EVALUACION DEL PERSONAL DOCENTE

CURSO: ANALISIS, DISEÑO E IMPLEMENTACION

FECHA: 2-23 MAYO ESCALA DE EVALUACION: 1 A 10

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACION CON EL ASISTENTE	PUNTUALIDAD
ING. JESSICA BRISEÑO CORTES				
ING. JOSE A. CHAVEZ FLORES				
ING. RAMON RAMIREZ				

EVALUACION DE LA ENSEÑANZA

ORGANIZACION Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD LOGRADO EN EL CURSO	
ACTUALIZACION DEL CURSO	
APLICACION PRACTICA DEL CURSO	

EVALUACION DEL CURSO

CONCEPTO	CALIF
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDACTICO UTILIZADO	

1.- ¿LE AGRADO SU ESTANCIA EN LA DIVISION DE EDUCACION CONTINUA?

SI	NO

SI INDICA QUE □NO□ DIGA PORQUE.

COORDINACION CURSOS DE COMPUTO
CENTRO DE INFORMACIÓN Y DOCUMENTACION

2.- MEDIO A TRAVES DEL CUAL SE ENTERO DEL CURSO:

PERIODICO EXCELSIOR		FOLLETO ANUAL		GACETA UNAM		OTRO MEDIO	
PERIODICO LA JORNADA		FOLLETO DEL CURSO		REVISTAS TECNICAS			

3.- ¿QUE CAMBIOS SUGERIRIA AL CURSO PARA MEJORARLO?

_____

_____

4.- ¿RECOMENDARIA EL CURSO A OTRA(S) PERSONA(S)?

SI		NO	

5.- ¿QUE CURSOS LE SERVIRIA QUE PROGRAMARA LA DIVISION DE EDUCACION CONTINUA.?

_____

_____

6.- OTRAS SUGERENCIAS:

_____

_____

7.- ¿EN QUE HORARIO LE SERIA CONVENIENTE SE IMPARTIERAN LOS CURSOS DE LA DIVISION DE EDUCACION CONTINUA? MARQUE EL HORARIO DE SU AGRADO

LUNES A VIERNES DE 16 A 20 HORAS	MARTES Y JUEVES DE 17 A 21 HS SABADO DE 10 A 14 HS.	OTRO
LUNES, MIERCOLES Y VIERNES DE 17 A 21 HORAS	VIERNES DE 17 A 21 HS. SABADOS DE 10 A 14 HS	

COORDINACION CURSOS DE COMPUTO
CENTRO DE INFORMACIÓN Y DOCUMENTACION

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

Análisis, Diseño e Implementación de Bases de Datos Relacionales

Jéssica Briseño C.

Abril de 1995

CONTENIDO

Introducción al ambiente de Bases de Datos

OBJETIVO:

En este capítulo se estudiarán los conceptos básicos de la teoría de Bases de Datos.

En este capítulo el asistente:

- Definirá el término Base de Datos.

- Definirá el término Sistema Manejador de Bases de Datos (*Data Base Management Sistem*, DBMS).

- Conocerá las ventajas de desarrollar sistemas en ambientes de Bases de Datos.

- Identificará a los diferentes tipos de usuarios de una Base de Datos.

- Conocerá los diferentes modelos de Bases de Datos .

INTRODUCCION

En sus orígenes, el término *Base de Datos* sólo era utilizado dentro de los cerrados ambientes de los grandes centros de cómputo. Las Bases de Datos fueron una atractiva alternativa para el desarrollo de sistemas de información más versátiles y eficientes. Actualmente, este concepto ha sido muy difundido gracias a la introducción de computadoras pequeñas y accesibles, y a la aparición de una gran cantidad de software de Bases de Datos; que permiten la distribución de los recursos en los Sistemas de Bases de Datos y el desarrollo de aplicaciones más eficientes y amigables, dando origen a esquemas Cliente/Servidor y de Bases de Datos distribuidas.

Sien embargo, no obstante que la tecnología de Bases de Datos ha avanzado considerablemente, la clave en el éxito de la adoptación de esquemas de Bases de Datos Relacionales, radica en el Diseño de la Base de Datos. Diseñar una Base de Datos no consiste en mapear los antiguos sistemas de archivos a un esquema de tablas, es necesario aprovechar todas las características del Modelo Relacional, así como del DBMS a utilizar.

En este curso se presentará un metodología para el análisis y diseño lógico de la Base de Datos partiendo del análisis de requerimientos del sistema. Por otra parte se presentarán una serie de técnicas para el Diseño Lógico y Físico que permiten la implementación de Bases de Datos de alto performance.

En los últimos temas del curso se tratarán temas realacionados con las nuevas Tecnologías de Bases de Datos, como lo son: Arquitectura Cliente/Servidor, Bases de Datos Distribuidas, CASE y Bases de Datos Orientadas a Objetos.

SISTEMAS DE INFORMACION DE PROCESAMIENTO DE ARCHIVOS

Supongase el siguiente ejemplo:

Una empresa bancaria, mantiene su sistema de ahorros en un sistema de archivos. El sistema tiene diversos programas de aplicación que permiten al usuario manejar los archivos:

- Programa de cargos y abonos a una cuenta.
- Programa de altas, bajas y cambios de cuentas.
- Programa para generar estados mensuales.
- Programa para obtener el saldo de una cuenta.

