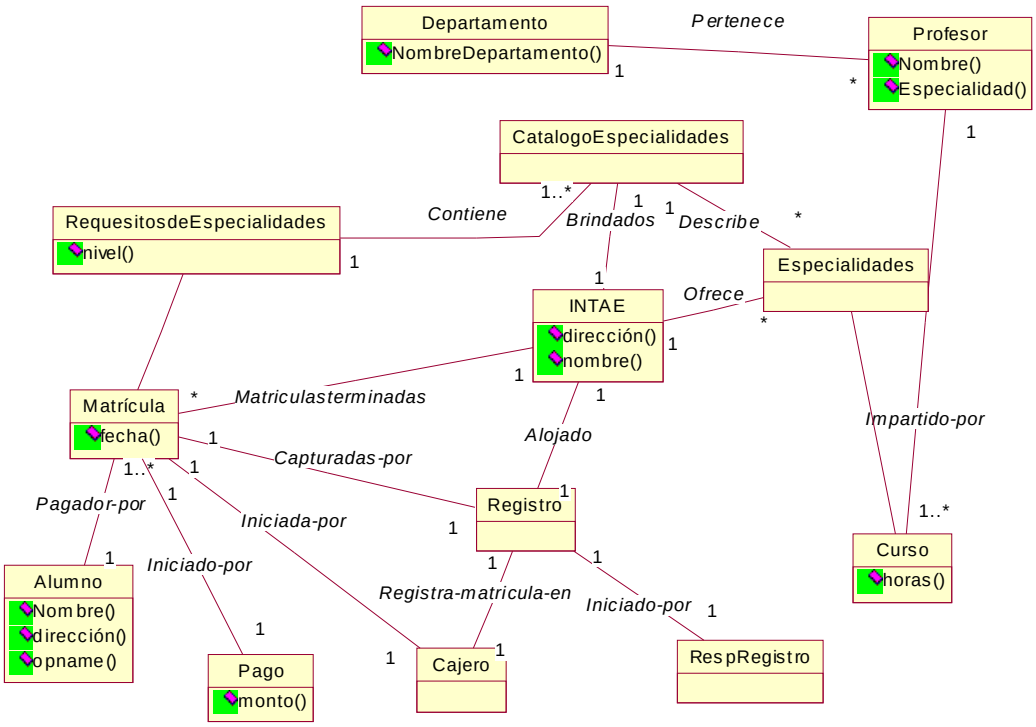


Figura 3.21 **Diagrama de clases**

11.26.4 Diagrama de objetos.

Un *diagrama de objetos* es un diagrama que representa un conjunto de objetos y sus relaciones en un momento concreto. Gráficamente, un diagrama de objetos es una colección de nodos y arcos. Estos diagramas modelan las instancias de los elementos contenidos en los diagramas de clases en un instante de tiempo determinado.

11.26.5 Propiedades comunes.

Un diagrama de objetos es un tipo especial de diagrama y comparte las propiedades comunes al resto de diagramas (un nombre y un contenido gráfico que es una proyección de un modelo). Lo que distingue a un diagrama de objetos de los otros tipos de diagramas es su contenido particular.

11.26.6 Contenido.

Los diagramas de objetos normalmente contienen:

Objetos.

Enlaces.

Al igual que los demás diagramas, los diagramas de objetos pueden contener notas y restricciones. También pueden contener paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes. Por ejemplo:

11.26.7 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulos 8 y 14: Diagramas de clases y Diagramas de objetos. Páginas 93-102, 169-175. Presenta de manera resumida las principales características de los diagramas de clases y de objetos.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 87-117. Explica de manera amplia las características de los diagramas de clases y relaciones.

UML Distilled “Applyng the standard Object Modeling Language”. Martin Fowler. Editorial Addison Wesley. Capítulo 6: Class Diagrams: The Essentials. Páginas 53-74. Resume los elementos esenciales de un diagrama de clases.

12 UNIDAD IV: MODELADO ESTRUCTURAL AVANZADO

12.1 Temas a abordar

- Características avanzadas de las clases.
- Características avanzadas de las relaciones.
- Paquetes.
- Instancias.

Esta unidad será impartida en cuatro sesiones.

12.2 CLASES 30,31 y 32: CARACTERÍSTICAS AVANZADAS DE LAS CLASES

Introducción, Términos y conceptos, Clasificadores

Visibilidad, Elementos abstractos, raíces, hojas y polimórficos

Multiplicidad, Atributos, Operaciones

Objetivo:

Aplicar características avanzadas como la visibilidad y multiplicidad en atributos y operaciones de una clase.

12.2.1 Introducción.

Un **clasificador** es un mecanismo que describe características estructurales y de comportamiento. Los clasificadores incluyen clases, interfaces, tipos de datos, señales, componentes, nodos, casos de uso y subsistemas.

Los clasificadores (y especialmente las clases) tienen varias características avanzadas aparte de los atributos y operaciones, se puede modelar la multiplicidad, visibilidad, signatura, polimorfismo y otras características.

12.2.2 Clasificadores.

Algunos elementos de UML no tienen instancias, por ejemplo, los paquetes y las relaciones de generalización. *Aquellos elementos de modelado que pueden tener instancias se llaman* ***clasificadores*** (las asociaciones y los mensajes también pueden tener instancias, pero sus instancias no son como las de una clase). Un clasificador tiene características estructurales (en forma de atributo), así como características de comportamiento (en forma de operaciones). Cada instancia de un clasificador determinado comparte las mismas características.

El tipo de clasificador más importante en UML es la clase. Una ***clase*** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Sin embargo, las clases no son el único tipo de clasificador. UML proporciona otros tipos de clasificadores para ayudar a modelar:

Interfaz Una colección de operaciones que especifican un servicio de una clase o componente.

Tipo de datos Un tipo cuyos valores no tienen identidad, incluyendo los tipos primitivos predefinidos (como números y cadenas de caracteres), así como los tipos enumerados (como los booleanos).

Señal La especificación de un estímulo asíncrono enviado entre instancias.

Componente Una parte física y reemplazable de un sistema que conforma y proporciona la realización de un conjunto de interfaces.

Nodo Un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, generalmente con alguna memoria y a menudo capacidad de procesamiento.

Caso de uso Descripción de una secuencia de acciones, incluyendo variantes, que ejecuta un sistema y produce un resultado observable para un actor particular.

Subsistema Agrupación de elementos, algunos de los cuales constituyen una especificación del comportamiento de los otros elementos contenidos.

La mayoría de los distintos clasificadores tiene tanto características estructurales como de comportamiento (excepto las interfaces que no pueden tener atributos). Cuando se modela con cualquiera de estos clasificadores, se pueden usar todas las características avanzadas que vamos a estudiar, para proporcionar el nivel de detalle necesario para capturar el significado de la abstracción.

Gráficamente, UML distingue entre los diferentes clasificadores, como se muestra a continuación:

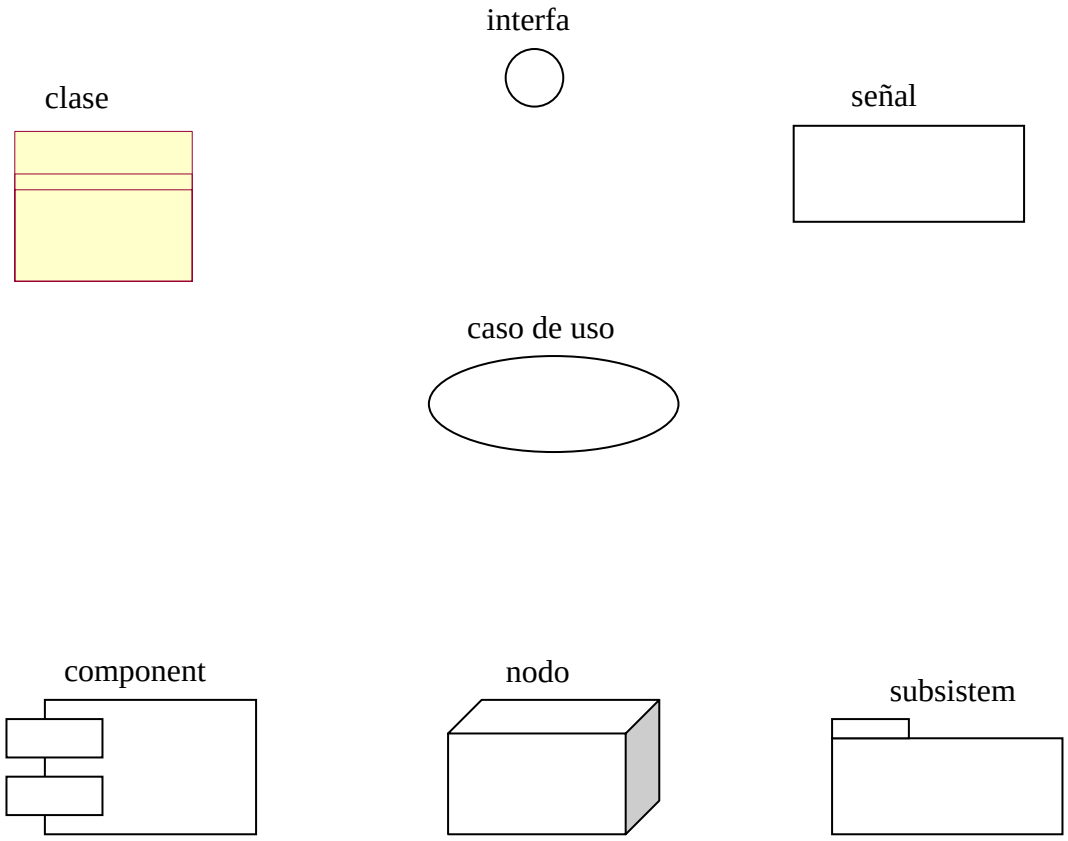


Figura 4.1 **Clasificadores**

12.2.3 Visibilidad.

Uno de los detalles más importantes que se puede especificar para los atributos y operaciones de un clasificador es su visibilidad. La visibilidad de una característica específica sí puede ser utilizada por otros clasificadores. En UML, se puede especificar una característica con cualquiera de los tres niveles de visibilidad disponibles.

Public Cualquier clasificador externo con visibilidad hacia el clasificador dado puede utilizar la característica; se especifica precediéndola del símbolo **+**

Protected Cualquier descendiente del clasificador puede utilizar la característica; se especifica precediéndola del símbolo **#**

Private Sólo el propio clasificador puede utilizar la característica; se especifica precediéndola del símbolo **–**

Si no se adorna explícitamente una característica con un símbolo de visibilidad, por lo general se puede asumir que es pública.

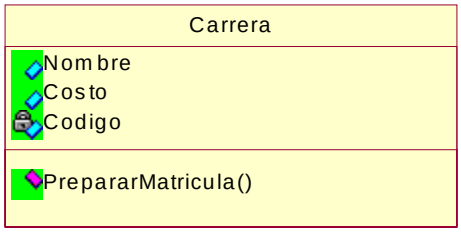


Figura 4.2 **Visibilidad**

12.2.4 Elementos abstractos, raíces, hojas y polimórficos.

La relación de generalización se utiliza para modelar una jerarquía de clases, con las abstracciones más generales en la cima y las más específicas en el fondo. Dentro de estas jerarquías es frecuente especificar que ciertas clases son **abstractas** (es decir, que no pueden tener instancias directas). En UML, se especifica que una clase es abstracta escribiendo su nombre en cursiva. Por ejemplo, como se muestra en la siguiente figura, *Figura* es una clase abstracta. Por otro lado, una **clase concreta** (como *Pentágono*) es aquella que puede tener instancias directas.

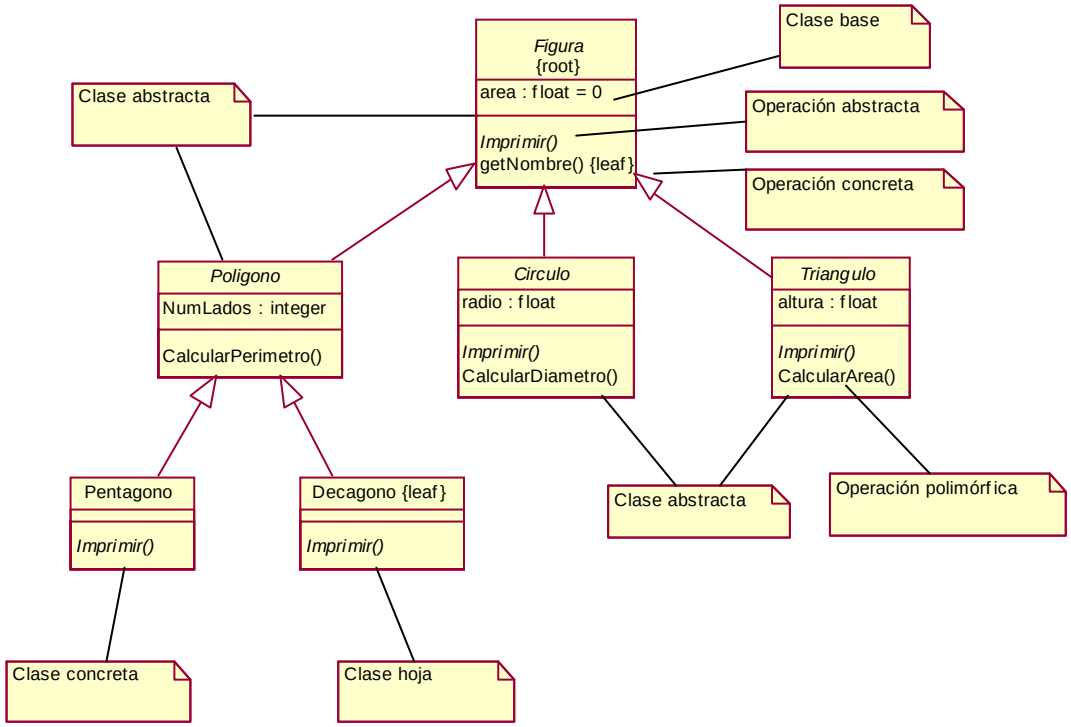


Figura 4.3 **Clases, operaciones abstractas y concretas**

Cuando se usa una clase, es probable que se desee heredar características de otras clases más generales, y también permitir que otras clases más específicas hereden características de ella. Sin embargo, también se puede especificar que una clase no puede tener hijos. Ese elemento se llama **clase hoja** y se especifica en UML escribiendo la propiedad **leaf** bajo el nombre de la clase. En la figura anterior, *Decágono* es una clase hoja, así que no puede tener hijos.

Cuando se especifica que una clase no puede tener padres, a esta clase se le llama **clase raíz** (ejemplo, la clase *Figura*) y en UML se indica escribiendo la propiedad **root** bajo el nombre de la clase. Especialmente cuando se tienen varias jerarquías de herencia independientes, es útil diseñar la cabeza de cada jerarquía de esta manera.

Las operaciones tienen propiedades similares. Normalmente, una operación es **polimórfica**, lo que significa que, en una jerarquía de clases, se pueden especificar operaciones con la misma signatura en diferentes puntos de la jerarquía. Las operaciones de las clases hijas redefinen el comportamiento de las operaciones de las clases padre. Cuando se envía un mensaje en tiempo de ejecución, la operación de la jerarquía que se invoca se elige polimórficamente (es decir, el tipo del objeto receptor en tiempo de ejecución determina la elección). En la figura anterior, *CalcularArea()* es una operación polimórmica. Una **operación abstracta**, es una operación incompleta, que necesita que una clase hija proporcione una implementación. En UML se especifica una operación abstracta escribiendo su nombre en cursiva, igual que se hace con una clase. Una **operación hoja,** es

señalada así por la propiedad **leaf**. Esto significa que la operación no es polimórfica y no puede ser redefinida.

12.2.5 Multiplicidad.

A veces, se desea restringir el número de instancias que una clase puede tener. Lo más frecuente es que se desee especificar cero instancias (en cuyo caso la clase es una clase utilidad que hace públicos sólo atributos y operaciones con alcance de clase), una única instancia (una clase unitaria o singleton), un número específico de instancias o muchas instancias (el valor por omisión).

El número de instancias que puede tener una clase es su **multiplicidad**. Esta consiste en una especificación del rango de cardinalidades permitidas que puede asumir una entidad. En UML se puede especificar la multiplicidad de una clase con una expresión en la esquina superior derecha del icono de la clase. En la figura, hay exactamente dos instancias de la clase *Producto*.

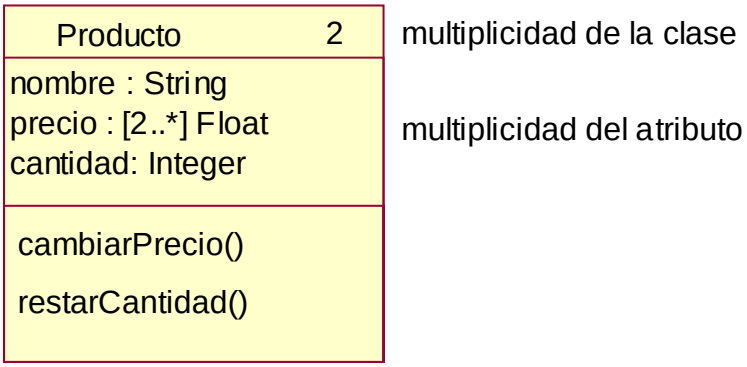


Figura 4.4 **Multiplicidad**

La multiplicidad también se aplica a los atributos. Se puede especificar la multiplicidad de un atributo mediante una expresión adecuada encerrada entre corchetes tras el nombre del atributo. En la figura anterior, hay una o más instancias de *Precio* en la instancia de *Producto*.

12.2.6 Atributos.

Generalmente, al modelar las características estructurales de una clase (sus atributos), simplemente se escribe el nombre de cada atributo, aunque también se puede especificar la visibilidad, el alcance y la multiplicidad de cada atributo. Además se puede especificar el tipo, el valor inicial y los cambios posibles de cada uno de ellos.

En su forma completa, la sintaxis de un atributo en UML es:

[visibilidad] nombre [multiplicidad] [: tipo] [= valor inicial] [{propiedades}]

Las siguientes son declaraciones legales de atributos:

autor Sólo el nombre

+autor	Visibilidad y nombre
cantidad:Entero	Nombre y tipo
filas[1..n] : Integer	Nombre, multiplicidad y tipo
anyo: Integer = 1997	Nombre, tipo y valor inicial

12.2.7 Operaciones.

Al nivel más abstracto, al modelar las características de comportamiento de una clase (sus operaciones y sus señales), simplemente se escribirá el nombre de cada operación. También se puede especificar la visibilidad y el alcance de cada operación. Además, podemos especificar los parámetros, el tipo de retorno, la semántica de concurrencia y otras propiedades de cada operación. El nombre de una operación junto a sus parámetros (incluido el tipo de retorno) se conoce como **signatura de la operación**.

UML distingue entre operación y método. Una operación especifica un servicio que se puede requerir de cualquier objeto de la clase para influir en su comportamiento; un método es una implementación de una operación. Cada operación no abstracta de una clase debe tener un método, el cual proporciona un algoritmo ejecutable como cuerpo (normalmente en algún lenguaje de programación o como texto estructurado). En una jerarquía de herencia, puede haber varios métodos para la misma operación, y el polimorfismo selecciona qué método de la jerarquía se ejecuta en tiempo de ejecución.

12.2.8 En su forma completa, la sintaxis de una operación en UML es:

[visibilidad] nombre [(lista de parámetros)] [: tipo de retorno] [{propiedades}]

12.2.9 Las siguientes son declaraciones legales de operaciones:

leerMatriz	Sólo el nombre
+leerMatriz	Visibilidad y nombre
asignarAutor(aut: String)	Nombre y parámetros
obtenerNroDeVolumen( ): Integer	Nombre y tipo de retorno

En la signatura de una operación se pueden proporcionar cero o más parámetros, donde cada uno sigue la siguiente sintaxis:

[dirección] nombre: tipo [= valor por defecto]

Dirección puede tomar uno de los siguientes valores:

in	Parámetro de entrada; no se puede modificar.
out	Parámetro de salida; puede modificarse para comunicar información al invocador.
inout	Parámetro de entrada; puede ser modificado.

12.2.10 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 9: Características avanzadas de las clases. Páginas 105-118. Aborda una serie de conceptos avanzados que tienen que ver con el uso de las clases, como por ejemplo clases abstractas y polimorfismo, explicando su notación en UML.

12.3 Clase 33: Características avanzadas de las relaciones

Introducción, Términos y conceptos, Dependencia, Generalización

Asociación, Navegación, Visibilidad, Composición, Clases asociación

OBJETIVO:

Identificar características avanzadas como herencia múltiple, navegación y composición, para aplicarlas en la creación de relaciones entre clases.

12.3.1 Introducción.

Las **dependencias,** las **generalizaciones** y las **asociaciones** son los tres bloques de construcción de relaciones más importantes de UML. Estas relaciones tienen varias propiedades como la herencia múltiple, navegación, composición y otras características. Un cuarto tipo de relación, **la realización**, permite modelar la conexión entre una interfaz y una clase o componente, o entre un caso de uso y una colaboración.

Las dependencias, las generalizaciones y las asociaciones son las relaciones más comunes que aparecen al modelar sistemas con gran cantidad de software. Sin embargo, se necesitan varias características avanzadas de estas relaciones para capturar los detalles de muchos sistemas, detalles importantes de considerar para evitar verdaderos fallos en el diseño.

UML proporciona una representación para varias características avanzadas. Esta notación permite visualizar, especificar, construir y documentar redes de relaciones al nivel de detalle que se desee.

12.3.2 Términos y conceptos.

*Una **relación** es una conexión entre elementos*. En el modelado orientado a objetos, los cuatro tipos más importantes de relaciones son las dependencias, las generalizaciones, las asociaciones y las realizaciones.

Gráficamente, una relación se dibuja como una línea, con diferentes tipos de líneas para distinguir las diferentes relaciones.

12.3.3 Dependencia.

Una **dependencia** es una relación de uso, la cual especifica que un cambio en la especificación de un elemento (por ejemplo, la clase `Ingrediente`) puede afectar a otro elemento que lo utiliza (por ejemplo, la clase `Producto`), pero no necesariamente a la inversa. Gráficamente, una dependencia se representa como una línea discontinua, dirigida hacia el elemento del que se depende. Las dependencias se deben aplicar cuando se quiera representar que un elemento utiliza a otro.

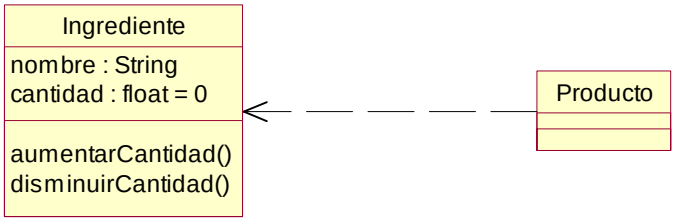


Figura 4.5 **Dependencia**

UML define varios estereotipos que pueden aplicarse a las dependencias. Mencionaremos algunos de ellos, clasificados según el grupo donde se utilicen.

Los siguientes cinco estereotipos se usan en las dependencias entre clases y objetos en los diagramas de clases:

derive Especifica que el origen puede calcularse a partir del destino.

Se utilizará `derive` cuando se desee modelar las relaciones entre dos atributos dos asociaciones, uno de los cuales sea concreto y el otro sea conceptual. Por ejemplo, na clase `Empleado` podría tener el atributo `fechaIngreso(`concreto), así como el tributo `antigüedad` (que se puede derivar de `fechaIngreso`, así que no se anifiesta de forma separada en la clase). Se podría mostrar la relación entre `ntigüedad` y f`echaIngreso` como una dependencia `derived,` donde `antigüedad` derive de `fechaIngreso.`

friend Especifica que el origen tiene una visibilidad especial en el destino.

Se utilizará `friend` para modelar relaciones como las clases amigas (friend) de C++.

instanceOf Especifica que el objeto origen es una instancia del clasificador

destino.

instantiate Especifica que el origen crea instancias del destino.

Estos dos últimos estereotipos permiten modelar relaciones clase/objeto explícitamente. Se utilizará `instanceOf` para modelar la relación entre una clase y un objeto en el mismo diagrama, o entre una clase y su metaclase. Se empleará `instantiate` para especificar qué elemento crea objetos de otro.

Use Especifica que la semántica del elemento origen depende de la semántica de la parte pública del destino.

Se aplicará `use` cuando se quiera etiquetar explícitamente una dependencia como una relación de uso.

Dos estereotipos se aplican a las relaciones de dependencias entre casos de uso:

extend Especifica que el caso de uso destino extiende el comportamiento del origen.

Include Especifica que el caso de uso origen incorpora explícitamente el

comportamiento de otro caso de uso en la posición especificada por el origen.

Se utilizará `extend` e `include(`y generalización simple) para descomponer los casos de uso en partes reutilizables.

12.3.4 Un estereotipo que aparece en el contexto de las máquinas de estados

send Especifica que la operación origen envía el evento destino.

Se utilizará `send` cuando se quiera modelar una operación (como la que puede aparecer en la acción asociada a una transición de estado) que envía un evento dado a un objeto destino (que a su vez puede tener una máquina de estados asociada). La dependencia `send` permite ligar máquinas de estados independientes.

12.3.5 Generalización.

Una *generalización* es una relación entre un elemento general (superclase o padre) y un tipo más específico de ese elemento (subclase o hijo). Por ejemplo, se puede encontrar la clase general *Animal* junto a un tipo más específico, *Ave*. Con una relación de generalización del hijo al padre, el hijo (*Ave*) heredará la estructura y comportamiento del padre (*Animal*).

El hijo puede añadir nueva estructura y comportamiento, o modificar el comportamiento del padre. En una generalización, las instancias del hijo pueden usarse donde quiera que se puedan usar las instancias del padre (o sea, el hijo es un sustituto del padre).

Una clase que tenga únicamente un padre utiliza *herencia simple*. A veces, la herencia múltiple es mejor, y también se pueden modelar estas relaciones en UML.

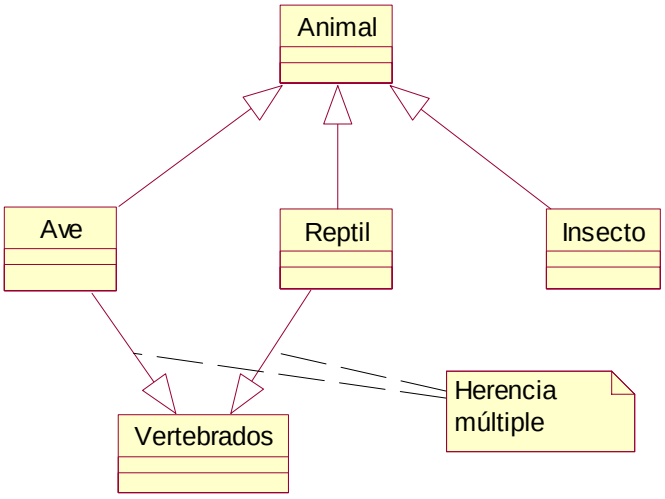


Figura 4.6 **Generalización**

En la figurar anterior, se muestra la clase `Animal`, con tres hijos, uno de los cuales es `Ave`, el cual, a su vez es hijo de Vertebrados. Los padres como Animal y Vertebrados,

son llamados mixins porque no se encuentran aislados, si no que están destinados a combinarse con otros padres para formar hijos.

Una generalización simple, es suficiente para la mayoría de las relaciones de herencia que aparecen en el modelado. Pero si se quieren especificar ciertos matices, UML define un estereotipo que puede aplicarse a las generalizaciones.

