Índice del Documento

Tabla de Figuras

1 INTRODUCCIÓN

A raíz de la utilización de computadores personales a mediados de los 80, en la UNAN – LEON y específicamente en el Departamento de Matemáticas, éste coamienza a ofrecer los primeros laboratorios estando estos a cargo de profesores del departamento que de forma autodidacta se dedican al estudio de algunas aplicaciones soportadas sobre MS-Dos. Es así como, en función de este grupo de profesores, se crea el Área de Computación en 1986 adscrita a este departamento. Eventualmente se ofrecen carreras técnicas y una licenciatura en Estadística y Computación, lo cual genera un mayor interés por la informática en esta nueva área. Este interés se limita fundamentalmente a aspectos básicos de programación y bases de datos.

En 1994, se propone y es aprobado el inicio de la carrera "Licenciatura en Computación", debido a lo cual el área de Computación se convierte en Departamento de Computación, contando desde ese año con el apoyo del departamento de Automática de la Universidad de Alcalá, mediante la suscripción de un programa de colaboración el cual hoy en día se mantiene activo.

Desde el inicio de la Licenciatura en Computación hasta su actual evolución en Ingeniería en Sistemas de Información el componente curricular **Bases de Datos I**, ha jugado un papel fundamental dentro de los contenidos programáticos de las carreras antes mencionadas, esto es debido a que este área de la Informática ha representado uno de los campos informáticos más utilizados por las empresas para resolver sus diferentes problemáticas administrativas.

Dando un paso más hacia la mejora de la calidad docente con que se imparte esta asignatura en la carrera de Ingeniería en Sistemas de Información se abordarán en este documento los siguientes aspectos:

- Los temas teóricos del plan docente del componente **Base de Datos I**, su relación de dependencia con otros componentes curriculares de la carrera, el material didáctico de apoyo, la metodología a utilizar, el desarrollo teórico práctico de los temas y la Bibliografía utilizada.
- Las prácticas de laboratorio como parte esencial del plan docente procurando relacionarlas lo más posible con los aspectos teóricos, las prácticas culminarán en un diseño completo de un sistema.

De forma específica y precisa se desarrollarán los temas de cada uno de los capítulos de este componente, de forma tal que el departamento cuente con un documento oficial que además de servir de apoyo a la preparación de la asignatura, sirva como un ágil documento informativo al resto de profesores del departamento y a los estudiantes.

El desarrollo del presente plan docente prevé tres fases:

1) La definición del temario del componente y su distribución temporal a lo largo del Semestre

2) Recopilar la adecuada información en función de la cual desarrollar el contenido de la parte teórica ya previamente definido

3) Elaboración de las propuestas de las prácticas de Laboratorio especificando su distribución temporal a lo largo del semestre y la metodología de evaluación.

2 OBJETIVOS DEL PLAN DOCENTE

A continuación se describirán los objetivos que se plantean a la hora de desarrollar este plan docente.

Estos objetivos se desglosan en dos grupos, por un lado los objetivos generales que se buscan y, por otro, los objetivos específicos que están contenidos en los generales y son más detallados.

2.1 Objetivos Generales

Este plan docente tiene como objetivos generales los siguientes:

- Dotar al Departamento de Computación de un documento oficial que contenga el contenido programático (prácticas y laboratorios) del componente Curricular: **Bases de Datos I**

- Diseñar el plan docente del componente curricular Base de Datos I de tal forma que sirva de base para la preparación de cada una de las conferencias Teóricas y Laboratorios especificando la ubicación temporal y los prerrequisitos de dicho componente.

2.2 Objetivos Específicos

A continuación tenemos los objetivos específicos que están contenidos en los generales pero son más detallados y nos permiten establecer claramente lo que queremos alcanzar.

- Proporcionar al estudiante un documento de apoyo, el cual será el soporte fundamental de cada una de las conferencias teóricas y prácticas de laboratorio

- Priorizar en el desarrollo del plan docente los aspectos prácticos del análisis y el diseño de las bases de Datos

- Mostrar la importancia a lo largo de todo el documento, el papel que juegan en el análisis y diseño, el modelo conceptual y el modelo lógico

- Enseñar la importancia que tiene las herramientas en el diseño de un sistema como complemento al diseño de las bases de datos

- Proporcionar al estudiante mediante el documento una visión general de los principales lenguajes relacionales

- Enseñar mediante el presente plan docente los fundamentos de diseño de una base de datos partiendo de formas normales

- Enfatizar en las ventajas del usuario al utilizar un sistema gestor de base de datos

- Proporcionar al estudiante los pasos fundamentales para el diseño completo de un sistema.

3 SITUACIÓN DE LA ASIGNATURA EN EL PLAN DE ESTUDIOS DE LA UNAN-LEON

El Componente Curricular **Base de Datos I** se imparte en el quinto semestre (primer semestre de tercer año) de la carrera Ingeniería en Sistemas de Información de la UNAN–León, que ofrece actualmente la Facultad de Ciencias.

La carrera se desarrolla a lo largo de 10 semestres de estudios, es decir, 5 años académicos / lectivos. El período lectivo para esta asignatura consta de 6 horas a la semana, de las cuales 4 horas se dedican a la parte teórica y 2 horas a la parte práctica. El total de semanas de un semestre académico es de 16 por lo cual el número de horas teóricas disponibles es de 64 (32 Conferencias Teóricas) y un total de 32 horas para los laboratorios.

Lo indicado anteriormente se resume a continuación

Relación Horaria

1.	Nombre de la Asignatura:	**Base de Datos I**
2.	Total de Semanas por Semestre:	**16**
3.	Horas Teóricas Semanales:	**4**
4.	Horas Prácticas Semanales:	**2**
5.	Total de Horas Semanales:	**6**
6.	Total de Horas Teóricas por Semestre:	**64**
7.	Total de Horas Prácticas por Semestre:	**32**
8.	Total de Horas Teóricas y Prácticas por Semestre:	**96**

3.1 Relación entre Asignaturas

A continuación se mostrará la relación existente entre el componente **Base de Datos I** y otras asignaturas del plan de ISI (Ingeniería en Sistemas de Información).

En la siguiente tabla se muestran las asignaturas que aportan un conocimiento previo al componente: Base de Datos I.

Asignatura	Impartida en
Programación I y II	III y IV Semestre
Matemáticas Discretas	III Semestre
Álgebra Lineal	IV Semestre

Figura 3.1 Relación de asignaturas

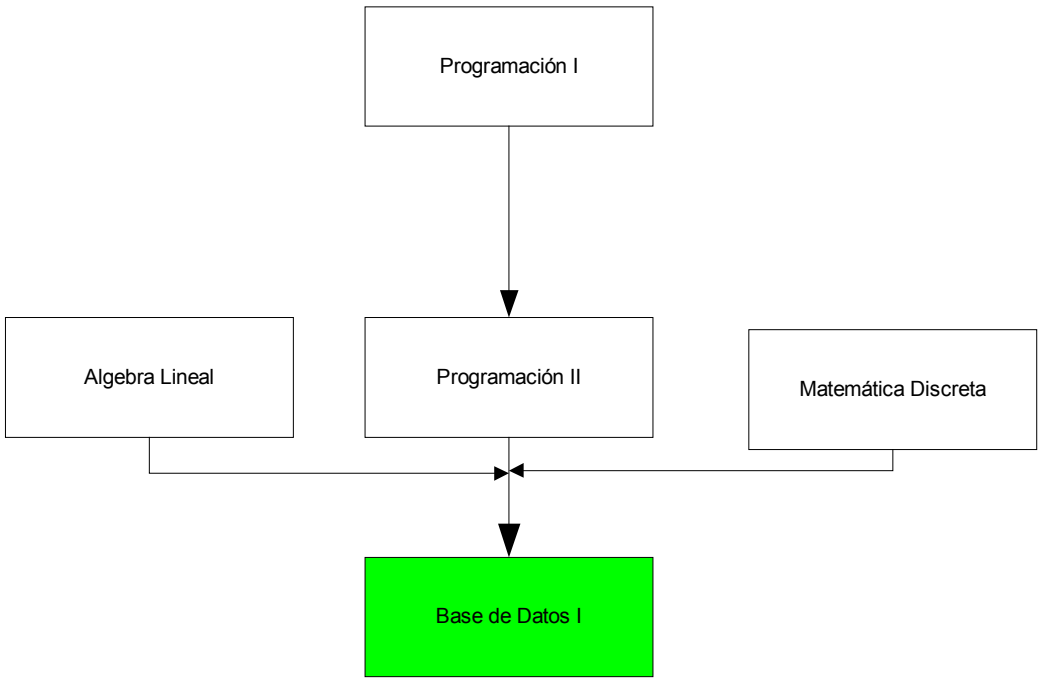


Figura 3.2 Relación Gráfica Entre Asignaturas

A continuación, se describirá de manera general el contenido de cada asignatura y su relación con las demás asignaturas mostradas en la Figura 3.2 (arriba).

3.1.1 Álgebra Lineal

Este componente es prerrequisito para el componente Base de Datos I, dado que en él el estudiante se introduce a estructuras algebraicas que le servirán de fundamento para entender adecuadamente el modelo matemático que sirve de base al diseño relacional de bases de datos.
Por otra parte el estudiante a través de este componente comenzará a relacionarse con la notación matemática requerida para definir el Álgebra relacional así como los otros lenguajes equivalentes al relacional que serán estudiados en el curso de Base de Datos I.

El contenido por temas de la asignatura es el siguiente:

Tema Nº 1:
Matrices: En este tema se estudia el concepto de matriz, las operaciones con matrices; suma, producto por un escalar, producto, inversa, etc.

Tema Nº 2:
Espacios vectoriales: En este tema se estudia el concepto de cuerpo, concepto de espacio vectorial, las propiedades de los espacios vectoriales, los subespacios, dependencia e independencia lineal y el sistema de generadores.

Tema Nº 3:

Transformaciones lineales y matrices: En este tema se estudia el concepto de transformación lineal entre dos espacios vectoriales, núcleo e imagen de una transformación lineal, dimensiones del núcleo y de la imagen, teorema fundamental de las transformaciones lineales, matriz asociada a una transformación lineal, composición de transformaciones lineales, transformación lineal no singular y espacio vectorial de transformaciones lineales.

3.1.2 Programación I

Esta asignatura tiene como objetivos, que el alumno logre manejar y explicar las técnicas de un lenguaje de programación, que logre explicar los conceptos básicos de programación estructurada y que pueda resolver problemas de cálculo haciendo uso de las técnicas para desarrollar algoritmos.
Es un componente primordial para los conocimientos que el estudiante adquirirá en el componente base de Datos I, dado que esta base de conocimientos y habilidades son esenciales para poder diseñar consultas relacionales, así también para poder resolver las diferentes prácticas de laboratorio, la mayoría de las cuales requiere un dominio adecuado de las técnicas la programación estructurada.

El contenido por temas de este componente es el siguiente:

Tema Nº 1:
Fundamentos de ordenadores: En este tema se estudia una Introducción a los ordenadores, el concepto de ordenador, la arquitectura propuesta por Von Neuman, definición de programa y lenguaje, qué son los periféricos y cómo funcionan, el concepto de un sistema operativo y qué tipos de programas existen.

Tema Nº 2:
Fases en el desarrollo de un programa: En este tema se hace una breve reseña histórica del Lenguaje C, se muestra la realización de un programa en C sencillo pero ilustrativo, se explican los caracteres que soporta C, los tipos fundamentales de datos con ejemplo de valores que puedan tomar, qué son los tipos derivados y cuándo usarlos, los nombres de tipos, constantes, identificadores, palabras claves, comentarios, variables, declaración de constantes, expresiones numéricas, los diferentes tipos de operadores que soporta C, qué son las expresiones y cómo se construyen, expresiones para expresar condición, y la prioridad y el orden de evaluación de una expresión.

Tema Nº 3:
Estructura de un programa: En este tema se explica cómo es la forma que debe tener un programa en C, qué son los ficheros de cabecera, expresiones, sentencias, declaraciones de funciones, definiciones de funciones, llamada a una función, los distintos tipos de argumentos que puede tener una función, ámbito y accesibilidad de una variable, las clases de almacenamiento, sentencias de asignación, y la entrada y salida estándar.

Tema Nº 4:
Sentencias de control: En este tema se estudian las sentencias condicionales if y else, las sentencias switch y las sentencias break, las sentencias while, do while, for, continue, y go to.

Tema Nº 5:
Tipos estructurados de datos: En este tema se estudian los arreglos, sus características, sus formas posibles de declaración, cómo se pasan los arreglos a una función, las cadenas de caracteres, las funciones para manipular cadenas de caracteres, funciones para conversión de datos, funciones para clasificar y convertir caracteres, qué son las estructuras y las uniones, qué son los campos de bits y cuando son útiles.

Tema Nº 6:
Punteros: En este tema se estudia qué son los punteros, para qué sirven y cómo se crean, las operaciones con punteros, se hace la relación entre punteros y arrays, los punteros a cadenas de caracteres, cómo se asigna memoria haciendo uso de los punteros y de las funciones de bibliotecas, y por último los punteros a estructuras.

3.1.3 Programación II

Esta asignatura supone un previo conocimiento de Programación I, y tiene como objetivos desarrollar en el estudiante la capacidad para utilizar las funciones de C, así como ser capaz de desarrollar sus propias funciones para la implementación de programas de forma estructurada; proporcionar el estudiante el conocimiento necesario para que sea capaz de utilizar y crear sus propios archivos en Lenguaje C, y que pueda distinguir entre los diferentes tipos de archivos que se pueden implementar en este lenguaje; que el estudiante desarrolle habilidades para utilizar las macros predefinidas y directrices en C en el proceso de pre-compilación; y que el alumno pueda diseñar y utilizar estructuras dinámicas de datos y algoritmos básicos de ordenación.
Así como en el caso de programación I, es fundamental para el desarrollo del componente Bases de Datos I debido a que refuerza los conocimientos adquiridos en Programación I sobre programación Estructurada, específicamente el estudiante con este curso sienta las bases para el uso de módulos de programación por medio del conocimiento adquirido al estudiar funciones. Por otra parte se adquieren habilidades para el manejo y tipo de archivos de archivos, fundamentales en el estudio de las bases de Datos.

El contenido por temas de la asignatura es el siguiente:

Tema Nº 1:
Funciones: En este tema se estudia el paso de diferentes tipos de parámetros a una función: tipo array, tipo estructura y tipo puntero; las funciones que retornan punteros; los argumentos en la línea de órdenes; la redirección de la entrada y la salida; las funciones recursivas y las funciones predefinidas de C.

Tema Nº 2:
Funciones estándar de E/S: En este tema se estudia la manipulación de ficheros en el disco, la apertura y cierre de ficheros, la detección de errores, los diferentes tipos de E / S, los comandos que controlan el buffer, los ficheros temporales y los comandos para el acceso aleatorio.

Tema Nº 3:
El preprocesador de C: En este tema se estudia las macros y directrices, la compilación condicional y la utilización de ficheros de cabecera.

Tema Nº 4:
Estructuras dinámicas de datos y ordenación: En este tema se estudian las listas lineales y sus operaciones básicas; las pilas, colas, listas circulares y listas doblemente enlazadas; los árboles binarios de búsqueda, los árboles binarios perfectamente equilibrados, la recursividad y la clasificación de datos; los algoritmos de búsqueda de datos; la ordenación de ficheros en disco; y los algoritmos hash.

3.1.4 MATEMÁTICAS DISCRETAS

Esta asignatura tiene como objetivos: Introducir al estudiante en los principios, leyes y cuantificadores fundamentales de la lógica formal, saber las propiedades principales del álgebra de boole, saber implementar los principales algoritmos de búsqueda, utilizando árboles para tal fin, aprender a reducir mapas binarios mediante Karnaugh.
Como podemos observar una parte esencial de los objetivos contenidos en este componente tiene relación directa con los conocimientos básicos requeridos en el Componente bases de Datos I, como son: El Dominio de los principios básicos de la lógica formal de primer orden así como sus cuantificadores relacionados. Por otra parte la inducción matemática como conocimiento adquirido ayudará a los estudiantes del componente base de Datos I a generalizar conceptos. y la revisión de la teoría de conjuntos y las relaciones tienen ya una relación directa con el álgebra relacional que es uno de los temas fundamentales de este componente.

El Contenido por temas del componente es el siguiente :

Tema Nº 1:
Lógica e Inducción: En este tema se estudian los diferentes cuantificadores utilizados en la lógica formal y se introduce al estudiante al cálculo de predicados finalizando el tema con el principio de la inducción matemática.

Tema Nº 2:
Relaciones: En este tema se estudian las relaciones entre conjuntos, conjuntos parcialmente ordenados, las relaciones de equivalencia, Composición y clausura de Relaciones.

Tema Nº 3:
Graficas: En este tema se estudiarán: Gráfica dirigidas, Isomorfismo e Invariantes, Digráficas con peso, Gráficas no dirigidas, problemas de recorridos de aristas, problemas de recorridos de vértice, Matrices, relaciones y gráficas, Algoritmos para gráficas, Modificaciones y aplicaciones de algoritmos.

Tema Nº 4:
Árboles: En este tema se estudian: Las propiedades de los árboles, Árboles enraizados, Algoritmos de búsqueda de primera profundidad, Notación polaca y árboles con peso.

Tema Nº 5:
Álgebra Booleana: En este tema se estudian : Las Latises, Latises distributivas booleanas, Álgebras de Bool, Expresiones Booleanas, redes lógicas e introducción a los mapas de Karnaugh.

4 METODOLOGÍA DIDÁCTICA Y MATERIAL DIDÁCTICO UTILIZADO

En cuanto a la metodología didáctica tenemos:

- Para impartir los temas de esta asignatura el método principal consiste en impartir lecciones magistrales, en las cuales se planificarán los apartados a desarrollar por medio de clases teóricas de 100 minutos de duración.
- En estas clases teóricas también se realizarán clases prácticas relacionadas con los temas teóricos previamente ofrecidos con el fin de mejorar la comprensión de la asignatura.
- En cada uno de los cursos se hará el máximo esfuerzo por cumplir con la programación temporal contemplada en el presente documento, aunque la experiencia pasada indica que siempre existen pérdidas debido a situaciones inesperadas como las huelgas.
- Si se da esta situación se procurará resumir los aspectos fundamentales pendientes en las últimas conferencias del respectivo semestre.

El material didáctico a utilizar es el siguiente:

- Retro proyector y transparencias siempre que sea posible.
- Se debe contemplar la posibilidad de utilizar solamente los medios tradicionales, pizarra y crayones debido a que, con relativa frecuencia, se dañan las lámparas de los proyectores.
- De ocurrir esto, se debe cubrir en la conferencia respectiva el material contemplado en las transparencias.

La metodología utilizada con los medios antes indicados será desarrollada así:

- En función del contenido del presente proyecto referido al desarrollo teórico del componente y tomando en cuenta la bibliografía básica y complementaria sugerida, se procederá a diseñar cada una de las transparencias, detallando en la pizarra cualquier explicación adicional requerida.

En cuanto al material de estudio:

- Se utilizará como material de estudio las respectivas transparencias de cada conferencia más la adecuada documentación adicional la cual podrá ser adquirida por el estudiante accediendo a la página web de la asignatura o solicitando dicho material para su reproducción a la secretaria del departamento.
- Además, el estudiante podrá complementar la documentación requerida haciendo uso de la biblioteca DEL DEPARTAMENTO o visitando la biblioteca CENTRAL si así se requiere

5 PROCESO DE EVALUACIÓN

El componente base de datos I está compuesto por clases teóricas y prácticas de laboratorios.

Evaluación de la Teoría
Durante el semestre se realizan 2 evaluaciones parciales y posteriormente un examen final o semestral. Las calificaciones obtenidas en las 2 evaluaciones parciales equivalen al 60% de la nota final teórica, el restante 40% de la nota final teórica lo representa el exámen final

	Primera Evaluación Parcial	**Segunda Evaluación Parcial**	**Exámen Final**	**Nota_Final**
Teoría	60	60	*100*	
Prácticas	40	40		
Total	100	100	*100*	
%	*30%*	*30%*	*40%*	**100%**

Evaluación de las Prácticas de Laboratorio

Figura 5.1 Proceso de Evaluación

Durante el Semestre se realizarán 2 evaluaciones parciales coincidiendo con los períodos de evaluación de la teoría, estas evaluaciones podrán ser pruebas, defensas de proyectos, o una combinación de estas. El valor relativo de cada una de estas evaluaciones es del 40.

Un resúmen de lo antes indicado puede verse en la figura 5.1

Cálculo de la Nota Final

1Ev_Par **=** 0.6 ***** 1Ex_Par_Teor **+** 0.4 * 1Ev_Par_Lab
2Ev_Par **=** 0.6 ***** 2Ex_Par_Teor **+** 0.4 * 2Ev_Par_Lab

Prom_Entrada **=** (1Ev_Par **+** 2Ev_Par) **/** 2

Nota Final = Prom_Entrada * 0.6 **+** 0.4 ***** Ex_Final_Teor

Donde:
1Ev_Par es la Primera Evaluación Parcial
1Ex_Par_Teor es el primer exámen parcial de la teoría
1Ev_Par_Lab es la primera evaluación parcial de las prácticas de laboratorios

2Ev_Par es la segunda evaluación parcial
2Ex_Par_Teor es el segundo exámen parcial de la teoría
2Ev_Par_Lab es la segunda evaluación parcial de las prácticas de laboratorios

Prom_Entrada es el promedio de las dos evaluaciones parciales (incluye teoría y laboratorios)

Ex_Final_Teor es el Exámen Final de la Teoría

Nota Final es la nota definitiva del estudiante

Aspectos especiales de la Evaluación :

- El estudiante podrá realizar el examen final, siempre y cuando el promedio de sus dos calificaciones parciales (Prom_Entrada) sea igual o superior a 50.

- Si la calificación final del estudiante (Nota Final) es menor de 60 (Aplazado), el estudiante puede presentarse a una segunda convocatoria, la que se conoce como examen especial. En ésta, su nota definitiva es la calificación que obtenga en dicha convocatoria, es decir, que para efectos de ésta, las calificaciones anteriores no se toman en cuenta.

6 TEMPORIZACIÓN

El Componente curricular **Base de Datos I** se imparte en el primer semestre de tercer año de la titulación Ingeniería en Sistemas de Información (ISI), ofrecida por el Departamento de Computación de la UNAN León. Cada semestre consta de 16 semanas lectivas. La asignatura tiene una frecuencia de 6 horas semanales (4 de teoría y 2 de prácticas de laboratorio), resultando en un total 64 horas teóricas y 32 horas prácticas para un total de 96 horas.
Si tomamos en cuenta los días festivos y las semanas de realización de exámenes, la cantidad de semanas por semestre se reducen a 14, obteniendo ahora **56 horas de teoría** y **28 horas de laboratorio** respectivamente. Hay que recordar que cada sesión de clase teórica y de laboratorio dura 100 minutos.

La temporización de la parte teórica se muestra en la siguiente tabla, figura 6.1

Sem.	Tema Nº	Clase Nº	Nº de Horas	Tema	Contenido del Tema
1	0	1	2	Presentación del Componente	Introducción al Curso.
1	1	2	4	Introducción	Aplicaciones de los Sistemas de BD Sistemas de Gestión de BD frente a Sistemas de Archivos Problemas Generados en la Gestión de los datos por el uso de un Sistemas de Archivos Niveles de Abstracción de los Datos Ejemplares y Esquemas. Modelo de Datos Lenguajes de BD
2	1	3	6	Introducción	Acceso a BD desde Lenguajes de Programación Usuarios de BD e Interfaces de Usuarios Administración de la BD y funciones del Administrador Gestión de Transacciones Estructura de un SGBD Módulo Gestor de Almacenamiento Módulo Gestor del Procesador de Consultas
2	2	4	8	Diseño de Base de Datos Relaciónales (1)	Diseño Lógico de las Bases de Datos en el modelo relacional Modelo de los Datos Transformación del Esquema Conceptual al relacional Otros Modelos de Datos
3	3	5	10	Modelo Entidad Relación (1)	Conceptos Básicos: Conjunto de entidades Concepto de atributo Conjunto relación Diagrama entidad relación Componentes básicos del diagrama Cardinalidad de un conjunto entidad Conjunto de entidades débiles Concepto de discriminante

					Clave primaria de una entidad débil
3	3	6	12	Modelo Entidad Relación(2)	Representación en el diagrama E-R Restricciones de las Relaciones Claves(Super Clave), Clave candidata, Clave Primaria Nomenclatura para las Cardinalidades de las Relaciones Atributos compuestos Atributos calculados Relaciones Recursivas
4	3	7	14	Modelo Entidad Relación(3)	Relaciones Ternarias Conjuntos de Entidades Débiles Representación de las Entidades Débiles en el E/R
4	3	8	16	Modelo Entidad Relación(4)	Características del Modelo E-R extendido Diseño de Esquemas E/R Reducción de un Esquema E/R a Tablas
5	3	9	18	Modelo Entidad Relación(5)	Clase Práctica del Tema 3
5	3	10	20	Modelo Entidad Relación(6)	Clase Práctica del Tema 3
6	4	11	22	El Modelo Relacional(1)	La Estructura de las Bases de Datos Relaciónales El Álgebra Relacional La Operación Selección La Operación Proyección La Operación Unión La Operación Menos
6	4	12	24	El Modelo Relacional(2)	La Operación Producto Cartesiano Definición Formal del Álgebra Relacional La Operación Producto Natural
7	4	13	26	El Modelo Relacional(3)	Operaciones del Álgebra Relacional Extendida Reunión Externa por la Izquierda y por la Derecha Reunión Externa Completa
7	4	14	28	El Modelo Relacional(4)	Modificación de la Base de Datos Inserción, Actualización, vistas El Cálculo Relacional de Tuplas El Cálculo Relacional de Dominios
8	4	15	30	El Modelo Relacional(5)	Tercera Forma Normal Forma Normal de Boyce-Codd
8	4	16	32	El Modelo Relacional(6)	Cuarta Forma Normal Quinta Forma Normal
9	4	17	34	El Modelo Relacional(7)	Clase Práctica del tema 4
9	4	18	36	El Modelo Relacional(8)	Clase Práctica del tema 4
10	5	19	38	Structured Query Language (SQL) (1)	La Cláusula Select, Where, From Variables Tupla Operaciones sobre cadenas La Cláusula Order By
10	5	20	40	Structured Query Language	La Operación Unión La Operación Intersección,

				Language (SQL)(2)	La Operación Exept Cláusula Group By y Funciones
11	5	21	42	Structured Query Language (SQL)(3)	Sub Consultas Anidadas La Clausula IN Comparación de Conjuntos Comprobación de relaciones vacías (La cláusula exists, La Cláusula Exept) Vistas Consultas Complejas
11	5	22	44	Structured Query Language (SQL)(4)	Modificación de la Base de Datos Borrado, Inserción, Actualizaciones Actualización de Vistas Transacciones Reunión de Relaciones
12	5	23	46	Structured Query Language (SQL)(5)	Lenguaje de Definición de Datos Tipos de Datos en SQL Definición de Esquemas en SQL Funciones de SQL/ORACLE)
12	5	24	48	Structured Query Language (SQL)(6)	Clase Práctica Tema 4
13	6	25	50	Otros Lenguajes Relaciónales (1)	Introducción Query By Example Consultas Sobre una Relación Consultas Sobre Varias Relaciones Caja de Condición La relación resultado Presentación Ordenada de las Tuplas Operaciones de Agregación modificaciones de la Base de Datos Borrado, Inserción, Actualización
13	6	26	52	Otros Lenguajes Relaciónales (2)	Datalog Estructura Básica Sintaxis de las Reglas de Datalog Semántica de Datalog no Recursivo Semántica de una Regla Semántica de un Programa Seguridad Operaciones Relaciónales en Datalog
14	6	27	54	Otros Lenguajes Relaciónales (3)	Datalog(Continuación) la Recursividad en Datalog La Potencia de la recursividad Recursividad en Otros lenguajes Interfaces de Usuarios y Herramientas Los Formularios e Interfaces Graficas de Usuarios Los Generadores de Informes Las Herramientas de Análisis de Datos
14	6	28	56	Otros Lenguajes Relaciónales (4)	Clase Práctica Tema 5

Figura 6.1 Temporización

7 DESARROLLO DEL TEMARIO

En esta sección se desarrollan los temas a impartir en Base de Datos I. Se comienza a partir del tema cero, que es una Introducción general a las bases de datos, y luego se prosigue con el desarrollo de los demás temas.

Para cada uno de los temas se incluirá un pequeño resumen en el que se incluyen los objetivos del tema, el contenido, el tiempo y la bibliografía básica recomendada.

7.1 Tema 0: "Presentación del Componente"

TEMA N° 0

OBJETIVOS:

- Dar a conocer al estudiante los aspectos relacionados a la docencia en general y horario de tutorías.
- Explicar los objetivos generales de la asignatura, el temario y su distribución en el tiempo, así como las relaciones de la misma con la carrera :Ingeniería en Sistemas de Información.
- Dar a conocer al estudiante la metodología de enseñanza que se va a seguir durante el curso, los criterios de evaluación y la bibliografía básica y complementaria.
- Brindar detalles del acceso a la documentación preparada por el profesor para impartir el componente.

CONTENIDO

- Documento de presentación del componente curricular al alumno.
- Información sobre la organización de los grupos del laboratorio.
  Acerca de las prácticas de Laboratorios: Todos los alumnos deben formar parte de un grupo para trabajar en el laboratorio; en total se conformarán grupos con un máximo de 25 alumnos, el horario de cada grupo será establecido por la dirección del Departamento de Computación, y en ese momento se usará el documento de presentación del laboratorio en donde se le brindará todos los datos necesarios para la realización de éste; dicho documento estará disponible en la página web del profesor y en la secretaría del Departamento de Computación. El documento antes mencionado se presentará en el proyecto en la sección de "Desarrollo de las Prácticas".

DURACIÓN: 2 HORAS.

COMPONENTE: BASE DE DATOS I CURSO 2005 / 2006
DEPARTAMENTO: COMPUTACIÓN – FACULTAD DE CIENCIAS
TITULACIÓN: INGENIERIA EN SISTEMAS DE INFORMACIÓN
AÑO: III
SEMESTRE: SEXTO
PROFESOR : RICARDO ESPINOZA

TEMARIO DE TEORÍA:

Tema 1: Introducción

- Introducción
  - Concepto de SGBD
  - Concepto de Base de Datos
  - Algunas Aplicaciones de los SGBD
- Sistemas de Bases de Datos Frente a Sistemas de Archivos
  - Redundancia e Inconsistencia de los Datos

- Dificultad en el Acceso a los Datos
- Aislamiento de los datos
- Problemas de Integridad
- Problemas de Atomicidad
- Anomalías en el Acceso Concurrente
- Problemas de Seguridad

- Visión de los Datos
  - Niveles de abstracción
    - Nivel Físico
    - Nivel Lógico
    - Nivel de Vistas
- Lenguajes de Bases de Datos
  - Lenguaje de Definición de Datos(LDD)
  - Lenguaje de Manipulación de Datos(LMD)
- Acceso a las Bases de Datos desde Programas de Aplicación
  - Metodología
- Usuarios y Administradores de Base de Datos
  - Los Tipos de usuarios de la base de Datos
  - Los Usuarios Normales
  - Los programadores de Aplicaciones
  - Los Usuarios Sofisticados
  - Los Usuarios especializados
- Administrador de La Base de Datos
- Funciones Principales del administrador
  - Mantenimiento Rutinario
- Gestión de Transacciones y Control de Concurrencia
  - Conceptos Básicos del módulo
- Estructura de un sistema de Bases de Datos
  - El Gestor de Almacenamiento
  - El Gestor de Autorización e Integridad
  - El Gestor de Transacciones
  - El Gestor de Archivos
  - El Gestor de Memoria Intermedia
  - Estructura de Datos
  - El Procesador de Consultas

Tema 2: Diseño de Bases de Datos

- Diseño Lógico de las Bases de Datos en el modelo relacional
  - Introducción
  - Etapas en una metodología de Diseño
    - Diseño Conceptual
    - Diseño Lógico
    - Diseño físico
- Modelo de los Datos
  - El Modelo Relacional
  - El Modelo Entidad Relación
- Transformación del Esquema Conceptual al relacional
  - El grafo relacional
- Otros Modelos de Datos
  - El Modelo de Datos Orientado a Objetos
  - El Modelo de Datos Relacional Orientado a Objetos
  - Los modelos Semiestructurados

Tema 3: El Modelo Entidad-Relación

- Conceptos básicos
  - Conjuntos entidades
  - Atributos y conceptos relacionados
  - Conjunto de relaciones
- Diagrama entidad relación
  - Componentes básicos del diagrama
- Cardinalidades de un conjunto Entidad

- Conjunto de entidades débiles
  - Concepto de discriminante
  - Clave primaria de una entidad débil
  - Representación en el diagrama E-R de una entidad débil
- La Especialización
- La Generalización
- Herencia de atributos y Relaciones
- Agregación

Tema 4: El Modelo Relacional

- El modelo Relacional
  - Repaso de algunas definiciones matemáticas
    - Concepto de producto cartesiano
    - Concepto de relación
    - Representación tabular de las Relaciones
  - Estructura de las bases de datos relacionales
    - Componentes del modelo y notación
    - Esquema o estructura de una relación
  - Lenguajes de consulta
    - El Lenguaje procedimental: Álgebra relacional
      - Operaciones fundamentales
      - Combinación de las operaciones básicas
      - Otras operaciones derivadas de las básicas
  - El álgebra Relacional y el Modelo E-R
- Operaciones del álgebra relacional extendida
  - La proyección Generalizada
  - Funciones de Agregación
- Reunión Externa
- Modificación de la base de Datos
  - La Operación inserción
- El Lenguaje Cálculo Relacional de Tuplas
  - El Equivalente de la proyección y la Selección en el Cálculo relacional de Tuplas
  - El Equivalente del Producto Cartesiano y El Producto Natural en el Cálculo relacional de Tuplas
- El Lenguaje Cálculo Relacional de Dominios
  - Concepto de Variable de Dominio
  - Representación de una relación en este lenguaje
  - Las Operaciones de Selección y Proyección en el Cálculo relacional de Dominios
  - El Producto Cartesiano y El Producto natural en el cálculo relacional de Dominios
- Teoría de la Normalización
  - Introducción
    - Formas Normales
      - Primera Forma Normal
      - Segunda Forma Normal
      - Tercera Forma Normal
  - La Forma Normal de Boyce-Codd
  - El Proceso de Descomposición de las Formas Normales
    - Descomposición por la aplicación de la Primera Forma Normal
    - Descomposición por la aplicación de la Segunda Forma Normal
    - Descomposición por la aplicación de la Tercera Forma Normal
    - Otros tipos de Dependencias
  - La Cuarta forma Normal
  - La Quinta forma Normal

Tema 5: Structured Query Language-SQL

- El Lenguaje SQL
  - Introducción
  - La Operación Select
    - Implementación de la Proyección y la Selección
    - Implementación del Producto cartesiano

    - Implementación del Producto Natural
- La operación Renombramiento
- Las Variables de Tupla
- El operador Like
    - El Carácter reservado %
    - El carácter reservado _
    - Exepciones
- La Cláusula Order By
    - Los parámetros Asc y Desc
- Las Operaciones de Conjunto
    - La Operación Unión
    - La Operación Intersección
    - La Operación Diferencia
- Funciones de Agregación
    - La Cláusula Group By y las funciones de Agregación
    - Uso de las cláusulas Where y having con funciones de Agregación
- SQL y los valores nulos
    - Algunas soluciones a las problemáticas de valores nulos y las funciones de agregación
- Sub Consultas Anidadas
    - Concepto
    - Sub Consulta de Pertenencia a Conjuntos (In,not in)
- Comparación de Conjuntos
    - Las Cláusulas Some y All
    - Uso de las funciones de Agregación con Some y All
    - Comprobación de relaciones vacías (Exists,Not exists)
    - Aspectos particulares de la cláusula not exists
    - Alcance de las variables de tupla en las sub consultas
    - Comprobación de tuplas duplicadas
- Vistas en SQL
    - Definición
    - Consultas Complejas
    - Relaciones Derivadas
    - La Cláusula With
    - Uso de Múltiples With en una Expresión SQL
- Modificación de la base de Datos
    - Borrado
    - Uso de Funciones de Agregación en una Sub Consulta de borrado
    - Inserción de tuplas en una relación
    - Actualización de las tuplas en una relación
- Anomalías al Utilizar Vistas
- Las Transacciones en SQL
- Mas sobre reuniones de relaciones
    - La Cláusula Inner Join
    - La Cláusula Left Outer join
    - La Cláusula Right Outer join
    - La Cláusula Full Outer join
- Definición de Esquemas relacionales en SQL
    - El Comando Create table
- Cláusulas Opcionales utilizadas en Oracle
- Tipos de Datos Pre Definidos por ORACLE/SQL
- Otros Comandos del DDL de SQL
    - Los Comandos:
        - Drop domain
        - Drop constraint
        - Drop Table
- Modificación del Esquema Relacional
    - El comando Alter Table
- Índices
    - El Comando Create index
    - El Comando Drop Index
- Funciones Definidas en Oracle

Tema 6: Otros Lenguajes Relacionales

- Query By example
  - Consultas sobre una relación
  - Consultas sobre varias relaciones
  - La relación Resultado
  - Presentación Ordenada de las tuplas
  - Operaciones de agregación
  - Modificaciones de la base de Datos
    - Inserción
    - Borrado
    - Actualización
- Datalog
  - Estructura Básica
  - Sintaxis de las Reglas de Datalog
  - Semántica de Datalog no Recursivo
  - Semántica de una Regla
  - Semántica de un Programa
  - Operaciones Relacionales en Datalog
- Datalog(Continuación)
  - la Recursividad en Datalog
  - La Potencia de la recursividad
  - Recursividad en Otros.lenguajes
  - Interfaces de Usuarios.y Herramientas
  - Los Formularios e Interfaces Graficas de Usuarios
  - Los Generadores de Informes
  - Las Herramientas de Análisis de Datos

MATERIAL DE ESTUDIO:

Transparencias y material de apoyo:

- http://isi.unanleon.edu.ni/~rem/

Bibliografía (Teoría)

- Silverschatz, Korth,Sudarschan. 2002. Fundamentos de Base de Datos (Cuarta edición). España: McGraw-Hill/Interamericana de España.
- Adoración de Miguel, Mario Piattini. 1999. Fundamentos y Modelos de Bases de Datos(Segunda edición). España: RA-MA
- Irene Luque Ruiz, Miguel Angel Gomez Nieto. 1997. Diseño y Uso de Bases de Datos Relaciónales. España: RA-MA.

TUTORÍAS:

- Oficina: Antiguo (Control Integrado de Plagas (CIP)), De las Payitas ½ abajo
- Teléfono: 311-2614, 311-5013 Extensión: 311-1153
- Horario: Martes 4 PM – 6 PM
  Jueves 4 PM – 6 PM
- Correo: rem@unanleon.edu.ni

EVALUACIÓN DE LA ASIGNATURA:

Requisitos mínimos:

Obtener una calificación superior a 60 en la nota final

Cálculo de la nota final:

60% →Examen
40% →Laboratorio

7.2 Tema 1: "Introducción"

TEMA N° 1

OBJETIVOS:

- Saber las limitaciones de los lenguajes de Bases de Datos y las posibles soluciones
- Conocer los diferentes tipos de usuarios de un SGBD y las diferentes tareas que debe asumir el administrador de la Base de Datos
- Conocer la Estructura de un SGBD y las principales funciones de cada una de sus partes

CONTENIDO

- Introducción
  - Concepto de SGBD
  - Concepto de Base de Datos
  - Algunas Aplicaciones de los SGBD
- Sistemas de Bases de Datos Frente a Sistemas de Archivos
  - Redundancia e Inconsistencia de los Datos
  - Dificultad en el Acceso a los Datos
  - Aislamiento de los datos
  - Problemas de Integridad
  - Problemas de Atomicidad
  - Anomalías en el Acceso Concurrente
  - Problemas de Seguridad
- Visión de los Datos
  - Niveles de abstracción
    - Nivel Físico
    - Nivel Lógico
    - Nivel de Vistas
- Lenguajes de Bases de Datos
  - Lenguaje de Definición de Datos(LDD)
  - Lenguaje de Manipulación de Datos(LMD)
- Acceso a las Bases de Datos desde Programas de Aplicación
  - Metodología
- Usuarios y Administradores de Base de Datos
  - Los Tipos de usuarios de la base de Datos
    - Los Usuarios Normales
    - Los programadores de Aplicaciones
    - Los Usuarios Sofisticados
    - Los Usuarios especializados
- Administrador de La Base de Datos
- Funciones Principales del administrador
  - Mantenimiento Rutinario
- Gestión de Transacciones y Control de Concurrencia
  - Conceptos Básicos del módulo
- Estructura de un sistema de Bases de Datos
  - El Gestor de Almacenamiento
  - El Gestor de autorización e Integridad
  - El Gestor de Transacciones
  - El Gestor de Archivos
  - El Gestor de Memoria Intermedia
  - Estructura de Datos
  - El Procesador de Consultas

DURACIÓN: 4 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill

- Adoración de Miguel, Mario Piattini. Fundamentos y Modelos de Base de Datos. 2ª Edición. RAMA

7.2.1 Introducción a los Sistemas de Gestión de Bases de Datos

Concepto de Base de Datos:

Son numerosas las definiciones que intentan describir el concepto de Base de Datos. Las definiciones presentan algunos aspectos comunes y muchas de ellas también dejan de incluir conceptos fundamentales que caracterízan las Bases de Datos, dando como resultado que no hay una definición comúnmente aceptada. Por este motivo, a continuación incluímos varias de estas definiciones:

> "Colección de datos interrelacionados almacenados en conjunto sin redundancia perjudicialese innecesarias; 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 determinados para incluir nuevos datos y para modificar o extraer los datos almacenados"
> (Martin 1975)

> "Colección o deposito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y una 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 gama de usos y aplicaciones"
> (Conference des Statisticiens Europpéens, 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 registrados sobre soportes accesibles por ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y tiempo oportuno"
> (Delobel, 1982)

> "Colección no redundante de datos que son compartidos por diferentes sistemas de aplicación"
> (Howe, 1983)

> "Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender todas las necesidades de los diferentes usuarios"
> (Deen, 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)

Breve Historia Sobre la denominación Base de Datos
La aparición de la expresión *base de datos* se produce a comienzos 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 *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.

El concepto de base de datos ha ido cambiando y configurándose a lo largo del tiempo; en la actualidad podemos definir una base de datos como:

> "Colección o depósito de datos integrados, almacenados en soporte secundario(no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes 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."

El Sistema de Gestión de Base de Datos
Es el conjunto de programas que permiten la implantación, acceso y mantenimiento de la base de datos. El SGBD junto con la base de datos y con los usuarios, constituyen el **Sistema de bases de datos**

Orígenes de los SGBD

Intentos previos a los SGBD

El origen del primer sistema gestor de base de datos se ubica a mediados de los años sesenta cuando las compañías IBM y Rockewell International desarrollaron las primeras versiones de un sistema conocido como Data Language/I (DL/I) con el propósito de administrar la gran cantidad de datos asociados con el proyecto espacial Apolo. Este sistema permitía el acceso a escrituras de datos jerárquicos desde programas en COBOL los cuales corrían bajo sistemas /360 de IBM. Posteriormente IBM le adiciona un componente conocido como Infrormation Control System/Data Language/I (ICS/DL/I), con el fin de permitir el acceso a los datos de forma concurrente.

Este primer producto se comercializa en 1969 bajo el nombre de Information Management System 360 (IMS/360) siempre bajo los sistemas/360.

En 1971 IMS/360 evoluciona al sistema IMS/360 versión 2 y dos años después aparece el IMS/VS para los equipos 370 de IBM.

Primera Generación

En forma paralela al desarrollo de los sistemas jerárquicos, aparecen los sistemas de base de datos de redes, los cuales representan un pequeño avance con respecto a los sistemas Jerárquicos.
La utilización del modelo de red para el desarrollo de un Sistema de base de Datos se inicia con el sistema IDS – Integrated Data Store, desarrollado por C.W. Bachman de la compañía general electric. Sin embargo, fue propuesto definitivamente por el grupo DBTG - Fdata Base Task Group – del comité CODASYL -Conference on Data Systemas- en el año 1969.

Entre los productos que marcaron esta segunda generación están:

DMS 1100 de UNIVAC, DMS-II de Burrougs, IDMS de Cullinet y TOTAL de la compañía Cincom system Inc.

Segunda Generación

La tercera generación de bases de datos se inicia con los trabajos del matemático Inglés Edward Codd, con la propuesta de un uevo modelo de datos. Este modelo, conocido como *modelo relacional de codd*, en donde los datos se presentan de f horma tabular, permitió acercar considerablemente al usuario final al desarrollador de la base de datos, situación que no se daba en los SGBD jerárquicos y de redes. De esta forma con los trabajos pioneros de Codd se inicia la *tercera generación* de bases de datos, dando origen a los SGBD relaciónales.
En la actualidad, el número de productos de bases de datos relacionales comercializados es superior a los 200. Entre los SGBD relacionales más importantes, tanto por el número de licencias vendidas, como por sus capacidades, se pueden citar entre otros:

- DB2 de IBM
- Oracle de Oracle Corporation
- Ingres de Relational Technology
- SQL-Server de Sybase
- SQL-Server de Micro Soft
- Informix de Informix Software Inc.
- SQL-Base de Gupta Corporation

Tendencias Actuales
Los SGBD relacionales se han establecido como una tecnología consolidada.
Sin embargo, muchas aplicaciones de multimedios que requieren el manejo de objetos complejos, como diseño y manufactura por computadora, necesitan otro tipo de administradores de datos.

Por ello, ya se ha comenzado a investigar nuevos enfoques, que permitan llenar esta laguna e inclusive, se puede afirmar que se está entrando a la llamada *cuarta generación* de SGBD.

Los modelos de datos de estos sistemas se basan en el paradigma de Orientación a Objetos (OO), que es en la actualidad uno de los más populares en el dominio de los lenguajes de programación.

En lo que respecta a los nuevos tipos de aplicaciones, los productos relacionales actuales son inadecuados, no así el modelo relacional, pues muchas características del modelo relacional no han sido explotadas en su totalidad por las diferentes empresas diseñadores de SGBD. Debido a las limitaciones de los SGBD actuales, en estos momentos se pueden distinguir dos grandes tendencias: ampliar los SGBD relacionales o bien desarrollar nuevos SGBD-OO_s. Otro problema importante es la referente al problema de la administración inteligente de la información. Esto es, módulos que junto con el SGBD evolucionado, permitan una recuperación fácil y rápida de la información requerida por el usuario. Esto constituye la *quinta generación de los SGBD*. Esta información no solo se encontrará almacenada en una base de datos, sino que se podrá generar nueva información a partir de estos módulos. De esta forma se espera contar con lenguajes de consulta en forma natural que permitan diálogos intuitivos con el usuario, así como poderosas herramientas de adquisición y representación de conocimiento, en donde se podrá ser difuso, es decir que se permita el manejo de información incierta y vaga, tan presente en la mayoría de los fenómenos naturales o que son manejados por el hombre.

Estos sistemas que se pueden denominar SGBD *deductivos* son motivo de grandes esfuerzos de investigación.

Las Bases de Datos y los Sistemas de Información
La aplicación de los computadores 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 rutinarias tales como la planilla, la contabilidad, inventario etc. Sin embargo debido a la potencia de los ordenadores en el tiempo el ordenador empezó a incursionar en otras áreas de la empresa, ayudando a la sistematización de las funciones de dirección y constituyendo una herramienta fundamental en la toma de decisiones. Es así como surgen los Sistemas de Información basados en el computador, cuyo principal objetivo es mejorar el proceso de información de la empresa para así lograr la máxima eficiencia.

Sub Sistemas del Sistema de Información
En toda organización se suele distinguir tres niveles de gestión: operacional, táctico, y estratégico, por lo que el SI (Sistema de Información) estará compuesto por 3 sub sistemas estructurados en forma jerárquica y que se corresponden con cada uno de estos tres niveles. En el sub sistema 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 sub sistema será muy voluminoso. De él mediante un proceso de elaboración adecuado, se podrán obtener los datos necesarios más los aportados desde el exterior esto para el funcionamiento de los otros dos sub sistemas, cuyo usuarios tienen unas exigencias muy distintas, y para los que tal volumen de información no solamente sería inadecuado, sino peor aún: inoperante y contraproducente.

Los tres niveles de gestión se encuentran representados en la figura 7.1 a continuación donde se puede observar que, mientras la información se trasmite en forma ascendente, las órdenes y planes se mueven en sentido descendente.

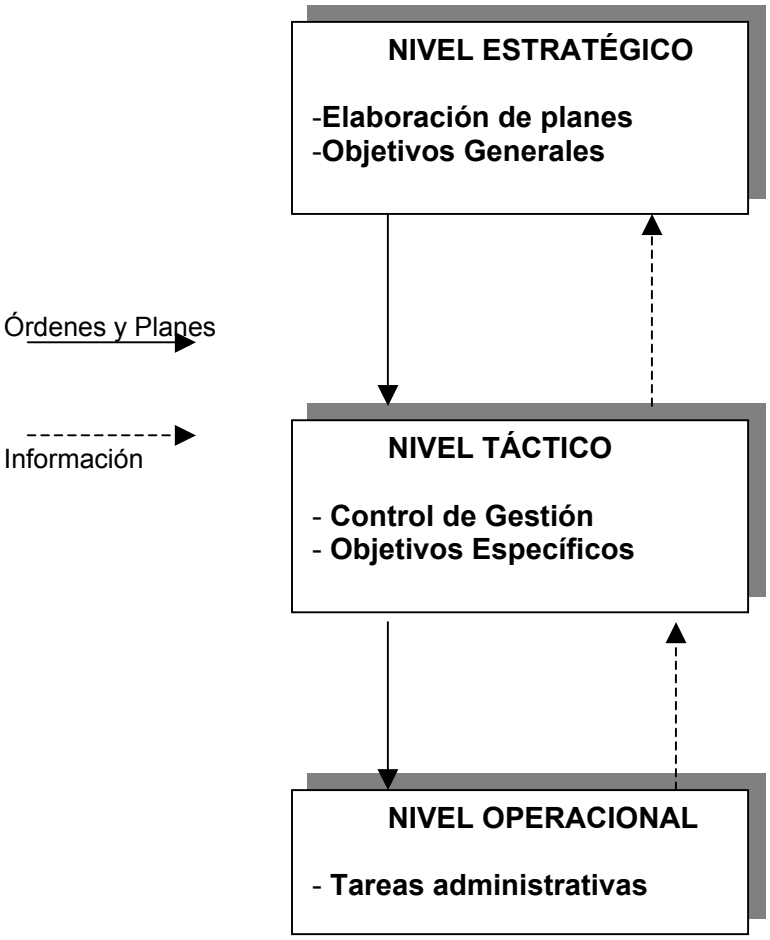


Figura 7.1 Niveles de un Sistema de Información

En un principio se atendieron las necesidades de información propias del nivel administrativo, desarrollando aplicaciones distintas y específicas 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, surgiendo la idea de utilizar una **base de datos** común que incorporara, sin redundancias indeseables, 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 (ver figura abajo)

Posteriormente hemos asistido a la difusión de sistemas diseñados para servir de soporte a la toma de decisiones dirigidos a los directivos (conocidos con las siglas inglesas D.S.S Decision Support Systems 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.

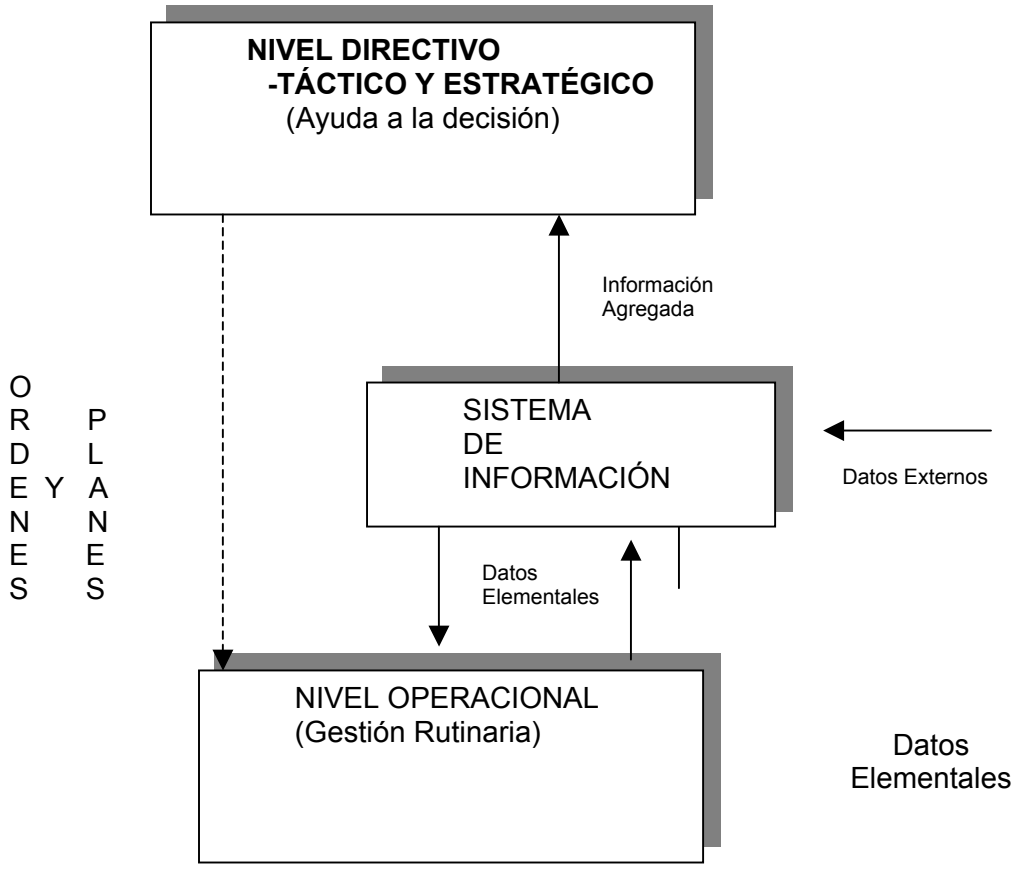


Figura 7.2 Sistema de Información Único(Nivel directivo y operacional)

Aplicaciones de los SGBD en las Empresas e Instituciones

- En el Sistema Bancario
  - Información actualizada a los clientes
  - Estado de cuentas y préstamos
  - Transacciones Bancarias en general

- En las Compañías de Líneas Aéreas
  - Reservaciones de vuelos
  - Información a los clientes sobre cupos disponibles
  - Asignación de asientos
  - Horas de llegada y salida de todos los vuelos

- En las Universidades
  - Asignaturas disponibles para el estudiante
  - Sistema academico
    - Control y seguimiento de las notas de los estudiantes
    - Rendimiento Académico
  - Gestión Financiera de la Universidad

- Gestión de las Becas

- Transacciones con tarjetas de Crédito:
  - Autorización de las compras del cliente
  - Informe mensual para cada cliente
  - Bloqueo de la tarjeta

- Telecomunicaciones:
  - Control de las Llamadas Efectuadas por los usuarios
  - Sistema de facturación
  - Control de Personal

- Finanzas:
  - Sistemas Contables
  - Almacenamiento de Información de la Empresa
  - Control de Ventas y Compras de Documentos Financieros

- Producción:
  - Gestión de la cadena Productiva
  - Seguimiento de las Sucursales de la Empresa
  - Control del Inventario

- Recursos Humanos:
  - Información General de los empleados
  - Elaboración de la Planilla de Pago

- Aplicaciones Internet:
  - Compra/Venta de artículos en Tiendas Virtuales
  - Información en Línea Sobre el Estado de las Cuentas de los Clientes
  - Búsquedas

7.2.2 Sistemas de Bases de Datos Frente a Sistemas de Archivos

Como un aspecto histórico del desarrollo de los sistemas de Bases de datos nos encontramos con los **Sistemas de Archivos**, como su nombre indica son conjuntos de archivos generalmente aislados y un conjunto de programas que manipulan cada uno de estos archivos para obtener información para la empresa. Esta metodología para el procesamiento de los datos dio paso a los sistemas de base de datos actuales, por las problemáticas generadas para almacenar y obtener de forma eficiente los datos requeridos algunas de las cuales se mencionan a continuación :

Redundancia e Inconsistencia de los Datos

Esta problemática se origina del hecho de que tanto los archivos de datos como los programas de aplicación son creados por diferentes programadores en un período de tiempo más o menos largo, programas generalmente escritos en diferentes lenguajes con archivos de datos que utilizan formatos diferentes y por tanto incompatibles entre si. La independencia de estos archivos genera **redundancia** (Exceso innecesario) de la información así por ejemplo la dirección y el teléfono de los clientes de un banco puede estar contenida en un archivo de

cuentas de ahorro y en otro de cuentas corrientes. Por otra parte el problema de la **inconsistencia de los datos** es factible en el sistema de archivos dado que la redundancia tiene como consecuencia, en la medida que los datos crecen la posible representación de la copia de un dato de forma diferente en diferentes archivos así supongamos que un determinado cliente tiene su dirección en 2 archivos diferentes e independientes y este la actualiza solamente en uno de ellos, esto produciría la inconsistencia de los datos.

Dificultad en el acceso de los Datos
El tener la información aislada produce dificultad en obtener la información requerida de los archivos. Así supongamos que se desea averiguar en un determinado banco el nombre de todos los clientes que viven en el distrito 2 de Managua, dado que no se ha requerido de esta información, el programa no existe, solamente podemos acceder a los nombres de todos los clientes del banco, claro está que el o los programadores del mencionado banco pueden diseñar dicho programa pero no es una solución eficiente ya que si posteriormente se requiere los nombres de los clientes del distrito 2 pero que tengan cuentas de ahorro superiores a 5,000 córdobas, de nuevo se tendrá que diseñar el respectivo programa, en otras palabras para cada consulta particular se tendrá que esperar un determinado tiempo para el diseño de la misma y el control adicional que se debe invertir para averiguar si ya fue diseñada o no una determinada consulta dado que el número de ellas crecería de forma considerable en el tiempo.

Aislamiento de los datos
Los programas de Aplicación se tornan cada vez más difíciles de diseñar debido a la independencia que existe entre sí entre los diferentes archivos, además que la información en general está dispersa.

Problemas de Integridad
En general los datos almacenados deben de cumplir ciertos tipos de restricciones de consistencia así por ejemplo que una cuenta bancaria sea siempre mayor o igual a 1000 córdobas, como se ha indicado antes se pueden diseñar los programas respectivos siempre y cuando la información se encuentre en un mismo archivo, el problema resulta cuando dichas restricciones de integridad se refieren a datos que se encuentran en archivos diferentes con formatos diferentes.

Problemas de atomicidad
La atomicidad en este contexto se refiere a un conjunto de operaciones (Una Transacción) que se deben de efectuar en la base de datos de manera completa, lo cual no siempre se produce debido a la ocurrencia de fallos, así por ejemplo supongamos que una transacción consta de 10 operaciones y el fallo se produce al comenzar la sexta operación, en este caso decimos que se ha violado la atomicidad de la transacción. Por lo antes expuesto el sistema debe de contar con un mecanismo para mantener la base de datos en un estado consistente, este existe en los SGBD pero no en un Sistema de Archivos.

Anomalías en el acceso concurrente
Los sistemas operativos actuales hacen posible diseñar sistemas de bases de datos con módulos encargados de resolver la problemática del acceso concurrente a los datos, este aspecto sería muy difícil de implementar en los sistemas de archivos debido a la independencia total de los datos y en general a

la diferencia de formato de estos. Una de estas anomalías puede verse claramente en el siguiente ejemplo :Considérese una cuenta bancaria A con 500 córdobas, supongamos que dos clientes retiran fondos de dicha cuenta, uno retira 50 y el otro 100 en aproximadamente el mismo tiempo (Desde dos terminales distintas), ambos usuarios leen de disco 500, y luego el disco será actualizado a 450 o 400 dependiendo quien guarde por ultimo, esto por supuesto dejaría la base de datos inconsistente.

Problemas de Seguridad
En un sistema de archivos este aspecto es también difícil de implementarlo ya que los archivos no conforman un sistema y sería extremadamente difícil el acceso a determinada porción de los datos, este aspecto de la seguridad es muy importante en las bases de datos debido a que en una empresa no interesa y no es conveniente que todos tengan acceso a todos los datos así en un sistema bancario el personal de nominas no tiene porque conocer la información almacenada sobre las cuentas de los clientes

En Conclusión estas y otras dificultades generadas por los sistemas de archivos han motivado al diseño de los Sistemas de Bases de Datos los cuales precisamente se encargan de llevar a cabo en forma adecuada las diferentes problemáticas discutidas anteriormente

7.2.3 Visión de los Datos

El usuario a diferencia de lo que ocurría con el uso de organizaciones clásicas de almacenamiento de la información, no tiene necesidad de conocer como se organizan los datos fisicamente en la base de datos.

El usuario ya sea un experto o lego en informática, conoce las carácterísticas del problema que representa la base de datos. El usuario conoce la organización, todo o parte del sistema, la información que se maneja, las relaciones existentes en esta información, como debe ser tratada esta información y en que tiempo. En definitiva el usuario conoce el dominio del problema que va a ser tratado con una base de datos. Si es esta la visión de los datos que la base de datos presenta al usuario, el usuario podrá utilizar el sistema e integrarse en él adecuadamente.

Si por el contrario, la base de datos presenta al usuario la visión de cómo está organizada la información fisicamente tanto los datos como las múltiples relaciones que pueden existir entre ellos, este no reconocería el problema de la organización que está tratando si no tiene una alta formación para reconocer estructuras físicas muy complejas.

Dependiendo de quien acceda o use la base de datos, esta debe presentar una visión de los datos que sea capaz de reconocer, interpretar y manejar. Además si una base de datos debe satisfacer las características antes expuestas, la organización física de los datos debe ser lo más independiente posible de los procedimientos que manejan la información y de los posibles cambios que surjan en el dominio del problema.
Se puede hablar entonces que existen tres visiones de los datos en una base de datos:

Visión externa: Es la visión de datos que tienen los usuario finales de una base de datos. Un usario Un (operador de terminal) trata sólo una visión parcial de la información, sólo aquella que interviene en el dominio de la actividad (El subsistema de la organización donde interviene). Este usuario debe "ver" la información que maneja como un registro, una ficha de datos con independencia con independencia de a qué entidad pertenecen los ítems de datos, correspondientes a ese registro, en el dominio del problema y en que relaciones se ven implicados esos datos. Cada usuario puede tener su "Registro Particular"

Visión Conceptual: Es la visión o representación del problema tal y como éste se representa en el mundo real.

La visión conceptual de una base de datos es una representación abstracta del problema e independiente en principio, de como va a ser tratada esta información, de las visiones Externas y de cómo va a ser almacenada fisicamente la información, de modo que esta visión no cambia a menos que cambie la naturaleza del problema.

Visión Física: Esta visión describe las estructuras u organizaciones físicas, dispositivos, volúmenes, ficheros, tipos de datos, punteros, etc., estructuras de mayor o menor complejidad que representan el dominio del problema de una forma entendible por el sistema informático.

Dado que los usuarios de la base de datos pueden por lo menos percibir tres visiones diferentes de los datos con un mismo dominio podemos decir que estas tres formas de descripción de los datos son en realidad, tres niveles de abstracción diferentes que describen un mismo problema.

En cada nivel se describen aquellos objetos de interés que pueden ser entendidos por los usuarios de ese nivel. Así el usuario final sólo entiende de registros o formas de comunicación similares a los documentos externos que son manejados en la organización, mientras que el diseñador o analista de sistema sólo entiende de las clases de objetos que intervienen en el dominio del problemaque la organización desea que se trate mediante una base de datos, de las relaciones existentes entre ellas y de los procedimientos que son llevados a cabo para la solución del problema que se está tratando.

Por otro lado es el administrador de la base de datos el encargado de describir el nivel físico para determinar aquella organización física que pueda garantizar el desempeño óptimo del sistema.

Las visiones diferentes de la base de datos indicadas anteriormente se muestran en el gráfico abajo.

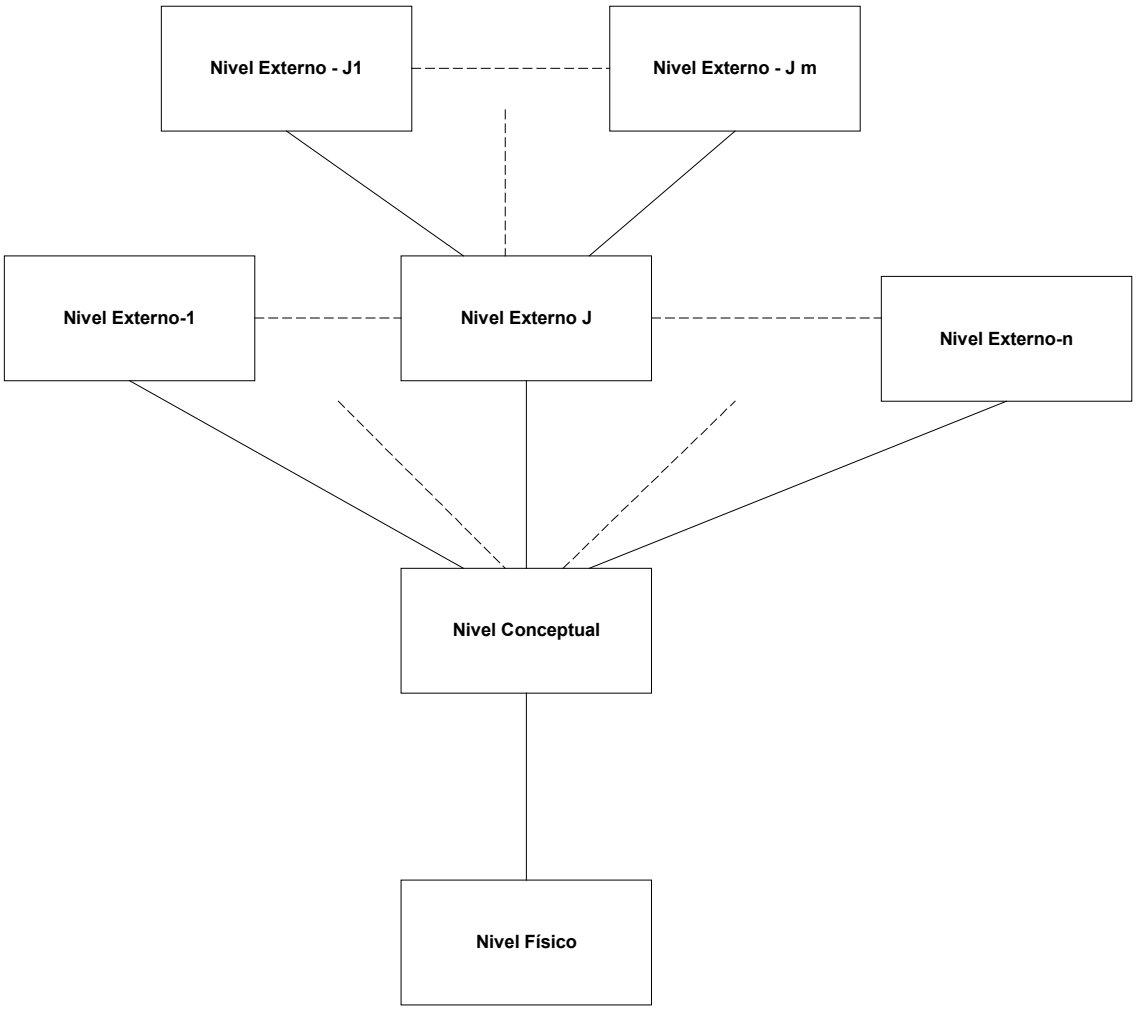


Figura 7.3 Tres visiones diferentes en la base de datos

La descripción de los datos a estos tres niveles de abstracción garantiza en gran medida la independencia de los datos que es uno de los objetivos a lograr en el diseño de las bases de datos, en otras palabras:

- Que la Organización Física de los datos pueda ser modificada, y que ello no implique la modificación de los programas de aplicación

- Que pueda ser modificada la representación conceptual del problema que está representado por la base de datos y que ello no implique la modificación de la estructura interna de la información, claro está que si se consideran nuevos objetos estos deben de ser reflejados, ni la de los programas de aplicación, siempre y cuando no se eliminen objetos que intervienen en estos programas

- Es evidente que las visiones externas pueden cambiar conforme nuevos requerimentos o necesidades funcionales o de operación son incorporadas al dominio del problema por la organización y/o su entorno,

y sin por ello deba ser modificada ninguna de las descripciones de los datos a ninguno de los restantes niveles de abstracción.