Estos programas de aplicación los han escrito programadores de sistemas en respuesta a las necesidades de la empresa.

Según surge la necesidad, se agregan nuevos programas de aplicación al sistema. A los programas ya existentes se les agregan nuevos cambios "parches", para cumplir con las necesidades de la empresa. Por ejemplo, suponga que la directiva del banco ha decidido que el saldo mínimo de las cuentas de ahorro será de $500; por lo que será necesario modificar el programa de cargos y abonos y todos aquellos programas que manejen el saldo de la cuenta bancaria.

El típico *Sistema de Infromación de Procesamiento de Archivos* descrito está apoyado por un sistema operativo convencional. Los registros permanentes se almacenan en varios archivos, y se escribe un número de diferentes programas de aplicación para extraer y añadir registros. Este sistema tiene un número de desventajas importantes:

- Redundancia e inconsistencia de los datos
- Dificultad para tener acceso a los datos
- Aislamiento de los datos
- Anomalías del acceso concurrente
- Problemas de seguridad
- Problemas de integridad

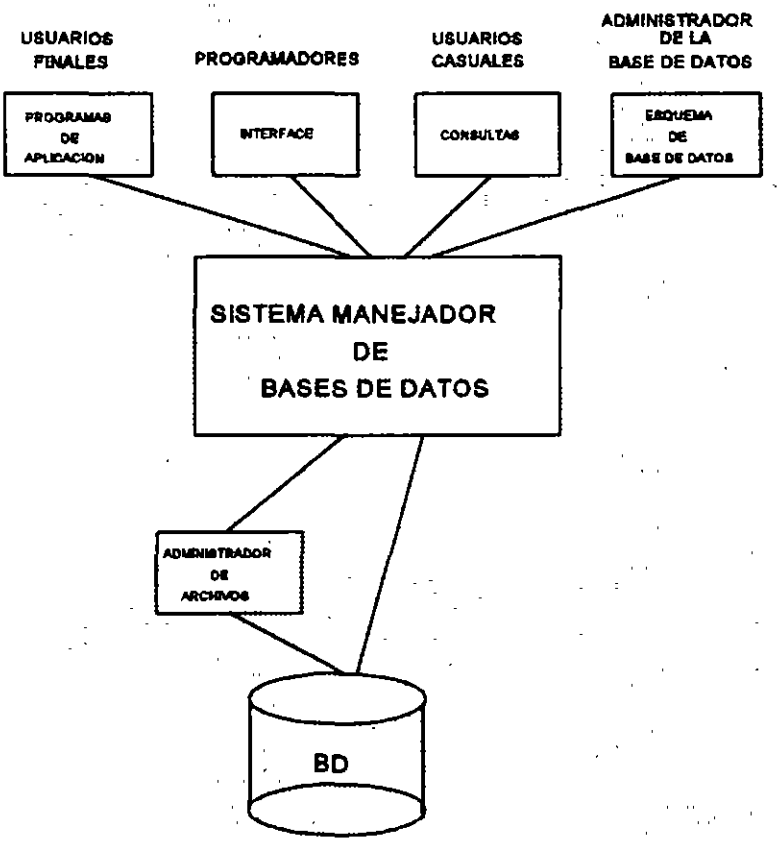
BASE DE DATOS

Conjunto de datos interrelacionados con redundancia controlada para servir a una o más aplicaciones; los datos son independientes de los programas que los usan .

La Base de Datos representa la información de una empresa y su modelo debe de ser tal, que responda a todas las espectativas de información de la misma.

SISTEMA MANEJADOR DE BASES DE DATOS

Conjunto de programas que sirven para administrar, controlar, accesar y manipular una base de datos.



EJEMPLOS:

- SYBASE

- Oracle

- Informix

- Ingres

Generalmente, las base de datos requieren una gran cantidad de almacenamiento, las base de datos de las empresas se miden en terminos de gigabytes o terabytes. Puesto que la memoria principal de las computadoras no puede almacenar esta información, la base de datos se almacena en disco. Es de gran importancia que el sistema de base de datos minimice la necesidad de mover los datos entre la memoria y el disco.

Las funciones que debe cumplir un DBMS son las siguientes:

- Simplificar y facilitar la definición y el acceso a los datos.
- Interactuar con el sistema de archivos.
- Permitir la implantación de esquemas de integridad.
- Permitir la implantación de esquemas de seguridad.
- Esquemas de seguridad y recuperación.
- Control de concurrencia

LENGUAJES

Una de las funciones de un DBMS es la de simplificar la definición y el acceso a los datos. Para definir el esquema de la Base de Datos, así como para explotar la información que se encuentra en ella es necesario contar con un medio de comunicación proporcionado por el DBMS, es decir con un lenguaje. Existen dos tipos de lenguajes básicamente, los cuales en la mayoría de los casos están inmersos en uno:

- Lenguaje de Definición de Datos

- Lenguaje de Manipulación de Datos

El Lenguaje de Manipulación de Datos puede ser de alguno de los siguientes tipos:

- Procedurales: se especifica que datos se necesitan y cómo obtenerlos.