El estereotipo es:

implementation Especifica que el hijo hereda la implementación del padre, pero no hace públicas ni soporta sus interfaces.

Se usará ***implementation*** cuando se quiera modelar la herencia privada, como la que aparece en C++.

12.3.6 Asociación.

Una **asociación** es una relación estructural que especifica que los objetos de un elemento se conectan a los objetos de otro.

Hay cuatro elementos básicos que se aplican a las asociaciones: **nombre, rol en cada extremo de la asociación, multiplicidad en cada extremo y agregación.** Hay otras propiedades que permiten modelar detalles sutiles, como la **navegación** y algunas variantes de la **agregación**.

12.3.7 Navegación.

Dada una asociación simple entre dos clases, como *Cliente y Reservación*, es posible navegar de los objetos de un tipo a los del otro. A menos que se indique lo contrario, la navegación a través de una asociación es bidireccional. Pero hay ciertas circunstancias en las que se desea limitar la navegación a una sola dirección. Se puede representar de forma explícita la dirección de la navegación con una flecha que apunte en la dirección de recorrido.

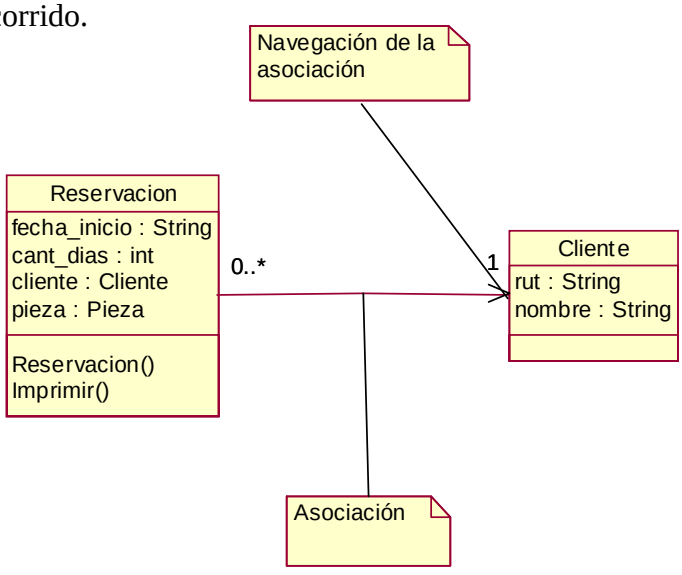


Figura 4.7 **Navegación**

Especificar que una asociación es navegable es una afirmación de que, dado un objeto en un extremo, se puede llegar fácil y directamente a los objetos del otro extremo, normalmente debido a que el objeto inicial almacena algunas referencias a los objetos en el destino.

12.3.8 Visibilidad.

Dada una asociación entre dos clases, los objetos de una clase pueden ver y navegar hasta los objetos de la otra, a menos que se restrinja por un enunciado explícito de navegación. Sin embargo, hay circunstancias en las que se quiere limitar la visibilidad a través de esta asociación relativa a los objetos externos a ella. En UML se pueden especificar tres niveles de visibilidad para el extremo de una asociación, igual que para las características de una clase, adjuntando un símbolo de visibilidad al nombre de un rol. A menos que se indique lo contrario, la visibilidad de un rol es **pública**. La **visibilidad privada** indica que los objetos de ese extremo no son accesibles a ningún objeto externo a la asociación; la **visibilidad protegida** indica que los objetos de ese extremo no son accesibles a ningún objeto externo a la asociación, excepto los hijos del otro extremo.

12.3.9 Composición.

La agregación es un concepto simple con una semántica bastante profunda. La agregación simple es completamente conceptual y no hace más que distinguir un "todo" de una "parte". La agregación simple no cambia el significado de la navegación a través de la asociación entre el todo y sus partes, ni liga las vidas del todo y las partes.

Existe una variación de la agregación simple, la composición. La **composición** es una forma de agregación, con una fuerte relación de pertenencia y vidas coincidentes de la parte con el todo. La "parte" puede crearse después de la parte compuesta a la que pertenece, pero una vez creada vive y muere con ella. Tal parte, también se puede eliminar explícitamente antes de la eliminación de la parte compuesta.

En una agregación compuesta, un objeto puede formar parte de sólo una parte compuesta a la vez. Por ejemplo, un *Techo* únicamente pertenece a una *Vivienda*. Esto contrasta con la agregación simple, en la que una parte se puede compartir por varios agregados. En una agregación compuesta, la parte compuesta es responsable de gestionar la creación y destrucción de las partes. Por ejemplo, al crearse un objeto de tipo *Techo*, debe asignarse a una *Vivienda.*

La composición es un tipo especial de asociación y se especifica adornando una asociación simple con un rombo relleno en el extremo del todo.

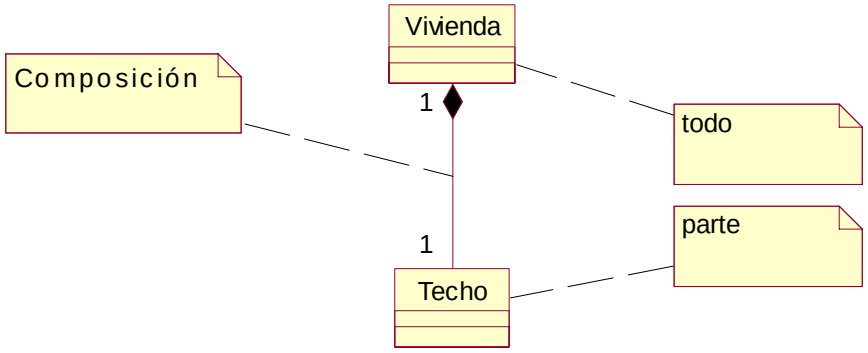


Figura 4.8 **Composición**

12.3.10 4.4.3.4 Clases asociación.

En una asociación entre dos clases, la propia asociación puede tener propiedades. Por ejemplo, en una relación *persona/automóvil* entre una *Persona* y un *Automóvil*, hay una *Factura* que representa las propiedades de esa relación y que se aplican exactamente a un par de *Persona* y *Automóvil*. No sería apropiado modelar esta situación con una asociación de *Persona* a *Factura*, junto con una asociación de *Factura* a *Automóvil*, ya que esto no ligaría una instancia específica de *Factura* al par específico de *Persona* y *Automóvil*.

Para dar solución a esto, se utiliza una **clase asociación,** la cual es un elemento de modelado con propiedades tanto de asociación como de clase. Una clase asociación puede verse como una asociación que también tiene propiedades de clase, o una clase que también tiene propiedades de asociación. Una clase asociación se dibuja con un símbolo de clase unido por una línea discontinua a una asociación.

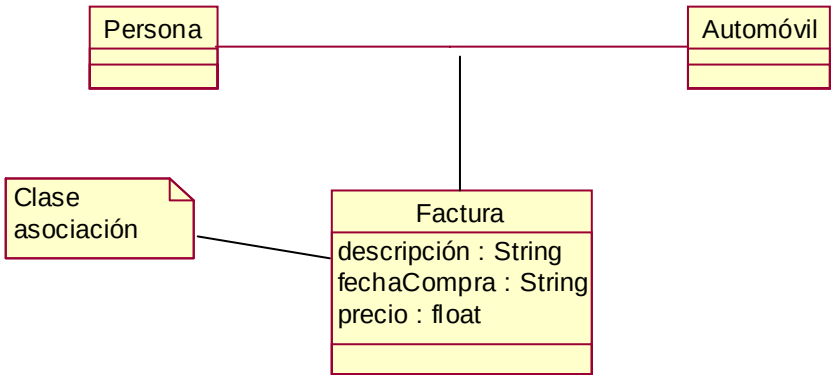


Figura 4.9 **Clases asociación**

Una clase asociación no se puede conectar a más de una asociación, ya que una clase asociación es la propia asociación.

12.3.11 Bibliografía comentada

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 10: Características avanzadas de las relaciones. Páginas 119-133. Explica los aspectos avanzados de las relaciones con muchos ejemplos.

12.4 Clase 34: Paquetes

Introducción, Términos y conceptos, Nombres, Elementos contenidos

Visibilidad, Generalización, Modelado de grupos de elementos

OBJETIVO:

Hacer uso de los paquetes, para organizar elementos de modelado en grupos a los que se puede dar un nombre y manejar como un conjunto.

12.4.1 Introducción.

Una de las preguntas más antiguas en los métodos de software es: ¿cómo se puede fragmentar un sistema grande en sistemas más pequeños? Se pregunta esto porque, en la medida en que los sistemas se hace más grande, se vuelve más difícile comprender, así como entender sus cambios.

Los métodos estructurados se valieron de la descomposición funcional, en la cual el sistema en su conjunto se correlacionaba como función y se dividía es subfunciones, que a su vez se dividían en otras sub funciones , y así sucesivamente. Las funciones representaban algo que hacía el sistema como un todo.

Eran los días en que el proceso y los datos estaban separados. De tal modo que, además de una descomposición funcional, también había una estructura de datos. Esta última ocupaba el segundo lugar, aunque ciertas técnicas de ingeniería de información agrupaba los riesgos de datos en áreas temáticas y producían matrices que mostraban la interrelación entre las funciones y los registros de datos.

Es desde este punto de vista que podemos apreciar el gran cambio que han significado los objetos. Ha desaparecido la separación entre el proceso y los datos, y la descomposición funcional, pero la vieja pregunta en pie. Una idea es agrupar las clases en unidades de nivel más alto. Esta idea aparece, aplicada de manera muy libre, en muchos métodos orientados a objetos. En el UML, a este mecanismo de agrupamiento se le llama paquetes.

La idea de un paquete se puede aplicar a cualquier elemento de un modelo, no sólo a las clases. Sin cierta heurística que agrupe las clases, el agrupamiento se vuelve arbitrario. El se ha encontrado más útil y que recibe más énfasis en el UML, es la dependencia. Se emplea el término diagrama de paquetes para indicar un diagrama que muestra los paquetes de clases y las dependencias entre ellos.

Visualizar, especificar, construir y documentar grandes sistemas conlleva manejar una gran cantidad de clases, interfaces, componentes, nodos, diagramas y otros elementos. Conforme va creciendo el sistema hasta alcanzar un gran tamaño, se hace necesario organizar estos elementos en bloques mayores.

Los paquetes se utilizan para organizar los elementos de modelado en partes mayores que se pueden manipular como un grupo. La visibilidad de estos elementos puede controlarse para que algunos sean visibles fuera del paquete mientras que otros permanezcan ocultos. Los paquetes bien diseñados agrupan elementos cercanos semánticamente y que suelen cambiar juntos.

12.4.2 Términos y conceptos.

Un **paquete** es un mecanismo de propósito general para organizar elementos en grupos. Los paquetes ayudan a organizar los elementos en los modelos con el fin de comprenderlos más fácilmente.

En UML un paquete se representa como una carpeta. Esta notación gráfica permite visualizar grupos de elementos que se pueden manipular como un todo y en una forma que permite controlar la visibilidad y el acceso a elementos individuales.

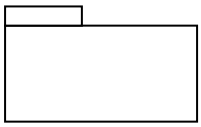


Figura 4.10 **Paquete**

Nombres.

Cada paquete debe tener un nombre que lo distinga de otros paquetes. Un ***nombre*** es una cadena de texto. El nombre solo se denomina ***nombre simple*****;** un ***nombre de camino*** consta del nombre del paquete precedido por el nombre del paquete en el que se encuentra, si este fuera el caso. Un paquete se dibuja normalmente mostrando sólo su nombre, aunque también se pueden dibujar paquetes adornados con valores etiquetados o con apartados adicionales para mostrar sus detalles.

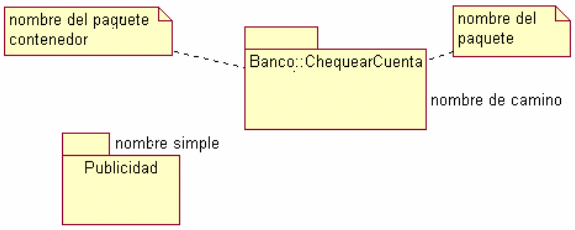


Figura 4.11 **Nombres de paquetes**

El nombre de un paquete puede ser texto con cualquier número de letras, dígitos y ciertos signos de puntuación (excepto signos como los dos puntos, usados para separar el nombre de un paquete del nombre de su paquete contenedor) y puede extenderse a lo largo

de varias líneas. En la práctica, los nombres de los paquetes son nombres cortos o expresiones nominales extraídos del vocabulario del modelo.

Elementos contenidos.

Un paquete puede contener otros elementos, incluyendo clases, interfaces, componentes, nodos, colaboraciones, casos de uso, diagramas e incluso otros paquetes. El elemento se declara en el paquete. Si el paquete se destruye, el elemento es destruido. Cada elemento pertenece exclusivamente a un único paquete.

Los elementos de la misma categoría deben tener nombres únicos en el contexto de su paquete contenedor. No se pueden tener dos clases llamadas ClientesHabituales dentro del mismo paquete, pero se puede tener una clase ClientesHabituales en el paquete ClientesLargoPlazo y otra clase diferente llamada ClientesHabituales en el paquete ClientesGenerales. Las clases Clientes*LargoPlazo::ClientesHabituales* y ClientesGenerales*::ClientesHabituales* son clases diferentes y se pueden distinguir por sus nombres de camino. Elementos de diferentes tipos pueden tener el mismo nombre dentro de un paquete, por ejemplo, podemos tener una clase y un componente llamados de la misma forma dentro del mismo paquete. En la práctica para evitar confusión, es mejor asociar a cada elemento un nombre único para todas las categorías dentro de un paquete.

Los paquetes pueden contener a otros paquetes. Por ejemplo, se puede tener una clase *CuentaDolares* en el paquete Cuentas que a su vez esté contenido en el paquete *Banco*. El nombre completo de la clase es Banco*::Cuentas::CuentaDolares*. En la práctica es mejor evitar paquetes muy anidados.

Se puede mostrar explícitamente el contenido de un paquete, bien textualmente, bien gráficamente.

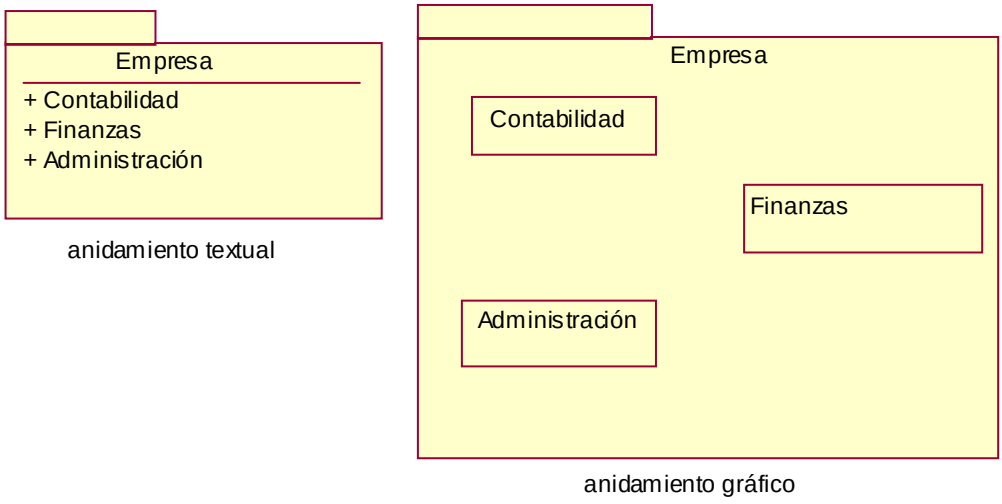


Figura 4.12 **Representaciones del contenido de un paquete**

12.4.3 Visibilidad.

Se puede controlar la visibilidad de los elementos contenidos en un paquete del mismo modo que se puede controlar la visibilidad de los atributos y operaciones de una clase. Normalmente, un elemento contenido en un paquete es *público*, es decir, es visible a los contenidos de cualquier paquete que importe al paquete contenedor del elemento. Por el contrario, los elementos *protegidos* sólo pueden ser vistos por los hijos y los elementos *privados* no son visibles fuera del paquete en el que se declaran. En la figura anterior, Contabilidad, Finanzas y Administración, son una parte pública del paquete Empresa. Un paquete que importe a Empresa verá a las tres clases. El nombre totalmente de Finanzas, por ejemplo, sería Empresa::Finanzas.

Para especificar la visibilidad de un elemento contenido en un paquete se antepone al nombre del elemento el símbolo de visibilidad apropiado. Los *elementos públicos* se muestran con su nombre precedido del símbolo +. *El conjunto de las partes públicas de un paquete constituye la* ***interfaz del paquete***.

Al igual que con las clases, se puede designar a un elemento como protegido o privado, con el nombre del elemento precedido del símbolo # o del símbolo -, respectivamente. Los elementos *protegidos* sólo son visibles para los paquetes que heredan de otro paquete; los elementos *privados* no son visibles para nadie fuera del paquete.

Los paquetes que son amigos de otro paquete pueden ver todos los elementos de éste, sin importar cuál sea su visibilidad.

Generalización.

Un tipo de relación que puede darse entre paquetes, es la *generalización,* que permite especificar familias de paquetes.

La generalización entre paquetes es muy parecida a la generalización entre clases. En el siguiente ejemplo, el paquete *Lenguaje* contiene dos clases (Operadores y Palabras reservadas). Los paquetes Datos Numéricos y Datos Alfanuméricos especializan al paquete más general (Lenguaje). Los paquetes especializados heredan los elementos públicos y protegidos del paquete más general (en caso que tuviera elementos protegidos). Al igual que con la herencia entre clases, los paquetes pueden reemplazar a los elementos más generales. El paquete Datos Numéricos hereda de Lenguaje, de forma que incluye las clases Lenguaje::Operadores y Lenguaje::Palabras reservadas. Datos Numéricos añade otras clases nuevas (Integer, Float, Double y Boolean).

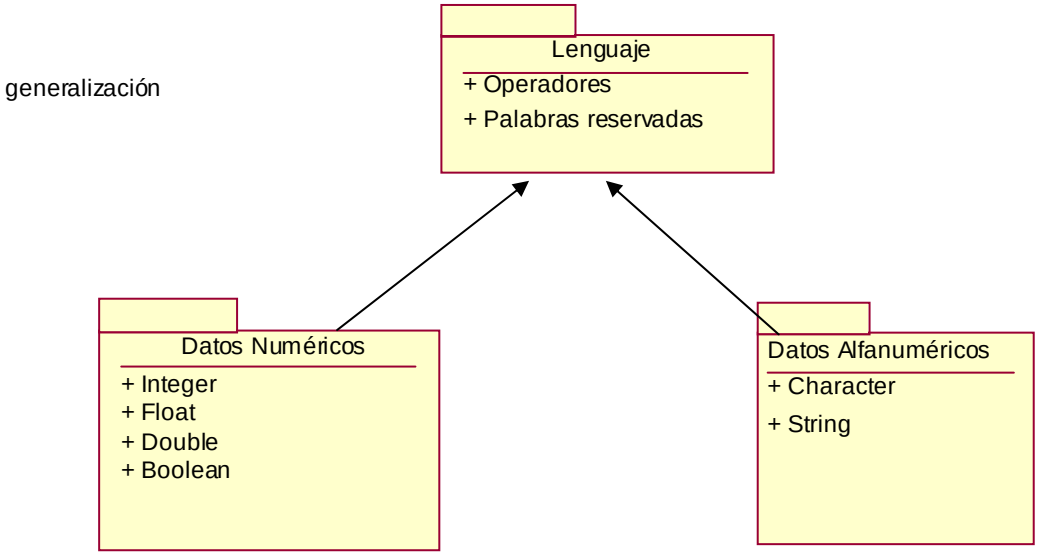


Figura 4.13 **Generalización entre paquetes**

Los paquetes implicados en las generalizaciones siguen el mismo principio de sustitución de las clases. Un paquete especializado puede usarse dondequiera que se use un paquete más general.

12.4.4 Modelado de grupos de elementos.

El objetivo más frecuente para el que se utilizan los paquetes, es organizar elementos de modelado en grupos a los que se puede dar un nombre y manejar como un conjunto. Estos grupos se modelan como paquetes.

Hay una distinción importante entre clases y paquetes: las clases son abstracciones de cosas encontradas en el problema o en la solución; los paquetes son los mecanismos que se emplean para organizar los elementos del modelo. Los paquetes no tienen identidad, es decir, no se pueden tener instancias de paquetes. Las clases tienen identidad (tienen instancias que son elementos de un sistema en ejecución).

La mayoría de las veces, los paquetes se utilizarán para agrupar el mismo tipo de elementos, aunque también se pueden emplear para agrupar diferentes tipos de elementos.

Para modelar grupos de elementos:

Hay que examinar los elementos de modelado de una determinada vista arquitectónica en busca de grupos definidos por elementos cercanos entre sí desde un punto de vista conceptual o semántico.

Hay que englobar cada uno de esos grupos en un paquete.

Para cada paquete, hay que distinguir los elementos que podrán ser accedidos desde fuera. Deben marcarse estos elementos como públicos, y los demás como protegidos o privados.

Hay que conectar explícitamente los paquetes que dependen de otros a través de dependencias de importación.

En el caso de familias de paquetes, hay que conectar los paquetes especializados con sus partes más generales por medio de especializaciones.

Si se tienen abstracciones que se manifiestan como objetos en el sistema real, no se deben utilizar paquetes. En vez de ello, se utilizarán elementos de modelado, tales como clases o componentes.

12.4.5 Cuando se dibuje un paquete en UML:

Hay que emplear la forma simple del icono de un paquete a menos que sea necesario revelar explícitamente el contenido.

Cuando se revele el contenido de un paquete, hay que mostrar sólo los elementos necesarios para comprender el significado del paquete en el contexto.

Ejemplo

Hablando estrictamente, los paquetes y las dependencias son elementos de un diagrama de clases, por lo cual un diagrama de paquetes es sólo un forma de un diagrama de clase.

Existe una dependency dependencia entre dos elementos si los cambios a la definición de un elemento puede causar cambios al otro. En las clases, la dependencia existe por varias razones; una clase envía un mensaje a otra; una clase tiene a otra como parte de sus datos; una clase menciona a otra como parámetros para una operación. Si una clase cambia su interfaz, entonces los mensajes que envían pueden dejar de ser válidos.

En forma ideal, sólo los cambios a una interfaz de clase deberían afectar a otra clase. El arte del diseño en gran escala implica minimizar las dependencias, de modo tal que se reduzcan los efectos del cambio y se requiera de menos esfuerzos para cambiar el sistema.

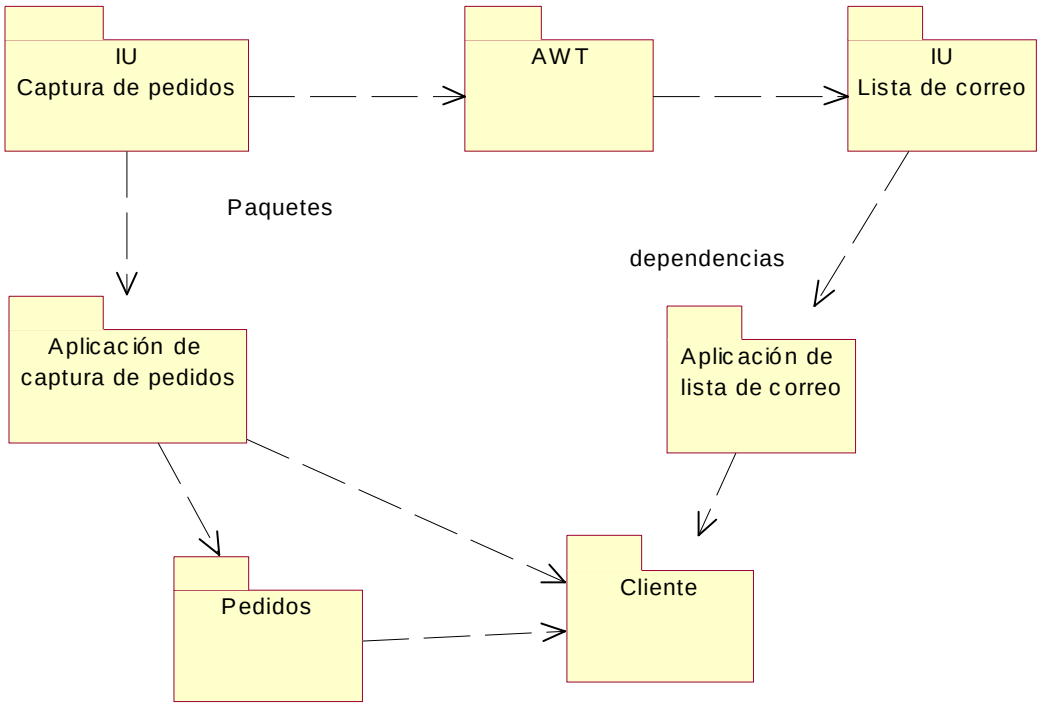


Figura 4.14 **Diagrama de Paquetes**

En la figura 4.10 se presentan las clases de dominio que modelan el negocio, las cuales se agrupan en dos paquetes: Pedidos y Clientes. Ambos paquetes son parte de un paquete que abarca todo el dominio. La aplicación de Captura de pedidos tiene dependencias con los dos paquetes del dominio. La interfaz de Usuario (IU) para Captura de pedidos tiene dependencia con la Aplicación Captura de pedido y con AWT (un juego de herramientas GUI).

Existe una dependencia entre dos paquetes si existe algún tipo de dependencia entre dos clases cualquiera en los paquetes. Por ejemplo, si cualquier clase en el paquete Lista de correo depende de cualquier clase de paquete Clientes, entonces se da una dependencia entre sus paquetes correspondientes.

Un ejemplo de relación transitiva es aquella en la que Juan tiene una barba más larga que Pedro y éste, una más larga que Silvio, por lo que se deduce que Juan tiene una barba más larga de Silvio. Otros ejemplos incluyen relaciones como está “está al norte de” y “es más alto que”. Por otra parte “es un amigo de” no constituye una relación transitiva.

Para apreciar por qué es importante esto para las dependencias, obsérvese de nuevo la figura 4.10. El cambio a una clase del paquete pedidos no indica que el paquete IU Captura de Pedidos deba ser cambiado. Indica tan sólo que debe revisarse el paquete de aplicación Captura de pedido para ver si cambia. Sólo si se altera la interfaz del paquete de aplicación Captura de pedidos hay necesidad de cambiar el paquete IU Captura de pedido. Si esto es así, la aplicación Captura de pedido está protegiendo a la IU Captura de pedidos de cambios a los pedidos.

¿Qué significa trazar una dependencia con un paquete que contenga subpaquetes? Los diseñadores se sirven de convenciones diferentes.

Algunos suponen que dibujar una dependencia hacia un paquete "contenedor" da visibilidad al contenido de todos los paquetes contenidos y a sus respectivos contenidos. Otros consideran que sólo se ven las clases dentro del paquete contenedor, pero no las clases dentro de los paquetes anidados (o sea, la visión es opaca).