Para que exista esta independencia de los datos, tan deseada, los tres niveles de abstracción mediante los cuales los datos se describen deben de ser totalmente independientes, lo cual no es totalmente cierto en la mayoría de las bases de datos. Sin embargo se puede obtener una buen independencia de los datos, si se cumple que:

- La representación interna de los datos no es una fiel traducción de la representación conceptual. Una misma representación conceptual puede representarse de varias formas fisicamente. Son las exigencias funcionales y de desempeño las que van a determinar esta representación.

- Si bien las representaciones externas son dependientes de la representación conceptual por el hecho de que los registros (representaciones externas) deben de estar formados por ítems de datos existentes y por tanto representados en el nivel conceptual, la estructura de estos registros (Cantidad de ítems y dispocisión de los mismos) debe de ser independiente de cómo estos ítems han sido representados en el nivel conceptual y de las relaciones que mantienen en el mismo.

7.2.3.1 Independencia del nivel de Descripción Conceptual

El nivel de descripción conceptual es seguramente el mas importante, o por lo meos aquel en el que se apoyan en menor o mayor grado los otros niveles y, con seguridad, en el que, en base a su calidad, se garantiza que la base de datos solucione el problema de la organización.

Una descripción conceptual de calidad describirá todas y cada una de las entidades o clases de objetos que intervienen en el problema, sus propiedades y atributos, así como las características de las relaciones existentes entre las mismas. En este nivel se describe cada uno de los ítems de datos o elementos de información que intervienen en el comportamiento del sistema y cuya información es necesario considerar.

Pero existen muchas maneras válidas de describir un sistema para obtener una visión conceptual de un determinado problema, aunque cualquier procedimiento no puede reconocer e interpretar cualquier descripción. Se puede hacer una descripción severa dentro de un problemautilizando técnicas como los diagramas de estructuras, de contexto, tablas de cualquier tipo, diagramas de entidades y relaciones, árboles, tablas, redes etc y sin embargo no todos los los procedimientos son capaces de interpretar cualquier clase de representación conceptual.

De hecho existen muchas formas de representar de forma abstracta un fenómeno (Un problema) observado del mundo real. Una representación abstracta (conceptual) de un problema supone la aplicación de una serie de reglas que restringen y dirigen la forma en que ese problema es representado. Pero por otra parte el problema del mundo real representado conceptualmente debe ser independiente de esta representación.

Por lo antes expuesto cabe hablar de un cuarto nivel de abstracción en la representación de la información en una base de datos, (ver figura abajo), se trata del nivel lógico o canónico, este es derivado de la descripción conceptual de la base de datos en base a la aplicación de una serie de reglas y restricciones

que toman en cuenta como la información representada puede ser tratada por los procedimientos que van a manejar y definir la información en base a las otras representaciones. Así, pueden existir múchas representaciones lógicas de una misma representación conceptual, al igual que de una representación lógica pueden ser derivadas muchas representaciones externas.
La inclusión de este nuevo nivel de representación del dominio del problema, un nivel dependiente del software encargado de manipular la información, va a garantizar la independencia d ela información en una base de datos.

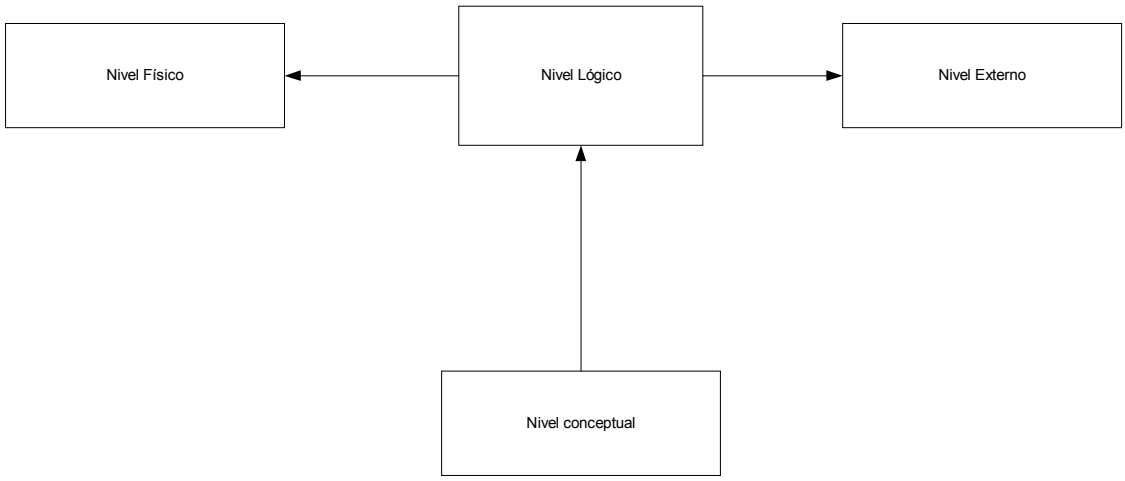


Figura 7.4 Niveles de Abstracción en la Base de Datos

7.2.4 Lenguajes de Bases de Datos

Un sistema de base de Datos proporciona varios tipos de lenguajes: Un lenguaje de nominado LDD, lenguaje de Definición de Datos (en Inglés DDL, Data Definition Language) y otro denominado LMD, Lenguaje de Manipulación de Datos (en Inglés DML, Data Manipulation Language), el primero tiene como función primordial el definir la estructura de la base de datos: Nombre de las tablas, sus atributos, los tipos de los atributos etc, mientras que el segundo tiene como función trabajar sobre los datos de la base de datos para obtener la información pertinente o modificar la base de Datos.

Cabe mencionar que el LDD cuenta con un sublenguaje encargado del control y seguridad de los datos, permite el control del acceso a la información almacenada en el diccionario de datos(definición de privilegios y tipos de acceso), así como el control de la seguridad de los datos, este sublenguaje se conoce como LCD, Lenguaje de Control de Datos (en Inglés DCL, Data Control Language).

El lenguaje utilizado ampliamente por la mayoría de SGBD es el denominado **SQL (Structured Query Language)**, el cual se puede utilizar tanto como LDD como LMD o LCD.

7.2.5 Acceso a la Base de Datos desde Programas de Aplicación

El Lenguaje SQL a pesar de su potencia para la manipulación de los datos en general no satisface todos los requerimientos de los usuarios por lo que se hace necesario combinar su potencia con la de un lenguaje de programación para completar estos requerimientos, así por ejemplo el uso de formularios como interfase para la introducción de datos en la base de datos, las validaciones complejas de los datos, generación de reportes etc, son tareas que en general no puede efectuar SQL por lo que se requiere utilizarlo dentro de un lenguaje anfitrión como C, C++ o Cualquier otro.

Para acceder de esta manera a la base de Datos existen fundamentalmente dos maneras:

- Proporcionando una interfaz de programas de aplicación (Un Conjunto de procedimientos) para enviar instrucciones SQL a la base de datos y posteriormente recuperar los resultados.
  Las interfaces estándar son: ODBC (Open Data Base Connectivity) diseñado por la empresa Microsoft para la conexión con Bases de Datos desde el lenguaje C, otro Estándar es el JDBC (Java Data Base Connectivity) el cual proporciona características similares al ODBC pero para el lenguaje Java.

- Otra manera de hacer la conexión con las Bases de Datos desde el lenguaje anfitrión es incorporando llamadas directamente desde el lenguaje anfitrión, en este caso un carácter o conjunto de caracteres precede a la sentencia SQL y otro símbolo especial para indicar el final de esta sentencia dentro del lenguaje anfitrión. En este caso se requiere de un precompilador, el cual traduce los comandos SQL dentro del lenguaje anfitrión a procedimientos de este lenguaje previo a la compilación del programa en cuestión.

7.2.6 Usuarios y Administradores de la Base de Datos

A grandes rasgos las personas que trabajan con las Bases de Datos podemos dividirlas en dos grandes grupos en dependencia de la utilización que realicen de la base de datos, estos son**: Los Usuarios de la Base de Datos** y **los administradores de la Base de Datos.**

7.2.6.1 Los Tipos de Usuarios de las Bases de Datos

Los Usuarios Normales:

Son usuarios que interactúan con el sistema mediante la invocación de alguno de los programas de aplicación diseñado para satisfacer las necesidad de información de una determinada empresa. por ejemplo, un cajero de un banco que necesita transferir 50 córdobas de la cuenta A a la cuenta B invoca un programa diseñado para este fin y que los informáticos que diseñaron la aplicación lo han Llamado por ejemplo **Transferir.**

La interfaz de usuario que generalmente utilizan estos tipos de usuarios son los formularios donde el usuario llena los campos apropiados para realizar una determinada acción de la aplicación. Los usuarios normales pueden también simplemente ejecutar y leer los informes generados por la aplicación.

Los Programadores de Aplicaciones:

Son profesionales informáticos que escriben programas de aplicación. Los programadores de aplicaciones cuentan con herramientas para tal fin. Las herramientas de desarrollo rápido de aplicaciones (DRA) son herramientas que permiten al programador de aplicaciones construir formularios e informes sin escribir un programa. Por otra parte se pueden utilizar lenguajes de especiales de programación que contiene los comando básicos de un lenguaje de un lenguaje de tercera generación más un conjunto de facilidades que mejoran la interacción con la base de datos, este es el caso de Access, Visual Basic y FoxPro productos de Microsoft, también podemos mencionar los módulos PL/SQL(Lenguaje de Cuarta Generación) y Forms de la Empresa Oracle.

Los Usuarios Sofisticados:
Estos interactúan con el sistema sin programas escritos. Su misión es la de formular consultas mediante el lenguaje SQL o utilizando herramientas especializadas para tal fin como las de Procesamiento Analítico en Línea (**OLAP, Online Analytical Processing**) las cuales simplifican la labor de los analistas permitiéndoles ver resúmenes de datos de formas diferentes, así por ejemplo un analista tienen la posibilidad con estas herramientas de visualizar las ventas totales de su empresa por Departamento o región, por producto. Las herramientas también permiten al analista seleccionar regiones específicas para su análisis, examinar los datos con más detalle (Por ejemplo, ventas por ciudad dentro de una región) o examinar los datos con menos detalle, agrupando los datos por Categoría por ejemplo.
Otra clase de herramienta para los análisis de los datos son las de **Recopilación de Datos (Data Mining),** las cuales ayudan a encontrar ciertas clases de patrones de datos.

Usuarios Especializados:
Son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas que no se adecuan al marco tradicional de las bases de datos. entre estas aplicaciones están Los sistemas de diseño asistido por computador, Sistemas de Bases de Datos del Conocimiento y Sistemas Expertos, sistemas que almacenan los Datos con tipos de datos Complejos (Gráficos y Audio, por ejemplo)

7.2.6.2 Administrador de la Base de Datos

El Administrador de la Base de Datos (ABD) es la persona que tiene el control absoluto sobre el Sistema.

Funciones Principales del Administrador

- Definición del Esquema.
  El ABD es responsable de crear el esqueleto original de la base de datos escribiendo un conjunto de instrucciones del lenguaje d definición de datos.

- Definición de la Estructura de las Tablas y el metodo de acceso a los Datos

- Modificación del esquema y de la organización fisica de los Datos Los ABD se encargan de realizar cambios en el esquema y en la organización

Física que reflejan las necesidades de actualización de la empresa o para mejorar el rendimiento de la aplicación.

- Concesión de los permisos para el acceso a los datos
  La concesión de diferentes tipos de autorización permite al administrador de la base de datos decidir a que porciones de esta puede acceder cada uno de los usuarios. La información de autorización se mantiene en una estructura especial del sistema en la cual el sistema de Base de Datos Consulta cuando se intenta el acceso a los datos en el sistema.

Mantenimiento Rutinario:

- Realizar una copia de Seguridad de la Base de Datos en períodos establecidos, sobre cinta o sobre servidores remotos esto con el fin de prevenir la pérdida de los datos en caso de desastres naturales

- El administrador debe de cerciorarse de que exista suficiente espacio libre en disco para las operaciones normales del sistema y aumentar este espacio según se requiera.

- Supervisión de cada uno de los trabajos que se ejecutan de forma simultanea en la base de datos y asegurarse que el rendimiento no se degrade por tareas que consumen una cantidad considerable de tiempo.

7.2.7 Gestión de Transacciones y Control de Concurrencia:

Conceptos Básicos del Módulo

Varias operaciones sobre la base de datos forman a menudo una unidad lógica de trabajo. Un ejemplo de ello puede ser es la transferencia de fondos de una cuenta Bancaria A a otra cuenta B, en la que existen dos operaciones fundamentales que son modificar la cuenta y la cuenta B, estas operaciones forman una unidad lógica dado que si solo se realiza una parte de estas operaciones el sistema de Base de Datos puede quedar en un estado inconsistente, es decir con datos que no se ajustan a la realidad.

Una transacción es una colección de operaciones que se lleva a cabo como una única función lógica en una aplicación de base de datos. Cada transacción es una unidad de atomicidad y Consistencia, entendiéndose por atomicidad el hecho de que todas las operaciones deben de realizarse por completo y por consistencia el hecho de que la transacción debe de respetar las reglas establecidas en los datos, en el ejemplo de arriba puede ser esta regla el hecho de que A+ B se mantenga inalterable antes y después de la transacción.

Finalmente y siempre referido al ejemplo de la transferencia de fondos de una cuenta a otra, tras la ejecución correcta de la transferencia de fondos los nuevos valores de las cuentas A y B deben de persistir a pesar de que ocurra algún fallo del sistema. Este requisito de persistencia de los datos modificados se denomina **durabilidad.**

Asegurar las propiedades de Atomicidad y durabilidad es responsabilidad del propio sistema de base de Datos, específicamente del módulo de **Gestión de transacciones,** quien se encargará también de dejar la base de datos como estaba antes de comenzar la transacción si ocurriese un fallo durante la ejecución de la misma y gestionar las transacciones de forma concurrente, es

decir garantizar también la consistencia de la base de datos cuando las transacciones sean efectuadas en un entorno de red, esto ultimo es responsabilidad del gestor de **Control de Concurrencias.**

Los Sistemas de Bases de Datos diseñados para computadoras personales pequeñas puede que no tengan todas las características vistas. así por ejemplo muchos de los sistemas pequeños imponen la restricción de permitir el acceso a un único usuario a la base de datos en un instante de tiempo. Otros dejan las tareas de efectuar copias de seguridad a los usuarios. Estas restricciones permiten el diseño de un gestor más pequeño, con menos requerimientos de hardware especialmente de memoria principal. Es claro que a medida que la empresa crezca en esa medida se deben ir incorporando los aspectos antes mencionados, lo que implicará eventualmente cambio de hardware y software.

7.2.8 Estructura de un Sistema de Base de Datos:

Un Sistema de base de Datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Los componentes funcionales de un sistema de base de datos se puede dividir en grades rasgos en los componentes : **Gestor de Almacenamiento** y **el Procesador de Consultas.**

7.2.8.1 El Gestor de Almacenamiento:

El gestor de almacenamiento es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel (almacenamiento físico), los programas de aplicación y consultas que eventualmente se le harán al sistema. El gestor de almacenamiento es responsable de la interacción con el gestor de archivos. Los datos en bruto se almacenan en disco usando un sistema de archivos que habitualmente son parte del sistema operativo, el gestor traduce los comandos SQL a ordenes de un sistema de archivos de bajo nivel. Por tanto el gestor de almacenamiento tiene como tarea fundamental del almacenamiento, recuperación y actualización de los datos en una base de Datos.
Los componente básicos del gestor son :

El Gestor de Autorización e Integridad
Su función es comprobar que se satisfagan las restricciones de integridad y la autorización de los usuarios para acceder a los datos.

El Gestor de Transacciones
Este módulo asegura que la base de datos quede en un estado consistente a pesar de los fallos que puedan ocurrir en el sistema y que la ejecución concurrente de las transacciones se efectúen sin generar conflictos.

El Gestor de Archivos
Este gestiona los espacios requeridos de almacenamiento de disco y las estructuras de datos utilizadas para representar la información almacenada en disco.

El Gestor de memoria Intermedia
Este módulo es el responsable de traer los datos del disco a Memoria principal y decidir que datos tratar en la memoria caché. El gestor de memoria intermedia es

una parte crítica de los sistemas de Bases de Datos, ya que permite que esta maneje datos con una capacidad mayor que la memoria principal.

Estructuras de Datos
El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema, estas son:

- Archivos de Datos
  Para almacenar la Base de Datos en sí

- Diccionario de Datos
  Almacena metadatos acerca de los componentes de la estructura de la Base de Datos, en particular el esquema de la base de datos

- Índices
  Conjuntos de archivos auxiliares para acelerar las búsquedas en la Base de Datos.

Los componentes del procesador de consultas incluyen:

- Interprete del LDD
  Este interpreta las instrucciones del LDD y registra cada una de las definiciones en el Diccionario de Datos

- Compilador del LMD
  Traduce las instrucciones del LMD en un lenguaje de consultas como SQL a un plan de evaluación el cual consiste en instrucciones de bajo nivel compatibles con el motor de evaluación de consultas, también realiza el proceso de optimización de las consultas es decir elige el plan de evaluación de cada una de las consultas para minimizar el costo en tiempo.

- Motor de Evaluación de Consultas
  Se encarga de ejecutar las instrucciones de bajo nivel generadas por el compilador del LMD

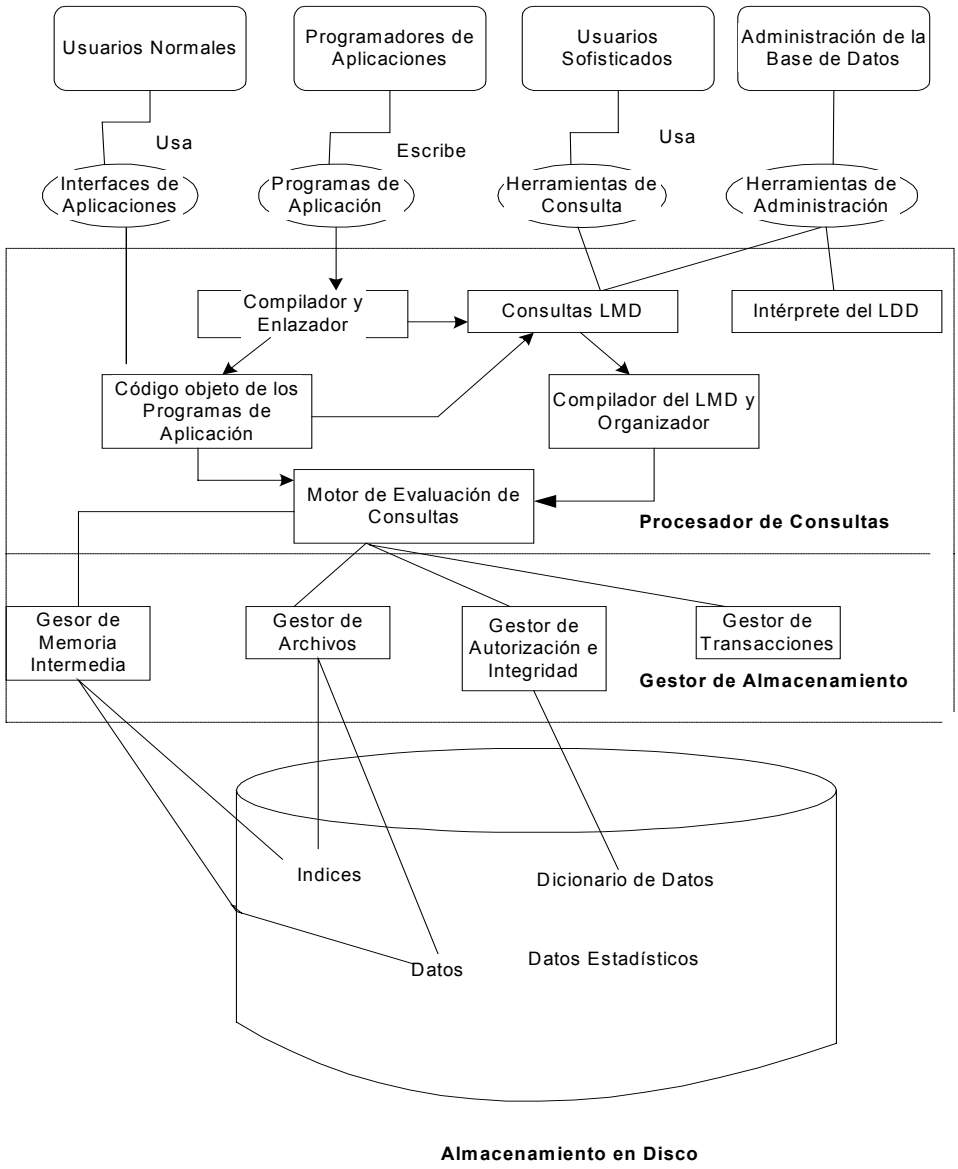


Figura 7.5 Estructura del Sistema

7.3 Tema 2: "Diseño de Base de Datos"

TEMA N° 2

OBJETIVOS:

- Conocer las diferentes etapas para el diseño de una base de datos
- Saber diferenciar los tres tipos de diseño
- Saber construir el Grafo Relacional a partir del esquema E-R
- Poder Implementar el Diseño físico a partir del Grafo Relacional

CONTENIDO

- Diseño Lógico de las Bases de Datos en el modelo relacional
  - Introducción
  - Etapas en una metodología de Diseño
    - Diseño Conceptual
    - Diseño Lógico
    - Diseño físico
- Modelo de los Datos
  - El Modelo Relacional
  - El Modelo Entidad Relación
- Transformación del Esquema Conceptual al relacional
  - El grafo relacional
- Otros Modelos de Datos
  - El Modelo de Datos Orientado a Objetos
  - El Modelo de Datos Relacional Orientado a Objetos
  - Los modelos Semiestructurados

DURACIÓN: 2 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill
- Adoración de Miguel, Mario Piattini. Fundamentos y Modelos de Base de Datos. 2ª Edición. RAMA

7.3.1 Diseño Lógico de las Bases de Datos en el modelo Relacional

7.3.1.1 Introducción

Una vez expuestos los fundamentos de las bases de datos y analizadas las características fundamentales de los modelos Entidad/relación y relacional, se puede abordar el problema del diseño de una base de datos de manera práctica
Formas Normales. La metodología de diseño que se propone comienza con una primera etapa en la cual se modela el mundo real utilizando un esquema E/R, para luego transformar este al modelo relacional (Esquema Relacional). La teoría de Normalización como parte del diseño nos va a permitir comprobar si el Esquema relacional cumple con una serie de requisitos en cuanto a la teoría de Normalización se refiere de modo que de no ser así llevarlo a las formas normales más adecuadas

7.3.1.2 Etapas de una Metodología de Diseño

En el proceso de diseño de una base de datos hemos de distinguir tres grandes fases:Diseño conceptual, Diseño lógico y Diseño físico.

Diseño conceptual:
El objetivo de esta fase es obtener una representación adecuada de los recursos de información de la empresa, sin tomar en cuenta las necesidades de los usuarios o aplicaciones particulares que se deseen del sistema, no tomando en cuenta además consideraciones relacionadas con la eficiencia del ordenador (servidores/Estaciones de trabajo).En esta etapa la comunicación con el o los usuarios es vital para la feliz culminación del diseño por lo que la utilización de un esquema E/R de fácil comprensión al usuario ayudará a la comunicación Diseñador-Usuario cuyo fin último es el de efectuar una representación lo más fiel posible de la información de la empresa.

Diseño Lógico
Esta parte del diseño está basada en el Diseño Conceptual, es decir una vez generado el esquema E/R adecuado para representar la información de la empresa, este se transforma en un modelo de datos estrechamente relacionado al modelo en que se sustenta el SGBD, en este caso el modelo relacional. Es decir se trata de pasar del esquema E/R al Esquema Relacional, esta conversión de un esquema a otro se estudiará posteriormente.

Diseño Físico
El objetivo de este diseño consiste en obtener la implementación del esquema lógico lo más eficiente posible

7.3.2 Modelos de los Datos

Introducción
En una primera a proximación 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 denominamos **esquema.** Según el nivel de abstracción de la arquitectura ANSI a tres niveles en la que se encuentre la estructura descrita, el modelo que permite su descripción será un modelo: **externo**, **global** o **interno.**
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 a describir los datos para el conjunto de usuarios, se puede decir que es la información a nivel de empresa

MODELOS GLOBALES

- **CONCEPTUALES**
  - Enfocados a describir el mundo real con independencia de la máquina
- **CONVENCIONALES O LÓGICOS**
  - Implementados en SGBD
    - Jerárquico
    - Codasyl
    - Relacional

Los modelos internos

- Están orientados a la máquina, siendo sus elementos de descripción punteros, índices, agrupamientos, etc.

El modelo Relacional

En este modelo lógico los datos y las relaciones entre ellos son representados por un solo objeto denominado Tabla donde cada tabla está compuesta por Columnas y filas. A continuación se presenta un ejemplo de este modelo :

Id_cliente	Nombre_Cliente	Calle_Cliente	Ciudad_Cliente
1	Juan	Sutiava	León
2	Pedro	Jalteva	Granada
3	Carlos	Monzón	Matagalpa
4	María	Central	León

Figura 7.6 Tabla Cliente

Id_cliente	Número_Cuenta
1	C2
2	C3
3	C4
4	C1
2	C5

Figura 7.7 Tabla Cliente_Cuenta

Número_Cuenta	Saldo
C1	1000
C2	1200
C3	1500
C4	600
C5	1700

Figura 7.8 Tabla Cuenta

En este caso la tabla clientes almacena información sobre cada uno de los clientes, del Banco Popular por otra parte la tabla cuenta almacena información sobre el estado de cada una de las cuentas del mismo banco y finalmente la tabla Cuenta_Cliente lo que almacena es una relación entre los clientes y las cuentas, es decir cuales son los clientes de este banco que poseen cuentas y cuales son estas.
El modelo relacional es un ejemplo de modelo basado en registros pues en general la implementación se realiza utilizando registros de formato fijo de varios tipos.

Modelo Entidad Relación
El modelo conceptual de base datos Entidad Relación está basado en una percepción del mundo real que consta de : Una colección de objetos básicos denominados Entidades y de Relaciones entre estos objetos.

En el ejemplo anterior : 1, Juan, Sutiava, León es una entidad que define de forma única a un Cliente del Banco Popular. por otra parte :
C5, 1700 identifica de forma única el estado de esta cuenta es decir define una cuenta. Finalmente 2,C5 es una relación la cual indica que el cliente cuyo Id_cliente es 2 tiene asignado la cuenta cuyo código es C5, decimos entonces que 2 C5 vincula a dos entidades :
2, Pedro, Jalteva, Granada y la entidad : C5, 1700.
Los componentes básicos tanto de las entidades como la de las relaciones reciben el nombre de atributos.
En este modelo las entidades del mismo tipo se denominan :Conjunto de entidades, similarmente las relaciones del mismo tipo se denominan Conjunto de Relaciones, ambas tiene una representación gráfica en este modelo.

Componentes Básicos del Modelo :

- Rectángulos: Representan Conjunto de entidades
- Elipses: Representan atributos
- Rombos: Representan Conjunto de Relaciones
- Líneas: Unen los atributos a los conjuntos entidades o Conjuntos Relaciones, unen también Conjunto de entidades con Conjunto de relaciones

A continuación se presenta el gráfico completo del diagrama Entidad-Relación referido al Ejemplo Bancario anterior

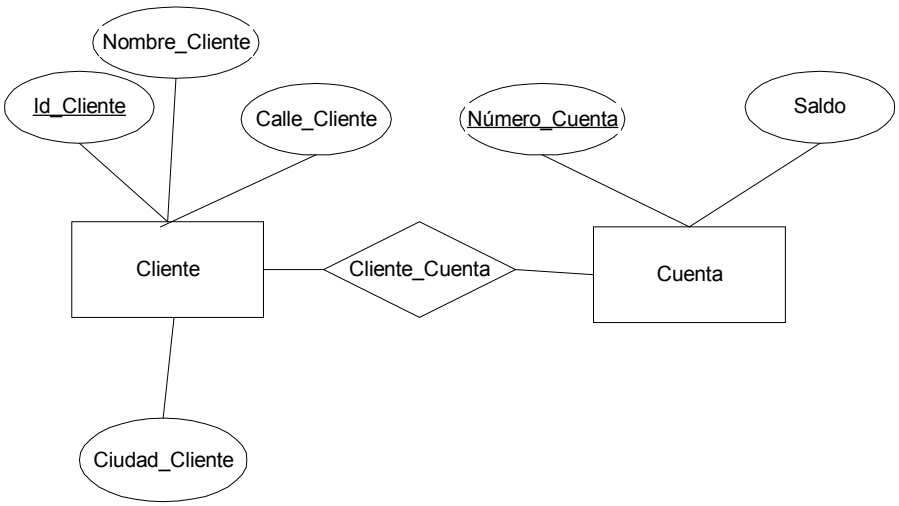


Figura 7.9 Ejemplo de Diagrama E/R

7.3.3 Transformación del Esquema Conceptual al Relacional

El paso de un esquema en el modelo E/R se basa en los tres principios siguientes:

- Cualquier tipo de conjunto entidad se convierte en una relación
- Cualquier tipo de conjunto relación con cardinalidad muchos a muchos (N:M) se transforma en una relación
- En el caso de los conjuntos de entidades débiles en que la cardinalidad del conjunto relación es de 1:N, este no produce ninguna relación sino que solamente el Conjunto Entidad fuerte propaga o hereda la llave primaria a la entidad débil. Por otra parte si la cardinalidad es 1:N y no existe entidad débil, es decir ambas entidades son fuertes en este caso el conjunto relación si genera una tabla o relación.

A primera vista y por lo antes indicado se puede observar que en el paso de modelo E/R al relacional se pierde semántica, dado que ya no existen los dos objetos básicos: Entidades y relaciones sino uno solo el objeto relación o tablade forma que ya no es posible distinguir entre unas y otras. En el caso de los Conjuntos de relaciones con cardinalidad 1:N la pérdida es mayor ya que incluso desaparece el nombre del conjunto relación.

Ejemplo
Supóngase que por medio de entrevistas con los usuarios del sistema de bases de a desarrollar se ha analizado “El mundo real” real determinándose “El Universo del discurso” relacionado con editoriales que editan libros, llegándose a diseñar el siguiente esquema conceptual (E/R):

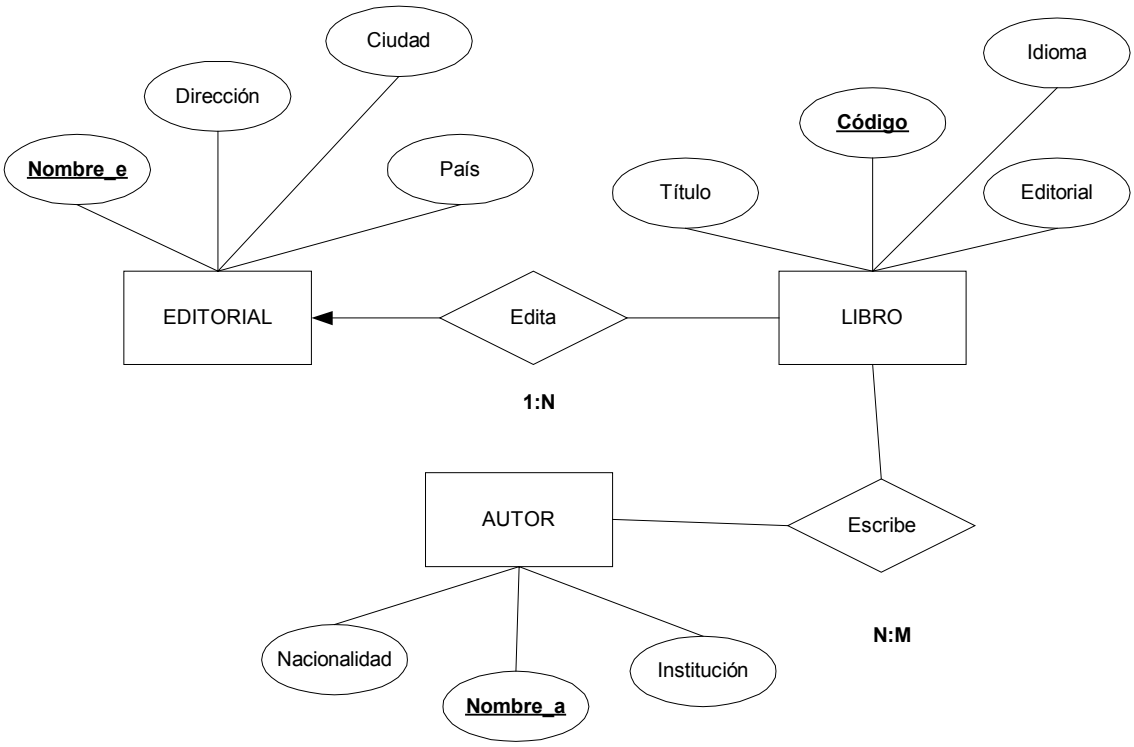
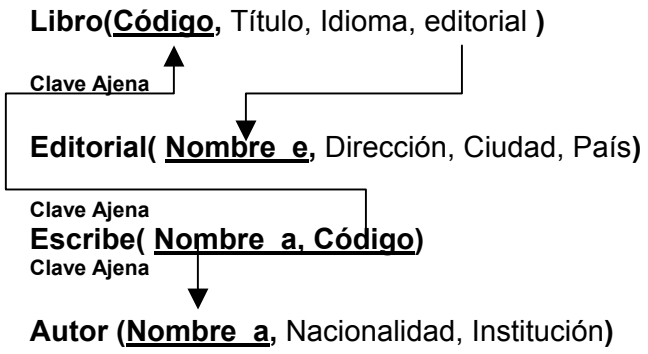


Figura 7.10 Diagrama E/R Sobre información de las Editoriales

Esquema Relacional correspondiente



Para evitar confusiones en el diagrama relacional, en lugar de utilizar la frase "clave ajena" en el diagrama, se coloca un doble sub rayado en la parte superior del o los atributos que conforman una clave ajena respecto a otra relación o tabla.

En este diagrama siguiendo las reglas antes indicadas para obtener el esquema relacional a partir del esquema conceptual se destaca que:

- Los Conjuntos Entidades :Editorial, Libro y Autor son relaciones indicándose en cada una la llave Primaria
- Existen en el Diagrama E/R la representación de dos Conjuntos de relaciones donde uno de ellos: **Edita** no genera tabla pues la cardinalidad de la relación es de **1:N.** Por otra parte **Escribe** tiene cardinalidad N:M y

genera una tabla en el esquema relacional, esta tabla tiene dos atributos heredados de los conjuntos de entidades fuertes que están involucrados en esta relación: El atributo **Nombre_a** y **Código.**

- Se debe Notar que las tablas ajenas de la tabla referenciada no tienen que llamarse obligatoriamente de igual manera que las claves primarias de la tabla referenciada. Muchas veces incluso, es costumbre asignar a la clave ajena el nombre de la tabla referenciada tal como se ha hecho al propagar la clave de **EDITORIAL** a la relación **LIBRO.**

Opciones que puede tomar la clave Ajena en la Transformación de Esquemas

Si bien se ha indicado anteriormente que el paso del Esquema E/R al relacional implica pérdida de semántica, se debe de procurar que esta pérdida sea lo más mínima posible, además se debe de asegurar la máxima consistencia de los datos, es decir mantener la integridad de la base de datos al efectuarse las diferentes operaciones de actualización. La opciones de clave ajena ("NO ACTION","SET NULL","CASCADE" Y "SET DEFAULT") del DDL de SQL utilizadas cuando se borran o modifican ocurrencias de la relación referenciada ayudan a conseguir el objetivo de mantener la integridad de la Base de Datos.

(a) Cardinalidad del Conjunto Relación: **1:N**
Sea el conjunto relación **Edita** del ejemplo anterior, donde se supone que todo libro es siempre editado por una única Editorial, abajo se muestra el trozo de diagrama E/R y su correspondiente transformación relacional.

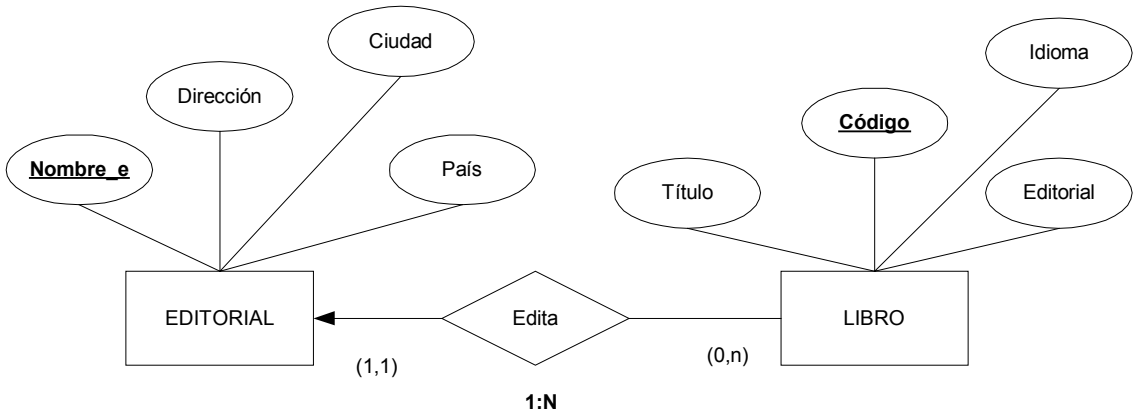
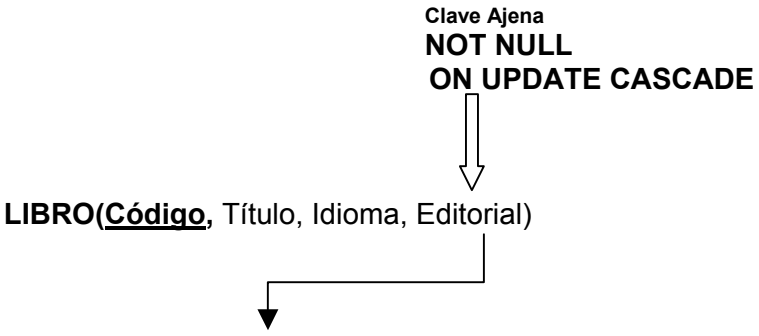


Figura 7.11 Diagrama E/R, base para el diseño del esquema relacional

Modelo Relacional



Las posibles opciones de borrado que pudieran aplicarse en este caso son:

- Restingido("NO ACTION"): Esta cláusula impide el borrado o actualización de una ocurrencia en EDITORIAL en tanto existan en la base de datos libros editados por dicha editorial (Esta es la opción que el sistema toma en caso de que no se de ninguna de forma explicita)

- Cascade ("CASCADE") :Se utilizará esta cláusula si se desea que al borrar una ocurrencia de EDITORIAL,se borren en la relación LIBRO todos los libros editados en ella.

- Valor por Defecto ("SET DEFAULT") : Pondría el valor por defecto para el atributo Editorial en la tabla LIBRO en todas aquellas ocurrencias asociadas a una editorial borrada en la relación EDITORIAL

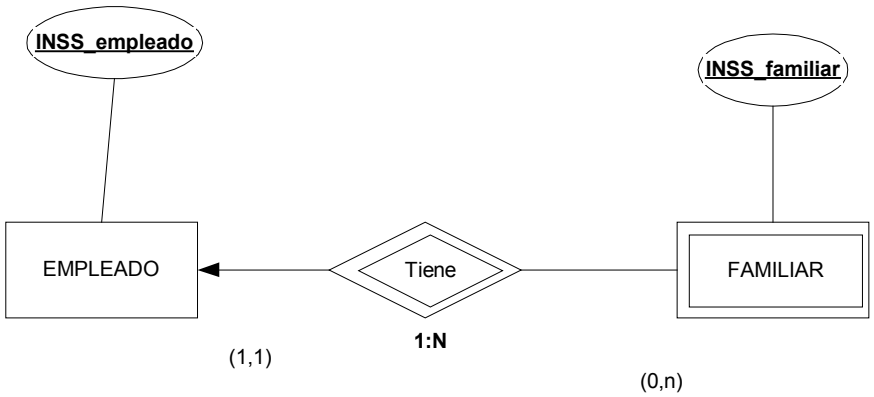
Nótese que no podría utilizarse la opción Null (nulo), debido a que la cardinalidad mínima de uno en editorial en la tabla LIBRO significa que todo libro ha de ser editado por alguna editorial, por lo que el atributo Editorial en LIBRO no admite este valor.

En el caso de modificación, lo más común es utilizar la opción de cascada, ya que por regla general, se desea que el atributo editorial en la relación LIBRO se modifique de la misma forma en que se modifica el nombre de la editorial (Nombre_e) en la relación EDITORIAL.

Por las reglas y comentarios expuestos anteriormente podemos observar que las opcines de borrados y modificación de SQL respecto a la clave ajena ayudan a mantener la integridad de la Base de Datos.

En el caso en que el conjunto relación determine una **dependencia de existencia** la transformación se realiza tal como aparece en la siguiente figura:7.80

Modelo E/R



Modelo Relacional

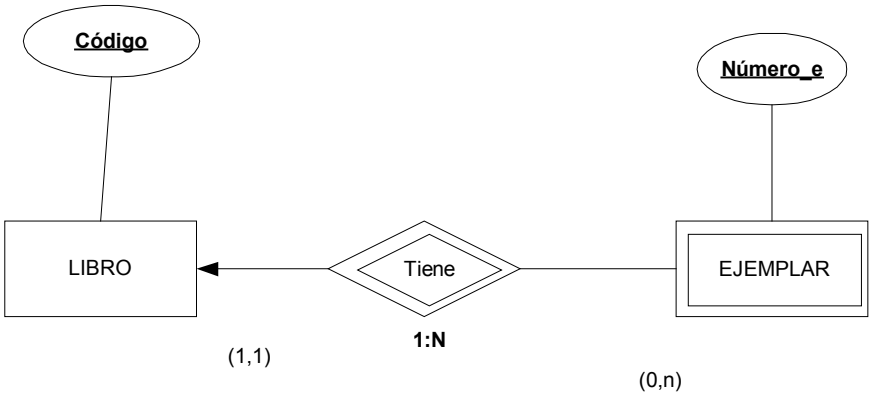
EMPLEADO(**INSS_empleado**,.....)

FAMILIAR(INSS_empleado,**INSS_familiar**,.....)

Clave ajena
NOT NULL
ON DELETE CASCADE
ON UPDATE CASCADE

Figura 7.12 Ilustración de una dependencia de existencia

Si la dependencia fuese de **identificación** como en el ejemplo que se muestra posteriormente, la única diferencia con el caso anterior es que la clave primaria de **EJEMPLAR** serìa la concatenación del atributo identificador principal de la entidad fuerte **LIBRO**, es decir **Código**, con el **Número_e** de la entidad débil **EJEMPLAR**, ya que **Número_e** por sí solo no identifica los ejemplares



Identificador de Ejemplar: **Código+ Número_e**

Esquema Relacional

LIBRO(Código,.....)

EJEMPLAR(Código, Número_e,.....)

Clave Ajena
Not Null
On Delete Cascade
On Update Cascade

Figura 7.13 Ilustración una dependencia de Identificación

Transformación al Crearse Una Nueva Relación(Tabla) (Cardinalidad N:M)
Se el conjunto relación de la figura 7.14 mostrada abajo, donde se muestran las Cadinalidades máximas y mínimas de los conjuntos entidades así como la del conjunto de relaciones.

Modelo E/R

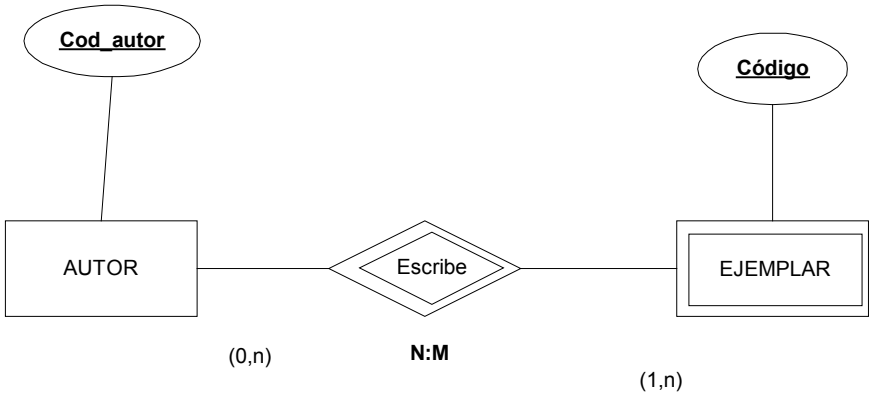


Figura 7.14 Relación con cardinalidad muchos a muchos

Modelo relacional

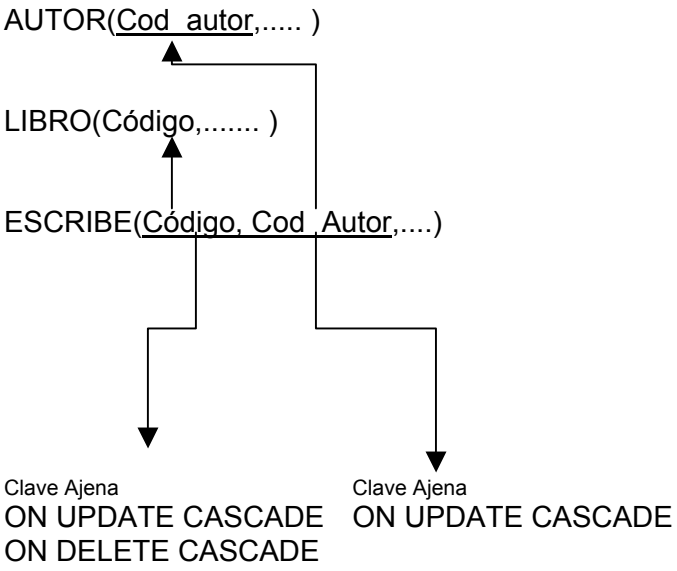


Figura 7.15 Esquena E/R (N:M) a un Esquema relacional

Las opciones de borrado y de modificación, en el caso de conjuntos de relaciones con cardinalidad N:M, generalmente son del tipo cascada, pero existen casos en los que se justifica elegir otra opción; así en el ejemplo de la figura 7.15 anterior, para la clave ajena Cod_autor que referencia al conjunto entidad AUTOR no se puesto ninguna opción de borrado (Toma por tanto la opción por defecto NO ACTION), ya que se supone que no se desea que se borre un autor si existe una ocurrencia de ESCRIBE que asocie a ese autor con algún libro.

Cuando el conjunto relación tiene un atributo diferente de los heredados de los conjuntos entidades este pasa a ser un atributo de la tabla que representa el conjunto de relaciones.

El Grafo Relacional
Una forma de sencilla de representar el esquema relacional es el denominado **Grafo relacional**. es un grafo compuesto por un conjunto de nodos cada uno de ellos con múltiples particiones donde cada nodo representa el Esquema relacional de una tabla de la base de Datos. Para cada nodo ha de aparecer como mínimo el Nombre de la relación o tabla y los atributos que la conforman indicando además las claves primarias de las relaciones estas en el grafo aparecerán en subrayado y las claves ajenas de las cuales parten arcos si están compuestas por más de un atributo señalándose mediante una flecha el atributo (y la tabla) referenciada). Las Claves alternas se indican igual que las llaves primarias pero con trozo discontinuo. En la figura 7.84 a continuación se muestra un ejemplo de un Grafo relacional para el esquema de una Biblioteca.

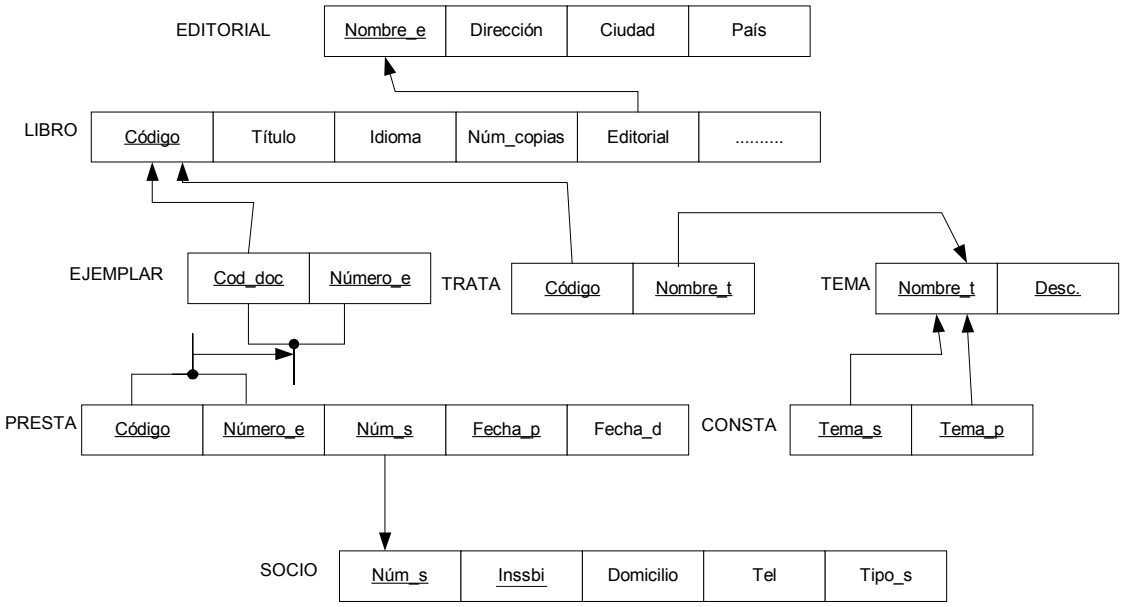


Figura 7.16 Ejemplo de un Grafo Relacional

7.3.4 Otros modelos de Datos

El modelo de Datos Orientado a Objetos :
Este es un modelo que en los últimos años ha recibido una atención creciente de parte de los diseñadores de Base de Datos, este podemos verlo como una extensión del modelo E-R fortalecido por la programación orientada a objetos

El modelo de Datos Relacional Orientado a Objetos :
Este modelo combina las características del modelo de datos orientado a objetos y las del modelo relacional.

Los Modelos de Datos Semiestructurados :
Estos se distinguen porque permiten asignar conjuntos de atributos diferentes a las filas individuales a diferencia de los modelos mencionados anteriormente en que todos los objetos básicos que componen los modelos deben de tener los mismos atributos.

7.4 Tema 3: "El Modelo Entidad – Relación"

TEMA N° 3

OBJETIVOS:

- Conocer la representación de cada uno de los objetos que componen el modelo E-R
- Saber diseñar un modelo E-R en función de una determinada información de la vida real
- Saber traducir a tablas un determinado modelo E-R
- Justificar la extensión del modelo E-R
- Saber diseñar un modelo E-R extendido utilizando especialización
- Conocer la Diferencia entre especialización y generalización
- Saber generar la herencia de atributos y relaciones al utilizar especialización o generalización
- Justificar el concepto de Agregación
- Saber utilizar la agregación en un diagrama E-R

CONTENIDO

- Conceptos básicos
  - Conjuntos entidades
  - Atributos y conceptos relacionados
  - Conjunto de relaciones
- Diagrama entidad relación
  - Componentes básicos del diagrama
- Cardinalidades de un conjunto Entidad
- Conjunto de entidades débiles
  - Concepto de discriminante
  - Clave primaria de una entidad débil
  - Representación en el diagrama E-R de una entidad débil
- La Especialización
- La Generalización
- Herencia de atributos y Relaciones
- Agregación

DURACIÓN: 12 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill
- Adoración de Miguel, Mario Piattini. Fundamentos y Modelos de Base de Datos. 2ª Edición. RAMA

El modelo de Datos Entidad Relación (E/R) está basado en una percepción del mundo real consistente en objetos básicos llamados entidades y Relaciones (Que vinculan o relacionan a las entidades). Se desarrolló con el fin de facilitar el diseño de las base de datos, mediante un gráfico que representa la estructura lógica completa de una base de datos.

7.4.1 Conceptos Básicos

Existen tres nociones básicas que utiliza el modelo E-R : Conjuntos entidades, Conjuntos de Relaciones y Atributos.

7.4.1.1 Conjuntos de Entidades

Es una colección de **Entidades** del mismo tipo.

Entidad

Conjunto de **atributos** que definen un objeto único del mundo real, en este contexto un atributo es un componente básico de un objeto del modelo E-R.

Ejemplo: El Conjunto de Entidades Estudiante en Notación de conjunto y en forma tabular

Estudiante = { [1,Juan, 22, M], [2,Roberto, 24, M], [3,María, 22, F] }
Representa a un conjunto entidad compuesto por tres entidades del mismo tipo donde cada entidad está compuesta por tres atributos que representan en el mundo real al Número de Carné, El nombre la edad y el sexo de un estudiante

El mismo conjunto de entidades en forma tabular :

1	Juan	22	M
2	Roberto	24	M
3	María	22	F

Figura 7.17 Conjunto de Entidades en Forma Tabular

En ambas representaciones se ha utilizado el código M para Masculino y el código F para femenino.

Cada objeto entidad es diferente de los demás bajo el supuesto razonable impuesto por el mundo real de que los números de carné de cada uno de los estudiantes es diferente.

7.4.1.2 Conceptos Relacionados con los atributos

Dominio del Atributo :

Es el conjunto de valores permitidos para un atributo en particular. El dominio del atributo Nombre_cliente es el conjunto de todas las cadenas de texto con una cierta longitud máxima.
Tomando en cuenta este último aspecto se puede definir el concepto de **atributo** de una manera más formal : Una función que asigna al conjunto de entidades un Dominio.

Dado que un conjunto de entidades puede tener diferentes atributos, cada entidad se puede describir como un conjunto de pares de la forma (atributo, valor), un par para cada atributo del conjunto de entidades. Por ejemplo una entidad concreta de *cliente*
Se puede describir mediante el conjunto:
{(Id_Cliente,67.789.901), (Nombre_Cliente,López),
(Calle_Cliente, Central), (Ciudad_Cliente,Granada)},

significando con esta notación que la entidad describe un cliente llamada López que tiene un número de cedula igual a 67.789.901 y que reside en Granada en la calle Central.

Atributos Simples y Compuestos

Los atributos compuestos: Son aquellos que pueden dividirse en sub-partes cada una de las cuales corresponde a otro atributo.
Atributo Simple : Es aquel que no puede sub dividirse en más atributos de forma lógica.

Ejemplo: Nombre_cliente podría estructurarse como un atributo compuesto de las siguientes sub-partes : Nombre, Primer_Apellido, Segundo_Apellido siendo cada uno de los componentes atributos simples. Se utilizan los atributos compuestos en un diagrama E-R con el fin de hacer más claro el modelo.

Los atributos Simples

Son aquellos que de forma lógica no pueden ser descompuestos en otros atributos.

Jerarquía de Atributos

La utilización reiterada de atributos compuestos genera una jerarquía de atributos en el diagrama E-R esto sucede cuando un atributo compuesto contiene a su vez atributos compuestos, tal como se muestra en el siguiente ejemplo.

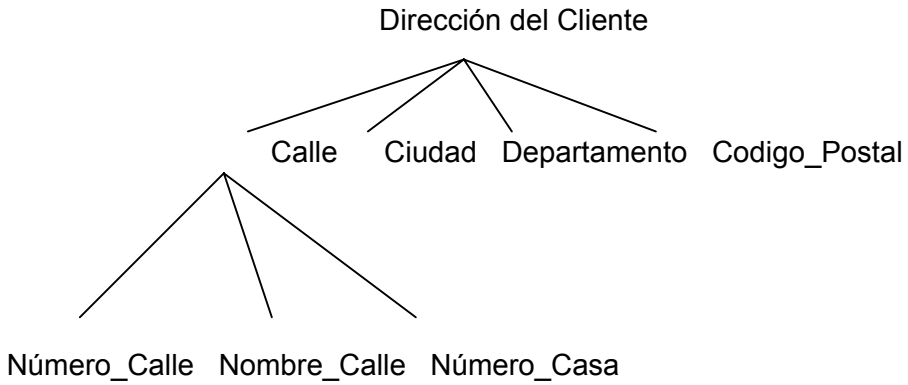


Figura 7.18 Atributos Compuestos: Dirección del Cliente y Calle

Atributos Monovalorados y Multivalorados

Los atributos que se han especificado en los ejemplos anteriores tienen todos un solo valor para una entidad concreta. Por ejemplo el atributo Número_Prestamo para una entidad específica del conjunto de entidades Préstamo referencia a un numero de préstamo, si al contrario la entidad referencia a más de un valor se dice que el atributo es Multivalorado, en este caso específico sería que Número_Préestamo para una entidad específica pudiese asumir más de un valor.

Atributos Derivados

El valor para este tipo de atributos se puede derivar de los valores de otros atributos o entidades relacionadas. Por ejemplo sea el conjunto de entidades *cliente* que tiene un atributo denominado préstamos que representa cuantos

prestamos tienen un cliente en el banco. este atributo se puede derivar contando el número de entidades préstamo asociadas con ese cliente.
Como otro ejemplo, considérese el conjunto de entidades Empleado el cual tiene un atributo edad, que indica la edad del cliente. Si el conjunto de entidades también un atributo Fecha_de_Nacimiento, se puede calcular la edad en función de la Fecha de Nacimiento, algo similar sucede con la antigüedad del empleado la cual se puede obtener a partir de la fecha de ingreso de este a la empresa.

Atributos con valores nulos
Un atributo toma un valor **nulo** cuando una entidad no tiene un valor para un atributo. El valor nulo también puede indicar "no aplicable" es decir, que el valor no existe para esa entidad. Ejemplo : No todas las personas tienen un segundo nombre. Nulo puede también significar que el valor de un atributo es desconocido. Un valor desconocido puede ser de dos tipos : Perdido (el valor existe pero no se cuenta con esa información) o desconocido (no se conoce si el valor existe realmente o no). Ejemplos de Valores Nulos: (1)El Número de Cedula de una persona este dato existe pero la persona perdió su cedula y no se acuerda
(2) una dirección que tiene entre sus componentes la calle pero la persona vive en las afueras y no se puede relacionar ninguna calle con la dirección, en este caso el dato no existe, no es aplicable.

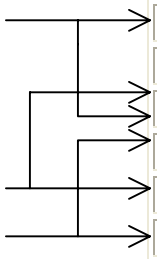
7.4.1.3 Conjunto de Relaciones

Una relación es una asociación entre diferentes entidades.
Un **Conjunto de Relaciones** es un conjunto de relaciones del mismo tipo. Formalmente es una relación matemática con n>=2 de conjuntos entidades no necesariamente diferentes. Si E1,E2,E3....,En son conjuntos de entidades, entonces un conjunto de relaciones es un sub conjunto de :

$$\{(e_1, e_2, e_3.....,\ e_n) \mid e_1 \in E_1,\ e_2 \in E_2,\ e_3 \in E_3,.....,\ e_n \in E_n\}$$

Donde $(e_1, e_2, e_3.....,\ e_n)$ es una relación.

C1	Juan	Central	León
C2	Pedro	La Rosa	Granada
C3	Carlos	Rubén Darío	León
C4	María	Colón	Managua
C5	Yolanda	Bolivar	Managua
C6	Teresa	Las Flores	Masaya
C7	Salvador	El Edén	Granada



P-1	2000
P-2	3000
P-3	4000
P-4	6000
P-5	5000
P-6	2500

Figura 7.19 Ejemplo de relaciones entre dos tablas

Considérense los dos conjuntos de entidades Cliente y Prestamo de la figura 7.9 Se define el conjunto de relaciones *Prestatario* para denotar la asociación entre clientes y los prestamos bancarios que los clientes tengan.

Note que no todos los clientes tienen un préstamo y algunos tienen más de uno

Como otro ejemplo, considérense los dos conjuntos de entidades *Préstamo* y *Sucursal.* Se puede definir el conjunto de relaciones *Sucursal-Préstamo* para denotar la asociación entre un préstamo y la sucursal en que se mantiene ese préstamo.

Concepto de Participación y Ejemplar
La asociación entre conjuntos de entidades se conoce como *participación* ; es decir los conjuntos de entidades $E_1,E_2,E_3,.....,E_n$ participan en el conjunto de relaciones R.

Un Ejemplar de relación en un esquema E-R representa que existe una asociación entre las entidades de la empresa del mundo real que se está modelando, ejemplo :

Sean los Conjuntos de entidades *Cliente y Préstamo* dadas en forma tabular arriba :
Por medio de este gráfico se indica que la entidad
Cliente(Id_Cliente, Nombre_Cliente,Calle_Cliente,Ciudad_Cliente)

C1	Juan	Central	León

está en relación con la entidad préstamo(Cod_Préstamo, Saldo_Prestamo)

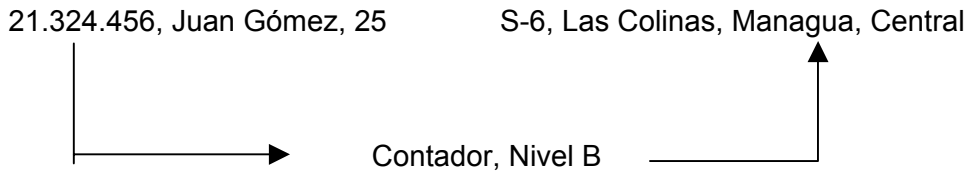
P-1	2000

La regla que vincula ambas entidades es producto de la vida real de la empresa esto quiere decir que en la realidad que el cliente con identificación C1 de nombre Juan que vive en la calle Central en la ciudad de León tiene un préstamo de 2,000 córdobas con código P-1, de igual manera se pueden interpretar el resto de vinculaciones. En este caso podemos afirmar que tanto la fila de cliente como la fila de préstamo conforman un *ejemplar* de la relación, así mismo podríamos decir que el cliente Juan con identificación C1 con el préstamo P-1 *participan* en este ejemplar de relación (Son una parte de la relación).

Concepto de papel de una entidad en la relación
La función que desempeña una entidad en una relación se llama *Papel* de la entidad (en la relación). Este concepto es útil y tiene sentido cuando la relación es entre una entidad y ella misma (una relación recursiva). Por ejemplo considérense un conjunto de entidades *empleado* que almacena información sobre los empleados de un Banco. Se puede tener un Conjunto de relaciones definidas por la regla “Trabaja Para” así tendría sentido relacionar por ejemplo a Pedro Gutiérrez, M, 25 con José López, M, 32, ambos pertenecientes al mismo conjunto entidad empleado indicando que el empleado Pedro Gutiérrez trabaja para José López, la regla contraria “es jefe de” no se puede mezclar con la “Trabaja Para” por lo tanto en este tipo de relaciones es muy importante especificar el papel de la entidad en la relación en otras palabras el orden de los componentes de la relación importa.

Atributos Descriptivos de una Relación
Una relación puede tener también atributos descriptivos que complementan la información de la relación, estos atributos complementarios tienen la particularidad que no pertenecen a ninguna de las entidades involucradas en la relación así por ejemplo a la relación entre los conjuntos entidades *Cliente y Préstamo* se puede añadir el atributo Fecha_Acceso (La Fecha más reciente en que se accedió a la cuenta).

Relaciones Ternarias
La mayoría de relaciones son como la indicada entre Cliente y Préstamo es decir relaciones binarias, sin embargo en una relación pueden intervenir más de dos conjuntos de entidades, así por ejemplo Considérense los conjunto de entidades : Empleado, Sucursal y Trabajo donde Empleado contenga los datos básicos de los trabajadores de una empresa, Sucursal contenga los datos básicos de cada una de las sucursales del banco como ciudad donde está ubicada la sucursal, su dirección etc y Trabajo indique el tipo de trabajo que realiza: Gerente, Contador, Cajero, Auditor etc, así como el nivel o Jerarquía con respecto al tipo de trabajo que desempeña: Nivel A, Nivel B. Así un ejemplar de esta relación podría ser :



La relación Empleado-Sucursal-Trabajo indica en este ejemplar que : El empleado Juan Gómez con número de identidad 21.324.456 y 25 años de edad ocupa el cargo de Contador Nivel B en la sucursal del Banco Popular “Las Colinas” ubicada en Managua en la calle Central, este tipo de relación se denomina relación Ternaria. de hecho pueden existir relaciones que involucren cuatro, cinco relaciones pero como se verá posteriormente en general lo más adecuado es realizar los diseños del modelo E-R procurando hasta donde sea posible utilizar relaciones binarias.

7.4.2 Diagrama Entidad Relación

Objetivo del diagrama : representar la lógica general de una base de datos mediante un diagrama simple.