- No Procedurales: se especifica únicamente los datos que serán recuperados y no la forma de obtenerlos.

Los LMD no procedurales normalmente son más sencillos de aprender y de usar que los procedurales. Sin embargo, puesto que el usuario no tiene que especificar como conseguir los datos, estos lenguajes pueden generar código que no sea tan eficiente como el producido por los lenguajes procedurales.

Un lenguaje será más productivo en tanto más No Procedural sea y más versátil para aplicaciones especiales en tanto más Procedural se comporte.

Una *consulta* es una sentencia de DML que solicita la recuperación de datos. Al subconjunto de instrucciones de DML que permiten realizar consultas se le llama *lenguaje de consultas*.

SQL (Structured Query Language)

SQL es un lenguaje tanto para la definición de los datos como para la manipulación de ellos que se ha convertido en estándar en el campo de los DBMS´s.

Características:

- Es un lenguaje estándar reconocido por ANSI e ISO.

- Se encuentra implementado en la mayoría de los DBMS más populares.

- Es un 4GL.

- Es muy fácil de utilizar.

- No incluye referencias físicas de los datos.

- Es utilizado desde muchos programas de aplicación que forman parte de un DBMS.

- Es utilizado para la obtención, modificación y definición de los datos, así como para la Administración de la Base de Datos.

ESQUEMAS

El esquema de la Base de Datos lo constituye el diseño de la misma. Existen varios esquemas en la Base de Datos de acuerdo al nivel que describen, el objetivo es que estos esquemas sean independientes. A la capacidad de modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior se denomina *independencia de datos*.

Los esquemas que existen en una Base de Datos son:

- Esquema lógico

  En este esquema se describen cuáles son los datos reales que están almacenados en la Base de Datos y que relaciones existen entre ellos. Un mismo modelo tendrá diferentes esquemas físicos al ser implantado en diferentes DBMS.

- Esquema físico

  En este esquema se define la forma en como se almacena el modelo lógico de la Base de Datos.

SISTEMAS DE BASES DE DATOS

Los Manejadores de Bases de Datos nos ayudan a implementar sistemas de información bajo la perspectiva de Bases de Datos, proporcionandonos sistemas con las siguientes características:

- Eficiente control de los datos.
- Interacción con el sistema operativo para efectos del almacenamiento, recuperación y actualización de los datos en la base de datos.
- Implantación de la integridad.
- Seguridad.
- Esquemas de Respaldo y recuperación.
- Control de concurrencia.
- Herramientas para la explotación de los datos.
- Fácilidad para explotar la Base de datos.
- Performance.

USUARIOS DE LA BASE DE DATOS

Existen diferentes puntos de vista y aplicaciones que los usuarios llevan a cabo sobre una Base de Datos, es por ello que el Manejador de Base de Datos debe contar con los mecanismos necesarios que esten de acuerdo con el tipo de usuario involucrado. Podemos definir los siguientes tipos de usuarios:

- Usuarios finales

- Usuarios casuales

- Programadores de aplicaciones

- Programadores especializados

- El Administrador de la Base de Datos

EL ADMINISTRADOR DE LA BASE DE DATOS

Una de las razones por las que se utilizan sistemas en Bases de Datos es el tener un control centralizado de la información que se maneja y de los programas que la accesan. El Administrador de la Base de Datos o DBA (*Data Base Administrator*) es un usuario especial que tiene como función controlar la Base de Datos y la forma en como esta es accesada.

Las funciones principales del DBA son:

- Definición del esquema de la Base de Datos

- Definición de las estructuras de almacenamiento y de los métdos de acceso

- Modificación del esquema y de la organización física

- Autorización para acceso a los datos

- Especificación de restricciones de integridad

- Especificación de politicas para con la Base de Datos

MODELOS DE BASES DE DATOS

Una Base de Datos se compone esencialmente de datos, además existen mecanismos que permiten tener un acceso rápido y eficiente a los mismos; pero ¿como definimos el modelo que representa a nuestra Base de Datos?

Para describir la estructura de una Base de Datos es necesario definir el concepto de Modelo de Base de Datos.

Un Modelo de Bases de Datos es un conjunto de herramientas conceptuales que sirven para la descripción de los datos, relaciones entre ellos, semántica asociada y restricciones de consistencia.

Los diversos Modelos de Bases de Datos que se han propuesto se dividen dos grupos:

- Modelos lógicos basados en objetos
- Modelos lógicos basados en registros

MODELOS LOGICOS BASADOS EN OBJETOS

Los modelos lógicos basados en objetos se usan para describir datos en los niveles conceptual y de visión. Se caracterizan por el hecho de que proporcionan capacidad de estructuración bastante flexible, permiten especificar restricciones de datos explícitamente y son independientes de la forma en que los datos se almacenan y manipulan.

Algunos de los modelos más extensamente conocidos son:

- El modelo Orientado a Objetos.
- El modelo binario.
- El modelo semántico de datos.
- El modelo infológico.

El Modelo Orientado a Objetos

El Modelo Orientado a Objetos se basa en una colección de objetos, cada uno de los cuales representa un ente abstracto del Sistema de Información a modelar.