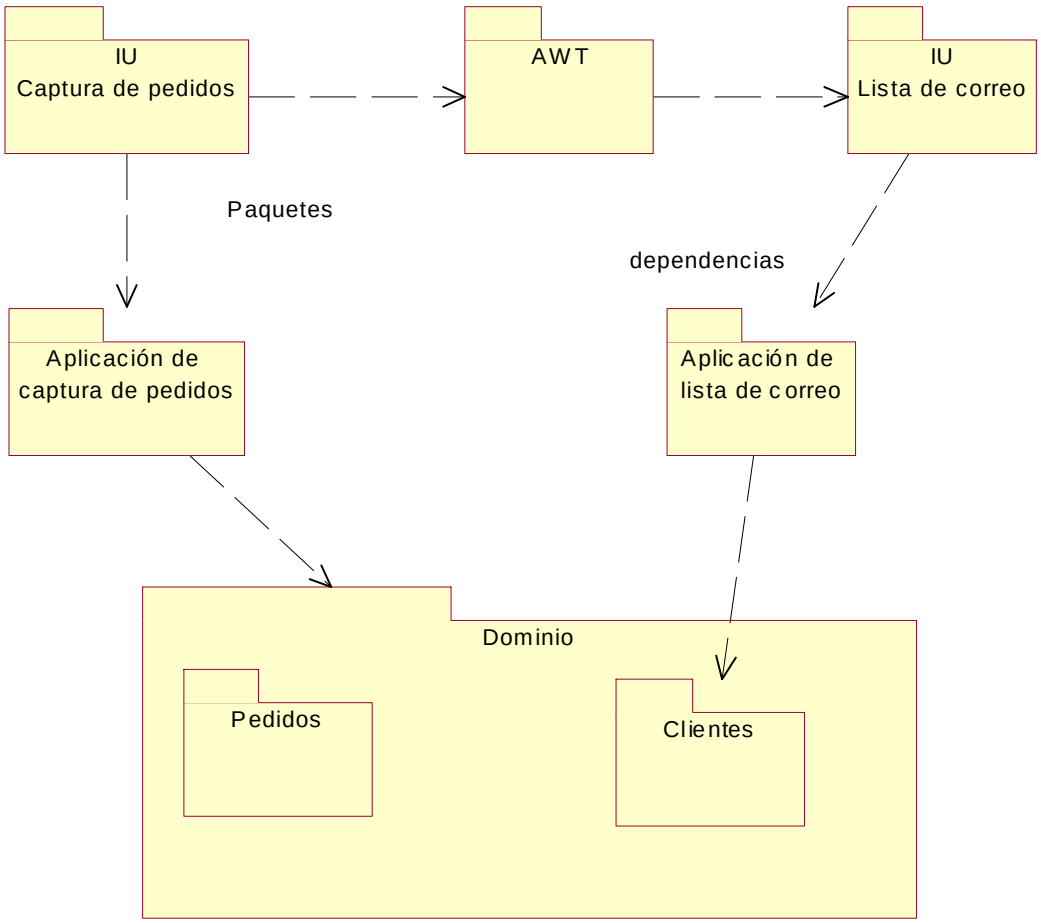


Figura 4.15 **Diagrama de paquetes avanzados**

12.4.6 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 12: Características avanzadas de las relaciones. Páginas 147-158. Presenta el concepto de paquete y sus características principales.

UML Gota a Gota, Martín Fowler con Kendall Scott, Editorial Person

Capítulo 7 Diagramas de Paquetes127- 131

12.5 Clase 35 36 37: Instancias

Introducción, Términos y conceptos, Abstracciones e instancias, Nombres Operaciones,

Estado, Modelado de instancias concretas

Sugerencias al modelar instancias

OBJETIVOS:

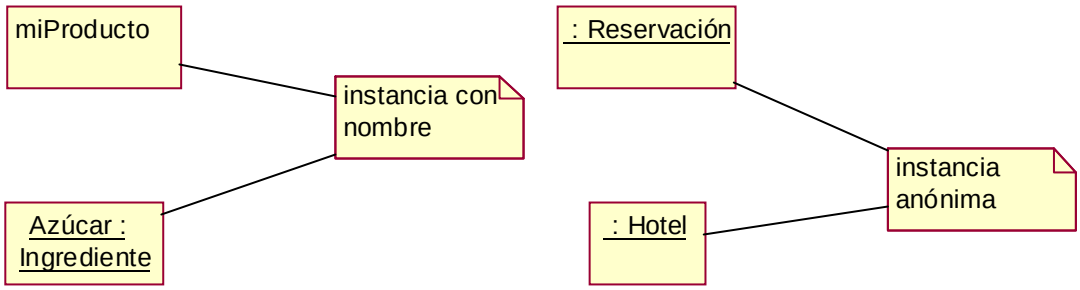
Modelar instancias concretas del mundo real, haciendo uso de objetos.

Distinguir una instancia de una abstracción.

12.5.1 Introducción.

Las instancias se utilizan para modelar cosas concretas o prototípicas del mundo real. Es necesario establecer la diferencia entre una abstracción y una instancia. Una abstracción denota la esencia ideal de una cosa; una instancia denota una manifestación concreta. Esta separación de abstracción e instancia aparecerá en todo lo que se modele. Para una abstracción dada puede haber innumerables instancias. Para una instancia dada, habrá una abstracción que especifique las características comunes a todas las instancias.

En UML se pueden representar abstracciones y sus instancias. Casi todos los bloques de construcción de UML (principalmente las clases, componentes, nodos y casos de uso) pueden modelarse en términos de su esencia o en términos de sus instancias. La mayoría de las veces se trabajará con ellos como abstracciones. Cuando se desee modelar manifestaciones concretas se necesitará trabajar con sus instancias. UML proporciona una notación gráfica para las instancias. Esta notación permite visualizar instancias con nombre, así como otras anónimas.



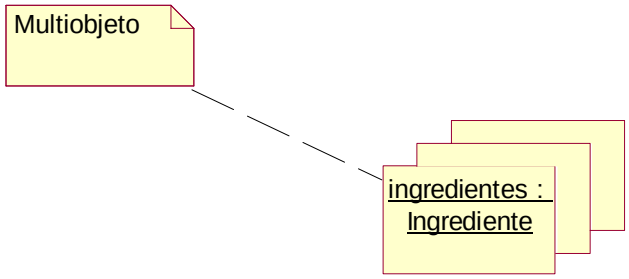


Figura 4.16 **Instancias**

Una *instancia* es una manifestación concreta de una abstracción a la que se puede aplicar un conjunto de operaciones y que posee un estado que almacena el efecto de las operaciones. *Instancia* y *objeto* son en gran parte sinónimos, y por ello, la mayoría de las veces pueden intercambiarse. Gráficamente una instancia se representa subrayando su nombre.

La manifestación concreta de una clase se llama ***objeto***. Los objetos son instancias de clases, así que es apropiado decir que todos los objetos son instancias, aunque algunas instancias no son objetos, por ejemplo la instancia de una asociación no es un objeto, es sólo una instancia también, llamada ***enlace***. Un enlace es una conexión semántica entre objetos. Un enlace se representa con una línea, al igual que una asociación, pero puede distinguirse de ella porque los enlaces sólo conectan objetos.

12.5.2 Abstracciones e instancias.

Las instancias no aparecen aisladas; casi siempre están ligadas a una abstracción. La mayoría de las instancias que se modelen en UML serán instancias de clases, aunque se pueden tener instancias de otros elementos, como componentes, nodos, casos de uso y asociaciones. En UML una instancia es fácilmente distinguible de una abstracción. Para identificar una instancia, se subraya su nombre.

Un objeto es algo que ocupa espacio en el mundo real o conceptual, y al que se le pueden hacer cosas. Por ejemplo una instancia de un nodo es normalmente una computadora que se encuentra físicamente en una habitación; una instancia de un componente ocupa algo de espacio en el sistema de archivos; una instancia de un registro de un cliente consume algo de memoria física.

Una clase abstracta por definición, no puede tener instancias directas.

Cuando se modelan instancias, éstas se colocan en los diagramas de objetos (si se quiere visualizar sus detalles estructurales) o en diagramas de interacción y de actividades (si se desea visualizar su participación en situaciones dinámicas). Se pueden colocar objetos en diagramas de clases para representar explícitamente la relación de un objeto con su abstracción.

Cada instancia debe tener un nombre que la distinga de las otras instancias dentro de su contexto. Un nombre es una cadena de texto como `triángulo`. Ese nombre solo se llama ***nombre elemental***. La abstracción de la instancia puede tener un nombre elemental (como Azúcar) o puede tener un ***nombre de camino*** (como Depósito::Azúcar), el cual consta del nombre de la abstracción precedido por el nombre del paquete en el que ésta se encuentra.

Cuando se da nombre a un objeto de forma explícita, realmente se le está dando un nombre utilizable por una persona. También se puede dar nombre simplemente a un objeto (tal como miProducto) y omitir el nombre de su abstracción si es obvia en el contexto dado.

Cuando se modelan grandes colecciones de objetos, resulta incómodo representar la propia colección más sus instancias individuales. En lugar de ello se pueden modelar multiobjetos (tal como ingredientes), que representa una colección de objetos anónimos.

El nombre de una instancia puede ser texto formado por cualquier número de letras, números y ciertos signos de puntuación (excepto los dos puntos, que se utilizan para separar el nombre de una instancia del nombre de su abstracción) y puede extenderse a lo largo de varias líneas. En la práctica los nombres de las instancias son nombres cortos o expresiones nominales extraídos del vocabulario del sistema que se está modelando. Normalmente, en el nombre de una instancia se pone en mayúsculas la primera letra de cada palabra, excepto la de la primera palabra, como en miPedido.

Un objeto no sólo es algo que normalmente ocupa espacio en el mundo real, también es algo a lo que se le pueden hacer cosas. Las operaciones que se pueden ejecutar sobre un objeto se declaran en la abstracción del objeto. Por ejemplo, suponga que una clase Hotel define la operación realizarReserva, entonces, dada la instancia h1: Hotel, se pueden escribir expresiones como h1.realizarReserva( ). La ejecución de esta expresión significa que sobre h1 (el objeto) opera realizarReserva (la operación). Según la jerarquía de herencia asociada con Hotel, esta operación podría ser invocada polimórficamente o no.

Un objeto también tiene estado, que en este sentido incluye todas las propiedades (normalmente estáticas) del objeto más los valores actuales (normalmente dinámicos) de esas propiedades. Estas propiedades incluyen los atributos del objeto. El estado de un objeto es dinámico, de forma que al visualizar el estado, realmente se está especificando el valor de éste en un momento dado del tiempo y del espacio.

Cuando se opera sobre un objeto, normalmente se cambia su estado; cuando se consulta a un objeto, su estado no se modifica. Por ejemplo, al hacer un cambio de moneda (representada por el objeto dólar : Moneda), se podría establecer el valor de uno de sus atributos (por ejemplo, precio_venta=192). Si el valor del dólar cambia por el deslizamiento de la moneda en un país X, entonces el estado podría cambiar (por ejemplo, precio_venta=200).

12.5.3 Modelado de instancias concretas.

Una de las cosas para las que se emplean los objetos es para modelar instancias concretas del mundo real. Por ejemplo, si se quiere modelar la topología de una red de computadoras de una empresa, se utilizarán diagramas de despliegue que contendrán instancias de nodos. Si se desea modelar los componentes que se encuentran en los nodos

físicos de la red, se utilizarán diagramas de componentes que contendrán instancias de los componentes.

Para modelar instancias concretas:

Hay que especificar aquellas instancias necesarias y suficientes para visualizar, especificar, construir o documentar el problema que se está modelando.

Hay que representar esos objetos en UML como instancias. Hay que dar un nombre a cada objeto. Si no hay un nombre significativo para el objeto, puede representarse como un objeto anónimo.

Hay que mostrar el estereotipo, los valores etiquetados y los atributos (con sus valores) de cada instancia necesarios para modelar el problema.

Hay que representar estas instancias y sus relaciones en un diagrama de objetos u otro diagrama apropiado al tipo de instancia (nodo, componente, etc).

Sugerencias al modelar instancias.

Cuando se modelan instancias en UML, recuerde que toda instancia debe denotar una manifestación concreta de alguna abstracción (una clase, componente, nodo, caso de uso o asociación). Una instancia bien estructurada:

Está asociada explícitamente con una abstracción específica.

Tiene un nombre único extraído del vocabulario del dominio del problema o del dominio de la solución.

Cuando se dibuje una instancia en UML:

Hay que representar el nombre de la abstracción de la cual es instancia, a menos que sea obvio por el contexto.

Hay que mostrar el estereotipo de la instancia y el estado sólo cuando sea necesario para comprender el objeto en su contexto.

12.5.4 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 13: Características avanzadas de las relaciones. Páginas 159-168. Explica de manera amplia el concepto de instancia.

13 UNIDAD V: MODELADO BASICO DEL COMPORTAMIENTO

Temas a abordar:

Diagramas de actividades.

Esta unidad se impartirá en una sesión.

13.1 CLASES 38 y 39 : DIAGRAMAS DE ACTIVIDADES.

Diagramas de actividades.

OBJETIVO:

Dar a conocer la importancia que tienen los diagramas de actividades para modelar los aspectos dinámicos de un sistema.

13.1.1 Diagrama de actividades.

Los diagramas de actividades constituyen una de las partes más imprevistas del UML.

El diagrama de actividades, a diferencia de la mayor parte de las demás técnicas del UML, no tiene su origen evidente en los trabajos de anteriores de los tres amigos. El diagrama de actividades combina ideas de varias técnicas: el diagrama de eventos de Jim Odell, las técnicas de modelado de estado de SDL y las redes de Petri. Estos diagramas son particularmente útiles en conexión con el flujo de trabajo y para la descripción del comportamiento que tiene una gran cantidad de proceso paralelo.

Un *diagrama de actividades* muestra el flujo de actividades. Una actividad es una ejecución no atómica en curso, dentro de una máquina de estados. Las actividades producen finalmente una acción, que está compuesta de computaciones atómicas ejecutables que producen un cambio en el estado del sistema o la devolución de un valor. Las acciones incluyen llamadas a otras operaciones, envío de señales, creación o destrucción de objetos o simples cálculos, como la evaluación de una expresión. Gráficamente, un diagrama de actividades es una colección de nodos y arcos.

Un diagrama de actividades es un tipo especial de diagrama y comparte las propiedades comunes al resto de diagramas (un nombre y un contenido gráfico que es una - proyección de un modelo). Lo que distingue a un diagrama de actividades de los otros tipos de diagramas es su contenido.

Normalmente, los diagramas de actividades contienen:

- Estados de actividad y estados de acción.
- Transiciones.
- Objetos.

Un diagrama de actividades es un tipo de máquina de estados, por lo que se le aplican todas las características de las máquinas de estados. Esto significa que los

diagramas de actividades pueden contener estados simples y compuestos, bifurcaciones, divisiones y uniones.

13.1.2 Estados de acción y estados de actividad.

Los estados de acción son computaciones ejecutables y atómicas, se llaman así porque son estados del sistema y cada una representa la ejecución de una acción. Por ejemplo, una acción podría ser el evaluar una expresión que estableciera el valor de un atributo o que devolviera algún valor. Un estado de acción se representa con una figura en forma de píldora (un símbolo con líneas horizontales arriba y abajo y lados convexos) y dentro de esta figura se puede escribir cualquier expresión. Los estados de acción son atómicos, esto quiere decir que pueden ocurrir eventos, pero no se interrumpe la ejecución del estado de acción. Los estados de acción no se pueden descomponer. Generalmente se considera que la ejecución de un estado de acción conlleva un tiempo insignificante.

Un estado de actividad puede ser visto como un elemento compuesto, cuyo flujo de control se compone de otros estados de actividad y estados de acción. Los estados de actividad pueden descomponerse aún más, representando su actividad con otros diagramas de actividades. Los estados de actividad no son atómicos, lo que quiere decir que pueden ser interrumpidos y, en general, se considera que invierten algún tiempo en completarse. Un estado de acción se puede ver como un caso especial de un estado de actividad. Un estado de acción es un estado de actividad que no se puede descomponer más.

En cuanto a la notación, no hay distinción en cuanto a los estados de actividad y los estados de acción, excepto que un estado de actividad puede tener partes adicionales, como acciones de entrada y salida (entry/exit) (acciones relacionadas con la entrada y la salida del estado, respectivamente) y especificaciones de submáquinas.

13.1.3 Transiciones.

Cuando se completa la acción o la actividad de un estado, el flujo de control pasa inmediatamente al siguiente estado de acción o estado de actividad. Este flujo se especifica con transiciones que muestran el camino de un estado de actividad o de un estado de acción al siguiente. Una transición se representa como línea dirigida. Se puede especificar un estado inicial (un círculo relleno) y un estado final (un círculo relleno dentro de una circunferencia).

13.1.4 Bifurcación.

Como en los diagramas de flujo, se puede incluir una bifurcación, que especifica caminos alternativos, elegidos según el valor de alguna expresión booleana. Una bifurcación se representa como un rombo. Una transición puede tener una transición de entrada y dos o más de salida.

13.1.5 División y unión.

En un diagrama de actividades también es posible encontrar flujos concurrentes, especialmente cuando se modelan flujos de trabajo de procesos de negocio. Por este motivo, en UML se utiliza una barra de sincronización para especificar la división y unión

de estos flujos de control paralelos, dicha barra de sincronización se representa como una línea horizontal o vertical ancha.

13.1.6 Calles (swimlanes).

Se pueden dividir los estados de actividad de un diagrama de actividades en grupos, donde cada uno representa una parte de la organización responsable de ciertas actividades. Cada calle tiene un nombre único dentro del diagrama y no tiene una semántica profunda. En un diagrama de actividades organizado en calles, cada actividad pertenece a una única calle, pero las transiciones pueden cruzar las calles.

13.1.7 Flujo de Objetos.

Dentro de un diagrama de actividades pueden verse involucrados objetos y por tanto se pueden especificar esos objetos colocándolos en el diagrama, conectados con una dependencia a la actividad o transición que los crea, los destruye o los modifica. Este uso de las relaciones de dependencia y de los objetos se denomina flujo de objetos, porque representa la participación de un objeto en un flujo de control. El estado de un objeto se representa dando al estado un nombre entre corchetes bajo el nombre del objeto. También se puede representar el valor de los atributos de un objeto en un compartimento bajo el nombre del objeto. Ejemplo:

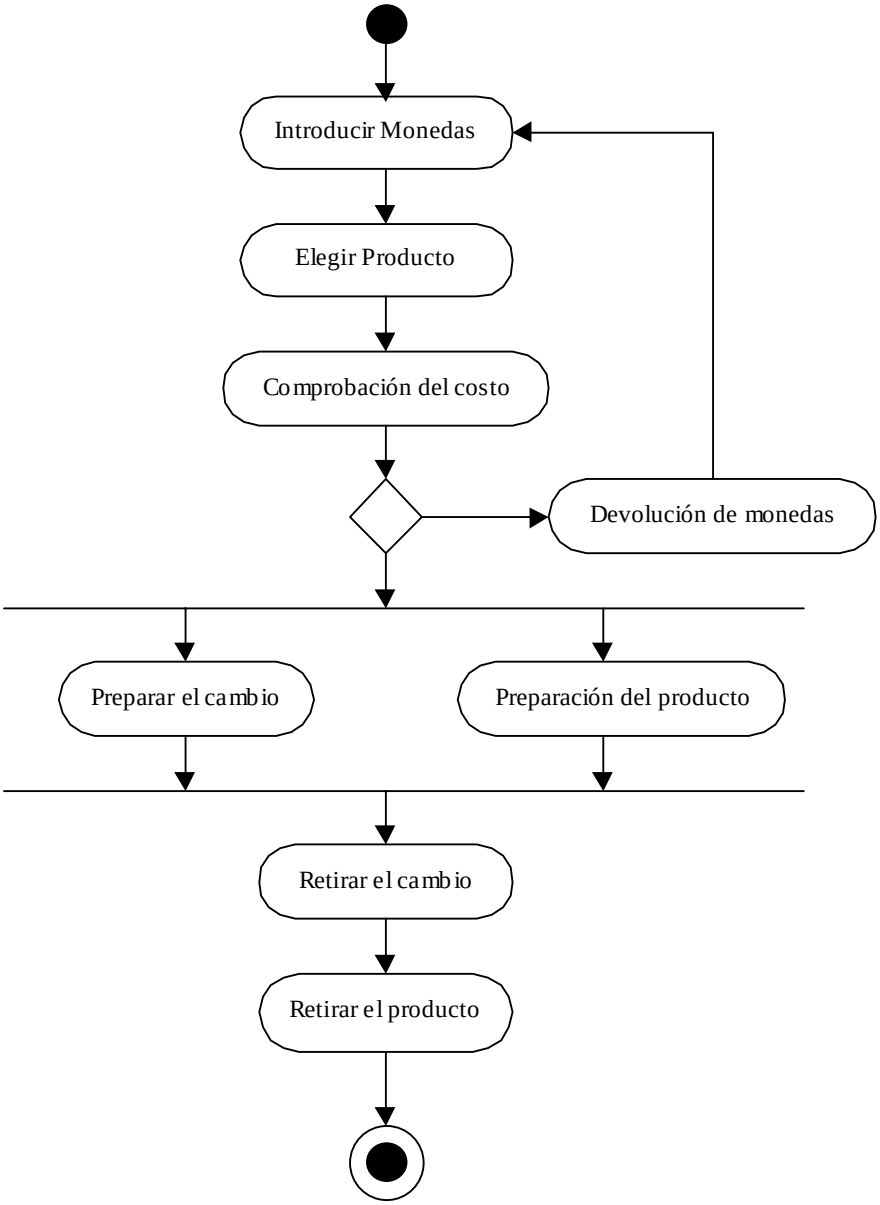


Figura 5.1 **Diagrama de actividades**

En este otro ejemplo se puede ver el uso de las calles, las cuales dividen los estados de actividad en grupos, donde cada uno de esos grupos representa la parte del sistema responsable de esas actividades.

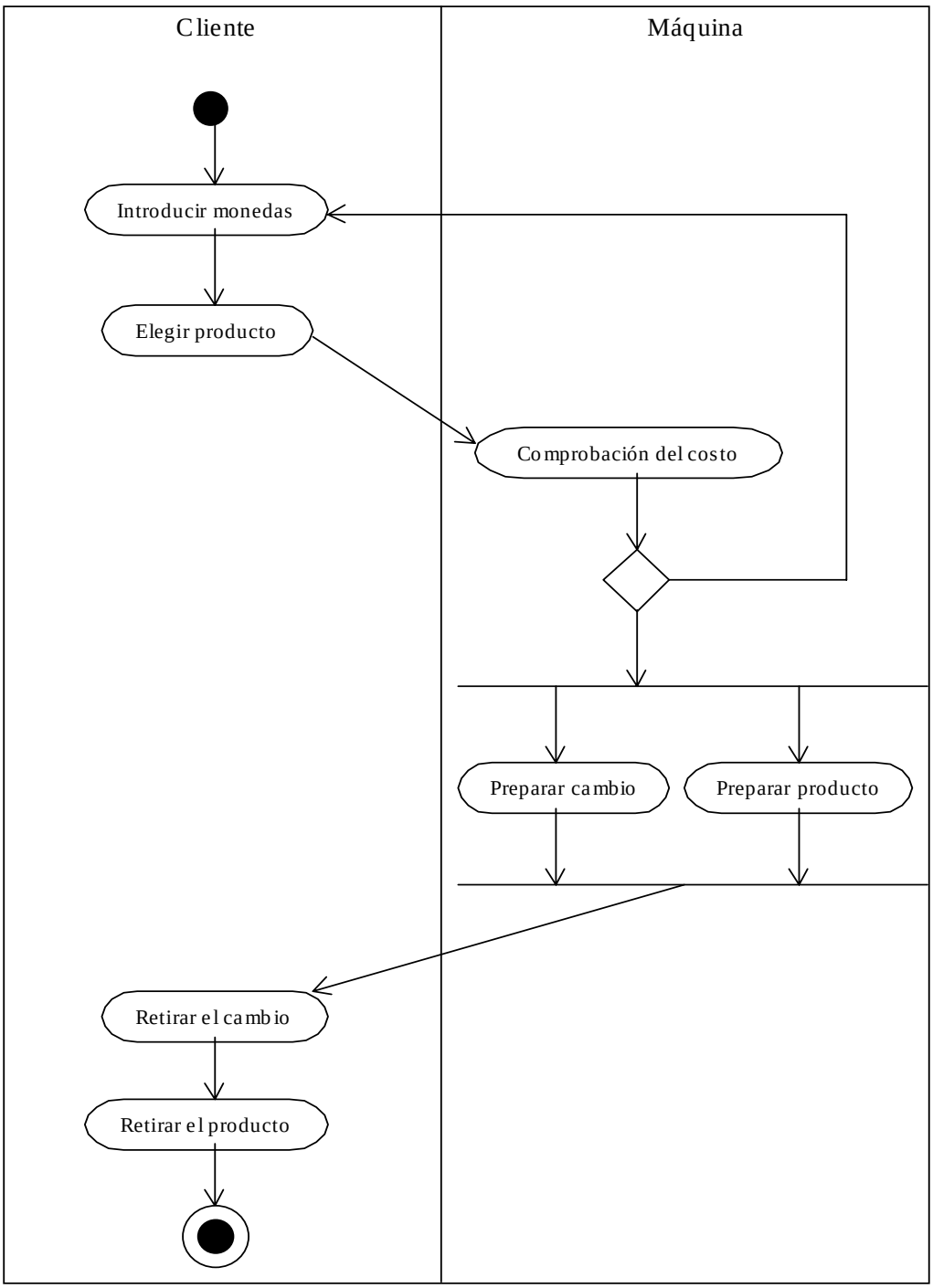


Figura 5.2 **Diagrama de actividades (con calles)**

13.1.8 Bibliografía

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Editorial Addison Wesley. Capítulo 19: Diagramas de actividades. Páginas 225-239. Explica ampliamente las características de un diagrama de actividades y sus componentes.

Modelado de objetos con UML. Pierre-Alain Muller. Editorial Eyrolles. Capítulo 3: La notación UML. Páginas 171-175. Resume las características de un diagrama de actividades.

UML Distilled “Applyng the standard Object Modeling Language”. Martin Fowler. Editorial Addison Wesley. Capítulo 6: Activity Diagrams. Páginas 129-142. Presenta de manera breve las principales características de los diagramas de actividades.

13.2 ANEXO

Enunciado del Problema

El ejercicio consiste en realizar el diseño de un control de acceso a un edificio universitario.

El edificio consta de diferentes tipos de personas

Al edificio solamente se puede acceder tipos de personas

Al edificio solamente se puede acceder en determinadas franjas horarias

Dependiendo del tipo de personas, se podrá acceder a unas zonas del edificio o a otras.

En caso de que se intente acceder a zonas no autorizadas se debe indicar con una alarma en el monitor del guarda de seguridad.

Los espacios a proteger se distribuyen por cuatro niveles dentro de un edificio de una superficie total de unos 5000 m2. El edificio se divide en cinco zonas: dos aulas de investigación, un aula de trabajo prácticos, un aula para la administración y un cuerpo central que arbitra las aulas de clases y los dos anfiteatros.

Tras la desaparición de objetos diversos, se ha decidido restringir los accesos a ciertas dependencias por medio de puertas de cierre automático. La apertura de cada una de estas puerta se controla por un lector de tarjetas.

Las tarjetas que permiten la apertura de las puertas solamente se entregan a las personas que deben acceder a los locales protegidos en el ejercicio de sus actividades. Los derechos de acceso se asignan entre los grupos de personas y los grupos de puertas, de modo que una persona o una puerta siempre debe estar al menos en un grupo (el suyo).