7.4.2.1 Componentes Básicos del Diagrama

- **Rectángulos**: Estos representan conjunto de Entidades
- **Elipses**: Que representan atributos
- **Rombos**: Representan conjuntos de Relaciones
- **Líneas**: Se utilizan para unir los atributos a conjuntos de entidades, asimismo unen los conjuntos de entidades a conjuntos de relaciones
- **Elipses Dobles**: Representan atributos multivalorados
- **Elipses Discontinuas**: Denotan Atributos Derivados
- **Líneas Dobles**: Que indican participación total de una entidad en un conjunto de relaciones
- **Rectángulos Dobles**: Que representan Conjuntos de Entidades débiles

Ejemplo:

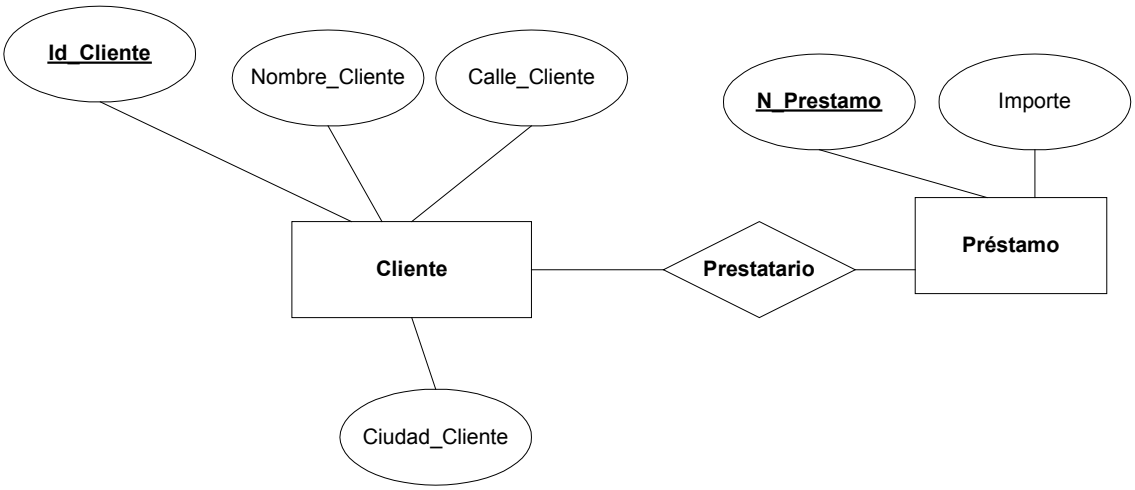


Figura 7.20 Ejemplo de un Diagrama E-R Simple

Cliente Es una Entidad, con atributos:
Id_Cliente, Nombre_Cliente,Calle_Cliente y Ciudad_Cliente
Nótese que el atributo Id_Cliente está subrayado a diferencia del resto de los atributos que no lo están, esto indica que Id_Cliente es la llave primaria del Conjunto entidad Cliente es decir que este atributo debe de ser diferente en cada una de las entidades y además debe existir en cada una de ellas (No se permiten valores nulos para este atributo). Por otra parte este atributo representa al conjunto de entidades clientes en la relación.
Préstamo es una Entidad, cuyos atributos son:N_Prestamo e Importe en este caso la llave primaria es el atributo N_Prestamo.

Atributos de la relación: Los atributos de la relación en este ejemplo son:
Id_Cliente y N_Prestamo, las cuales son las llaves primarias de las entidades involucradas en la relación, es decir en este caso las entidades heredan sus llaves primarias como atributos a la relación. Como veremos posteriormente esto no siempre sucede va a depender fundamentalmente de la *cardinalidad* de las relaciones.

7.4.3 Cardinalidades de un Conjunto Entidad

Una de las formas utilizadas para determinar la cardinalidad de una relación entre dos conjuntos de entidades A, B es determinar la cardinalidades mínimas y máximas de un conjunto de entidades respecto a una determinada relación. Los posibles valores para las cardinalidades mínimas y máximas de un conjunto entidad son: (0,1), (1,1), (0,n), (1,n), los cuales representan los siguientes conjuntos de valores: 0,1 ; 1 ; 0,1,2,3...n y 1,2,3...n respectivamente.

Veamos el siguiente ejemplo :

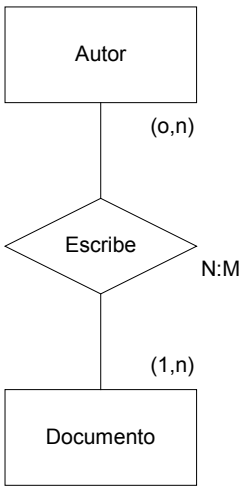


Figura 7.21 Relación de uno a muchos

lo cual significa que una entidad de **Autor** eventualmente se puede relacionar con 1,2,3,...n entidades de **Documento** y por otro lado una entidad de **Documento** se puede relacionar con 0,1,2,3....n entidades de **Autor,** es decir que cada autor Escribe por lo menos un documento y por el otro lado pueden existir Documentos sin autor o con uno o más autores. Dado que cada autor se puede relacionar con varios documentos y a su vez cada Documento se puede relacionar con varios autores se dice que el conjunto Relación tiene cardinalidad de Muchos a Muchos, lo cual se indica con la notación N:M, y en forma adicional este hacho se representa en el diagrama con líneas sin flechas como puede verse en este diagrama.

Este tipo de cardinalidad es muy común en los diagramas E-R y en este caso los atributos del conjunto relación son las llaves primarias de las entidades involucradas en la relación, se dice que estos atributos son heredados al conjunto relación.
Supóngase que los atributos de **Autor** son: { N_Cedula, Nombre, Ciudad} donde la llave primaria es N_Cedula(Número de Cedula), por otro lado los atributos de **Documento** son: {Cod_Doc, Tipo, N_Pag) donde Cod_Doc es el código del documento(La llave primaria), Tipo, el tipo de documento (Libro, Tesis, Novela etc) y N_Pag representa el número de páginas del documento y por lo discutido anteriormente los atributos del conjunto relación **Escribe** son: { N_Cedula, Cod_Doc}.
Remarcamos que en los diagramas E-R no se estila representar los atributos heredados.

Representación Tabular del ejemplo Anterior

Autor

N_Cedula	Nombre	Ciudad
213264-9	Juan	Masaya
324567-8	Pedro	Managua
224789-4	Carlos	Masaya
345678-1	Teresa	Granada
334566-3	Marvin	Managua
214365-7	Fernando	Masaya

Documento

Cod_Doc	Tipo	N_Pag
D-2	Libro	345
D-3	Folleto	155
D-4	Revista	125
D-8	Novela	250
D-15	Libro	355
D-18	Novela	215

Escribe

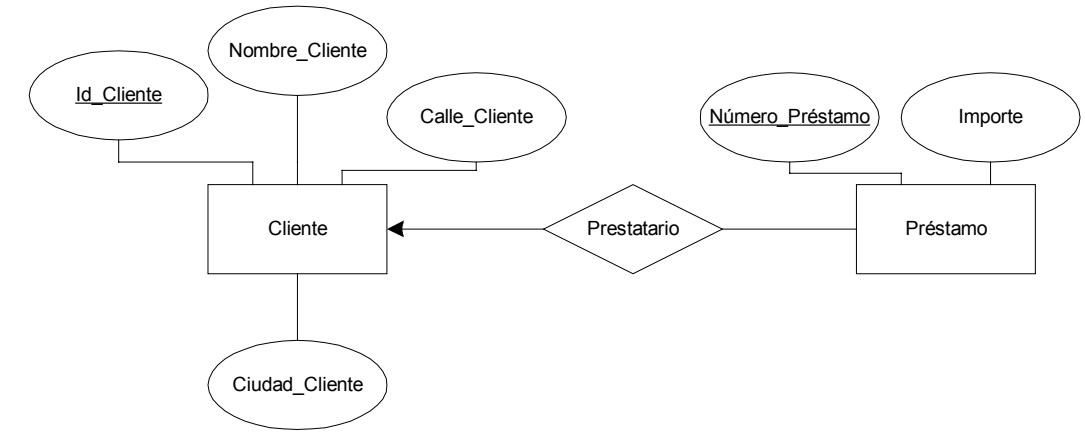
N_Cedula	Cod_Doc
324567-8	D-8
224789-4	D-2
345678-1	D-3
334566-3	D-15
214365-7	D-18

Figura 7.22 Dos conjuntos entidades y un conjunto relación

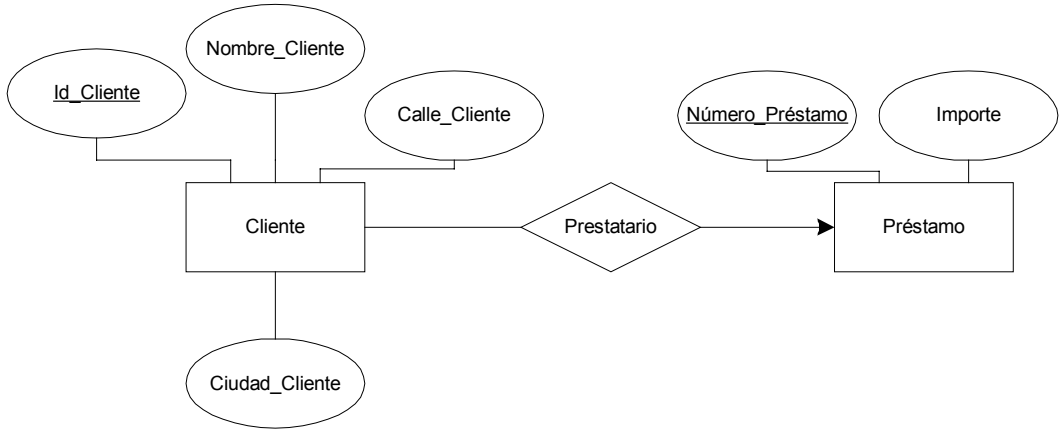
Este conjunto relación cuya regla es un reflejo de la vida real indica que la entidad representada por 324567-8 de **Autor** está relacionada con la entidad D-8 del Conjunto de entidades **Documento**, nótese además que D-4 que corresponde a un tipo revista no tiene autor aspecto que ya se había tomado en cuenta.

Otro aspecto a notar en este ejemplo es que el conjunto de relaciones Escribe solo cuenta con los atributos heredados de **Autor** y **Documento.**

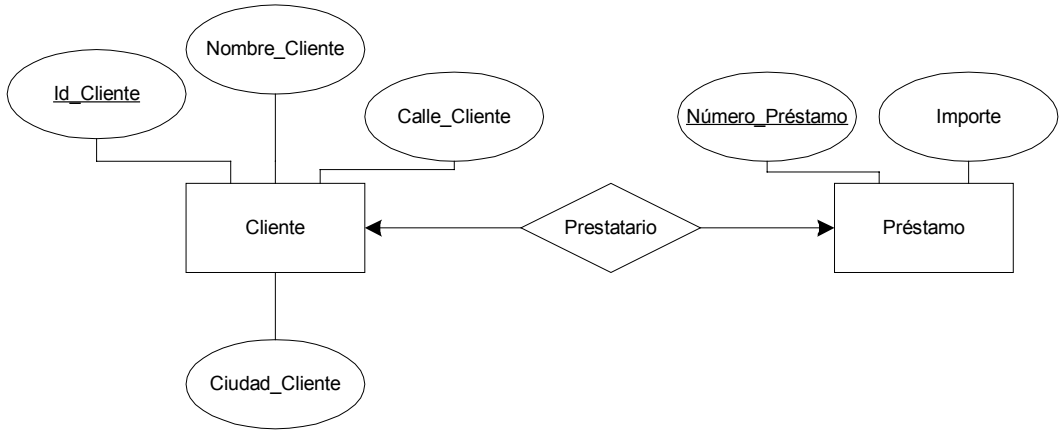
Ejemplos de Conjuntos de relaciones con diversas cardinalidades.



(a)



(b)



(c)

Figura 7.23 Relaciones (a) Uno a varios (b) Varios a uno (c) Uno a uno

Para simplificar las notaciones relacionadas con las cardinalidades de las relaciones usaremos la siguiente notación : Un conector que termine en flecha indicara que se trata de la entidad de la parte uno de la cardinalidad. Por otra parte si el conector no termina en flecha indicará que esta entidad es la parte de muchos de la cardinalidad.

Así en la figura 7.13 (a) podemos notar que la relación *prestatario* tiene una cardinalidad de uno a Muchos, es decir un cliente puede tener varios préstamos. Por otra parte la figura 7.13 (b) muestra una relación de muchos a uno entre cliente y préstamo es decir que un préstamo puede ser asumido por varios clientes y que cada cliente solo puede tener un préstamo.
Finalmente la figura 7.13(c) muestra una relación de uno a uno, los conectores de las entidades involucradas en al relación terminan en flecha, lo que indica que cada cliente solo puede tener un préstamo y que cada préstamo solo puede afectar a un cliente.
Como podemos haber observado en estos ejemplos el tipo de cardinalidad impone restricciones sobre los datos que pueda contener el conjunto de relaciones así la cardinalidad fuese de uno a muchos como el caso del inciso (a) del ejemplo son validos, los datos: (21.325.67, P-8), (21.325.67,P-12) pero no serían validos los datos (62.456.88, P-10), (72.678.78, P-10) pues cada préstamo solo puede afectar a un cliente por tanto al intentar introducir el segundo dato sería rechazado por el sistema gracias a que se diseñó adecuadamente la cardinalidad del conjunto de relaciones.

Existen muchas maneras de representar las cardinalidades de una relación en un diagrama E-R, por simplicidad se han mostrado solamente dos de las más utilizadas, en nuestro desarrollo posterior utilizaremos esta última por su simplicidad, haciendo notar si que perdemos cierta información ya que en estos gráficos E-R no se muestran las cardinalidades mínimas.

Ejemplo 4

En este ejemplo se muestra un diagrama E-R, (ver figura 7.14) en que el conjunto de relaciones cuenta con un atributo asociado, nótese que la relación es de muchos a muchos.

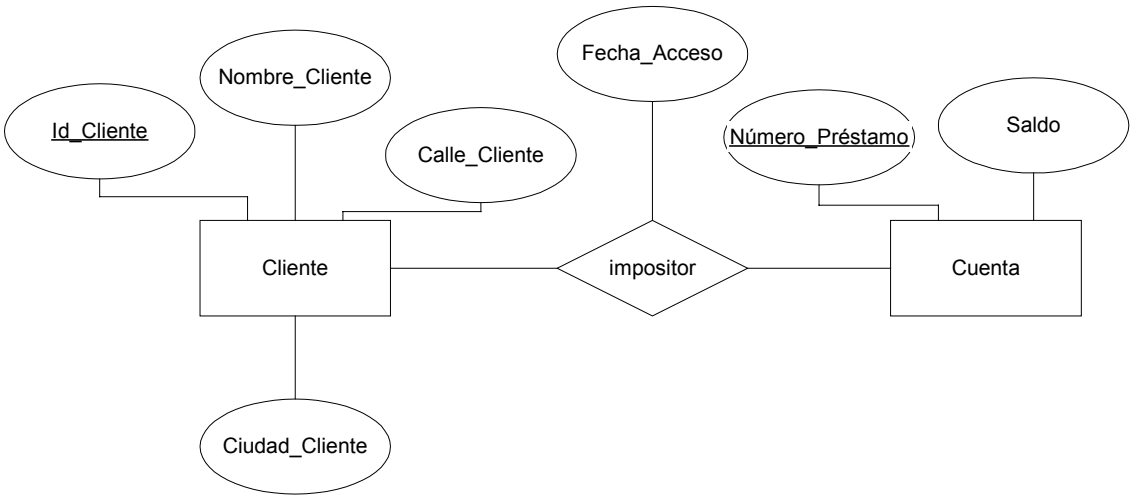


Figura 7.24 Conjunto de Relaciones con atributo Asociado

Ejemplo 6
En este ejemplo de diagrama E-R compuesto de una sola entidad, se muestra la utilización de los diferentes tipos de atributos indicados con anterioridad : Atributos, Compuestos, atributos multivalorados y atributos derivados.

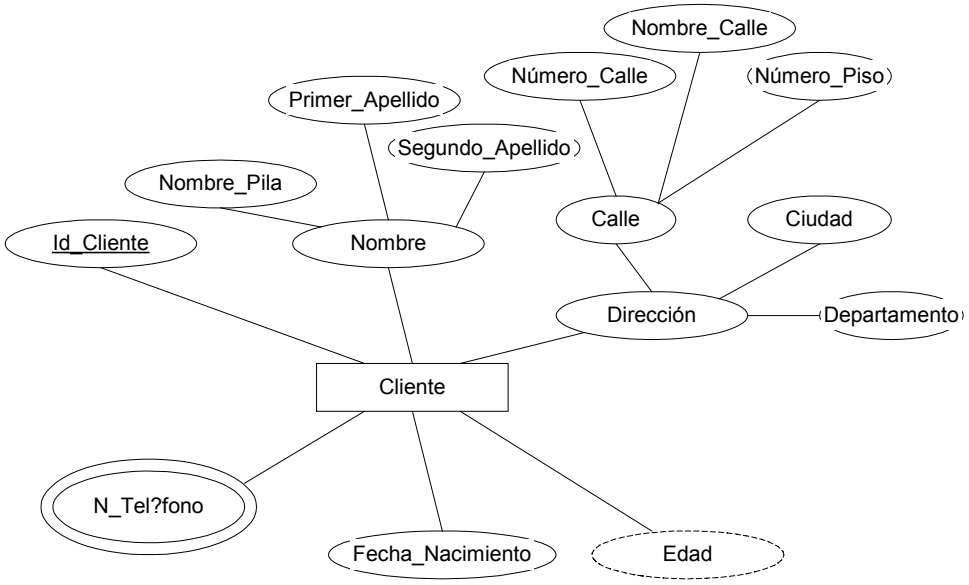


Figura 7.25 Conjunto entidad con los diferentes tipos de atributos

Llave primaria: Id_Cliente

Atributos Compuestos : Nombre y Dirección quien a su vez contiene un atributo compuesto: Calle.

Atributo Multivalorado: N_Teléfono

Atributo Calculado: edad, que se obtiene a partir del atributo: fecha_nacimiento

Papeles en las relaciones recursivas
En el siguiente ejemplo se ilustra el concepto de papel en los conjuntos de relaciones recursivos (Relación de una entidad consigo misma).

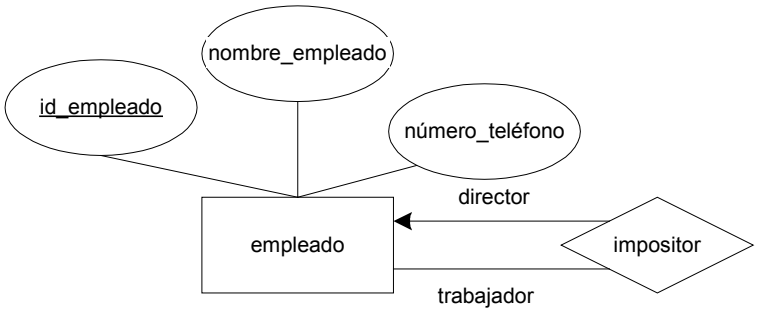


Figura 7.26 Representación de Papeles en las relaciones recursivas

Como puede verse en el diagrama de la figura 7.16, los papeles se indican mediante etiquetas en las líneas que unen el conjunto relación (El rectángulo) con el conjunto de relaciones (el rombo).
En Forma tabular:

Empleado

Id_empleado	Nombre_empleado	Número_teléfono
213264-9	Juan	311-2134
324567-8	Pedro	311-6545
224789-4	Carlos	311-7890
345678-1	Teresa	315-5678
334566-3	Marvin	315-4589
214365-7	Fernando	02-6789

Trabaja-para

Trabajador	Director
213264-9	324567-8
224789-4	345678-1
334566-3	324567-8
214365-7	345678-1

Figura 7.27 Representación Tabular de una Relación Recursiva

De los 6 empleados, 4 hacen el papel de trabajadores en trabaja-para y dos hacen el papel de directores, así por ejemplo Juan *trabaja-para* Pedro.

Representación de la participación total de un conjunto de entidades en un conjunto de relaciones.

Este hecho se representa uniendo con líneas dobles al conjunto entidad con el conjunto de relaciones, indicando con este símbolo que no puede existir una entidad que no este presente en alguna relación del conjunto de relaciones.

Un ejemplo de este hecho se muestra en la figura 7.18, indicando que todos los prestamos deben estar asignados a por lo menos uno de los clientes del banco.

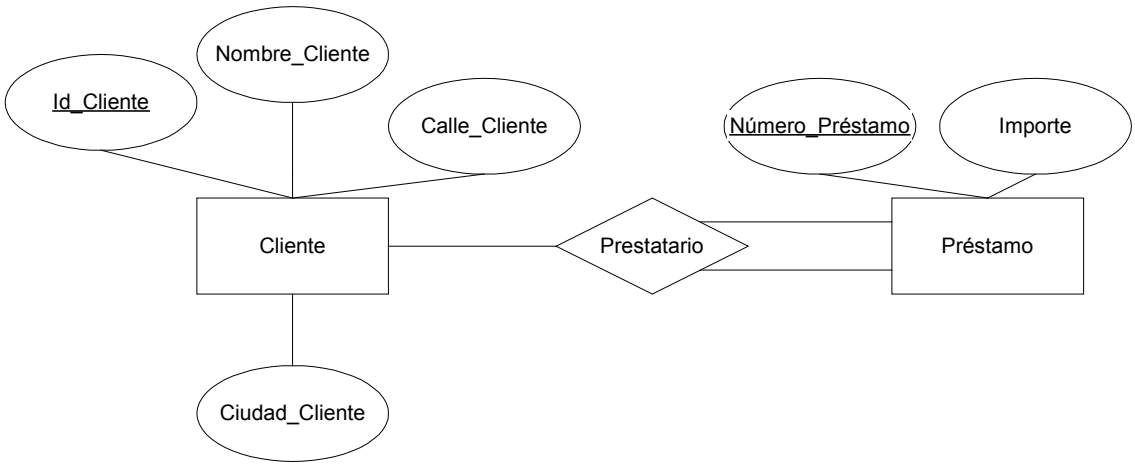


Figura 7.28 Participación total de un conjunto Entidad

Especificación de restricciones complejas en un diagrama E-R

Los diagramas E-R proporcionan una forma de indicar restricciones más complejas sobre el número de veces en que cada entidad participa en las relaciones en un conjunto de relaciones. Esto se indica en el diagrama utilizando la notación min...max donde min es la mínima cardinalidad y max es la máxima. Un valor mínimo de 1 indica una participación total del conjunto de entidades en el conjunto de relaciones. Un valor máximo de 1 indica que la entidad participa a lo sumo en una relación, mientras que un valor máximo de * in dica que no existe limite en la participación. así por ejemplo el símbolo 1..* sobre la línea que une un conjunto entidad con un conjunto relación es equivalente a una línea doble(Cada entidad debe participar por lo menos una vez en la relación).

En la figura 7.19 se muestra un diagrama E-R donde se especifican los valores mínimos y máximos de la ocurrencia de las entidades en el conjunto relación.

Según este diagrama pueden existir clientes que no tengan préstamos y cada préstamo es individual.

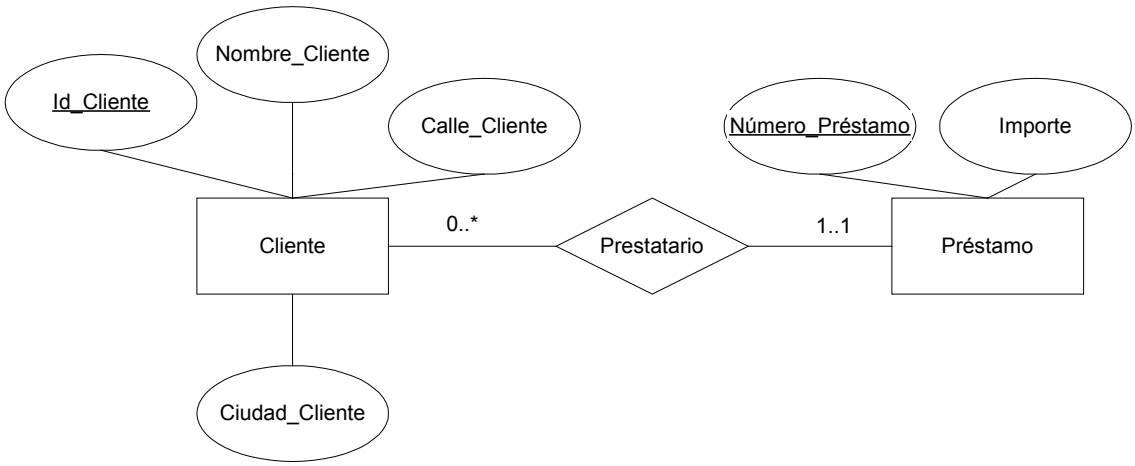


Figura 7.29 Cardinalidad máxima y mínima

7.4.4 Conjunto de Entidades Débiles

Un conjunto de entidades débiles, es aquel cuya existencia depende de otra entidad que llamaremos fuerte. Las entidades débiles no cuentan con llave primaria. Ejemplo :Supóngase que se desea llevar control de los abonos que realizan los clientes a sus prestamos, en este caso el abono en sí no tiene sentido sin el préstamo, entonces en este caso *préstamo* es la entidad fuerte y los respectivos *pagos* entidades débiles, de forma tabular:

préstamo: {Número_préstamo, importe}

Abonos: {número_pago,Fecha_pago,importe_pago}

Entidad Fuerte

P-215	5000

Entidades Débiles

P-215	1	500	1/01/03
P-215	2	600	1/02/03
P-215	3	600	1/03/03

Figura 7.30 Representación Tabular de un conjunto de entidades débiles

Como puede notarse cada entidad débil debe tener como atributo el representante de la entidad fuerte a la cual pertenece es decir la entidad fuerte hereda la llave primaria a la entidad débil.

7.4.4.1 Concepto de discriminante

Este es el atributo de la entidad débil que distingue a cada una de las entidades débiles del resto que se relacionan con una misma entidad fuerte, así en el ejemplo anterior el número_pago es un discriminante.

7.4.4.2 Clave Primaria de una entidad débil

Aunque la entidad débil no tiene clave primaria tomando en cuenta solamente sus propios atributos, se puede hablar de clave primaria de una entidad débil al heredarle el atributo respectivo de la entidad fuerte con la que se relaciona.
La llave primaria de una entidad débil esta formada por la llave primaria de la entidad fuerte de la cual depende más el discriminante. Así en el ejemplo anterior la llave primaria de Abonos es: número_préstamo+número_pago. Con esto se muestra que las claves primarias pueden estar conformadas por más de un atributo.

7.4.4.3 Representación en el diagrama E-R de una entidad débil

En el diagrama E-R un conjunto de entidades débiles se representa por 2 objetos gráficos: Un Rombo con doble borde y un rectángulo también con doble borde, unidos por línea doble para indicar que la participación del conjunto de entidades débiles en la relación es total.
Un ejemplo de la representación de un conjunto de entidades débiles se muestra a continuación, siempre referido al actual ejemplo.

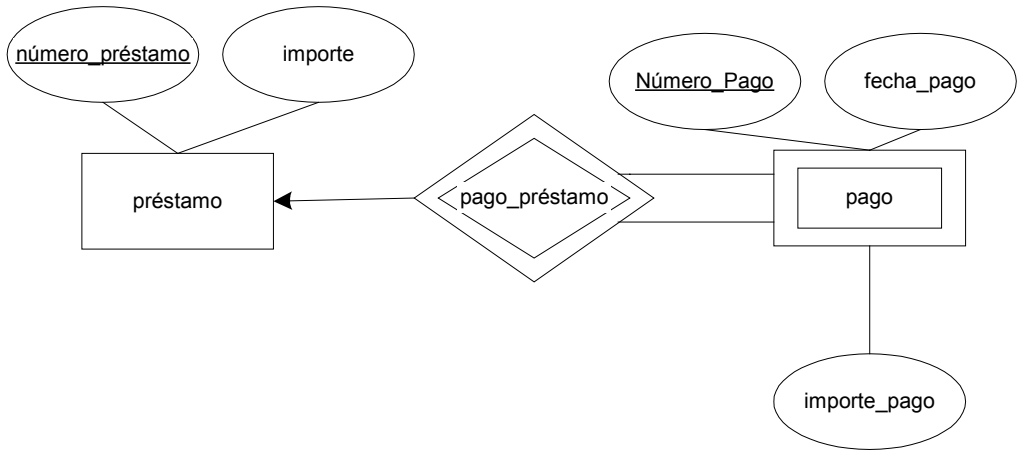


Figura 7.31 Representación en el Diagrama E/R de una entidad débil

La flecha en la parte de préstamo indica que la participación de los pagos en la relación pago-préstamo es total.

7.4.5 Características del Modelo E-R Extendido

Los conceptos básicos del modelo E-R ya estudiados pueden modelar las mayoría de las características de las bases de datos sin embargo algunos aspectos particulares se pueden modelar de mejor forma usando ciertas extensiones del modelo E-R. Estas son : La especialización, la generalización y la agregación.

7.4.5.1 La Especialización

Un conjunto de entidades puede incluir subgrupos de entidades que se diferencian de alguna forma de las otras entidades del conjunto. Por ejemplo un subconjunto de entidades en un conjunto de entidades puede tener atributos que no son compartidos por todas las entidades del conjunto de entidades. El modelo E-R proporciona una forma de representación de estos grupos de entidades distintos en algunos atributos.

Ejemplo : Considérese el conjunto de entidades *persona* con atributos : nombre, calle y ciudad. Por otra parte una persona puede clasificarse como *Cliente* o *empleado* en el se nítido de que todo Cliente y todo empleado es una persona, es decir el conjunto entidad *cliente* contará con los atributos de *persona más* otros atributos que lo van a distinguir de las demás personas como cliente algo similar podemos decir de empleado. Este proceso de la generación de conjuntos de entidades a partir de una entidad base se denomina especialización. Podemos utilizar la especialización de forma reiterada conformando así una jerarquía de conjuntos entidades así un cajero es un empleado y a su vez el empleado es una persona.

Representación de la Especialización en el diagrama E-R extendido

La relación de especialización entre dos conjuntos de entidades se representa por un triángulo invertido., veamos el siguiente ejemplo mostrado en la figura 7.22

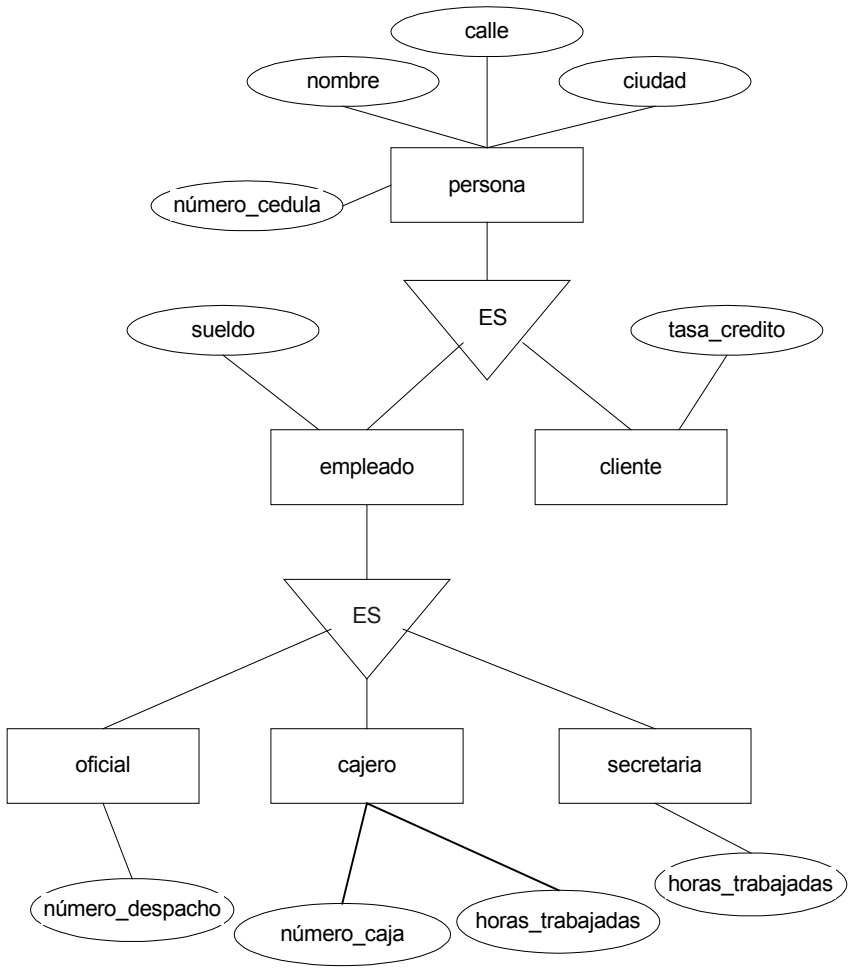


Figura 7.32 Representación de la especialización en el E/R extendido

En este gráfico se puede ver claramente los niveles de jerarquía entre cada una de las entidades.

7.4.5.2 Generalización

Con la especialización se ha realizado un proceso descendente a partir de un conjunto de entidades base, tal como se ha explicado arriba sin embargo el proceso de diseño puede ser al contrario de forma ascendente es decir que varios conjunto de entidades se sinteticen en un conjunto de entidades de nivel más alto basado en atributos comunes de un varios conjuntos de entidades iniciales. En este caso el diseñador de la Base de Datos puede haber identificado primero el conjunto de entidades *cliente* con los atributos nombre, calle, ciudad e id_cliente y haber identificado también al conjunto de entidades *empleado* con los atributos nombre, calle, ciudad, id_empleado y sueldo, en este caso la generalización consistiría en generar el conjunto de entidades *persona* con los atributos comunes nombre, calle y ciudad.

No se hará distinción alguna en el modelo E-R extendido si se llegó a el a través de una Especialización o de una Generalización.

7.4.5.3 Herencia de Atributos y relaciones

Una propiedad muy importante que justifica la extensión de los diagramas E-R en cuanto a la Especialización y la generalización es los conjuntos de entidades de más alto nivel heredan sus atributos a los de más bajo nivel.
De forma similar un conjunto de entidades de nivel más bajo también hereda la participación en los conjuntos de relaciones en los que su entidad de nivel más alto participa. Así por ejemplo y siempre referidos a la figura 7.22 *cliente* y *empleado* heredan los atributos de persona., a su vez *oficial, cajero* y *secretaria* heredan los atributos de empleado. Otro aspecto de suma importancia sobre esta extensión de los diagramas E-R es que los conjunto de entidades de nivel más bajo heredan también los conjuntos de relaciones así supongamos que empleado tiene la relación recursiva es empleado_de esta se trasmite a los conjuntos de entidades: *oficial, cajero y secretaria.*

7.4.5.4 Agregación

Una limitante del modelo E-R es que no permite efectuar relaciones entre relaciones si no que solamente relaciones entre entidades, veamos el siguiente ejemplo que obedece a los siguientes predicados semánticos :
Cada Empleado trabaja en varios proyectos
Cada empleado usa determinada maquinaria en función del proyecto en que trabaje.
Distinguimos claramente tres conjuntos entidades fuertes : Empleado, Proyecto y maquinaria y dos relaciones : La relación Trabaja, entre Empleado y proyecto y una relación ternaria : usa entre las entidades empleado, proyecto y maquinaria.
Instancia de la base de Datos : Conjuntos de Entidades

Empleado

Nombre
Emp1
Emp2
Emp3
Emp4

Proyecto

Id_P
P1
P2
P3

Maquinaria

Id_M
Emp1
Emp2
Emp3
Emp4

Figura 7.33 Tablas Empleado,Proyecto y Maquinaria

Conjuntos de Relaciones :

Nombre	Id_P	N_Horas
Emp1	P1	4
Emp1	P2	4
Emp2	P3	8
Emp3	P1	2
Emp3	P2	6
Emp4	P1	4
Emp4	P3	4

Figura 7.34 conjunto de Relaciones trabaja

Nombre	Id_P	Id_M
Emp1	P1	M3
Emp1	P2	M2
Emp2	P3	M1
Emp2	P3	M4
Emp3	P1	M1
Emp3	P1	M2
Emp3	P2	M4
Emp4	P1	M1
Emp4	P1	M2
Emp4	P3	M1
Emp4	P3	M4

Figura 7.35 Conjunto de Relaciones USA (Ternaria)

La figura 7.36 muestra la problemática generada (una relación redundante) al diseñar el respectivo diagrama E-R

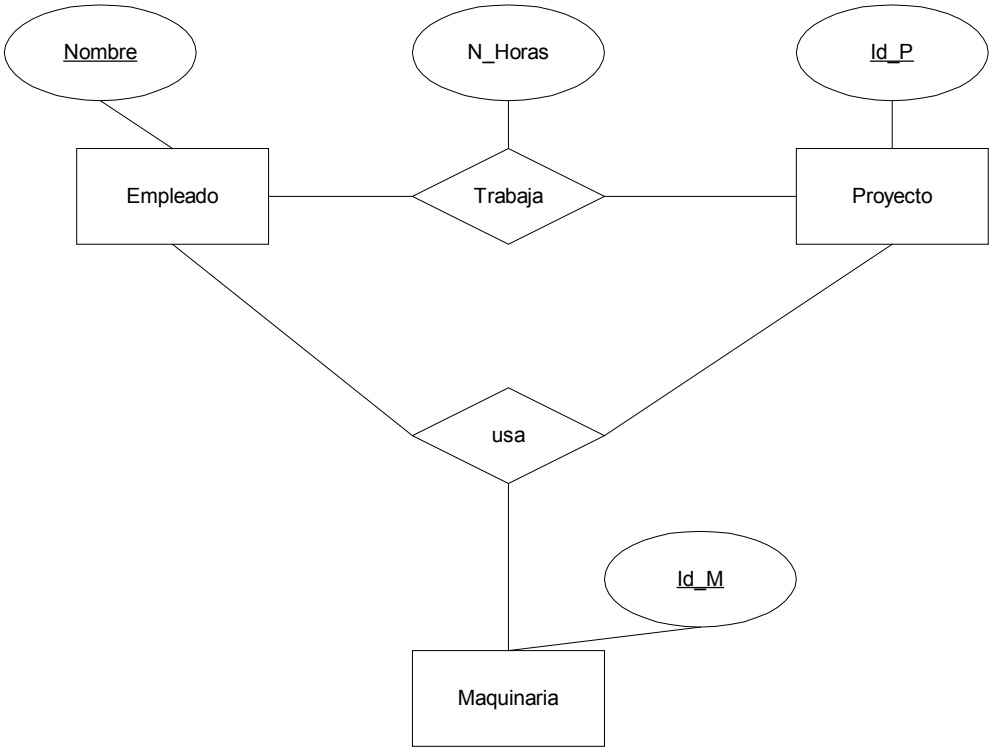


Figura 7.36 E/R sin utilizar Agregación

Como se ha podido observar en el diagrama E-R, Empleado y Proyecto se relacionan 2 veces en forma innecesaria, una manera de corregir esta problemática es utilizando agregación que no es más que considerar a un conjunto relación como un conjunto identidad fuerte, este es el caso de *Trabaja* con lo cual el diagrama E-R quedaría de la siguiente manera :

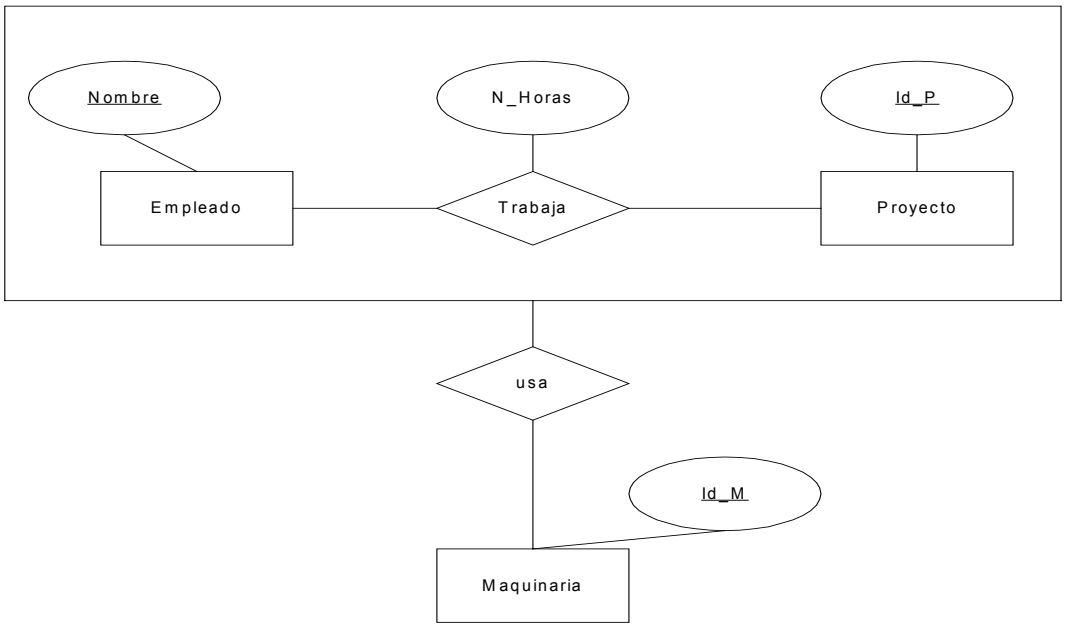


Figura 7.37 E/R Utilizando Agregación

El uso de la agregación se indica por medio del rectángulo que contiene a la relación trabaja y sus respectivas entidades, indicando este que la relación trabaja es una entidad fuerte en la relación binaria *usa.* El sistema ahora cuenta con 4 conjuntos entidades y un conjunto relación. Las tablas mostradas arriba quedan iguales con la diferencia de que el conjunto relación trabaja es una entidad fuerte con clave primaria Nombre + Id_P, con esta extensión de los diagramas E-R aplicada al ejemplo anterior se ha logrado construir un diagrama sin relaciones duplicadas.

Analizemos una instancia de esta base de datos

Empleado

Nombre
Juan
Pedro
María
Carlos
Teresa

Proyecto

Id_P
Pr_1
Pr_3
Pr_4
Pr_5
Pr_6

Trabaja

Nombre	Id_P	N_Horas
Juan	Pr1	10
Juan	Pr2	15
Pedro	Pr1	12
María	Pr3	8
María	Pr4	10
Carlos	Pr5	12
Carlos	Pr1	15
Teresa	Pr6	18

Maquinaria

Id_M
M-1
M-2
M-3
M-4
M-5

Usa

Juan	Pr1	M-1
Juan	Pr2	M-2
Pedro	Pr1	M-1
María	Pr3	M-2
María	Pr4	M-3
Carlos	Pr5	M-3
Carlos	Pr1	M-4
Teresa	Pr6	M-5

Figura 7.38 Instancia de la Base de Datos Utilizando agregación

Como se explicó anteriormente nótese que Trabaja es una entidad fuerte que se relaciona con la entidad Maquinaria, originando el conjunto relación Usa.

7.5 Tema 4: ”El Modelo Relacional”

TEMA N° 4

OBJETIVOS:

- Saber el concepto matemático de Relación
- Saber aplicar el concepto matemático de relación al modelo relacional
- Conocer los objetos del modelo relacional y la notación utilizada
- Saber definir el esquema o estructura de una relación
- Conocer las diferentes operaciones del álgebra relacional
- Saber aplicar las operaciones del álgebra relacional para resolver consultas, utilizando como herramienta de ayuda los diagramas E-R
- Saber cuando aplicar una consulta de proyección generalizada y las funciones a utilizar
- Saber distinguir entre una consulta con y sin funciones de agregación
- Conocer el concepto de valor nulo en una relación
- Saber aplicar los diferentes tipos de reuniones externas
- Conocer y saber aplicar las diferentes operaciones del álgebra relacional relacionadas con la modificación de la base de Datos
- Conocer las diferencias entre un lenguaje procedimental y no procedimental
- Obtener Consultas de una base de datos por medio del lenguaje Cálculo relacional de Tuplas
- Obtener Consultas de una base de datos por medio del lenguaje Cálculo relacional de Dominios
- Comparar el cálculo relacional de tuplas y el de Dominio con el álgebra relacional
- Conocer la Importancia de la Normalización en el diseño de las Bases de Datos
- Saber Distinguir cuando una relación tiene la problemática de la Primera Forma Normal y como resolverla
- Saber Distinguir cuando una relación tiene la problemática de la Segunda Forma Normal y como resolverla
- Saber Distinguir problemáticas del tipo 3FN y resolverlas
- Saber Distinguir cuando una relación de la base de Datos viola la forma Normal de Boyce-Codd
- Saber Normalizar una Tabla utilizando la FNBC
- Conocer los diferentes procesos de descomposición que son realizados al efectuar una determinada normalización en una relación
- Saber Distinguir cuando una relación tiene la problemática de la Cuarta Forma Normal y como resolverla
- Saber Distinguir cuando una relación tiene la problemática de la Cuarta Forma Normal y como resolverla

CONTENIDO

- El modelo Relacional
  - Repaso de algunas definiciones matemáticas
    - Concepto de producto cartesiano
    - Concepto de relación
    - Representación tabular de las Relaciones
  - Estructura de las bases de datos relacionales
    - Componentes del modelo y notación
    - Esquema o estructura de una relación
  - Lenguajes de consulta
    - El Lenguaje procedimental: Álgebra relacional
      - Operaciones fundamentales
      - Combinación de las operaciones básicas
      - Otras operaciones derivadas de las básicas
  - El álgebra Relacional y el Modelo E-R
- Operaciones del álgebra relacional extendida
  - La proyección Generalizada
  - Funciones de Agregación
- Reunión Externa

- Modificación de la base de Datos
  - La Operación inserción
- El Lenguaje Cálculo Relacional de Tuplas
  - El Equivalente de la proyección y la Selección en el Cálculo relacional de Tuplas
  - El Equivalente del Producto Cartesiano y El Producto Natural en el Cálculo relacional de Tuplas
- El Lenguaje Cálculo Relacional de Dominios
  - Concepto de Variable de Dominio
  - Representación de una relación en este lenguaje
  - Las Operaciones de Selección y Proyección en el Cálculo relacional de Dominios
  - El Producto Cartesiano y El Producto natural en el cálculo relacional de Dominios
- Teoría de la Normalización
  - Introducción
    - Formas Normales
      - Primera Forma Normal
      - Segunda Forma Normal
      - Tercera Forma Normal
  - La Forma Normal de Boyce-Codd
  - El Proceso de Descomposición de las Formas Normales
    - Descomposición por la aplicación de la Primera Forma Normal
    - Descomposición por la aplicación de la Segunda Forma Normal
    - Descomposición por la aplicación de la Tercera Forma Normal
    - Otros tipos de Dependencias
  - La Cuarta forma Normal
  - La Quinta forma Normal

DURACIÓN: 16 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill
- Adoración de Miguel, Mario Piattini. Fundamentos y Modelos de Base de Datos. 2ª Edición. RAMA

Este modelo salió a luz en 1970(Codd), sus antecesores el modelo de red y Jerárquico están más ligados a la implementación física que el relacional

(Utilización) de punteros, el modelo en Red está basado en grafos y el Jerárquico en árboles.

7.5.1 Repaso de Algunas Definiciones Matemáticas

7.5.1.1 Concepto de Producto Cartesiano

Sean A_1, A_2,..... A_n conjuntos cualquiera entonces el producto cartesiano de A_1, A_2,..... A_n se representa por $A_1 \times A_2 \times A_3 \times ..... \times A_n$ y se define de la siguiente manera :

$A_1 \times A_2 \times A_3 \times ..... \times A_n = \{(a_1, a_2, a_3, a_4 ..... a_n) \mid a_1 \epsilon A_1, a_2 \epsilon A_2, ... a_n \epsilon A_n \}$

Ejemplos:

Si A= {1,2} y B={a,b}, entonces A x B= {(1,a),(1,b),(2,a),(2,b) }

Sea A= {0,1},B= {0,1}, C={0,1}, entonces

A x B x C={ (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1) }

7.5.1.2 Concepto de Relación

Sean A_1, A_2,..... A_n Conjuntos cualquiera, entonces R es una relación si y solo si $R \subseteq A_1 x A_2 ..... x A_n$.

Ejemplo 9.3
El Conjunto D= { (0,0,1), (1,0,1), (1,1,1) } es una relación con respecto al conjunto A x B x C del ejemplo anterior

Representación tabular de las Relaciones
En lugar de utilizar la tradicional representación de las relaciones utilizaremos una forma alterna de representarlas que se ajusta mejor al contexto de las bases de Datos, esta es la representación de las relaciones en forma de Tablas, veamos un ejemplo

Ejemplo:
Sea A= { (0,0,1), (1,0,1), (1,1,1) } y B={(a,b,c), (d,c,b), (e,d,a)}
Encontrar A x B en notación de Conjunto y representarlo también en forma de tabla.

A x B= { (0,0,1,a,b,c), (0,0,1,d,c,b), (0,0,1,e,d,a), (0,0,1,a,b,c), (1,0,1,d,c,b), (1,0,1,e,d,a), (1,1,1,a,b,c), (1,1,1,d,c,b), (0,0,1,e,d,a) }

Utilizando la notación de tablas :

En este Contexto podemos ver a los conjuntos A, B como tablas

A

0		0	a	1	b	c
1		0	d	1	c	b
1		1	e	1	d	a

B

0	0	1	a	b	c
0	0	1	d	c	b
0	0	1	e	d	a
1	0	1	a	b	c
1	0	1	d	c	b
1	0	1	e	d	a
1	1	1	a	b	c
1	1	1	d	c	b
1	1	1	e	d	a

Figura 7.39 Producto Cartesiano en Forma Tabular

7.5.2 Estructura de las Bases de Datos Relaciónales

Una base de Datos relacional consiste en una colección de tablas cada una de las cuales tienen un nombre único. cada fila de una tabla representa una relación entre un conjunto de valores, dado que cada una de las tablas es una colección de dichas relaciones podemos ver a una tabla como un conjunto relación.

7.5.2.1 Componentes del Modelo y Notación

En este modelo llamaremos relación a cualquier tabla, denominaremos tupla a cualquier fila de una tabla y el conjunto de valores de las columnas se denominarán atributos.
Una variable t puede representar una tupla de una tabla o relación de modo que t ϵ r indica que la tupla representada por la variable t está en la relación r.
Ahora bien si u es el nombre de un atributo, t[u] representa un determinado valor para ese atributo.

Sea la tabla Estudiante:

Nombre	Edad	Carrera
Juan	22	Matemáticas
Pedro	32	Estadística
María	24	Computación

Figura 7.40 Tabla estudiante

Entonces si t es una variable de tupla para esta relación o tabla tienen sentido las expresiones, t[Nombre] = Pedro, t[edad] = 32, t[Carrera] = Estadística

7.5.2.2 Esquema o Estructura de una Relación

Es una lista compuesta de los nombres de sus atributos y sus correspondientes dominios.
Ejemplo :
Esquema o estructura de la relación Suc_Cliente :
{Nombre_Sucursal: Cadena, Número_Cuenta: Entero,Nombre_Cliente: Cadena, Saldo: Entero }

7.5.3 Lenguajes de consultas

Son aquellos lenguajes en que el usuario interroga o solicita información a la base de Datos. En general estos lenguajes son normalmente de más alto nivel que los lenguajes estándar de programación o de uso general. Los lenguajes de consulta se clasifican en procedimentales y no procedimentales. En un lenguaje procedimental, el usuario indica al sistema que realice una secuencia de operaciones en la base de datos con el fin de obtener el resultado deseado. En un lenguaje no procedimental, el usuario describe la información deseada sin un procedimiento específico para obtener dicha información. En este modelo relacional se cuentan con lenguajes de los dos tipos así el álgebra Relacional es Procedimental mientras que el Cálculo relacional de Tuplas y el Cálculo relacional de Dominios son no procedimentales.

7.5.3.1 El Lenguaje Procedimental Álgebra Relacional

Este lenguaje consta de las siguientes operaciones básicas: La operación selección, representada por el Símbolo σ, la operación proyección representada

por el símbolo **Π**, la operación producto Cartesiano representada por el símbolo x, la operación Unión (**U**), la operación Intersección (**∩**)
Donde estas operaciones se definen de la siguiente manera:

La Operación Selección
Sean R una relación o tabla y sea cond una condición lógica sobre los atributos de R que determina un sub conjunto de R, entonces esta nueva relación o tabla se representa por: σ_{cond} (R).

La Operación Proyección
Por otro lado supongamos que el conjunto de los atributos de R es
T={ Atr_1, Atr_2, Atr_3,.... Atr_n } entonces π_S (R) tal que s $\subset$ T, se define como una nueva relación con todas las tuplas de R, restringida al sub conjunto s de T.

La Operación Producto Cartesiano
El producto cartesiano A x B donde A y B son relaciones, genera una nueva relación con las siguientes particularidades :
Conjunto de Atributos de A x B =
Conjunto de atributos de A ∪ Conjunto de atributos de B
En cuanto al número de tuplas o filas de A x B, estas se obtienen combinando cada fila de con todas las de B de modo que si A tiene N filas y B tiene M filas el número de filas de A x B es N x M.

La Operación Unión

Sean A, B dos relaciones entonces A **U** B es una nueva relación que consiste en la unión de conjunto de ambas relaciones, en otras palabras si una tupla t está en ambas relaciones solo aparecerá una sola vez en la relación A **U** B.
La operación ~
Sean A y B dos relaciones o tablas entonces A ~ B es la relación que resulta al eliminar de A todas las tuplas que pertenecen a la relación B

Ejemplo A:
Sean las tablas Estudiante1 y Asignatura1 indicadas abajo, obtener las relaciones resultantes de las siguientes expresiones del Álgebra Relacional :

1. $\sigma_{Becado=”si”}$ (Estudiante1)

2. $\Pi_{Nombre,Año}$ (Estudiante1)

3. Estudiante1 x Asignatura1

Estudiante1 Asignatura1

Nombre	Becado	Año
Est1	Si	2
Est2	No	4
Est3	Si	3
Est4	Si	2
Est5	No	2

NombreA	Sem
As1	I
As2	II
As3	II

Figura 7.41 Tablas Estudiante1 y Asignatura1

Solución

$\sigma_{Becado="si"}$ (Estudiante1)

Nombre	Becado	Año
Est1	Si	2
Est3	Si	3
Est4	Si	2

$\pi_{Nombre,Año}$ (Estudiante1)

Nombre	Año
Est1	2
Est2	4
Est3	3
Est4	2
Est5	2

Estudiante1 x Asignatura1

Nombre	Becado	Año	NombreA	Sem
Est1	Si	2	As1	I
Est1	Si	2	As2	II
Est1	Si	2	As3	II
Est2	No	4	As1	I
Est2	No	4	As2	II
Est2	No	4	As3	II
Est3	Si	3	As1	I
Est3	Si	3	As2	II
Est3	Si	3	As3	II
Est4	Si	2	As1	I
Est4	Si	2	As2	II
Est4	Si	2	As3	II
Est5	No	2	As1	I
Est5	No	2	As2	II
Est5	No	2	As3	II

Ejemplo B :
Sean las tablas o relaciones Estudiante_Ciencias y Estudiante_Derecho, en la primera se almacenan datos relacionados con los estudiantes activos de la facultad de Ciencias, mientras que en la segunda relación se almacenan datos sobre los estudiantes de la facultad de Derecho. Se desea obtener una tabla o relación que muestre: El Nombre de los estudiantes de sexo masculino de ambas Facultades procedentes de Managua.

Estudiante_Ciencias

Carnet	Nombre	Sexo	Procedencia
C1	Juan	M	Managua
C5	Pedro	M	Masaya
C8	Carlos	M	Managua
C6	Teresa	F	Managua
C4	María	F	Masaya

Estudiante_Derecho

Carnet	**Nombre**	**Sexo**	**Procedencia**
C11	Juana	F	Masaya
C15	Rolando	M	Masaya
C18	Azucena	F	Managua
C16	Javier	M	Managua
C14	Luisa	F	Masaya

$$\Pi_{\text{Estudiante\_Ciencias.Nombre}}\ \cup\ \sigma_{\text{Estudiante\_Ciencias.sexo=”M” } \wedge\ \text{Estudiante\_Ciencias.Procedencia=”Managua”}}(\text{Estudiante\_Ciencias})$$

$$\cup$$

$$\Pi_{\text{Estudiante\_Derecho.Nombre}}\sigma_{\text{Estudiante\_Derecho.sexo=”M”} \wedge\ \text{Estudiante\_Derecho.Procedencia=”Managua”}}\ (\text{Estudiante\_Derecho})$$

Tabla Resultante

Nombre
Juan
Carlos
Javier

Figura 7.43 Resultado de las Operaciones del ejemplo B

La Operación Renombramiento
La necesidad de introducir esta operación en el álgebra relacional es debido a que las relaciones resultantes de expresiones de esta álgebra no cuentan con un

nombre, por lo que existen situaciones en que estos resultados deben de utilizarse más de una vez en una determinada expresión lo que originaría expresiones sin sentido.

Forma General de la operación renombramiento : $\rho_{x(A1,A2,..An)}$ (E) donde E es la expresión del álgebra relacional que se renombrará como x y A1,A2,..An son los nuevos nombres que tendrán los atributos originales de la expresión x. El renombramiento de los atributos al renombrar una relación es opcional.
Ejemplo
Buscar el máximo saldo de las cuentas de un determinado banco, a partir de la relación cuenta con atributos Nombre_Cliente y Saldo.

Solución
Si intentamos resolver la consulta sin utilizar la operación renombramiento llegaríamos a la siguiente problemática :
Como paso intermedio se requiere obtener el producto cartesiano de la relación cuenta con ella misma, esto nos generaría atributos con el mismo nombre en la relación resultante y no se pueden distinguir por el nombre de la relación original ya que en ambos casos es *cuenta.*
Utilizando renombramiento: en el producto cartesiano de cuenta x cuenta se obtendrá la siguiente expresión.
cuenta x (ρ_d (cuenta)) lo que permitirá distinguir todos los atributos involucrados en el producto cartesiano.
Expresión resultante de la consulta solicitada

$\pi_{cuenta.saldo}$ ~ $\pi_{cuenta.saldo}$ ($\sigma_{cuenta.saldo<d.saldo}$ (cuenta x (ρ_d (cuenta)))

donde: $\sigma_{cuenta.saldo<d.saldo}$ (cuenta x (ρ_d (cuenta)) genera todas las tuplas en que el saldo (de cuenta.saldo) es menor a los saldos restantes, por lo que en este conjunto de tuplas no aparece el mayor de cuenta.saldo. Por lo tanto al efectuar la operación diferencia de la expresión resultante se obtendrá el resultado deseado.

Combinación de las operaciones Básicas del Álgebra Relacional
Dado que cada una de las operaciones básicas del Álgebra Relacional generan una relación es por tanto permisible aplicarlas en cualquier orden y las veces que sean necesarias para obtener el resultado deseado, así por ejemplo podemos realizar un producto cartesiano y a ese resultado aplicarle la operación de Selección y posteriormente la operación de Proyección. es precisamente este hecho el que le da la potencia requerida a este lenguaje tal como se pudo observar en el ejemplo anterior.

Ejemplo:

Obtener la relación resultante de la siguiente expresión del Álgebra Relacional utilizando las tablas estudiante1 y Asignatura1 del ejemplo anterior.

Expresión:

$$\pi_{Estudiante1.Nombre}\ \sigma_{Asignatura1.NombreA=\text{"As1"}}\ (\text{Estudiante1 x Asignatura1})$$

Solución

Nombre	Becado	Año	NombreA	Sem
Est1	Si	2	As1	I
Est2	No	4	As1	I
Est3	Si	3	As1	I
Est4	Si	2	As1	I
Est5	No	2	As1	I

Figura 7.44 Tabla resultado al combinar las operaciones de: Selección y Producto Cartesiano

Nombre
Est1
Est2
Est3
Est4
Est5

Figura 7.45 Tabla Resultado al añadir la operación Proyección

Otras Operaciones del Álgebra Relacional derivadas de las Básicas.
La operación Intersección, el Producto Natural y la división

La operación Intersección
Asimismo sean A, B dos relaciones entonces A ∩ B es una nueva relación que contiene las tuplas comunes a las relaciones A y B.

El Producto Natural
Este se denota por el símbolo lxl y se deriva de las operaciones de Proyección, Selección y Producto cartesiano, de forma tal que si A y B son dos relaciones que tienen en común el atributo t,

A lxl B = $\Pi_{A(R) \cup B(S)} \sigma_{A.t=B.t}$ (A x B), donde A(R) significa que los atributos de A están contenidos en el conjunto R y similarmente los de B en el conjunto S. en otras palabras, solo se efectuará el producto cartesiano con aquellas tuplas en que los valores del atributo en común sean iguales y el atributo repetido en ambas relaciones aparece solamente una vez en A lxl B.

Ejemplo
Sean las tablas A y B indicadas abajo, obtener la relación A lxl B

At1	**At2**	At3	At4
a	b	c	e
x	y	z	u
t	r	s	u

Bt1	Bt2	**At2**
1	2	b
1	3	y
2	4	b
3	2	y

Atributo Común

Solución

At1	At2	At3	At4	Bt1	Bt2
a	b	c	e	1	2
a	b	c	e	2	4
x	y	z	u	1	3
x	y	z	u	3	2

Figura 7.46 Producto natural de las relaciones A y B

Se puede modificar ligeramente la definición anterior para poder efectuar la operación producto natural con respecto a atributos con diferentes nombres en las relaciones A y B. En este caso se define:

A $lx_{r\theta s}l$ B $= \Pi_{A(R) \cup B(S)} \sigma_{A.r=B.s}$ (A x B) donde rθs indica que el producto Natural se efectuará respecto al atributo r en la relación A y al atributo s en la relación B.

Ejemplo :

Sean las siguientes relaciones A, B obténgase A $lx_{At1\theta Bt4}l$ B

A

At1	At2	At3
a	b	c
x	y	z
t	r	s

B

Bt1	Bt2	Bt3	Bt4
t	r	s	a
x	y	z	x
a	b	c	x

Solución

At1	At2	At3	At4	Bt1	Bt2	Bt3	Bt4
a	b	c	e	t	r	s	a
x	y	z	u	x	y	z	x
x	y	z	u	a	b	c	x

Figura 7.47 Producto Natural explicito entre las relaciones A y B

La Operación División
Sean r(R) y s(S) relaciones de modo que S⊆ R, entonces la relación r ÷ s es una relación de esquema R – S. Una tupla t Є r ÷ s ssi para cada tupla t_s Є s,existe una tupla t_r que satisface las condiciones siguientes: $t_r[S] = t_s[S]$
∧ $t_r[R-S] = t[R-S]$.
Ejemplo 9.11
Sean las relaciones r y s Definidas de la siguiente manera, obtener la relación
r ÷ s

r

A	B	C
a	b	c
e	f	g
h	i	j
l	m	n
i	j	k

s

A	B	C	D	E
x	y	z	u	w
r	m	o	s	t
e	f	g	l	m
a	b	c	l	m
l	m	n	l	m
i	j	k	l	m
h	i	j	l	m

En este caso S={A,B,C}, R={A,B,C,D,E} de modo que S⊆ R, por otro lado la estructura de r ÷ s está dada por R ~ S = {D, E}, se cumple además que :
$t_r[S] = t_s[S]$, para cada fila en s. Por lo que el resultado es:

D	E
l	m

Figura 7.48 Resultado de la Operació División

Esta operación es útil para las consultas del tipo “Para todo”, ”Todos” así en el ejemplo anterior se indicaría de la siguiente manera : Encontrar las Tuplas de r(R ~ S) que están relacionadas con todas las filas de s a través de los atributos de r(S). Como podemos observar solo (l,m) se relaciona con todos las filas de s

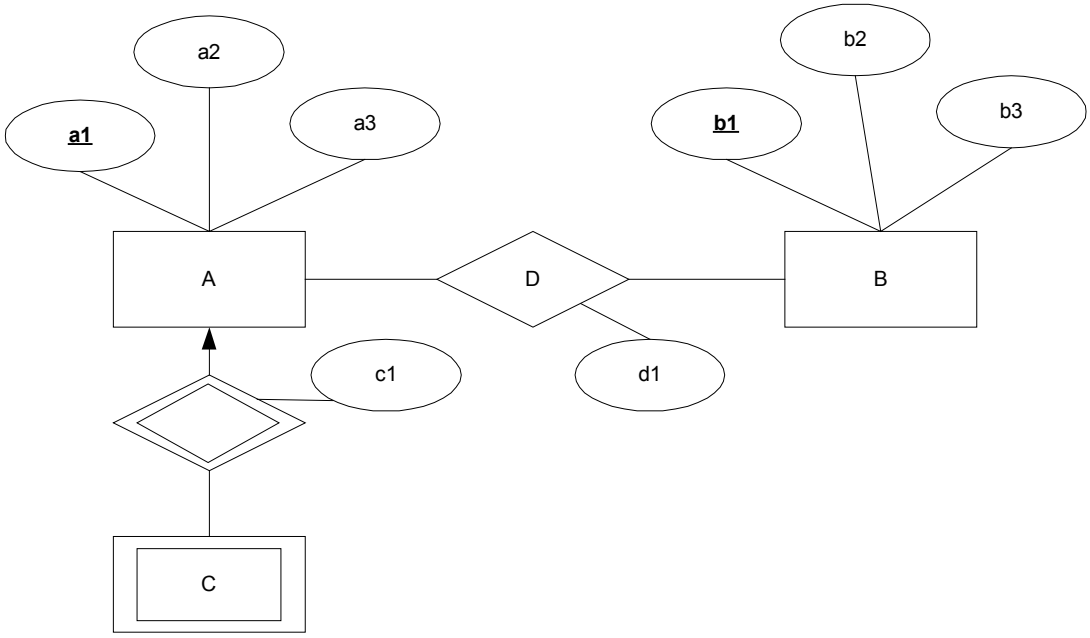
7.5.3.2 El Álgebra Relacional y el modelo E-R

Una vez diseñado el diagrama E-R se puede utilizar para facilitar el obtener las consultas requeridas del sistema, utilizando el álgebra relacional, dado que el diagrama nos ofrece de forma gráfica una panorámica de cómo están vinculadas todas las tablas de la Base de Datos.

Ejemplo :

Sea el siguiente Diagrama E-R, compuesto por dos entidades fuertes
A, B y una entidad débil C.

Utilizando el Álgebra relacional Obtener : Los siguientes predicados basados en este Diagrama E-R
a)Obtener a2,b2, siempre y cuando d1 cumpla una determinada condición que llamaremos Cond (d1).
b)Obtener a3, c1 con Cond(a2,a3) (en este caso la condición lógica involucra tanto a a2 como a a3)
c)Obtener b2 con cond (c1)



Solución

a) Según el predicado las tablas involucradas son : A, B y D donde A, B son entidades fuertes y D la relación de estas entidades. Para resolver el predicado debemos unir estas tablas mediante el producto natural filtrar las filas mediante la operación de Selección y finalmente filtrar las columnas mediante la operación de Proyección, de la siguiente manera :

$$\pi_{A.a2,B.b2}\sigma_{Cond(d1)}\ ((D\ |x|\ A)\ |x|\ B) =$$
$$\pi_{A.a2,B.b2}\sigma_{Cond(d1)}\ (D\ |x|\ A\ |x|\ B)$$

Se puede demostrar con facilidad que el producto cartesiano posee las propiedades conmutativas y asociativas por lo tanto no debe preocupar el orden en que se coloquen las tablas en el producto natural.

b)En este caso se trata de unir una entidad fuerte A con una entidad débil C, la condición involucra a dos atributos de A.