El objeto contiene información(atributos) que representa su estado. Por otra parte, los objetos tienen asociado código que opera sobre el objeto (métodos).

Los objetos que tienen los mismos tipos de valores y los mismos métodos se agrupan en clases. Una clase puede ser vista como una definición de tipo para objetos.

La única forma en la que un objeto puede accesar la información de otro objeto, es por medio de un método de ese otro objeto, es decir *enviando mensajes* al objeto. Los métodos constituyen la interfaz a un objeto y son el único medio de modificar la información interna del objeto, su estado.

Cada objeto tiene su propia identidad, independiente de los valores que contiene. Así, dos objetos que contienen los mismos valores son, sin embargo, distintos. La distinción entre los objetos se mantiene en el nivel físico por medio de identificadores de objeto.

MODELOS LOGICOS BASADOS EN REGISTROS

Los modelos lógicos basados en registros se utilizan para describir datos en los niveles conceptual y físico.

Los modelos lógicos basados en registros se llaman así, porque la Base de Datos está estructurada en registros de formato fijo de varios tipos. Cada registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. Esto cosntrasta con los modelos orientados a objetos en los que los objetos pueden estar compuestos por objetos a un nivel de anidamiento de profundidad arbitraria.

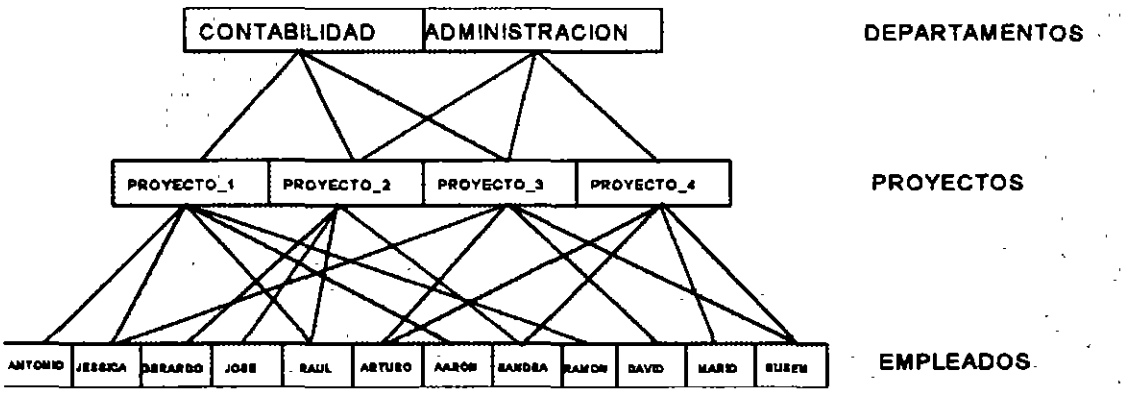
Los modelos lógicos basados en registros, no incluyen un mecanismo para la representación directa de código en la Base de Datos. En cambio, tienen asociados lenguajes que permiten expresar consultas y actualizaciones sobre la Base de Datos.

Dentro de los modelos lógicos basados en registros tenemos los siguientes:

- Modelo de red
- Modelo jerárquico
- Modelo relacional

Modelo de Red

En este modelo, los datos se representan como una colección de registros, la relación entre ellos se da por medio de apuntadores. Los registros se organizan como una colección de gráficas arbitrarias.



Desventajas:

- Los apuntadores o direcciones se deben almacenar junto con los datos

- Para recuperar información se debe "navegar" a través de la gráfica

- No se puede obtener información no planeada antes de modelar

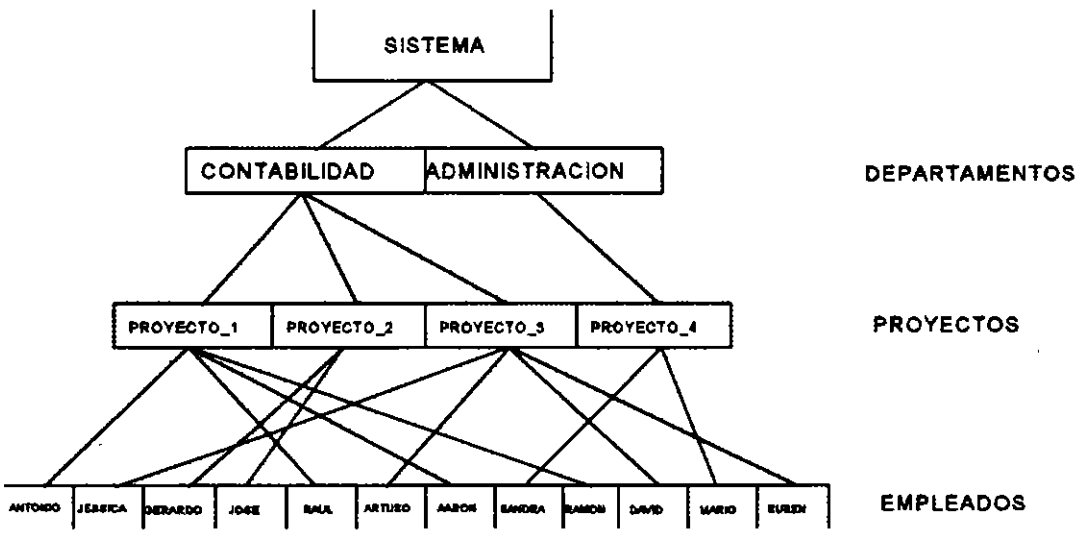
- Modificar la estructura de la Base de datos implica redefinir todo el esquema