Un grupo de puertas puede contener puertas dispersas por todo el edificio. Desde el punto de vista del control de acceso sólo es importante la noción de grupos de puertas: los caminos y los desplazamientos no se controlan. Una puerta dada solamente puede pertenecer a un grupo de puertas.

Una misma persona puede pertenecer a varios grupos, de modo que sus derechos de acceso corresponden a la unión de los derechos de acceso de cada uno de los grupos que la contienen.

La definición de los derechos de acceso se efectúa describiendo para cada grupos de personas los diferentes grupos de puertas que no son accesibles y bajo qué restricciones horarias.

Los derechos de acceso se describen en un calendario anual que describe la situación semana a semana. Dada la poca variación de los derechos en el tiempo, puede inicializarse un calendario por medio de semana tipo que describe una configuración de derechos dada. El supervisor puede crear tantas semanas tipo como desee. Los cambios aportados a una semana tipo se programan automáticamente en todos los calendarios que utilizan esta semana tipo. Los cambios aportados directamente en un calendario, por ejemplo para tener en cuenta un día festivo, no se ven efectuados en la modificación de la semana tipo.

El sistema de control de acceso debe funcionar de la manera más autónoma posible. Un supervisor es responsable de la configuración inicial y de la actualización de las diferentes informaciones de definiciones de los grupos de personas y puertas. Un guarda dispone de una pantalla de control y es informado de los intentos de paso infructuosos. Las alarmas se transmite en tiempo ligeramente diferido: la actualización de la información sobre la pantalla de control se efectúa cada minuto.

La interfaz de usuario debe ayudar al usuario a formular peticiones correctas. Los valores de parámetros deben ser sistemáticamente leídos en lista que definen el ámbito de los valores correctos.

14 PRÁCTICAS DE LABORATORIOS

14.1 Laboratorio I

Guía de laboratorio

14.1.1 Tema: Especificación de Requisitos

Funciones del Sistema

Categoría de las funciones

Representación de los actores y los casos de Uso

Casos de uso de alto nivel

14.1.2 Competencia o habilidad

Este Laboratorio pretende que usted adquiera la competencia necesaria para "Interpretar las especificaciones de Requisitos de un sistema como son: funciones, diagramas de casos de uso etc.

14.1.3 Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

Dado el análisis de un problema usando ingeniería de Software usando UML, interpretara los requisitos y funciones del sistema.

Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas.

14.1.4 Actividades de Aprendizaje

Lea los objetivos aquí propuestos, desvanezca dudas y no inicie el estudio del material si no tiene claro que competencia y objetivos se persiguen y como lograrlos.

Lea el enunciado del problema, después de leerlo y revisado el ejemplo de especificación de requisitos, usted analizará la situación, primero de manera individual y después en grupo. Durante el análisis del caso irá plasmando su punto de vista de acuerdo al lugar que ocupa en contra o a favor.

Dibujar el diagrama de casos de uso que usted considere es el correcto.

Modifique los casos de uso que considere mas idóneos.

Tiempo aproximado 2 horas

Al final se realizará la evaluación del laboratorio, mediante la entrega de su aportes con sus razones por las cuales esta a favor o en contra.

Desarrollo

INTRODUCCION

El presente trabajo corresponde a la fase de Planificación para el desarrollo del sistema en el ejercicio propuesto **"APLICACION DE CONTROL DE ACCESO A UN EDIFICIO"**

II. ESPECIFICACION DE REQUISITOS

Presentación General

Este proyecto tiene por objeto crear un sistema automatizados que se utilizará para el control de acceso a todas las áreas de un edificio por medio de puertas de cierre automático.

Cliente

Edificio Universitario ESSAIM

Metas

Mejor control de acceso a los locales protegidos. Incluye:

Rápido acceso una vez autorizada la tarjeta

Análisis rápido y exacto de los locales donde se accedió el permiso y la tarjeta que lo autorizo

Glosario

1. ESSAIM: Escuela Superior de Ciencias Aplicadas para el Ingeniero-Molhouse.

Supervisor: Es el persona encargada de configurar inicialmente el sistema y realizarle las Modificaciones necesarias.

Portador de Tarjeta: Son todas las personas que tienen autorización para entrar al edificio.

Vigilante (Operador) Persona encargada monitorear los diferentes accesos al edificio y usar las diferentes opciónes del sistema, funcione del sistema, el producto debe realizar las siguientes funciones:

El sistema solicitará contraseña de acceso tanto al supervisor como al Operador o Vigilante.

Definir los conjuntos de puertas que pertenecen a cada uno de los niveles de restricciones y almacenarlos en el sistema.

Agregar al sistema todos los datos de las personas, definiendo a los grupos de puertas que tienen autorización.

Definir y agregar al sistema las semanas tipos con las restricciones de horarios y los días feriados.

Realizar las modificaciones de derecho a grupos de personas, variación de horario en semanas tipos, modificar semanas tipos.

Los visitantes portadores de tarjetas, la introducirán en el lector de tarjeta que se encuentra en cada una de las puertas y el sistema validara si esta persona tiene los derechos para entrar a este local por esta puerta, de lo contrario la puerta no se abrirá y emitirá una señal.

Al operador o vigilante (guarda) le llegará la información de los intentos de acceso (acertados y erróneos) , a través de un monitor.

- Deberá listar los siguientes listados e informes:
- Personas que tienen acceso a cada grupo de puertas.
- Personas que han entrado a un grupo de puerta determinado, en un momento determinado.
- Personas que han intentado acceder a grupos de puertas que no tienen derecho de entrar.
- El horario definido para ingresar al edificio o a un grupo de puerta en una semana tipo.
- Informes de eventos en período determinado.

Categorías de las funciones:

Ref. Núm.	Función	Categoría
R1.1	El sistema solicitará contraseña de acceso tanto al supervisor como al Operador o Vigilante.	Evidente
R1.2	Definir los conjuntos de puertas que pertenecen a cada uno de los niveles de restricciones y almacenarlos en el sistema.	Evidente
R1.3	Agregar al sistema todos los datos de las personas, definiendo a los grupos de puertas que tienen autorización.	Evidente
R1.4	Definir y agregar al sistema las semanas tipos con las restricciones de horarios y los días feriados.	Evidente
R1.5	Realizar las modificaciones de derecho a grupos de personas, variación de horario en semanas tipos, modificar semanas tipos.	Evidente
R1.6	Los visitantes portadores de tarjetas, la introducirán en el lector de tarjeta que se encuentra en cada una de las puertas y el sistema validara si esta persona tiene los derechos para entrar a este local por esta puerta, de lo contrario la puerta no se abrirá y emitirá una señal.	Evidente
R1.7	Al operador o vigilante (guarda) le llegará la información de los intentos de acceso (acertados y erróneos) , a través de un monitor.	Evidente
R1.8	Deberá listar los siguientes listados e informes: Personas que tienen acceso a cada grupo de puertas. Personas que han entrado a un grupo de puerta determinado, en un momento determinado. Personas que han intentado acceder a grupos de puertas que no tienen derecho de entrar. El horario definido para ingresar al edificio o a un grupo de puerta en una semana tipo. Informes de eventos en período determinado.	Evidente

Atributos del sistema en la especificación de funciones

Ref. Num	Función	Categoría	Atributos	Detalles y restricciones	Categoría
R1.6	Los visitantes portadores de tarjetas, la introducirán en el lector de tarjeta que se encuentra en cada una de las puertas y el sistema validara si esta persona tiene los derechos para entrar a este local por esta puerta, de lo contrario la puerta no se abrirá y emitirá una señal	Evidente	Tiempo de respuesta Tolerancia a fallos	Max 10 segundo Debe registrar el listado de control de las personas que entraron al edificio en un plazo de 12 horas aun cuando se produzcan fallas de energía o del equipo	Obligatorio Obligatorio
R1.7	Al operador o vigilante (guarda) le llegará la información de los intentos de acceso (acertados y erróneos) , a través de un monitor	Evidente	Tiempo de Repuesta Tolerancia a fallos	Cada minuto Aperturas de puertas manuales.	Obligatorio

Actores

El portador de Tarjeta

Administrador de Sistema

Operador Vigilante

Casos de Usos

Definición inicial de Parámetros

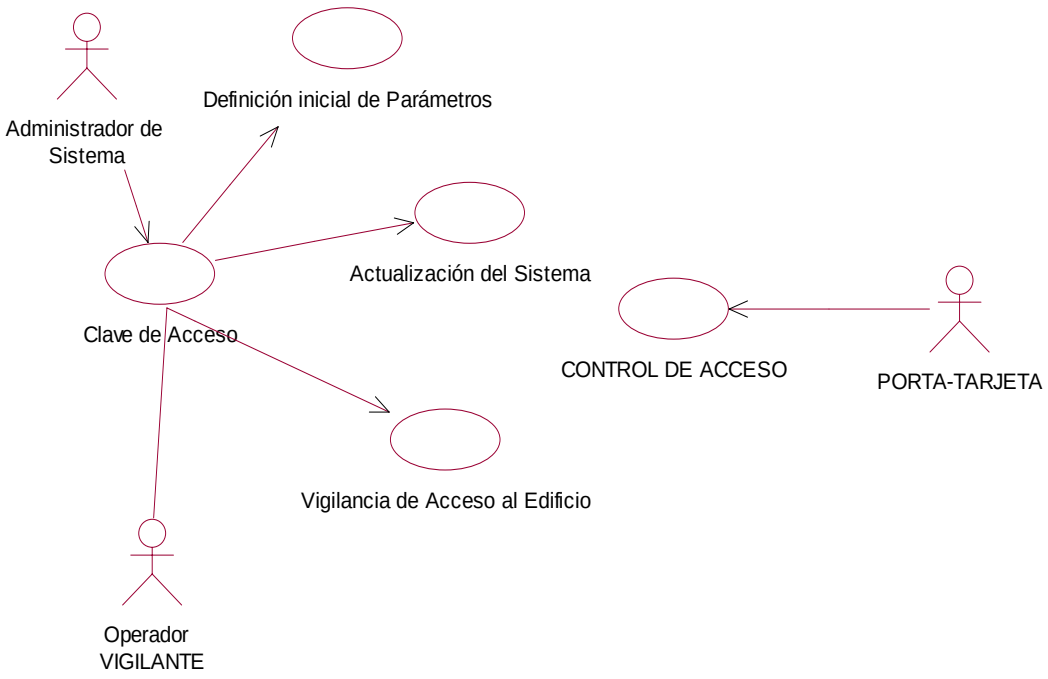
Actualización del Sistema

Clave de Acceso

Vigilancia de Acceso al Edificio

Control de tarjeta

Actividad de laboratorio: graficar el siguiente digrama en Rational Rose además interpretar el porque están indicados como casos de uso



Caso de Uso de Alto Nivel: Clave de acceso

Caso de Uso: Clave de acceso

Actores: Supervisor, Vigilante (operador)

Tipo: Primario

Descripción: El supervisor y el vigilante se comunican con el sistema a través de su clave de acceso para:

Definir inicialmente el sistema,

Actualización del sistema y

Vigilancia de acceso al edificio

Caso de Uso de Alto Nivel: Definición inicial del Parámetros

Caso de Uso: Definición inicial del Parámetros

Actores: Supervisor.
Tipo: Primario

Descripción: El supervisor, establece grupos de puertas de acceso, grupos de personas, tipos de tarjetas a utilizar y establece semanas tipo

Actividad de laboratorio realizar los casos de usos de alto nivel de los siguientes casos:

Respuestas de las casos de uso de alto nivel

Actualización de Sistema, Vigilancia de acceso al edificio, Control de Acceso,

Caso de Uso de Alto Nivel: Actualización de Sistema

Caso de Uso: Actualización de Sistema

Actores: Supervisor.

Tipo: Primario

Descripción: El supervisor modifica, grupos de puertas de acceso, grupos de personas, tipos de tarjetas a utilizar, semanas tipo

Caso de Uso de Alto Nivel: Vigilancia de acceso al edificio

Caso de Uso: Vigilancia de acceso al edificio

Actores: Vigilante (operador)

Tipo: Primario

Descripción: Vigila el acceso al edifico y realiza informes en un periodo de tiempo determinado. monitorea las alarmas por acceso negado, activa alarma contra: incendios, robos, inundaciones, terremotos, etc.

Caso de Uso de Alto Nivel: Control de Acceso

Caso de Uso: Control de Acceso

Actores: Portador de Tarjeta

Tipo: Primario

Descripción: El portador de tarjeta introduce la tarjeta en el lector de tarjeta , el sistema determinara su derecho de acceso, en caso de autorizado la puerta se abre. Si es denegado envía un mensaje.

14.2 Laboratorio II

Guía de laboratorio

14.2.1 Tema: Diagrama de clase

14.2.2 Competencia o habilidad

Este Laboratorio pretende que usted adquiera la competencia necesaria para "Interpretar el diagrama de Clase."

14.2.3 Objetivos de Aprendizaje

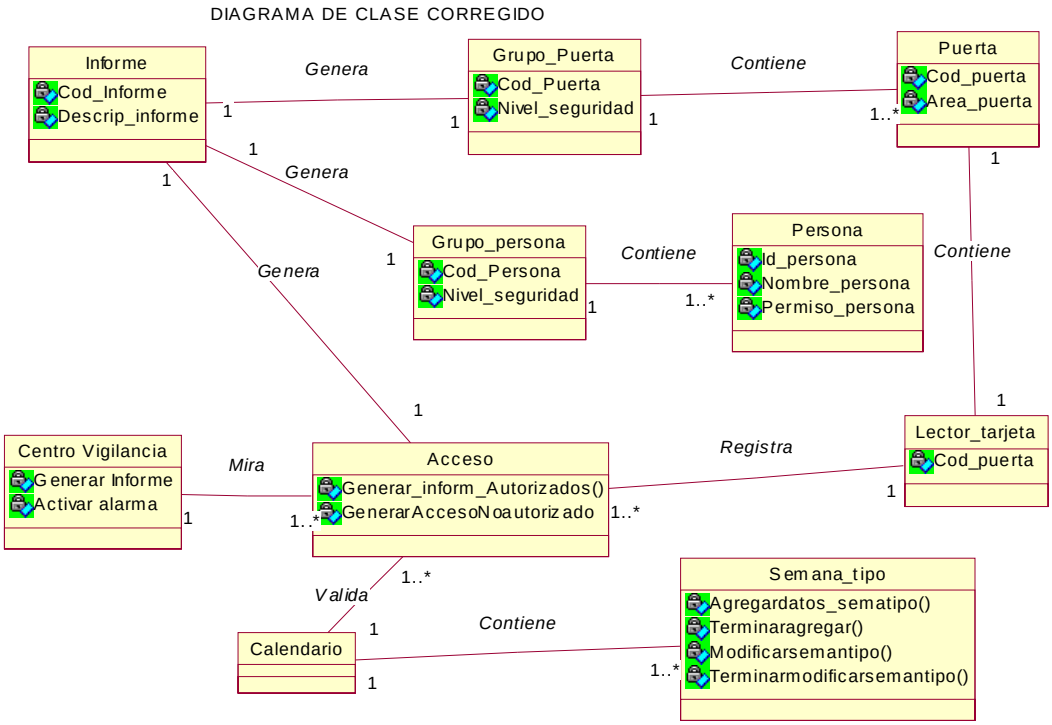
Al concluir el estudio de este laboratorio, usted podrá:

Interpretar el diagrama de clase.

14.2.4 Actividades de Aprendizaje

- Lea los objetivos aquí propuestos, desvanezca dudas y no inicie el estudio del material si no tiene claro que competencia y objetivos se persiguen y como lograrlos.
- Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas
- Crear el diagrama con las modificaciones que encuentre si es que encuentre.
- Tiempo aproximado 2 horas
- Al final se realizará la evaluación del laboratorio, mediante la entrega de el modelo de clase que usted considere si es que la hay y si no la hay argumento los conceptos.
- Desarrollo

Actividad de laboratorio: graficar el diagrama de clase usando el programa Rational Rose además interpretar para su confrontación.



14.3 Laboratorio III

Guía de laboratorio

14.3.1 Tema: Diagrama de Secuencia. Contrato de los Diagrama de Secuencia

14.3.2 Competencia o habilidad

Este Laboratorio pretende que usted adquiera la competencia necesaria para "Interpretar los diagrama de secuencia y dibujarlos e identificar la estructura de los contratos"

14.3.3 Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

Interpretar los diagrama de secuencia.

Dibujar diagrama de secuencia

Interpretar contratos de los casos de uso

Actividades de Aprendizaje

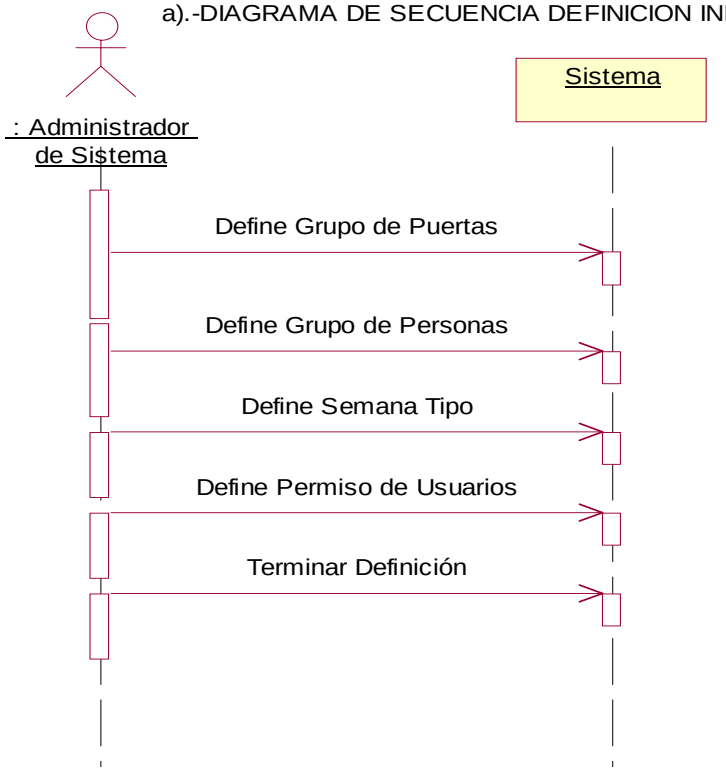
- Revise los ejemplos de diagrama de secuencia.
- Interprete los diagrama de secuencia anotando sus observaciones a favor y en contra
- Dibujar los diagramas de secuencia en el modelo UML
- Interpretar los diagramas de contratos anotando sus observaciones a favor y en contra
- Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas
- Tiempo aproximado 2 horas
- Al final se realizará la evaluación del laboratorio, mediante la entrega de su Reedición del enunciado del problema.

Desarrollo

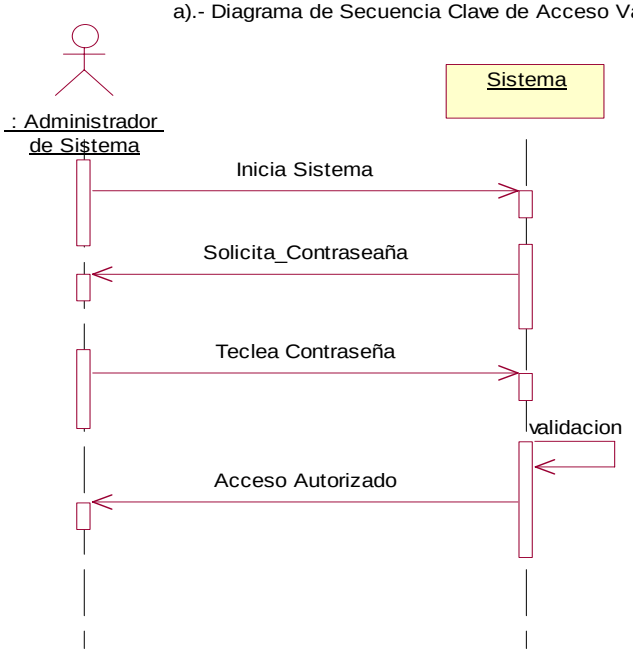
Actividad Representar gráficamente los diagramas de secuencias

CASO DE USO "DEFINICION INCIAL DE PARAMETROS"

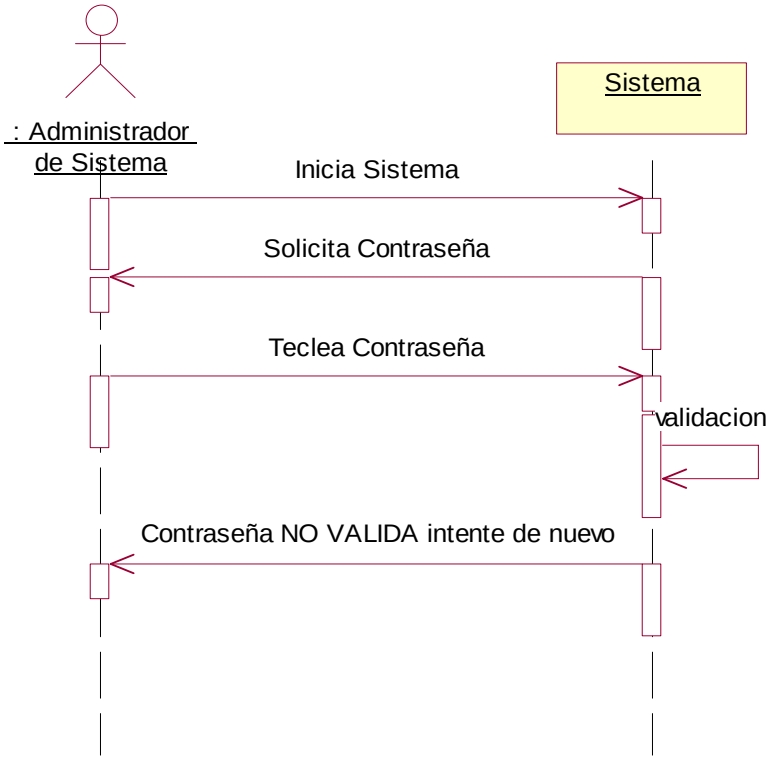
a).-DIAGRAMA DE SECUENCIA DEFINICION INICIAL DE PARAMETROS



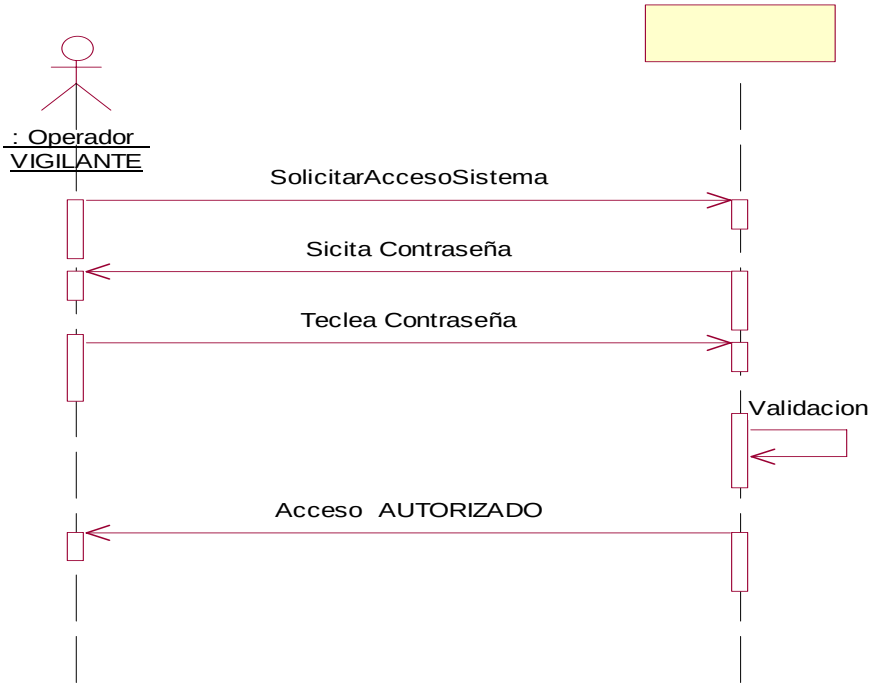
a).- Diagrama de Secuencia Clave de Acceso Valida-Administrador



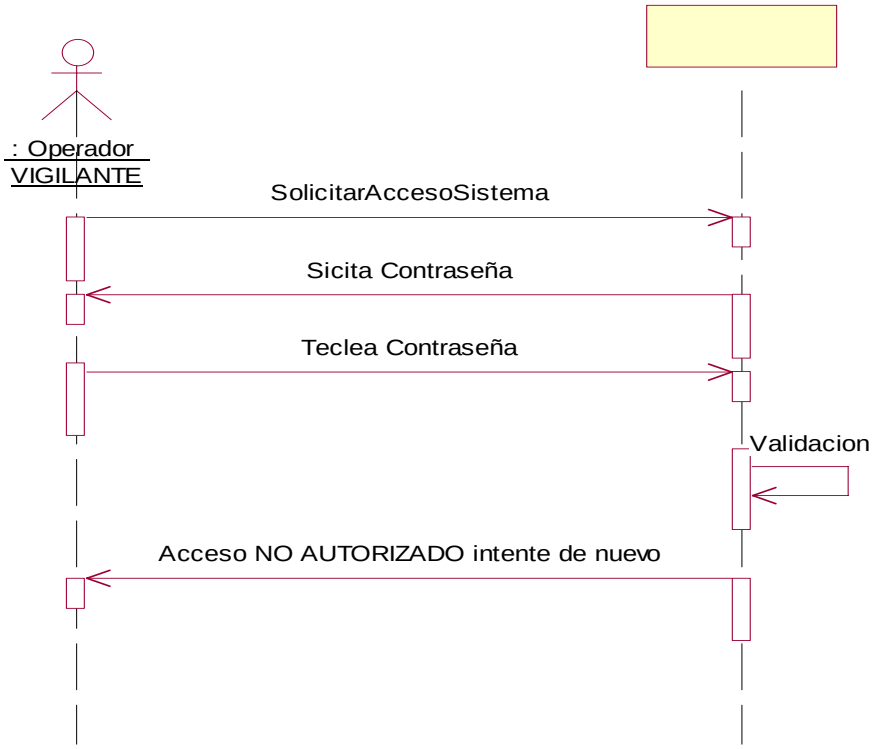
b).-Diagrama de Secuencia Clave de Acceso NO VALIDA- ADMINISTRADOR



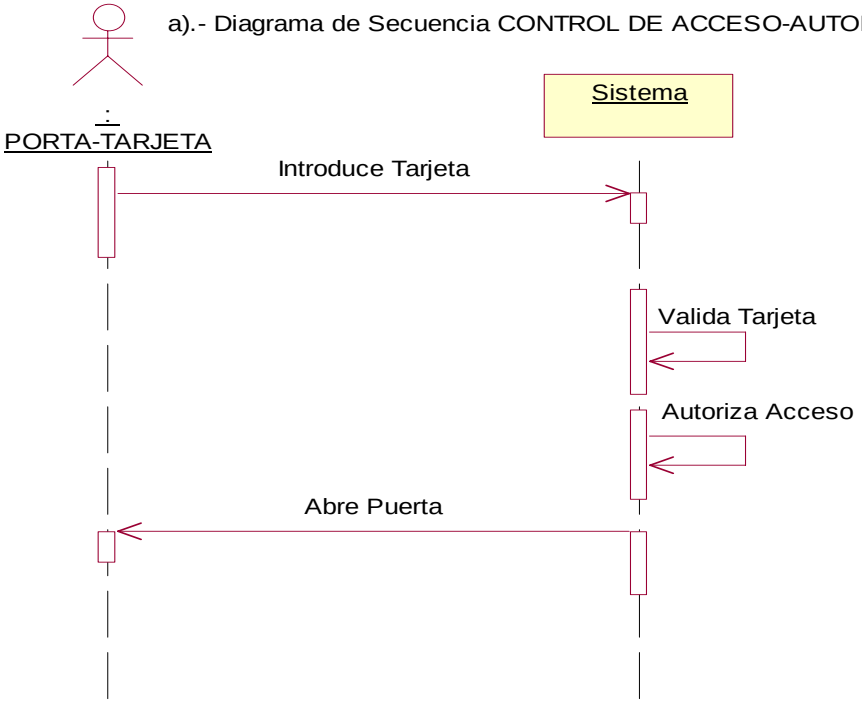
c).- Diagrama de Secuencia Clave de Acceso Valida-Vigilante