Solución

$$\Pi_{A.a3,C.c1}\sigma_{Cond(a2,a3)}(C \text{ lxl } A)$$

c) En este caso se involucran directamente las tablas B(una entidad fuerte) y C(Una entidad débil) sin embargo B si se observa el diagrama no existe una relación directa entre estos dos objetos del diagrama sino que de una forma indirecta a través de A.

Solución

$$\Pi_{B.b2}\sigma_{Cond(c1)}\ (C \text{ lxl } A \text{ lxl } B)$$

7.5.4 Operaciones del Álgebra Relacional Extendida

Las operaciones del álgebra relacional se han ampliado para poder efectuar operaciones que son de uso común en una base de datos o que resuelven determinada problemática. Entre estas operaciones están : a)La Proyección generalizada la cual posibilita implementar campos calculados b) Las funciones de Agregación que permiten obtener resultados a partir de funciones que trabajan sobre determinadas agrupaciones de los atributos de una relación c) La reunión Externa que posibilita efectuar operaciones similares al producto natural pero que toman en cuenta los valores nulos en una relación.

7.5.4.1 La Proyección Generalizada

Esta ampliación de la operación proyección permite incluir campos calculados posibilitando además nombrar de nuevo estos nuevos atributos.

Forma General : Sea una relación r con estructura $\{E_1,E_2,\ldots.E_n\}$ entonces la proyección generalizada tiene la forma : $\Pi_{F1 \text{ as } N1,F2 \text{ as } N2,\ldots. Fm \text{ as } Nm}$ (R) donde cada Fi es una función que depende de los atributos de la relación o es un valor constante.

Ejemplo:

Sea R una relación con atributos numéricos E_1,E_2 y se requiere obtener una nueva relación R1 con atributos $\{E_1,E_3\}$ donde $E3 = E_1*E_2$

R

E_1	E_2
1	5
2	3
3	9

Solución

$\Pi_{F1,F2 \text{ as } E3}$ (R) donde F1= E_1 (Una constante), F2= $E_1 * E_2$, así una instancia de R es :

E_1	E_3
1	5
2	6
3	27

Figura 7.49 Resultado de la proyección Generalizada

7.5.4.2 Funciones de Agregación

Estas funciones están estrechamente ligadas con la operación de agregación Ç cuya forma general es la siguiente:

$G_1, G_2, \ldots\ldots G_n$ Ç $_{F1(A1), F1(A2), \ldots. F1(Am)}$ (E)

Donde $G_1, G_2, \ldots\ldots G_n$ son atributos de E, por otra parte :

A_1 es el atributo de E donde se aplica la función F_1

A_2 es el atributo de E donde se aplica la función F_2

.

.

A_m es el atributo de E donde se aplica la función F_m

Las principales funciones de agregación son: La suma (sum), el promedio (Avg), conteo de valores en una determinada agrupación(count), el menor de los valores (min), el mayor de los valores (max).

Ejemplo:

Sea la tabla R dada por :

X1	X2	X3	X4
a	1	1	1
b	2	3	1
a	3	5	1
b	2	6	1
a	1	5	2
a	2	1	2
b	4	7	2

Obtener la consulta del álgebra relacional generalizada dada por la expresión :

X_1 Ç $_{sum(X2), count(X4)}$ (R); X_1 :Atributo de Agrupación,

Valores de Agrupación para X_1 = {a,b}.

Solución

X1	Sum(X2)	Count(X4)
a	8	4
b	8	3

Figura 7.50 agrupamiento con funciones de agregación

Ejemplo
Obtener la consulta sobre R, dada por la siguiente expresión :
$X_1, X_2 \mathcal{G}_{sum(X3), count(X4)}(R)$

Solución

X1	X2	sum(X3)	Count(X4)
a	1	6	2
b	2	9	2
a	3	5	1
a	2	1	1
b	4	7	1

Figura 7.51 Agrupamiento sobre dos atributos

Se debe hacer notar que si no existen atributos de agrupación en la consulta que utiliza una operación de agregación se aplican sobre todos los valores de los atributos es decir que no existe agrupación en este caso.

7.5.4.3 Reunión externa

Esta operación es una ampliación del producto natural dado que añade tuplas a la relación resultante tomando en cuenta los valores faltantes y asumiendolos como valores nulos.

En esta operación surgen los siguientes casos, sean R1 y R2 dos relaciones cuyo atributo común es A1, ahora bien si los conjuntos R1.A1, R2.a1 son iguales no hay problema de reunión externa y la problemática de reunión de las dos relaciones se resuelve con la operación producto natural R1 |x| R2.

La problemática en que interviene reunión externa se produce cuando los conjuntos R1.A1 y R2.a1 son diferentes.

Caso (a) Reunión Externa por la izquierda:
Se puede aplicar cuando el siguiente predicado es verdadero: $\exists\, x \in R1.A1 \ni x \notin R2.A1$, la operación en este caso se representa por]x| y se procede de la siguiente manera:

- Para aquellos valores en R1.A1 $\cap$ R2.A1 (Los que están en ambos) se aplica la operación producto natural
- $\forall\, x \in R1.A1 \ni x \notin R2.A1$ se aplica de nuevo el producto natural asumiendo que la tupla (Null, Null,.....x,....Null) existe en R2

Caso (b) Reunión externa por la derecha:
Se puede aplicar cuando el siguiente predicado es verdadero: $\exists\, x \in R2.A1 \ni x \notin R1.A1$ la operación en este caso se representa por |x[y se procede de la siguiente manera :

- Para aquellos valores en R1.A1 $\cap$ R2.A1 se aplica la operación producto natural
- $\forall\, x \in R2.A1 \ni x \notin R1.A1$ se aplica de nuevo el producto natural asumiendo que la tupla (Null, Null,.....*x*,....Null) existe en R1.

Caso (c) Reunión externa total :

Se puede aplicar cuando el siguiente predicado es verdadero : $(\exists\, x \in R2.A1 \ni x \notin R1.A1) \wedge (\exists\, y \in R2.A1 \ni y \notin R1.A1)$, es decir cuando se puede aplicar reunión externa por la derecha y por la izquierda. Se representa por el símbolo]x[. en este caso se produce de la siguiente manera:

- Para aquellos valores en R1.A1 $\cap$ R2.A1 se aplica la operación de producto natural
- Se aplica Reunión externa por la derecha o por la izquierda en función de que el elemento no común se encuentre en R2.A1 o en R1.A1, lo cual equivale a aplicar la operación R1 |x[R2 $\cup$ R1]x| R2

Ejemplo:
Sean R1 y R2 las siguientes tablas

R1

X	Y	Z
a	l	1
b	m	6
c	r	2
d	x	4
r	t	2
u	v	9

R2

A	Y	W
2	a	y
3	b	v
1	l	c
5	x	m
8	t	n

Obtener: (a) R1 |x[R2 (b) R1]x| R2 (c) R1]x[R2

Solución

(a) En este caso los elementos que están en R2.Y pero que no están en R1.Y $\cap$ R2.Y ={l,x,t} son a,b por tanto se efectuará el producto natural con los valores l,x,t y se añadirán 2 filas correspondiendo a los valores a,b que tendrán valores nulo en los atributos correspondientes a R1.

X	Y	Z	A	W
Null	a	Null	2	y
Null	b	Null	3	v
a	l	1	1	c
d	x	4	5	m
r	t	2	8	n

Figura 7.52 Reunión externa por la Derecha

b) Como antes R1.Y ⋂ R2.Y ={l,x,t}, entonces los elementos de R1.Y que no están en la intersección son: m,r,v

X	Y	Z	A	W
B	M	6	Null	Null
C	R	2	Null	Null
U	V	9	Null	Null
A	L	1	1	C
D	X	4	5	M
r	t	2	8	n

Figura 7.53 Reunión Externa por la izquierda

(c) R1]x[R2 = R1 |x[R2 ⋃ R1]x| R2

X	Y	Z	A	W
Null	a	Null	2	y
Null	b	Null	3	v
b	m	6	Null	Null
c	r	2	Null	Null
u	v	9	Null	Null
a	l	1	1	c
d	x	4	5	m
r	t	2	8	n

Figura 7.54 Reunión Externa Total

7.5.5 Modificación de la Base de Datos

Las operaciones que modifican el contenido de la base de datos son: (a) La operación de borrado (b) La operación de Inserción (c) La Operación de actualización

7.5.5.1 Operación de borrado

Esta consiste en eliminar de una relación R de la base de datos un conjunto finito de tuplas.

Forma General : Sea r una relación Y sea E una expresión del álgebra relacional que representa el conjunto finito de tuplas r a eliminar, entonces r ← r-E indica que r ha sido modificada extrayendo de ella las tuplas contenidas en la expresión E.
Ejemplo
Sea la relación Estudiante {N_Carnet, Nombre, Edad, Sexo, Procedencia} y se desea borrar a todos los estudiantes cuya procedencia es Managua.

Solución

Estudiante ← Estudiante ~ $\sigma_{\text{Procedencia="Managua"}}$ (Estudiante)

7.5.5.2 La Operación Inserción

Para insertar datos en una relación se debe especificar la tupla que se va a insertar o escribir una consulta cuyo resultado sea un conjunto de tuplas que vayan a insertarse. Evidentemente el valor de los atributos de las tuplas insertadas deben ser miembros del dominio de cada atributo. De manera similar, las tuplas insertadas deben ser de la aridad correcta. en el álgebra relacional las inserciones se expresan de la siguiente manera:

$r \leftarrow r \cup E$

donde r es una relación y E es una expresión del álgebra relacional. En el caso de la inserción de una sola tupla se expresa haciendo que E sea una relación constante que contiene una sola tupla.

Ejemplo:

Supóngase que se desea insertar la siguiente información: Gómez posee 1,200 en la cuenta C-973 en la sucursal “Linda Vista”. Las inserciones requeridas serían las siguientes:

$$cuenta \leftarrow \cup \{(C-973,"LindaVista",1200)\}$$
$$impositor \leftarrow \cup \{("Gomez", C-973)\}$$

De forma mas general, puede suceder que se desee insertar tuplas en función del resultado de una consulta.

Supóngase que se desea ofrecer una nueva cuenta de ahorro con C$ 200 de regalo a todos los clientes con préstamos concedidos en la sucursal “Linda Vista”. El mismo número de préstamo se utilizará como código de la nueva cuenta. La expresión es la siguiente:

$$r_1 \leftarrow (\sigma_{Nombre\_Sucursal="Linda\ Vista"}\ (prestatario \left|\times\right| préstamo))$$
$$r_2 \leftarrow \pi_{Nombre\_Sucursal,\ Número\_\text{Pr}éstamo}(r_1)$$
$$cuenta \leftarrow cuenta \cup (r_2 \times \{(200)\})$$
$$impositor \leftarrow impositor \cup \pi_{Nombre\_Cliente,\ Número\_\text{Pr}éstamo}(r_1)$$

En lugar de especificar las tuplas como se hizo anteriormente, se especifican un conjunto de tuplas que se insertan en las relaciones cuenta e impositor.

7.5.5.3 Actualización

Es común que en algunas situaciones se desee modificar un valor contenido en una tupla sin modificar el resto de valores. Para realizar esta modificación se puede utilizar la proyección generalizada de la siguiente manera:

$$r \leftarrow \pi_{F_1,F_2\ .....F_n}(r)$$

donde F_i significa el i-ésimo atributo de r, el cual puede ser el nombre del atributo en el caso de no existir modificación ó una expresión en función de los restantes atributos la cual será la responsable de la actualización del atributo de la relación r. ejemplo:

$$Cuenta \leftarrow \pi_{Nombre\_Sucursal,\ Número\_Saldo,\ Saldo*1.05}(Cuenta)$$

En este caso los saldos de las cuentas se han incrementado en un 5%

7.5.5.4 El Lenguaje Cálculo relacional de Tuplas

Este Lenguaje a Diferencia del Álgebra Relacional es no procedimental pues no indica mediante un conjunto de operaciones como se va a obtener la consulta requerida sino que simplemente la describe.
En general una consulta del cálculo relacional se expresa así :
$\{t|p(t)\}$: El Conjunto de tuplas t, de modo que el predicado p(t) es verdadero.
Ejemplo
Considérese la tabla préstamo, indicada abajo, Encontrar las tuplas de préstamo en que cantidad $\geq$ 2000

Préstamo

Cod_Cliente	Cod_Préstamo	Cantidad
001	P-005	2000
002	P-006	3200
003	P-004	1500
004	P-008	2600
005	P-002	1300

$\{x|x \in \text{Préstamo} \wedge [Cantidad] \geq 2000\}$, equivalente a $\sigma_{\text{Préstamo.Cantidad} \geq 2000}$ (Préstamo).

El Equivalente de la Proyección en el Cálculo Relacional de tuplas.
En este caso se deben de utilizar dos variables, ya que ahora la relación resultante de la consulta no tiene el mismo número de atributos que la relación sobre la cual se aplica el predicado p(t), entonces una variable que denominaremos t representará las tuplas de la relación resultante y la otra variable s que representará tuplas de la relación original, de modo que si
$A \subseteq R$
Donde A = { A1,A2,.. Am} entonces
π_A (R) = $\{t|\exists s \in R(t[A1] = s[A1] \wedge t[A2] = s[A2] \ldots t[Am] = s[Am])$
Combinando la selección y la proyección, la expresión equivalente en el Cálculo relacional de tuplas será la siguiente :
$\pi_A \sigma_{cond(R)}$ (R) =
$\{t|\exists s \in R(t[A1] = s[A1] \wedge t[A2] = s[A2] \ldots t[Am] = s[Am] \wedge \text{cond(R)})\}$
Ejemplox
Sea la Tabla indicada abajo obtener el equivalente de $\pi_{Atr1} \sigma_{Atr3 \geq 8}$ (Tablax) en el Cálculo relacional de Tuplas.

Tablax

Atr1	Atr2	Atr3
a	e	8
b	e	10
c	f	6
d	f	5

Solución

$\pi_{Atr1}\ \sigma_{Atr3 \geq 8}$ (Tablax)= $\{t \mid \exists s \in Tablax(t[Atr1] = s[Atr1] \wedge s[Atr3] \geq 8)\}$

Implementación del Equivalente al Producto Cartesiano

A x B= $\{t \mid \exists x \in A \wedge \exists y \in B\}$

Implementación del equivalente al Producto Natural.

A | x | B=

$\{t \mid \exists x \in A \wedge \exists y \in B \wedge (x[a] = y[b])\}$ *donde a es un atributo de A y b un atributo de B*

Ejemplo

En función del diagrama E-R indicado abajo, obtener usando el lenguaje Cálculo Relacional de Tuplas, los atributos T_2 y S_2 con la condición $S_1 \leq 10$

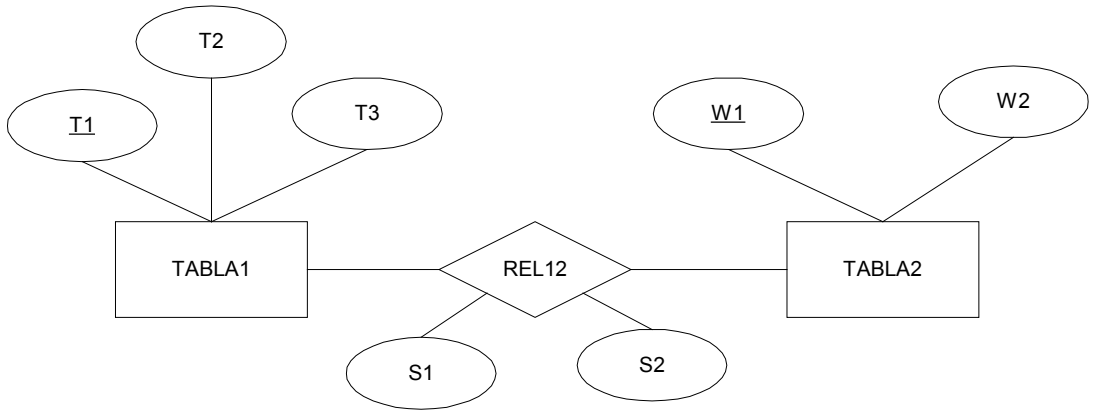


Figura 7.55 Diagrama E/R : unión de tablas en el cálculo relacional

Solución

Notemos que T_2 Є TABLA1 y S_2 Є REL12, por tanto para dar respuesta a la consulta se deben de unir estas tablas, para posteriormente efectuar la proyección y la condición que debe cumplir la relación resultante.

Consulta =

$\{t \mid \exists x \in TABLA1 \wedge \exists y \in REL12\ (x[T1] = y[T1]) \wedge (t[T2] = x[T2]) \wedge (t[S2] = y[S2]) \wedge y[S1] \leq 10\}$

7.5.5.5 El Cálculo Relacional de Dominios

Las expresiones del Cálculo Relacional de Dominios son de la forma :

$\{< x_1, x_2, \cdots, x_n > | P(x_1, x_2, \cdots, x_n)\}$ donde $x_1, x_2, \cdots, x_n$ representan las variables de Dominio, P representa una condición Lógica en función de las variables de Dominio.

Concepto de Variable de Dominio
a es una variable de Dominio si su dominio son los valores de un determinado atributo de una relación

Representación de una Tabla
Sea R una relación entonces la representación de R en el cálculo relacional de dominios se expresa de la siguiente manera :

$R \equiv \{< x_1, x_2, \cdots, x_n > | < x_1, x_2, \cdots, x_n > \in R\}$ donde $x_1, x_2, \cdots, x_n$ son variables de Dominio.

La operación Selección en el Cálculo Relacional de Dominios

$\sigma_{Cond(R)}$ (R) $\equiv \{< x_1, x_2, \cdots, x_n > | < x_1, x_2, \cdots, x_n > \in R \wedge Cond\}$ en este caso *Cond* es una condición lógica en función lógica con dominio en las variables de Dominio que definen la relación o Tabla.

La Operación Proyección en el Cálculo Relacional de Dominios.
Sea S un sub conjunto del conjunto de atributos de R tal que S={s_1, s_2, s_3,... s_m } entonces

π_S(R) $\equiv \{< x_{s1}, x_{s2}, \cdots, x_{sm} > | < x_1, x_2, \cdots, x_n > \in R\}$ donde $x_{s1}, x_{s2}, \cdots, x_{sm}$ son las respectivas variables de Dominio de s_1, s_2, s_3,... s_m.

Ejemplo
Sea la tabla Estudiante1 indicada abajo, generar mediante el cálculo relacional de Dominios Los Nombres de los estudiantes cuyas edades sean mayores a 25 años.

Estudiante1

Nombre	Edad	Carrera
Juan	22	Computación
Pedro	25	Matemáticas
María	32	Biología
José	21	Computación
Carlos	26	Matemáticas
Lorena	28	Computación

Consulta $\equiv \{< a > | < a, b, c > \in Estudiante1 \wedge b > 25\}$

En este caso las variables de tupla a, b, c representan los atributos: Nombre, Edad, Carrera respectivamente.

La relación resultante de la Consulta es

Nombre
María
Carlos
Lorena

Figura 7.56 Proyección y Selección en el cálculo relacional de tuplas

El Producto Cartesiano en el Cálculo Relacional de Dominios
Sean A, B dos relaciones y sean $x_1, x_2, x_3 \ldots x_n$ las variables de tupla que representan los atributos de A, asimismo $y_1, y_2, y_3 \ldots y_m$ representan los de B, entonces :
A x B

$$\equiv \{< x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_m > \mid \exists < x_1, x_2, \ldots x_n > \in A \wedge \exists < y_1, y_2, \ldots y_m > \in B\}$$

El Producto Natural en el Cálculo Relacional de Dominios
Sean A, B dos relaciones y sean $x_1, x_2, x_3 \ldots x_n$ las variables de tupla que representan los atributos de A, asimismo $x_1, y_2, y_3 \ldots y_m$ representan los de B, entonces :
A | x | B

$$\equiv \{< x_1, x_2, \ldots, x_n, y_2, \ldots y_m > \mid \exists x_2, \ldots x_n (< x_1, x_2, \ldots, x_n > \in A \wedge \exists y_2, \ldots y_m \ (< x_1, y_2, \ldots y_n > \in B)\}$$

En este caso hemos tomado como atributo con dominio y nombre común el representado por la variable de tupla x_1

Ejemplo
Dado el siguiente Esquema Relacional obtener las siguientes consultas utilizando el Cálculo Relacional de Dominios.

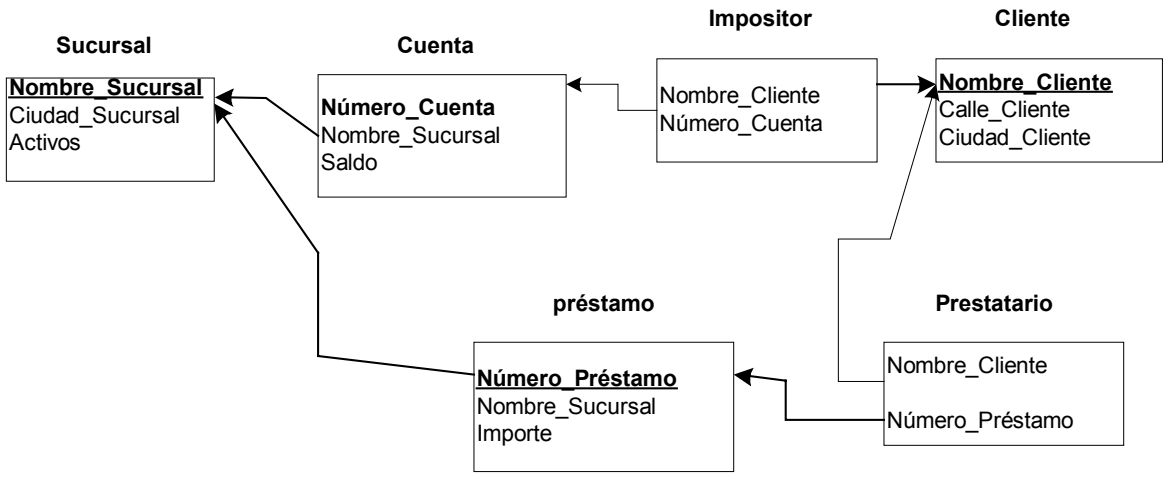


Figura 7.57 Esquema Relacional de una Entidad Bancaria

a) Obtener mediante una consulta el nombre de todos los clientes que tienen concedido un préstamo en la sucursal de Sutiava y averiguar el monto del mismo.

b) Obtener el Nombre de todos los clientes que tienen concedido un préstamo, una cuenta abierta o ambas cosas en la sucursal Sutiava

c) Obtener el nombre de todos los clientes que tienen una cuenta abierta en todas las sucursales situadas en la ciudad de Granada.

Solución

a) En este caso si observamos el esquema relacional, para dar respuesta a esta consulta se requiere unir dos tablas, la tabla préstamo y la tabla prestatario, nótese además las variables de Dominio asignadas a cada uno de los atributos de las tablas involucradas, esto se muestra en el esquema relacional.

Consulta $\equiv$

$$\{< n,i >\mid \exists n,p\,(<n,p>\in prestatario) \wedge \exists s,i\,(<p,s,i>\in préstamo \wedge s = "Sutiava")\}$$

b) En este caso el nombre de todos los clientes que tienen concedido un préstamo se obtiene en la tabla *prestatario,* los que tienen una cuenta se encuentran en la tabla *impositor,* Nótese consultando a las tablas *préstamo* y *cuenta*, se obtiene el nombre de la sucursal.
Por lo tanto la expresión solicitada es:

$\equiv$

$$\{< n >\mid \exists n,p <n,p>\in prestatario) \wedge \exists s,i <p,s,i>\in préstamo \wedge s = "Sutiava"\} \cup$$
$$\exists nc\,((<n,nc>\in impositor \wedge \exists s,sa(<nc,s,sa>\in cuenta) \wedge s = "Sutiava"\}$$

c)

$$\{< n >\mid \exists nc\,(<n,nc>\in impositor) \wedge (\forall s,a\,(<s,"Granada",a>\in sucursal) \Rightarrow$$
$$\exists nc,sa < nc,s,sa >\in cuenta\}$$

7.5.6 Teoría de la Normalización

El diseño de una base de datos relacional se puede realizar mediante la metodología que se acaba de estudiar, aplicando al mundo real, en una primera fase, un modelo semántico como el E/R a fin de obtener un esquema conceptual; en una segunda fase, se transforma dicho esquema al modelo relacional mediante las reglas indicadas anteriormente. Si bien se insiste en utilizar este enfoque, existe otra posibilidad y es la de plasmar directamente en el modelo relacional nuestra percepción del mundo real sin utilizar como interface el modelo lógico.

Aunque en general, la primera aproximación en la metodología recién estudiada produce un esquema relacional estructurado y con poca redundancia

siempre es conveniente aplicar un conjunto de reglas conocida como **Teoría de la Normalización,** que permite asegurar que un esquema relacional cumple determinadas propiedades, es decir que en este caso se utilizaría la Teoría de Normalización para mejorar el diseño. En el segundo enfoque (Sin E/R) la teoría de Normalización es Imprescindible y esto no es de extrañar pues precisamente el modelo lógico (E/R) està basado en el modelo relacional, recordemos que Codd descubre el modelo relacional en 1970 y que Chen da a conocer sus trabajos sobre el modelo E/R en 1976.

Ejemplo de una relación resultante de un diseño inadecuado

ESCRIBE

AUTOR	NACIONALIDAD	COD_ LIBRO	TITULO	EDITORIAL	AÑO
Date,C.	Norteamericana	23433	Databases	Adisson-W.	1990
Date,C.	Norteamericana	54654	SQL Standard	Adisson-W.	1986
Date,C.	Norteamericana	53235	Guide to Ingres	Adisson-W.	1988
Codd,E.	Norteamericana	97875	Relational M.	Adisson-W.	1990
Gardarin	Francesa	34245	Base de Datos	Paraninfo	1986
Gardarin	Francesa	55366	Comparación BD	Eyrolles	1984
Valduriez	Francesa	86754	Comparación BD	Eyrolles	1984
Kim,W.	Norteamericana	32176	OO Databases	ACM Press	1989
Lochosky	Canadiense	23456	OO Databases	ACM Press	1989

Figura 7.58 Tabla con un diseño inadecuado

Los principales problemas de esta relación se derivan de la gran cantidad de redundancia que presenta. Por ejemplo la nacionalidad del autor se repite por cada libro que este ha escrito y algo similar sucede cuando un libro tiene más de un autor, se repite la editorial y el año de publicación. Estas redundancias producen problemáticas relacionadas con las operaciones de Inserción, modificación y borrado

- Anomalías de inserción : Si se va a dar de alta a un libro, este hecho obliga a insertar en la base de datos tantas filas como autores tenga el libro

- Anomalías de modificación, ya que si se cambia la editorial que edita un determinado libro obliga a modificar todas las tuplas o filas que corresponden a ese libro

- Anomalías de Borrado, ya que el borrado de un libro obliga a borrar varias filas, tantas como autores tenga ese libro y asimismo el borrado de un autor hace que se borren tantas tuplas como libros ha escrito ese autor.

En función de este ejemplo podemos observar que en general las modificaciones involucran a varias tuplas, siendo en este caso el usuario el resposable de

efectuar en todas las tuplas afectadas de la relación, lo cual es un potencial peligro para la pérdida de integridad y unido a este hecho lo ineficiente que se vuelve el sistema al tener que realizarse tantas modficaciones que en general son innecesarias

Además de las anomalías antes indicadas existen problemas adicionales generados por estas relaciones con un diseño inadecuado así si se desea incluir solamnete la información de un autor es decir su nombre y Nacionalidad no podríamos hacerlo pues la llave primaria de la relación **Cod_Libro** no podría quedar a null por el hecho de ser parte de la llave primaria (Cod_Libro+Autor)
Por otro lado al eliminar un libro se eliminan también los autores de ese libro si solamente tuviesen un libro editado.

Esta relación presenta todos estos problemas pues viola un principio básico del diseño el cual indica que:

"HECHOS DISTINTOS SE DEBEN ALMACENAR EN OBJETOS DISTINTOS"
Uno de los grandes problemas que se presentan muy a menudo en el diseño que llevan a situaciones parecidas a la antes indicada son:

- Los Diseñadores no llegan a comprender de forma completa y precisa el Universo del Discurso

- Falta de Comunicación entre los futuros usuarios del Sistema y el Diseñador

De modo que para evitar los problemas listados anteriormente, en lugar de una relación tendríamos que utilizar 3 :

LIBRO(Cod_libro,Título,Editorial, Año)
AUTOR(Nombre,Nacionalidad)
ESCRIBE(Cod_Libro,Nombre)

Es precisamente la Teoría de la Normalización que bajo un determinado proceso hará concluir que este es el conjunto de relaciones adecuadas para evitar las problemáticas de la relación original. Aunque se pudiese pensar que algunas consultas serían más rápidas con una relación, la realidad es que una base de datos normalizada nos permite planificar mejor como vamos a obtener la información.

7.5.6.1 Formas Normales

A continuación examinaremos las seis formas normales más habituales utilizadas.

7.5.6.1.1 Primera Forma Normal

Un Relación R está en Primera Forma Normal (1FN) si cada uno de los Dominios de todos los atributos de R son atómicos.

Ejemplo de una Relación que no está en 1NF (donde la clave primaria es N_Carnet)

Estudiante

N_Carnet	Nombre	Teléfono
1	Juan	311-4268
2	Pedro	311-6432 ; 088-2192
3	Roberto	311-7522; 02-9641; 088-6232
4	María	311-4132

Figura 7.59 Tabla con violación a la Primera forma Normal

No está en 1 FN pues el Dominio del atributo Teléfono no es atómico, ya que algunas filas cuentan con valores múltiples en este atributo.

Paso a la Primera Forma Normal
Para resolver la problemática de la Primera Forma Normal, se convierten los atributos no atómicos en atómicos, añadiéndole filas a la relación original y añadiendo el campo que presentaba duplicados a la clave primaria, quedando la relación Estudiante de la siguiente manera:

N_Carnet	Nombre	Teléfono
1	Juan	311-4268
2	Pedro	311-6432
2	Pedro	088-2192
3	Roberto	311-7522;
3	Roberto	02-9641
3	Roberto	088-6232
4	María	311-4132

Figura 7.60 Tabla Normalizada a la Primera forma Normal

Como podemos observar en este ejemplo la solución de la primera forma normal nos lleva a otra problemática: la Redundancia de Información que será tratada por las posteriores formas normales.

7.5.6.1.2 Segunda Forma Normal

Antes de definir la segunda forma normal es necesario estudiar un concepto que está estrechamente ligado con la segunda forma Normal y es el concepto de

Dependencia Funcional, para lo cual repasaremos el concepto de función.
Una relación (Matemática) f es una función, ssi la relación cumple con las siguientes propiedades:
1) Los miembros de f son pares ordenados

2) Si (x,y) y (x,z) son miembros de f entonces y=z es decir la imagen de los elementos del dominio debe ser única

Ejemplo:

Sea la relación f definida por la siguiente regla (ver figura 7.91 abajo):

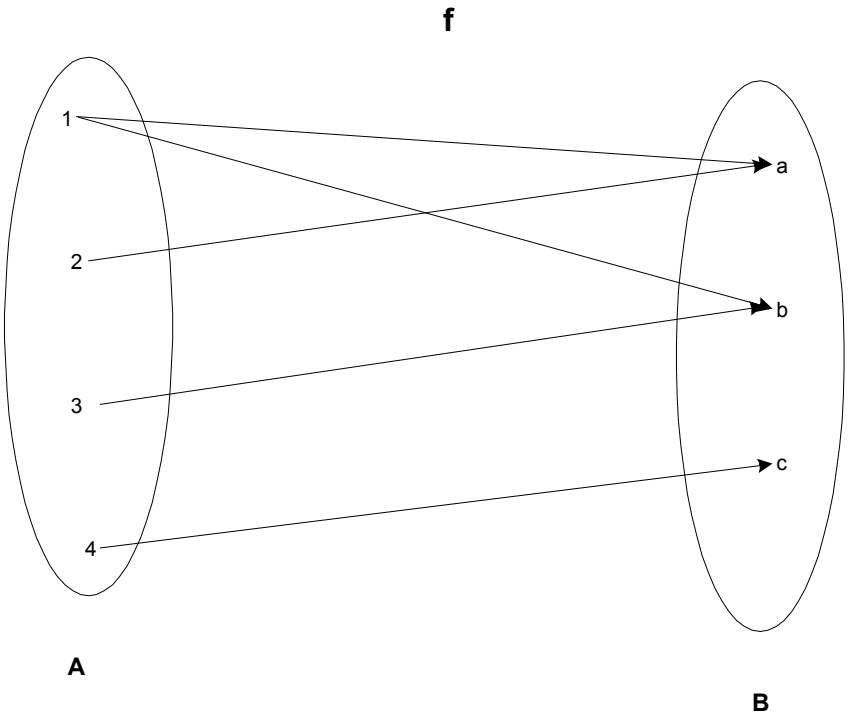


Figura 7.61 Violación del concepto de función

La relación f no es una función ya que el elemento 1 de A tiene dos imágenes, visto en forma columnar este mismo ejemplo la representación quedaría de la siguiente manera:

A	B
1	a
1	b
2	2
3	b
4	c

Figura 7.62 Forma Tabular de la relación etre los conjuntos A y B

En este caso decimos que el atributo **B** no depende funcionalmente de **A,** o que en este caso **A** no determina a **B**

Una Relación está en **Segunda Forma Normal** si está en 1FN y si la llave primaria de la relación determina completamente a todos los atributos de R
Se dice que una llave Primaria Compuesta por los atributos Atr1,Atr2,...Atrn Determina completamente a todos los atributos de R si y solo si ningún sub conjunto de la llave primaria determina a alguno de los atributos no primos de la relación.

Atributo no primo : Un Atributo X de una Relación R es no primo si y solo si X no pertenece a U donde U es el Conjunto de Atributos que conforman una llave candidata de la relación.
Retomando la tabla del ejemplo sobre la primera forma normal, observamos que existe redundancia de información y los problemas relacionados con esta al momento de efectuar modificaciones. La relación no se encuentra en Segunda Forma Normal :
N_Carnet y Nombre violan las reglas de unicidad a partir de la instancia presentada abajo por otra parte Teléfono podría violar la propiedad de unicidad de las llaves primarias dado que dos estudiantes eventualmente pueden compartir el mismo número de teléfono.
Llave Primaria: N_Carnet+ Teléfono, además el contexto de la vida real nos indica que N_Carnet determina a Nombre (Nombre depende funcionalmente de N_Carnet) esto último garantiza la unicidad de la llave primaria ya que si existiesen las tuplas (n1, Nombre1,tel1) y (n1,Nombre2,tel1) implicaría que Nombre1=Nombre2 debido a que N_Carnet determina a Nombre.

N_Carnet	Nombre	Teléfono
1	Juan	311-4268
2	Pedro	311-6432
2	Pedro	088-2192
3	Roberto	311-7522;
3	Roberto	02-9641
3	Roberto	088-6232
4	María	311-4132

Figura 7.63 Tabla con violación a la Segunda forma Normal

Ahora bien el hecho de que N_Carnet + Nombre es la llave primaria y que N_Carnet $\rightarrow$ Nombre indican que La relación Estudiante no se encuentra en **Segunda Forma Normal**

Solución a la problemática de la segunda Forma Normal
Siguiendo con el ejemplo anterior de la tabla original Estudiante se generan dos tablas :

N_Carnet	Nombre
1	Juan
2	Pedro
3	Roberto
4	María

Con Llave Primaria N_Carnet

N_Carnet	Teléfono
1	311-4268
2	311-6432
2	088-2162
3	311-7522
3	02-9641
3	088-6232
4	311-4132

Con Llave Primaria N_Carnet + Teléfono

Figura 7.64 Tablas Normalizadas a la segunda forma Normal

Las tablas resultantes de la descomposición de la tabla original están en la Segunda Forma Normal ya que en la primera, la llave primaria es simple, en la segunda es compuesta pero no hay más atributos es decir no existen atributos no primos.

7.5.6.1.3 Tercera Forma Normal

Una relación está en Tercera Forma Normal(3FN) si solo si está en 2FN y todo atributo no primo depende de forma completa y no transitiva de la llave primaria.

Un Atributo X de una relación r depende Completamente de la llave primaria si este no depende de ningún sub conjunto de ella

Un Atributo X de una relación r depende de forma Transitiva de la llave primaria U de la relación si existe un atributo S en r tal que

$$U \rightarrow S \rightarrow X$$

En otras palabras una relación R está en 3FN si y solo si los atributos no primos de R (Si los Hay) son dependientes en forma completa de la llave primaria (2FN) y mutuamente independientes (Es decir si no hay relación entre esos atributos)

Ejemplo sea la relación Estud donde cada Estudiante puede llevar cada asignatura solamente una vez

Estud

N_Carnet	Nombre	Edad	Cod_A	N_Asig	Dpto	Director
1	Juan	22	Mat1	Cálculo I	Mat	C.Flores
1	Juan	22	Mat2	Álgebra1	Mat	C.Flores
1	Juan	22	Comp1	Prog1	Comp	R.Espinoza
1	Juan	22	Comp2	Sist Oper1	Comp	R.Espinoza
2	Pedro	23	Mat1	Cálculo 1	Mat	C.Flores
2	Pedro	23	Comp2	Sist Oper 1	Comp	R.Espinoza

¿Esta relación está en Segunda Forma Normal ?
La relación Estud no se encuentra en 2FN ya que si tomamos como llave primaria la llave candidata N_Carnet + Cod_A bajo el supuesto que un mismo

estudiante solo puede llevar una determinada asignatura una sola vez, observamos que N_Carnet determina a Nombre, Edad y por otra parte el atributo primo cod_Asig determina a: N_Asig, Dpto y Director.
Siguiendo el procedimiento del ejemplo anterior se pasará Estud a 2NF, obteniéndose las siguientes 3 tablas normalizadas a 2FN

Estud1

N_Carnet	Nombre	Edad
1	Juan	22
2	Pedro	23

Estud2

Cod_A	N_Asig	Dpto	Director
Mat1	Cálculo I	Mat	C.Flores
Mat2	Álgebra1	Mat	C.Flores
Comp1	Prog1	Comp	R.Espinoza
Comp2	Sist Oper1	Comp	R.Espinoza

Estud3

N_Carnet	Cod_A
1	Mat1
1	Mat2
1	Comp1
1	Comp2
2	Mat1
2	Comp2

Estud1, Estud2 y estud3 están en 2FN pues Estud1 y estud2 tienen Llaves primarias simples y por otro lado Stud3 tiene como llave primaria N_Carnet+Cod_A por lo tanto no tiene atributos no primos.

Revisión de la Tercera forma Normal
Estud1 está en 3FN pues aunque Nombre determina edad no viola 3FN pues Nombre es una llave candidata, Estud3 está trivialmente en 3FN pues solo contiene la llave primaria (N_Carnet+Cod_A). Por otra parte Stud2 no se encuentra en 3FN ya que Cod_a determina a Dpto y Dpto determina a Director donde Dpto es un atributo no primo de la relación.

Paso de Stud2 a la Tercera Forma Normal.

Stud2

Cod_A	N_Asig	Dpto	Director
Mat1	Cálculo I	Mat	C.Flores
Mat2	Álgebra1	Mat	C.Flores
Comp1	Prog1	Comp	R.Espinoza
Comp2	Sist Oper1	Comp	R.Espinoza

La tabla original en este caso Stud2, se descompone en 2 tablas en función de los atributos involucrados en la violación de la Tercera forma normal en este caso los atributos: Dpto y Director.

1. Se forma una nueva tabla Stud21 con los atributos Dpto y Director con llave primaria Dpto

2. Se elimina el atributo Director de la tabla original es decir de Stud2, las nuevas tablas quedarían como sigue:

Stud21

Dpto	Director
Mat	C.Flores
Comp	R.Espinoza

Stud22

Cod_A	N_Asig	Dpto
Mat1	Cálculo I	Mat
Mat2	Álgebra1	Mat
Comp1	Prog1	Comp
Comp2	Sist Oper1	Comp

Figura 7.65 descomposión de la tabla Stud

Esquema Relacional resultante

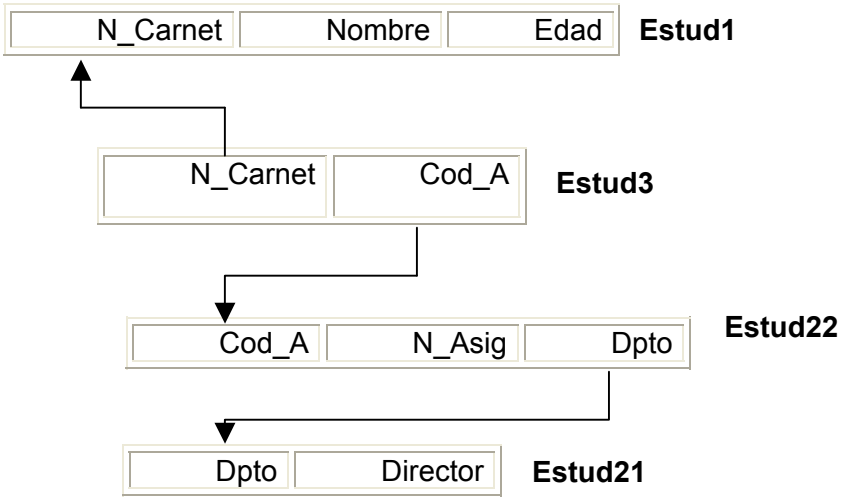


Figura 7.66 Esquema Relacional con relaciones normalizadas a 3FN

7.5.6.1.4 La Forma Normal de Boyce-Codd (FNBC)

La forma Normal de boyce-Codd es conceptualmente distinta y mucho más sencilla a la formas normales antes vistas. Las relaciones que satisfacen esta forma normal satisfacen también 2FN y 3FN. LA FNBC se basa en el concepto de **Determinante Funcional** y está soportada en las características de las claves candidatas de la relación.

Concepto de Determinante Funcional

Se Denomina determinante funcional a uno o un conjunto de atributos de una relación R del cual depende funcionalmente de forma completa algún otro atributo de la relación.

7.5.6.1.5 Forma Normal de Boyce-Codd, Definición

Una relación R Satisface la forma Normal de Boyce-Codd(FNBC) si, y solo si se encuentra en 1FN y si cada determinante funcional es una clave candidata

Ej : Sea el Esquema una Base de Datos relacionada con las aulas donde reciben las clases,el lugar donde se encuentran dichas aulas, Las Asignaturas que se imparten en determinado Año académico y las asignaturas que toman los estudiantes. El esquema es el siguiente :

Imparte (Cod_Asignatura, Año_A)

Alumno_2 (N_Carnet,Apellidos,Nombre)

Ubicación (Aula, Lugar)

Matricula_4 (N_Carnet,Cod_Asignatura,Nota,Aula)

Están en FNBC todas las relaciones de esta base de Datos?

La respuesta es si, dado que:

- Los únicos determinantes funcionales son las claves primarias en cada una de las relaciones ya que no existen claves candidatas o alternativas así, de otra forma:

- En todas las relaciones, las únicas dependencias funcionales son las existentes entre los atributos no primos de cada relación y la clave de la misma

Se debe de enfatizar que no se deben de tomar en cuenta las dependencias funcionales triviales, este concepto está más relacionado con las super claves y las Llaves primarias/candidatas compuestas.

Así si α es el conjunto de atributos que conforman una Clave Candidata y $\beta \subseteq \alpha$ entonces $\alpha \rightarrow \beta$ de forma trivial.

Se pueden dar casos en que la relación se encuentre en 3FN pero que no satísfaga FNBC puesto que esta es más restrictiva que 3FN, estos casos particulares se presentan generalmente cuando la tabla tiene determinada particularidad :

- Existen Varias claves Candidatas

- Esas Claves Candidatas son Compuestas y

- Las claves Candidatas se Traslapan Tienen por lo menos un atributo en común)

Por ejemplo consideremos el esquema de la relación Matricula :

Matricula
(N_Carnet, Cod_Asignatura, apellidos, Nombre, Nota, Año_C, aula, Lugar)

Esta relación tiene dos Claves Candidatas : N_Carnet+Cod_Asignatura y Cod_Asignatura+apellidos+Nombre bajo el supuesto que el nombre completo apellidos + Nombre es único.

Las dependencias existentes en la relación **Matrícula** son las siguientes:

Cod_Asignatura → Año_C (Cada Asignatura se ofrece en un determinado año de la carrera)

N_Carnet + Cod_Asignatura → aula (Los estudiantes reciben siempre una determina asignatura en la misma aula).
De forma Similar:
N_Carnet + Cod_Asignatura → Lugar,Nota
Por otra parte y de forma similar :
Cod_Asignatura + apellidos + Nombre → aula,lugar,nota
Aula → lugar
Como podemos observar por las dependencias funcionales existen problemas de la segunda y tercera forma normal en esta relación, utilizan do las descomposiciones explicadas anteriormente para estos casos obtenemos el siguiente esquema:

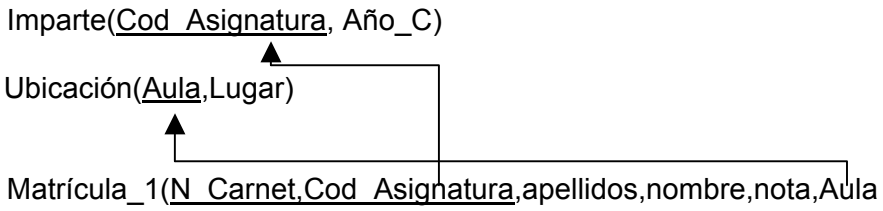


Figura 7.67 Esquema en 3FN con problemas de redundancia

Este esquema se encuentra en 2FN y 3FN, sin embargo podemos observar fácilmente que la relación Matricula_1 tiene redundancia de información ya que N_Carnet → apellidos,nombre (apellidos,nombre → N_Carnet) por cada nota de cada una de las asignaturas que toma el estudiante se repetirá el nombre completo, esta dependencia funcional viola la FNBC ya que N_Carnet es un determinante pero no es una llave candidata en la relación Matricula_1.
Para convertir el esquema a FNBC seguiremos la misma técnica en el sentido que efectuaremos una descomposición sin perdidas de la relación original. Se añadirá al esquema una relación con las claves candidatas de matricula_1 y otra relación que consistirá de la relación original pero eliminando de ella uno de los atributos determinantes que no son llave candidata, el esquema en FNBC quedaría de la siguiente manera:

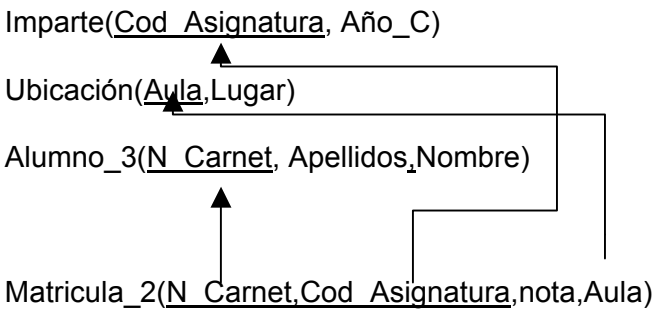


Figura 7.68 Esquema en FNBC

7.5.6.1.6 El Proceso de Descomposición de las Formas Normales

En los procesos de normalización de relaciones que se han llevado a cabo para lograr dichas normalizaciones se ha hecho uso en forma reiterada de descomposiciones binarias asumiendo que estas descomposiciones son sin pérdidas. Este proceso de Descomposición no es un proceso trivial y se debe realizar en función de los siguientes principios :

Descomposición por la aplicación de 2FN

Dada una relación R cuya clave es compuesta de la forma x+y y en esta relación existe una dependencia funcional incompleta de la forma y $\rightarrow$ z donde z es un atributo no primo de la relación R, el proceso de descomposición se debe de realizar de la siguiente manera :

- De la relación original se elimina el atributo z
- Se construye una nueva relación R1(y,z), y como llave primaria de la relación y en la relación original el atributo y se define como llave ajena.

En este proceso x,y,z pueden ser compuestos.

Descomposición por aplcación de la 3FN

Dada una relació r de clave primaria x y dos atributos no primos y,z y se presentan las siguientes dependencias funcionales: $x \rightarrow y \wedge y \rightarrow z$ y por tanto la dependencia transitiva $x \rightarrow z$, en este caso el proceso de descomposición se debe de realizar de la siguiente manera:

- De la relación original R se elimina el atributo que origina la dependencia funcional transitiva con la clave primaria de la relación
- Se genera una nueva relación R1 (y,z)
- En la relación original se define a y como clave ajena de la relación R1 Como antes los atributos x,y,z pueden ser compuestos.

Descomposición por Aplicación de FNBC

Dada una relación R con determinantes funcioales x,y,....z algunos de los cuales pueden ser compuestos. Sin pérdida de generalidad supóngase que el determinante compuesto por los atributos T= {a,b,c,...j} determina al atributo n de R. Bajo estos supuestos la descomposición sin pérdidas se realiza de la siguiente manera :

- De la relación R se elimina el atributo n
- Se construye una nueva relación R1, con los siguientes atributos: El atributo n como atributo no primo de la nueva relación, el conjunto de atributos M $\subset$ T que determinan a n, en este caso los componentes de M serán la llave primaria de la relación.
- Se definen a los componentes de M en la relación R como llave ajena de la llave primaria en R1

Este proceso se debe de realizar para cada uno de los casos de dependencias funcionales no completas en R, incluidos por supuesto los casos de traslape de claves candidatas en la relación.

En resumen podemos indicar que si una clave candidata compuesta es un determinante no completo de algún otro atributo de la relación R se viola 2FN y FNBC, si el atributo o los atributos que derminan a otro de R no conforman una clave candidata se viola 3FN y FNBC, Finalmente si existen claves candidatas compuestas traslapadas en que sub conjuntos de una determinen a un sub conjunto de la otra en general se tratará de un caso FNBC puro.
Por lo antes indicado una problemática de Normalización de una relación la podemos abordar de dos maneras en función de las formas normales antes vistas :

- Normalizar la relación en este orden: 1FN, 2FN, 3FN y FNBC,utilizando en cada caso los siguientes conceptos:
    - 1FN : Atomicidad
    - 2FN : Dependencias Funcionales
    - 3FN : Dependencias Funcionales Transitivas
    - FNBC : Traslape de Claves Candidatas

  alternativamente:

- Normalizar la relación en este orden 1FN,FNBC, utilizando en cada caso los siguientes conceptos:
    - 1FN: Atomicidad
    - FNBC: Concepto de Determinante y Traslape de claves Candidatas

Otros tipos de Dependencias
Hasta el ahora las dependencias que pueden estar presentes en una relación R son las dependencias funcionales. Sin embargo se pueden considerar otros tipos de dependencias funcionales en que un atributo de x de una relación R no tenga que determinar un único valor de otro atributo y de la relación sino un conjunto de posibles valores los cuales a su vez puedan ser determinados o no por valores de otro atributo z de la relación.
La existencia en una relación R de este tipo de dependencias ocasionan problemas de redundancia y de manipulación de la información, por lo que se hace necesario aplicar determinadas reglas que eliminen estas depedencias. Este será el objetivo de las reglas de normalización 4FN y 5FN.
Si bien estos tipos de dependencias pueden existir entre cualquiera de los atributos de una relación R, es más usual que estén presentes cuando los atributos implicados forman parte de la llave primaria de la relación. La razón de esto es porque de no ser así la problemática se podría resolver con las reglas vistas anteriormente.

7.5.6.1.7 La Cuarta Forma Normal

Las reglas de Normalización 4FN están basadas en la eliminación de las relaciones de las llamadas **Dependencias Multivaluadas (DMV).**

Concepto:

Dada una relación R, se dice que el atributo y de la relación R depende de forma multivaluada de otro atributo x de R o lo que es lo mismo x multidetermina a y ssi cada valor de x tiene asignado un conjunto bien definido de valores de y y este conjunto es independiente de cualquier valor que tome otro atributo z de R, el cual depende del valor de x.

Representación :

La Dependencia multivaluada se representa por $x \rightarrow\rightarrow y/z$

Ejemplo1:

Sea la tabla R con los atributos {x, y} y la siguiente instancia :

R

x	**y**
a	b
a	c
a	d
a	f
b	w
b	c
z	u
z	k
z	r

En este caso:

a determina los valores b,c,d,f
b determina los valores w,c
z determina los valores u,k,,r

Este ejemplo no se ajusta a la definición de dependencia multivaluada ya que aunque cada valor del atributo x multidetermina valores en y, la relación solo tiene 2 atributos.

En este caso la llave primaria es **x+y,** y la relación está normalizada.

Ejemplo2:
La relación anterior con un atributo adicional z

R1

x	y	z
a	b	1
a	b	2
a	d	1
a	d	2
b	w	8
b	w	9
z	u	0
z	u	6
z	r	7

Como se hizo antes la llave primaria de R1 es **x+y+z** pero en este caso por la existencia del atributo z habrá problemas de borrado e inserción. supóngase que se desea insertar x=a, y=e, entonces se t iene que agregar las tuplas (a,e,1) y (a,e,2), igualmente ocurre con la operación de borrado, si se borra la ocurrencia X=b, Y=w se tienen que borrar las tuplas (b,w,8) y (b,w,9), eliminándose el hecho de que b determina a al conjunto de valores {8,9} lo cual puede no ser deseable ya que los atributos **y** y **z** son independientes.

Ejemplo de Violación de la cuarta forma Normal
Supóngase que estamos interesados en darle seguimiento a las actividades específicas de determinado personal. Por ejemplo el empleado con código 10123 puede hacer trabajo voluntario en la Cruz Roja y en Los Bomberos.
Por otra parte el empleado está asignado a tres proyecto : El proyecto 1, el proyecto 5, y el proyecto 12 los cuales son independientes del trabajo voluntario que realiza en la cruz Roja y en los Bomberos, lo cual podemos resumir en la siguiente tabla:

E_Num	T_Voluntario	Proyectos
10123	Cruz Roja	1
10123	Cruz Roja	5
10123	Bomberos	12

Figura 7.69 Relación con Dependencias Multivaluadas

Como se puede observar cada una de la tabla se encuentra en FNBC siendo la llave primaria : E_Num + T_Voluntario + Proyectos

Solución

En este caso tenemos un problema de dependencias multivaluadas ya que **E_Num** multi determina a **T_Voluntario** y También a **Proyectos** donde **T_Voluntario** y **Proyectos** son independientes, es decir :

$$E\_Num \rightarrow\rightarrow T\_Voluntario/Proyectos$$

Solución a la Problemática de la Cuarta Forma Normal

El Problema causado por las dependencias, multi valuadas se resuelve creando dos tablas, una tabla por cada uno de los atributos que dependen de forma multivaluada. El conjunto de tablas normalizadas quedaría de la siguiente manera:

E_Num	T_Voluntario
10123	Cruz Roja
10123	Bomberos

E_Num	Proyectos
10123	1
10123	5
10123	12

Figura 7.70 Tablas en cuarta forma normal

En este caso las llaves primarias son : **E_Num + T_Voluntario** y **E_Num + Proyectos** respectivamente.

Ejemplo2: Normalizar la Relación: Asignatura_Profesor_Texto dada por:

Asignatura	Profesor	Texto
Física I	A.Peralta	Física Introductoria
Física I	A.Perlata	Principios de Mecánica
Física I	N.Genie	Física Moderna
Física I	N.Genie	Principios de Dinámica
Programación I	H.Ruiz	Programación en C
Programación I	H.Ruiz	Programando en C/C++
Programación I	Ana María Salgado	El lenguaje C

Igual que antes la relación se encuentra en 1FN y en FNBC pero presenta problemas de 4FN ya que $Asignatura \rightarrow\rightarrow Profesor/Texto$, asumiéndose que Profesor y Texto son independientes.

Relaciones Normalizadas:

Asignatura	Texto
Física I	Física Introductoria
Física I	Principios de Mecánica
Física I	Física Moderna
Física I	Principios de Dinámica
Programación I	Programación en C
Programación I	Programando en C/C++
Programación I	El lenguaje C

Asignatura	Profesor
Física I	A.Peralta
Física I	N.Genie
Programación I	H.Ruiz
Programación I	Ana María Salgado

Figura 7.71 Relaciones sin dependencias multivaluadas

7.5.6.1.8 La Quinta Forma Normal

Hasta ahora la única operación para resolver las problemáticas de las diferentes formas normales ha sido la sustitución de la relación original por 2 relaciones o proyecciones estando implícito que esta sustitución no genera pérdidas es decir que por medio de las operaciones del álgebra relacional podemos obtener de nuevo la relación original. Sin embargo existen relaciones imposibles de descomponer sin pérdidas en dos proyecciones pero sí en tres o mas, Ejemplo Consideremos la relación SPJ dada por:

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

Figura 7.72 Tabla Original SPJ

La clave primaria de esta relación es **S# + P# + J#** por lo que podemos afirmar que se encuentra en FNBC, además **S#** no multi determina a **P#** por lo que no existen dependencias multivaluadas con lo cual podemos afirmar que la relación está en 4FN también.

Esta relación tiene una particularidad, no se puede descomponer en 2 tablas. Sea la descomposición : SP, SJ (ver figura 7.104, abajo)

SP

S#	P#
S1	P1
S1	P2
S2	P1

SJ

S#	J#
S1	J2
S1	J1
S2	J1

Figura 7.73 Descomposición en 2 tablas de la relación SPJ

SP |x| SJ =

S#	P#	J#
S1	P1	J2
S1	P1	J1
S1	P2	J2
S1	P2	J1
S2	P1	J1

Figura 7.74 Unión de las relaciones SP,SJ

PJ

P#	**J#**
P1	J2
P2	J1
P1	J1

Figura 7.75 Proyección PJ

Como puede observarse con esta descomposición no se reproduce la Tabla original pues la tupla (S1,P2,J2) está en SP |x| SJ pero no en la relación original: SPJ, como puede comprobarse en la figura 7.101 arriba. Puede comprobarse fácilmente que algo similar ocurrirá si se realiza la descomposición SP y PJ.
Por otra parte comprobaremos que existe una descomposición de SPJ en tres relaciones sin pérdidas, esta descomposición es: SP,SJ y PJ.

P#	**J#**
P1	J2
P2	J1
P1	J1

Figura 7.76 Proyección PJ

Dado que ya se ha obtenido SP |x| SJ, obtendremos la proyección PJ (ver figura 7.76), de modo que:

SP |x| SJ |x| PJ=

S#	P#	J#
S1	P1	J2
S1	P1	J1
S1	P2	J1
S2	P1	J1

Figura 7.77 Unión de las tablas SP,SJ,PJ(Sin Pérdidas)

Es precisamente la relación original SPJ

Es de notar que este último producto natural se efectuó respecto a los atributos comunes P# y J#

Dependencias de Reunión

Ya hemos estudiado como cada una de las formas normales está vinculada a un determinado concepto así 1FN vinculada con el conceptos de atributos atómicos, 2F2 con el concepto de dependencias funcionales, 3FN con dependencias transitivas, FNBC con el concepto de determinante y 4FN con el concepto de dependencias mutivaluadas. La violación de la quinta forma normal estará estrechamente ligada al concepto de **Dependencia de Reunión.**

Definición: Se dice que una relación R satisface la dependencia de reunión sobre R1,R2,....Rn, denotándose *(R1,R2....Rn) si y solo si la relación R es igual a la reunión de sus proyecciones sobre R1, R2,...Rn.

En particular se dice que *(R1,R2,..Rn) es trivial si algún Ri es igual a R

Problemas Generados por algunas dependencias de Reunión

Sea R una relación con atributos: A, B y C entonces R satiface la dependencia de reunión *((A,B),(A,C)) si y solo si satisface la dependencia multivaluada $A \rightarrow\rightarrow B/C$ (Violación de cuarta forma normal)

Por otra parte si la relación cumple la dependencia de reunión: *((A,B),(B,C),(C,A)) implica la siguiente problemática: Si en el transcurso del tiempo aparecen las tuplas (a1,b1,c),(a1,b,c1) y (a,b1,c1) también aparecerá la tupla (a1,b1,c1), este hecho asimismo caracteriza a una dependencia de reunión del tipo indicado arriba.

Ejemplo

Sea la siguiente relación :

Lector#	Libro#	Proyectos#
Lector1	Lib1	Proy2
Lector1	Lib2	Proy1
Lector1	Lib1	Proy1
Lector2	Lib1	Proy1

Figura 7.78 Relación Proyectos

En este caso La relación Proyectos(figura 7.104) cumple la dependencia de reunión

*((Lector#, Libro#),(Libro#, Proyectos#),(Proyectos#, Lector#))

por lo que :

Si **Lector1** ha sacado el libro **Lib1**, El proyecto **Proy1** utiliza el libro **Lib1**

Y el **Lector1** aporta libros al proyecto,entonces **Lector1** aporta el libro **Lib1** al proyecto **Proy1.**

De modo que si una relación cumple una dependencia de reunión, pueden existir problemas en determinadas inserciones o borrados. En el caso del ejemplo anterior si las dos últimas tuplas de la tabla no existieran e insertáramos (Lector2,Lib1,Pr1),automaticamente habría que introducir(Lector1,Lib1,Proy1) y por otra parte si eliminamos esta última hará que borrar la anterior.

Definición de la Quinta Forma Normal
Una relación está en 5NF Si y solo si toda dependencia de reunión es trivial o es consecuencia de sus superclaves

Que sea consecuencia de sus superclaves significa que todos los componentes de la dependencia de reunión son super claves de R.

Ejemplo:

Sea la relación Lector(lec#,Nombre,Ciudad,Teléfono) bajo el supuesto que no existen dos lectores con el mismo nombre por lo cual Nombre es una llave candidata.
En esta relación podemos distinguir las siguientes dependencias de reunión:

*((lec#,Nombre,Teléfono),(lec#,Ciudad))

*((lec#,Nombre), (lec#,Teléfono),(Nombre,Ciudad))

No son triviales sin embargo los componetes de ambas son superclaves por tanto la relación Lector está en 5FN
Se puede notar que la relación **Proyectos** (Figura 7.104) no está en 5FN, ya que la única llave candidata es la copuesta por los tres atributos **Lector# + Libro# + Proyectos#** y la dependencia de reunión
*((Lector#, Libro**#),(** Libro#, Proyectos**#),(** Proyectos#, Lector**#))** no es una consecuencia de sus superclaves.
Para normalizar la relación se descompone en las relaciones sugeridas en la dependencia de reunión es decir:
Pro1(Lector#, Libro**#),** Pro2**(** Libro#, Proyectos**#)** y Pro3(Proyectos#, Lector**#)**

7.6 Tema 5: "El Lenguaje SQL (Structured Query Language)"

TEMA N° 5

OBJETIVOS:

- Conocer el origen y la Evolución del Lenguaje SQL
- Saber Traducir Las Operaciones Básicas del álgebra relacional al lenguaje SQL
- Saber aplicar la Operación renombramiento
- Saber definir y conocer la importancia de las variables de tupla en una expresión SQL
- Saber aplicar búsquedas utilizando el operador like de la cláusula Where de SQL
- Generar consultas ordenadas por uno ó más atributos
- Saber Aplicar consultas relacionadas con operaciones de conjunto
- Saber Diseñar consultas en SQL que involucren Agrupaciones y funciones de Agregación
- Saber resolver problemáticas que involucren valores nulos en determinados atributos
- Saber aplicar consultas anidadas en otra consulta principal referidas a la pertenecia de conjuntos
- Saber comparar conjuntos generados por una consulta principal y una sub consulta
- Saber Distinguir La diferencia del uso de la cláusula some y la cláusula all
- Saber diseñar consulta que involucren la cláusula exists
- Saber determinar el alcance de variables de tupla al utilizar sub consultas
- Saber Diseñar Vistas
- Saber Diseñar Cláusulas complejas utilizando la Cláusula With
- Conocer y saber aplicar consultas en SQL para modificar la Base de Datos
- Conocer las diferentes anomalías que pueden presentarse en una consulta al utilizar el comando Create View
- Saber implementar Consultas que implementen transacciones en SQL
- Saber aplicar los diferentes comandos relacionados con las reuniones externas en SQL
- Saber Traducir un Esquema relacional a Comandos SQL
- Conocer los Diferentes tios de datos utilizados por ORACLE
- Saber Implementar Consultas del DDL de SQL que modifiquen el Esquema Relacional
- Conocer la necesidad de implementar Índices en una Base de datos
- Saber generar consultas para la creación ó eliminación de índices en una base de Datos

CONTENIDO

- El Lenguaje SQL
  - Introducción
  - La Operación Select
    - Implementación de la Proyección y la Selección
    - Implementación del Producto cartesiano
    - Implementación del Producto Natural
  - La operación Renombramiento
  - Las Variables de Tupla
  - El operador Like
    - El Carácter reservado %
    - El carácter reservado _
    - Exepciones
  - La Cláusula Order By
    - Los parámetros Asc y Desc
  - Las Operaciones de Conjunto
    - La Operación Unión
    - La Operación Intersección
    - La Operación Diferencia
  - Funciones de Agregación
    - La Cláusula Group By y las funciones de Agregación
    - Uso de las cláusulas Where y having con funciones de Agregación
  - SQL y los valores nulos
    - Algunas soluciones a las problemáticas de valores nulos y las funciones de agregación
  - Sub Consultas Anidadas

    - Concepto
    - Sub Consulta de Pertenencia a Conjuntos (In,not in)
- Comparación de Conjuntos
    - Las Cláusulas Some y All
    - Uso de las funciones de Agregación con Some y All
    - Comprobación de relaciones vacías (Exists,Not exists)
    - Aspectos particulares de la cláusula not exists
    - Alcance de las variables de tupla en las sub consultas
    - Comprobación de tuplas duplicadas
- Vistas en SQL
    - Definición
    - Consultas Complejas
    - Relaciones Derivadas
    - La Cláusula With
    - Uso de Múltiples With en una Expresión SQL
- Modificación de la base de Datos
    - Borrado
    - Uso de Funciones de Agregación en una Sub Consulta de borrado
    - Inserción de tuplas en una relación
    - Actualización de las tuplas en una relación
- Anomalías al Utilizar Vistas
- Las Transacciones en SQL
- Mas sobre reuniones de relaciones
    - La Cláusula Inner Join
    - La Cláusula Left Outer join
    - La Cláusula Right Outer join
    - La Cláusula Full Outer join
- Definición de Esquemas relacionales en SQL
    - El Comando Create table
- Cláusulas Opcionales utilizadas en Oracle
- Tipos de Datos Pre Definidos por ORACLE/SQL
- Otros Comandos del DDL de SQL
    - Los Comandos:
        - Drop domain
        - Drop constraint
        - Drop Table
- Modificación del Esquema Relacional
    - El comando Alter Table
- Índices
    - El Comando Create index
    - El Comando Drop Index
- Funciones Definidas en Oracle

DURACIÓN: 12 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill
- Adoración de Miguel, Mario Piattini. Fundamentos y Modelos de Base de Datos. 2ª Edición. RAMA

7.6.1 Introducción

Fue la empresa IBM (International Business Machine) quien desarrolló la versión original de este lenguaje en el San José Research Center, originalmente se denominó Sequel, como parte del proyecto System R a comienzos de 1970. el lenguaje sequel ha evolucionado desde entonces y ha pasado a llamarse SQL(Structured Query Language) convirtiéndose en el lenguaje estándar para Bases de Datos relacionales.