Ventajas:

- Acceso rápido a los datos debido a los apuntadores

Modelo Jerárquico

En este modelo, los datos se representan como una colección de registros, mientras que la relación entre ellos se da por medio de ligas o apuntadores. Se diferencia del modelo de red, en que los registros están organizados como colecciones de árboles en vez de gráficas arbitrarias.



Desventajas:

- No puede haber ciclos y sólo puede haber asociaciones 1:N y 1:1

- Los apuntadores o direcciones se deben almacenar junto con los datos

- Para recuperar información se debe recorrer el árbol

- No se puede obtener información no planeada antes de modelar

- Modificar la estructura de la Base de datos implica redefinir todo el esquema

- Se pueden representar asociaciones M:N manteniendo datos duplicados

Ventajas:

- Acceso rápido a los datos debido a los apuntadores

Modelo Relacional

En el modelo relacional, los datos y las relaciones entre los datos se representan por medio de una serie de tablas, cada una de las cuales esta compuesta por columnas con nombres únicos. Una columna de una tabla representa una relación entre un conjunto de valores. Existe una correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual recibe su nombre el modelo relacional.

EMPLEADOS

NO_CTA	NOMBRE	DIR	TEL	NO_DEPTO
18456	Antonio	Revolucion 123	733-22-89	10
18272	Jessica	Norte 86B 98	657-28-92	40
72638	Gerardo	Rio chico 22	512-38-39	10
28289	Jose	Palmas 926	833-32-21	30
29829	Raul	Rosas 83-5	937-00-52	30
87719	Arturo	Churubusco 45	723-45-11	30
32983	Aaron	Taxqueña 372	632-73-21	20
32732	Sandra	Av. Central 13	743-03-01	40
32903	Ramon	Marina Nal. 86	732-37-77	20
95672	David	Almaraz 2-1	664-83-00	10
48568	Mario	Cozumel 11-3	731-82-83	40
84324	Ruben	Fco. Sosa 266	527-73-12	20

DEPARTAMENTOS

NO_DEPTO	NOMBRE
10	Sistemas
20	Finanzas
30	Contabilidad
40	Ingeniería

PROYECTOS

NO_PROY	DESCRIPCION
1	Diseño del Sistema de Inventarios
2	Sistema de Nomina
3	Sistema Integral de Servicios
4	Evaluación de Equipo de Cómputo

ASIGNADO

NO_CTA	NO_PROY
28289	2
95672	3
48568	4
84324	3
84324	4
18456	1
29829	2
32983	1
32732	4
18272	1
29829	1
72638	2
18272	3
87719	3
32732	2
32903	1
87719	4

Ventajas:

- Tiene una base matemática, conocida como Algebra y Cálculo Relacional.

- Se pueden representar fácilmente asociaciones M:N.

- No existen apuntadores u otro tipo de información que no sea la que creó la necesidad de la Base de Datos.

- Las operaciones efectuadas para obtener información se realizan a nivel de la tabla completa y no a nivel de registros.

- No es necesario diseñar el esquema de la Base de Datos de acuerdo a las operaciones o consultas que se van a llevar a cabo. Ees posible obtener información no prevista.

- Se puede modificar la estructura de la Base de Datos sin que esto obligue a un cambio de las aplicaciones.

- La forma de explotar la información es por medio de operaciones relacionales, a través de un lenguajes de cuarta generación.

RESUMEN

- Bases de Datos

Una Base de Datos es un conjunto de datos interrelacionados.

- Sistemas Manejadores de Bases de Datos (DBMS)

Conjunto de programas que sirven para administrar, controlar, accesar y manipular una Base de Datos.

- Modelo Relacional

Representación de la Base de Datos por medio de tablas relacionadas a través de columnas

- Esquema lógico y físico de una Base de Datos

Definición de la Base de datos a nivel conceptual y físico

- Lenguaje de Definición y Lenguaje de Manipulación de Datos

Lenguajes para la definición, modificación y consulta de la Base de Datos, proporcionados por el DBMS.

- SQL

Lenguaje comercial estándar para definición y manipulación de la información en una Base de Datos.

CAPITULO 2

MATERIAL ANEXO

ANALISIS, DISEÑO E IMPLEMENTACION

DE BASES DE DATOS

El Análisis y Diseño de Bases de Datos en el entorno de la Ingeniería de Software

OBJETIVO:

En este capítulo se determinará el papel que juega el Análisis y Diseño de Bases de Datos como disciplinas de la Ingeniería de Software.

En este capítulo el asistente:

- Definirá el concepto de Ingeniería de Software.
- Conocerá el ciclo de vida aplicado a la automatización de procesos.
- Conocerá el ciclo de vida aplicado al modelado de datos.