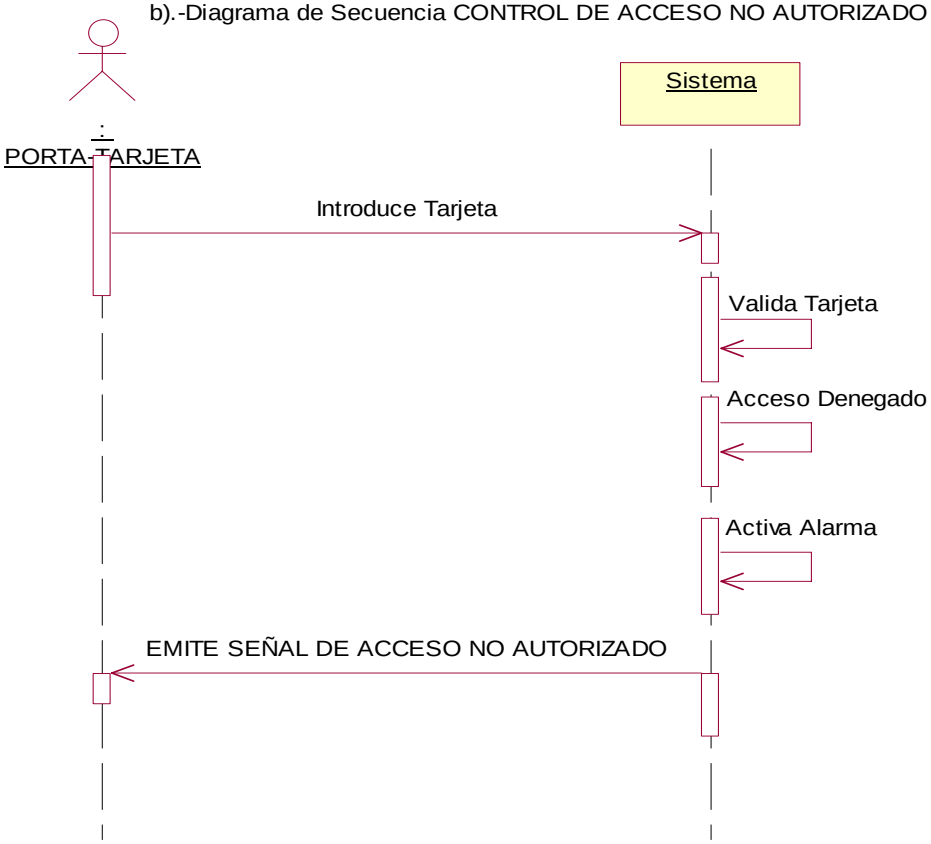
d).- Diagrama de Secuencia Clave de Acceso No Valida-Vigilante



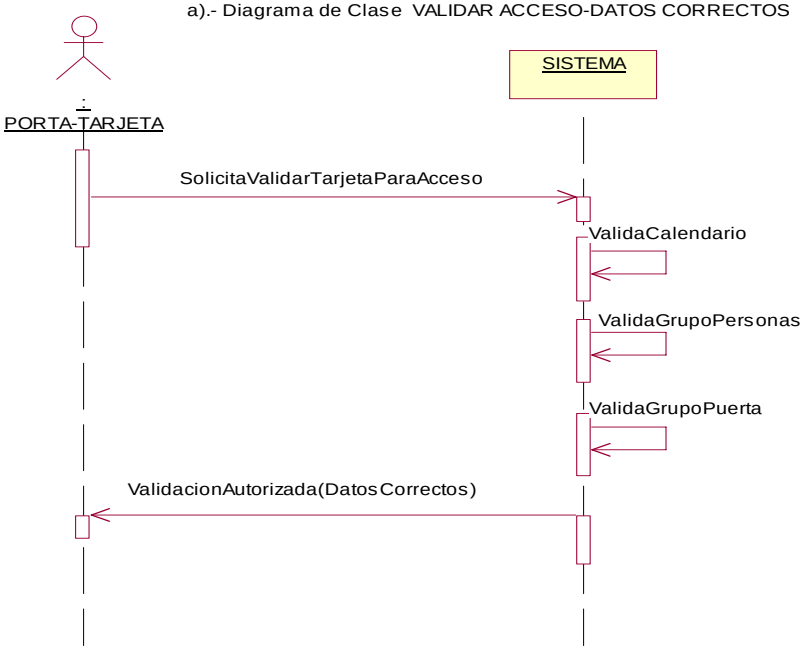
a).- Diagrama de Secuencia CONTROL DE ACCESO-AUTORIZADO



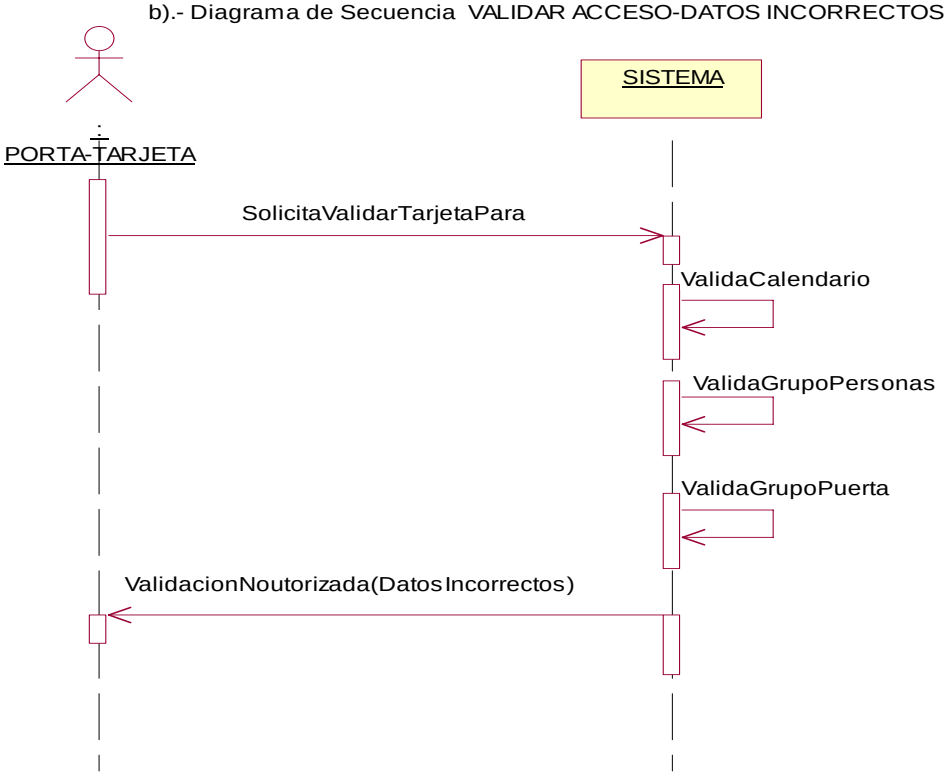
b).-Diagrama de Secuencia CONTROL DE ACCESO NO AUTORIZADO



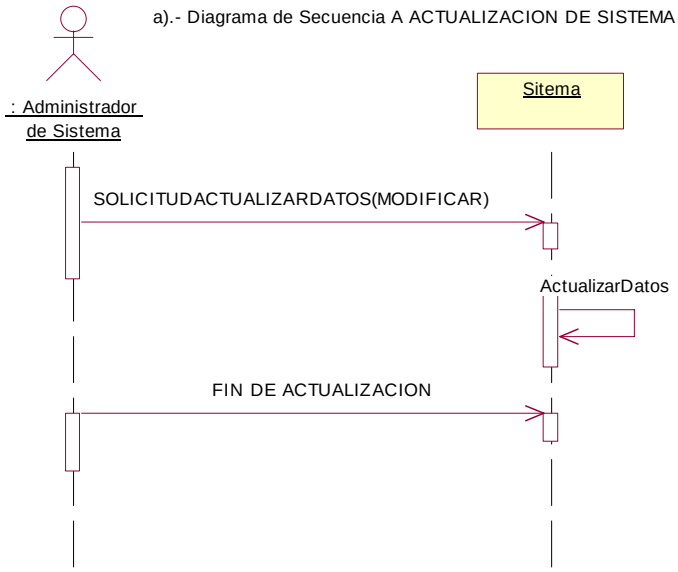
a).- Diagrama de Clase VALIDAR ACCESO-DATOS CORRECTOS



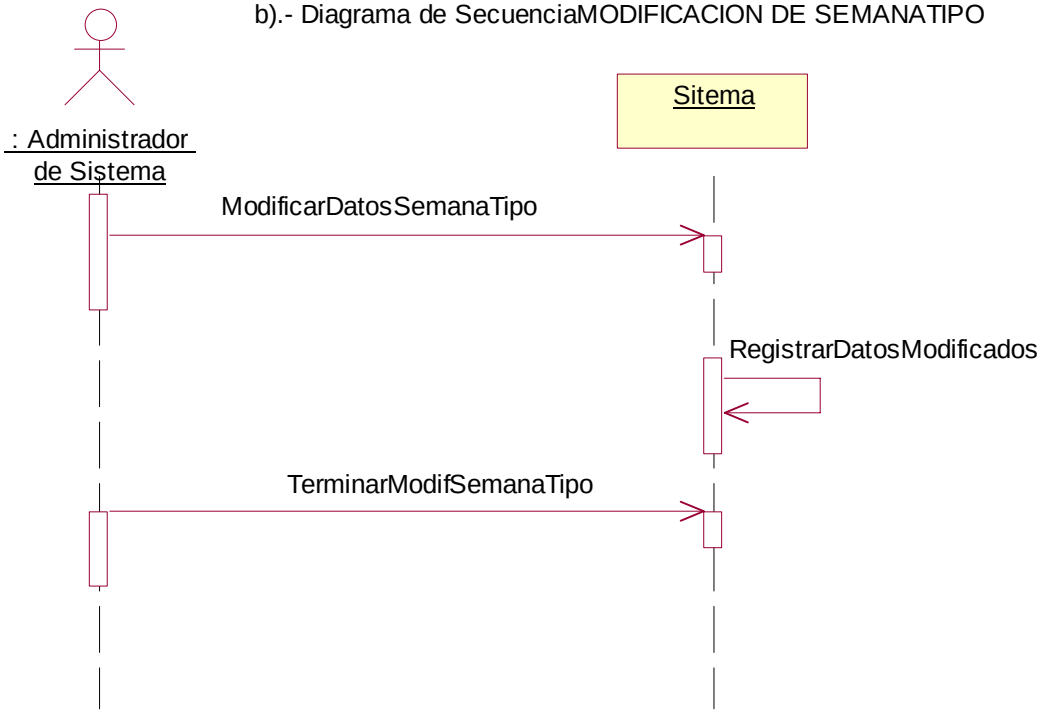
b).- Diagrama de Secuencia VALIDAR ACCESO-DATOS INCORRECTOS



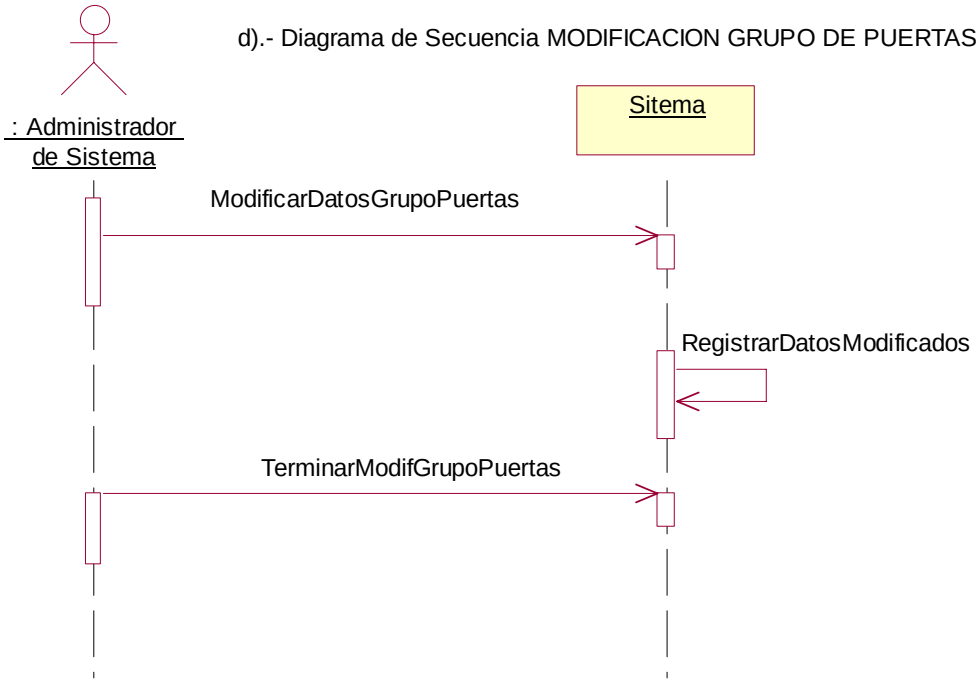
a).- Diagrama de Secuencia A ACTUALIZACION DE SISTEMA



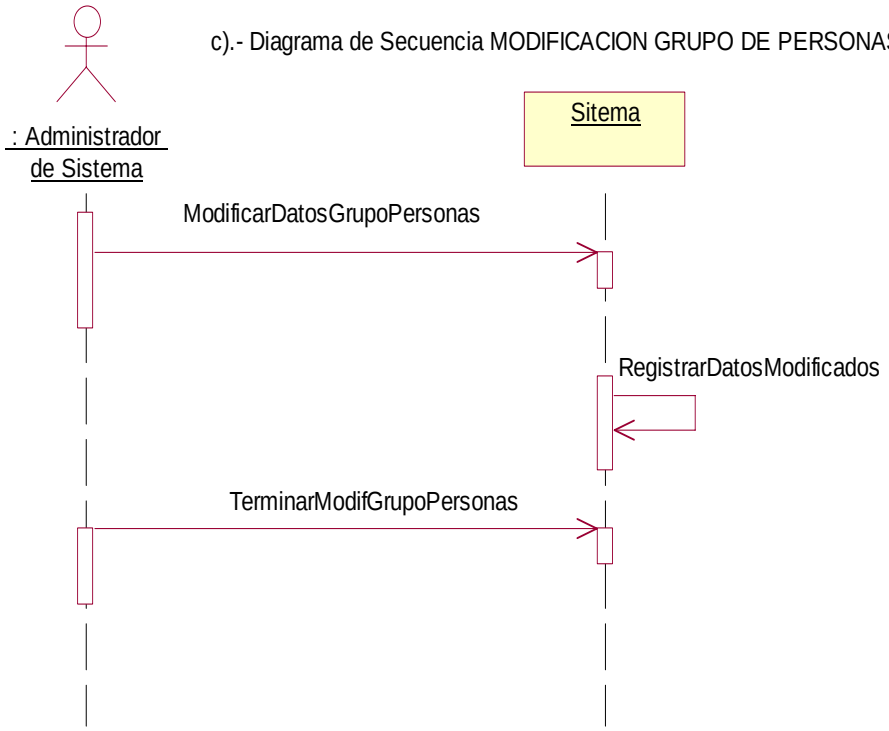
b).- Diagrama de SecuenciaMODIFICACION DE SEMANATIPO



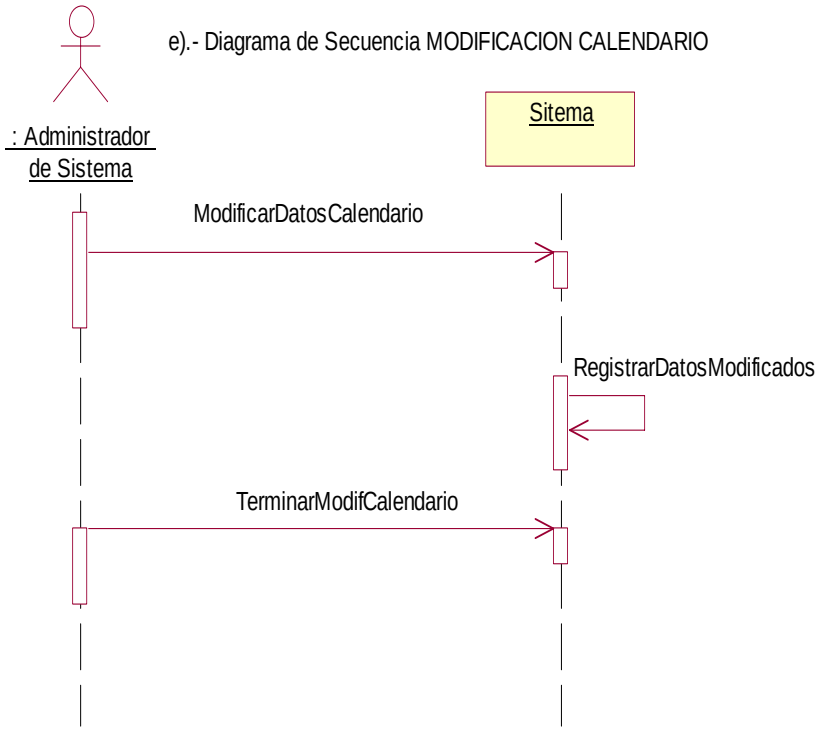
d).- Diagrama de Secuencia MODIFICACION GRUPO DE PUERTAS



c).- Diagrama de Secuencia MODIFICACION GRUPO DE PERSONAS

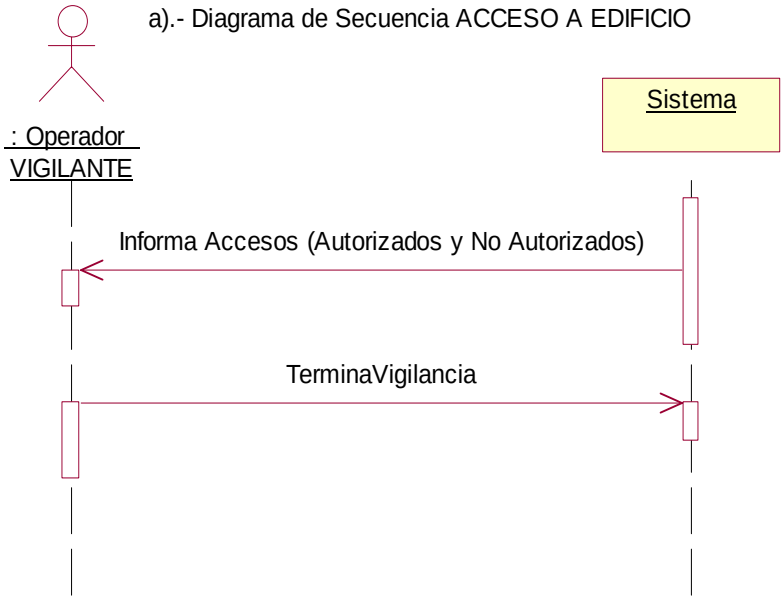


e).- Diagrama de Secuencia MODIFICACION CALENDARIO

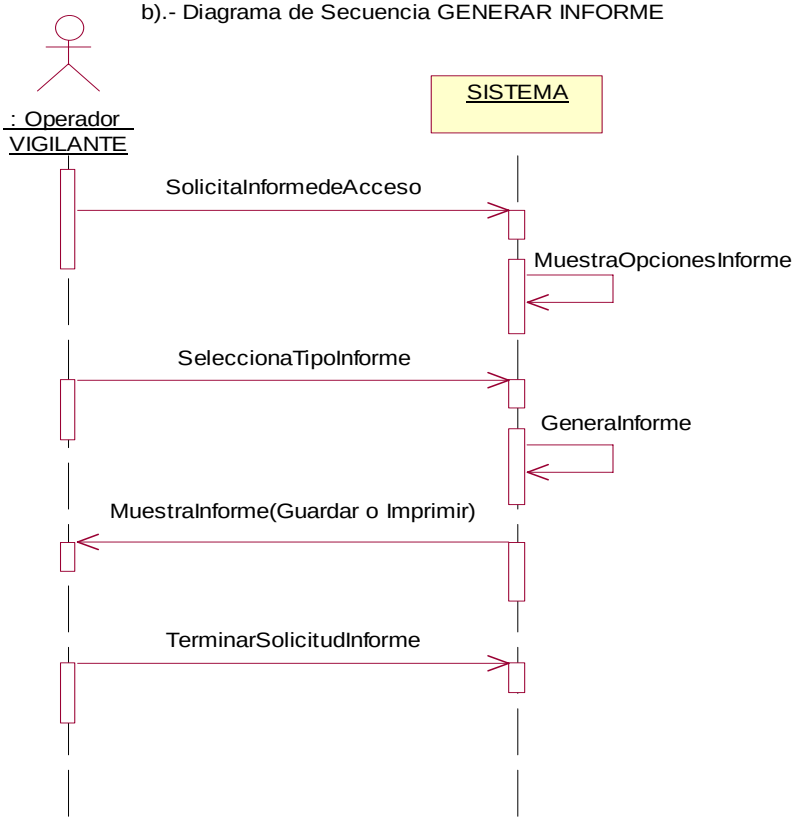


CASO DE USO “VIGILANCIA DE ACCESO AL EDIFICIO”

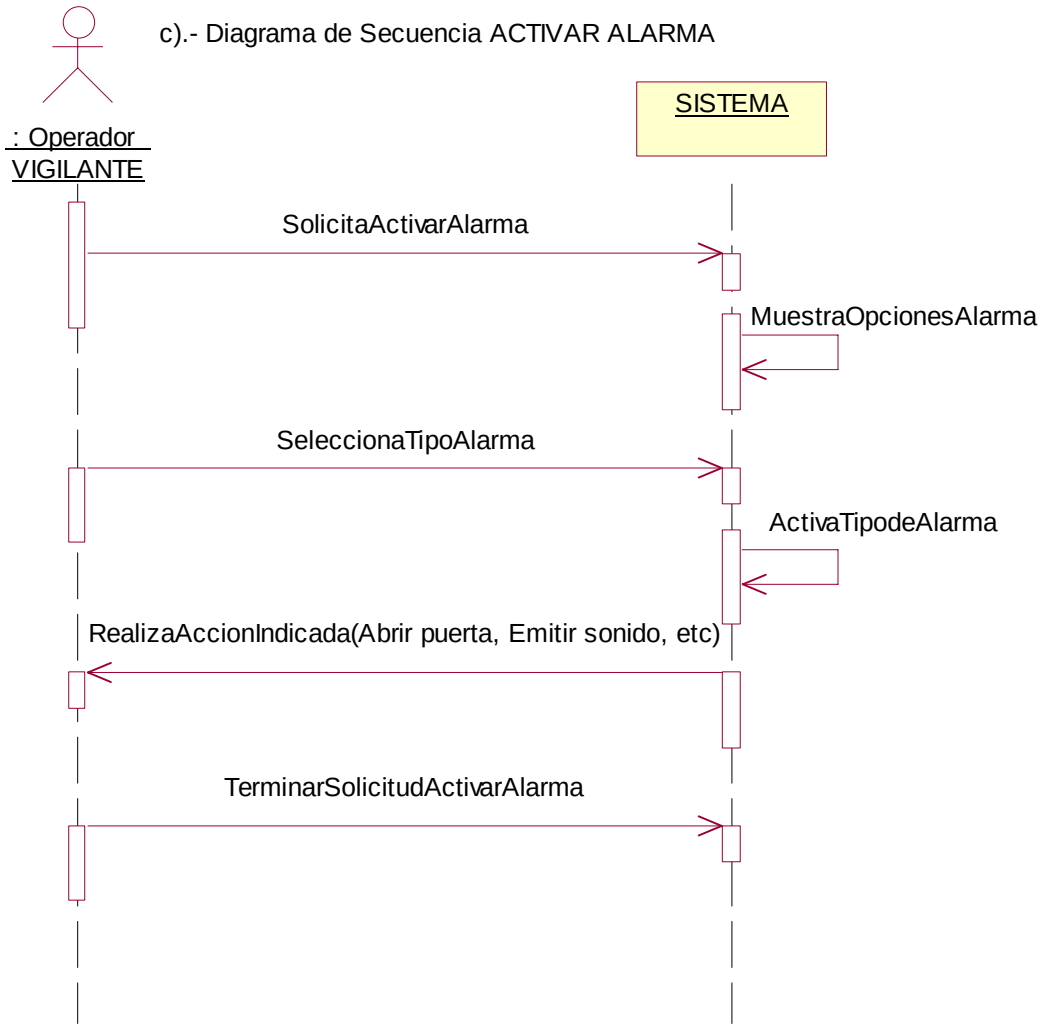
a).- Diagrama de Secuencia ACCESO A EDIFICIO



b).- Diagrama de Secuencia GENERAR INFORME



c).- Diagrama de Secuencia ACTIVAR ALARMA



CONTRATOS DE EVENTOS DE LOS DIAGRAMAS DE SECUENCIAS

CONTRATO(1)

1.- NOMBRE : Solicitar Acceso al Programa.()

2.- RESPONSABILIDAD: Correr el programa.

3.- TIPO: Sistema.

4.- REFERENCIAS CRUZADAS: Funciones del Sistema: R1.1

Caso de Uso: CLAVE DE ACCESO.

5.- NOTAS:

6.- EXCEPCIONES:

7.- SALIDA:

8.- PRECONDICIONES: El sistema se encuentra inactivo.

9.- POSCONDICIONES: El programa se cargo en memoria y muestra en pantalla solicitud de contraseña.

CONTRATO(2)

1.- NOMBRE : Introducir Contraseña (Clave).

2.- RESPONSABILIDAD: Introducir la contraseña de Acceso, validar y autorizar ingreso.

3.- TIPO: Sistema.

4.- REFERENCIAS CRUZADAS: Funciones del Sistema: R1.1

Caso de Uso : CLAVE DE ACCESO.

5.- NOTAS:

6.- EXCEPCIONES: Si la contraseña no es valida indicar que se cometió un error (acceso no permitido).

7.- SALIDA:

8.- PRECONDICIONES: El sistema conoce la contraseña.

9.- POSCONDICIONES: Se creo una instancia Edificio (Creación de instancia)

- Se creo una instancia Grupo de Puertas (Creación de instancia).
- Se creo una instancia Grupo de Personas (Creación de instancia).
- Se creo una instancia Semana Tipo (Creación de instancia).
- Se creo una instancia Puerta (Creación de instancia).
- Se asocio Semana Tipo con calendario (Asociación formada).
- Se asocio Grupo de Personas a Grupos de Puertas (Asociación Formada).
- Se asocio Puertas a Grupos de Puertas (Asociación Formada).
- Se asocio Lector de Tarjeta a Puerta (Asociacion Formada).

Actividad escribir los contratos para:

Definir_grupo_puertas, Definir_Grupo_de_personas, Define_semana_tipo

Respuestas

CONTRATO(3)

1.- NOMBRE : Definir_grupo_puertas (Cod._puerta, No._Grupo).