La evolución en el tiempo de este lenguaje ha sido la siguiente:

En 1986, los organismos ANSI (American National Standars Institute) e ISO (International Standars Organization) publicaron una norma para este lenguaje que se denominó SQL-86.En 1987 IBM publicó su propia norma de SQL corporativo denominada SAA –SQL (System Application Arquitecture Database Interface).En 1989 se publicó una norma extendida para SQL denominada SQL-89, actualmente los sistemas de bases de datos son compatibles con esta norma. La siguiente versión de la norma fue SQL-92, siendo la más reciente la denominada SQL:1999.

En este apartado se presentará una visión general de SQL-92 y algunas características de SQL:1999.

Los componentes de este lenguaje son : Un lenguaje de Definición de Datos, que incluye la definición de vistas, un lenguaje de Manipulación de Datos basado en el álgebra relacional, un módulo para el control de transacciones, SQL Dinámico e Incorporado para definir la forma como pueden incorporarse las instrucciones SQL en lenguajes de propósito general, también incluye comandos para la especificación de las restricciones de integridad que deben satisfacer los datos en la Base de Datos y Ordenes para especificar los derechos de acceso a las relaciones y vistas.

7.6.2 La operación Select

Esta operación de SQL es una de las más utilizada y obedece a la implementación de las operaciones básicas del álgebra relacional :Proyección, Selección, producto cartesiano, producto natural y operaciones del álgebra relacional extendida.

7.6.2.1 Implementación de la Proyección y la Selección

$\Pi_S \sigma_{cond}(R)$ donde $S = \{ s_1, s_2, s_3, ... s_m \} \subseteq T = \{ t_1, t_1, t_1, ... t_n \}$ es el conjunto de atributos de R y cond es una condición lógica en función de los atributos de R, entonces:

$\Pi_S \sigma_{cond}(R)$= Select $s_1, s_2, s_3, ... s_m$ From R Where Cond, como casos particulares en que la expresión del álgebra relacional se refiere solo a la proyección o solo a la Selección, las respectivas expresiones son las siguientes:

$\Pi_S(R)$= Select $s_1, s_2, s_3, ... s_m$ From R

$\sigma_{cond}(R)$= Select * From R Where Cond, donde * indica que se incluyen a todos los atributos de la relación R.

7.6.2.2 Implementación del Producto Cartesiano

Sean A, B dos relaciones, entonces A x B $\equiv$ Select * from A,B donde * significa en este caso todos los atributos de A y B.

7.6.2.3 Implementación del Producto natural

Sean A, B dos relaciones y sea t el atributo común de
A y B, entonces
A |x| B ≡ Select * From A,B Where A.t=B.t, donde como antes * significa todos los atributos de A y B.

Ejemplos

Los siguientes ejemplos están basados en la base de datos definida en la figura 9.5.3 referente a una entidad Bancaria.
usando SQL obtener :

a) Obtener los nombres de todas las sucursales en la relación *préstamo*

Solución

```
Select Nombre_Sucursal
    From Préstamo
```

Nota: el comando select permite eliminar datos duplicados en un determinado atributo, así en el presente ejemplo si se desea que los Nombres de las Sucursales aparezcan solo una vez se debe utilizar la cláusula Distinct, con lo cual el cmando solicitado quedaría así:

```
Select distinct Nombre_Sucursal
    From Préstamo
```

b) Obtener el nombre de la Sucursal, el número de cada préstamo y el importe del mismo incrementado en 10%.

Solución

En este caso se solicita la generación de un campo calculado,el cual se va a obtener mediante la expresión: Importe*1.01, como vimos en el álgebra relacional extendida a este campo se le puede dar un nombre de manera opcional mediante la cláusula as. El resultado de la expresión SQL es :

```
Select distinct Nombre_Sucursal, Número_Préstamo,
        Importe*1.01 as Nuevo_importe
    From Préstamo
```

c) Obtener todos los números de Préstamos, para préstamos efectuados en la sucursal “Linda Vista”, en los que el importe sea superior a C$ 5,000

Solución

En este caso se mostrará como utilizar la cláusula Where, dado que la consulta contiene una restricción o condición que deben de cumplir las filas resultantes, esta es : Importe superior a C$ 5,000. La expresión SQL es:

```
Select Número_Préstamo
    From Préstamo
    Where
        Nombre_Sucursal=”Linda Vista” and
        Importe>5000
```

Nota: Los predicados lógicos de las condiciones en la cláusula where usan los conectivos $\wedge$, $\vee$, $\neg$, los cuales en SQL son and, or y not respectivamente, por otra parte los conectores de comparación son <, <=, >, = y <> (menor,menor o igual,mayor,igual y diferente).

d) Obtener los números de préstamos cuyo importe esté entre C$ 20,000 y C$ 50,000 (Se incluyen los valores extremos).

Solución

```
Select Número_Préstamo
        From Préstamo
        Where Importe <= 50000 and Importe >= 20000
```

Nota: SQL cuenta con una función denominada between, la cual facilita la utilización de expresiones lógicas como la del ejemplo anterior. La expresión de SQL del ejemplo anterior usando la función lógica between es:

```
Select Número_Préstamo
        From Préstamo
        Where Importe between 20000 and 50000.
```

d) Obtener el Nombre de los clientes con préstamo, el número del préstamo, y el importe de cada uno de los préstamos.

Solución

Se debe notar que los atributos solicitados no pertenecen a una sola tabla como en los ejemplos pasados(ver el Esquema relacional) ya que el número del préstamo (Número_préstamo) y el Importe pertenecen a préstamo mientras que Nombre de los clientes con préstamo se encuentran en la tabla prestatario. Notemos además que préstamo es un conjunto entidad fuerte y prestatario es un conjunto de relaciones entonces para unir la información de ambas relaciones debemos de implementar la operación producto natural entre ambas relaciones, por lo que la expresión SQL queda de la siguiente manera:

```
Select prestatario.Nombre_Cliente, préstamo.número_préstamo,
                préstamo.importe
        From prestatario,préstamo
        Where Préstamo.Número_préstamo=
                Prestatario. Número_préstamo
```

e) Obtener la consulta del inciso anterior con la restricción de que los clientes sean de la sucursal “Linda Vista”.

Solución

La lógica de la solución es similar al del inciso anterior, haciendo énfasis en el hecho de que en la cláusula where primero se debe efectuar la unión de las tablas involucradas y después las restricciones de filas, el comando SQL queda de la siguiente manera en este caso:

```
Select prestatario.Nombre_Cliente, préstamo.número_préstamo,
        préstamo.importe
    From prestatario,préstamo
    Where
        Préstamo.Número_préstamo=
        Prestatario. Número_préstamo
      and
        préstamo.Nombre_Sucursal=”Linda Vista”
```

7.6.2.4 La Operación Renombramiento

La operación de renombramiento se puede utilizar tanto para relaciones como para atributos mediante la cláusula as, así si se desea renombrar el atributo préstamo.número_préstamo como id_préstamo, la nueva expresión tendría la forma:

```
Select prestatario.Nombre_Cliente, préstamo.número_préstamo
        as id_préstamo, préstamo.importe
    From prestatario,préstamo
    Where Préstamo.Número_préstamo=
        Prestatario. Número_préstamo
```

Las Variables de Tupla

La cláusula as es de particular utilidad si se desean definir variables de tupla en una expresión SQL, en este caso las variables se definen en la clausula from
Ejemplo (a)

```
Select Nombre_Cliente,T.número_préstamo,S.importe
    From prestatario as T,préstamo préstamo as S
    Where T.número_préstamo=S.número_préstamo
```

En este caso la variable de tupla T representa filas de prestatario, mientras que la variable de tupla S representa filas de préstamo

Ejemplo(b) Obtener los nombres de las sucursales cuyos activos son mayores que alguna de las sucursales localizadas en Barcelona

Solución

```
Select distinct T.Nombre_Sucursal
    From sucursal as T,sucursal as S
    Where T.activo>S.activo
      and
        S.ciudad_Sucursal=’Barcelona’
```

En este caso se han definido dos variables de tupla T, S (con dominios diferentes) para una sola relación: sucursal, en este caso es Imprescindible la utilización de la cláusula as. Nótese que T representa todas las tuplas de sucursal, mientras que S solamente aquellas de sucursal que cumplen la condición ciudad_Sucursal=’Barcelona’.

7.6.2.5 El Operador Like

Este operador se utiliza en la cláusula Where de SQL para comparar patrones de carácteres para lo cual se utilizan los caracteres % y _

El carácter % :
Significa cualquier sub cadena dentro del patrón de comparación así: 'ma%' significa que la condición de comparación al utilizar like es verdadera para cadenas como 'managua', 'masaya'. En cambio, para esta cadena, valores como ‘nacional’, ‘m’, ‘agua’, no serían válidos porque no cumplen con la parte fija del patrón.

El carácter _ :
Representa cualquier carácter dentro del patrón de comparación así: 'ma_ay_' significa que la condición de comparación al utilizar like es verdadera para cadenas como, 'masaya', 'malaya' o, 'mapaye'.

Nota: Ambos caracteres se pueden utilizar en una sola expresión lógica que involucre a la cláusula like.

Ejemplos

a) Obtener el nombre de todos los clientes que viven en una calle cuyo nombre termine con la sub cadena 'Mayor'

Solución

```
Select Nombre_Cliente
        From Cliente
        Where Calle_Cliente Like '%Mayor'
```

b) Obtener el nombre de todos los clientes que viven en una calle cuyo nombre comienza y termina con 'a' y que contenga la sub cadena 'Darío'

Solución

```
Select Nombre_Cliente
        From Cliente
        Where Calle_Cliente Like 'a%Darío%a'
```

Exepciones
En algunos casos se requiere utilizar a % y _ como parte del patrón de comparación y no como símbolos especiales dentro de la cláusula like en este caso se utiliza el símbolo \ para eliminarles la condición de caracteres especiales, así like 'ab\%cd%' compara con todas las cadenas que comienzan con ab%cd y like 'ab\\cd%' compara con todas las cadenas que comienzan con ab\cd en este último caso el mismo símbolo \ se ha eliminado su condición de carácter especial.

7.6.2.6 La cláusula Order By

Se utiliza dentro de una expresión SQL para ordenar la consulta con respecto a uno ó un grupo de campos, en general se ubica al final de la expresión. Así la siguiente expresión SQL:

```
Select prestatario.Nombre_Cliente, préstamo.número_préstamo,
        préstamo.importe
From prestatario,préstamo
Where  Préstamo.Número_préstamo=
        Prestatario. Número_préstamo
  And  préstamo.Nombre_Sucursal=”Linda Vista”
Order by prestatario.Nombre_Cliente
```

obtiene los nombres de los clientes de la Sucursal =”Linda Vista” (ordenados en forma ascendente) que han efectuado préstamos, el código de cada préstamo y el importe del mismo.

Las cláusulas asc y desc

La Cláusula asc y desc se utilizan como complemento de la cláusula Order By para especificar el tipo de ordenamiento deseado en la consulta:ascendente ó descendente. Si el ordenamiento involucra más de un atributo se asume que el primero obedecerá a un ordenamiento principal y el resto de atributos tendrán un ordenamiento secundario.

Ejemplo:

Obtener todos los atributos de la relación préstamo, ordenados por el atributo Importe de manera descendente y por el atributo número_Préstamo de manera ascendente

Solución

```
Select * from préstamo Order By Importe desc, número_Préstamo asc
```

Sea la siguiente instancia de la relación préstamo:

Número_Préstamo	Nombre_Sucursal	Importe
100	Linda Vista	1500
110	La Colonia	1600
115	Las Brisas	1200
120	El Convento	1500
125	La Ceiba	1200
130	Linda Vista	1600
135	La Colonia	1600
140	La Colonia	1200

Entonces el resultado de la consulta aplicado a esta instancia es:

Número_Préstamo	Nombre_Sucursal	Importe
115	Las Brisas	1200
125	La Ceiba	1200
140	La Colonia	1200
100	Linda Vista	1500
120	El Convento	1500
110	La Colonia	1600
130	Linda Vista	1600
135	La Colonia	1600

Figura 7.79 Salida de consulta SQL con Ordenamiento

Nota: En este caso el ordenamiento principal se efectuará en el atributo Importe de forma descendente y el ordenamiento secundario en el atributo número_Préstamo de forma ascendente.

7.6.2.7 Las Operaciones de Conjunto: Unión, Intersección y Diferencia.

La Operación Unión
Forma General: Sean R1 y R2 dos relaciones. Sean $r1_1$, $r1_2$, $r1_3$,... $r1_m$ atributos de R1 y $r2_1$, $r2_2$,... $r2_m$ atributos de R2, entonces:

Select $r1_1$, $r1_2$, $r1_3$,... $r1_m$ from R1
Union
Select $r2_1$, $r2_2$,... $r2_m$ from R2

genera la unión de conjunto de las filas de la relación
Select $r1_1$, $r1_2$, $r1_3$,... $r1_m$ from R1 con la relación Select $r2_1$, $r2_2$,... $r2_m$ from R2

Ejemplo: Obtener el Nombre de todos los clientes que tienen una cuenta en el banco así también los nombres de los que tengan un préstamo.

Solución

En este caso se requiere unir los nombres de los clientes en Impositor(Los que tienen cuenta en el banco) con los nombres de los clientes de prestatario(Los que tienen un préstamo). Por tanto la expresión SQL es la siguiente:

```
Select Nombre_Cliente From Impositor
Union
Select Nombre_Cliente From Prestatario
```

Conservación de duplicados
Dado que la operación unión es una operación de conjunto por defecto no presenta valores duplicados por lo que si se desea conservarlos se debe utilizar la cláusula **union all** en lugar de union, así en el ejemplo anterior, la expresión SQL quedaría de la siguiente manera:

```
Select Nombre_Cliente
From impositor
Union all
Select Nombre_Cliente
From prestatario
```

La operación Intersección.
Esta operación obtiene las tuplas comunes de dos relaciones.

Forma General
Sean R1 y R2 dos Relaciones, sean $R1_1$, $R1_2$... $R1_m$ atributos de R1 y $R2_1$, $R2_2$, $R2_3$... $R2_m$ atributos de R2 entonces la expresión SQL para la intersección es:

Select $R1_1$, $R1_2$… $R1_m$ from R1 Where P1

Intersect

Select $R2_1$, $R2_2$… $R2_m$ from R2 Where P2

- Donde P1 y P2 son expresiones lógicas sobre los atributos de R1 y R2 respectivamente.

Ejemplo:
Obtener todos clientes que tienen tanto un préstamo como una cuenta en el Banco.

Solución

Select distint Nombre_Cliente from Impositor

Intersect

Select distint Nombre_cliente from prestatario

- Como en el caso de la operación unión, la operación de intersección elimina los duplicados por lo que como antes si estos se desean en la consulta se debe de utilizar la cláusula all es decir intersect all en lugar de intersect

La operación excepto.
Esta operación representa a la operación de conjunto menos (-) del álgebra relacional.

Forma General
Sean R1, R2 dos relaciones y sean $R1_1$, $R1_2$... $R1_m$ y $R2_1$, $R2_2$,... $R2_m$ atributos de R1 y R2 respectivamente, entonces la expresión SQL para esta operación es:

Select $R1_1$, $R1_2$… $R1_m$ from R1 Where P1

Except

Select $R2_1$, $R2_2$,... $R2_m$ from R2 Where P2

- Donde P1 y P2 son expresiones lógicas sobre los atributos de R1 y R2 respectivamente. Esta expresión obtiene las tuplas generadas por la expresión
  Select $R1_1$, $R1_2$…$R1_m$ from R1 Where P1 menos las tuplas generadas por la expresión: Select $R2_1$, $R2_2$... $R2_m$ from R2 Where P2.
  Dado que esta operación es también una operación de conjunto, elimina los duplicados por lo que si estos se desean en la consulta se debe utilizar excepto all en lugar de except.

Ejemplo:
Obtener todos los clientes que tienen una cuenta en el banco pero que no tienen un préstamo.

Solución

Select distint Nombre_Cliente from Impositor

Except

Select distint Nombre_Cliente from Prestatario

7.6.2.8 Funciones de Agregación.

Estas funciones ya fueron estudiadas anteriormente como parte del álgebra relacional extendida, se trata ahora de estudiar la implementación de éstas en SQL. Las funciones de agregación son funciones que toman una colección de valores como entrada y producen un único valor como salida. Las funciones primitivas de agregación son:

El promedio (Avg), El Mínimo (Min), El Máximo (Max), la Suma (Sum) y la Cuenta (Count).

Los valores de entrada a Sum y Avg debe ser una colección de números pero el resto de funciones pueden operar también sobre cadenas.

Ejemplo:
Obtener la media de los saldos de las cuentas de la sucursal “Linda Vista”

Solución

- El conjunto de los Datos sobre los cuales actuará la función de agregación es el determinado por el atributo Saldo de la relación cuenta, la expresión SQL es la siguiente:

```
Select Avg (saldo)
From cuenta
Where Nombre_Sucursal = 'Linda Vista'
```

- La cláusula Group by y las funciones de Agregación.
  Existen situaciones en que es deseable aplicar las funciones de Agregación a determinadas agrupaciones de un atributo o conjunto de atributos, estas agrupaciones se logran utilizando la cláusula **group by**

Ejemplo:
Dada la relación Cuenta indicada abajo, obtener el promedio del saldo de las cuentas agrupadas por los nombres de las sucursales

Cuenta

Número_Cuenta	Nombre_Sucursal	Saldo
C-100	Linda Vista	10.500
C-125	Sutiava	3,900
C-150	Granada	6,200
C-230	Sutiava	8,200
C-621	Granada	6,500
C-348	Linda Vista	4,500
C-290	Granada	9,200
C-98	Linda Vista	12,000
C-25	Sutiava	5,200

Solución

```
Select Nombre_Sucursal, Avg (saldo)
From cuenta
Group by Nombre_Sucursal
```

Relación resultante

Nombre_Sucursal	Avg (Saldo)
Granada	7300
Linda Vista	9000
Sutiava	5766.67

Figura 7.80 Salida : agrupación y funciones de Agregación

- De forma opcional, como siempre podemos renombrar el rótulo de alguna de las salidas usando la cláusula **as.** Así si se desea variar el Nombre por defecto Avg (saldo) Por promedio_saldo, la expresión SQL sería la siguiente:

```
Select Nombre_Sucursal, Avg (saldo) as Promedio_Saldo
From cuenta
Group by Nombre_Sucursal
```

Note que el atributo ó los atributos que servirán de base para la agrupación deben aparecer tanto en la cláusula Select como en la cláusula group by.

Uso de la cláusulas Where y Having con funciones de Agregación.
La cláusula Where en este contexto se utiliza de forma similar a como se utiliza con comandos Select que no utilizan funciones de agregación. Sin embargo se debe tener particular cuidado cuando se requiera realizar filtros sobre una consulta que ha utilizado funciones de agregación con agrupaciones, en este caso no se debe utilizar la cláusula Where si no que la cláusula **having**. En otras palabras si en una misma consulta se desea efectuar un filtro para determinar la relación sobre la cual actuarán las funciones de agregación y otro filtro sobre la relación generada por estas funciones de agregación y la cláusula Group by, entonces primero se utiliza la cláusula Where y posteriormente en la expresión la cláusula **having.**

Ejemplo(a):
Obtener el promedio de los saldos mayores de 6,000 agrupados por Nombre_Sucursal.

Solución

- En este caso solo existe filtro a nivel de la salida agrupada (Promedio de los saldos > 6,000), es decir solo se utilizará la cláusula Having, la expresión SQL es:

```
Select Nombre_Sucursal, Avg (Saldo)
from cuenta
group by Nombre_Sucursal
Having Avg (Saldo) > 6,000
```

La Salida en este caso utilizando la instancia anterior de saldo es:

Nombre_Sucursal	Avg (Saldo)
Granada	7,300
Linda Vista	900

Ejemplo (b):
(Ilustración del uso de Where y Having)
Obtener el promedio del saldo de todos los clientes que viven en Managua y tienen como mínimo tres cuentas.

Solución

Utilizando el esquema relacional en la figura 7.4.7 podemos notar que para dar respuesta a esta consulta se hace necesario unir las siguientes tablas: Saldo $\in$ Cuenta, Ciudad_Cliente $\in$ Cliente, Nombre_Cliente (con cuenta) $\in$ Impositor, por tanto la relación base de la consulta de agrupación es:

$$\sigma_{\text{Cliente.Ciudad\_Cliente='Managua'}} (\text{Impositor lxl Cliente lxl Cuenta})$$

La agrupación se hará en función del atributo Impositor.Nombre_Cliente y el filtro (having) de la consulta de agrupación es 'tener como mínimo tres cuentas', por tanto la expresión SQL es:

```
Select Impositor.Nombre_cliente, Avg (Saldo)
        From Impositor, Cuenta, Cliente
        Where
                Impositor.Número_cuenta = cuenta.Número_cuenta and
                Impositor.Número_cliente = cuenta.Número_cliente and
                Ciudad_cliente = 'Managua'
        Group by Impositor.Nombre_cliente
        Having count (distinct impositor.Número_cuenta) > = 3
```

Nótese como en la cláusula having se pueden utilizar funciones de agregación, lo cual no es posible en la cláusula Where.

7.6.2.9 SQL y valores Nulos.

SQL permite el uso de valores nulos, indicando así la ausencia de información sobre el valor de un atributo.

Para identificar este tipo de valor SQL utiliza la palabra reservada **null**

Ejemplo:
Obtener los números de préstamos en la relación préstamo que no cuentan con datos en el atributo Importe.

Solución

```
Select Número_préstamo from préstamo
Where
Importe is null
```

Nota: Si a contrario de esta consulta, se está interesado en obtener los números de prestamos con datos de Importe, en lugar de is null se utiliza la cláusula in not null

Algunas soluciones a las problemáticas de valores nulos en SQL

(a) a op b = null para op = +, -, *, /
si a = null ó b = null

(b) a comp b = desconocido para comp = (< =,<,> =,>,=,<>)
si a ó b son nulos

Las reglas extendidas para los conectores lógicos and y or son los siguientes:

And	*cierto*	*desconocido*	*Falso*
Cierto	cierto	desconocido	Falso
Desconocido	desconocido	desconocido	Falso
falso	falso	falso	Falso

Or	*cierto*	*desconocido*	*Falso*
Cierto	cierto	cierto	cierto
Desconocido	cierto	desconocido	descono cido
Falso	cierto	desconocido	Falso

Por otra parte SQL para expresiones del tipo Select... from R1, R2,... Rn Where P si el predicado P es falso ó desconocido para un determinado conjunto de tuplas, estas no se añaden al resultado de la consulta.

SQL también permite decidir si el resultado de una operación aritmética ó lógica es desconocida utilizando la cláusula is unkown ó por el contrario se puede utilizar la cláusula is not unknown para indicar que la operación genera un determinado valor que no es desconocido.

Los valores Nulos y las funciones de Agregación.
En general las funciones de agregación tratan a los valores nulos según la siguiente regla:
Todas las funciones de agregación excepto la función count ignoran los valores nulos del conjunto de datos de entrada de la función. En el caso de count si todos los valores del conjunto de datos son nulos entonces count devuelve 0, el resto de funciones devuelven un valor nulo en este caso.

7.6.2.10 Sub consultas Anidadas.
Una sub consulta es una expresión SQL de la forma Select-from-where que está dentro de otra consulta SQL. Un uso común de esta subconsultas es posibilitar al

SQL el poder decidir si una determinada salida se encuentra ó no en un conjunto de valores generados por esta sub consulta, poder comparar una salida con un conjunto de valores ó comparar la cardinalidad de dos conjuntos de datos.

Sub consultas de pertenencia a conjuntos.
En este caso con la cláusula conectiva **in** se comprueba la pertinencia a un conjunto donde este conjunto es generado por la sub consulta, al contrario si se desea generar filas en que el resultado de una condición no se encuentra en un conjunto generado por la sub consulta se utiliza la cláusula **not in**.

(a) Encontrar los nombres de los clientes que tengan un préstamo y que tengan una cuenta.

Solución

```
Select Nombre_cliente from prestatario
Where Nombre_cliente
In
Select Nombre_cliente from Impositor
```

Nota: Este resultado se puede generar, utilizando la cláusula Intersect de SQL.

(b) Obtener los nombres de los clientes que tienen una cuenta y un préstamo en la sucursal 'Linda Vista'

Solución

En este caso la consulta involucra directamente a dos relaciones ó tablas impositor (clientes con cuentas) y prestatario (clientes con prestamos), como puede verse en el diagrama de la figura 7.4.7, es posible acceder al Nombre de la Sucursal desde prestatario y desde Impositor por medio de las relaciones préstamo y cuenta respectivamente, y la expresión SQL es la siguiente:

```
Select distinct Nombre_Cliente
        From prestatario, prestamo
        Where
                prestatario.Número_préstamo =
                Préstamo.Número_préstamo
          and
                Nombre_sucursal = 'Linda Vista'
          and
                (Nombre_sucursal,Nombre_Cliente)
                 in
                (Select Nombre_sucursal, Nombre_cliente
                        from impositor, cuenta
                        where impositor.Número_cuenta
                                = cuenta.Número_cuenta)
```

Ilustración del uso de la cláusula not in

Ejemplo:
Obtener todos los clientes que tienen un préstamo en el banco pero que no tienen una cuenta en el banco.

Solución

```
Select distinct Nombre_Cliente
    From prestatario
    Where Nombre_cliente not in
            (Select Nombre_cliente from Impositor)
```

Mas sobre las cláusulas in y not in
Estos operadores se pueden utilizar también sobre un conjunto finito de valores dados explícitamente.

Ejemplo:
Obtener el Nombre de los clientes del banco, que tienen un préstamo a excepción de 'Gutiérrez' y 'Espinoza'

Solución

```
Select distinct Nombre_Cliente
   From prestatario
    Where Nombre_cliente not in ('Gutiérrez', 'Espinoza')
```

Comparación de conjuntos.
La comparación de conjuntos en SQL, se realiza utilizando una subconsulta dentro de un comando Select vinculadas las dos expresiones Select por la cláusula **some** ó por la cláusula **all**.
El mecanismo de comparación es el siguiente: el conjunto generado por el Select principal se compara elemento a elemento con el conjunto generado por la subconsulta, el atributo ó los atributos involucrados en la comparación se determinan en la cláusula where del Select principal y la forma de comparación depende de que si la cláusula a utilizar es **some** ó **all**.

Efecto de la cláusula some.
Sea x un elemento del conjunto generado por el Select principal sean
{Y1, Y2,...Yn} el conjunto generado por la subconsulta, sea comp una expresión de comparación;
Entonces si existe Yi, i=1...n ∋ x comp Yi es verdadero para algún i entre 1 y n, la tupla donde se encuentra el valor x en la relación generada por el Select principal será parte de la consulta.

Ejemplo:
Sea X = {5,18,10,24,33,1} el conjunto principal de comparación (el generado por el Select principal)
Sea Y= {13,2,1,8,5,4,9} el conjunto definido por la sub consulta, sea comp la expresión $< =$, entonces X $< =$ some Y es verdadero para los siguientes valores

del conjunto X : 5,10,1 pues 5 < 13; 10 < 13; 1 < 13, así mismo no se incluye a 18 por ejemplo pues no existe un elemento en Y tal que 18 sea < = a ese elemento.

Ejemplo:
Obtener los Nombres de las Sucursales cuyos activos sean mayores que alguna de las sucursales ubicadas en 'Masaya'

Solución

```
Select Nombre_Sucursal
        From sucursal
        Where activo > some (select activo From sucursal
                                Where ciudad_sucursal
                                = 'Masaya' )
```

*Nota en algunas implementaciones de SQL, se utiliza la palabra any en lugar de some.

Efecto de la cláusula all.
Esta de forma similar que some (any) compara dos conjuntos definidos de forma similar que en el caso de la cláusula some con la diferencia de que la tupla generada por el select principal será parte de la consulta de salida si la comparación
X comp yi ; i = 1...n es verdadera $\forall$ yi, i = 1... n

Ejemplo:
Sea x = {5,18,10,24,33,1}el conjunto generado por el select principal y sea y={13,2,1,8,5,4,9} el conjunto definido por la subconsulta sea comp la expresión > =, encontrar los elementos de x para los cuales x > = all Y es verdadero.

Solución

Estos son: 18, 24, 33 pues 18 es mayor que todos los elementos del conjunto Y, igual ocurre con 24 y 33 por el contrario la desigualdad no es válida para 5 pues 5 no es mayor que todos los elementos en Y.

Ejemplo:
Obtener el Nombre de las sucursales cuyos activos sean mayores a todas aquellas sucursales ubicadas en Masaya

Solución
En este caso, la expresión SQL es:

```
Select Nombre_Sucursal
        From sucursal
        Where activo > all (select activo From sucursal
                              Where ciudad_sucursal= 'Masaya')
```

Uso de las funciones de agregación con some, all
En este caso se debe tener particular cuidado ya que en SQL no está permitida la composición de funciones de agregación así expresiones como max (avg (...)) no pueden ser utilizadas, veamos el siguiente caso:

Se desea obtener el Nombre de las sucursales que tiene el mayor saldo promedio

Solución

```
Select Nombre_Sucursal
        From cuenta
        Group by Nombre_sucursal
        Having Avg (saldo) ≥ all (select avg (saldo)
                                  From cuenta
                                  Group by Nombre_sucursal)
```

Supóngase la siguiente instancia de la relación cuenta:

Cuenta

Número_Cuenta	Nombre_Sucursal	Saldo
C-2132	Linda Vista	7000
C-2667	Las Flores	10,000
C-3845	Linda Vista	6000
C-9562	Linda Vista	8,000
C-1518	Las Flores	12,000
C-1325	Las Flores	20,000

En este caso los conjuntos a comparar son los mismos: {7,000, 14,000} y {7,000, 14,000} donde 7,000 es el promedio de las cuentas en 'Linda Vista' y 14,000 es el promedio de las cuentas en 'Las Flores'. Por tanto en este caso específico la solución es 14,000.

Comprobación de Relaciones Vacías.
SQL incluye la cláusula exists (not exists) para comprobar si una subconsulta produce ó no produce resultado alguno.
Ejemplo:
Obtener el Nombre de los clientes que tienen tanto una cuenta como un préstamo en el banco. En este caso se revisarán cada una de las filas de Impositor de tal forma que si el valor del Nombre_cliente en esa fila está en Impositor y prestatario, entonces este valor es parte de la consulta, la expresión en este caso es:

Solución

```
Select Nombre_cliente
        From Impositor
        Where exists
                (Select * from Impositor
                        Where Impositor.Nombre-cliente =
                        Prestatario.Nombre_cliente)
```

Nota: En el Select Principal, se pudo utilizar la relación prestatario en lugar de Impositor.

Aspectos particulares de la cláusula not exists.
Con esta operación es posible comprobar en SQL si una relación A contiene a otra relación B ($A \supseteq B$) con la expresión not exists (B exept A) de modo que $A \supseteq B$ si la salida de esta expresión SQL es vacía. Aunque no forma parte del SQL estándar el operador contains aparece en algunos sistemas relacionales.

Ejemplo:
Utilizando el hecho comentado anteriormente de que es posible por medio de la expresión not exists (B exept A) simular el hecho de que $A \supseteq B$, se desea obtener todos los clientes que tienen una cuenta en todas las sucursales de 'Granada'.

El mecanismo para obtener la consulta deseada será la siguiente: Para cada uno de los clientes, se generarán dos conjuntos, el conjunto B que va a contener el nombre de las sucursales ubicadas en la ciudad de Granada y un conjunto A que va a contener el nombre de las sucursales donde un determinado cliente tiene todas sus cuentas, es claro que si $A \supseteq B$, el cliente tiene cuentas (no necesariamente todas) en todas las sucursales de Granada. La expresión SQL es la siguiente:

```
Select distintc S.Nombre_cliente
        From impositor as S
        Where not exists
                ((select nombre_sucursal
                        From sucursal
                        Where ciudad_sucursal = 'Granada')
                except
                 (Select R.Nombre_sucursal
                        from impositor as T, cuenta as R
                        where T.Número_cuenta
                                = R.Número_cuenta
                          and
                        T.Nombre_cliente = S.Nombre_cliente))
```

Supóngase las siguientes instancias de las relaciones Impositor, Cuenta y Sucursal

Impositor

Nombre_Cliente	Número_Cuenta
Máximo Guido	C-121
Máximo Guido	C-690
Máximo Guido	C-391
Máximo Guido	C-422
Máximo Guido	C-132
Ana María Salgado	C-245
Ana María Salgado	C-291
Ana María Salgado	C-128

Cuenta

Número_Cuenta	Nombre_Sucursal	Saldo
C-121	Linda Vista	5,000
C-128	Linda Vista	8,000
C-132	Sutiava	3,000
C-245	San Felipe	4,500
C-690	Jalteva	6,000
C-391	Las Isletas	3,000
C-422	Las Isletas	2,500
C-291	Jalteva	5,000

Sucursal

Nombre_Sucursal	Ciudad_Sucursal	Activos
Linda Vista	Granada	120,000,000
Sutiava	León	80,000,000
Jalteva	Granada	75,000,000
San felipe	León	100,000,000
Las Isletas	Granada	50,000,000

El Select principal Select distinct S.Nombre_cliente from Impositor as s, determina el siguiente conjunto:

Nombre_Cliente
Máximo Guido
Ana María Salgado

Esta salida será filtrada por la condición where en función del conjunto generado por cada uno de los componentes de la salida.

Antes de analizar los conjuntos generados por la salida del Select principal, analicemos el conjunto generado por la primera subconsulta:
Select Nombre_sucursal from sucursal Where ciudad_sucursal = 'Granada',
Este es:

B=

Nombre_Sucursal
Linda Vista
Jalteva
Las Isletas

Es decir, todas las sucursales ubicadas en Granada. El conjunto determinado por la segunda subconsulta va a depender del elemento del Select principal a que nos estemos refiriendo, en el caso de 'Máximo Guido' el conjunto generado por:

```
Select R.Nombre_Sucursal from Impositor as T, cuenta as R
        Where T.Número_Cuenta  = R.Número_Cuenta and
              S.Nombre_Cliente = T.Nombre_Cliente
```

Es el nombre de las sucursales donde 'Máximo Guido' tiene sus cuentas es decir:

Nombre_Sucursal
Linda Vista
Jalteva
Las Isletas
Sutiava

Por lo tanto not exists (B exept A) no contiene filas por lo que A $\supseteq$ B ('Máximo Guido' tiene cuentas en todas las sucursales de 'Granada')

Similarmente puede comprobarse en el caso de 'Ana María Salgado' de que A $\supseteq$ B es falso es decir, no tiene cuentas en todas la sucursales de 'Granada'. Por tanto en este caso y con las instancias antes indicadas, el resultado de la consulta sería:

Nombre_Cliente
Máximo Guido

Alcance de las variables de tuplas.
En consultas que contegan subconsultas la regla del alcance de estas variables es similar a la que se utiliza en los lenguajes de programación, así en una subconsulta solo se pueden utilizar variables de tupla que estén definidas en la propia subconsulta o en cualquier consulta que contenga dicha subconsulta.

7.6.2.11 Comprobación de Tuplas Duplicadas en la consulta.

Esta comprobación se realiza en SQL mediante el uso de la palabra reservada **unique**, forma general:

unique (consulta SQL).

Ejemplo:
Obtener todos los clientes que tienen solo una cuenta en la sucursal 'Linda Vista'

Solución

```
Select T.Nombre_cliente
        From Impositor as T
        Where unique (select R.Nombre_cliente
                      From cuenta, Impositor as R
                      Where T.Nombre_cliente
                      = R.Nombre_cliente and
                      R.Número_cuenta
                      = cuenta.número_cuenta and
                      cuenta.Nombre_sucursal = 'Linda Vista')
```

Nótese que el Dominio de T es la relación impositor (lo cual garantiza que los clientes representados por T.nombre_cliente tiene por lo menos un préstamo.
Por otra parte el conjunto generado por la subconsulta son todos los clientes proyectados de la expresión.

$$\sigma_{\text{Nombre\_Sucursal='Linda Vista'}} (\text{Impositor} \mid x \mid \text{cuenta})$$

donde se incluyen nombres repetidos, entonces la expresión unique (select R.Nombre_cliente...) generará un valor de verdad solo cuando el nombre del cliente determinado por la variable de tupla T se encuentre solo una vez en (select R.nombre_cliente...) en cuyo caso dicho nombre será parte de la consulta.

Veamos cual sería la salida resultante con las siguientes instancias de Impositor y Cuenta.

Impositor

Nombre_cliente	Número_cuenta
Ernesto	C-2166
Ernesto	C-3284
Ernesto	C-9264
Ricardo	C-1722
Ricardo	C-1642
Zoraida	C-1934

Cuenta

Número_Cuenta	Nombre_Sucursal	Saldo
C-2116	Linda vista	6.000
C-3284	Las Brisas	5,000
C-9264	Linda Vista	4.000
C-1722	Monseñor Lezcano	3,000
C-1642	Linda Vista	7.000
C-1934	Linda Vista	5,500

La subconsulta genera la salida.

Nombre_Cliente
Ernesto
Ernesto
Ricardo
Zoraida

Figura 7.81 Salida de Consulta con Comprobación de tuplas duplicadas

Es decir los clientes con cuenta en 'Linda Vista', entonces debido a la cláusula unique, los clientes que conformarán la consulta son: Ricardo y Zoraida.

7.6.3 Vistas.

Una vista en SQL se define con el comando Create View, forma general:

Create View V as <expression de consulta>
Se debe notar que la notación utilizada para definir una vista en el álgebra relacional está basada en esta definición de vista del SQL.

Ejemplo:
Considérese la vista que consiste de los nombres de las sucursales y los nombres de los clientes que tienen una cuenta ó un préstamo en esa sucursal. El Nombre de la vista es: Todos_los_clientes

Solución

```
Create view  Todos_los_clientes as
(select nombre_sucursal, nombre_cliente
from Impositor, cuenta
Where Impositor.Número_préstamo
= cuenta.numero_cuenta)

Union

(select nombre_sucursal, nombre_ cliente
from prestatario, préstamo
Where Prestatario. Nombre_préstamo
= préstamo.Nombre_préstamo)
```

La orden Create View V as <expresión de consulta> almacena de forma permanente el resultado de la <expresión de consulta> y se actualiza cada vez que se modifican las relaciones involucradas en la <expresión de consulta>, se pueden utilizar las vistas como si fuesen relaciones para crear consultas ó para crear otras vistas.

7.6.4 Consultas complejas.

Son aquellas difíciles de obtener con un único bloque ó utilizando varios bloques relacionados entre sí por la operación unión, intersección ó diferencia.
Las formas de componer varios bloques que serán estudiadas a continuación son: Las relaciones derivadas y la cláusula with.

7.6.4.1 Relaciones Derivadas.

Son expresiones de subconsulta en la cláusula from en las que la subconsulta se le debe dar un nombre y opcionalmente se pueden renombrar los atributos.

Ejemplos:

(a) Considérese la siguiente expresión SQL con relación derivada

```
Select nombre_sucursal, saldo_medio
        From (select nombre_sucursal, Avg (saldo)
                From cuenta
                Group by nombre_sucursal)
                As resultado (nombre_sucursal, saldo_medio)
        Where saldo_medio > 1200
```

En este caso la relación derivada se le ha dado el nombre de resultado y se ha renombrado Avg (saldo) por saldo_medio.
Nótese que en este caso se utiliza la cláusula where en lugar de la cláusula having, where reconoce a saldo_medio como atributo.

(b) Obtener el máximo del total de saldos de todas las sucursales.

```
Select max (saldo_total)
        From (select nombre_sucursal, sum (saldo)
                From cuenta
                Group by nombre_sucursal) as
        Total_sucursal (nombre_sucursal, saldo_total)
```

Note como primero se genera la relación total_sucursal a partir de la subconsulta dentro del from y a partir de ahí el select se ejecuta normalmente reconociendo los atributos generales, nótese además que en este caso no hubiese sido posible resolver la consulta utilizando la cláusula having ya que como comentamos anteriormente no se pueden utilizar función de funciones (composición de funciones) si estas son de agregación.

La cláusula with.
Al estudiar las vistas se ha indicado que estas se pueden utilizar para crear otras vistas, es decir que expresiones complejas se pueden resolver usando vistas de forma reiterada con el inconveniente de tener que guardarlas. La idea de la cláusula with es de generar expresiones SQL reutilizables pero que no se almacenen de manera permanente. Esta cláusula ha sido introducida con SQL:1999.

Ejemplo:
Considérese la siguiente consulta: obtener los números de las cuentas con máximo saldo.

Solución

```
With saldo_máximo (valor) as
        Select max (saldo)
        From cuenta
Select número_cuenta
        From cuenta, saldo_máximo
        Where cuenta.saldo=saldo_máximo(valor)
```

Nótese que la expresión SQL

```
With saldo_máximo (valor) as
Select max (saldo)
From cuenta
```

Genera la vista temporal de nombre saldo_máximo que contiene el saldo máximo y para efectos posteriores el atributo de este saldo, max (saldo) será reconocido como valor.

La consulta principal

```
Select número_Cuenta
From cuenta, saldo.máximo
Where cuenta.saldo = saldo_máximo(valor)
```

Proyecta los números de cuenta pero solamente aquellos que son iguales al valor máximo posibilitando de esta manera listar no solo el valor del saldo máximo, sino todas las cuentas con ese valor.

Uso de múltiples With en una expresión SQL

Se pueden utilizar múltiples cláusulas with para generar las vistas temporales requeridas en el select principal

Ejemplo:
Encontrar todas las sucursales donde el depósito de cuentas es mayor que la media del total de los depósitos de cuentas en todas las sucursales.

Solución

Estrategia para encontrar la expresión SQL:

(a) Encontrar cada sucursal con su respectivo total de depósito de cuentas (primer with).
(b) Utilizando la salida del with anterior generar el promedio de los totales de todas las sucursales.
(c) Utilizar las vistas temporales generales con los with utilizados anteriormente para obtener el resultado deseado mediante el select principal.

La expresión es la siguiente:

```
With total_sucursal (nombre_sucursal, valor) as
Select nombre_sucursal, sum (saldo)
From cuenta
Group by nombre_sucursal

With total_media_sucursal (valor) as
Select Avg (valor)
From total_sucursal

Select nombre_sucursal
From total_sucursal, total_media_sucursal
Where total_sucursal.valor > = total_media_sucursal.valor
```

7.6.5 Modificación de la Base de Datos

SQL posee comandos destinados también para modificar la base de datos, entre estos se tienen operaciones de borrado, inserción y actualización.

7.6.5.1 Borrado

La expresión SQL relacionada con el borrado es:

```
Delete from r
  Where p
```

Donde r es una relación y p es una condición lógica que determina las tuplas a ser eliminadas de r, si la condición p es omitida, se eliminan todas la tuplas de r

Ejemplos:

(a) Delete from préstamo

En este caso se eliminan todas las tuplas de la relación préstamo

(b) Delete from cuenta
Where nombre_sucursal = 'Linda Vista'

Elimina todas las tuplas en cuenta siempre y cuando la sucursal sea Linda Vista.

(c) Borrar todos los préstamos en los que el monto del préstamo esté entre 5,000 y 10,000 córdobas.

Solución

```
Delete from préstamo
Where importe between 5,000 and 10,000
```

(d) Borrar las cuentas de todas las sucursales de Masaya

Solución

En este caso le generará mediante una subconsulta los nombres de las sucursales ubicadas en Masaya. De modo que la expresión SQL requerida es:

```
Delete from cuenta
    Where nombre_sucursal in (select nombre_sucursal
                              From sucursal
                              Where
                              Ciudad_sucursal = 'Masaya')
```

Uso de funciones de agregación en una subconsulta de borrado.

Ejemplo:

```
Delete from cuenta
Where saldo < (select avg (saldo) from cuenta)
```

En este caso se eliminarán las tuplas de cuenta cuyos saldos sean menores al saldo promedio.

7.6.5.2 Inserción de Tuplas en una Relación.

El comando SQL encargado de realizar la inserción de una ó más tuplas en una relación es el comando Insert, este tiene dos variantes una para insertar una sola tupla indicando los valores de los atributos de forma explícita, la forma general en este caso es insert into r values (tupla a insertar) en el caso de que se respete el orden de los atributos en la relación r, en caso de que la tupla a insertar no respete ese orden, el nombre de los atributos se debe listar también en el comando insert

Ejemplos:

a) Insert into cuenta
   Values ('C-239', 'Linda Vista', 1,500)

El efecto en este caso es que se añade la tupla indicada a la relación, nótese además que se ha respetado el orden de los atributos en cuenta.

Ejemplo (b)
Insert into cuenta(Nombre_Sucursal,úmero_Cuenta,Saldo)
Values ('Linda Vista', 'C-239', 1,500)

El efecto es similar al ejemplo anterior con la diferencia de que se han listado explicitamente los nombres de los atributos, esto debido a que en la cláusula values no se ha guardado el orden que estos tienen en la estructura de la tabla

La otra variante de la instrucción Insert tiene como finalidad el insertar un conjunto de tuplas resultantes de una operación Select

Ejemplo:
Como incentivo a los clientes que tienen una cuanta de ahorro en la sucursal 'Linda Vista' se les abrirá una cuenta de ahorro con 200, el número de la cuenta de ahorro será el mismo número del préstamo

En este caso se deben de dar de alta a las cuentas en la relación cuenta y por otra parte afectar también a la relación Impositor para indicar a quienes pertenecen dichas cuentas.

Solución

```
Insert into cuenta
        Select nombre_sucursal, número_préstamo, 200
        From préstamo
        Where nombre_sucursal = 'Linda Vista'
```

Así mismo la afectación a Impositor estará dada por:

```
Insert into Impositor
        Select nombre_cliente, número_préstamo
        From prestatario, préstamo
        Where prestatario.número_préstamo =
                Préstamo.número_préstamo and
                Nombre_sucursal = 'Linda Vista'
```

7.6.5.3 Actualizaciones de las tuplas de una relación.

Esta acción se logra mediante el uso del comando update

Forma general

Sea r una relación y sea exp la expresión de actualización, entonces la forma general del comando de actualizaciones:

```
Update r
Set exp
```

Ejemplos

(a)
```
Update cuenta
Set saldo = saldo * 1.05
```

Efecto: Todos los saldos de la relación cuenta se incrementa en el 5 %

(b)
```
Update cuenta
Set saldo = saldo * 1.05
Where saldo > = 1000
```

Efecto: Similar al ejemplo (a) con la diferencia de que solo se actualizarán aquellas cuentas en que el saldo sea mayor o igual a 1000

(c)
```
Update cuenta
Set saldo = saldo * 1.05
Where (saldo > select Avg (saldo) from cuenta)
```

En este caso en forma similar a los casos anteriores se producirá un incremento del 5% en los saldos de las cuentas pero solo serán afectadas aquellas cuentas cuyos saldos sean mayores que el promedio de todos los saldos.

(d) (Actualización condicional)
Se requiere modificar el saldo de las cuentas de modo que los saldos superiores a 10,000 recibirán un incremento del 6%, mientras que los saldos inferiores ó iguales a 10,000 recibirán solamente un incremento del 5%

Solución

En este caso la actualización se debe realizar en 2 partes, debido a una de las debilidades de SQL la cual consiste en que con este lenguaje no se puede recorrer una relación fila a fila como se hace en los lenguajes de propósito general. Las expresiones SQL son:

```
Update cuenta
Set saldo = saldo * 1.06
Where saldo > = 10,000
```

```
Update cuenta
Set saldo = saldo * 1.05
Where saldo  > = 1,000
```

Este tipo de solución tiene el inconveniente de que si se modifica el orden de las consultas de actualización se pueden generar actualizaciones erróneas. Así si se realiza primero el incremento del 5%, valores cercanos a 1000 pasarían a ser valores mayores de 1000 con lo cual se les aplicaría el 6% de incremento, llegándose a un incremento neto del 1.05 * 1.06 = 1.113 es decir el 11.3%, en lugar del 5% para evitar esta problemática SQL cuenta con el comando case muy similar al comando de los lenguajes de propósito general, así la expresión del ejemplo anterior utilizando case es:

```
Update cuenta
Set saldo = case
When saldo < = 10,000 then saldo * 1.05
Else saldo * 1.06
End
```

La forma general de la cláusula case es:

```
Case
When pred1 then result1
When pred2 then result2
…
when predn then resultn
else result0
end
```

En este caso se analizan cada uno de los predicados y si $pred_i$ es verdadero entonces se ejecuta $result_i$.
Por otra parte si ninguno es verdadero se ejecuta $result_0$

7.6.5.4 Anomalías al Utilizar Vistas

La problemática de las vistas se produce al utilizar la operación Insert Into cuando se intenta insertar tuplas en una relación a partir de una vista cuyas tuplas no son compatibles con las de la relación. Así supóngase que se ha creado la siguiente vista:

```
Create view Préstamo_sucursal as
Select Nombre_Sucursal,Número_Préstamo
From Préstamo
```

La vista resultante es similar a Préstamo con la diferencia de que la vista no contiene el atributo Importe.
Al utilizar Préstamo_Sucursal para insertr datos en Préstamo mediante la expresión: Insert Into Préstamo_Sucursal Values ('Linda Vista', 'P-213'), esto producirá la inserción de ('P-213', 'Linda Vista', null) en la relación Préstamolo cual genera en esta relación una anomalía de datos faltantes.

Por la problemática antes expuestas muchas bases de datos relacionales que utilizan SQL imponen la siguiente restricción:

Los datos contenidos en la vista solo pueden ser modificados por los comandos Update, Insert o Delete si estos utilizan la relación original sin el empleo de Agregación, así la operación
Insert Into Préstamo_Sucursal Values('Linda Vista', 'P-213')
no estaría permitida.

7.6.5.5 SQL y las Transacciones

Una transacción es un conjunto de instrucciones SQL de consultas y actualizaciones que deben de ejecutarse todas de forma exitosa o no ejecutar ninguna. La norma SQL especifica que una transacción comienza de forma implícita (no hay comando de comienzo) cuando se ejecuta una instrucción SQL, la transacción finaliza con cualquiera de las siguientes opciones dependiendo de la decisión del usuario.

Commit: Compromete la Transacción actual, es decir hace que los cambios realizados por la transacción sean permanentes en la base de datos. Después de ejecutarse este comando comienza una nueva transacción de forma automática

RollBack: Su efecto es retroceder la transacción actual es decir deshace todas las actualizaciones realizadas por las instrucciones SQL,de tal forma que la base de datos se restaura al estado que existía previo a la primera instrucción de la transacción.

En el caso de falla ó caída del sistema los efectos de la transacción se retrocederán, claro está que este proceso se produce cuando el sistema se recupera. El detalle sobre transacciones se estudiará en el curso de Base de Datos II.
Nota: el tratamiento de las transacciones en SQL:99 es más clara ya que la transacción en si tiene un comienzo y un final explícito con los comandos :Begin Atomic y end.

7.6.5.6 Más Sobre Reuniones de Relaciones

Hasta ahora se ha implementado la unión de relaciones utilizando el equivalente SQL del producto cartesiano ó el producto natural. Se estudiará a continuación una cláusula para obtener de forma más directa el producto natural (natural Inner Join), además variantes de este para listar todos los atributos de las relaciones a reunir y la posibilidad de tomar en cuenta valores que no se encuentran en ambas relaciones. Las cláusulas en este caso van ubicadas dentro de la cláusula from de la consulta de selección. Las diferentes variantes son:
Inner Join, Left Outer Join, Right Outer Join, Full Outer Join, Natural Inner Join, Natural Right Outer Join, Natural Left Outer Join, Natural Full Outer Join.
Para mostrar el efecto de estas cláusulas se utilizará una instancia de las relaciones Préstamo y Prestatario

Relación Préstamo

Número_préstamo	Nombre_Sucursal	Importe
P-170	Linda Vista	50,000
P-230	Las Brisas	40,000
P-260	Nejapa	35,000

Relación Prestatario

Nombre_Cliente	Número_Préstamo
Alvaro	P-170
Máximo	P-230
Hermógenes	P-155

La Cláusula Inner Join
Sean R1 y R2 dos relaciones, sea X un atributo de R1 y Y un atributo de R2,ambos con el mismo dominio, entonces la forma general de la expresión SQL que utiliza esta cláusula es:

Select $r1_1$, $r1_2$,… $r1_m$, $r2_1$, $r2_2$,… $r2_n$ From **R1 Inner Join R2 on** R1.X =R2.Y

Donde $r1_1$, $r1_2$,… $r1_m$ son algunos atributos de R1 y $r2_1$, $r2_2$,… $r2_n$ son algunos atributos de R2, como siempre * indica la inclusión de todos los atributos de R1 y R2 en la salida.

Ejemplo:
Obtener la relación resultante de la expresión SQL:

Select * From Préstamo Inner Join Prestatario On
Préstamo.Número_Préstamo=Prestatario.Número_Préstamo

Solución

Número_ Préstamo	Nombre_ Sucursal	Importe	Nombre_ Cliente	Número_ Préstamo
P-170	Linda Vista	50,000	Alvaro	P-170
P-230	Las Brisas	40,000	Máximo	P-230

Figura 7.82 Salida de consulta del tipo inner join

Nótese que el Inner Join se diferencia del Producto natural en que en este último el atributo en común solo se presenta una sola vez.

La Cláusula Left Outer Join
La reunión usando está cláusula además de las filas que genera la cláusula Inner Join genera tantas filas como elementos diferentes a los comunes existen en la relación que se encuentra a la izquierda de la operación estas filas extras se generan de la siguiente manera: se enlistan los valores de los atributos de la relación a la izquierda y la parte derecha correspondiente a los atributos de la segunda relación se ponen a null, para las Préstamo y prestatario la reunión del tipo Left Outer Join tiene la siguiente expresión:

Select * from préstamo Left Outer Join Prestatario on
Préstamo.Número_Préstamo=Prestatario.Número_Préstamo

El resultado es el siguiente:

Número_ Préstamo	Nombre_ Sucursal	Importe	Nombre_ Cliente	Número_ Préstamo
P-170	Linda	50.000	Alvaro	P-170
P-230	Las	40.000	Máximo	P-230
P-260	Nejapa	35.000	null	null

Figura 7.83 Salida de consulta del tipo Reunión externa por la izquierda

La Cláusula Right Outer Join
Su efecto es similar a la anterior con la evidente diferencia de que en este caso las tuplas contarán con valores por la derecha y null por la izquierda. Para las relaciones Préstamo y Prestatario, la salida al efectuar una reunión con la cláusula Right Outer Join es la siguiente:

Número_ Préstamo	Nombre_ Sucursal	Importe	Nombre_ Cliente	Número_ Préstamo
P-170	Linda Vista	50,000	Alvaro	P-170
P-230	Las Brisas	40,000	Máximo	P-230
null	null	null	Hermógenes	P-155

Figura 7.84 Salida de consulta del tipo Reunión externa por la Derecha

La Cláusula Full Outer Join
Su efecto es la unión del efecto del Left Outer Join y el Right Outer Join de modo que en este caso la cantidad de filas que genera son las del Inner Join más las extras generadas por el Left y Right Outer Join.

7.6.5.7 Ejemplo de definición de Esquemas Relaciones (ORACLE/SQL)

Recordemos que un esquema de una base de datos relacional está formado basicamente por la definición de un conjunto de tablas (relaciones).

Cada tabla debe de tener un nombre único en el esquema y estar definida en base a la especificación de un conjunto de atributos, es decir las columnas de la tabla que definen las propiedades del objeto del mundo real representado por esta tabla.

Cada atributo debe tener un nombre único para una tabla y estará definido en un determinado dominio de datos. Entre los atributos que conforman la definición de una tabla, uno o un conjunto de ellos se elegirán como clave primaria de la misma.

Opcionalmente, en la definición de una tabla se especificarán las claves ajenas o foráneas. SQL utiliza el comando **CREATE TABLE** para la definición de las tablas del esquema relacional. El formato general de este comando es el siguiente:

CREATE TABLE <Nombre de la Tabla>
(Nombre-Atributo 1: Dominio (opción),
Nombre-Atributo 2: Dominio (opción),
…….. ……..
Nombre-Atributo n: Dominio (opción),
Cláusulas opcionales)

Cláusulas Opcionales utilizadas por el SGBD ORACLE
Las cláusulas opcionales son:

Primary Key (Lista de Atributos) : Indica el o los atributos de la tabla que conforman la llave primaria de la relación.

Foreign Key (Lista de Atributos)
References Nombre de Tabla (Llave Primaria)
Estas dos cláusulas del Comando **Create Table** se complementan **Foreign key** indica los atributos de la relación que se está definiendo que sirven como enlace para unirse con otra relación del esquema, esa otra relación se especifica en la cláusula **References**

Constraint <Nombre del Constraint>
Cuerpo del Constraint
Esta cláusula permite definir las diferentes restricciones de integridad de la relación. En función de los requerimientos de la relación que está siendo definida, la cláusula Constraint puede utilizarse varias veces en un comando **Create Table**.

Restricciones Utilizadas en la Cláusula Contraint
Not Null:
Indica que el atributo no podrá asumir valores nulos para las tuplas de la tabla. Esta opción le dará esta propiedad entre otros a la llave primaria de la relación

Primary Key :
Opcionalmente esta cláusula puede ir dentro de la cláusula Constrint con el mismo efecto de lo explicado anteriormente

Unique :
Impide la posibilidad de duplicación de valores en un atributo

Foreign Key, References :
El mismo efecto del explicado anteriormente

On Delete Cascade :
Indica con su presencia que en el proceso de borrado de la tupla referenciada por tuplas de otra tabla estas deben ser borradas también

Check :
Define las restricciones específicas en cuanto al conjunto de valores que puede tomar un atributo de la tabla

Disable/Enable :
Posibilita la desactivación/activación de un Constraint

7.6.5.8 Ejemplo de tipos de Datos predefinidos (ORACLE/SQL)

Char(Longitud): Cadenas de caracteres de tamaño fijo, hasta un máximo de 255
Varchar(Longitud),Varchar2(Longitud): Cadenas de Caracteres de longitud variable de hasta un máximo de 2000 caracteres
Number(p,e): Números de longitud variable donde p representa el número total de dígitos y e el número de decimales
Float(p): Números reales en el sistema de coma flotante con una precisión binaria expresada por p, el valor por defecto es de 126
Long: Cadenas de caracteres de hasta 2 gigabytes se utiliza para representar campos generalmente del tipo memo o cometario
Date: Representa información cronológica(Fecha/Hora), desde el 1 de Enero de 4712 AC hasta el 31 de Diciembre del 4712 DC
Raw(Longitud): Representa valores binarios de longitud máxima de 255 bytes
Long Raw: Igual que el tipo anterior con la diferencia que el tamaño máximo es de 2 gigabytes
Rowid: Tipo de dato binario que representa la dirección de una tupla de una tabla.

Ejemplos

(a) Sea el siguiente comando del DDL del SQL/ORACLE

```
Create Table Cine
        (cine               Varchar2(15) NOT NULL,
         ciudad_cine        Varchar2(15),
         direccion_cine     Varchar2(25),

         CONSTRAINT         pk_cine
         PRIMARY KEY        (cine),
         CONSTRAINT         ck_cin
         CHECK      (cine=INITCAP(cine)) );
```

En este caso se creará una tabla de nombre **Cine** con los siguientes atributos, dominios y restricciones

Atributo cine de hasta 15 caracteres, este atributo no permitirá dejar en blanco el valor respectivo, es decir el valor del atributo es obligatorio. Por otra parte como parte del CONSTRAINT pk_cine, se indica que este atributo será la llave primaria de esta relación por lo que debe de cumplir con todas las reglas de integridad de las llaves primarias. Por otra parte la cláusula CONSTRAINT ck_cin obliga que la función INITCAP se ejecute sobre el valor del atributo dando como resultado que los valores en cine comienzen con mayúsculas y el resto en minúsculas.

Atributo ciudad_cine de longitud máxima de 15 caracteres
Atributo direccion_cine de longitud máxima de 25 caracteres

(b) Dado el siguiente Esquema relacional de una base de datos relacionada con información de películas que han sido proyectadas en el país, diseñar la tabla **Proyección** de este esquema. Antes de darle solución a este problema estudiaremos algunos aspectos relacionados con la notación utilizada en este diagrama

Aspectos Particulares de Notación en este Esquema Relacional

- En este esquema las relaciones consultan en las entidades lo cual está indicado por la dirección de la flecha

- Cuando las llaves ajenas son compuestas, como puede verse en el diagrama se utiliza un símbolo especial (Círculos oscuros conectados) para indicar cual es la relación que deben de consultar.

- Las llaves primarias compuestas se indican en negritas. Se debe notar que en este diagrama se indican no solo las llaves primarias "Naturales" de las entidades sino que también las llaves primarias de todos los conjunto relación esto con el fin de garantizar en estas tablas las reglas de integridad.

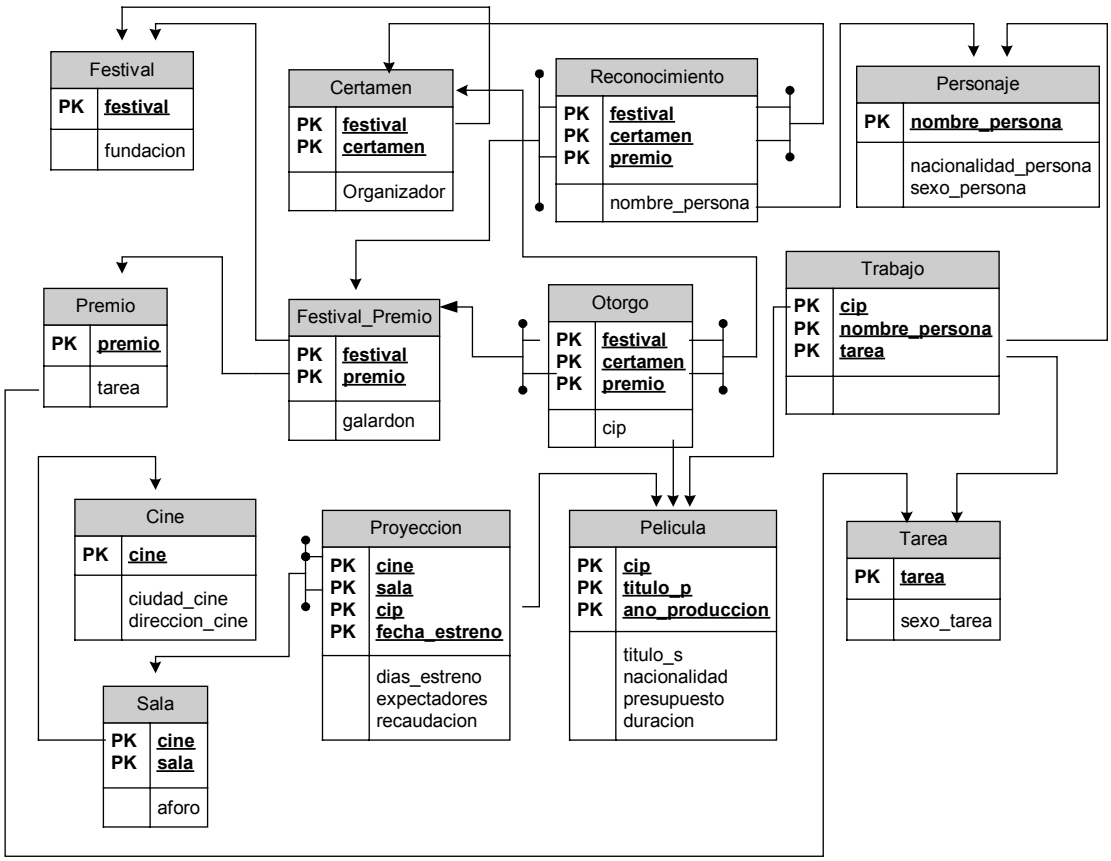


Figura 7.85 Esquema Relacional: proyección de películas

Solución

Observando el diagrama, este nos ofrece la siguiente información:

- Los atributos de **Proyección** son: cine, sala, cip, fecha_estreno, díasestreno, expectadores y recaudación
- Consulta a la relación Sala y a la relación Película, la consulta a Sala la realiza utilizando la llave ajena compuesta por los atributos cine y sala,mientras que la llave ajena para consultar en película es simple y es el atributo cip.

- La Llave primaria de Proyección está compuesta por los atributos: Cine,sala,cip,fecha_estreno.

- En resumen: Proyección es un conjunto de relaciones vinculado con los conjuntos de entidades fuertes: Sala y Pelicula, lo cual debe quedar reflejado en el comando Create Table :

```
Create Table Proyeccion
        ( cine                  Varchar2(15) not null,
          sala                  Number(2) not null,
          cip                   varchar2(10) not null,
          fecha_estreno         Date not null,
          dias_estreno          Number(3),
          espectadores          Number(6),
          recaudacion           Number(8),
        Constraint      pk_proyeccion
                Primary Key(cine,sala,cip,fecha_estreno),
        Constraint      fk_pro_sal
                Foreign         Key (cine,sala)
                References      Sala(cine,sala)
                On Delete Cascade,
        Constraint fk_pro_pel
                Foreign         Key (cip)
                References      Pelicula(cip)
                On Delete Cascade,
        Constraint ck_dia
                Check (dias_producción >0) );
```

Note Que se usa la regla referencial **On Delete Cascade** indicando que la relación **Proyeccion** es una entidad débil respecto a las relaciones **Sala** y **Pelicula**, de modo que si se borra una fila de **Sala** por ejemplo, se borrarán también todas las filas de **Proyeccion** vinculas con esa fila de **Sala** y de igual manera en el caso de la relación **Pelicula**: Al borrar una tupla de esta tabla se eliminarán también las tuplas vinculas en **Proyeccion.** También nótese que se ha utilizado la regla de validación **Check** para restringir la entrada a determinados valores así Check (dias_producción >0) indica que el atributo dias_produccion debe contener solamente valores positivos.