INGENIERIA DE SOFTWARE

La Ingeniería de Software consiste en el establecimiento y uso de principios y metodologías de ingeniería robustos, orientados a obtener software que sea fiable y funcione eficientemente sobre máquinas reales.

La Ingeniería de Software abarca un conjunto de tres elementos claves: *métodos*, *herramientas* y *procedimientos*, que facilitan controlar el proceso de desarrollo de software y suministran las bases para construir software de alta calidad de una forma productiva.

Los métodos de la Ingeniería de Software suministran el "cómo" construir técnicamente el software, abarcan tareas como:

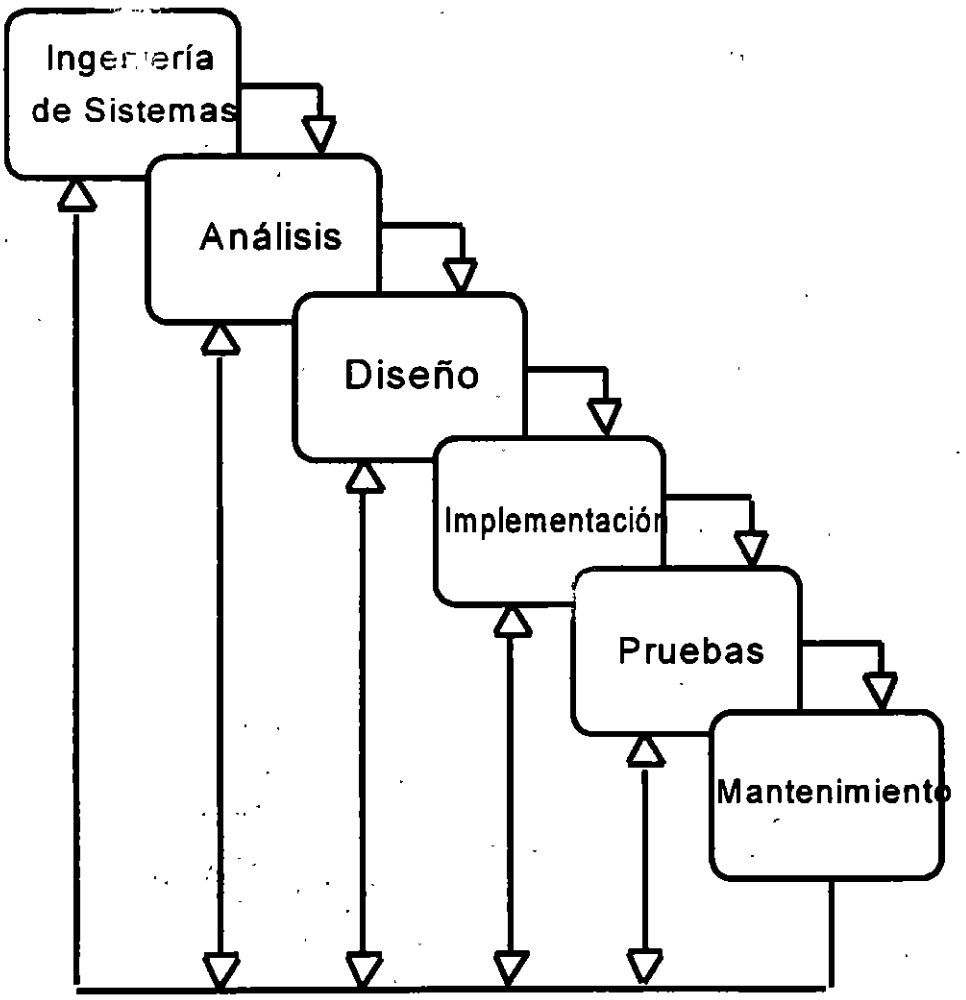
- Planificación y estimación de proyectos.
- Análisis de los requerimientos del sistema y del software.
- Diseño de los datos.
- Arquitectura de programas y procedimientos.
- Codificación.
- Pruebas.
- Mantenimiento.

Las herramientas de la Ingeniería de Software suministran un soporte automático o semiautomático para los métodos (por ejemplo herramientas CASE).

Los procedimientos de la Ingeniería de Software definen la secuencia en la que se aplican los métodos, la secuencia en la generación de documentos, los controles de calidad, y las guías que facilitan el establecer el desarrollo del software.

EL CICLO DE VIDA

El ciclo de vida en la Ingeniería de Software exige un enfoque sistemático, secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, pruebas y mantenimiento.



Las etapas de la Ingenierá de Software se aplican tanto a Procesos, como a Datos:

En el caso de procesos, el ciclo de vida nos conduce a la creación de programas que automatizan los procesos.

Para poder automatizar procesos, antes se debe análisar y diseñar los datos.

La aplicación de las etapas de la Ingeniería de Software a datos conduce a la creación de un modelo de datos, que puede ser por ejemplo un esquema de archivos planos, o bien una Base de Datos. Este modelo será utilizado en el proceso de automatización de procesos.

Cuando los procesos automatizados no se implementan soportados en Bases de Datos, la implementación del diseño de datos se puede hacer conjuntamente; pero va ha existir una dependencia datos/programas.