2.- RESPONSABILIDAD: Definir los grupos de puertas asociado a cada uno de los niveles y el sistema los almacena.

3.- TIPO: Sistema.

4.- REFERENCIAS CRUZADAS: Funciones del Sistema: R 1.2

Caso de Uso: Definición Inicial de Parámetros.

5.- NOTAS:

6.- EXCEPCIONES:

7.- SALIDA:

8.- PRECONDICIONES: El Sistema no conoce los grupos de puertas.

9.- POSCONDICIONES: Se han creado los grupos de puertas (creación de instancia).

- Se han asociado las puertas a un nivel de acceso y a un lector de tarjeta (asociación formada).

CONTRATO(4)

1.- NOMBRE : Definir Grupo de personas (Id_persona, Nombre, No. grupo, Descrip_persona)

2.- RESPONSABILIDA: Definir los grupos de personas asociado a cada grupo de puerta y el sistema los almacena

3.- TIPO :Sistema

4.- REFERENCIAS CRUZADAS: Funciones del sistema: R1.3

5.- NOTAS:

6.- EXCEPCIONES:

7.- SALIDA:

8.- PRECONDICIONES: El sistema no conoce a los grupos de personas

9.- POSCONDICIONES: Se han creado los grupos de personas (creación de instancia) de

- Se han asociado personas a diferentes grupos de personas (Asociación formada).
- Se han asociado grupos de personas a grupos de puertas y a lector de tarjeta.(Asociación formada)

CONTRATO(5)

1.- NOMBRE :Define semana tipo(cod_semana, dia, hora)

2.- RESPONSABILIDAD: Clasificar los tipos de semanas que regirán el acceso al edificio

3.- TIPO: Sistema

4.- REFERENCIAS CRUZADAS: Funciones del Sistema: R1.4

Casos de Uso: Definición inicial del Sistema

5.- NOTAS: Semanas con opción a modificaciones en base al calendario.

6.- EXCEPCIONES:

7.- SALIDA:

8.- PRECONDICIONES: El sistema no conoce las semanas tipos ni los días contenidos en ellas.

9.- POSCONDICIONES: Se crearon semanas tipos (creación de instancia)

- Se asociaron semanas tipos con calendarios(Asociación formada)

Laboratorio IV

Guía de laboratorio

14.3.4 Tema: II Caso de Uso, Caso de Uso Múltiple, Caso de Uso Expandidos

14.3.5 Competencia o habilidad

Este laboratorio pretende que usted adquiera la competencia necesaria para "Identificar las modificaciones a los diagramas de casos de uso, usando las opciónes extend, uses"

14.3.6 Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

- Interpretar los diagrama de caso de uso múltiple y expandido.
- Dibujar diagrama caso de uso múltiple y expandido.
- Interpretar contratos de los casos de uso expandido

14.3.7 Actividades de Aprendizaje

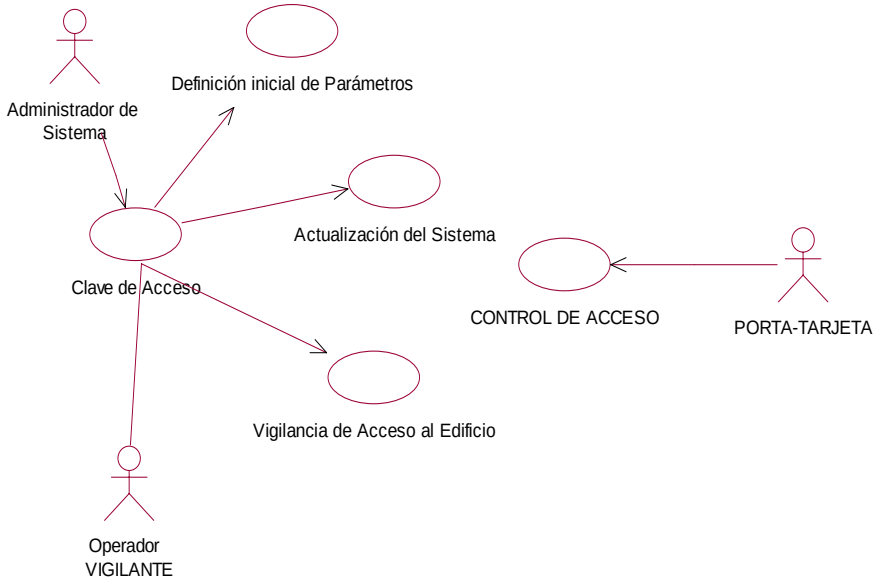
1. Revise el diagrama de casos de uso.
2. Interprete los diagrama casos de uso.
3. Dibujar los diagramas casos de uso en el modelo UML
4. Interpretar los diagramas de casos de uso anotando sus observaciones a favor y en contra
5. Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas
6. Tiempo aproximado 2 horas
7. Al final se realizará la evaluación del laboratorio, mediante la entrega de su diagrama de caso de uso y sus contratos.

Actividad de laboratorio: graficar los diagrama de caso de uso usando el programa Rational Rose además interpretar para su confrontación.

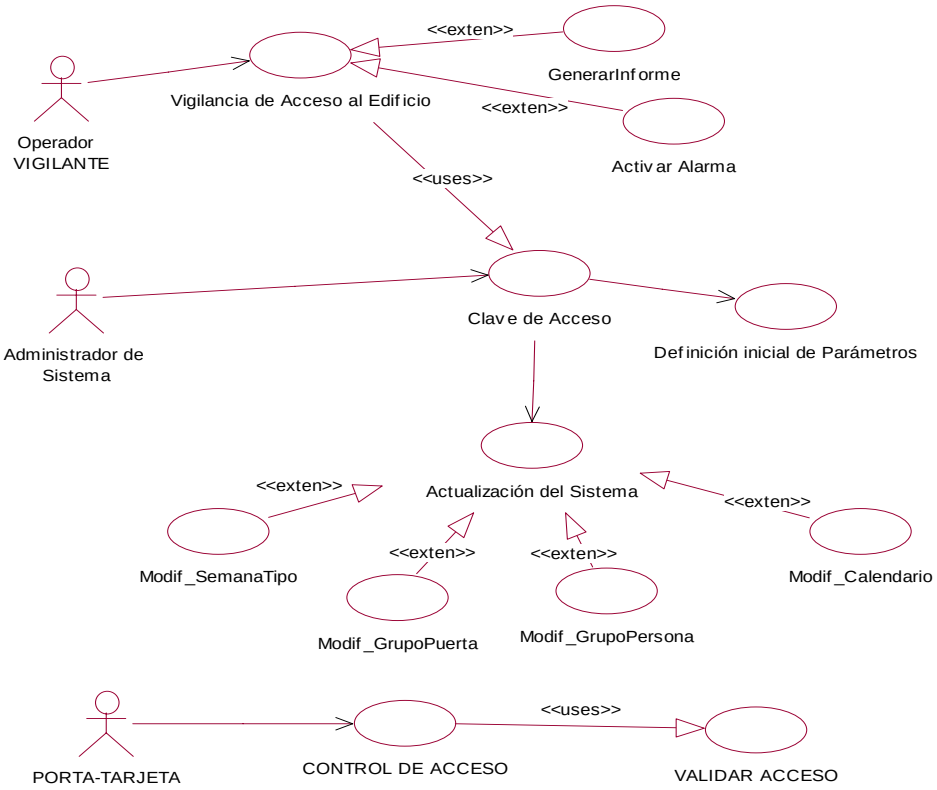
II. CASOS DE USO

a).-DIAGRAMA DE CASO DE USO FASE I

.



b).-DIAGRAMA DE CASOS DE USO MULTIPLES



c).- CASOS DE USO

Caso de uso: Vigilancia de Acceso al Edificio

Actores: Vigilante

Resumen: El Vigilante realiza el control de los intentos de paso infructuosos a través de una pantalla y puede generar informes de los ingresos a los grupos de puertas y los diferentes permisos, lo mismo que activar alarmas de emergencia.

Curso Normal de los Eventos

Acción de Actores	Respuesta del sistema
1. Este Caso de uso inicia cuando el vigilante ingresa al sistema, para observar los distintos accesos al edificio	2. El sistema muestra las pantallas de vigilancia y las opciones para generar informes y activar alarmas de emergencias
3. Si el vigilante desea generar cualquier informe, entonces inicia **Informes**	
4. Si el vigilante es avisado que hay una emergencia entonces inicia, **Activar Alarmas**	
5. Si el vigilante desea terminar la sección de vigilancia y selecciona terminar	6. El Sistema se cierra

Caso de uso Relacionados

Usa Informes

Usa Activar Alarmas

Caso uso : Generar Informes

Actores: Vigilante

Resumen: El vigilante selecciona y genera diferentes tipos de informes de acceso, permisos y parámetros.

Curso Normal de los Eventos

Acciones de Actores	Respuesta del sistema
1. Este caso de uso inicia cuando el vigilante selecciona la opción Generar Informe.	2. El sistema muestra los diferentes tipos de informes predefinidos que se pueden generar.
3. El vigilante selecciona un tipo de informe.	4. Muestra el informe en pantalla con la opción imprimir o Guardar.
5. Selecciona la opción Imprimir o Guardar	6. Imprime o Almacena el Informe

Cursos Alternos:

Sección 5: El Vigilante no desea Imprimir el informe y puede terminar la acción de generar informe (Guardar).

Actividad de laboratorio: crear los siguientes Casos de Uso Extendidos
Activar Alarma, Actualizar Sistema, Modificar Grupo de Puertas, Modificar Grupo de Personas.

Respuestas

CASO DE USO: Activar Alarma

AUTORES: Vigilante

RESUMEN: El Vigilante en el Centro de Vigilancia Activa una alarma.

CURSO NORMAL DE LOS EVENTOS

Acciones de Actores	Respuesta del sistema
1. Este caso de uso inicia cuando el vigilante selecciona la opción Activar alarma.	2. El sistema muestra los diferentes tipos de alarmas
3. El vigilante selecciona un tipo de alarma a activar (incendio, terremotos, etc.).	4. Activa la alarma seleccionada
	5. Realiza la actividad correspondiente con la alarma activada (Abrir o Cerrar puertas, generar sonidos, etc).
6. El vigilante elige finalizar activar alarma	7. Finaliza Activar Alarma.

CASO DE USO: Actualizar Sistema

AUTORES: Administrador

RESUMEN: El Administrador elige la opción actualizar sistema , realiza las modificaciones necesarias, el sistema registra las modificaciones realizadas en el sistema activo del centro de vigilancia.

CURSO NORMAL DE LOS EVENTOS

ACCION DE LOS ACTORES	RESPUESTA DEL SISTEMA
1.- Este caso de uso inicia cuando el Administrador elige la opción Actualizar Sistema	2.- El sistema muestra las diferentes opciones de actualizar sistema que se encuentran predefinidas.
3.- El Administrador elige una de las opciones de actualizar sistema a. Si el Administrador quisiera Modificar Grupo de Puertas , activa la opción Modificar Grupo de Puertas. b. Si el Administrador quisiera Modificar Grupo de Personas , activa la opción Modificar Grupo de Personas. c. Si el Administrador quisiera Modificar Calendario , activa la opción Modificar Calendario. d. Si el Administrador quisiera Modificar Semana Tipo, activa la opción Modificar Semana Tipo .	
4.- El administrador elige abandonar la opción de Actualizar Sistema y sale	5.- El Sistema Cierra la opción Actualizar Sistema.

CASO DE USO: Modificar Grupo de Puertas

AUTORES: Administrador

RESUMEN El Administrador realiza la actualización de Grupo de Puertas el sistema ubicado en el centro de Vigilancia los registra y almacena

.CURSO NORMAL DE LOS EVENTOS

ACCION DE LOS ACTORES	RESPUESTA DEL SISTEMA
1.-Este caso de uso inicia cuando el Administrador selecciona la opción Modificar Grupo de Puertas .	2.- El sistema muestra la ventana de Modificar Grupo de Puertas con sus respectivas alternativas.
3.-El Administrador realiza las modificaciones que corresponda a cada Grupo de Puertas	4.- El sistema registra y almacena las modificaciones realizadas.
5.- El Administrador indica al sistema que ha finalizado la modificacion de grupo de puertas y que debe abandonar (cerrar) dicha opción.	6.-El sistema cierra la opción Modificar Grupo de Puertas y se mantiene activo en Actualizar Sistema

CASO DE USO: Modificar Grupo de Personas

AUTORES: Administrador

RESUMEN: El Administrador realiza la actualización de Grupo de Personas en el sistema ubicado en el centro de Vigilancia, registra y almacena dichas actualizaciones

CURSO NORMAL DE LOS EVENTOS

ACCION DE LOS ACTORES	RESPUESTA DEL SISTEMA
1.-Este caso de uso inicia cuando el Administrador selecciona la opción Modificar Grupo de Personas .	2.- El sistema muestra la ventana de Modificar Grupo de Personas con sus respectivas alternativas.
3.-El Administrador realiza las modificaciones que corresponda a cada Grupo de Personas	4.- El sistema registra y almacena las modificaciones realizadas.
5.- El Administrador indica al sistema que ha finalizado la modificación de grupo de personas y que debe abandonar (cerrar) dicha opción.	6.-El sistema cierra la opción Modificar Grupo de Personas y se mantiene activo en Actualizar Sistema

14.4 Laboratorio V

Guía de laboratorio

14.4.1 Tema:Caso Reales de Uso

14.4.2 Competencia o habilidad

Este laboratorio pretende que usted adquiera la competencia necesaria para "Interpretar los Casos de Uso"

Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

- Interpretar los diagrama de Caso Reales de Uso.
- Dibujar diagrama caso de Uso Reales.

Actividades de Aprendizaje

1. Revise el diagrama de Casos Reales de Uso.
2. Interprete los diagrama Casos de Uso.
3. Dibujar los diagramas Casos de Uso en el modelo UML
4. Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas.
5. Tiempo aproximado 2 horas
6. Al final se realizará la evaluación del laboratorio, mediante la entrega de su diagrama de caso uso reales.

Desarrollo

Caso Real de Uso: Clave de Acceso

Caso de Uso : Clave de Acceso

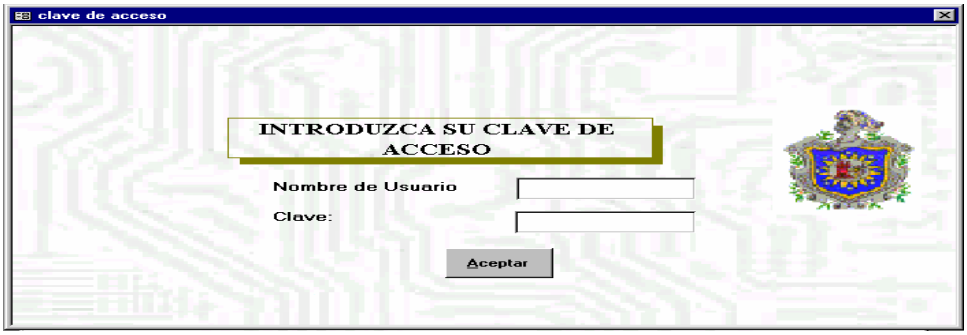
Actores : Administrador y Operador o Vigilante.

Propósito : Capturar la contraseña del Administrador y del Operador o Vigilante para que puedan entrar al sistema.

Resumen : El Administrador y el Vigilante llegan al sistema, encienden el equipo , corren el software , introducen su clave, el sistema valida la contraseña y permite el ingreso en caso que la clave sea correcta.

Tipo : Primario y esencial.

Referencias Cruzadas: Funciones de R1.1 .



Curso normal de los eventos

Acción del Actor	Respuesta del Sistema
1 Este caso de uso inicia cuando el Administrador o el Vigilante se presenta al centro de Vigilancia, enciende el equipo y corren el programa.	2 Muestra pantalla para capturar palabra clave .
3 El Administrador o Vigilante ingresan su nombre de usuario y su palabra clave.	4 El sistema valida el nombre de usuario y Palabra clave y en caso que sea correcta permite el acceso, de lo contrario envía un mensaje de error.

Caso Expandido de Uso: Definición inicial de parámetros

Caso de Uso :Definición Inicial de Parámetros

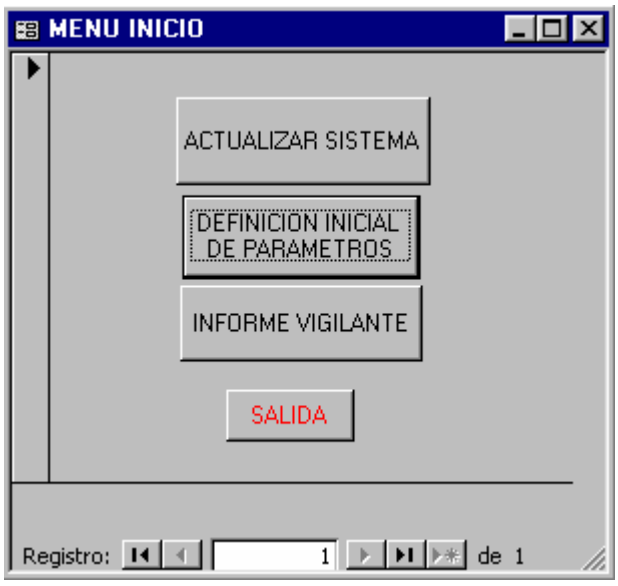
Actores :Administrador del Sistema

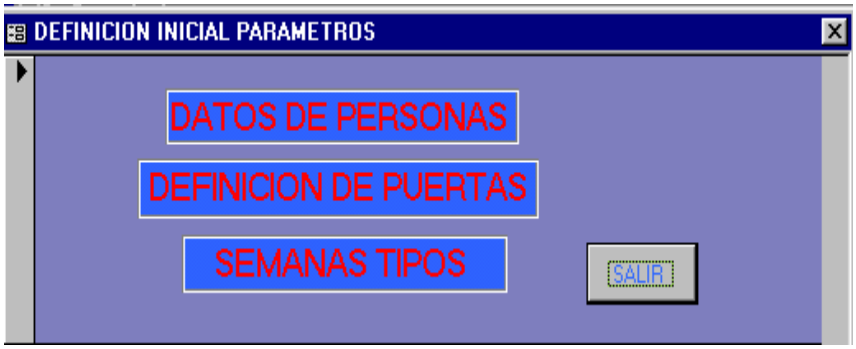
Propósito : Permitir que el Administrador defina: Grupo de Personas, Grupo de Puertas, Semanas Tipo, Asignar Permisos de Personas.

Resumen : El Administrador ingresa al programa , asigna los permiso de usuario, define los grupos de personas, define los grupos de puertas a los cuales tienen acceso, define las semanas tipos según calendario.

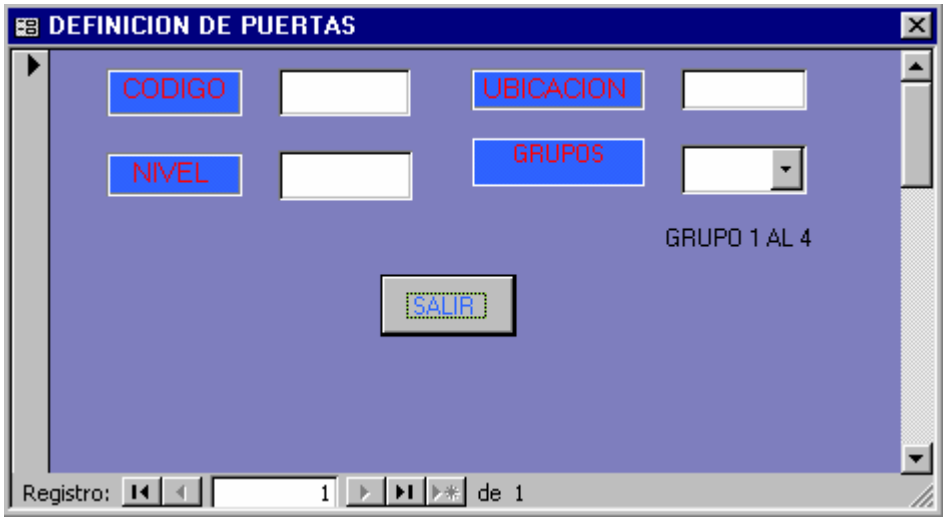
Tipo : Primario y esencial

Referencias Cruzadas: Funciones de R1.1, R1.2, R1.3, R1.4.


MENU INICIO
ACTUALIZAR SISTEMA
DEFINICION INICIAL DE PARAMETROS
INFORME VIGILANTE
SALIDA
Registro:
1
de 1


DEFINICION INICIAL PARAMETROS
DATOS DE PERSONAS
DEFINICION DE PUERTAS
SEMANAS TIPOS
SALIR

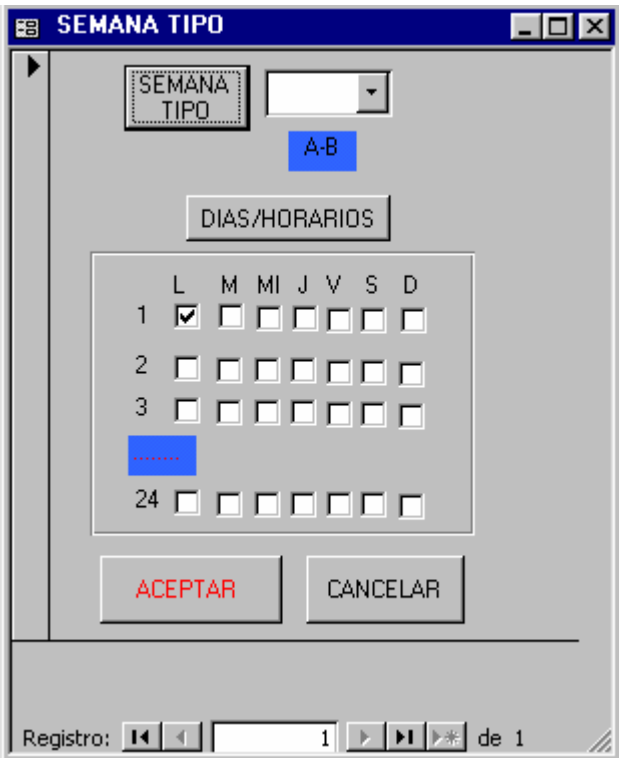

DATOS DE PERSONA
NOMBRE
APELLIDOS
DIRECCION
EDAD
SEXO
CARGO
DERECHOS
OPCIONES
GRUPO NO. 1
GRUPO NO. 2
GRUPO NO. 3
GRUPO NO. 4
ACEPTAR
CANCELAR
Registro:
1
de 1


DEFINICION DE PUERTAS
CODIGO
UBICACION
NIVEL
GRUPOS
GRUPO 1 AL 4
SALIR
Registro:
1
de 1


SEMANA TIPO
DEFINIR CALENDARIO
DEFINIR DIAS FERIADOS
DEFINIR SEMANA TIPO
SALIR
Registro:
1
de 1


CALENDARIO
AÑO
MES
SEMANA
TIPO DE SEMANA
1 - 52
A - B
ACEPTAR
CANCELAR
Registro:
1
de 1


DIAS FERIADOS
SEMANA
DE LA 1 A LA 52
DIA
L - V
HORARIO
HORARIOS
1
2
3
12
13
14
15
24
ACEPTAR
CANCELAR
Registro:
1
de 1


SEMANA TIPO
SEMANA TIPO
A-B
DIAS/HORARIOS
L M MI J V S D
1
2
3
24
ACEPTAR
CANCELAR
Registro:
1
de 1

Curso normal de los eventos

Acción del Actor	Respuesta del Sistema
1.- este caso de uso inicia cuando el administrador solicita al sistema definir los parámetros iniciales del sistema (ventana menú inicio)	2.-Muestra la ventana de interfaz numero 1 (Definición Inicial)
3.-En caso de seleccionar la opción Defin. Inicic. Parámetros	4.-Muestra la ventana Datos de Persona
5.- En caso de escoger la opción Definir Puertas	6.- Muestra la ventana Definir Puerta
7.- En caso de seleccionar la opción Semana tipo	8.-Muestra la ventana Semana Tipo , en la cual se pueden seleccionar tres sub opciones.
9.-En caso de seleccionar la sub opción Definir Calendario	10.-Muestra la ventana Calendario
11.- En caso de seleccionar la sub opción Definir Días Feriados	12.-Muestra la ventana Dias feriados
13.-En caso de seleccionar la sub opción Definir Semana Tipo	14.-Muestra la ventana Semana Tipo
15.- Al considerarse terminadas las definiciones de cada uno de las opciones existentes y utilizadas , debe indicarse que el proceso Definición de parámetros a finalizado	16.- Retorna al menú principal.

Caso Real de Uso: Actualización de Sistema

Caso de Uso: Actualización de Sistema.

Actores: Administrador

Propósito: Permitir que el Administrador modifique los grupos de personas, grupos de puertas, las semanas, tipos y modifique los permisos de personas

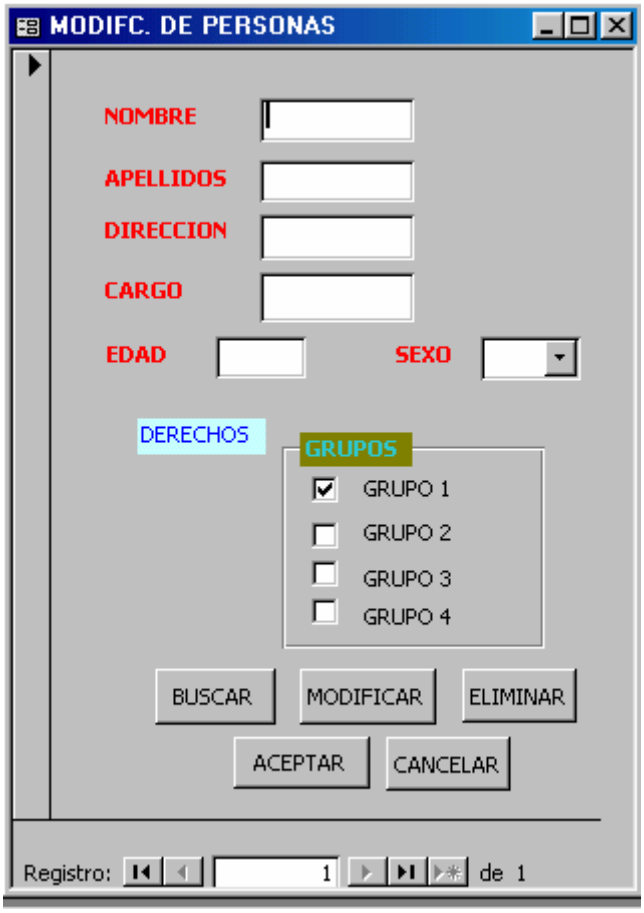
Resumen: El Administrador ingresa al sistema y realiza las modificaciones de derechos para los grupos de puertas, grupos de personas, modifica los permisos asignados a los usuarios y modifica las semanas tipos en dependencia de la variación de horarios

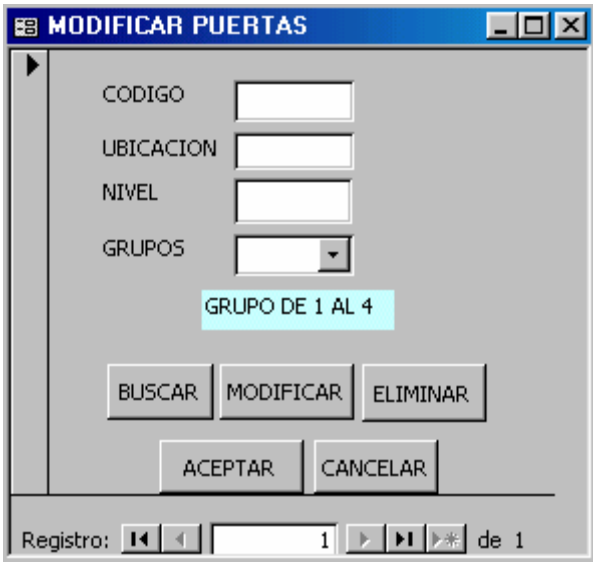
Tipo: Primario.