(c) En este ejemplo se utiliza la cláusla Check para efectuar restricciones de integridad dependiendo del valor asumido por un determinado atributo.Otro aspecto importante que se resalta en este ejemplo es la utilización de un Dominio generado por el usuario, este es el dominio Tipos_Doc que se genera con el comando Create Domain, de la siguiente manera:

```
Create Domain Tipos_Doc Varchar(1)
    Constraint Artículos_o_Libros
        Check(Value In ('A', 'L'));
```

El Comando Create Table es el siguiente:

```
Create Table Documento
    (Tipo          Tipos_Doc,
     Cod_Doc       Varchar(4),
     Titulo        Varchar(25) Not Null,
     Idioma        Idiomas,
     Nombre_E      Nombres,
     Año           Integer(4) Check (Año >1950)
     Isbn          Integer(10),
      Primary Key (Tipo,Cod_Doc),
     Unique(Isbn),
     Check((Tipo='A' And Isbn Is Null And Nombre_E Is Null)
            Or
            (Tipo='L'  And Isbn Is Not Null And
                  Nombre_E Is Not Null)),
    Foreign Key (Nombre_E) References To Editorial
            On Update Cascade
            On Delete No action));
```

Note que además del atributo **Tipo** con dominio diseñado por el usuario, también **Idioma** y **Nombre_E** sus dominios han sido diseñado de la misma forma.La llave primaria en este ejemplo está compuesta por los atributos: **Tipo** y **Cod_Doc**, se ha definido como llave candidata al atributo **Isbn** por medio de la cláusula unique y finalmente el la cláusula **Check** en este caso es condicional dependiendo la restricción de que el Tipo de Documento sea un Articulo o un Libro de modo que si se tratase de un Artículo los atributos Isbn y Nombre de la editorial (Nombre_E) la relación aceptará valores nulos en dichos campos pero si por el contrario el tipo de documento es un Libro los valores en Isbn y Nombre_E son obligatorios. Finalmente se vinculan la tabla que se está diseñando con la tabla Editorial utilizando la cláusula **Foreign Key**, indicándose además que las modificaciones se efectuarán en cascada y no así los borrados.

7.6.5.9 Otros Comandos del DDL de SQL

El Comando **Drop Domain <Nombre de Dominio>**
Este Comando permite eliminar un Dominio previamente definido

El Comando **Drop Constraint <Nombre del Constraint>**
Permite eliminar un determinado constraint definido en un comando **Create Table** o en un comando **Create Domain**

El Comado **Drop Table <Nombre de la Tabla>**
Este Comando elimina una determinada tabla de una base de Datos

Modificación del Esquema
El Comando:

Alter Table <Nombre de la Tabla>
Especificación de la Modificación

Ejemplos:

(a) Alter Editorial
Add Column Director Varchar (30)

Efecto:
Se Modifica el Esquema de la tabla Editorial agregándole un atributo de nombre director con dominio Varchar(30)

(b) Alter Table
Editorial
Alter Column Ciudad Set Default 'Managua' ;

En este caso la modificación del esquema de Editorial consiste en determinar a "Managua" como el valor por defecto para el atributo Ciudad.

(c) Alter Table
Editorial
Drop column
Dir ;

El efecto en este ejemplo será de que se elimina el atributo Dir del esquema de la tabla Editorial

7.6.5.10 Índices

En la definición de los esquemas relacionales es muy importante la definición de índices con el el fin de mejorar el rendimiento en cuanto al acceso de a los datos de la base de Datos. Se debe de tener cuidado en no utilizar en forma innecesaria la creación de índices ya que al actualizarlos afectan el desempeño de ciertas operaciones como borrado e inserción, por lo que se recomienda indexar hasta que se han introducido una cantidad considerable de tuplas. La sintaxis para la creación de índices en SQL es la siguiente:

Create Index <Nombre del índice> **ON**
<Nombre de la tabla> (Lista de Campos) ;

El efecto es que se crea el un archivo índice para una determinada tabla definida en el comando. La lista de Campos indica los atributos que conformarán el índice siendo el primero de la lista el principal.
Ejemplo:

Create Index Ind_1 ON
Estudiante (Nombre) ;

El efecto es que se crea un índice con nombre Ind_1 que afectará al atributo Nombre de la tabla Estudiante

Un índice ya creado se elimina con el comando:
Drop Index <Nombre del Índice>

7.6.6 Ejemplo de funciones definidas (ORACLE)

Pueden emplearse en las cláusulas SELECT, WHERE y ORDER BY. El argumento de una función puede ser una columna, variables o constantes. *Una función puede actuar como argumento de otra función.*
Las funciones se pueden clasificar en:

1.- Aritméticas
2.- De maneio de caracteres
3.- De maneio de Fechas
4.- De Conversión
5.- Otras Funciones

7.6.6.1 Funciones aritméticas

ARITMETICAS	
FUNCIÓN	**DESCRIPCIÓN**
Abs(n)	Valor absoluto de n
Ceil(n)	Entero inmediatamente superior a n (n no es entero)
Floor(n)	Entero inmediatamente inferior a n (n no es entero)
Mod(valor,divisor)	Genera el residuo de una división entera
Power(valor,exponente	Eleva el valor al exponente indicado
Round(valor,precisión)	Redondea valor a la precisión indicada
Sign(n)	Si n<0 -1 Si n=0 0 Si n>0 1
Sqrt(n)	Raíz cuadrada d en, si n<o NULL
Trunc(valor,precisión)	Valor es truncado a una determinada precisión

7.6.6.2 Funciones de Cadenas de caracteres

CADENA DE CARACTERES	
FUNCION	**DESCRIPCION**
Ascii(n)	Valor ascii del primer carácter a la izquierda de la cadena “n”
Chr(n)	El carácter cuyo valor ascii es el número “n”

Initcap(n)	Pone en mayúsculas la primera letra de cada palabra de la cadena "n"
Instr(cadena,conjunto[,comienzo[,ocurrencia]])	Busca en la cadena un cierto conjunto de caracteres
Lenth(n)	El número de caracteres de "n"
Lower(n)	Cadena "n" con todas sus letras convertidas a minúsculas
Lpad(S1,n[,S2])	S1 visualizado con longitud "n" y justificado a la derecha,"S2" es la cadena con la que se rellena por la izquierda
Rpad(S1,n[,S2])	Similar a Lpad pero rellenado por la derecha
Substr(s,m[,n])	Devuelve la sub cadena s sustituyendo las ocurrencias del conjunto de caracteres de la cadena f, por los del conjunto t. Cada carácter de s que aparece en f es traducido a su correspondiente en t.
Translate(s,f,t)	Devuelve la cadena s sustituyendo las ocurrencias del conjunto de caracteres de la cadena f, por los del conjunto t. Cada carácter de s que aparece en f es traducido a su correspondiente en t
Upper(s)	Cadena "s" con todas sus letras en mayúsculas
Rtrim(s)	Suprime los blancos que contengan la cadena "s" a la derecha
Ltrim(s)	Suprime los blancos que contengan la cadena "s" a la izquierda

7.6.6.3 Funciones de Manejos de Fechas

DE FECHAS	
FUNCION	**DESCRIPCION**
Add_Months(d,n)	Fecha"d" más "n" meses
Last_Day (d)	Fecha del último día del mes contenido en "d"
Moths_Between (d1,d2)	Diferencia en meses entre la fecha "d1" y la fecha "d2" el resultado puede ser negativo
Next_Day(d,day)	Devuelve la fecha del día de la semana siguiente identificado por day.
Sysdate	Día de hoy

7.6.6.4 Funciones de Conversión

DE CONVERSION	
FUNCION	**DESCRIPCION**
To_Number(s)	Convierte una cadena alfanuméerica que solo contiene números, a un valor numérico
To_Char(d,[,fmt])	Convierte una fecha “d” a un valor de tipo VARCHAR2 en el formato especificado por fmt, si se omite fmt, “d” se convierte al formato de fecha estándar
To_date (s, [,fmt])	Convierte una cadena “s” en una fecha con el formato especificado por fmt, si se omite fmt, “s” debe tener el formato de fecha estándar ' dd-mon-yy '

Las funciones TO_CHAR y TO_DATE utilizan máscaras de formato, estas máscaras son *“pictures”* del dato final que permiten el control de las funciones de conversión de datos.

Ejemplo:

Esta función convierte la cadena '121234' en el número 121234

```
Select To_Number ('121234') From dual ;
```

7.6.6.5 Funciones en los Datos

Las funciones anteriores como TO_CHAR y TO_DATE utilizan máscaras de presentación del formato elegido, como las que se presentan a continuación.

MASCARA	**DESCRIPCION**
cc ó scc	Valor del siglo
y,yyy ó sy,yyy	Año con coma, con o sin signo
yyyy	Año sin signo
yyy	Últimos 3 dígitos del año
yy	Últimos 2 dígitos del año
y	Últimos dígito del año
q	Número de trimestre
ww	Número de la semana del año
w	Número de la semana del mes
mm	Número del mes
ddd	Número del día del año
dd	Número del día del mes
d	Número del día de la semana
hh ó hh12	Hora(1-12)
hh24	Hora(1-24)

mi	Minutos
ss	Segundos
sssss	Segundos Trancurridos desde media noche
j	Juliano
syear o year	Año en Inglés(Ej:nineteen-eighty-two)
month	Nombre del mes
mon	Abreviatura de tres letras del nombre del mes
day	Nombre del día de la Semana
dy	Abreviatura de tres letras del nombre del día
a.m. ó p.m.	Ante-Meridem o Post-meridem
b.c. ó a.d.	Indicador para el año(antes de cristo o después de Cristo)

7.6.6.6 Otras Funciones

OTRAS FUNCIONES	
FUNCION	DESCRIPCION
DECODE (var,val1,cod1,val2,cod2,...,valor_por_defecto)	Si var es igual a cualquier valor de la lista devuelve el correspondiente código, en caso contrario se obtiene el valor por defecto. val debe ser un dato del mismo tipo que var
Greatest (Expr,Expr,....)	Devuelve el mayor valor de la lista
Least(Expr,Expr,...)	Devuelve el menor de la lista
Nvl(x,expr)	Si x es Null, devuelve expr, si no lo es, devuelve x. “x” y “Expr” puede ser de cualquier tipo. El tipo de valor resultante es el mismo que el de x • se utiliza para evitar los valores nulos en expresiones

	aritméticas y funciones • Los valores nulos en las expresiones siempre darán como resultado un valor nulo • Por lo Tanto, deberá utilizarse esta función cuando quieran evitarse las acciones descritas en los dos puntos anteriores

7.7 Tema 6: "Otros Lenguajes Relacionales"

TEMA N° 6

OBJETIVOS:

- Características del Lenguaje QBE
- Saber ejecutar consultas en una o varias tablas utilizando el lenguaje QBE
- Presentar los resultados de una consulta en el lenguaje QBE de manera agrupada en función de un determinado atributo
- Saber ejecutar consultas con agrupación y utilizando funciones de agrupación por medio del lenguaje QBE
- Ser capaz de modificar la base de datos con el QBE
- Conocer las características Generales del lenguaje Datalog
- Conocer la sintaxis básica del Lenguaje
- Saber generar consultas utilizando las operaciones básicas vistas en el álgebra relacional
- Profundizar en las características Generales del lenguaje Datalog
- Saber diseñar programas en datalog que utilizen recursividad
- Saber la importancia en el diseño final de la utilización de herramientas

CONTENIDO

- Query By example
  - Consultas sobre una relación
  - Consultas sobre varias relaciones
  - La relación Resultado
  - Presentación Ordenada de las tuplas
  - Operaciones de agregación
  - Modificaciones de la base de Datos
    - Inserción
    - Borrado
    - Actualización
- Datalog
  - Estructura Básica
  - Sintaxis de las Reglas de Datalog
  - Semántica de Datalog no Recursivo
  - Semántica de una Regla
  - Semántica de un Programa
  - Operaciones Relacionales en Datalog
- Datalog(Continuación)
  - la Recursividad en Datalog
  - La Potencia de la recursividad
  - Recursividad en Otros.lenguajes
  - Interfaces de Usuarios.y Herramientas
  - Los Formularios e Interfaces Graficas de Usuarios
  - Los Generadores de Informes
  - Las Herramientas de Análisis de Datos

DURACIÓN: 8 HORAS.

BIBLIOGRAFÍA:

- Silverschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ª Edición. McGraw Hill

En el tema anterior se ha descrito SQL, el lenguaje relacional de mayor influencia comercial. En este capíulo se estudiaran los aspectos básicos de dos lenguajes mas: QBE y Datalog. A diferencia de SQL, QBE es un lenguaje gráfico donde las consultas se asemejan a tablas. QBE y sus variantes se usan ampliamente en sistemas de bases de datos para computadoras personales. Por otro lado Datalog tiene una sintaxis derivada del lenguaje Prolog. Aunque actualmente no se utiliza de forma comercial, Datalog se ha utilizado en el desarrollo de diversos sistemas de bases de datos.
En este tema se presentan las constructoras y conceptos fundamnetales en lugar de un manual de usuario para estos lenguajes. Se debe tener presente que la implementación específica de cada len guaje puede diferir en los detalles, o puede dar soporte sólo a un sub conjunto del lenguaje completo.

7.7.1 Query By Example

Query-by-Example (QBE, consulta mediante ejemplos) es el nombre tanto de un lenguaje de manipulación de datos como el de un sistema de Base de Datos que incluyó a este lenguaje. El sistema de base de datos QBE se desarrolló en el centro de investigación T.J. Watson de IBM, a principios de los años setenta y el lenguaje de manipulación de Datos QBE se usó más tarde en QMF(Query Management Facility, mecanismo de gestión de consultas),también de IBM.Actualmente, muchos de los sistemas de bases de datos para computadoras personales soportan variantes del lenguaje QBE.

7.7.1.1 Características Distintivas

1. A Diferencia de muchos lenguajes de consulta y de programación, QBE presenta una **sintaxis bidimensional** en el sentido de que las consultas se asemejan a tablas. Una consulta en un lenguaje unidimensional (como SQL) se puede formular en una línea. Un lenguaje bidimensional necesita dos dimensiones para la formulación de sus consultas.

2. Las consultas QBE se expresan mediante un “ejemplo”. En lugar de incluir un procedimiento para obtener la respuesta deseada, se usa un ejemplo de qué es lo que se desea de forma tal que el sistema generaliza este ejemplo para así obtener la respuesta a la consulta.

A pesar de estas características tan poco comunes existe una correspondencia entre QBE y el cálculo relacional de dominios.
Las consultas QBE se expresan utilizando **esqueletos de tablas.** Estos esqueletos de tablas presentan el esquema de la relación, como se muestra en la siguiente figura 7.85

Sucursal	Nombre sucursal	Ciudad Sucursal	Activo

Cliente	Nombre Cliente	Calle Cliente	Ciudad Cliente

Préstamo	Número préstamo	Nombre Sucursal	**importe**

Prestatario	Nombre cliente	Número préstamo

Cuenta	Número cuenta	**Nombre Sucursal**	saldo

impositor	Nombre_cliente	Número_cuenta

Figura 7.86 Esqueleto de las Tablas QBE para un ejemplo bancario

En lugar de llenar la pantalla con esqueletos de tablas, el usuario elige los esqueletos que necesita para una determinada consulta y rellena dichos esqueletos con expresiones acordes a la consulta denominadas **filas ejemplo**.
De modo que una fila ejemplo está formada por constantes y elementos ejemplo que son variables de dominio. Para evitar confusiones, en QBE las variables de dominio van precedidas por un carácter de subrayado (_) como en _x, y las constantes aparecen sin ninguna indicación particular. Este convenio contrasta con la mayoría de los lenguajes, en los que las constantes se encierran en comillas y las variables aparecen sin ninguna indicación especial.

7.7.1.2 Consultas sobre una Relación

En función del ejemplo bancario en que se han basado los ejemplos pasados, para obtener todos los números de préstamo de la sucursal “linda Vista” se utilizará el esqueleto de la relación Préstamo y se rellenará así:

Préstamo	Número Préstamo	Nombre Sucursal	importe
	P._x	Linda Vista	

Figura 7.87 Salida de una consulta con Proyección y selección

La consulta anterior provoca que el sistema busque las tuplas en *prestamo* que tienen la ocurrencia Linda Vista en el atributo nombre_sucursal entonces para cada tupla de este tipo,el valor del atributo número_préestamo se asigna a la variable x, posteriormente este valor se imprime en pantalla debido a que la orden P.(Print) aparece en la columna número_préstamo junto a la variable x.
Expresión equivalente del álgebra relacional de Dominios:

$$\{ < x > \mid \exists s,c\ (< x,s,c > \in \text{préstamo} \wedge s=\text{"Linda Vista"})\}$$

QBE asume que una fila vacía contiene una variable única. Como resultado, si una variable no aparece más de una vez en una consulta, se puede omitir. La consulta anterior podría escribirse utilizando solamente la expresión P. en lugar de P._x.
QBE a diferencia de SQL realiza eliminación automática de duplicados, para evitar estas eliminaciones se debe insertar la cláusula ALL después de la orden P. así en el ejemplo anterior si deseáramos todas las ocurrencias debemos usar la expresión : P:ALL en el atributo número_préstamo. Para enlistar toda la relación se puede colocar el símbolo P. en cada uno de los atributos de la relación ó de una manera más comoda colocarlo debajo de la celda correspondiente al nombre de la relación.
QBE también permite formular consultas que conlleven comparaciones aritméticas así por ejemplo: Encontrar todos los números de préstamos de aquellos préstamos con una cantidad mayor de 10,000 córdobas

Préstamo	Número_Préstamo	Nombre_Sucursal	Importe
	P.		>10.00

Figura 7.88 Consulta QBE con omisión de variables

Las comparaciones solo pueden contener una expresión aritmética en el lado derecho de la operación de comparación así por ejemplo es valida la expresión:>(_x + _y – 20)
Donde como puede verse en el ejemplo las expresiones pueden contener tanto variables como constantes. Es de notar que el lado izquierdo del símbolo de comparación debe de estar vacío., esta restricción implica que no se pueden comparar dos variables de distinto nombre, veamos el siguiente ejemplo: Obtener los números de préstamos de todos los préstamos pedidos conjuntamente por Santos y Gómez.

Prestatario	Nombre_cliente	Número_préstamo
	Santos Gómez	P._x _x

Figura 7.89 Utilización del conector lógico “or” en la condición

Para ejecutar la consulta anterior el sistema localiza todos los pares de tuplas de la relación *prestatario* que coinciden en el atributo número_préstamo, donde el valor del atributo nombre_cliente es santos para una tupla y Gomez para la otra. Una vez localizadas dichas tuplas se muestra el valor del atributo número_préstamo.

En el cálculo relacional de dominios:

$$\{ < p > \mid \exists x\ (< x,p > \in \text{prestatario} \wedge x=\text{"Santos"}) \wedge \exists\ x\ (< x,p > \in \text{prestatario} \wedge x=\text{"Gomez"})\}$$

Como otro ejemplo considérese, la consulta "Obtener los nombres de todos los clientes que viven en la misma ciudad de Santos".

Cliente	Nombre Cliente	Calle Cliente	Ciudad Cliente
	P._x		-y
	Santos		-y

Figura 7.90 Uso de varias variables en niveles diferentes

7.7.1.3 Consultas Sobre varias Relaciones

QBE permite formular consultas que involucran varias relaciones distintas (de igual forma que el producto cartesiano o la reunión natural en el álgebra relacional). Las conexiones entre varias relaciones se llevan a cabo a través de variables, que obligan a algunas tuplas a tomar el mismo valor con ciertos atributos. Como ejemplo supóngase que se desea encontrar los nombres de todos los clientes que tienen un préstamo en la sucursal "Linda Vista". Esta consulta se puede formular de la siguiente manera:

préstamo	Número_préstamo	**Nombre__Sucursal**	importe
	_x	Linda	

Prestatario	Nombre Cliente	Número Préstamo
	P._y	

Figura 7.91 El Producto natural Utilizando el QBE

Como podemos observar el producto natural entre dos relaciones se efectúa de manera sencilla utilizando el QBE

Como otro ejemplo consideremos la consulta "Obtener el nombre de todos los clientes que tienen una cuenta en el banco pero que no tiene un préstamo en el mismo.

En QBE las consultas que expresan negación se representan con el símbolo ¬ (not). La consulta es la siguiente:

Impositor	Nombre Cliente	Número Cuenta
	P._x	

Prestatario	Nombre cliente	Número préstamo
¬	_x	

Figura 7.92 Consulta QBE que utiliza la cláusula not

7.7.1.4 Caja de Condición

Algunas veces es poco conveniente o imposible expresar todas las restricciones de las variables de dominio dentro de esqueletos de tablas. Para solucionar este problema, QBE incluye una caja de condición que permite expresar restricciones más generales sobre cualquiera de las variables de dominio. QBE permite que aparezcan condiciones lógicas en una caja de condición. Los conectore lógicos son :and,or. ejemplos:

Prestatario	Nombre cliente	Número préstamo
	_n	P. x

condiciones
_n=Santos or _n=Gomez

Figura 7.93 Consulta QBE que utiliza caja de condiciones

Correspondiente a listar Los números de préstamo correspondientes a los prestamos efectuados por Santos ó Gómez

Si se desea especificar condiciones en que estas estén conectadas por el conector lógico and (“y”) simplemente se listan en una fila diferente así por ejemplo la caja de condición que está a continuación indica la expresión lógica: $\_x \geq 1300 \wedge \_x \leq 1500$.

condiciones
_x ≥ 1300 _x ≤1500

Figura 7.94 Uso del conector lógico “y”

Consultas que Involucran los Términos: "Algunos" y "Todos"
Ejemplo:
Obtener todas las sucursales con activos superiores al activo de al menos (alguno) una sucursal con sede en Masaya.

Sucursal	Nombre sucursal	Ciudad Sucursal	Activo
	P._x	Masaya	**_y** **_z**

condiciones
_y >1* _z

Figura 7.95 Consulta QBE del tipo "mayor que alguno"

Si el predicado anterior hubiese indicado al menos dos esto se hubiese reflejado en la caja de decisiones transformándose la expresión en: _y >2* _z
Por otra parte el contenido de la caja de condición para la expresión lógica: saldo entre 1,300 y 2,000 pero que no sea exactamente igual a 1,500 estará dada por :

$$\_x=(\geq 1300 \text{ and } \leq 2{,}000 \text{ and } \neg 1500)$$

donde la variable _x representa al atributo saldo.
Con la conexión lógica or sucede algo similar: así para obtener todas las sucursales que tiene sede tanto en Managua como en Masaya, se escribirá en la caja de condiciones: _X=(Managua or Masaya) donde _x es la variable que representa el atributo sucursal.

7.7.1.5 La Relación Resultado

Todas las consultas que se han formulado hasta ahora tienen una característica común: Los resultados que se muestran aparecen en un único esquema de relación. Si el resultado de una consulta contiene atributos de varios esquemas, se necesita un mecanismo para mostrar el resultado deseado en una única tabla
Para lograr este propósito se puede declarar una relación temporal resultado que incluya todos los atributos del resultado de la consulta. Por otra parte para mostrar el resultado de la consulta basta con incluir la orden P. en el esqueleto de la tabla resultado.
Ejemplo: Considérese la consulta : "Obtener el nombre del cliente, el número de cuenta y el saldo de todas las cuentas en la Sucursal "Linda Vista", la solución a esta consulta estará dada en los siguientes pasos:

1. Se crea un esqueleto de tabla, denominado resultado, con los atributos: nombre_cliente,número_cuenta y saldo. El nombre del esqueleto de tabla recién creado (es decir, resultado) no debe existir previamente en la base de datos como nombre de otra relación.

2. Se escribe la consulta.

El detalle de la consulta se describe a continuación:

Cuenta	Número cuenta	Nombre Sucursal	saldo
	_y	Linda Vista	_z

Impositor	Nombre_cliente	Número_cuenta
	_x	_y

resultado	Nombre cliente	Número cuenta	saldo
P.	_x	_y	_z

Figura 7.96 Consulta QBE que utiliza Esquema de Salida

7.7.1.6 Presentación Ordenada de las Tuplas

La ordenación en el QBE se expresa mediante las ordenes AO (órden ascendente) y DO (órden descendente), las cuales deben de escribirse en las columnas apropiadas. Por otra parte la prioridad de ordenación se expresa mediante un número así por ejemplo AO(1) indicará que el atributo efectuado se ordenará de forma ascendente con una prioridad de ordenación 1 es decir será el atributo con la ordenación primaria.

Ejemplo:

Cuenta	Número Cuenta	Nombre Sucursal	Saldo
	P.AO(1).	Linda Vista	P.DO(2).

Figura 7.97 Consulta QBE ordenada por varios campos

Con lo cual se genera la consulta consistente en el listado de los números de cuenta ordenados de forma ascendente (órden primario), el saldo de ellas ordenados de forma descendente (orden secundario) siempre y cuando sean de la sucursal Linda Vista.

7.7.1.7 Operaciones de Agregación

QBE incluye los siguiente operadores de agregación: AVG, MAX, MIN, SUM y CNT. A todos estos operadores se les debe añadir la cláusula ALL, para crear un

multi conjunto en el que se llevan a cabo las operaciones de agregación. El operador.ALL asegura que no se eliminarán todos los duplicados. Así, para encontrar el saldo total de todas las cuentas de la sucursal Linda vista se escribirá:

Cuenta	Número cuenta	Nombre Sucursal	saldo
		Linda Vista	P.SUM .ALL

Figura 7.98 Consulta QBE que utiliza funciones de agregación

Se usa el operador UNQ, para especificar que se desean eliminar duplicados. Así, para encontrar el número total de clientes que tienen una cuenta en el banco se escribirá:

Impositor	Nombre Cliente	**Número Cuenta**
	P.CNT. UNQ.	

Figura 7.99 Consulta QBE con eliminación de duplicados

QBE ofrece también la posibilidad de generar consultas con agrupaciones respecto a un atributo ó conjunto de atributos, así por ejemplo si se desea obtener la media de los saldos agrupados por nombre de sucursal usaríamos el siguiente esquema:

Cuenta	Número Cuenta	Nombre Sucursal	saldo
		P.G.	P.AVG .ALL

Figura 7.100 Consulta QBE que utiliza agrupación

Donde se indica que la agrupación se efectuará con respecto al atributo Nombre_sucursal y que la función de agregación a utilizar será el promedio, se indica además que se tomarán en cuenta valores repetidos en el atributo saldo.
Si se desea efectuar condiciones sobre la relación resultante se debe hacer uso de una caja de condición así la expresión para listar solamente aquellos promedios mayores de 1200 es AVG.ALL._x >1,200 donde x es la variable que representa el doinio saldo.

7.7.1.8 Modificaciones de la Base de Datos

El borrado de tuplas de una relación se expresa del mismo que una consulta con la diferencia que se utiliza el D. en afectando la caja correspondiente al nombre de la relación.

Ejemplo1:
El siguiente esquema QBE borra de la tabla cliente todas aquellas tuplas en el nombre del cliente es “Santos”

Cliente	Nombre Cliente	Calle Cliente	Ciudad Cliente
D.	Santos		

Figura 7.101 Consulta de eliminación de tuplas del QBE

Ejemplo2:
Borrar todos los préstamos con una cantidad comprendida entre 1,300 y 1,500

Préstamo	Número_préstamo	Nombre_Sucursal	cantidad
D.	_y		_x

Prestatario	Nombre cliente	Número Préstamo
D.		_y

condiciones
_x=(≥1,300 and ≤1,500)

Figura 7.102 eliminación condicional de Tuplas en el QBE

Obsérvese que para borrar préstamos se deben borrar tuplas tanto de la relación *préstamo* como de *prestatario*

Inserción
Para insertar datos en una relación se necesita especificar la tupla que se va a insertar o escribir una consulta cuyo resultado sea el conjunto de tuplas que se van a insertar. La inserción se expresa mediante el operador I. Obviamente los valores de los atributos para las tuplas insertadas deben de ser miembro del dominio de los atributos.
La inserción más sencilla es la inserción de una única tupla.
Supóngase que se desea insertar la cuenta C-9732 en la sucursal Linda Vista con un saldo de 700. Se escribirá de la siguiente manera:

Cuenta	Número cuenta	Nombre Sucursal	saldo
I.	C-932	Linda Vista	700

Figura 7.103 consulta de Inserción de tuplas en el QBE

También se puede insertar una tupla con información parcial, en este caso los valores faltantes de la fila se ponen a null.
De manera más general, es posible insertar tuplas basadas en el resultado de una consulta. Considérese de nuevo la situación en la que se desea obsequiar a todos los clientes con préstamos en la sucursal Linda Vista con una nueva cuenta de ahorro por el monto de 200 por cada cuenta de ahorro que ellos tengan. Se escribirá la consulta de la siguiente manera:

Cuenta	Número Cuenta	Nombre Sucursal	saldo
I.	_x	Linda Vista	200

Impositor	Nombre cliente	**Número cuenta**
I.	_y	_x

Préstamo	Número préstamo	Nombre Sucursal	cantidad
	_x	Linda Vista	

Prestatario	Nombre cliente	Número Préstamo
	_y	_x

Figura 7.104 Inserción condicional de tuplas en el QBE

El esquema debe de leerse así: insertar para
_x= Prestatario.Número_Préstamo = Préstamo.Número_préstamo y la restricción: Préstamo.Nombre_Sucursal= ”Linda vista”
Efectuar las siguientes operaciones :
-y=Prestatario.Nombre_Cliente
Insertar en Cuenta la tupla: _x, “Linda Vista”, 200
Insertar en Impositor la tupla: _y, _x

Actualización
Existen situaciones en las cuales se desea cambiar un valor en una determinada tupla sin cambiar todos los valores de la misma. Para este propósito se utiliza el operador U.
Como ocurría con la inserción y el borrado, se pueden elegir las tuplas que se van a actualizar por medio de una consulta QBE, sin embargo, no permite que los usuarios actualicen los campos de la clave primaria.
Supóngase que se desea actualizar el valor del activo de la sucursal Linda Vista a 10,000,000. esta actualización se expresa de la siguiente manera:

Sucursal	Nombre sucursal	Ciudad Sucursal	Activo
	Linda Vista		U.10,000,000

Figura 7.105 Actualización incondicional de una relación utilizando QBE

Existen circunstancias, donde se necesita actualizar un valor utilizando el valor antiguo. Supóngase que se van a incrementar los saldos en un 5%. La expresión es la siguiente:

Cuenta	Número cue	Nombre Suc	saldo
			U._x*1.05

Figura 7.106 Actualización condicional en el QBE

7.7.2 Datalog

Es un lenguaje de consultas, no procedimental basado en el lenguaje de programación lógica Prolog. Así como se hace en el calculo relacional, el usuario describe la información que se desea sin especificar como se va a obtener esta información.

La sintaxis del Datalog se asemeja a la de Prolog. Sin embargo el significado de los programas en Datalog se definen de una manera puramente declarativa, a diferencia de la semántica más procedimental de Prolog. Datalog simplifica la escritura de consultas simple y hace que la optimización de consultas sea más sencilla.

7.7.2.1 Estructura Básica.

Un programa en Datalog consiste en un conjunto de reglas.
Ejemplo de una regla en Datalog.

Definir una relación de vistas N_1 que contiene los números de cuenta y los saldos de las cuentas de la sucursal 'Linda Vista', cuyos saldos sean superiores a 700 córdobas:
V_1 (C,S): - cuenta (C, 'Linda Vista', S), S > 700

En función de este ejemplo podemos notar que:
Las reglas de Datalog definen vistas y la expresión indicada arriba que define la vista, se lee:

Si (C, 'Linda Vista', S) está en cuenta y S > 700 entonces (C,S) está en V_1

Interrogación de las vistas.

Sea la siguiente instancia de la relación cuenta

Número_Cuenta	Nombre_Sucursal	Saldo
C-101	Sutiava	500
C-215	Jalteva	700
C-102	Linda Vista	400
C-305	Los Robles	350
C-201	Linda Vista	900
C-222	San Felipe	700
C-217	Linda Vista	750

Ejemplos

1. Para calcular el saldo de la cuenta C-217 en la vista N_1, se puede formular la consulta siguiente:

? V_1 (C-217,S)
Siendo el resultado
(C-217, 750)

2. Obtener el numero y el saldo de todas las cuentas de la vista V1 con saldo superior a 800 córdobas

? V_1 (C,S), S > 800
El resultado de la consulta anterior es
(C-201,900)

En general puede ser necesaria más de una regla para definir una vista. En este caso cada regla define un conjunto de tuplas que debe contener la vista. Así el conjunto de tuplas de la vista re refiere como la unión de todos esos conjuntos de tuplas.

Ejemplo:
Tipo – interés (C,5) : - cuenta (C, N, S), S < 10,000
Tipo – interés (C,6) : - cuenta (C, N, S), S > = 10,000

El programa tiene dos reglas que definen la vista tipo_interés, cuyos atributos son el Número_Cuenta y el tipo de interés.

Se lee $\forall$*, C, N, S Si* $\langle C,N,S\rangle \in$ *Cuenta ^ S < 10,000*
Entonces $\langle C,5\rangle \in$ tipo – interés

Si <C,N,S> ∈ Cuenta ^ S ≥ 10,000
Entonces <C,6> ∈ tipo – interés

Las reglas de Datalog pueden utilizar la negación.
Ejemplo:
Las reglas siguientes definen una vista C, que contiene los nombres de todos los clientes que tienen cuenta pero que no tienen ningún préstamo en el banco:

C(N): - impositor (N,C), not Es – prestatario (N)
Es –prestario (N) : - prestatario (N,P)

Para todo N,C
Si <N,C> ∈ impositor ^ N ∉ es – prestatario
Entonces N ∈ C(N)
Donde se especifica que
Es_prestatario (N) es la relación prestatario (N,P)

7.7.2.2 Sintaxis de las Reglas de Datalog

Un literal positivo tiene la forma
$P (t_1, t_2,... t_n)$
Donde P es el nombre de una relación con n atributos y $t_1, t_2,... t_n$ son constantes o variables.

Un literal Negativo tiene la forma
Not $P (t_1, t_2,... t_n)$

Ejemplo de Literal positivo:
Cuenta (C, “Linda Vista”, S)

Los literales que contienen operaciones aritméticas se tratan de un modo especial por ejemplo el literal B > 700 aunque no tiene la sintaxis descrita puede entenderse conceptualmente como > (B,700) que tiene la sintaxis requerida y donde > es una relación.

Se pueden definir otras literales con las operaciones aritméticas por ejemplo
A = B + C equivalente a + (B,C,A)
= {<X,Y,Z> / X ∈ B, Y ∈ C, Z ∈ A (Z= X +Y)}

Un hecho tiene la forma general
$P (V_1, V_2,... V_n)$

Implica que la tupla $< V_1, V_2,... V_n>$ está en la relación P. un conjunto de hechos de una relación se puede escribir en la forma habitual tabular.

Las reglas se construyen a partir de literales y tienen la siguiente forma:

$P (t_1, t_2,... t_n) : - L_1, L_2,... L_n$

Donde cada L_i es un literal positivo o negativo). El literal $P(t_1, t_2,... t_{n)}$ se denomina cabeza de la regla y el resto de literales constituyen el cuerpo de la misma.

Un programa Datalog consiste en un conjunto de reglas; el orden de las reglas es indiferente. Como se vio antes una relación se puede definir utilizando varias reglas.
Ejemplo de un programa Datalog.
Interés (C,I) :-cuenta (C,"Linda Vista",S),
Tipo_interés (C,T), I = S * T/100
Tipo_interés (C,5):-cuenta (C,N,S), S < 10,000
Tipo_interés (C,6):-cuenta (C,N,S), S > = 10,000

Una vista V_1 se dice que depende directamente de una vista V_2 si V_2 se usa en la expresión que define a V_1. Así en el ejemplo anterior la vista tipo_interés depende directamente de las relaciones tipo_interés y cuenta. A su vez la relación tipo_interés depende directamente de cuenta.

Una vista V_1 se dice que depende indirectamente de una vista V_2 si hay una secuencia de relaciones intermedias i_1, i_2,... i_n para algún n tal que V_1 depende directamente de i_1, a la vez i_1 depende directamente de i_2 y así sucesivamente hasta i_{n-1} que depende directamente de i_n.
En el ejemplo anterior la relación interés depende indirectamente de cuenta.

Se dice que una vista V es recursiva si depende de si misma.. Si no depende de si misma, se dice no recursiva.

Ejemplo:
Empl (X,Y) :-Jefe (X,Y)
Empl (X,Y) :-Jefe (X,Y), empl (Z,Y)

Este programa la vista empl depende de si misma por la segunda regla y por tanto es recursiva.

7.7.2.3 Semántica de Datalog no recursivo.

A continuación se considera la semántica formal de los programas Datalog referidos a programas no recursivos, la semántica de los programas recursivos se estudiará posteriormente.

Semántica de una Regla.
Un ejemplar básico de una regla es el resultado de sustituir cada variable de la regla por una constante. Si una variable aparece varias veces en una regla, todas las apariciones de la variable se deben sustituir por la misma constante. Los ejemplos básicos se llaman habitualmente ejemplares.

A continuación se muestra el ejemplo de definición de la vista V_1 y un ejemplar de la regla:
V1 (C,S):-cuenta (C, "Linda Vista", S), S > 700

Sea C= "C-217" y S = 750
Entonces un ejemplar de la regla en función de estos valores es:
V_1 ("C-217", 750) :-cuenta ("C-217", "Linda Vista", 750), 750>700

Normalmente una regla tiene muchos ejemplares posibles. Estos ejemplares se corresponden con las diversas formas de asignar valores a cada variable de regla.

Dada la regla R siguiente:

$P(t_1, t_2,... t_n)$:- $L_1, L_2,... L_n$ y un conjunto de hechos I asociados a las relaciones que aparecen en la regla.

Considérese cualquier ejemplar R' de la regla R:

$P(V_1, V_2,... V_n)$:-$L_1, L_2,... L_n$

Donde cada literal L_i es de la forma

$q_i (Vi_1, Vi_2,... Vi_{ni})$ ó

not $q_i (Vi_1, Vi_2,... Vi_{ni})$ y donde cada Vij y cada Vi es una constante

Se dice que el cuerpo de una ejemplar R' se satisface en I si

1. Para cada literal positivo $q_i (Vi_1, Vi_2,... Vi_{ni})$ del cuerpo de R', el conjunto de hechos I contiene el hecho $q_i (Vi_1, Vi_2,... Vi_{ni})$.

2. Para cada literal negativo not $q_i (Vi_1, Vi_2,... Vi_{ni})$ del cuerpo de R', el conjunto de hechos I no contiene el hecho $q_i (Vi_1, Vi_2,... Vi_{ni})$.

Se define el conjunto de hechos que se pueden inferir a partir de un conjunto de hechos I dado usando la regla R como:

Inferir (R,I) = { $P(t_1, t_2,... t_n)$ / existe un ejemplar R' de R, donde $P(t_1, t_2,... t_n)$ es la cabeza de R' y el cuerpo de R' se satisface en I }

Dado un conjunto de Reglas R = { $R_1, R_2,... R_n$} se define:
Inferir (R,I) = inferir (R_1,I) U inferir (R_1,I) U... U inferir (R_n, I)
Dado un conjunto de hechos I, que contiene las tuplas de la relación cuenta que se muestra en la figura 7.76 un posible ejemplar de la regla, ejemplo R sería:

V_1 (“C-217”,750) :-cuenta (“C-217, “Linda Vista”, 750), 750 > 700

El hecho cuenta (“C-217, “Linda Vista”, 750) pertenece al conjunto de hechos I. Además 750 es mayor que 700 y así conceptualmente, (750, 700) está en la relación “>”. Por tanto, el cuerpo del ejemplar de la regla se satisface en I. Existen otros posibles ejemplares de R y utilizándolos se comprueba que inferir (R,I) tiene exactamente el conjunto de hechos para V_1 que se muestra en la figura 7.79 a continuación.

Número_Cuenta	Saldo
C-201	900
C-217	750

Figura 7.107 Relación Cuenta

7.7.2.4 Operaciones Relacionales en Datalog.

Las expresiones de Datalog no recursivas sin operaciones aritméticas son equivalentes en poder expresivo a las expresiones que utilizan las operaciones básicas del álgebra relacional U, ~, x, σ, Π.

En lugar de efectuar una demostración general de este hecho, se mostrará mediante ejemplar como se pueden expresar en Datalog operaciones del álgebra relacional, en todos los ejemplares la vista generada se denominará consulta.

El caso de la selección ya fue abordado caso de la proyección

$$\Pi \text{ nombre\_cuenta (Cuenta)} \equiv \text{consulta (C) :-cuenta (C,N,S)}$$

es decir $< C> \in$ consulta $<=> <C,N,S> \in$ cuenta

caso del producto cartesiano

$$r_1 \times r_2 \equiv \text{consulta } (x_1,... x_n, y_1, y_2,... y_m) :- r_1 (x_1, x_2,... x_n), r_2 (y_1, y_2,... y_n),$$

$$\text{Es decir } < x_1, x_2,... x_n, y_1, y_2,... y_n > \in \text{consulta } <=>$$
$$< x_1, x_2,... x_n, > \in r_1 \wedge < y_1, y_2,... y_m > \in r_2$$

La unión de Relaciones $r_1 \cup r_2$

$$r_1 \cup r_2 \equiv \text{consulta } (x_1, x_2,... x_n):- r_1 (x_1,... x_n)$$
$$\text{consulta } (x_1, x_2,... x_n):- r_2(x_1,... x_n)$$

Lo cual equivale a:

$$< x_1, x_2,... x_n> \in \text{consulta } <=>$$
$$< x_1,... x_n> \in r_1 \vee < x_1, x_2,... x_n> \in r_2$$

El conjunto diferencia de dos Relaciones ($r_1 \sim r_2$)

$$r_1 \sim r_2 \equiv \text{consulta } (x_1, x_2,... x_n) :- r_1 (x_1,... x_n), \text{not } r_2 (x_1, x_2,... x_n)$$
$$\text{, lo cual es equivalente a:}$$
$$< x_1, x_2,... x_n> \in \text{consulta } <=> < x_1, x_2,... x_n> \in r_1 \wedge < x_1, x_2,... x_n> \notin r_2$$

En Datalog el operador renombramiento P no se necesita. Una relación puede aparecer más de una vez en el cuerpo de la regla, pero en lugar de renombrar para dar nombres diferentes a las apariciones de la relación, simplemente se utilizan diferentes nombres de variables en las diferentes apariciones.

Las operaciones relacionales extendidas de inserción, borrado y actualización son compatibles con ciertas extensiones en Datalog. La sintaxis para tales operaciones varía entre implementaciones distintas. Algunos sistemas permiten el uso de + ó – en la parte izquierda de las reglas para denotar la inserción y borrado relacional.

Ejemplo: Trasladar todas las cuentas de la sucursal Linda Vista a la sucursal Managua, ejecutando los comandos:

```
+ cuenta (C, “Managua”, S) :- cuenta (C, “Linda Vista”, S)
-cuenta (C, “Linda Vista”, S) :-cuenta (C, “Linda Vista”, S)
```

Que equivale a : ∀ <C, “Linda Vista”, S> ∈ cuenta
Insertar <C, ”Managua”, S>
∀ <C, “Linda Vista”, S> ∈ cuenta, eliminar <C, “Linda Vista”, S>

La recursividad en Datalog.

Diversas aplicaciones de Bases de Datos manejan estructuras similares a los árboles de datos.

Así por ejemplo considérense los ejemplares de una empresa, en ella algunos empleados son Jefes, cada jefe dirige un conjunto de personas cada una de las cuales puede ser jefe. De esta forma los empleados se pueden organizar en una estructura similar a la de un árbol.

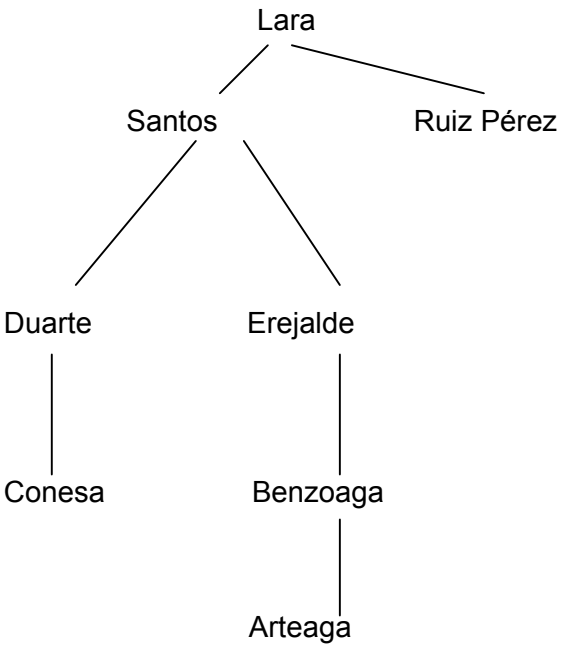
Supóngase el siguiente esquema de la relación jefe.
Esquema-jefe = (nombre-empleado, nombre-jefe)

En la figura 7.80 se tiene una instancia de esta relación

Nombre_empleado	Nombre_jefe
Arteaga	Benzoaga
Benzoaga	Erejalde
Conesa	Duarte
Duarte	Santos
Erejalde	Santos
Santos	Lara
Ruíz Pérez	Lara

Figura 7.108 Instancia de la relación Jefe

La cual obedece a la siguiente estructura de árbol



Supóngase ahora que se desea obtener listado de los empleados que supervisa (directa ó indirectamente) un determinado Jefe por ejemplo Santos es decir los empleados Duarte, Conesa, Erejalde, Benzoaga y Arteaga.

Utilizando la idea de la recursividad se puede definir el conjunto de personas dirigidas por Santos de la siguiente manera:

(1) Empl – Santos (x) : - Jefe (x, “Santos”)
(2) Empl – Santos (x) : - Jefe (x, y), empl-Santos (y)

El caso (1) es directo:

$< x > \in$ empl-santos $< = > < x,$ “Santos” $> \in$ Jefe

La regla (2) es la parte recursiva

$< x > \in$ empl-santos $< = >$

$\exists\ y < x, y > \in$ Jefe $\wedge\ < y > \in$ empl – Santos

En lugar de crear una vista para los empleados supervisados por Santos se puede crear la vista empl, más general, que contenga todas las tuplas $< x, y >$ tales que x sea directa o indirectamente supervisado por y, de la siguiente manera:

Empl (x,y) :- Jefe (x,y)
Empl (x,y) :- Jefe (x,z), empl (z,y)

Así para obtener los empleados supervisados directa ó indirectamente por Santos, se utilizará la consulta

? empl (x, “Santos”)

que devuelve para x el mismo conjunto de valores de la vista empl-Santos.

7.7.3 Interfaces de Usuarios y Herramientas

Aunque muchos usuarios interactúan con la base de datos, pocas usan lenguaje de consulta para interactuar directamente con un sistema de base de datos. La mayoría interactúan mediante uno de los siguientes medios:

1. **Los formularios e interfaces gráficas de usuario** permiten a los usuarios introducir valores que complementan las consultas predefinidas. El sistema ejecuta las consultas y da un formato apropiado mostrando los resultados al usuario. Las interfaces gráficas de usuario proporcionan una forma fácil de interactuar con el sistema de bases de datos. Estas herramientas se han vuelto un estándar de modo que la mayoría de sistemas de Gestión de Base de Datos cuentan con estas facilidades. En general la idea es que el usuario no introduzca los datos directamente en las tablas de la base de datos sino que lo haga de forma simple utilizando como interfaz de uno ó más formularios,

estos heredan las restricciones de integridad de las tablas y por otro lado estos permiten efectuar restricciones complejas en los atributos, Ofrecer un Formato adecuado al usuario para la introducción de los datos, ejecutar consultas, mejor presentación en cuanto a la forma de la entrada de los datos, facilita la comunicación entre los usuarios y la aplicación etc. En Nicaragua para aplicaciones de gestión pequeñas y medianas en general se utiliza como motor el del Sistema Gestor *access* y se utilizan las herramientas de visual basic para la gestión de los formularios.

2. **Los Generadores de Informes** Permiten generar informes predefinidos sobre los contenidos actuales de la base de datos. Los analistas examinan estos informes para la toma de decisiones. Estos informes están basados sobre la información contenida en las Tablas y vistas de la aplicación con el fin de presentarla en un formato adecuado de fácil interpretación para los usuarios

3. **Las Herramientas de Análisis de Datos** permiten a los usuarios examinar y analizar los datos de forma interactiva

Se debe enfatizar que estas interfaces usan lenguajes de consulta para comunicarse con los sistemas de Bases de Datos. Es un problema muy fuerte la no estandarización de estas herramientas, cada sistema de base de datos proporciona su propia interfaz para el usuario en general distinta a los otros sistemas.

7.7.3.1 Formularios e Interfaces gráficas de Usuarios

Las interfaces de formularios se usan ampliamente para introducir y extraer datos en la base de datos mediante consultas predefinidas. Por ejemplo, los motores World Wide Web proporcionan formularios que se utilizan para introducir palabras clave. Al pulsar el botón “Enviar” se provoca que el motor de búsqueda ejecute una consulta usando la palabra clave introducidas y mostrando posteriormente el resultado al usuario.

Como otro ejemplo más orientado a las bases de datos, es posible conectarse a un sistema de matrícula de una universidad, donde se pide que se rellene un código y una contraseña en un formulario. El Sistema usa esta información para comprobar la identidad, así como para extraer de la base de datos la información pertinente como el nombre y las asignaturas en las que el alumno se ha matriculado, y mostrarla. Puede haber más vínculos en la página web que permitan buscar asignaturas y encontrar más información acerca de las asignaturas como los programas y los profesores que ofrecerán las asignaturas.
Los exploradores Web compatibles con HTML constituyen los formularios e interfaces gráficas de usuario más ampliamnete utilizados actualmente. La mayoría de fabricantes de sistemas de bases de datos también proporcionan interfaces de formularios propietarias que ofrecen características más allá de las incluidas en los formularios HTML.
Los Programadores pueden crear formularios e interfaces gráficas utilizando HTML o lenguajes de programación tales como el C o Java. La mayoría de los fabricantes de Sistemas de Gestión de Bases de Datos sin embargo también proporcionan herramientas que simplifican la creación de formularios de una forma declarativa sencilla, usando programas que editan formularios. Los usuarios pueden definir el tipo, el tamaño de cada campo de un formulario

usando el editor de formularios. Las acciones del sistemas se pueden asociar con las acciones de los usuarios tales como rellenar un campo pulsar una tecla de función en el teclado o enviar un formulario. Por ejemplo, la ejecución de una consulta para rellenar el los campo de nombre y dirección se pueden asociar con rellenar un campo de código, y la ejecución de una instrucción de actualización se puede asociar con el la carga de un formulario.

Se pueden realizar comprobaciones de errores sencillas definiendo instrucciones sobre los campos del formulario (Conocidos en Oracle como disparadores de formulario). Por ejemplo, una restricción sobre el campo número de asignatura podría comprobar que el número de asignatura escrito por el usuario corresponde realmente con una determinada asignatura. Aunque estas restricciones se pueden comprobar cuando se ejecuta la transacción, la detección temprana de errores ayuda a los usuarios a corregir de forma rápida los errores. Las listas que indican los valores válidos que se pueden escribir en un campo eliminan la posibilidad de errores que comenten los usuarios al introducir los datos. Los desarrolladores de sistemas encuentran que su trabajo es mucho más fácil cuando se tiene la posibilidad de controlar tales características de forma declarativa con la ayuda de una herramienta de desarrollo de interfaces de usuario, en lugar de crear el formulario directamente usando un lenguaje de guiones o de programación.

7.7.3.2 Los Generadores de Informes

Estas son herramientas que generan informes de resumen legibles de una base de datos en función de las tablas y consultas de ella. En general los generadores de informes tiene la posibilidad de resumir los datos de la base datos mediante el uso de gráficos estadísticos, mejorar la salida por medio de texto con un determinado formato. Por ejemplo, un informe podría mostrar las ventas totales de los dos meses anteriores para cada zona donde se producen las ventas.

El desarrollador de aplicaciones puede especificar formatos de informes usando las características de formato del generador de informes. Se pueden utilizar variables para alacenar parámetros tales como el mes y el año así como para definir determinados campos del informe. Se pueden definir tablas, gráficas de barras, gráficas de pastel u otros tipos de gráficos utilizando consultas o tablas de la base de datos. Las definiciones de las consultas pueden utilizar parámetros cuyos valores específicos pueden ser tomados del contenido de las variables.

Una vez que se ha definido el formato del informe con un generador de informes este se puede almacenar y ejecutar este formato cada vez que se genere un informe. Los sistemas generadores de informes proporcionan varias características para estructurar una salida tabular, tal como definir las cabeceras de tabla y columna, mostrar sub-totales por cada grupo en una tabla, dividir automáticamente una tabla en varias páginas y mostrar sub-totales al final de cada página. A continuación se muestra un ejemplo de reporte.

Compañía de Seguros Acme S.A
Informe Trimestral de Ventas

Período: 1 de Enero a 31 de Marzo del 2004

Región	Categoría	Ventas	Subtotal
Norte	HardWare	1,000,000	
	SoftWare	500,000	
	Todas Las Categorías		1,500,000
Sur	HardWare	200,000	
	SoftWare	400,000	
	Todas las categorías		600,000

Ventas Totales: 2,100,000

Este ejemplo muestra a un informe con formato. Los datos del informe como puede verse se agrupan en función de los datos recopilados en la base de datos relacionados con los pedidos, generando también sub-totales y totales.

La colección de herramientas de desarrollo de aplicaciones proporcionadas por los sistemas de bases de Datos tales como los paquetes de formularios y los generadores de informes se conocen como lenguajes de cuarta generación (L4G). El nombre resulta dado que estas herramientas ofrecen un paradigma de la programación diferente del paradigma de programación imperativa ofrecida por los lenguajes de tercera generación como Pascal y C. sin embargo, este término es menos relevante actualmente, dado que los formularios y los generadores de reportes se crean normalmente con herramientas gráficas en lugar de utilizar lenguajes de programación.

Contenido Práctico

8 PRÁCTICAS DE LABORATORIOS

Introducción

Para complementar los conocimentos de la teoría se han diseñado nueve prácticas de laboratorio en ocho de las cuales se ha utilizado el SGBD Microsoft Access y en la novena Microsoft SQL Server. Las primeras ocho prácticas tienen como objetivo familiarizar al estudiante con los aspectos básicos necesarios para diseñar una base de datos, haciendo uso además de las herramientas de Microsoft Access para la introducción de los datos, la generación de reportes y el diseño de los menú. Estas ocho prácticas se han articulado mediante una aplicación denominada “Mi Ventesita” con el fin de que los conocimientos prácticos contenidos en las prácticas se vinculen, para lograr finalmente el diseño de la aplicación ejemplo. La práctica nueve tiene como objetivo el introducir al estudiante a un sistema gestor de base de datos de mayor potencia(MicroSoft SQL Server), el cual es generalmente utilizado para diseñar con él la base de datos de las aplicaciones y los datos son manipulados utilizando las herramientas de MicroSoft Access.

Criterios de Evaluación

Como ya se indicó anteriormente el valor de las prácticas será de un 40% del total de cada una de las calificaciones (Primero y Segundo Parcial). Las nota de las prácticas tendrá dos componentes :

1. **Cumplimiento en la entrega y calidad del trabajo**

para ello pueden conformarse grupos de hasta un máximo de tres estudiantes

2. **Pruebas Escritas**

Se harán un total de tres test escritos en función de los aspectos abordados en las prácticas correspondientes a los parciales y el Final, estos se diseñarán de modo que el estudiante haga uso del computador para contestar los respectivos test. El valor relativo de cada uno de los criterios de evaluación antes indicados será del 20%.

Duración del Laboratorio

Los laboratorios están diseñados como ya se dijo anteriormente con una frecuencia de 2 horas semanales, al inicio de cada práctica el profesor responsable del laboratorio explicará el contenido de la práctica y deberá estar presente para aclarar dudas que presenten los estudiantes durante las dos horas que dura cada una de las sesiones de laboratorio. Los laboratorios se diseñarán tomando en cuenta la posibilidad de trabajar en forma independiente de los estudiantes.

Enunciados de las prácticas

Los enunciados de las prácticas estarán a disposición de los estudiantes desde el primer día de clases ya sea en el sitio web del profesor o por medio de fotocopias del material en la secretaría del departamento.

Software a Utilizar

El SGBD comercial que se utilizará en los diferentes laboratorios es Access 2000

Sobre la evolución de las prácticas y futuras propuestas

Las prácticas serán revisadas en cada año académico para mejorarlas y actualizarlas, cambiando si es pertinente, las aplicaciones utilizadas y el contenido de los mismos. Asimismo deberán revisarse los mecanismos de evaluación en función de los analisis efectuados en el departamento.

8.1 Planificación Temporal

Como ya se ha indicado la frecuencia de los laboratorios es de 2 horas semanales.Tomando en cuenta la experiencia podemos establecer en 14 el número de laboratorios o encuentros reales para los laboratorios de la asignatura Base de Datos I (14 Semanas).

El número de horas asignadas a cada una de las prácticas están en dependencia de la complejidad de las mismas. Se pretende con estas prácticas el relacionar al máximo la parte teórica de la asignatura con las prácticas de laboratorio tomando en sí en cuenta el desfase natural que existe entre ambos aspectos debido a que existen partes de la teoría que no requieren de prácticas, este aspecto se resuelve en parte ya que cada una de ellas debe contar una introducción teórica lo cual ayudará a diseñar prácticas con una mayor independencia de los contenidos teóricos. La problemática de este último aspecto se debe a la independencia relativa que ha existido entre los laboratorios de las asignaturas y la parte teórica de las mismas esta discontinuidad de los conocimientos (teóricos-Prácticos) es mi criterio que afectan el entendimiento global de la asignatura por parte de los estudiantes, es claro que hacer lo contrario requiere de un esfuerzo especial de coordinación, lo ideal entonces sería el no dividir las asignaturas en teoría y prácticas de laboratorio sino que hacer prácticas cuando se requieran para afianzar los conocimientos teóricos. En resumen se podría decir que quizás sea lo más adecuado pero no es práctico ya que entre otras cosas no se optimizarían el uso de los locales para los laboratorios. Este comentario resulta de la experiencia de observar que dos aspectos que deberían estar estrechamente relacionados se ven como que si fuesen dos aspectos diferentes.

En la figura 8.1 se muestra la duración de las prácticas y su relación con la teoría.

Semana	Teoría	Práctica	Semanas necesarias
1	Presentación de la asignatura (2 hrs)		
1 2	Introducción	Diseño de tablas (énfasis en máscaras de entrada)	1
3 4	Modelo Entidad relación	Uso de la Herramienta “Relaciones” de access y Diseño de formularios(Introducción)	1
5 6 7	El modelo relacional	Formularios con sub formularios	2
8 9 10	SQL	Formularios con sub formularios (Continuación)	2
11	Otros Lenguajes relacionales	Importación de Datos Externos	2
12 13 14	Diseño de Bases de Datos	Diseño del Menú Principal de la Aplicación	2
		Diseño de Consultas	2
		Diseño de Informes	2
		Introducción al SQL Server	2

Figura 8.1 Duración de las prácticas y relación de los aspectos teóricos

8.2 Propuesta de prácticas

8.2.1 Práctica 1: Diseño de tablas (énfasis en máscaras de entrada)

En esta práctica se diseñarán las tablas básicas a utilizar en la aplicación que se denominará “Mi Ventesita”, utilizando los diferentes tipos de datos soportados por MicroSoft Access, reglas de validación y formato adecuado para cada uno de los atributos. Se hará énfasis de forma particular en las máscaras de entrada.

OBJETIVO DE LA PRACTICA

- Saber Crear una base de Datos
- Utilizar la Ventana de la Base de Datos
- Crear Tablas utilizando la herramienta de Access y diseñar los atributos con los formatos adecuados

INTRODUCCIÓN TEÓRICA

Concepto de Base de Datos y Tabla

Es una colección de archivos relacionados donde los archivos que componenen la base de Datos se denomina Tabla. Las tablas están conformadas por columnas y filas. Las columnas generalmente reciben el nombre de atributo y cada fila representa un objeto del tipo entidad o relación, de modo que para diseñar las tablas adecuadamente estas deben de estar adecuadamente clasificadas en Conjuntos entidades y conjuntos relaciones(Diseño Conceptual), así como los vínculos entre ellas(Diseño Lógico). MicroSoft Access como lo comprobará el estudiante cuenta con herramientas eficientas y sencillas tanto para la creación de bases de datos como para el diseño de tablas.

Creación de una nueva tabla en MicroSoft Access

Después de diseñar la tabla sobre el papel, es preciso crear el diseño en Access, que conlleva las etapas siguientes:

1. Crear una nueva Base de Datos (Archivo mdb)
2. En la ventana de la Base de Datos Escoger Tablas
3. Insertar el nombre, tipo, y descripción de cada campo
4. Establecer las propiedades para cada campo
5. Fijar una clave Principal (Obligatorio si se trata de un conjunto identidad)
6. Crear los indices(Opcional)
7. Guardar el diseño

Pasos Específicos en MicroSoft Access

2. Se ha creado una nueva base de datos de nombre “**Ejemplo de Tabla**” (ver figura abajo)

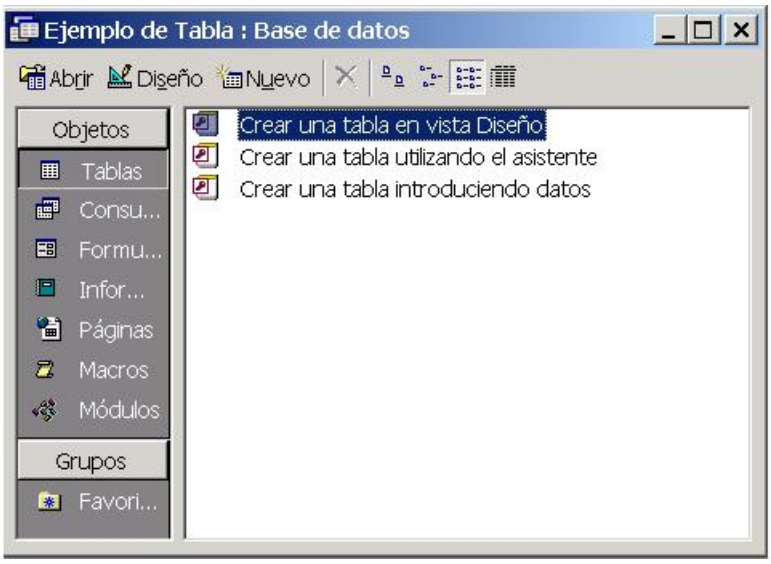


Figura 8.2 Cuadro de Diálogo inicial para crear una tabla

3. Escoger del cuadro de diálogo la opción "Crear una tabla en vista diseño"
Esta acción nos lleva al ambiente de diseño de la Tabla (ver figura 8.3 abajo)
En la cual diseñamos las propiedades de los atributos así:
El Nombre en "**Nombre del Campo**"
En "**Tipo de Datos**" se define el dominio del atributo, siendo los principales:

Texto: Puede contener hasta 255 caracteres, siendo el tamaño predeterminado de 50

Memo: Almacena grandes cantidades de texto, hasta 64,000 caracteres

Numérico: Almacena datos numéricos, los posibles tipos específicos entre los cuales el usuario puede escoger son: Entero Largo, Entero, Byte, Simple, doble, Id. de Replica y Decimal.

Fecha/Hora: posibilita el almacenamiento de fechas, permitiendo validar fechas y horas de modo automático y realizar cálculos basados en las entradas, como el número de días transcurridos entre fechas, etc

Moneda: En este tipo de datos Microsoft Access añade automaticamente un número fijo de dígitos a la derecha de la coma decimal, para evitar errores de redondeo.

Autonumérico: Incrementa de forma automática un valor numérico para cada registro que se agregue a la tabla

Si/No: Almacena valores lógicos. Durante la entrada de datos los campos pueden contener Verdadero o Falso, Si o No, 0 ó 1.

Objeto OLE: Cotiene datos OLE almacenados en otras palicaciones de Windos que soporten OLE, como video clips, sonidos etc.

Hipervínculo: Vincula a un recurso de Internet. Guarda combinaciones de texto y números como texto y usadas como una dirección hipervínculo.

Asistente de Búsqueda: Permite restringir el tipo de campo para que este solamente acepte datos de una lista de valores o de un campo de otra tabla

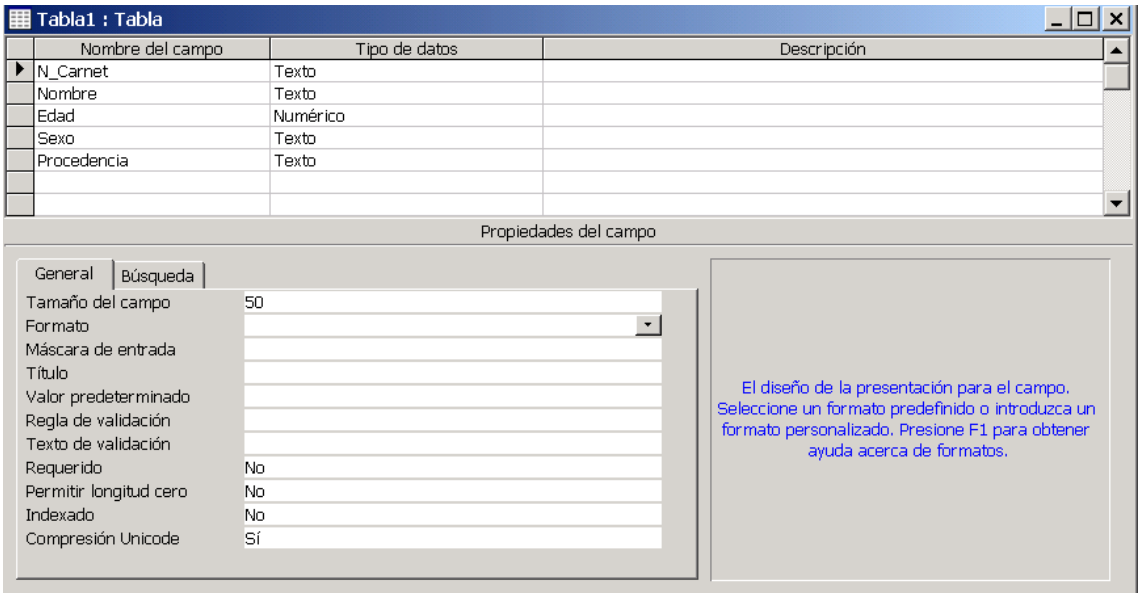


Figura 8.3 Ambiente de diseño de la Tabla

4. Establecer las propiedades para cada campo

Las diferentes propiedades de un campo se indican en la figura 8.3 arriba, estos son:

Tamaño del campo: Especifíca la longitud máxima para campos de texto. Con campos numéricos, los límites son los valores permisibles.

Formato: Determina la forma de presentación de los datos. Se puede elegir un formato predeterminado o crearlo.

Máscara de Entrada: Especifíca un patrón que deben de seguir los datos entrados en el campo.

Título: especifica una etiqueta predeterminada, distinta del nombre del campo, que aparecerá en formularios e informes

Valor Predeterminado: Especifica el valor que aparecerá automaticamente en el campo cuando se agreguen los registros.

Regla de Validación: especifica reglas de entrada de datos que hay que seguir para que el campo acepte un dato.

Texto de Validación: Especifica el mensaje que aparecerá en la barra de estado, si los datos no cumplen la regla de validación.

Requerido: Especifica si un campo es obligatorio.

Permitir Longitud cero: Especifica si se permiten cadenas vacías.

Indexado: nombra un índice de un solo campo que se añade al campo. Útil particularmente para acelerar búsquedas.