No todos los sistemas basados en computadora hacen uso de una Base de Datos, para aquellos que lo hacen, la Base de Datos es la guía para todas las funciones del sistema.

El análisis, diseño e implementación de una Base de Datos forman parte del ciclo de vida de la Ingeniería de Software y son actividades que no estan ligadas al desarrollo de alguna aplicación en particular y pueden iniciarse una vez que se ha definido el dominio de la información.

EL CICLO DE VIDA EN LA AUTOMATIZACION DE PROCESOS (DESARROLLO DE SOFTWARE)

Ingeniería y Análisis del Sistema

Debido a que el software es siempre un componente de un sistema mayor, el trabajo comienza estableciendo los requerimientos de todos los elementos del sistema y luego asignando algún subconjunto de estos requerimientos al software.

En esta etapa se fijan las limitantes del software, los alcances, sus funciones, sus interfaces (conexión con otros elementos del sistema global), los datos utilizados y los generados, etc.

Análisis

Para comprender la naturaleza de los programas a construir, el Ingeniero de Software debe comprender el dominio de los procesos a automatizar, las funciones, rendimientos e interfaces requeridas.

En esta etapa se análisan detalladamente cada uno de los procesos involucrados, también se documentan.

Este proceso se realiza en estrecha comunicación con el cliente.

La documentación generada en esta etapa, refleja en forma detallada los requerimientos del sistema.

Diseño

El diseño de software es un proceso que se enfoca a la generación de la arquitectura del software y detalle procedimental.

El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación.

Implementación

En el caso de procesos, esta etapa se conoce comunmente como codificación.

En esta etapa, el diseño debe traducirse en una forma legible para la máquina.

Si el diseño se ejecuta de una manera detallada, la codificación puede realizarse mecánicamente.

Pruebas

Las pruebas se enfocan sobre la lógica interna del software, asegurando que todas las sentencias sean probadas, y sobre las funciones externas, realizando pruebas para asegurar que la entrada definida producirá los resultados que se requieren.

Mantenimiento

El software sufrirá cambios después de que sea liberado.

Los cambios pueden ser debido a nuevas adaptaciones (Mantenimiento adaptativo) o a que se han encontrado errores (Mantenimiento correctivo).

EL CICLO DE VIDA EN LA GENERACION DE UN MODELO DE DATOS (DESARROLLO DE BASES DE DATOS)

Ingeniería y Análisis del Sistema

En esta etapa se determina la información que maneja el sistema, las características de esta información.

Se determina también el alcance del modelo que se desea generar y sus limitantes.

En esta atapa se deben reflejar las reglas del negocio como requerimientos del modelo de datos. Hay que puntualizar que cada empresa tiene reglas y requerimientos diferentes.

Por ejemplo, si se desea desarrollar un modelo de datos para un banco, se debe determinar la información del cliente que se utiliza, los tipos de cuentas que maneja el banco, restricciones en cuanto a movimientos, reglas de integridad, etc.

Los requerimientos necesarios se obtienen en base a:

- Entrevistas con el cliente
- Cuestionarios
- Lectura de manuales de procedimientos, reglamentos, reportes, etc.
- Prototipos

Los requerimientos para el modelo de datos deben reflejar el sistema de información de la empresa, así como también futuros cambios.

El documento de requerimientos generado es la base y guía del análisis y diseño de la Base de Datos.

Un ejemplo de un documento sencillo de requerimientos podría ser el siguiente:

El video club "**Patito S.A. de C.V**" usa actualmente un sistema de control de rentas de películas en base a los siguientes archivos:

SOCIOS	**RENTAS**
Nombre socio	Nombre Socio
Dirección	Nombre de película
Teléfono	Tipo
No. tarjeta	Formato
	Fecha de renta
	Fecha de devolución
	Importe

La utilización de estos archivos ha resultado ser impractica y costosa ya que se tiene gran redundancia, mucha inconsistencia y un alto costo de mantenimiento. De esta manera, el dueño desea un nuevo sistema en Bases de Datos ya que quiere eliminar estos problemas

Se deben considerar los siguientes requerimientos:

1. Se desea tener una clave de socio que cuente con la siguiente información: dirección, teléfono, numero de tarjeta de crédito, el nombre de dos titulares indicando cual es el principal.

2. Se tiene un lote de 2000 películas diferentes cada uno de las cuales tiene de 5 a 20 copias, para las cuales se desea saber su nombre, su genero (comedia, terror, infantil, suspenso, etc.), clasificación (A, B, C, XXX) y el director.

3. El precio que se debe pagar por la renta diaria de una película depende de la categoría de esta, se tienen 5 categorías diferentes (línea dorada, etiqueta roja,. etc.).

4. El dueño desea saber en un momento dado cuantas copias de una película se han rentado, cuales son estas y de que formato son (beta o VHS). Por otra parte desea saber la fecha a partir de la cual se puso a disposición de los socios una copia, esto con el fin de dar el mejor servicio y poder determinar cuales son las copias que ya se necesitan reemplazar.

5. Un socio puede rentar hasta tres películas; si no devuelve una película, se le aplica una multa por cada día que transcurra después de la fecha de entrega igual al 20% del precio por renta diaria de la película en cuestión.