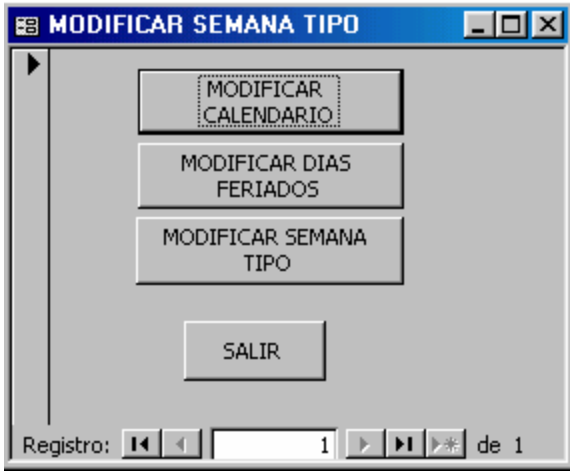
Referencias Cruzadas: Funciones de R1.5 .


MENU INICIO
DEFINICION INICIAL DE PARAMETEROS
ACTUALIZACION DEL SISTEMA
INFORME DE VIGILANCIA
SALIR
Registro: 1 de 1


ACTUALIZAR SISTEMA
MODIFIC. PERSONAS
MODIFICAR PUERTAS
MODIFICAR SEMANA TIPO
SALIR
Registro: 1 de 1

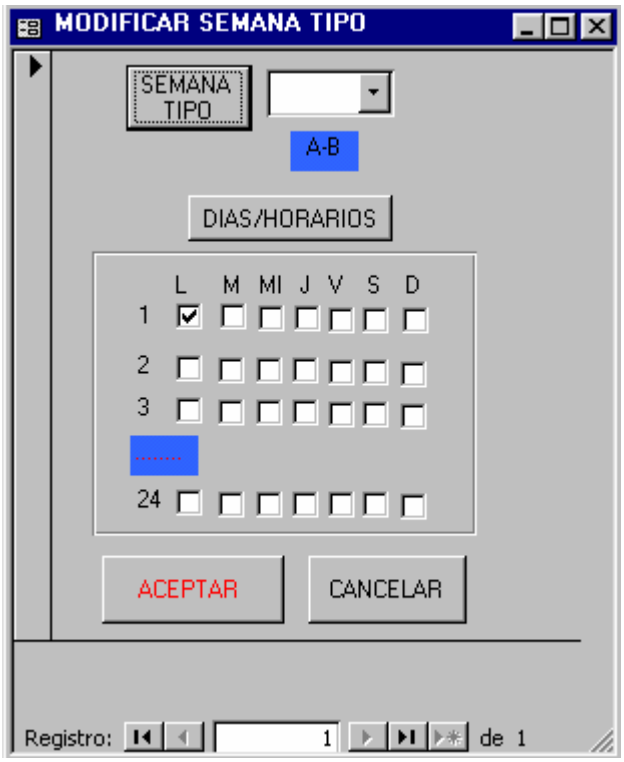

MODIFC. DE PERSONAS
NOMBRE
APELLIDOS
DIRECCION
CARGO
EDAD
SEXO
DERECHOS
GRUPOS
GRUPO 1
GRUPO 2
GRUPO 3
GRUPO 4
BUSCAR
MODIFICAR
ELIMINAR
ACEPTAR
CANCELAR
Registro:
1
de 1


MODIFICAR PUERTAS
CODIGO
UBICACION
NIVEL
GRUPOS
GRUPO DE 1 AL 4
BUSCAR
MODIFICAR
ELIMINAR
ACEPTAR
CANCELAR
Registro:
1
de 1


MODIFICAR SEMANA TIPO
MODIFICAR CALENDARIO
MODIFICAR DIAS FERIADOS
MODIFICAR SEMANA TIPO
SALIR
Registro:
1
de 1


MODIFCAR CALENDARIO
AÑO
SEMANA
MES
Cuadro con
BUSCAR
MODIFICAR
ELIMINAR
ACEPTAR
CANCELAR
Registro:
1
de 1


MODIFICAR DIAS FERIADOS
SEMANA
DIAS
HORA
1
2
3
.....
12
13
14
15
.....
24
BUSCAR
MODIFICAR
ELIMINAR
ACEPTAR
CANCELAR
Registro:
1
de 1


MODIFICAR SEMANA TIPO
SEMANA TIPO
A-B
DIAS/HORARIOS
L M MI J V S D
1
2
3
.....
24
ACEPTAR
CANCELAR
Registro:
1
de 1

Curso normal de los eventos

ACCION DEL ACTOR	RESPUESTA DEL SISTEMA
1.- Este caso de uso inicia cuando el administrador se introduce al sistema para modificar o actualizar el sistema (parámetros) (ventana menú inicio)	2.-Muestra la ventana de interfaz numero 1 (Actualización del sistema)
3.-En caso de seleccionar la opción Modificar Persona	4.-Muestra la ventana Modificar Datos de Persona
5.- En caso de escoger la opción Modificar Puertas	6.- Muestra la ventana Modificar Puerta
7.- En caso de seleccionar la opción Modificar Semana tipo	8.-Muestra la ventana Modificar Semana Tipo , en la cual se pueden seleccionar tres sub opciones (Modificar Calendario, Modificar dias feriados, modificar semana tipo).
9.-En caso de seleccionar la sub opción Modificar Calendario	10.-Muestra la ventana Modificar Calendario
11.- En caso de seleccionar la sub opción Modificar Días Feriados	12.-Muestra la ventana Modificar Días feriados
13.-En caso de seleccionar la sub opción Modificar Semana Tipo	14.-Muestra la ventana Modificar Semana Tipo
15.- Al considerarse terminadas las modificaciones de cada uno de las opciones existentes y utilizadas , debe indicarse que el proceso Modificar o actualizar sistema a finalizado.	16.- Retorna al menú principal.

14.5 Laboratorio VI

Guía de laboratorio

14.5.1 Tema: Diagrama de Colaboración, Diagramas de Clase, Paquetes

14.5.2 Competencia o habilidad

Este laboratorio pretende que usted adquiera la competencia necesaria para "Interpretar los diagramas colaboración el diagrama de clase y los paquetes"

14.5.3 Objetivos de Aprendizaje

Al concluir el estudio de este laboratorio, usted podrá:

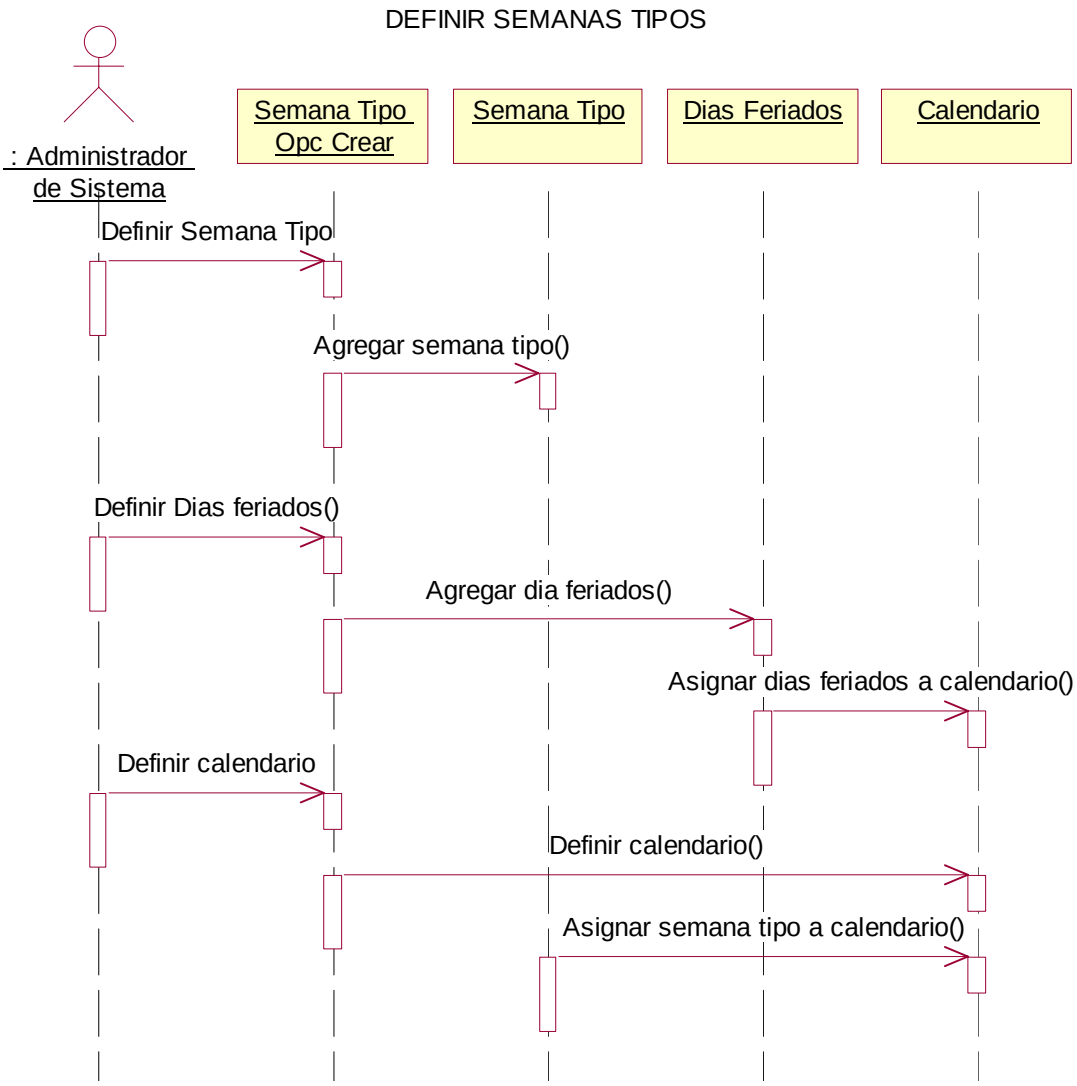
Interpretar los Diagrama de Colaboración , Diagrama de Clase y Diagrama de Paquetes.

Dibujar diagrama Diagramas de Colaboración, Diagrama de Clase y Diagrama de Paquetes.

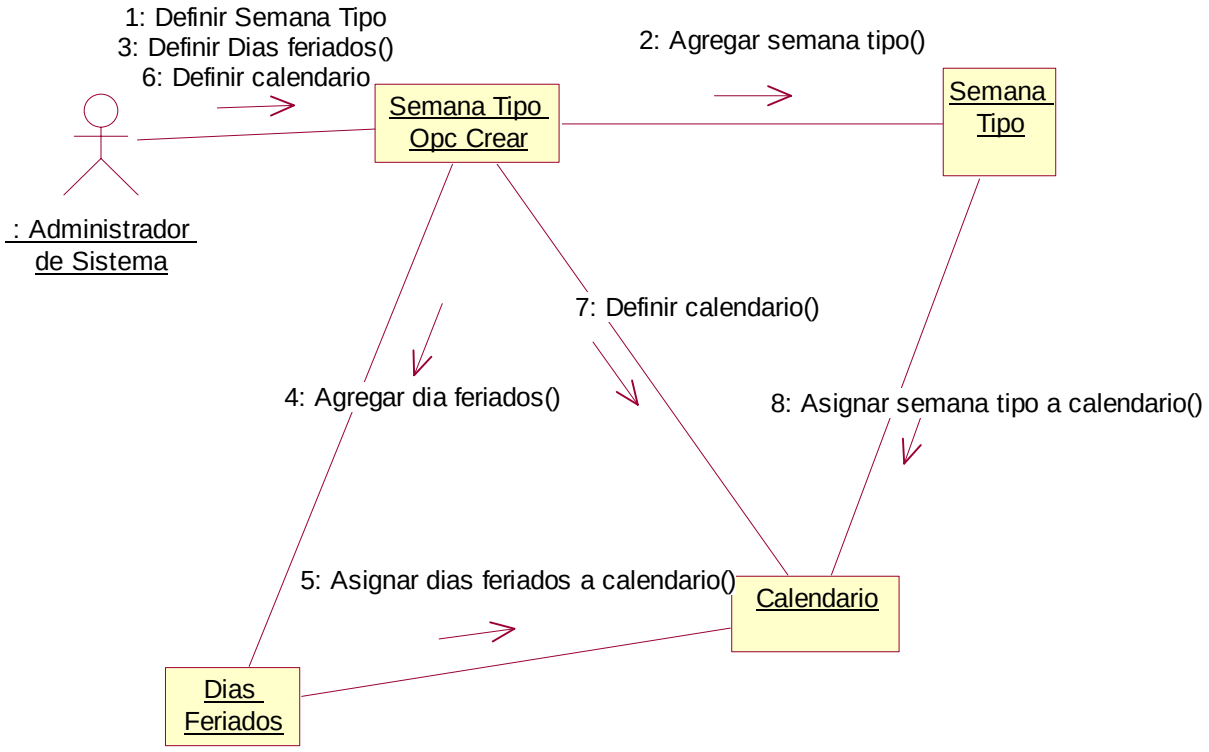
Actividades de Aprendizaje

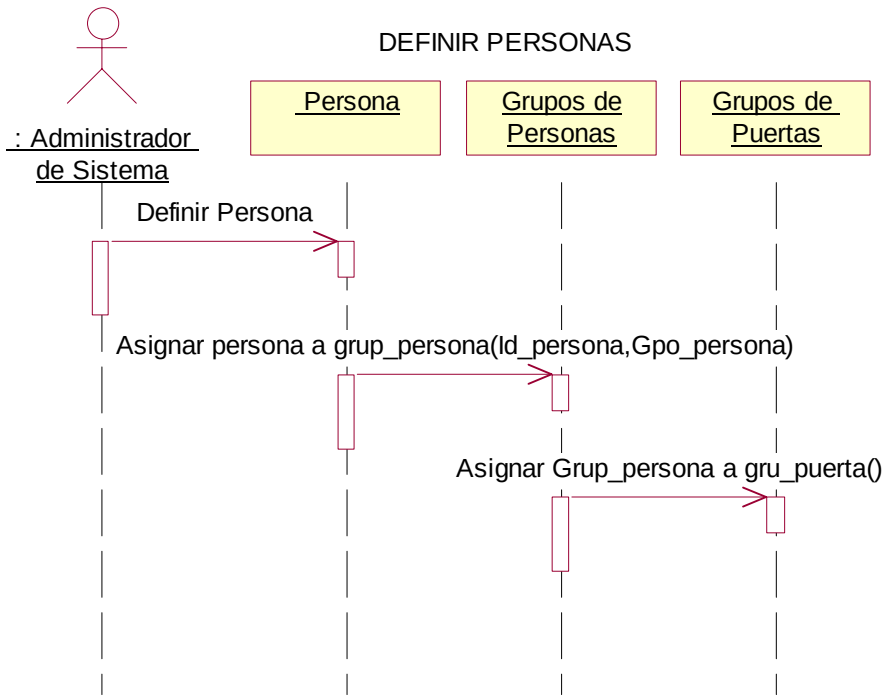
1. Revise los diagrama de colaboración , diagrama de clase y diagrama de paquetes.
2. Interprete los diagrama de colaboración , diagrama de clase y diagrama de paquetes.
3. Dibujar los diagrama de colaboración , diagrama de clase y diagrama de paquetes.
4. Fomentar la participación activa del grupo mediante la discusión del diseño usando la confrontación de idea unos a favor del análisis y otros en contra, argumentando sus ideas
5. Tiempo aproximado 6horas
6. Al final se realizará la evaluación del laboratorio, mediante la entrega de su Diagrama de Colaboración , Diagrama de Clase y Diagrama de Paquetes elaborados en Rational Rose.

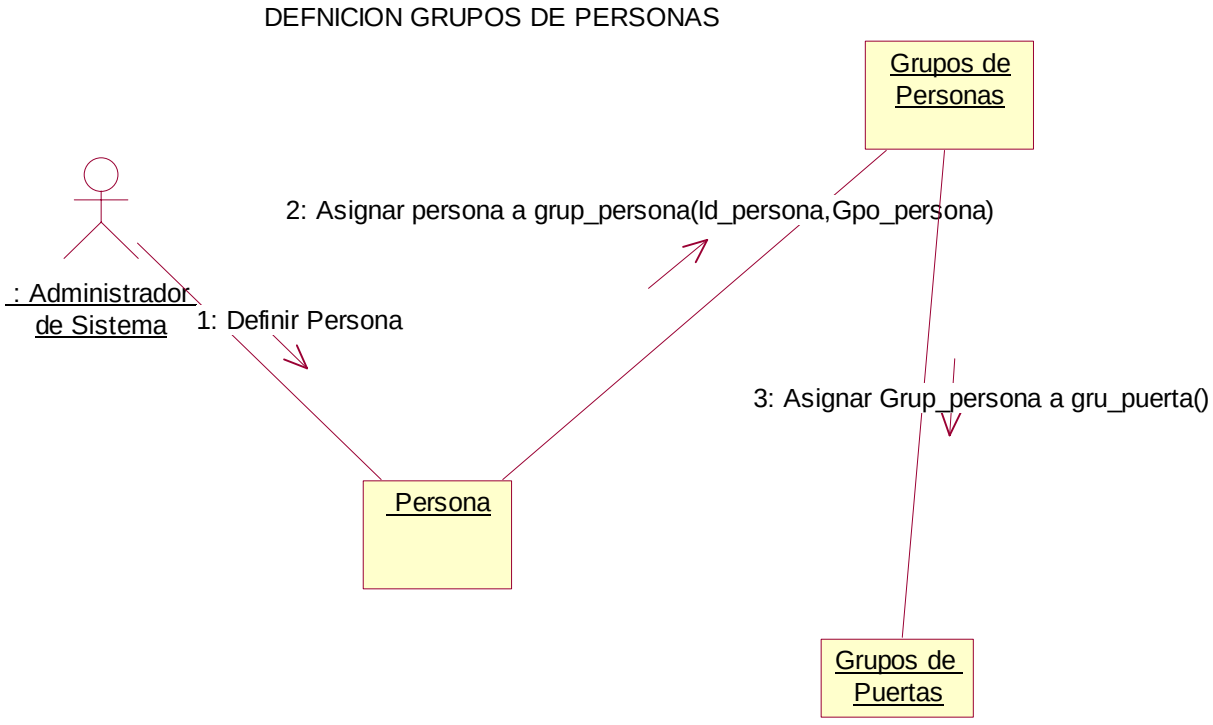
CASO DE USO DEFINICION INICAL DE PARAMETROS

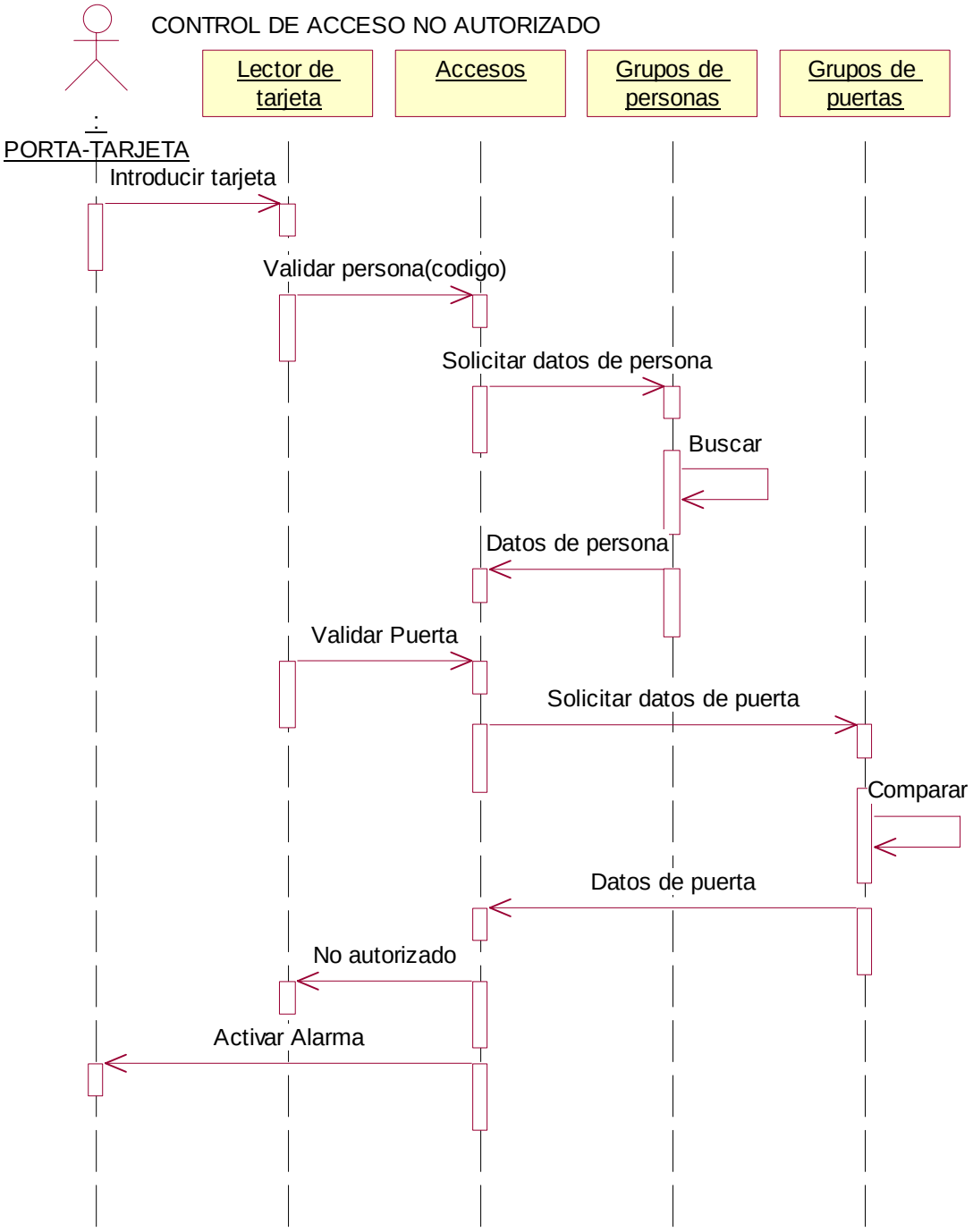


DEFINICION DE SEMANA TIPO

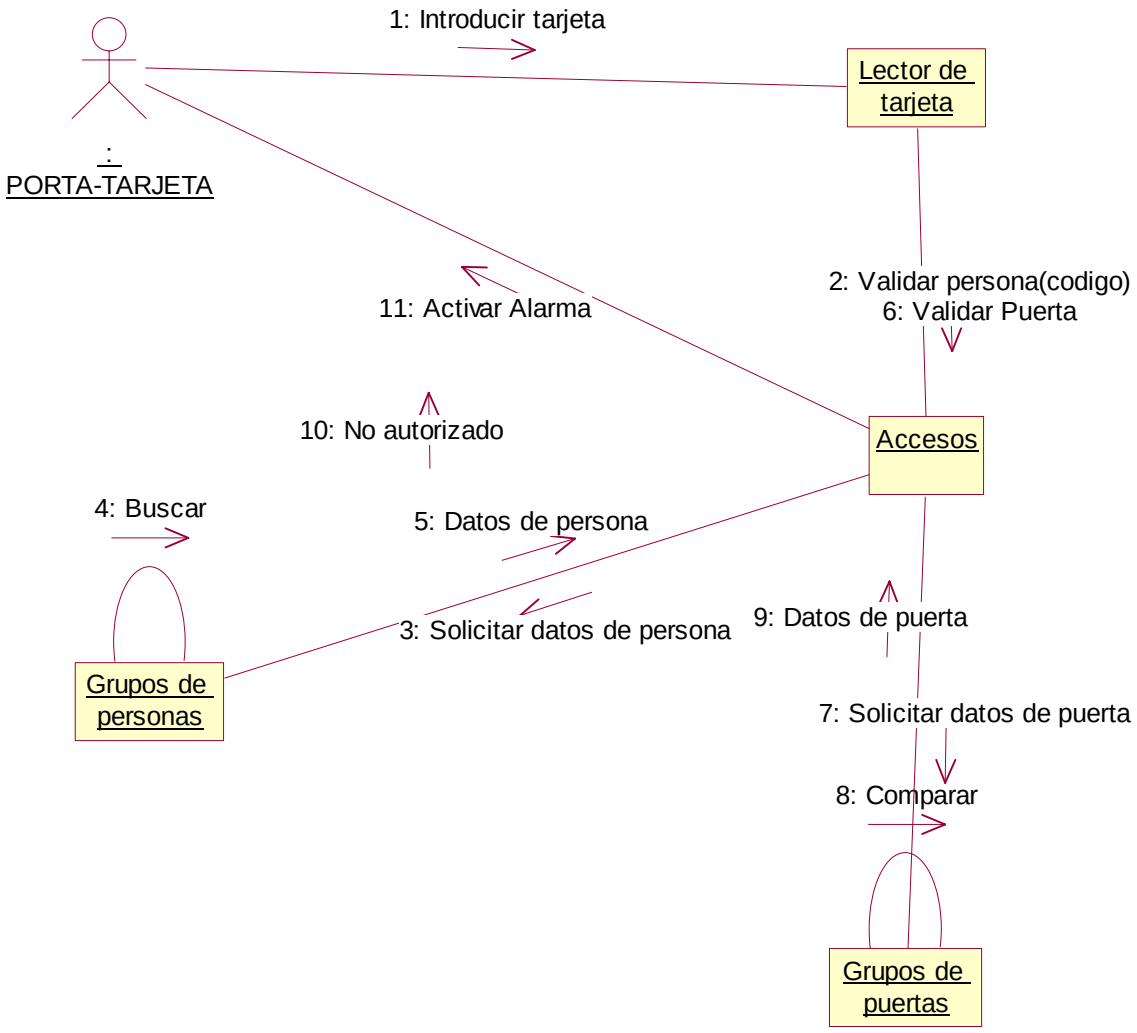



DEFINIR PERSONAS
: Administrador de Sistema
Persona
Grupos de Personas
Grupos de Puertas
Definir Persona
Asignar persona a grup_persona(Id_persona,Gpo_persona)
Asignar Grup_persona a gru_puerta()