Compresión Unicode: Utilidad de Compresión de datos utilizada para los campos memo

5. **Creación de una Clave única**
Si no se designa un campo como clave principal, Access crea un campo Autonumérico y lo añade al principio de la tabla. Este campo contiene un número único para cada registro de la tabla, pero conviene que sea el usuario quien lo genere por las siguientes razones:

- Una Clave Principal es un índice
- Los índices mantienen un orden preclasificado de uno o más campos, lo cual acelera las consultas, búsquedas y clasificaciones.
- Cuando se añaden registros a la tabla, Access verifica los datos duplicados y no los permite en el campo clave principal
- Access muestra la información en el orden que determina la clave principal.

Creación de una Clave Principal
Existen tres formas de crear la clave principal:

- Seleccionar el campo a utilizar como clave principal y elegir: Edición, Clave principal
- Seleccioar el campo a utilizar como clave principal y luego elegir el botón Clave Principal (icono de Llave) en la barra de herramientas.
- Seleccionar el Campo a utilizar como clave principal y posteriormente hacer clic con el botón secundario y elegir Clave Principal en el menú contextual.

6. Para diseñar un índice se debe de invocar la ventana de índices:
Ver, índices en este ambiente se especifican las propiedades. La generación de índices es opcional ya que aunque acelera procesos de búsqueda, cada vez que utilizamos esta opción se genera un archivo en el sistema.

Máscaras de entrada.
Una máscara de entrada se utiliza en campos (en tablas y consultas) y en cuadros de texto y cuadros combinados (en formularios) para dar formato a los datos y para proporcionar algún **control sobre qué valores pueden introducirse**. Una máscara de entrada está formada por caracteres literales (como espacios, puntos, guiones y paréntesis) y por los valores que suministra el usuario.

Definición
La definición de máscara de entrada puede contener hasta **tres secciones** separadas por signos de punto y coma, por ejemplo: *(999) 000-0000!*;0;*" "*.

Sección	Significado
Primera	La más cara de entrada en sí.
Segunda	Define si deben almacenarse los caracteres literales mostrados. 0 = almacenar los caracteres literales con el valor introducido. 1 o espacio en blanco = almacenar únicamente los caracteres introducidos en los espacios en blanco.
Tercera	Carácter que se muestra para los espacios en blanco en la más cara de entrada. Puede utilizarse cualquier carácter; escriba " " (dobles comillas, espacio) para mostrar un espacio. Si se deja esta sección en blanco, se utiliza el subrayado (_).

Caracteres válidos de máscaras de entrada

Microsoft Access interpreta los caracteres de la primera sección de la definición de la propiedad Más cara de entrada tal como se muestra en la sección siguiente de caracteres de función de más cara. Para definir un carácter literal deberá introducirse cualquier otro carácter no incluido en esta sección, incluidos espacios y símbolos. Para definir un carácter como un carácter literal, deberá anteponer a dicho carácter una barra diagonal inversa (\) o si es un grupo ponerlo entre comillas dobles.

Carácter	Descripción
0	Dígito (0 a 9, entrada obligatoria; signos más [+] y menos [-] no permitidos).
9	Dígito o espacio (entrada no obligatoria; signos más y menos no permitidos).
#	Dígito o espacio (entrada no obligatoria; las posiciones en blanco se convierten en espacios; se permiten los signos más y menos).
L	Letra (A-Z, entrada obligatoria).
?	Letra (A-Z, entrada opcional).
A	Letra o dígito (entrada obligatoria).
a	Letra o dígito (entrada opcional).
&	Cualquier carácter o un espacio (entrada obligatoria).
C	Cualquier carácter o un espacio (entrada opcional).
<	Convierte a minúsculas todos los caracteres que siguen.
>	Convierte a mayúsculas todos los caracteres que siguen.
!	Hace que la máscara de entrada se muestre de derecha a izquierda, en lugar de hacerlo de izquierda a derecha. Los caracteres escritos en la máscara siempre la rellenan de izquierda a derecha. Puede incluir el signo de exclamación en cualquier lugar de la más cara de entrada.

Carácter	Descripción
\	Hace que el carácter que viene a continuación se muestre como carácter literal. Se utiliza para presentar cualquiera de los caracteres detallados en esta tabla como caracteres literales (por ejemplo, \A se muestra sencillamente como A).
Contraseña	Al establecer la propiedad Máscara de entrada (InputMask) a la palabra Contraseña, se crea un cuadro de texto de entrada de contraseña. Cualquier carácter escrito en este cuadro de texto se almacena como tal, pero se muestra como un asterisco (*).

Ejemplos de máscaras de entrada

Máscara	Ejemplo de entrada válida
(000) 000-0000;0;_	(206) 555-0248
(999) 999-9999! ;0;_	(206) 555-0248
(000) AAA-AAAA;0;_	(206) 555-TELE
>L????L?000L0;0;_	GREENGR339M3
>LL00000-0000;0;_	DB51392-0493
000\-000000\-0000>L;0;#	001-241076-0006D

DESARROLLO DE LA PRÁCTICA

Actividad1: Diseñar con la herramienta de access BD y diseñe las siguientes tablas con las características que se muestran a continuación.

Nombre del campo	Tipo de datos	Descripción
NombreCliente	Texto	25 (Nombre y Apellido): Todos a mayúsculas.
Direccion	Texto	50: Primera mayúscula resto minúsculas.
Telefono	Texto	8: 3dígito-4dígitos, ej-> 311-3356
E_Mail	Texto	50: Forzar minúsculas 1Letra23Opcional@25Opcional

Figura 8.4 Tabla: Cliente

	Nombre del campo	Tipo de datos	Descripción
🔑	NoFactura	Autonumérico	Entero Largo: Sin máscara.
	Fecha	Fecha/Hora	Fecha Corta: 2DígitosObl/2DígitosObl/4DígitosObl
	TipoPago	Sí/No	Efectivo/Tarjeta: Sin máscara.
	NombreVendedor	Texto	25(Nombre y Apellido): Convertir a mayúsculas.
	NombreCliente	Texto	25(Nombre y Apellido): Convertir a mayúsculas.
	MontoTotal	Moneda	Decimal,2: Sin máscara.

Figura 8.5 Tabla: Factura

	Nombre del campo	Tipo de datos	Descripción
🔑	IdProd	Texto	10: 3LetrasOblig-5DígitosOblig1LetraOblig.
	NombreProd	Texto	25: Convertir a mayúsculas.
	PrecioCompra	Moneda	Decimales,2: Sin máscara. Regla:mayor a 0.
	PrecioVenta	Moneda	Decimales,2: Sin máscara. Regla: mayor a 0.
	Existencias	Numérico	Entero: 4DígitosOp. Regla: 1..10,000.
	UnidadMedida	Texto	7: Sin máscara. Regla: "LIBRA", "CAJA", "PAQUETE", "UNIDAD", "GRAMO", "LITRO".
	Caduca	Sí/No	Si caduca llenar FechaCaducacion: Sin máscara.
🔑	FechaCompra	Fecha/Hora	Fecha Corta: 2DígitosObl/2DígitosObl/4DígitosObl
	FechaCaducacion	Fecha/Hora	Fecha Corta: 2DígitosObl/2DígitosObl/4DígitosObl

Figura 8.6 Tabla: Producto

	Nombre del campo	Tipo de datos	Descripción
🔑	NombreVendedor	Texto	25: Convertir mayúsculas.
	FechaNac	Fecha/Hora	Fecha Corta: 2DígitosObl/2DígitosObl/4DígitosObl.
	Direccion	Texto	50: Primera a mayúsculas.
	Telefono	Texto	8, Ej:311-3356
	Casado	Sí/No	Casado/Soltero
	FechaIngreso	Fecha/Hora	Fecha Corta: 2DígitosObl/2DígitosObl/4DígitosObl.
	Antiguedad	Numérico	Byte: 2DígitoOp.
	ContraseñaID	Texto	20, máscara de contraseña

Figura 8.7 Tabla: Vendedor

8.2.2 Práctica 2: Diseño del esquema relacional y Formularios

En esta práctica se diseñará un formulario sencillo basado en una sola tabla haciendo uso de las propiedades de los formularios y del control de ficha de la caja de herramientas.

OBJETIVO DE LA PRACTICA

- Saber Diseñar un esquema relacional a partir del Diagrama E/R
- Utilizar las diferentes opciones de la herramienta “Relaciones”
- Conocer los componentes básicos de un formulario y de su caja de Herramientas
- Saber utilizar el control de fichas dentro de un formulario
- Saber utilizar en un formulario los atributos capaces de almacenar archivos gráficos
- Saber incorporar los atributos de una tabla al Formulario

INTRODUCCIÓN TEÓRICA

Definición del Esquema relacional

Antes de definir el esquema relacional vamos a realizar algunos cambios al ejemplo anterior. Recuerde que el **ámbito** del problema en el que estamos trabajando, es una **pequeña venta** que lleva el control de sus clientes y los vendedores que realizan las facturaciones de productos, de forma que se lleva el control de las entradas y salidas de mercancía con la que trabaja el negocio. Los cambios son los siguientes:

Modificar el esquema anterior en lo siguiente

- **Producto** (Idprod, NombreProd, PrecioVenta, Existencias, UnidadMedida), [FechaCompra, Caduca, FechaCaducacion, PrecioCompra] moverlos a:
- **EntradaProducto** (IdProd, FechaCompra, Caduca, FechaCaducacion, PrecioCompra). Sin PK.
- De **DetalleFactura** quite el campo PrecioUnitario.

Estableciendo las relaciones.

Se entiende por establecer las relaciones el diseñar el esquema relacional, el cual està basado en el diseño lógico, es decir en nuestro caso en el respectivo Diagrama E/R del sistema, lo cual implica el establecer de forma clara y precisa las reglas del negocio de la empresa o institución donde se aplicará el sistema.
La semántica del sistema al cual se le diseñerá el esquema relacional es el siguiente:
En la empresa “Mi Ventesita” se desea llevar control de las facturas, estas tiene un encabezado al que llamaremos Factura donde se recogerá la siguiente información: El código de la factura (NoFactura), la fecha en que se efectuó la compra o emisión de la factura(Fecha), el Tipo de pago, ya que el cliente puede pagar en varias modalidades (Tipopago), El nombre de la persona que realizó la venta(NombreVendedor),Nombre del cliente que realizó la compra (NombreCliente) y el monto total de la Factura(MontoTotal).

Cada Factura consta de un detalle de factura donde se indican los artículos comprados, los atributos de esta relación son los siguientes:IdProd, NombreProd, Cantidad.

Además se desea llevar un control de la entrada de los productos es decir los productos que son adquiridos por la empresa. Esta relación (EntradaProducto) consta de los siguientes atributos: IdProd, FechaCompra, Si caduca o no (Caduca), FechaCaducación,PrecioCompra.

Según los requerimientos indicados anteriormente, el primer esbozo de un diagrama E/R sería el siguiente:

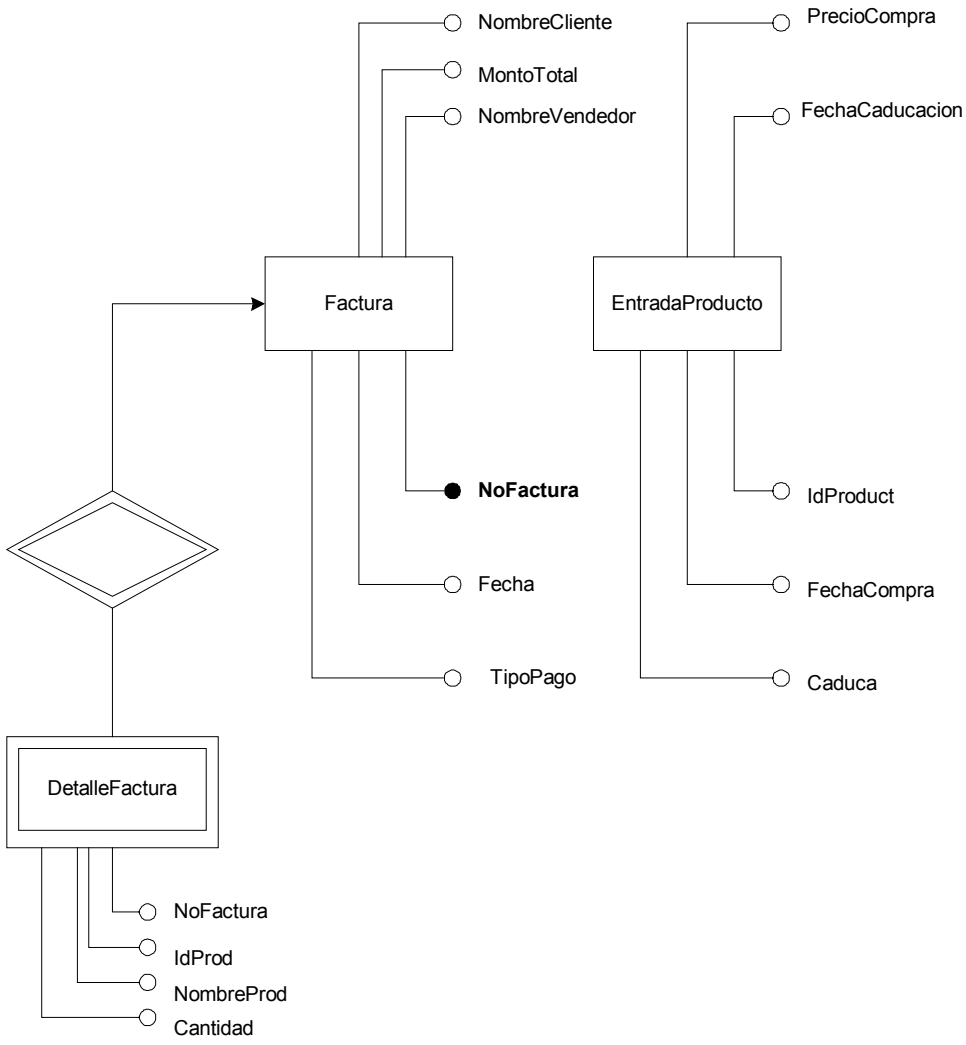


Figura 8.8 Primer Diagrama E/R "Mi Ventesita"

En este diagrama E/R básico, observamos que **EntradaProducto** no es un conjunto entidad sino que un conjunto relación (una entidad débil) que debe complementar la información del atributo IdProducto que también complementa la información del Detalle de factura, por los requisitos del usuario y el análisis del sistema se requiere conocer del objeto producto la siguiente información: El nombre del Producto (NombreProd), el precio de Venta (PrecioVenta), Existencias y la unidad de Medida del producto (UnidadMedida). Es de notar que no es adecuado añadir los atributos antes indicados en las tablas EntradaProducto y DetalleFactura debido a que son conjuntos relaciones y se produciría entre otros efectos negativos el de la redundancia de información, la solución es añadir el conjunto entidad Producto, al cual consultarán los conjuntos relaciones DetalleFactura y EntradaProducto. Con el mismo argumento, se van a crear los conjunto entidades: Vendedor y Cliente, las cuales complementarán la información del conjunto de Entidades Factura.
Una versión más acabada del diagrama E/R es la siguiente:

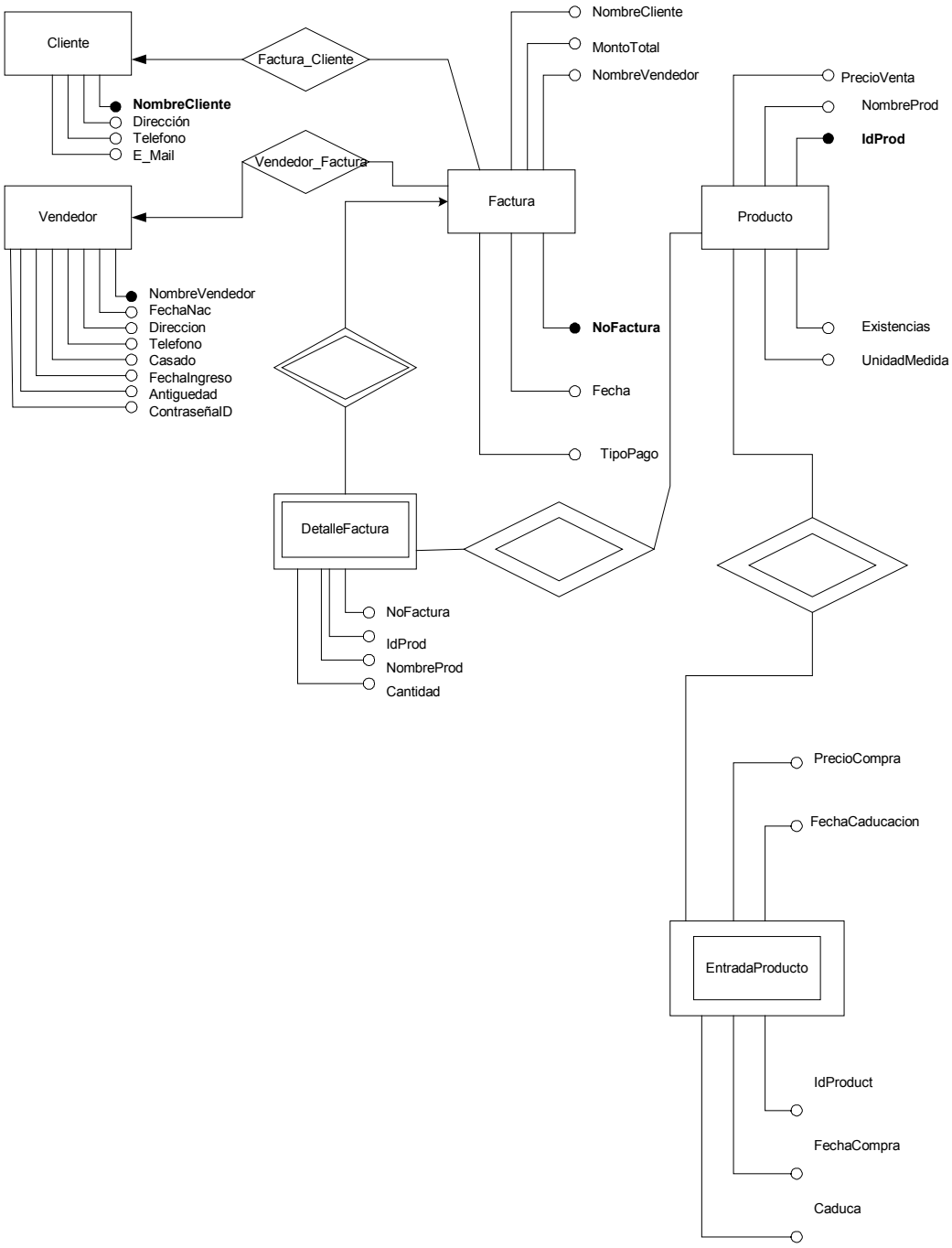


Figura 8.9Diagrama E/R básico de “Mi Ventesita”

Una vez diseñado el modelo lógico, el siguiente paso es diseñar el esquema relacional utilizando las herramientas de Access para este propósito. El objetivo de realizar este esquema es el garantizar las restricciones de integridad que garantiza el SGBD una vez que se diseña este, asimismo se facilita el diseño de consultas utilizando el Query by Example de Access.

DESARROLLO DE LA PRÁCTICA
Diseño del Esquema Relacional (Relaciones en access)

Para establecer una relación en Access tiene que cargar el entorno de relaciones. Una vez hecho esto basta con agregar las tablas que participarán en la relación y por último establecer propiamente la relación, esto se logra arrastrando el campo origen de la relación A al campo destino de la relación B. Por ejemplo para relacionar Producto con EntradaProducto:

Figura 8.10 Ambiente para el diseño de una “relación” en access

-**Exigir Integridad Referencial**: Cardinalidad 1,N (Al agregar un elemento en el lado ‘N’, debe obligatoriamente estar enlazado con uno de la otra relación).
-**Actualizar/Eliminar en cascada**: Repercusión en cascada sobre el lado ‘N’ de las modificaciones que se realicen sobre los elementos relacionados en el lado de ‘1’.

El esquema relacional en su base de datos debe quedar como se muestra en la siguiente figura 8.8 (**Esquema relacional general** de la aplicación MiVentesita):

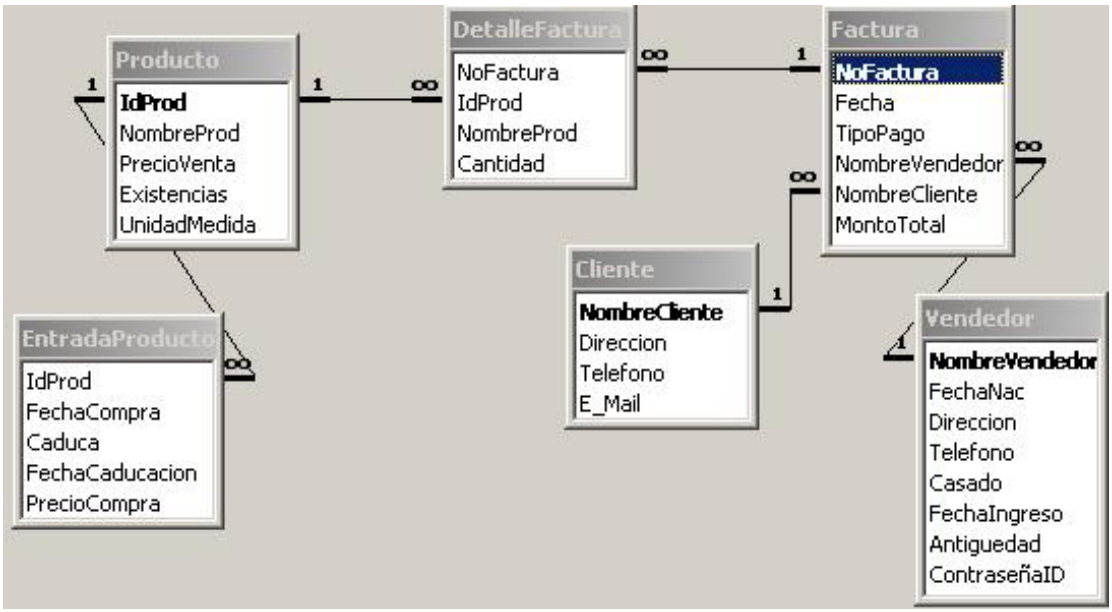


Figura 8.11 Esquema relacional del Sistema

Interfeces gráficas de usuario (Formularios).
Los Formularios son herramientas utilizadas para que el usuario haga uso de una interfaz sencilla al momento de introducir los datos, de otra manera tendría que introducirlos directamente en las tablas. Con ellos es posible la visualización simultanea de varias tablas relacionadas, ejecutar programas que realizan determinadas acciones al momento de la introducción de los datos con las cuales se pueden realizar validaciones complejas en los campos sin salirse del ambiente del formulario poder implementar campos calculados complejos y en general hacer uso de una gran cantidad de herramientas incorporadas.

Access trae incorporado en su entorno de implementación de sistemas de bases de datos un apartado para desarrollar interfaces de mantenimiento de datos de forma gráfica. Miremos a través del siguiente gráfico una breve descripción de este entorno:

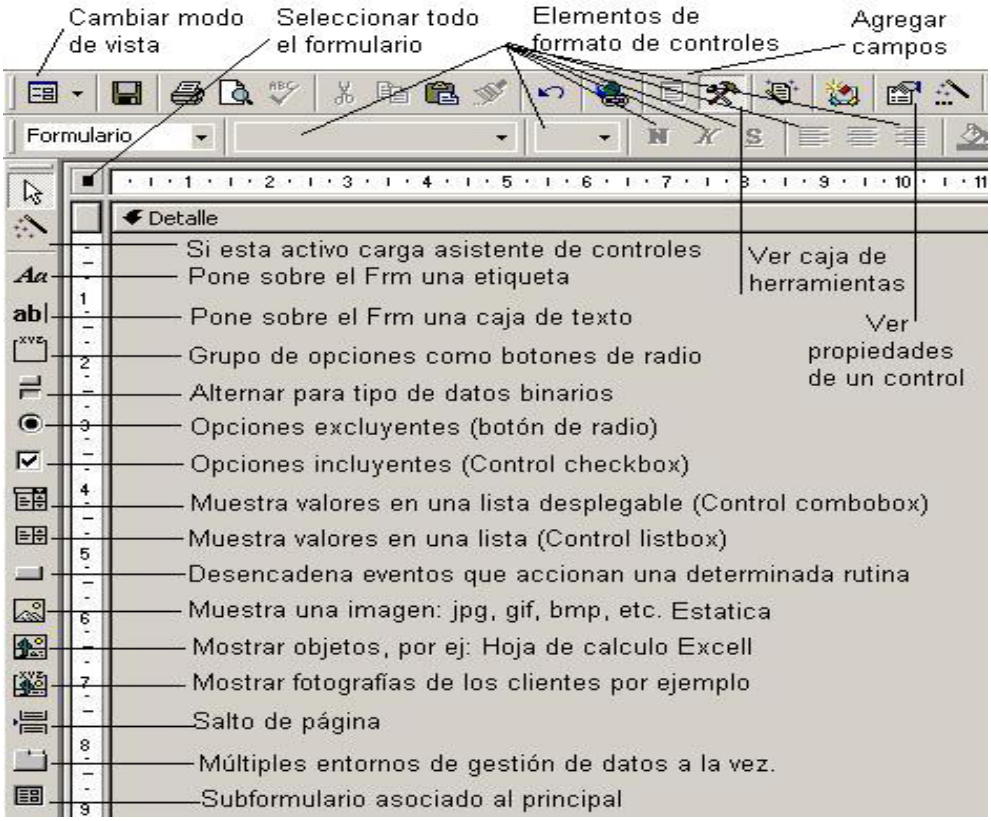


Figura 8.12 Partes Esenciales de un Formulario

Diseño del primer formulario.

Todos los ejemplos de formularios en esta práctica no toman en cuenta el uso de asistentes, se deja al estudiante que trate esta temática por su cuenta, ya que cuentan con un mecanismo sencillo de seguir contando con una guía muy fácil de atender. A continuación se indicará paso a paso la elaboración de nuestro primer formulario para la **aplicación MiVentesita**.

Se diseñará el formulario para dar mantenimiento a la tabla vendedor. Antes de continuar **se agregará el atributo Fotografia** a la tabla, como un tipo de datos *ObjetoOLE*.
Los pasos para crear el formulario son los siguientes:

1. Crear un nuevo formulario en vista de diseño.
2. Indicar la fuente de datos que manejará este formulario(La fuente de datos en este caso es la tabla **Vendedor**.)
3. Seleccionar todo el formulario (ver figura 8.9) y haciendo click con el botón derecho ir a propiedades del formulario.
4. Hacer Clic en la pestaña rotulada Datos
5. Llenar las siguientes opciones:
   Origen del registro: Seleccionar la Tabla vendedor
   Permitir Filtros: No

6. Salirse de la definición de las propiedades del formulario e Insertar (Arrastrar del cuadro de herramientas al área de diseño del formulario) un control ficha
7. Cambie los nombres los nombres por defecto de las pestañas del control(Los Nombres por defecto son página1 y página2), para ello active las propiedades de la primera pestaña, Haga click en formato y Cambie el título a **Vendedor**, hacer algo similar con la siguiente pestaña para cambiar el título a **Datos personales**.

El formulario quedará de la siguiente manera (Ver Fig.).

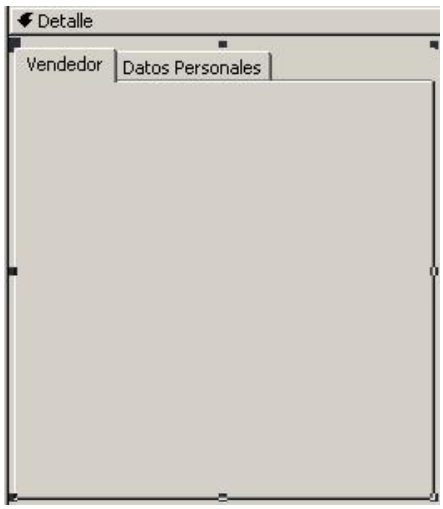


Figura 8.13 Control de Ficha para Datos de Vendedor

8. Agregue (arrastre al área de diseño del formulario) los campos **nombre** y **fotografía** en la pestaña *Vendedor*. El resto de campos ponerlos en la pestaña *Datos Personales* (Ver Figs).

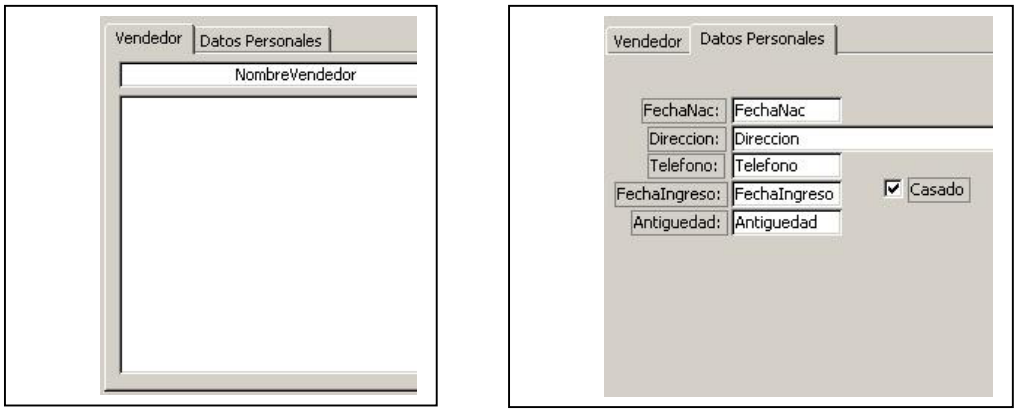


Figura 8.14 Fichas para los Datos del Vendedor

9. Tralade el archivo donde se encuentra la fotografía al Paint de Window, si está en formato bmp páselo a formato JPG, para ahorrar espacio y realice la acción de copiar y pegar al atributo Fotografía, luego Ajuste la ventana al formulario.

10. Cambiar en la propiedad formato, Titulo: **Datos de los vendedores** y Cambie en la propiedad formato, Estilo de bordes: **Diálogo**.

Corra el formulario para ver los cambios (ver fig). y Guarde el formulario como *FrmVendedor*.

Figura 8.15 Instancia de los datos Personales del Vendedor

De forma Similar Crear el resto de formularios cuya fuente de datos es una sola tabla.

Formulario FrmCliente.
Cambios en formato:

- Titulo: **Entrada de Datos del Cliente**. (Título que se visualizará en modo de entrada de datos)
- Vista predeterminada: **Formularios continuos**.
- Estilo de bordes: **Fino**.
- Centrado automático: **Si**.
- Botones minimizar maximizar: **Solo minimizar**.

Otros cambios:

- Color de Fondo del formulario a blanco(Botón Derecho en sección Detalle).
- Agregar un encabezado de formulario (Opción **Ver +** Encabezado/Pie de Formulario).
- En el encabezado Escribir utilizando etiquetas los Rótulos que hacen referencia a los campos de cliente (Nombre del Cliente, Dirección Actual, Teléfono, Correo Electrónico).

- En **detalle de formulario dejar las cajas de texto** que muestran los contenidos de las instancias que pertenecen a este esquema.

La figura 8.10 muestra el formulario en vista de diseño.

Figura 8.16 Formulario FrmCliente en vista diseño

Formulario FrmProducto.
Este formulario es muy parecido al anterior las acciones nuevas:

- Quite el cuadro de texto asociado a Unidad de Medida. Para mostrar los valores utilizaremos un cuadro de texto especial denominado cuadro combinado, el cual arrastraremos hasta el lugar donde se encontraba la caja de texto, eliminar la etiqueta asociada al cuadro combinado.
- En la propiedad *Tipo de Origen de Fila* de la pestaña DATOS asociado al cuadro combinado (combo box), escoja la opción **Lista de Valores** esto con el fin de posteriormente suministrar los valores que el usuario puede escoger de este combo box, estos valores se determinan de la siguiente manera:
- En *Origen de la Fila* escriba: "LITRO"; "LIBRA"; "PAQUETE"; "GRAMOS"; "UNIDAD"; "CAJA".
- En *Valor Predeterminado* escriba: "LITRO" (Este será el valor por defecto que aparecerá en el combo).
- En *Regla de Validación* escriba: ="LITRO" Or "LIBRA" Or "PAQUETE" Or "GRAMOS" Or "UNIDAD" Or "CAJA" Esta regla de validación tiene el objetivo de no permitir de introducir valores que no aparezcan en la lista de valores.
- En *Texto de Validación* ponga: SOLO SON PERMITIDOS LOS VALORES "LITRO";"LIBRA";"PAQUETE";"GRAMOS";"UNIDAD";"CAJA", este texto de validación aparecerá cada vez que se intente violar la regla de validación.
- Por último en *Origen de Control* pondremos el campo de la tabla asociado a este control : **UnidadMedida,** con lo cual relacionamos el combo box con el atributo UnidadMedida de la tabla Producto. Nótese como el rótulo independiente que tenía de fondo el combo box pasa a UnidadMedida

- Finalmente para lograr la introducción de datos por filas como se muestra en la figura de abajo, la propiedad de formato del formulario denominada vista predeterminada se debe poner a “Hoja de Datos”.

La siguiente figura 8.11 muestra el formulario en ejecución.

Figura 8.17 Formulario Listo para la Introducción/Modificación de Datos

Actualizar en los formularios restricciones agregadas en el diseño.

Como última acción de este laboratorio cambie las propiedades de las cajas de texto que utiliza para mostrar los campos de las tres tablas que tienen asociado un formulario, esto es, cambiar la propiedad de más cara de entrada de cada caja de texto al valor que tiene diseñado para cada campo en estas tres tablas. Realice esta misma acción en los valores de regla de validación si procede. Con lo anterior se desea enfatizar el hecho de que las máscaras de entrada diseñadas en las tablas no se heredan a los formularios.

Ayuda: Copy – Paste del valor en tiempo de diseño al valor de propiedad en el formulario.

8.2.3 Práctica 3: Formularios con Sub Formularios

En esta práctica se diseñará un formulario con un sub formulario con el fin de que el usuario de la aplicación "Mi Ventesita" pueda introducir los datos de varias tablas en un solo ambiente.

OBJETIVO DE LA PRACTICA

- Profundizar en el diseño de formularios con sub formularios
- Saber diseñar formularios con sub formularios relacionados con más de 2 fuentes de datos

INTRODUCCIÓN TEÓRICA

Control de formulario Maestro-Detalle.

En el laboratorio anterior se realizaron una serie de formularios con el objetivo de introducir información, cuya fuente de datos provenía de una sola tabla. Empezaremos este laboratorio mostrando como se generan formularios que contienen información proveniente de dos o más tablas.

Antes de empezar con el desarrollo de la práctica se hará énfasis en los siguientes aspectos:

- Siempre que se trabaja con fuentes de datos múltiples, es necesario diferenciar quien será la tabla "Maestra" y la(s) tabla(s) "Detalle".
- Tabla Maestra: Es una relación dentro del esquema generalmente uno a muchos con exigencia de integridad referencial. La parte maestra es **la tabla** del lado de la **cardinalidad uno**.
- Tabla Detalle: Bajo un esquema de relación uno a muchos es **el lado de cardinalidad muchos**.

¿Para qué se utilizan este tipo de formularios?
Estos resuelven la problemática de introducción de datos desde una sola interfaz hacia múltiples tablas o fuentes de datos que se encuentran relacionadas a través de uno o más campos.

En el ejemplo actual denominado "MiVentesita" existen los siguientes ejemplos que pueden admitir un tratamiento Maestro-Detalle:

Figura 8.18 Maestro-Detalle simple (dos fuentes de datos):

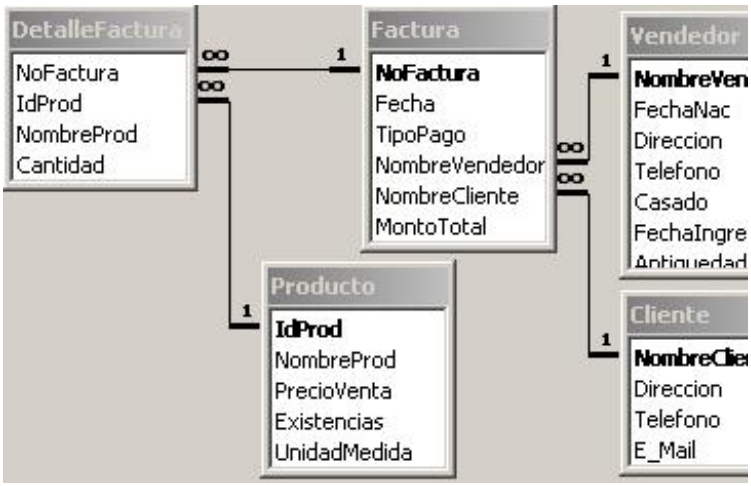


Figura 8.19 Maestro-Detalle compuesto (más de dos fuentes de datos)

Como puede ver el primer caso implica únicamente a dos tablas, donde la relación Producto corresponde a la **tabla maestra** y la tabla DetalleProducto es la parte de **detalle**. En el segundo caso, la tabla Maestra es Factura y el Detalle es DetalleFactura, las otras tablas participan de la siguiente forma:

- **Producto** suministra la información del los productos en DetalleFactura.
- **Vendedor y Cliente**, suministran la información relacionada al vendedor y al cliente que esta realizando la venta y compra de productos respectivamente.

DESARROLLO DE LA PRÁCTICA
Generar el formulario Maestro-Detalle: Producto-DetalleProducto.
Este formulario tiene como objetivo registrar las entradas a bodega de los productos que se venden en la Ventesita. Para crear este formulario haga lo siguiente:

1. Antes de comenzar a diseñar el formulario agregue un nuevo campo a la tabla EntradaProducto, esto con la finalidad de llevar el control del número de nuevo producto introducido, póngale al campo **Cantidad** (Numérico, Entero).
2. Abra un **nuevo formulario para diseño** y ponga como fuente de datos: Producto.
3. Añada al formulario únicamente los campos: **IdProd, NombreProd**.

4. Agregue al formulario de la barra de herramientas el objeto SubFormulario, ponga como fuente de datos de este objeto la tabla EntradaProducto (se deben vincular por defecto los campos **Producto.IdProd con Entradaproducto.IdProd**).
5. Cambie el título del subformulario a: **Entrada de Nuevos productos**.
6. Cambie el título del principal a: **Control de Entrada de Productos a Bodega**.
7. De la propiedad *Formato* del principal quitar *Selectores de Registro y Barras de Desplazamiento*.
8. Vamos a realizar algunos cambios en el principal para que los datos mostrados de productos no puedan ser modificados desde esta pantalla, para ello realice lo siguiente: De la pestaña de *datos* de las propiedades del principal, quite las opciones de modificación de datos: *Permitir Ediciones, Permitir Eliminación, Permitir Agregar* y Cambie la forma de obtener el Conjunto de Datos (Recordset), de Dynaset a *SnapShot*.
9. Guarde el formulario como **FrmControlEntradaBodega**.
   Agregue un botón que abra el formulario de Productos (por si necesita agregar uno nuevo: ver código al final de la guía).

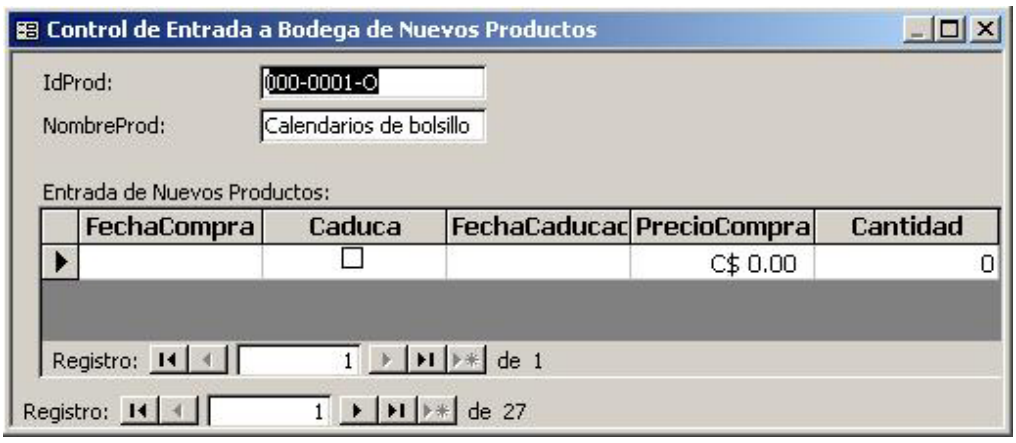


Figura 8.20 Maestro-Detalle de Productos que entran a bodega

Algunos comentarios del formulario Maestro-Detalle anterior:

- **Se cargarán siempre todos los datos** que existan en productos sin posibilidad de restringir la fuente de datos. Se podrían filtrar datos

siempre y cuando se cambiase la fuente de datos del SubFormulario, por ejemplo una consulta.

- La información en el Subformulario se debe ver como un conjunto y no se pueden hacer inspecciones sobre grupos de datos o registros de forma individual (**problema de captura** de información en tiempo de exe).

- Se dejaron algunos cabos sueltos como por ejemplo la actualización de existencias de producto.

Generar formulario Maestro-Detalle para Facturación.
Este formulario tiene como objetivo elaborar las facturas al cliente. Antes de empezar a describir los pasos para generar esta interfaz, es necesario explicar un poco el escenario de la facturación. El siguiente diagrama muestra las fuentes de datos que suministran información a la factura (las flechas indican la ruta de la información):

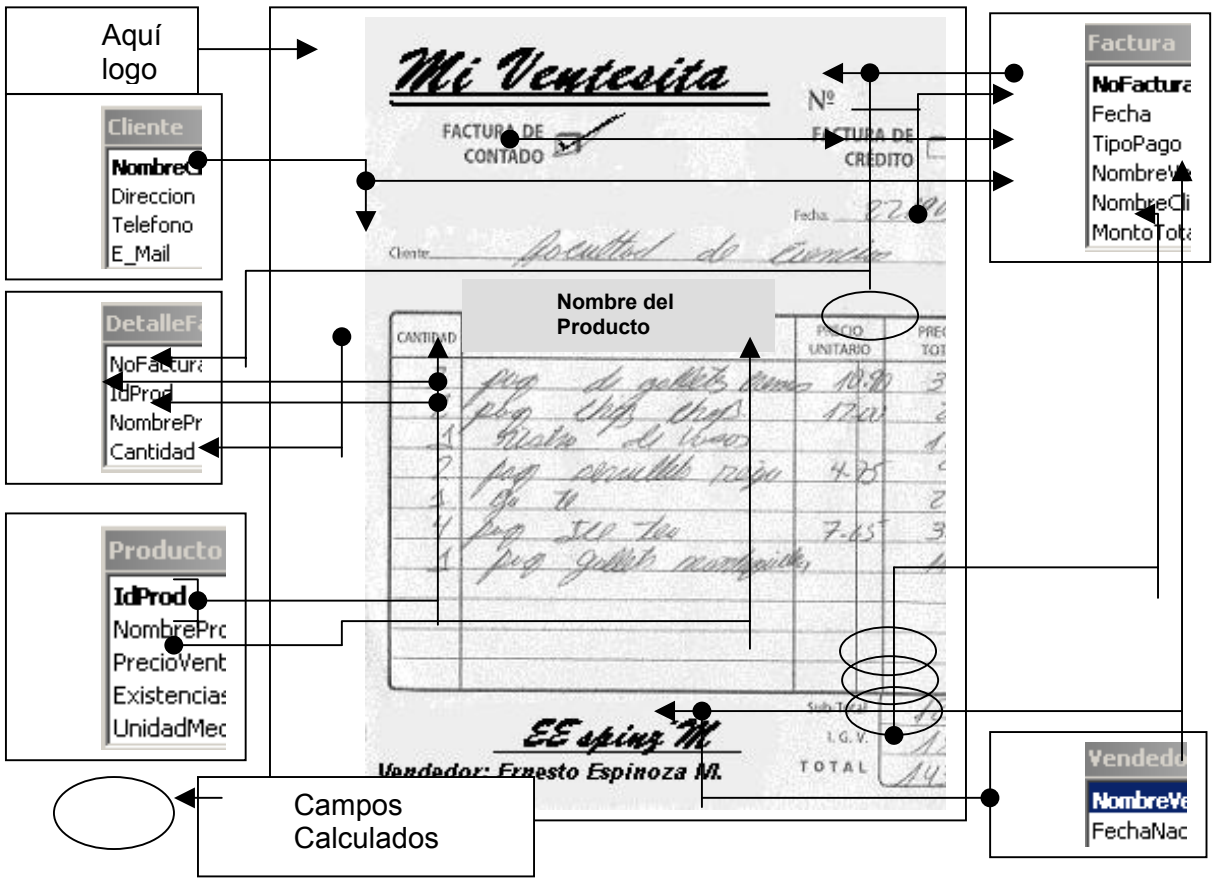


Figura 8.21 Componentes de una Factura

Los pasos para generar el formulario son los siguientes:

1. Agregue un nuevo formulario en vista de diseño.

2. Tome como fuente de datos la tabla **Factura**, en la pestaña Datos ponga *Entrada de datos* a SI.

3. Inserte en el formulario **NoFactura y Fecha**.

4. Para el **campo TipoPago** se asociarán **botones de radio** para definir el tipo de pago efectuado por el cliente. Inserte un grupo de opciones, cambie la etiqueta a Tipo de Pago en vez de MarcoX. Dentro del grupo de opciones agregue dos botones de radio. Cambie las etiquetas de los botones a Pago en Efectivo y Pago al Crédito. Para Access en el botón de radio el **valor –1(true) y 0(false)**, de forma que cambie de la pestaña Datos asociada a cada botón de radio la propiedad *Valor de Opción* a –1 y 0 respectivamente. Por último cambie de la propiedad **Datos del Grupo de Opciones los valores Origen de control a TipoPago**, y **Valor predeterminado a verdadero** (Pago en Efectivo). debe quedar algo similar a lo siguiente:

Figura 8.22 Encabezado Parcial de la Factura

5. Los siguientes pasos son agregar las fuentes de datos correspondientes a los clientes y a los vendedores recuerde que esta información se obtiene de dos tablas diferentes. Para agregar los nombre de los clientes a la factura inserte en el formulario un cuadro combinado. Cambie en Datos la propiedad *Origen de Control* a **NombreCliente,** para que los valores depositados en el cuadro combinado se agreguen a este campo en la factura, luego cambie *Origen de la Fila* a **SELECT [NombreCliente] FROM Cliente;** para que obtenga los Nombres de cliente que ya tenemos dentro de la tabla Cliente.

6. Repita esto mismo para recuperar los nombres de los vendedores.

7. Agregar un botón Agregar/Modificar Cliente (*) ver figura.

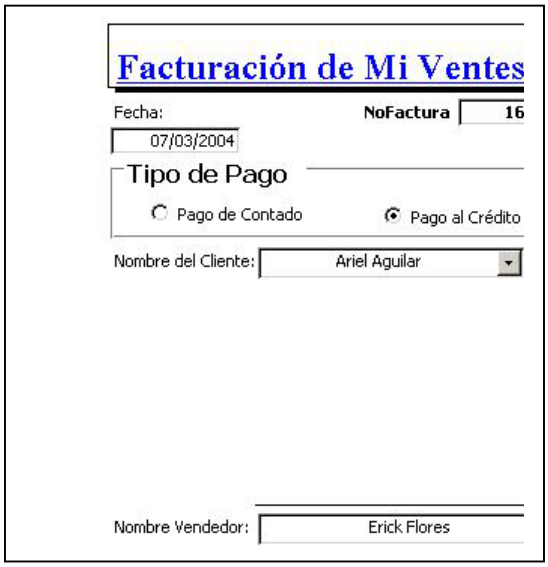


Figura 8.23 Encabezado de Facturación

8.2.4 Práctica 4: Formularios con Sub formularios(Cont)

En esta práctica se completará el diseño de la entrada de datos del usuario utilizando formulario con sub formulario para lo cual se hará uso de una lista combinada(Combo Box) para introducir datos del detalle de la factura, asimismo mediante el evento before update se procederá a obtener el costo total de cada articulo también en el detalle de la factura

OBJETIVO DE LA PRACTICA

- Uso de listas combinadas para realizar actualizaciones tomando como base una consulta SQL sencilla del tipo Select
- Diseñar campos Calculados en los formularios
- Utilizar eventos para para actualizar los datos de la aplicación
- Insertar y utilizar el objeto sub formulario de la barra de herramientas

INTRODUCCIÓN TEÓRICA

Listas y Listas Cobinadas

Una lista muestra un conjunto de datos en pantalla, al igual que un menú desplegable, pero la lista está siempre abierta. Se puede elegir un elemento de la lista, moviendo el cursor y pulsando enter o haciendo clic. También se puede escribir la primera letra de la opción para seleccionar la entrada deseada. Una vez que se ha seleccionado un elemento el valor pasa al campo dependiente. Las listas combinadas se diferencian de las listas en que: (a) la lista Combinada aparece inicialmente como una sola fila, con una flecha que hay que pulsar para desplegarla (b) Como opción la lista combinada permite insertar un valor que no se encuentra en la lista. Como se verá en la práctica las listas combinadas son particularmente útiles para realizar búsquedas en función de los datos cotenidos en una tabla o una consulta

Consultas Sencillas SQL del tipo Select

Una consulta simple del tipo Select tiene la forma general:

Select <Lista de Atributos> From <Nombre de la Tabla> Where <Condición> donde <Lista de Atributos> es el conjunto de atributos de la tabla cuyos contenidos se desean enlistar y <condición> es una condición Lógica de modo que los contenidos de los atributos se enlistan siempre y cuando <Condición> es verdasdero.

Asi: Select Nombre, Edad From estudiantes Where Procedencia =”Masaya” el efecto es que se enlistará el Nombre y la Edad en aquellas filas de la tabla Estudiantes en que se cumple que la procedencia del estudiante es “Masaya”.

Procedimientos de Eventos

Basicamente, un procedimiento de evento es un código de programa que se ejecuta cuando se produce un evento. El código se agrega directamente al formulario o informe que contiene el evento que se está procesando. Cada vez que cambiamos el valor de un control, se producen una serie de eventos. La mayorí de ellos tiene que ver con la pulsación de teclas o el cambio de foco, pero hay dos eventos que se activan específicamente con los cambios en los datos: BeforeUpdate y AfterUpdate. BeforeUpdate tiene lugar cuando el usuario intenta abandonar un control cambiado o guardar un registro modificado, AfterUpdate se activa cuando el cambio se completa de forma satisfactoria. En el caso específico de BeforeUpdate podemos utilizarlo para comprobar si determinadas condiciones se cumplen en cuyo caso el o los datos se almacenan, de lo contrario se cancela la acción.

DESARROLLO DE LA PRÁCTICA

En la práctica anterior se indicó el marco general de la factura, ahora se continuará diseñando la parte interna de la factura, es decir, el detalle de los productos más el cálculo del valor a pagar por el cliente. En este laboratorio también se incluirá el código asociado a los botones de comando agregados en los formularios FrmEntradaBodega y FrmFactura.

Generando el detalle de la facturación.

Para generar el detalle de la factura no bastará con incluir un subformulario dentro del formulario principal y asociarle como fuente de datos el detalle de la factura, debido a que se necesitan realizar otras actividades con los datos como: cálculos, obtención de otras fuentes de datos y operaciones sobre controles. Los pasos para generar el formulario de detalle de factura son los siguientes:

1. Antes de empezar a diseñar el formulario, quite de la tabla DetalleFactura el campo NombreProducto, note que este campo se puede obtener a través de los códigos de los distintos productos que se van incorporando al detalle, es decir es una información redundante. Al usuario, sin embargo, como lo indicaremos posteriormente le mostraremos los nombres de los productos aunque el valor que se almacene en el detalle sea el código asociado a ese producto. Con el fin de Obtener el total de cada una de las facturas se requiere conocer el precio unitario de cada producto y el precio total de ese producto, el cual será como lo veremos luego un campo calculado que depende del precio unitario y de la cantidad de productos comprados : Inserte **dos nuevos campos** a DetalleFactura: **PrecioUnitario y PrecioTotal** como valores de doble precisión formato moneda.

2. Abra un nuevo formulario en vista de diseño y asocie como fuente de datos DetalleFactura, el formulario es de tipo continuo.

3. Insertar el campo cantidad en el formulario.

4. Insertar un cuadro combinado que mostrará la lista de los Nombres de los productos y al escoger el producto requerido retornará el código del mismo para ser almacenado en el detalle. Esta operación se realiza de esta manera pues es más fácil buscar por el Nombre del producto que aprederse los códigos de los mismos. Recordemos por otra parte que la fuente de datos de un Combo Box puede ser una tabla, una consulta ó un conjunto de datos constantes, en este caso utilizaremos una consulta SQL sencilla.

El SQL(Stuctured Query Language) como su nombre lo indica es un lenguaje utilizado ya en forma normalizada para interrogar las bases de datos en función de la información almacenada.
Uno de los comandos básicos es el comando Select cuya forma general es :
Select <atributos a listar> from <Nombre de la tabla o tablas involucradas>
Where <Condición>.
Ejemplos: Sea la Tabla Estudiante(**N Carnet**, Nombre, Edad, Sexo, Procedencia)
Entonces:

a) Select Nombre,Procedencia from Estudiante)
Enlista el Nombre y la Procedencia de de la relación estudiante.

b) Select Nombre from Estudiante Where Edad>18 and Procedencia = “Managua”

Enlista el Nombre de los Estudiantes pero solamente aquellos cuya edad es mayor que 18 y que proceden de Managua.
En los laboratorios posteriores se profundizará sobre este tema

Para diseñar la consulta requerida, en la pestaña datos modifique las propiedades del cuadro combinado: *Origen de control* a **Idprod** (en este campo se almacenará el valor elegido por el usuario), en Origen de la fila coloque la consulta: **SELECT Producto.NombreProd, Producto.IdProd, Producto.PrecioVenta FROM Producto;**
ahora debemos indicar el número de columna que suministra el valor al campo en *Columna Dependiente* poner **2** (que corresponde a Idprod).

5. Para finalizar con las propiedades del cuadro combinado cambie en la pestaña Formato la propiedad *Número de columnas* a **3** (Para poder visualizar en tiempo de ejecución el nombre del producto, su código y precio, es decir los tres componentes de la consulta recién diseñada). En esta misma pestaña cambie la propiedad *Ancho de la lista* a **10cm** (Para que al momento de desplegar la lista de valores se puedan observar las tres columnas de forma simultánea).

6. En cuanto a la columna Precio se deben de realizar otras actividades para captar el valor de la tercer columna (Producto.PrecioVenta) y ubicarlo dentro de

el detalle de la factura. Para esto **incluiremos el campo PrecioUnitario** al formulario.

7. Para actualizar adecuadamente los datos cada vez que se cambie el valor al detalle en un producto este cambio debe reflejarse en el nuevo campo agregado al formulario. Para hacer esto necesitamos utilizar el evento **BeforeUpdate** asociado a la lista combo. Para activar el editor de código haga clic derecho sobre la lista Combo, pulse *Generar Evento* y llame al *Generador de Código*.

8. Una vez en el generador de código asocie al combo el evento BeforeUpdate, escriba el siguiente código que le pasa el valor del precio del producto que se ha seleccionado: **Me.PrecioUnitario = ComboListaProductos.Column(2, ComboListaProductos.ListIndex)**. Observe que la propiedad Column acepta dos parámetros el primero la columna que puede tenerlos valores (0,1,**2**), correspondiendo el 0 para la primera columna,1 para la segunda etc y el segundo el valor del índice de fila activo (ListIndex). En nuestro caso específico se escogió el valor de 2, correspondiente al precio de venta en la consulta que se diseñó para ser visualizada en la lista combinada.

9. Para que el precio del producto no pueda ser modificado por el usuario se pasará a **bloquear la introducción de datos**. Esto último utilizando la pestaña datos **asociada a precioUnitario**.

10. Ahora queda por obtener los subtotales por producto llevado. Para esto incluya el campo **PrecioTotal**. obtenga el precio total a partir de la Cantidad de productos llevados por el precio Unitario de cada producto (**el precio total se modifica cada vez que se elige un nuevo producto o cuando se cambia la cantidad de productos a llevar**) *AYUDA*: Se Puede ocupar el mismo evento BeforeUpdate asociado al campo Cantidad y comprobar si el precio unitario del producto es diferente de 0, en cuyo caso recalcular el precio total, de lo contrario no hacer nada, ya que no hay ningún producto activo. Un Ejemplo de cómo debe quedar el formulario FrmDetalleFactura es el siguiente:

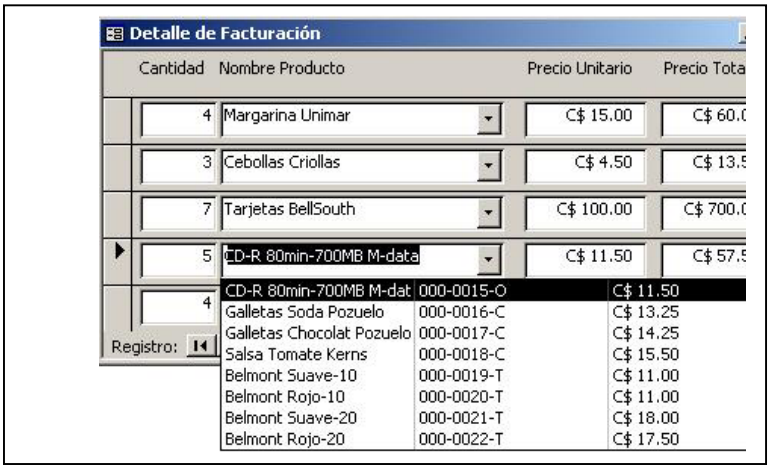


Figura 8.24 Lista Combinada de los productos con su respectivo precio

Inserción del detalle de factura en la Factura.

1. Inserte un subformulario en tiempo de diseño en FrmFactura y asocie el Formulario FrmDetallefactura a través de los campos enlace.

2. En este punto ya puede trabajar con la facturación de los productos. Falta realizar las operaciones de Totales y Subtotales de la facturación. **Agregue el campo MontoTotal** al formulario y haremos uso nuevamente del evento BeforeUpdate. Se debe recordar que se debe actualizar siempre que se modifique o bien, la cantidad de productos solicitados, o el producto que se esta agregando a la factura. El código siguiente del formulario FrmDetallefactura se encarga de actualizar cuando se modifica el número de productos que se va a llevar:

```
Private Sub Cantidad_BeforeUpdate(Cancel As Integer)
   'Si está puesto a cero no se habia elegido producto
   If Me.PrecioUnitario <> 0 Then
      'Quitar el valor anterior al total de la factura
      Forms!FrmFactura.MontoTotal = Val(Forms!FrmFactura.MontoTotal) -
Me.PrecioTotal
      'Si el usuario modifica la cantidad de productos actualizar
      Me.PrecioTotal = Me.PrecioUnitario * Me.Cantidad
      'Poner el nuevo valor al monto total de la factura
      Forms!FrmFactura.MontoTotal = Val(Forms!FrmFactura.MontoTotal) +
Me.PrecioTotal
   End If
End Sub
```

- El estudiante debe Obtener mediante cajas de texto independientes el IGV y el Total General. El aspecto final debe ser similar al de la figura 8.13(abajo)

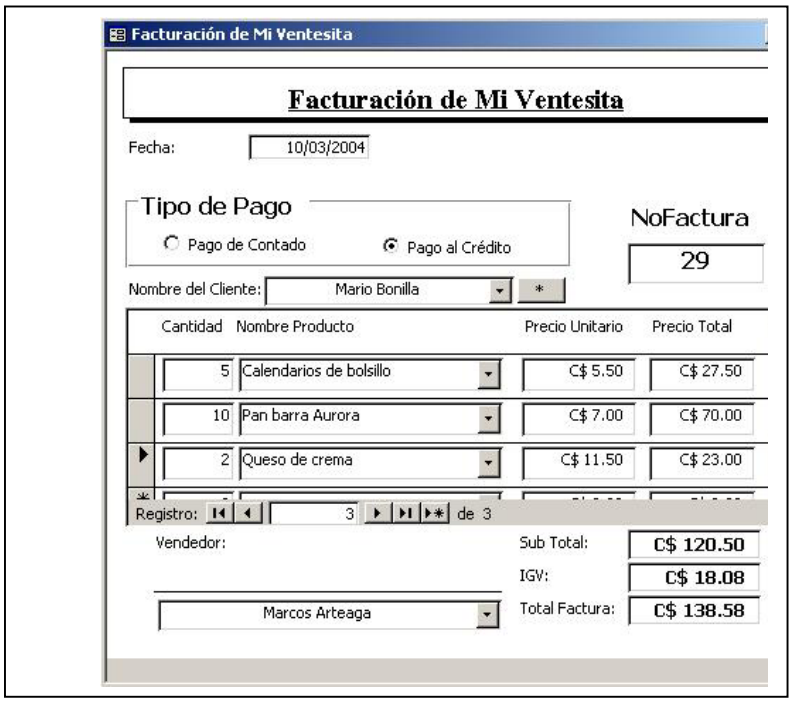


Figura 8.25 Encabezado y detalle de la Factura con totalizaciones

Para concluir con esta parte del laboratorio se colocará el código correspondiente a los botones de comando que se agregaron tanto en FrmFactura (nuevo cliente) como en FrmEntradaProductos para (modifica productos: para el **estudiante**).

Para FrmFactura

```
Private Sub CmdModificarCliente_Click()
'Si ocurre un error avisar al usuario
On Error GoTo Err_CmdModificarCliente_Click
   'Declarar Cadena
   Dim NombreFormulario As String
   NombreFormulario = "FrmCliente"
   DoCmd.OpenForm NombreFormulario, , , , , acDialog

NormalExit_CmdModificarCliente_Click:
   'Salir del procedimiento
   Exit Sub

Err_CmdModificarCliente_Click:
   'Ventana mensaje de error
   MsgBox "Error del sistema:" + Err.Description
End Sub
```

8.2.5 Práctica 5: Importación de Datos Externos

OBJETIVO DE LA PRACTICA

- Obtener datos externos a partir de archivos de texto
- Obtener datos externos a partir de archivos de excel
- Comprender y aplicar el concepto de recordset
- Saber efectuar una conexión a una base de datos utilizando las DAO del motor de access
- Conocer las propiedades básicas de los objetos recordset
- Aplicar los conocimientos básicos de programación utilizando VBA

DESARROLLO DE LA PRÁCTICA

Obtener datos externos.

Hasta ahora se han completado todas las interfaces básicas para dar mantenimiento a los datos, en el primer punto de este laboratorio se va a obtener información que ya está almacenada pero desde otras fuentes de datos que no comparten el formato de Access. Los formatos de otras fuentes de datos soportados por Access son diversas desde hojas de cálculo hasta archivos de texto formateados mediante un delimitador.

Obtener datos de clientes desde un archivo de texto.

Primero eliminar toda información relacionada con clientes en su aplicación. Bajar el archivo cliente.txt de la red. Ahora del menú archivo elijir obtener datos

externos e importe los datos contenidos en el archivo txt. (profundizar sobre esta forma de obtención de datos en la ayuda de Access).

Obtener datos de productos y vendedor desde un archivo en excel.

Primero eliminar toda información relacionada con productos y vendedores en su aplicación. Baje el archivo producto.xls y vendedor.xls de la red. Obtener los datos que se encuentran en las hojas de cálculo.

Agregar datos de prueba de forma masiva.

En este punto se va a implementar código para agregar datos a factura y a detalle de facturas de forma que no se tenga que escribir esta información en la base de datos directamente.

Agregar un número X de facturas a nuestra aplicación.

El siguiente código muestra paso a paso la forma de trabajar con datos llamados desde código escrito en Visual Basic Access (VBA) como soporte de Access a los mecanismos de programación y acceso a datos. Se han colocado los comentarios considerados pertinentes dentro del código para ayudar a comprender la función de cada línea. Es conveniente profundizar sobre los siguientes conceptos: ADO, DoCmd, Rnd, CurrentDB, Cadenas de Conexión, Tipos de Recordset, los mecanismos de desplazamiento Move y Open para la apertura de fuentes de datos. A continuación se ha diseñado un formulario con un botón de comandos el cual tiene asociado el código que realiza las operaciones de inserción.

Figura 8.26 Botón que agrega 100 facturas aleatorias al sistema

El siguiente código agrega de forma aleatoria 100 facturas en la tabla Factura de la base de datos MiVentesita.mdb.

```
Option Compare Database
'**************** Evento click asociado al botón CmdAgregarFactura
'**************** para agregar facturas

Private Sub CmdAgregarFactura_Click()

'**************** Se Declaran la variables que se asociarán a los conjuntos
'**************** de datos

    Dim rstCliente As ADODB.Recordset
    Dim rstVendedor As ADODB.Recordset
    Dim rstFactura As ADODB.Recordset

'**************** La Cadena de conexión mantiene el driver y ruta de la base
'**************** de datos
```

```
   Dim strconexion As String

'**************** Estos dos arreglos almacenan los nombres de vendedores
'**************** y clientes

   Dim NombreVendedor(20) As String
   Dim NombreCliente(20) As String

'**************** Se inicializa una conexión sencilla a una BD de
'**************** Access2000 (Jet 4.0)

   strconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
   CurrentDb.Name

'**************** Se crea un nuevo objeto Recordset y se asocia a las
'**************** variables establecidas

   Set rstCliente = New ADODB.Recordset
   Set rstVendedor = New ADODB.Recordset
   Set rstFactura = New ADODB.Recordset

'**************** Definir Cliente y Vendedor solo para lectura
'**************** (Profundizar en la ayuda)

   rstCliente.CursorType = adOpenStatic
   rstVendedor.CursorType = adOpenStatic


'**************** Apertura de los recordset con los parámetros
'**************** correspondientes (Profundiza en la ayuda)

   rstCliente.Open "Cliente", strconexion, , , adCmdTable
   rstVendedor.Open "Vendedor", strconexion, , , adCmdTable
   rstFactura.Open "Factura", strconexion, adOpenDynamic, adLockOptimistic,
   adCmdTable

'**************** v: número de vendedores. c: número de clientes

   Dim v, c As Integer

'**************** Poner los recordsets correspondientes apuntando al primer
'**************** registro

   rstCliente.MoveFirst
   rstVendedor.MoveFirst

'**************** Cuenta de los Los vendedores
   v = 0
   While Not rstVendedor.EOF
      NombreVendedor(v) = rstVendedor!NombreVendedor
      v = v + 1
      rstVendedor.MoveNext
```

```
   Wend

'**************** Cuenta de los Los Clientes

   c = 0
   While Not rstCliente.EOF
      NombreCliente(c) = rstCliente!NombreCliente
      c = c + 1
      rstCliente.MoveNext
   Wend

'**************** Comienzo del llenado de los registros aleatorios de Factura
   Dim i As Byte
   For i = 0 To 100

'**************** Se agrega un nuevo registro de factura en blanco
      rstFactura.AddNew

'**************** Construir fechas aleatorias
'**************** Dias entre 1..27 para simplificar

      midia = Int(Rnd() * 26 + 1)

'**************** Meses 1..12

      mimes = Int(Rnd() * 11 + 1)

'**************** Las fechas solo corresponden al 2004 (Construir)

      mifecha = Str(midia) & "/" & Str(mimes) & "/" & "2004"

'**************** Asignar la fecha construida de forma aleatoria

      rstFactura!Fecha = mifecha

'****************Asignar de Forma aleatoria el tipo de pago

      rstFactura!TipoPago = Int(Rnd() * 2)

'**************** Asignación de Vendedores aleatorios entre
'**************** 0..totalvendedores

      rstFactura!NombreVendedor = NombreVendedor(Int(Rnd() * (v - 1)))

'**************** Asignación de Clientes aleatorios entre 0..totalclientes
      rstFactura!NombreCliente = NombreCliente(Int(Rnd() * (c - 1)))

'**************** Asignación de Montos aleatorios a rectificarse
'**************** posteriormente con el detalle de factura

      rstFactura!MontoTotal = Round(Rnd() * 10000, 2)

'**************** Guardar cambios a disco
```

```
        rstFactura.Update
    Next i

'**************** Cerrar los recordsets

    rstCliente.Close
    rstVendedor.Close
    rstFactura.Close
    MsgBox "Las 100 facturas han sido agregadas satisfactoriamente",
    vbInformation, "Facturas Agregadas"
    DoCmd.Close acForm, Me.Name
End Sub
```

Agregar 10 Detalles de Factura a cada Factura.