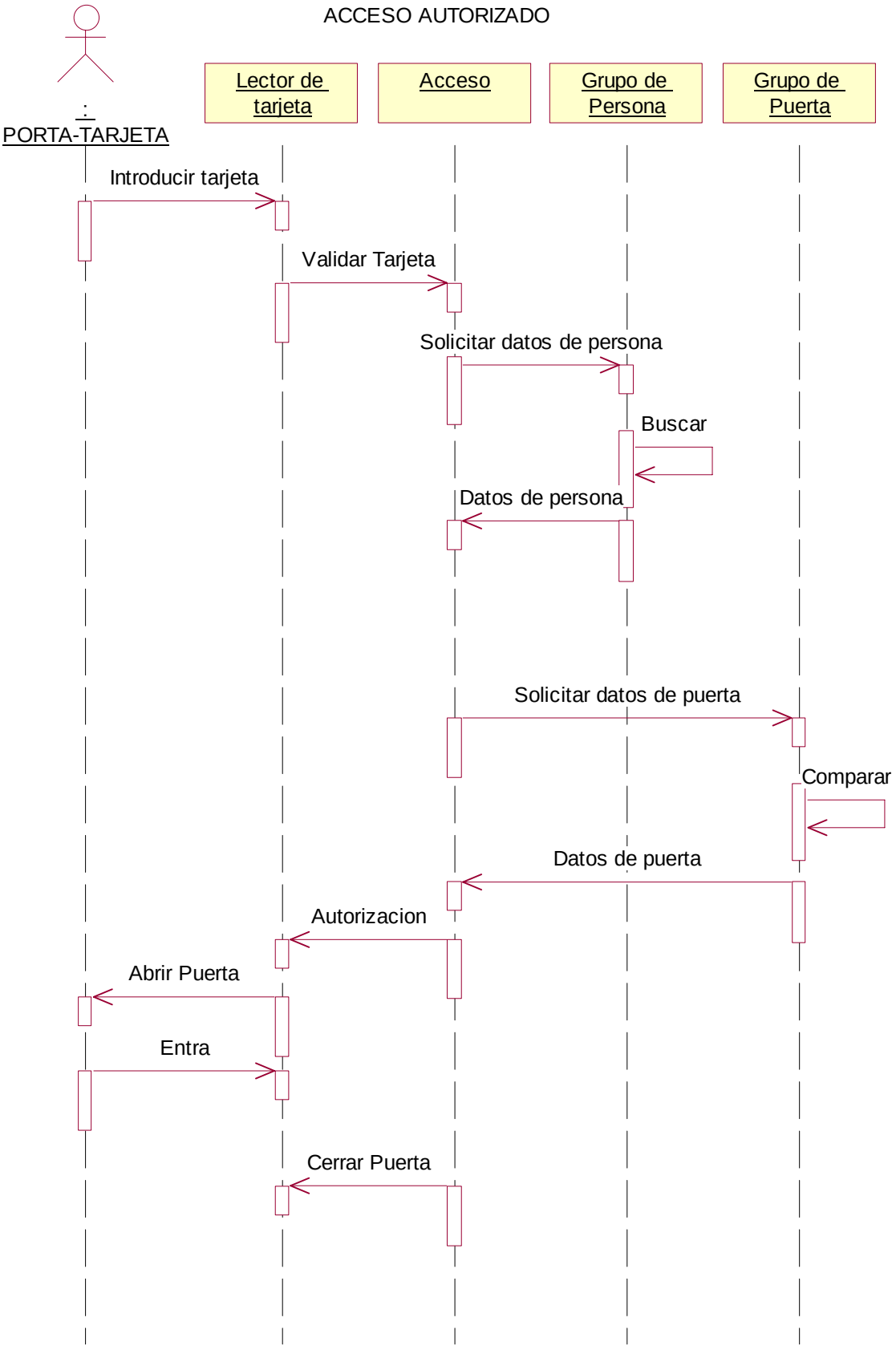

DEFNICION GRUPOS DE PERSONAS
Grupos de Personas
: Administrador de Sistema
2: Asignar persona a grup_persona(Id_persona,Gpo_persona)
1: Definir Persona
3: Asignar Grup_persona a gru_puerta()
Persona
Grupos de Puertas


CONTROL DE ACCESO NO AUTORIZADO
Lector de tarjeta
Accesos
Grupos de personas
Grupos de puertas
:
PORTA-TARJETA
Introducir tarjeta
Validar persona(codigo)
Solicitar datos de persona
Buscar
Datos de persona
Validar Puerta
Solicitar datos de puerta
Comparar
Datos de puerta
No autorizado
Activar Alarma

CONTROL DE ACCESO NO AUTORIZADO



CASO DE USO CLAVE DE ACCESO



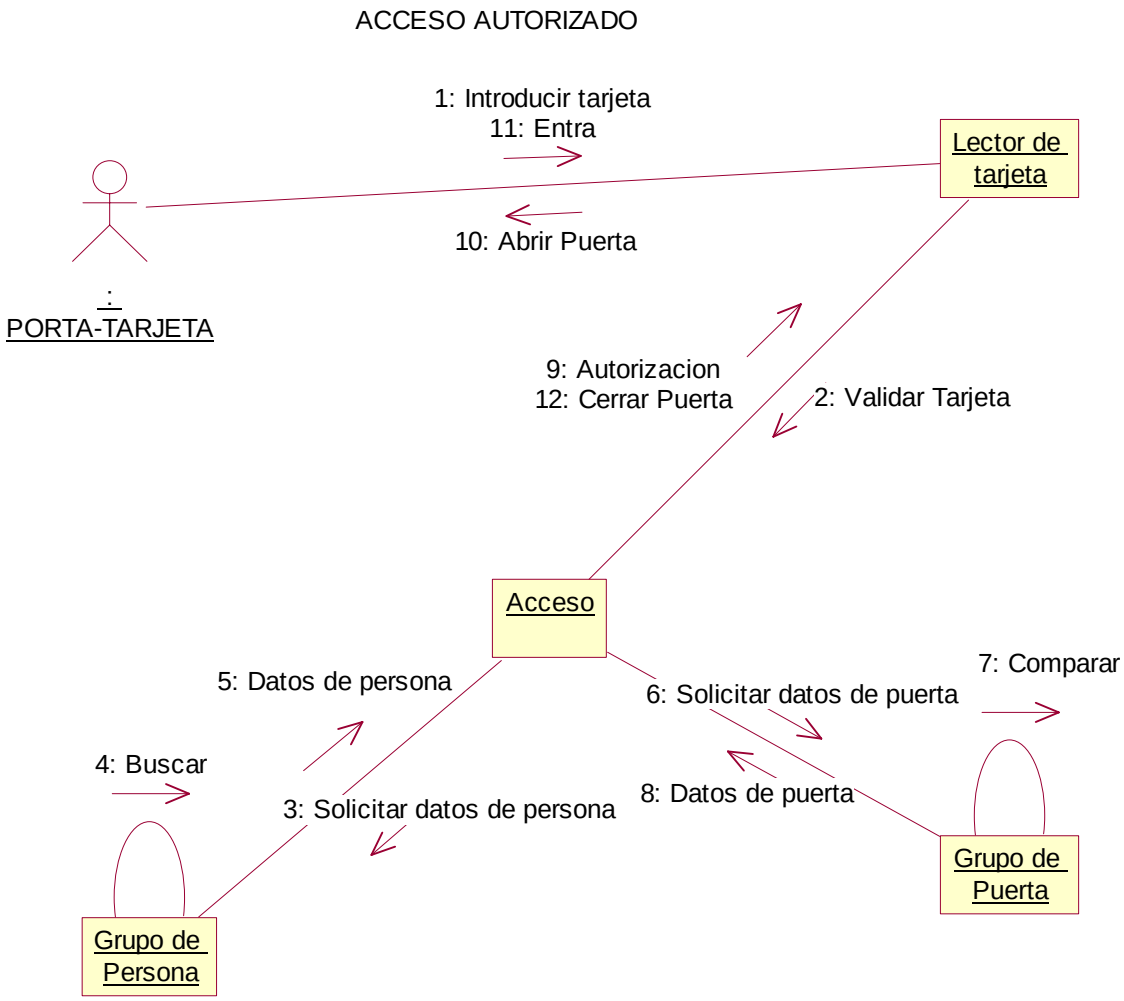
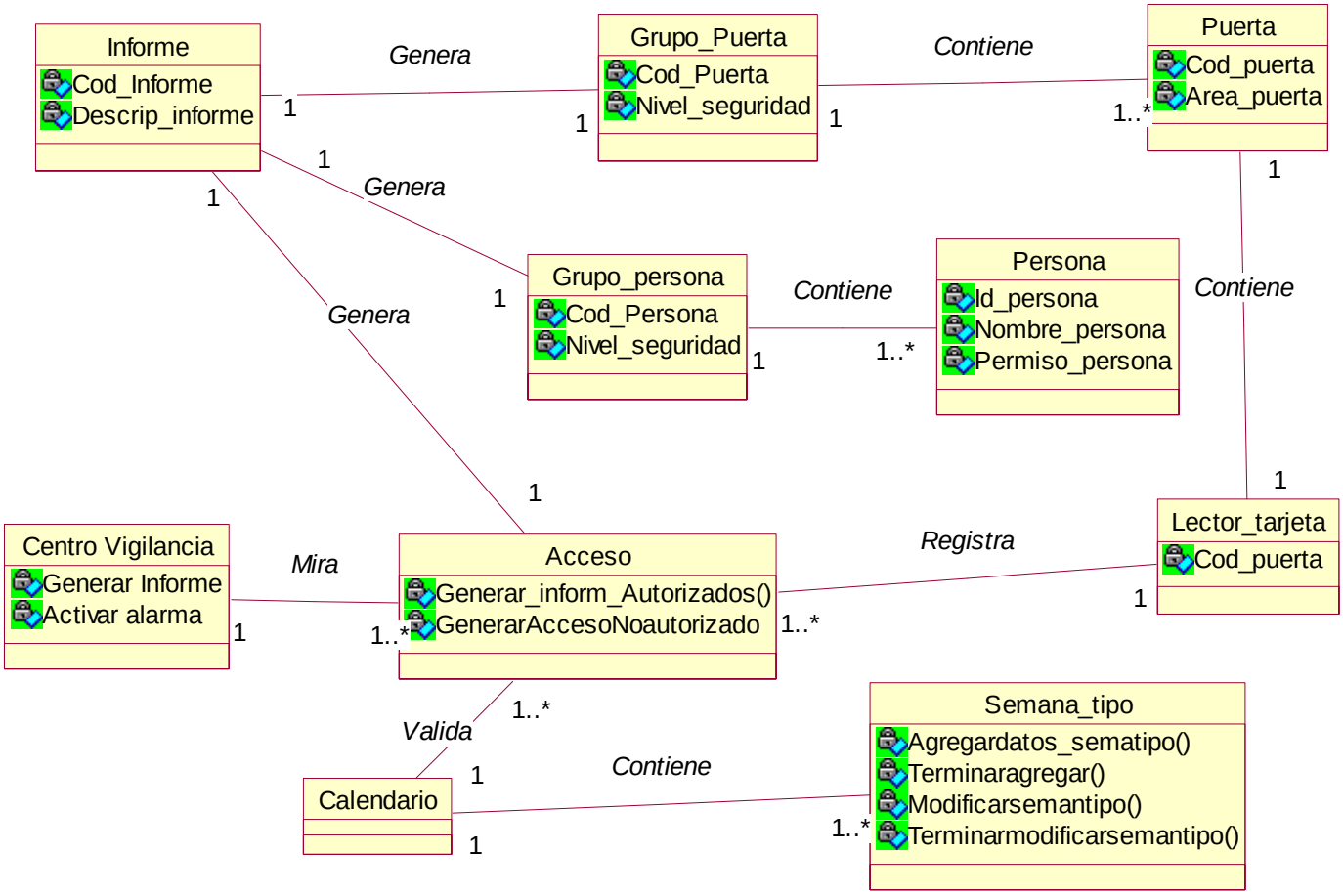
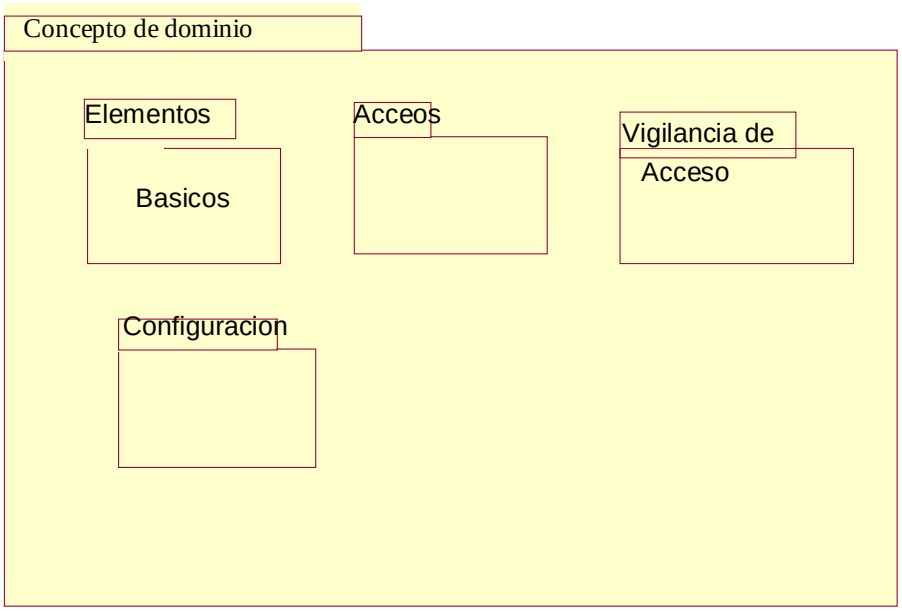
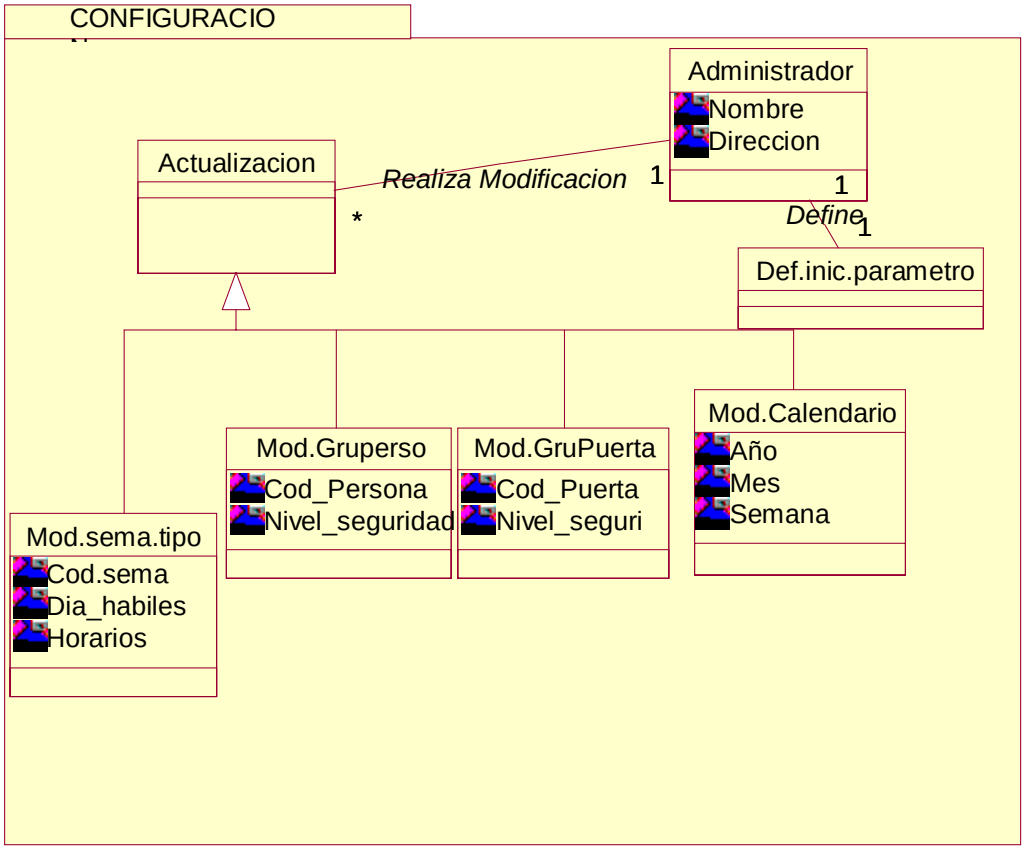
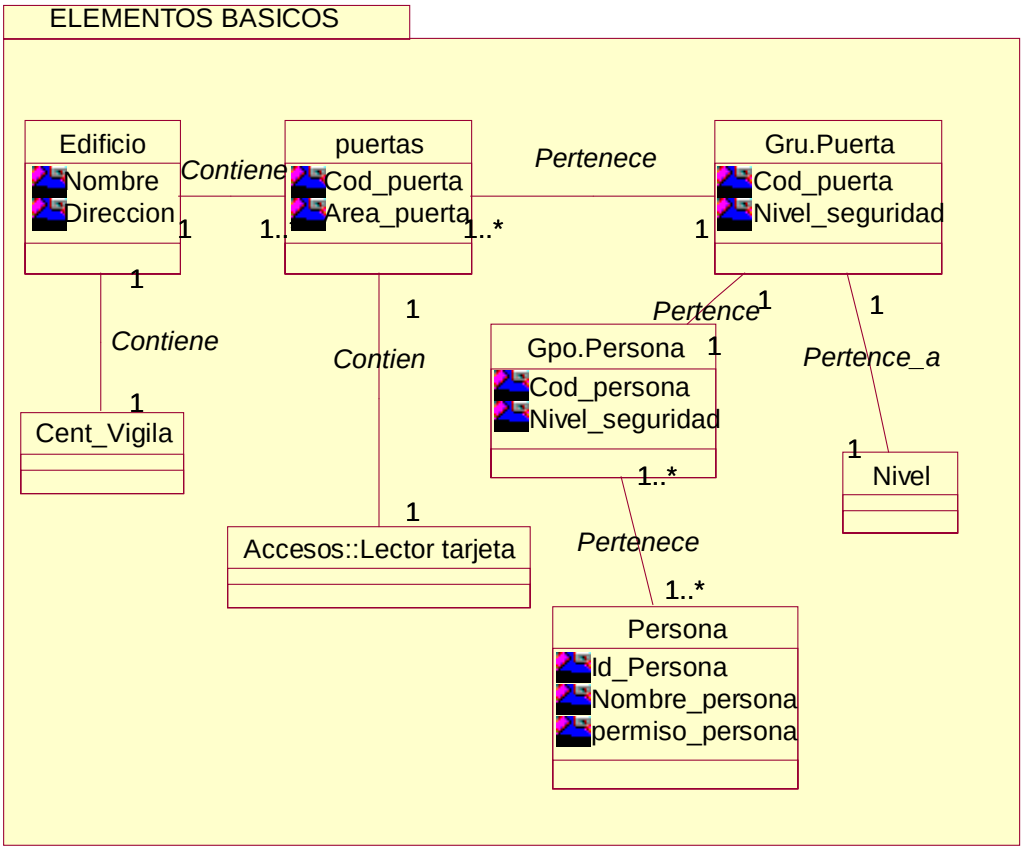

ACCESO AUTORIZADO
1: Introducir tarjeta
11: Entra
Lector de tarjeta
10: Abrir Puerta
:
PORTA-TARJETA
9: Autorizacion
12: Cerrar Puerta
2: Validar Tarjeta
Acceso
5: Datos de persona
6: Solicitar datos de puerta
7: Comparar
4: Buscar
8: Datos de puerta
3: Solicitar datos de persona
Grupo de Puerta
Grupo de Persona

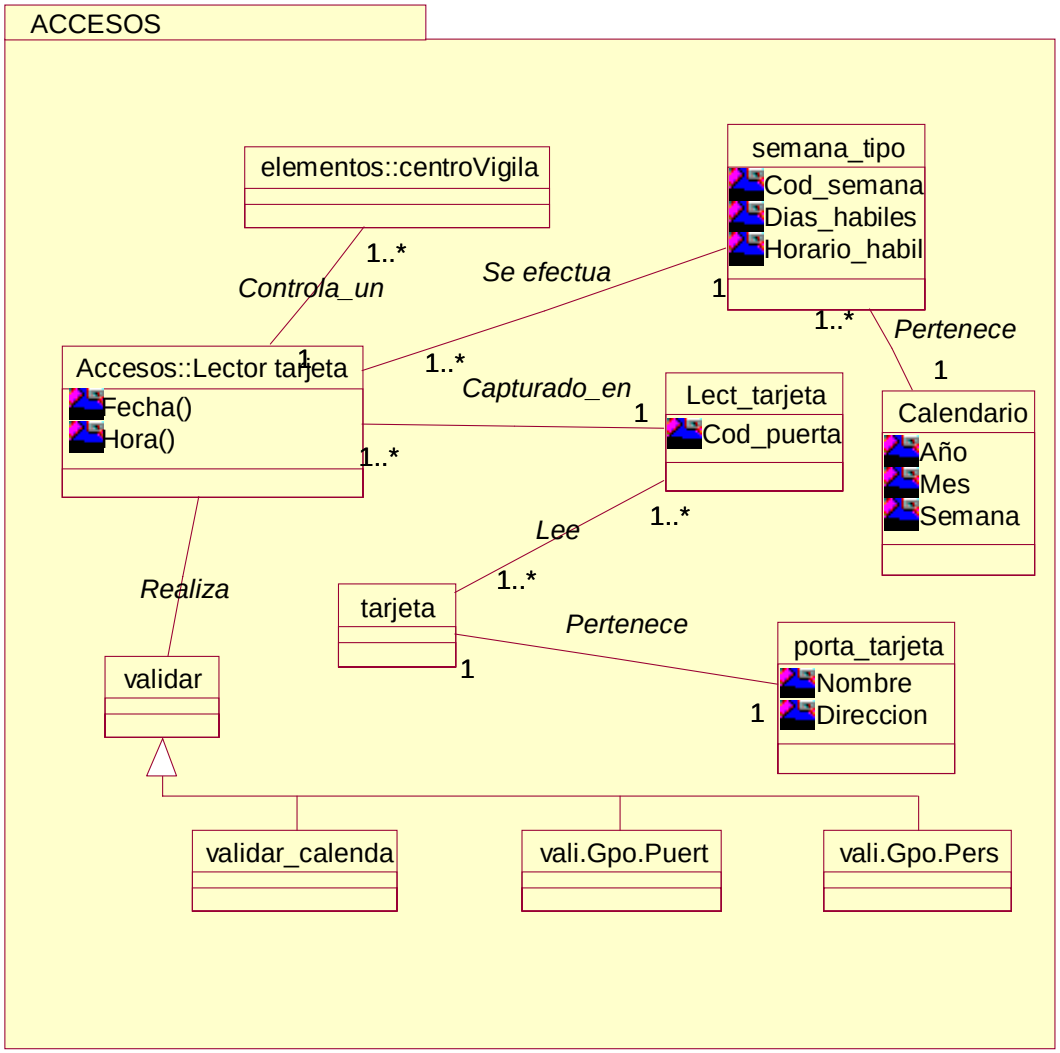
DIAGRAMA DE CLASE CORREGIDO

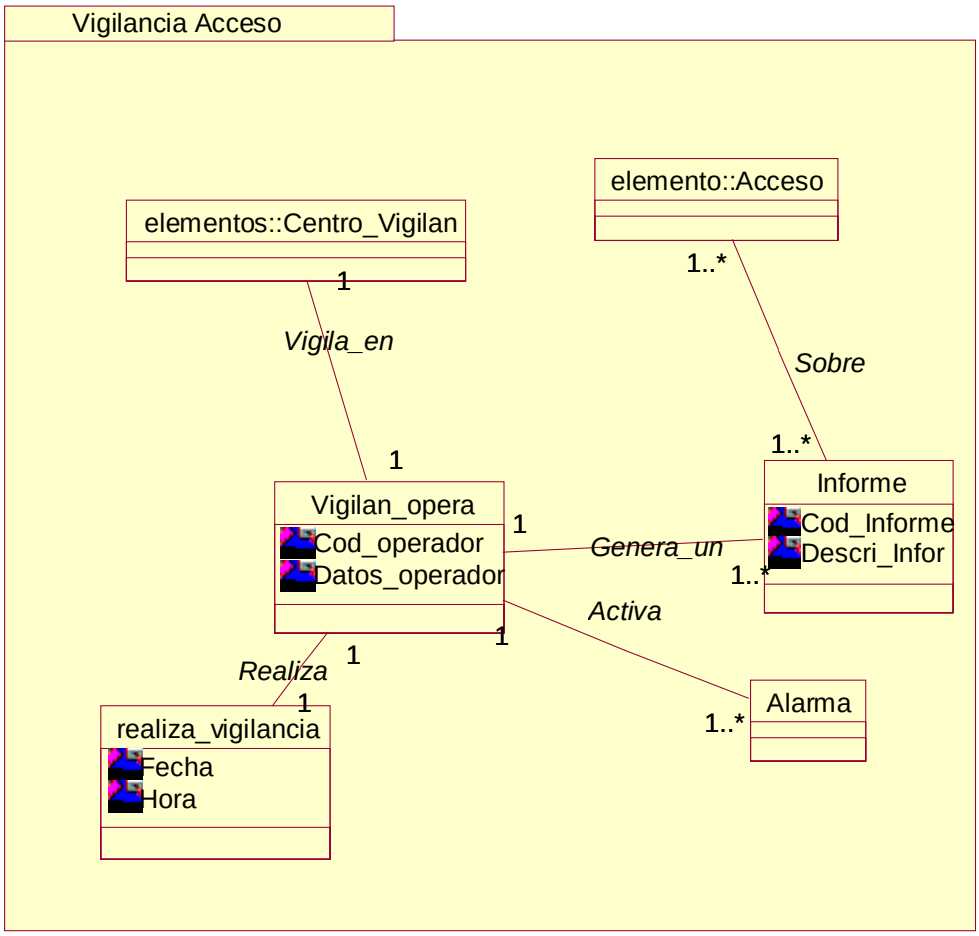



Concepto de dominio
Elementos
Basicos
Acceos
Vigilancia de
Acceso
Configuracion


CONFIGURACIO
Administrador
Nombre
Direccion
Actualizacion
Realiza Modificacion
1
*
1
Define
1
Def.inic.parametro
Mod.Calendario
Año
Mes
Semana
Mod.Gruperso
Cod_Persona
Nivel_seguridad
Mod.GruPuerta
Cod_Puerta
Nivel_seguri
Mod.sema.tipo
Cod.sema
Dia_habiles
Horarios


ELEMENTOS BASICOS
Edificio
Nombre
Direccion
Contiene
puertas
Cod_puerta
Area_puerta
Pertenece
Gru.Puerta
Cod_puerta
Nivel_seguridad
1
1..
1..*
1
1
Contiene
1
Cent_Vigila
1
Contien
1
Accesos::Lector tarjeta
Pertence
1
1
Gpo.Persona
Cod_persona
Nivel_seguridad
1
1..*
Pertence_a
1
Nivel
Pertenece
1..*
Persona
Id_Persona
Nombre_persona
permiso_persona


ACCESOS
elementos::centroVigila
semana_tipo
Cod_semana
Dias_habiles
Horario_habil
1..*
Controla_un
Se efectua
1
1..*
Pertenece
1
Accesos::Lector tarjeta
Fecha()
Hora()
1
1..*
Capturado_en
1
Lect_tarjeta
Cod_puerta
Calendario
Año
Mes
Semana
1..*
Lee
1..*
Realiza
tarjeta
Pertenece
1
porta_tarjeta
Nombre
Direccion
1
validar
validar_calenda
vali.Gpo.Puert
vali.Gpo.Pers


Vigilancia Acceso
elementos::Centro_Vigilan
1
Vigila_en
1
elemento::Acceso
1..*
Sobre
1..*
Informe
Cod_Informe
Descri_Infor
Vigilan_opera
Cod_operador
Datos_operador
1
Genera_un
1..*
Activa
1
1
Realiza
1
realiza_vigilancia
Fecha
Hora
Alarma
1..*