Se agregarán detalles de factura tomando en cuenta lo siguiente:

1. Debe abrirse la tabla de facturas y recorrer cada factura generada por el código anterior.
2. Por cada factura se asignará un número aleatorio de detalle de esa factura que va de 1 al 10 que será el máximo de productos para esa factura.
3. Una vez elegido el número de registros detalle se procederá a elegir de forma aleatoria los productos que componen la facturación. Tenga en cuenta que se necesita los identificadores de producto, nombres y precios de venta para cada producto elegido.
4. Si al elegir un producto para una factura ya aparece en esta debe sumar a la cantidad del producto ya elegido en la factura o elegir otro (de otra forma aparecería un mismo producto repetido con diferentes cantidades en la factura). Las cantidades de producto se eligen de forma aleatoria: 1..50.
5. Una vez que tenga el total de elementos de detalle se debe actualizar en la factura el monto Total generado por el detalle actual.

8.2.6 Práctica 6: Diseño del menú Principal de la Aplicación

OBJETIVO DE LA PRACTICA

- Saber diseñar el menú principal de una aplicación
- Saber diseñar el formulario de inicio de la aplicación
- Saber Generar acciones del menú por medio de la utilización de macros
- Saber diseñar barras de herramientas personalizadas para la aplicación

DESARROLLO DE LA PRÁCTICA

Últimos detalles en la creación de las interfaces.

En los laboratorios anteriores se han trabajado las distintas pantallas de captación de datos. Ahora falta brindar al usuario un entorno desde el cual este pueda acceder a las distintas interfaces de la aplicación.

Formulario Principal de la aplicación.

Este formulario tendrá los enlaces necesarios para el resto de elementos en la base de datos, en principio las otras interfaces que se han creado. A continuación se dará una breve reseña de los pasos requeridos.

1. Abra un nuevo frm en vista de diseño y quite todos los controles relacionados con manipulación de registros.

2. Inserte un TabsStrip con estilo de botones. Ponga 3 divisiones y llámeles: ENTRADA DE DATOS, CONSULTAS AL SISTEMA E INFORMES GENERADOS.

3. A continuación ingresará un botón por cada formulario que contiene la aplicación. Para dar mayor realce vamos a poner sobre los botones imágenes que hagan referencia a la interfaz que abren. Ver la figura siguiente:

Figura 8.27 Menú principal de “Mi Ventesita”

4. Ahora vamos a establecer un formulario de inicio (splash) para nuestra aplicación. Para ello abra un nuevo formulario en vista de diseño, quite todos los controles referentes a registros, barras de desplazamiento, etc y ponga la propiedad estilo de bordes a ninguno. En segundo lugar modifique las propiedad *Al cronómetro* para que ejecute una rutina que cierre automáticamente este formulario y abra el formulario principal. Muestre el formulario de inicio de 3 a 5 segundos, modificando la propiedad *Intervalo de cronómetro*. Lea la ayuda de Access si se requiere obtener más información sobre estas dos propiedades.

El formulario de inicio luce de la siguiente forma:

Figura 8.28 Formulario de Inicio de “Mi Ventesita”

5. Ahora cada vez que se inicie la aplicación aparecerá la pantalla de inicio y luego el formulario principal. Para lograr este objetivo: abrir el cuadro de inicio del menú herramientas y poner ahí el nombre de la aplicación, un icono y el formulario splash para iniciar la aplicación.
6. Posteriormente asociar una barra de menú y una barra de herramientas al formulario principal.
7. Para la barra de menú:

- Herramientas, Personalizar, Nueva Barra: **MenuPrincipal**.
- Escoger propiedades y cambiar el tipo de barra a **Barra de Menú**.
- Para asignar un nuevo elemento de menú: cambiar a la pestaña comandos y elijir la opción nuevo menú, posteriormente arrastrar y agregar el nuevo menú de la columna derecha. Cambie el nombre del elemento a **&Abrir Formulario**.
- Para insertar nuevos comandos en el nuevo elemento de menú busquese en la columna derecha: *Archivo* y arrástrese el objeto requerido de la columna derecha *Personalizado*. Con esto se puede construir la estructura de Menú que se desee.

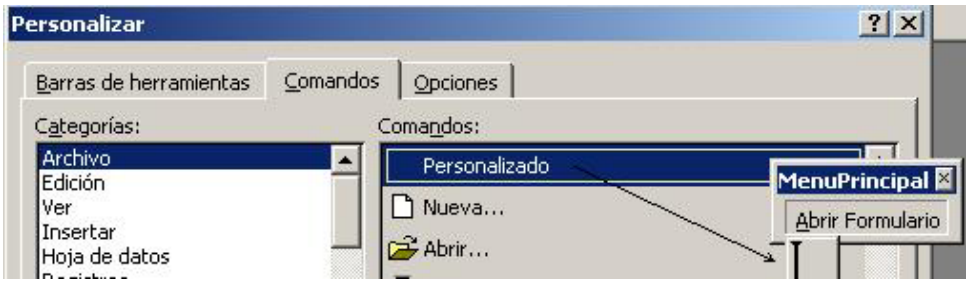


Figura 8.29 Ventana de diseño del Menú

8. Diseñar la siguiente estructura de menú:
&Abrir Formulario [Ver/Modificar Clientes][Ver/Modificar Vendedores] [Ver/Modificar Productos][Agregar Factura][Entrada Bodega].
Inserciones &Masivas [Agregar Facturas de Prueba] [Agregar Detalles de Prueba].

Aplicación [Acerca de Mi Ventesita...].[Salir].
La barra de menú luce así (los iconos a la izquierda de cada elemento son opcionales):

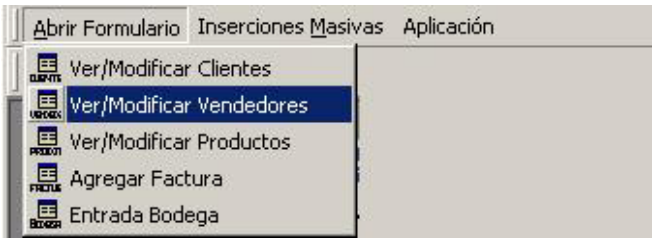


Figura 8.30 Barra de Menú

9. El siguiente paso es asociar una determinada acción a cada elemento de menú, Para ello se utilizarán macros: Pasar a la pestaña de macros y crear una nueva macro, en la parte de acción ponga “Abrir Formulario” y en la parte nombre de formulario “FrmClientes”. Esta es una forma bastante sencilla de asociar acciones a cada elemento del menú. Ahora bien, para enlazar esta macro con el menú, debemos cargar nuevamente el cuadro *personalizar*, Mostrar la barra de menú creada y en la casilla de acción de las propiedades del elemento seleccionar la macro que se debe ejecutar cuando el usuario haga clic sobre ese elemento. Realice esto para el resto de opciones. El elemento Acerca de, muestra un formulario con una descripción más detallada del software. Para el elemento salir se usará la acción proporcionada por Access.

10. Diseñar una barra de herramientas (muy parecido a la barra de menú que se acaba de crear) con enlaces a los siguientes grupos: **grp1->** FrmClientes, FrmProductos, FrmFactura, FrmVendedor y FrmEntradaBodega. **grp2**->Importar, Vincular Tabla. **grp3**->Calculadora, Explorer, Block de notas y Word. La barra de herramientas de lucir así:

Figura 8.31 Barra de Herramientas Propia del Sistema

*__Diseñar una barra de Menú y una Barra de Herramientas__ para FrmClientes, FrmProductos, FrmFactura, FrmVendedor y FrmEntradaBodega, que contengan como mínimo opciones de desplazamiento, manipulación de registros y opciones de búsqueda. Para finalizar asignar a cada formulario los elementos creados.

8.2.7 Práctica 7 :Diseño de Consultas

OBJETIVO DE LA PRACTICA

- Generar consultas de selección utilizando el QBE y el editor SQL de access
- Generar consultas de actualización de tablas
- Generar consultas de borrado de filas
- Saber generar una tabla receptora de las filas resultantes de una consulta
- Generar consultas con parámetros

DESARROLLO DE LA PRÁCTICA

Consultas al sistema.

Se utilizan consultas para visualizar, modificar y analizar datos de la base de datos de formas diferentes. También pueden utilizarse como el origen de registros para formularios, informes y páginas de acceso a datos. Access da soporte a las formas básicas de trabajo con el lenguaje SQL, aunque existen algunas diferencias con respecto al estándar. A continuación se ofrecerán algunos ejemplos del entorno para gestión de consultas tanto utilizando el QBE de access como el entorno SQL.

EJ: Consulta de selección una tabla :**BuscarClientePorNombre**

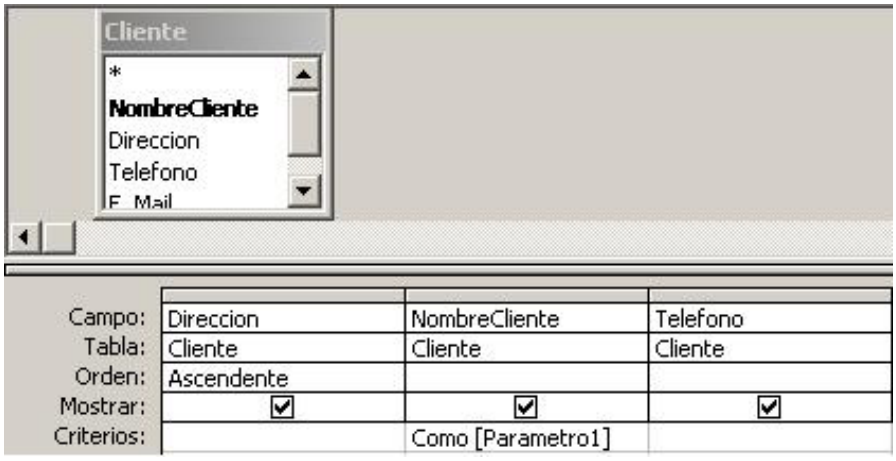


Figura 8.32 Consulta con una Sola Tabla con parámetro

Nota: la zona mostrar hace referencia a la proyección en una instrucción SQL, la zona tabla a From, criterios a la cláusula Where, por último se especifica el orden en que se mostrarán los datos mediante Orden (Order By). En esta consulta se obtiene la dirección, Nombre y teléfono de los clientes que empiezan por [Parametro1], donde parámetro1 es una cadena que el usuario ingresa al momento de ejecutar la consulta. En esta consulta parametrizada se pueden utilizar comodines en el parámetro pasado, por ejemplo si pasamos la cadena a*, el resultado serán todos los clientes que comienzan por la letra a. Si quisiéramos buscar los clientes que tengan el apellido aguilar escribiríamos *aguilar*. Puede cambiar el texto del parámetro por algo más descriptivo para el usuario como [Introduzca el Nombre de Cliente a Buscar]. Esta misma consulta en el entorno SQL de access:

```
SELECT Cliente.Direccion, Cliente.NombreCliente, Cliente.Telefono

FROM Cliente

WHERE

Cliente.NombreCliente Like [Nombre Del Cliente a Buscar]

ORDER BY Cliente.Direccion;
```

Consulta de selección que involucra varias Tablas(QBE):
FacturacionProductoMayorqueCantidad

Se necesita generar una consulta que obtenga de forma dinámica las cantidades facturadas por uno o varios productos en un rango de fechas que se delimita al momento de ejecutar la consulta. La cantidad de producto facturado debe cumplir con un mínimo el cual es suministrado por el usuario al momento de ser ejecutada la consulta. El orden que se muestra será por fecha de forma descendente.

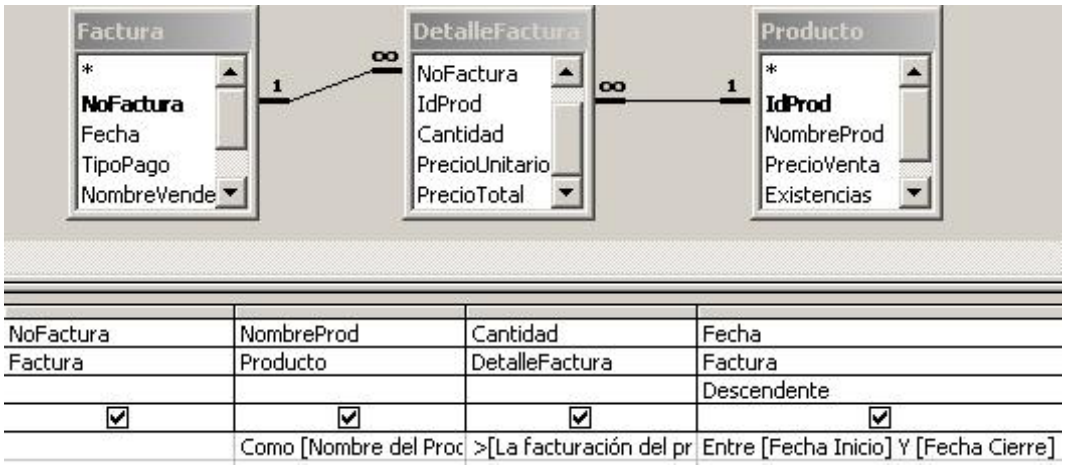


Figura 8.33 Consulta con Varias Tablas Vinculadas

La consulta SQL para el esquema anterior es:

```
SELECT Factura.NoFactura, Producto.NombreProd,
           DetalleFactura.Cantidad, Factura.Fecha
FROM Producto
          INNER JOIN
          (Factura INNER JOIN DetalleFactura ON Factura.NoFactura =
           DetalleFactura.NoFactura)
                ON
          Producto.IdProd = DetalleFactura.IdProd
WHERE
          Producto.NombreProd Like [Nombre del Producto a Buscar])
               AND
           DetalleFactura.Cantidad)>[La facturación del producto mayor que])
               AND
           Factura.Fecha Between ([Fecha Inicio] And [Fecha Cierre])
ORDER BY
           Factura.Fecha DESC;
```

Consulta de actualización: Aumento3porcientoPrecioventadeProducto
La siguiente consulta obtiene como parámetro el nombre de uno o más productos actualizando su precio de venta en un 3% más del valor anterior.

Figura 8.34 Consulta que incrementa los precios de venta en 3%

La consulta SQL es la siguiente:

```
UPDATE Producto
SET
Producto.PrecioVenta = [PrecioVenta]*1.03
WHERE
Producto.NombreProd Like [Producto que va a actualizar];
```

Creación de tabla receptora del resultado de una consulta: CreaciontmpClientesdebenmas10000

se creará una consulta la cual generará la tabla: tmp para volcar en ella todas las Facturas con sus detalles de los clientes que han llevado productos al crédito (TipodePago=no), con montos superiores a los C$10,000 sin tomar en cuenta el

IGV, es decir tendremos que recalcular el monto de lo llevado por cada factura. Para ello agruparemos por número de factura y por nombre de cliente resumiendo en la relación Factura-Detalle. la siguiente figura 8.14 muestra en el QBE la solución:

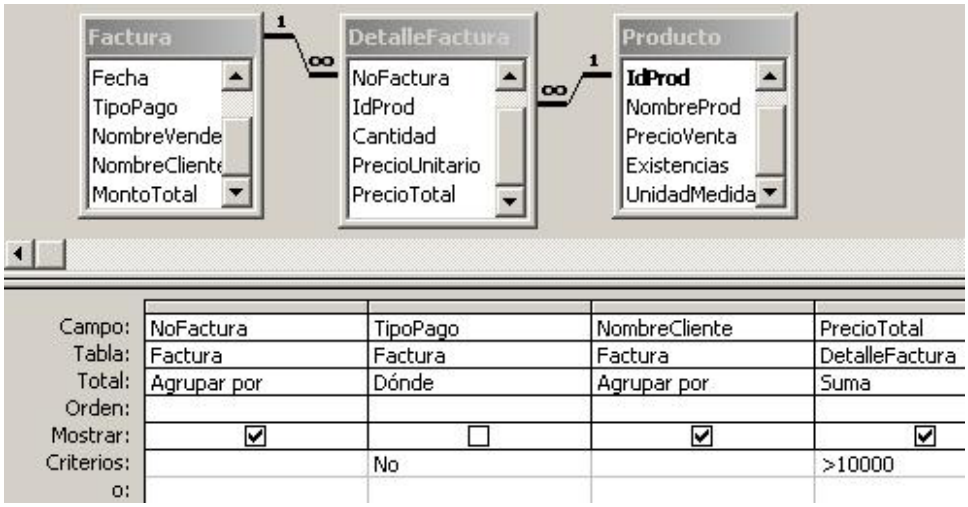


Figura 8.35 Consulta de Selección con agrupación y salida a una tabla

La consulta en SQL:

```
SELECT Factura.NoFactura,Factura.NombreCliente,
       Sum(DetalleFactura.PrecioTotal) AS SumaDePrecioTotal INTO tmp
FROM   Producto INNER JOIN (Factura INNER JOIN DetalleFactura ON F
       Factura.NoFactura = DetalleFactura.NoFactura) ON Producto.IdProd =
       DetalleFactura.IdProd
WHERE (((Factura.TipoPago)=No))
GROUP BY     Factura.NoFactura, Factura.NombreCliente
HAVING (((Sum(DetalleFactura.PrecioTotal))>10000));
```

Borrado de datos: **BorrardelaTablaTMP**

De la nueva tabla creada (tmp) elimine los registros para las facturas que tienen montos entre 10000 y 15000, se pedirá la exclusión de un número de factura que cumpla con la condiciones para ser eliminado. Mire el gráfico siguiente:

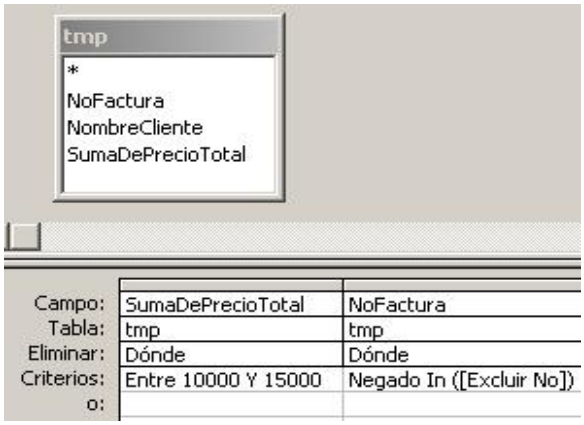


Figura 8.36 Consulta de Borrado de Filas

La sentencia SQL:

```
DELETE tmp.SumaDePrecioTotal, tmp.NoFactura
FROM   tmp
WHERE (((tmp.SumaDePrecioTotal) Between 10000 And 15000) AND
       (tmp.NoFactura) Not In ([Excluir No])));
```

EJERCICIOS Varios: (Puede bajar los datos que se utilizan en los resultados de estos ejercicios de la web)

1. Listar los consolidados de las facturaciones de los cinco Clientes con los 5 montos totales por factura más altos (usar cláusula TOP). **Ver** Ejemplo de **Salida**:

Top5MontosFacturacionCliente : Consult...

NombreCliente	FacturacionTotal
Adalberto Agurcia	C$ 218,733.25
Ariel Aguilar	C$ 200,509.00
Claudia Calderon	C$ 194,362.25
Juan Anton	C$ 161,208.00
Martha Berrios	C$ 155,821.75

Figura 8.37 Los clientes que tiene los cinco mayores Montos

2. Listar al cliente cuya suma de montos por facturación sea el más alto.

NombreCliente	SumaDeMontoTotal
Adalberto Agurcia	C$ 218,733.25

Figura 8.38 Consulta con la mayor suma de montos

3. Mostrar los 10 productos más vendidos para un rango de fechas especificado en tiempo de ejecución de la consulta. (la consulta muestra el TopTen 01/04/2004 al 01/05/2004).

NombreProd	SumaDeCantidad
Calendarios de bolsillo	905
Margarina Unimar	865
Flor de Caña Extra Lite	776
Pan barra Bimbo	765
Belmont Suave-20	722
Jamón Delmor	679
Tamales Pisques	668
Galletas Soda Pozuelo	647
Flor de Caña Gran Reserva	637
Repollo Criollo	602

Figura 8.39 Productos mas vendidos en un dterminado rango de fechas

4. Listar las fechas de las últimas facturaciones hechas por los vendedores antes de una determinada fecha pasada en tiempo de ejecución (en el gráfico se utilizó antes de 01/05/2004).

	NombreVende	ÚltimaFechaFa
▶	Anixia Lindo	20/03/2004
	Maria Lezama	17/01/2004
	Karla Esquivel	09/01/2004

Figura 8.40 Facturaciones de vendedores antes de una fecha

5. Seleccionar el número de factura, tipo de pago=credito, nombre del cliente, dirección, teléfono y correo electrónico, número de meses que no ha pagado. Para aquellos morosos con más de un mes sin paga desde el 01/01/2004. Puede redondear los meses con Int(Expresión). En total son 183 regs.

No Factura	Num Meses Desde 01Enero	NombreCliente	Direccion	Telefono	E_Mail
1850	8	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1854	10	Ana Blandon	Billares Lopez 3c al norte y 2c	311-4774	ablandon2000@yahoo.c
1857	4	Ana Blandon	Billares Lopez 3c al norte y 2c	311-4774	ablandon2000@yahoo.c
1859	6	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1860	4	Juan Anton	La Proquinsa 1c. 1/2 al sur,	311-1115	janton@hotmail.com
1863	5	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1864	10	Claudia Calderon	Chalecito 2c y 1/2 al sur.	311-9999	claudia_calderon@amig
1865	5	Francisco Castro	Del tenis 2c abajo.	311-0265	francas@yahoo.com
1868	1	Adalberto Agurcia	Hotel Colonial 20v al norte, #23	311-2454	adal2154@yahoo.com
1870	3	Martha Berrios	Del chenchuntle 2 piezas abajo,	311-9878	marbe@yahoo.com
1871	8	Francisco Castro	Del tenis 2c abajo.	311-0265	francas@yahoo.com
1875	5	Adalberto Agurcia	Hotel Colonial 20v al norte, #23	311-2454	adal2154@yahoo.com
1878	6	Francisco Castro	Del tenis 2c abajo.	311-0265	francas@yahoo.com
1879	1	Juan Anton	La Proquinsa 1c. 1/2 al sur,	311-1115	janton@hotmail.com
1881	3	Karla Berrios	Sutiava, del indio 2c. Al sur, #22	311-5559	karlaberrios99@latinmail
1883	6	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1884	3	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1885	9	Karla Berrios	Sutiava, del indio 2c. Al sur, #22	311-5559	karlaberrios99@latinmail
1886	8	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1888	1	Francisco Castro	Del tenis 2c abajo.	311-0265	francas@yahoo.com
1892	3	Martha Berrios	Del chenchuntle 2 piezas abajo,	311-9878	marbe@yahoo.com
1893	3	Adalberto Agurcia	Hotel Colonial 20v al norte, #23	311-2454	adal2154@yahoo.com
1894	5	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1897	7	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1902	9	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1903	8	Ariel Aguilar	Los Zarzales #234	311-9966	arielag@hotmail.com
1904	4	Juan Anton	La Proquinsa 1c. 1/2 al sur,	311-1115	janton@hotmail.com
1908	7	Ana Blandon	Billares Lopez 3c al norte y 2c	311-4774	ablandon2000@yahoo.c

1911	8	Claudia Calderon	Chalecito 2c y 1/2 al sur.	311-9999	claudia_calderon@amig
1912	6	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1913	9	Mario Bustamante	Posada del sol 3 etapa, casa	311-5544	mbust223@hotmail.com
1918	1	Martha Berrios	Del chenchuntle 2 piezas abajo,	311-9878	marbe@yahoo.com
1919	8	Adalberto Agurcia	Hotel Colonial 20v al norte, #23	311-2454	adal2154@yahoo.com
1920	8	Juan Anton	La Proquinsa 1c. 1/2 al sur,	311-1115	janton@hotmail.com
1921	2	Juan Anton	La Proquinsa 1c. 1/2 al sur,	311-1115	janton@hotmail.com
1924	2	Adalberto Agurcia	Hotel Colonial 20v al norte, #23	311-2454	adal2154@yahoo.com

Figura 8.41 Ejemplo de resultado de consulta del problema 5

*Enlace las consultas realizadas al panel principal.

8.2.8 Práctica 8: Generación de Informes

OBJETIVO DE LA PRACTICA

- Conocer los diferentes componentes de la herramienta para generar informes en Access
- Saber crear y utilizar las diferentes zonas en que se divide el ambiente del informe
- Saber generar formatos de salida en el encabezado y pie de página del informe
- Saber generar informes con agrupaciones de los datos, sub totalizaciones por grupos y grandes totales

DESARROLLO DE LA PRÁCTICA

Informes.

Concepto: Los informes son herramientas utilizadas por el programador de la Base de Datos para brindar al usuario final datos en formato impreso. Los informes muestran la información desde listas simples de registros hasta agrupaciones con cálculos complejos sobre cada grupo de datos. Access permite crear informes basados en tablas o consultas, si la fuente de datos se encuentra almacenada en una tabla del sistema puede utilizar como fuente de datos la misma tabla o una vista de la misma; si los datos que desea mostrar se encuentran en distintas tablas cree una consulta con la información necesaria y dele formato al informe para mostrar la información dispuesta a su voluntad.

Familiarización con el entorno:

Algunos ejemplos de informes a partir de los datos de MiVentesita

Listado de todos los Clientes en MiVentesita

1. Crear un nuevo informe en tiempo de diseño.
2. Escoger como fuente de datos la tabla cliente.
3. En la sección *encabezado de página* ponga los elementos de encabezado de la lista: **Nombre del Cliente, Dirección y Teléfono** (Utilizando Etiquetas).

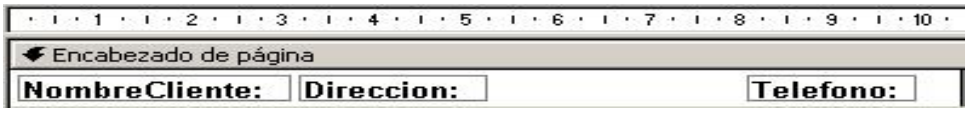


Figura 8.42 Encabezado de página del Informe

4. hacer click en *lista de campos* de la fuente de datos, agregar todos los campos de la tabla clientes a la sección *detalle* (Quite las etiquetas que se agregan junto con cada campo).

Figura 8.43 Área de Datalle del Informe

5. Para finalizar en la sección *pié de página* vamos a agregar la hora y fecha actual. Para realizar esto del menú *insertar* elejir [Fecha y Hora] se insertará un elemento que de forma automática reflejará la fecha y hora actual.

Figura 8.44 Pie de página del Informe con fecha y Hora

6. Observe los resultados y guarde el informe como ListaDeClientes.

Consolidados de facturas por producto mediante rangos de fecha.

1. Crear un nuevo informe en vista de diseño.
2. La fuentes de este informe es una consulta (Se debe crear y con el nombre->Consolidados de FacturacionXProductorango de Fechas):

```
SELECT Factura.Fecha, Producto.NombreProd, DetalleFactura.Cantidad,
DetalleFactura.PrecioTotal
FROM Producto INNER JOIN (Factura INNER JOIN DetalleFactura ON
Factura.NoFactura = DetalleFactura.NoFactura) ON Producto.IdProd =
DetalleFactura.IdProd
WHERE (((Factura.Fecha) Between [Fecha de inicio] And [Fecha final]));
```

3. Asignar esta consulta como fuente de datos para el informe.
4. Observar que al correr la consulta se cuenta con un grupo de datos que permiten la **agrupación por fecha y producto**. De esta agrupación se obtendrán consolidados de montos vendidos y cantidades vendidas por producto.

5. Para realizar lo propuesto en el punto anterior se comenzará por colocar un encabezado de página para el informe actual. (Ver Fig.).

Figura 8.45 Inicio de Diseño del Informe

6. Ahora se estudiarán los componentes de agrupación del informe: Un primer **grupo de fechas** en que se realizaron las facturas, el segundo **grupo** estará compuesto por los **productos** que se facturaron en esa fecha, por último **para cada producto** se mostrarán **las cantidades y subtotales facturados**. Para finalizar al cerrar un grupo de fechas se realizará un consolidado general de ventas realizadas en esa fecha. Los **pasos** siguientes hacen referencia a como este aspecto debe de realizarlo el alumno:

a. Seleccionar **Ordenar/Agrupar**  utilizar los siguientes parámetros para Fecha:

Figura 8.46 Diseño de Formato de salida de la Fecha

b. Los parámetros para Nombreproducto son:

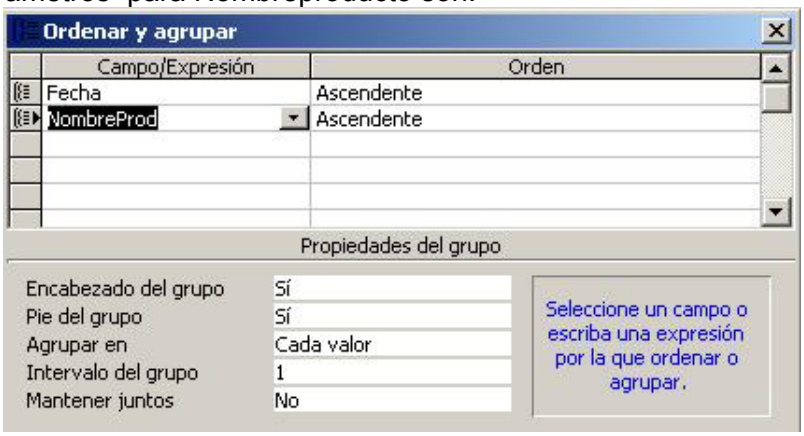


Figura 8.47 Formato de Salida para el Nombre del Producto

c. Cerrar el cuadro de diálogo y notándose que existen 4 nuevos componentes en la estructura estándar del informe: **Encabezado** Fecha, Encabezado Nombreprod, **Pié** Fecha y Pié NombreProd.

d. Ahora en el **encabezado de grupo** Fecha ubique su **campo representante**: Fecha. En la sección detalles únicamente debe quedar la nombreprod, cantidad y el preciototal.

e. El siguiente paso es **indicar** lo **que** se desea **hacer con estos dos campos**. En la sección Pié Nombreprod, agreguese una caja de texto independiente, ubicarla debajo de preciototal. Escriba en esta caja **=Suma(preciototal)**, para que se calculen lo subtotales generales por producto.

f. Repetir este paso en la sección Pié de fecha para calcular los totales generales por venta de productos. (Ver la figura 8.16 de la estructura del informe).

Figura 8.48 Ejemplo de Diseño del Informe

g. Para finalizar nombrese el informe de la siguiente manera: Consolidados
h. Por Producto Rango Fechas. (Ver salida en la Fig. siguiente).

Figura 8.49 Salida del Informe

Ejercicios Varios:

1. Generar un informe de las facturas que han sido facturadas como crédito por nombre de cliente y con las fechas y montos que debe cada cliente, también debe ir incorporada la dirección y el teléfono del cliente.
2. Generar un informe del Top Ten de productos más vendidos entre el 1 de enero del 2004 hasta la fecha actual donde se corre el informe.
3. Crear un informe con los montos de facturación generados por los vendedores en un rango de fechas introducidas en tiempo de ejecución. Según el monto el vendedor cobrará el 1% del total facturado.
4. Crear un informe con el nombre de los productos donde la suma de las cantidades compradas en todo el primer semestre del 2004 superan los 100 productos.
5. Generar un informe que contega el nombre y la antigüedad de cada empleado (vendedor).

8.2.9 Práctica 9: Introducción al SQL Server

En esta práctica el estudiante instalará el software, se familiziará con el ambiente de trabajo de MicroSoft SQL Server 7, se diseñarán tablas utilizando SQL, así como consultas básicas. Finalmente se hará una conexión a una tabla SQL Server desde Access

OBJETIVO DE LA PRACTICA

- Saber instalar y conocer el ambiente básico de trabajo del SQL Server
- Saber Diseñar Tablas usando el DDL de SQL
- Conocer aspectos básicos del Transac-SQL del SQL Server
- Diseñar diferentes tipos de consultas en SQL Server
- Ser capaz de añadir filas de una tabla de una base de datos de SQL Server,
  utilizando como interfaz MS Access

INTRODUCCIÓN TEÓRICA

Que es el SQL Server?

Microsoft SQL Server es un sistema de administración de bases de datos relacionales (RDBMS;Relational Database Management System) que utiliza la tecnología cliente servidor con un alto rendimiento y eficiencia.

Se ha diseñado para admitir un elevado volumen de procesamiento de transacciones, además de aplicaciones relacionadas con el almacén de datos y de ayuda en la toma de decisiones, cuenta con un potente motor para admitir una gran variedad de aplicaciones exigentes como el procesamiento de transacciones en línea. Por otra parte cuenta con una implementación de SQL denominada Transact-SQL (T-SQL) la cual proporciona una enorme potencia y flexibilidad para recuperar información de las bases de datos del sistema cumpliendo con la norma SQL-92 del ANSI (American National Standars Institute). Además de dar soporte a los comandos SQL este proporciona capacidades que van más allá de las implementaciones típicas del SQL como la proyección de un análisis multidimensional de una manera eficiente en el servidor de la base datos mediante operadores especiales, soporta operadores como case, if then else que simplifican en gran med ida el desarrollo de aplicaciones.

Otras Características
SQL Server proporciona al usuario otros aspectos que son de gran utulidad en el diseño de aplicaciones:

Procedimientos Almacenados
Son un conjunto de instrucciones SQL almacenadas dentro de una base de datos de SQL Server, que pueden ser invocados desde una aplicación, ejecutándose mas rápido que los archivos por lotes de comandos SQL y otros como:
Integridad Referencial Declarativa, Tipos de datos Definidos por el usuario, restricciones de comprobación y reglas de validación (Check), valores pre determinados, Disparadores para imponer restricciones de integridad complejas que se ejecutan de forma automática.

Procesamiento de transacciones
El procesamiento de las transacciones en SQL Server asegura que todas se lleven a cabo como una única uidad de trabajo. Las transacciones de SQL Server tiene la propiedad ACID(Atomicidad, coherencia, aislamiento y durabilidad).

Rendimiento Multiusuario
La eficiencia del modelo de subprocesos de SQL Server se ve superado por su rendimiento multisuario. SQL Server es capaz de manejar de manera eficiente cientos e incluso miles de usuarios activos de forma simultanea.

Seguridad
SQL Server proporciona numerosos niveles de seguridad. En la capa mas externa la seguridad de registro de acceso de SQL server está integrada directamente con la seguridad de Windows NT. Con esta seguridad integrada puede aprovechar las ventajas de las características de seguridad de Windows NT, tales como el cifrado de contraseñas, su caducidad y las restricciones de longitud.

Desarrollo de la práctica
1) Como parte de la práctica el estudiante deberá instalar el SQL Server, opción Desk Top

2) Creación de una Base de Datos
Para ello:
(a) Cargar SQL Server Enterprise Manager, ver figura 8.17 abajo
(b) Escoger Bases de Datos y con el botón secundario del ratón escoger “Nueva Base de Datos” y darle el nombre “Base de Datos Ejemplo”
(c) Escoger la Base de Datos a utilizar, en este caso “Base de Datos Ejemplo”

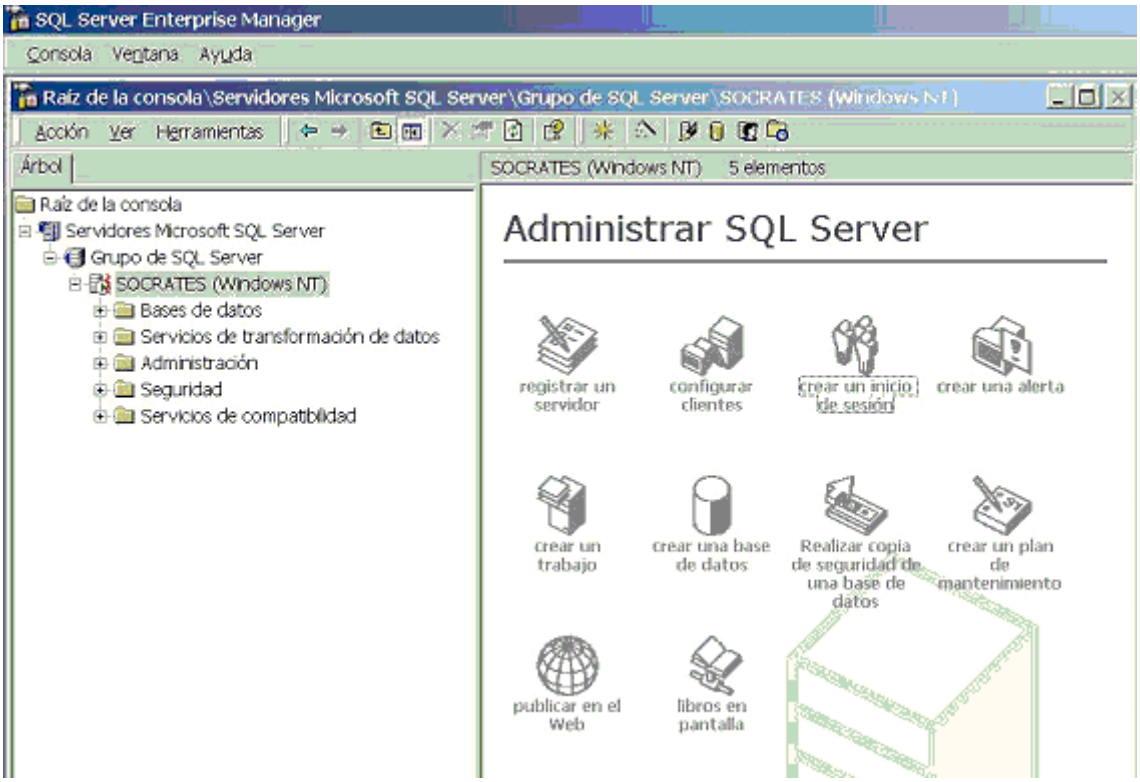


Figura 8.50 Enterprise Manager de SQL Server

3) Dado el siguiente diagrama E/R Ver figura 8.18, utilice las herramientas de SQL Server para diseñar el esquema relacional del sistema.

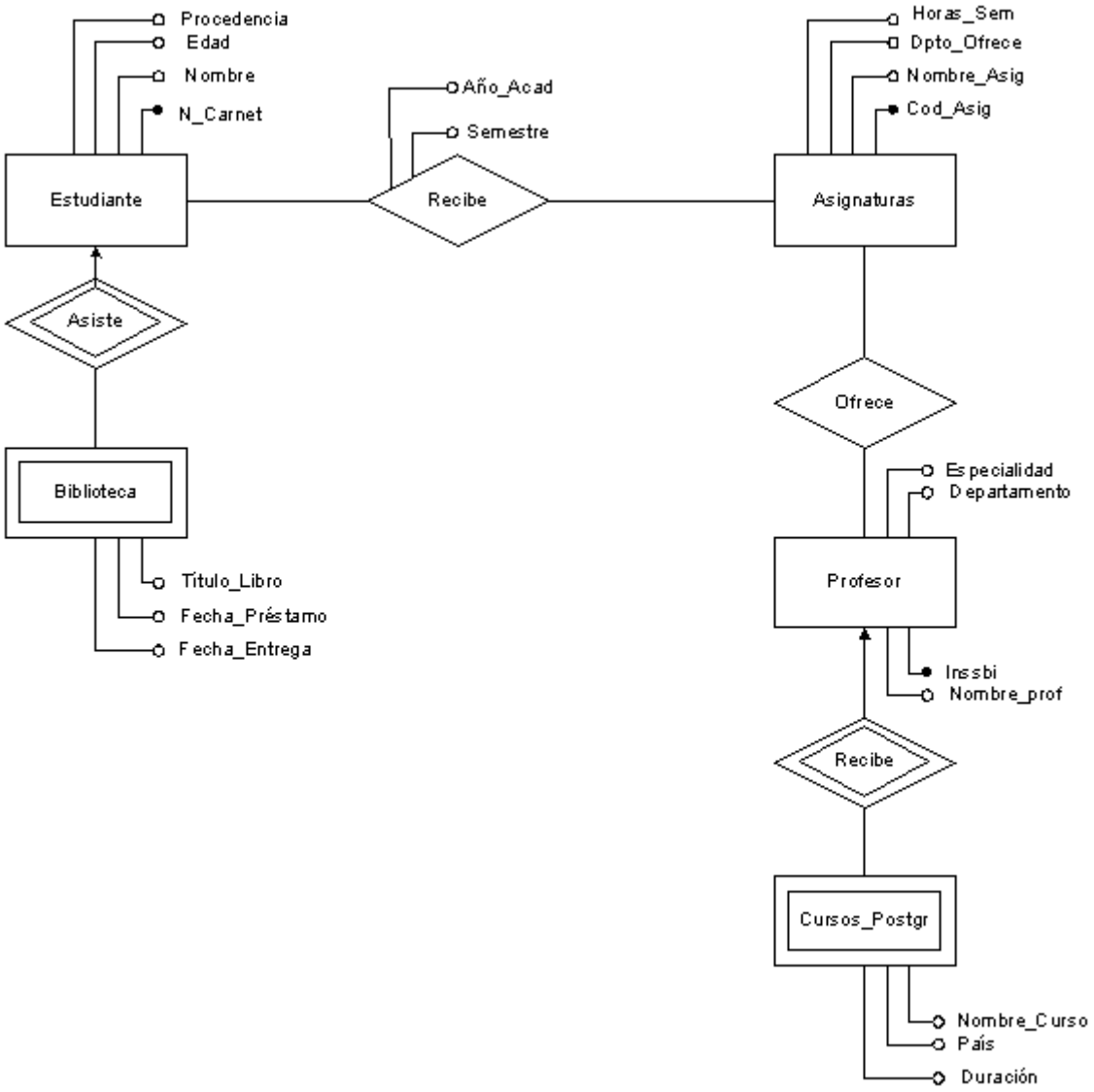


Figura 8.51 Diagrama E/R del Sistema

Los principales pasos para realizar esta Actividad son los siguientes:

a) Active el SQL Server Enterprise Manager
b) Escoja la Base de Datos, en este caso “Base de Datos Ejemplo”
c) Usando el componente *Tablas* que cuelga de “Bases de Datos Ejemplo”, diseñe cada una de las tablas presentes en el diagrama
d) Escoja la Herramienta para el diseño de Esquemas relacionales: Diagramas
e) Diseñe el esquema relacional trasladando las tablas que genera el diagrama E/R al área de diseño
f) Genere los respectivos vínculos entre los conjuntos entidades y los conjuntos relaciones “arrastrando” la llave primaria hacia la tabla que representa el conjunto relación
g) Guarde el gráfico resultante

El Esquema relacional quedará de la siguiente manera:

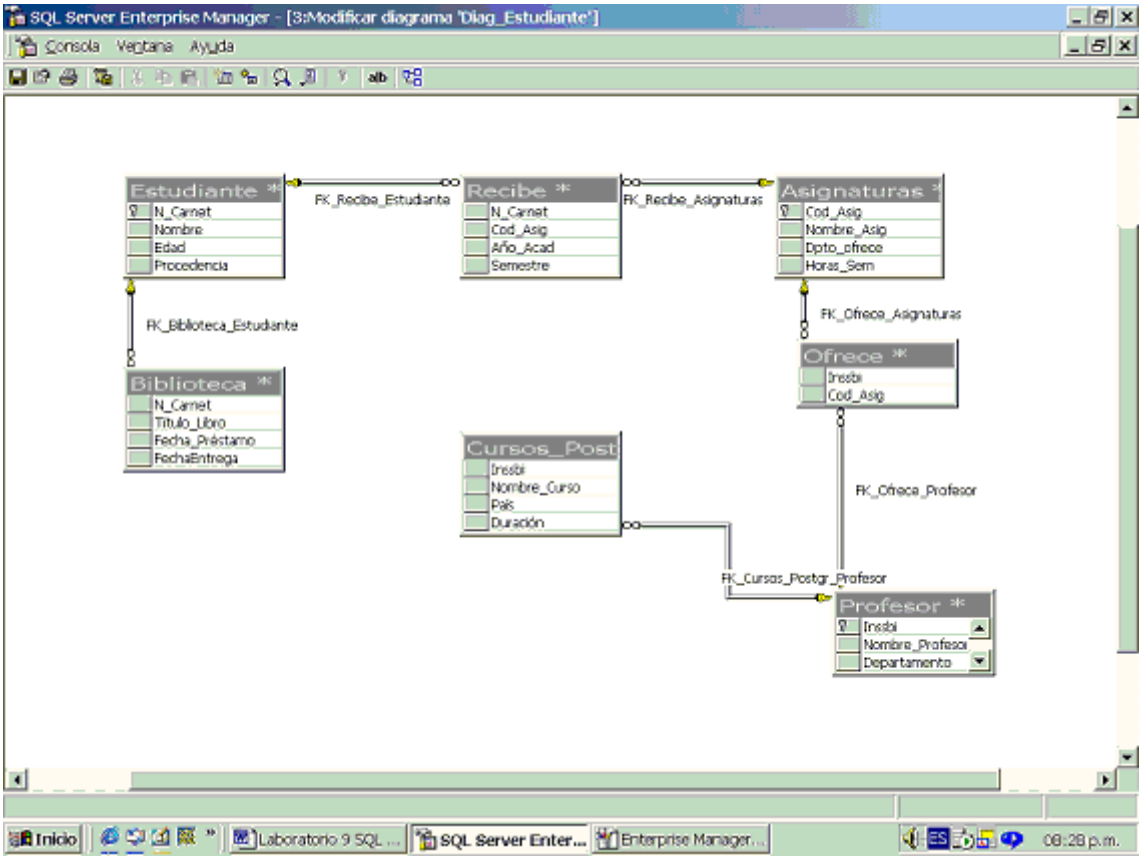


Figura 8.52 Ventana de Diseño del Esquema Relacional

4)Utilizando la herramienta para inserción de datos insertar las filas indicadas abajo en cada una de las tablas:

Tabla Estudiante

N_Carnet	Nombre	Edad	Procedencia
2132	Roberto	22	Managua
2234	Juan	21	Masaya
3578	Carlos	20	Managua
2268	María	22	Managua
3567	Rodolfo	22	Masaya

Tabla Asignaturas

Cod_Asig	Nombre_Asig	Dpto_Ofrece	Horas_Sem
A-100	Cálculo I	Matemáticas	6
A-101	Álgebra Lineal	Matemáticas	4

A-105	Programación I	Computación	6
A-110	Estructura de Datos	Computación	4
A-115	Base de Datos I	Computación	6

Tabla Profesor

Inssbi	Nombre_Prof	Especialidad	Departamento
324895	Ernesto	Base de Datos	Computación
334567	Rafael	Análisis Funcional	Matemáticas
356789	José Miguel	Álgebra	Matemáticas
223467	Ana María	Programación	Computación
256756	Aldo	Arquitectura	Computación

Tabla Recibe

N_Carnet	Cod_Asig	Año_Acad	Semestre
2132	A-100	2000	I
2132	A-101	2000	I
2132	A-100	2001	I
3578	A-105	2000	II
3578	A-110	2000	II

Tabla Biblioteca

N_Carnet	Título_Libro	Fecha_Préstamo	Fecha_Entrega
3567	El Quijote	21/12/02	21/01/03
3567	Azúl	21/12/02	21/01/03
2268	El Principe	6/2/02	18/2/02
2268	La Laguna azul	15/2/03	26/02/03
2234	De aquí a la eternidad	10/05/03	28/05/03

Tabla Cursos_Postgr

Inssbi	Nombre_Curso	País	Duración
324895	Evaluación Sistemática	Nicaragua	1
324895	Métodos no paramétricos	España	9
356789	Java Avanzado	Nicaragua	2
356789	Visual C++	España	6

Tabla Ofrece

Inssbi	Cod_Asig
324895	A-100
334567	A-101
324895	A-105

334567	A-110
356789	A-105

5) Utilizando el QBE de SQL Server obtenga :

a) La procedencia de los Estudiantes becados
b) El Nombre de cada profesor y la cantidad de cursos de post grado efectuados en el extranjero
c) El Nombre de los profesores que han dado clase a estudiantes becados de "Masaya"

Nota: Se accede al QBE, escogiendo *vistas* que cuelga de "Base de datos Ejemplo", luego utilizando el botón derecho escoger "Nueva Vista", en la figura 8.19 abajo se muestra el área de diseño del QBE del SQL Server, donde el icono:

es utilizado para cargar las tablas que serán utilizadas en el diseño de la vista.

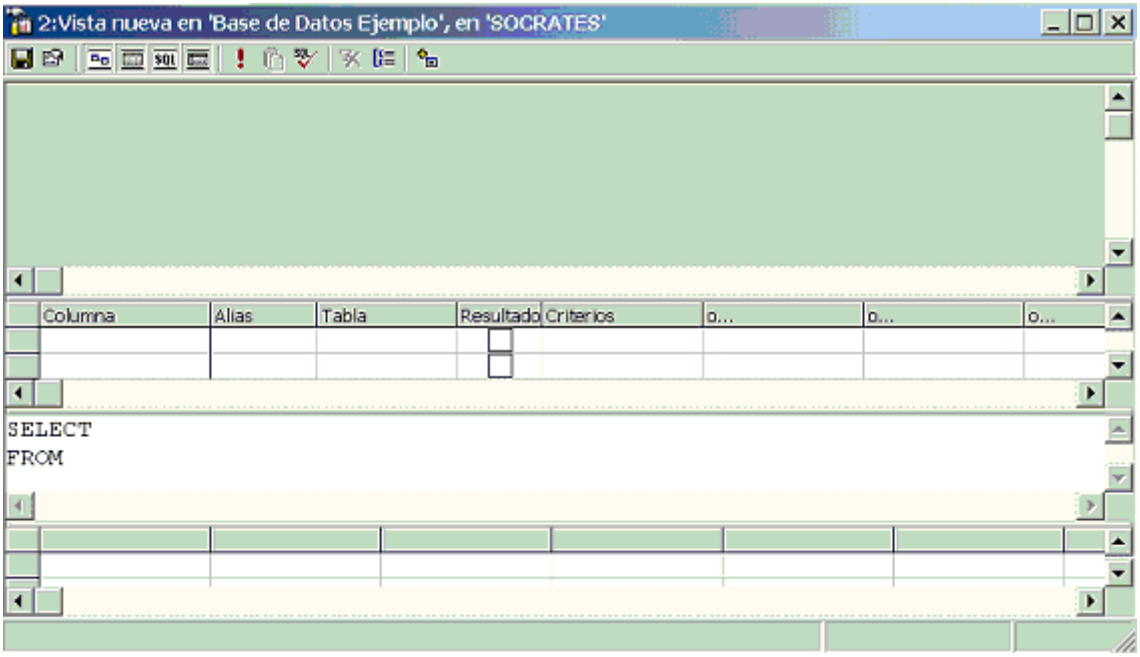


Figura 8.53 Area de diseño del QBE del SQL Server

6) Utilizando el comando create table visto en las conferencias teóricas y el Analizador de Consultas de SQL Server, crear las tablas del diagrama E/R de la figura 8.20 con la siguientes restricciones adicionales:
a) Estudiante1.Edad no debe de ser mayor de 100
b) Las opciones de Estudiante1.Procedencia son: "Managua","Masaya", "León" y "Granada".
c) El valor por defecto de Estud_asig1.Año_Acad es 2000
d) Asignaturas1.Horas_Sem tiene un valor mínimo de 2 y un máximo de 8

e) Se deben especificar las llaves primarias de las tres tablas así como las claves ajenas

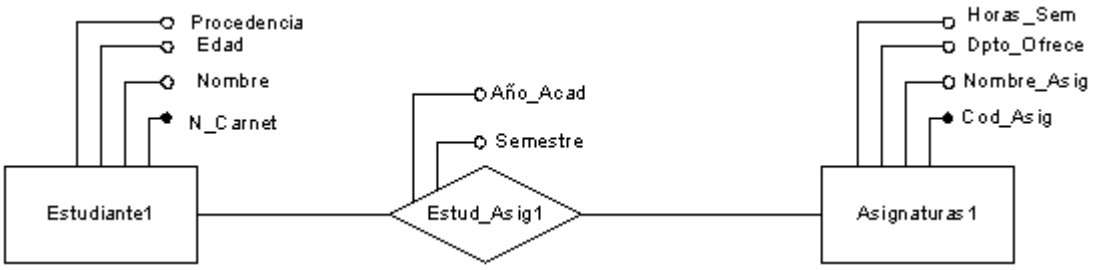


Figura 8.54 Trozo de Diagrama E/R base para creación de tablas (Comando Create Table)

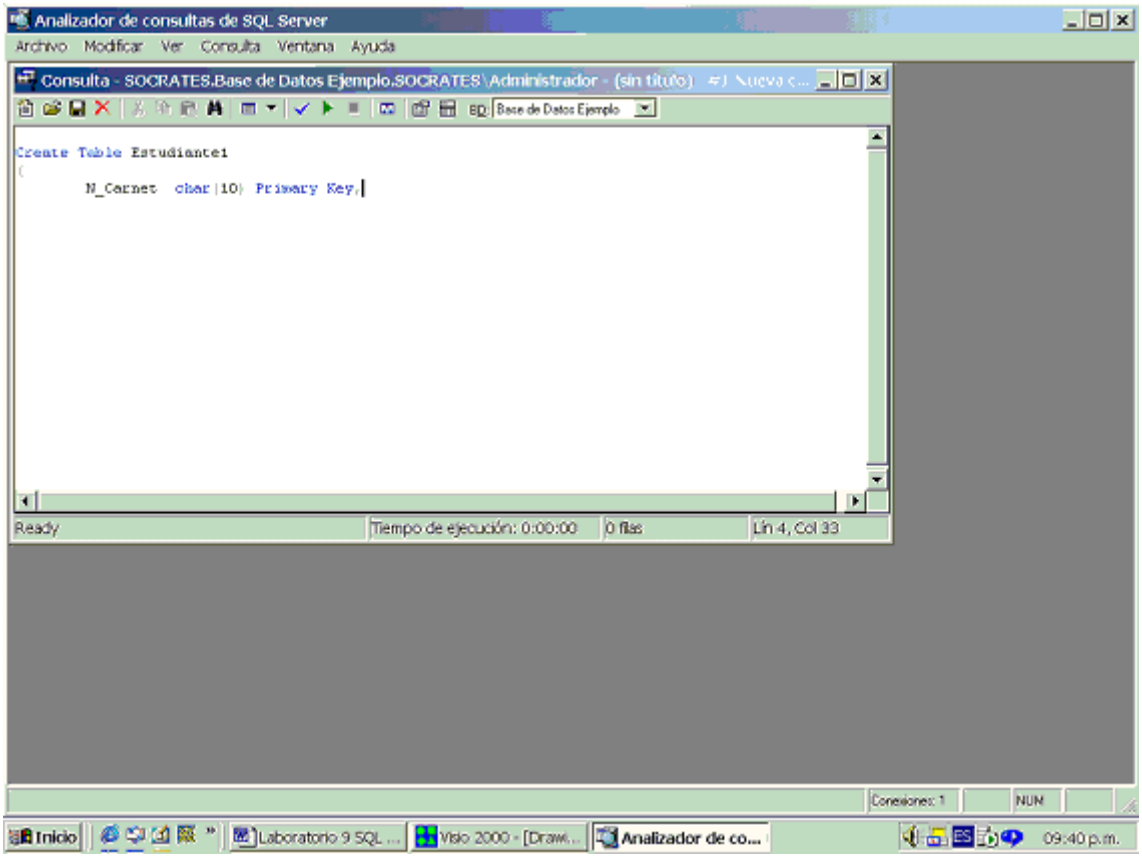


Figura 8.55 Ambiente diseño de Consultas del SQL Server

7) Utilizar el Sistema Gestor de Base de Datos Access para introducir 5 filas de datos en la tabla *Estudiante* (mediante un formulario) de la *Base de Datos Ejemplo,* diseñada previamente en SQL Server. Siga los pasos siguientes:

a) Cargue el panel de control del menú inicio del sistema operativo, y Escoja *Herramientas Administrativas + Origenes de Datos (ODBC) y* escoja DSN de Archivo, se presentará un formulario similar al de la figura 8.22

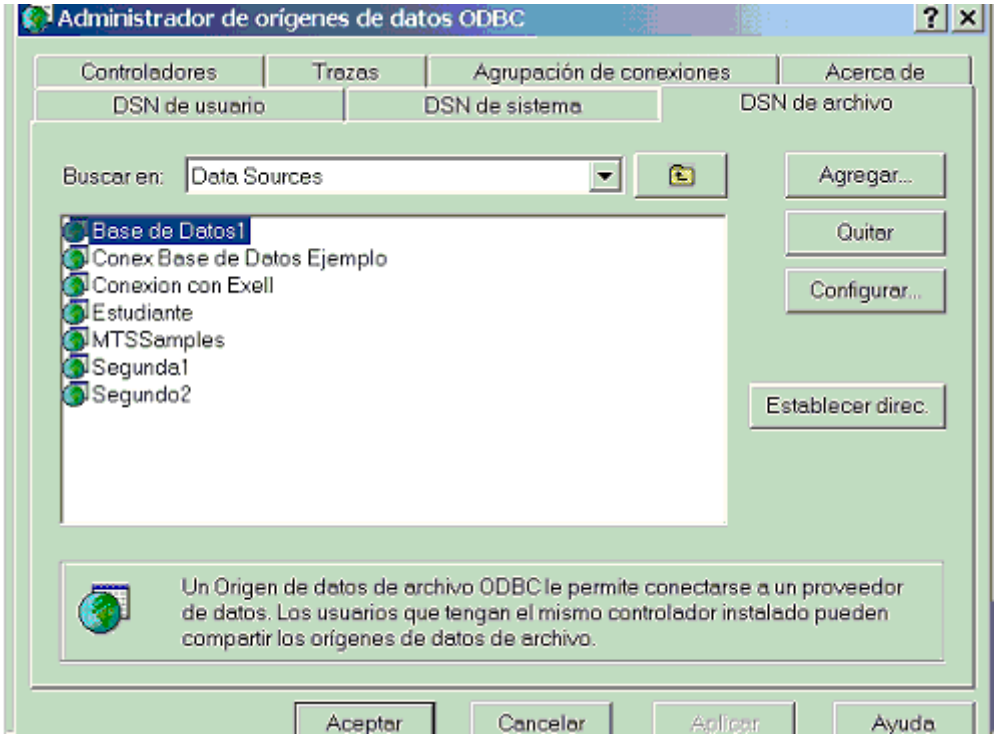


Figura 8.56 Origenes de Datos de archivo ODBC

b) Escoja Agregar y seleccionar el controlador de SQL Server para crear un nuevo origen de datos
c) Pulse siguiente e introduzca el nombre que se le dará al archivo de conexión, digamos “Conexión con SQL Server”, luego Finalizar
d) Llenar las cajas de texto del Formulario “Crear un Nuevo Origen de Datos Para SQL Server”, ver figura 8.23, abajo

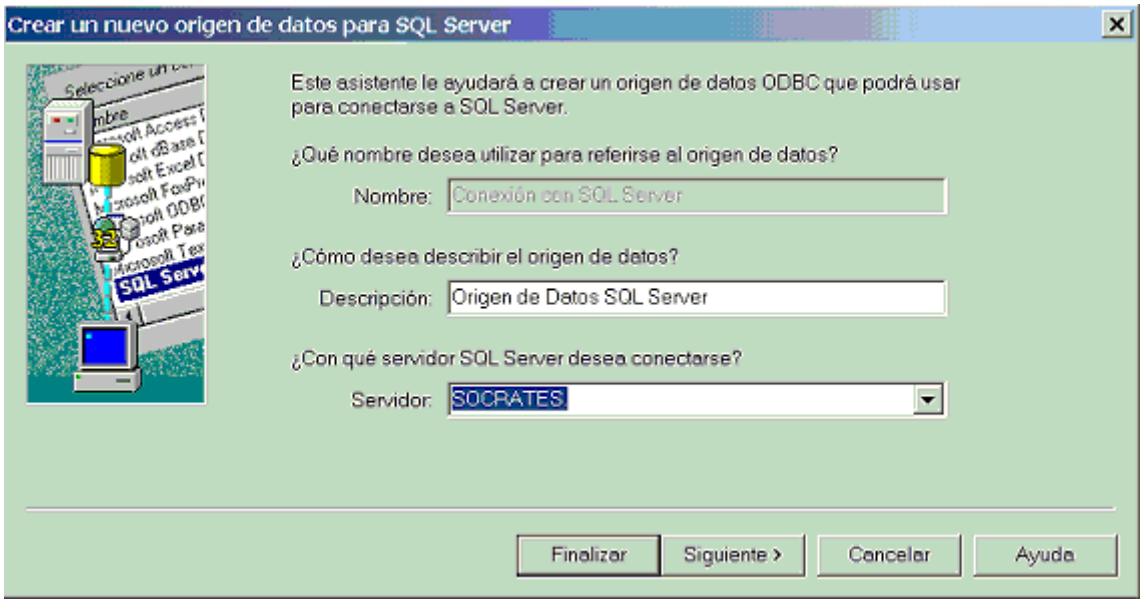


Figura 8.57 Formulario con la información del origen de Datos

e) En el siguiente formulario: Escoja la autenticación de NT (está por defecto)
f) En el siguiente formulario escoger a base de Datos Ejemplo como la base de datos predeterminada y finalizar
g) En el siguiente formulario realice una prueba de la conexión y compruebe que esta se ha realizado de forma satisfactoria y note como la nueva conexión se ha agregado a la carpeta Data Sources

h) Cargue el SGBD MS Access y crear una Base de Datos, digamos "Conexión con Base de Datos Ejemplo"

i) De la Opción archivo del menú principal Escoja "Obtener Datos externos" + "Vincular Tablas" en el formulario vincular escoger de la lista "Tipo de archivo" la opción ODBC Databases lo cual nos lleva de forma automática a la lista de conexiones ODBC diseñadas que se encuentran en la carpeta Data source, de la lista escoger la recién diseñada: "Conexión SQl Server" con lo cual aparece un listado de las tablas de "base de Datos Ejemplo" Convertidas a ODBC, escoger dbo.Estudiante, insertándose en el listado de tablas de MS access ver figura 8.58 abajo

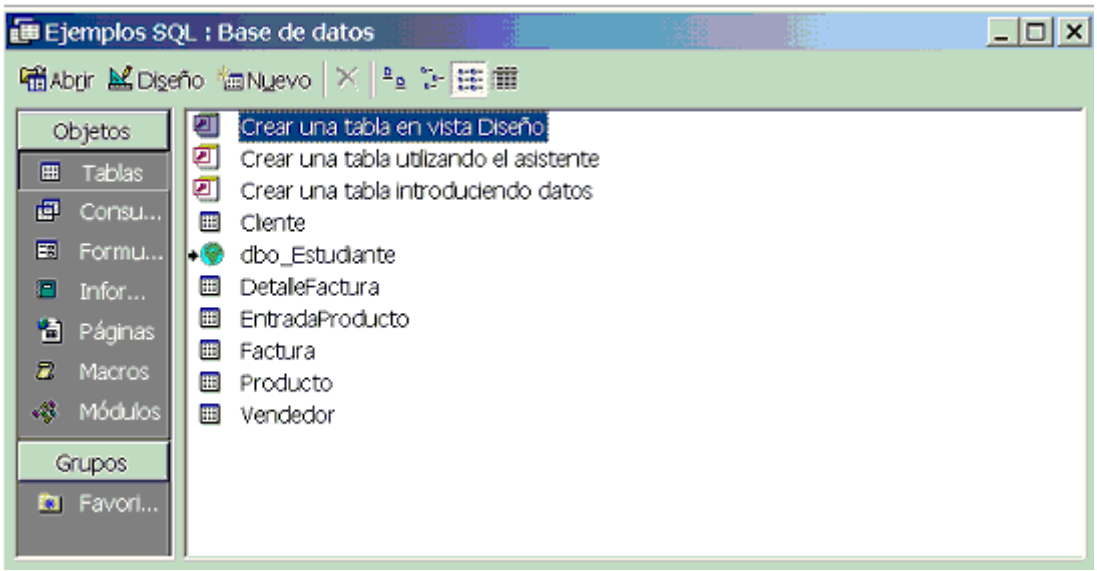


Figura 8.58 Tabla Dbo_Estudiante incorporada al Listado de Tablas

j) En Función de la Tabla vinculada dbo_Estudiante, Diseñe un formulario con base en esta tabla e introduzca los 5 registros antes indicados, verifique en SQL Server que los datos se han introducidos adecuadamente.

9 BIBLIOGRAFÍA

Bibliografía Básica

- **Silverschatz, Korth,Sudarschan. 2002. Fundamentos de Base de Datos (Cuarta edición). España: McGraw-Hill/Interamericana de España.**
- **C.J. Date. 1990. An Introduction to Database Systems (Quinta Edición), Addison Wesley.**
- **Adoración de Miguel , Mario Piattini. 1999. Fundamentos y Modelos de Bases de Datos(Segunda edición). España: RA-MA**
- **Irene Luque Ruiz, Miguel Angel Gomez Nieto. 1997. Diseño y Uso de Bases de Datos Relaciónales. España: RA-MA.**
- **Cary N. Prague, Michael R. Irwin. 1997. El Libro de Access 97. Ediciones Anaya Multimedia**
- **Evan Callagan, 1997. Microsoft access 97 / Visual Basic. Paso a Paso. McGraw-Hill.**
- **Alfons González. 1997. Visual Basic Programación Cliente/Servidor. RA-MA**
- **Ron Soukop. 1997. A Fondo Microsoft SQL Server. Microsoft® Press**

Referencias en Internet

- http://sqlzoo.net/
- http://www.w3schools.com/sql/default.asp
- http://elies.rediris.es/elies9/index-5.htm
- http://sel.unsl.edu.ar/licenciatura/ingsoft1/Apuntes/p3.df
- http://www.desarrolloweb.com/manuales/9/
- http://www.monografias.com/trabajos5/norbad/norbshl