Análisis

Esta etapa es una etapa de retroalimentación con el cliente, en la que seguramente se proponen y cambian requerimientos.

En esta etapa se analizan los requerimientos y se obtiene una descripción del sistema de información a modelar.

Diseño

El diseño de Bases de Datos incluye el Diseño Lógico de Bases de Datos y el Diseño Físico de la Base de Datos.

En el Diseño Lógico se determina de acuerdo a un modelo teórico las relaciones y atributos de los datos.

En el Diseño Físico se determina la forma en como se implementará la Base de Datos en base al DBMS utilizado (indices, dispositivos, esquemas de protecciones, etc.).

Implementación

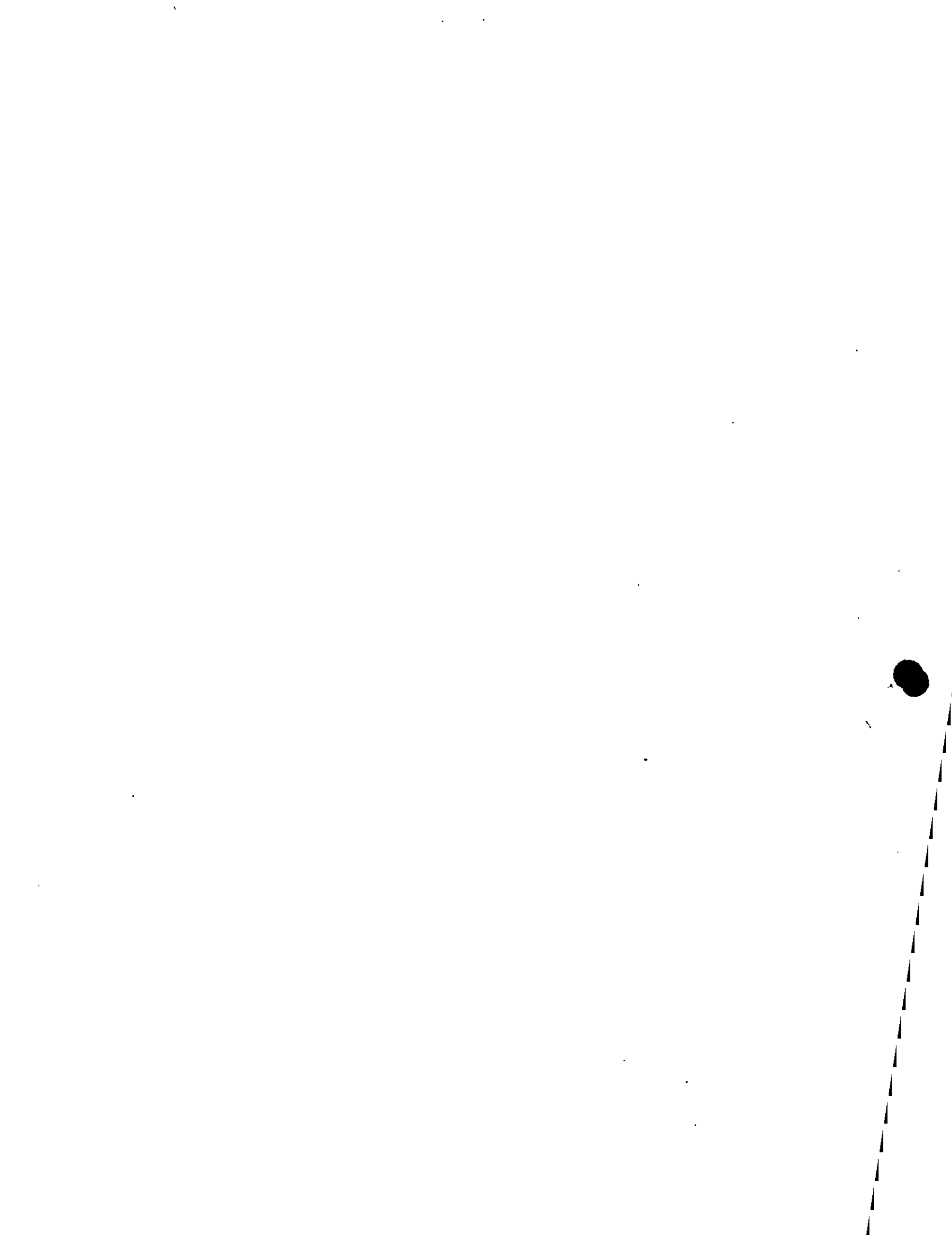
En esta etapa se genera la Base de Datos de acuerdo al Diseño Físico.

CAPITULO 3

MATERIAL ANEXO

ANALISIS, DISEÑO E IMPLEMENTACION

DE BASES DE DATOS



Análisis de Bases de Datos Relacionales

OBJETIVO:

En este capítulo se estudiará la metodología utilizada para el análisis de Bases de Datos relacionales.

En este capítulo el asistente:

- Aprenderá a identificar las entidades a partir de los requerimientos de la empresa.

- Aprenderá a identificar las relaciones existentes entre entidades.

- Determinará los atributos de las entidades y relaciones.

- Identificará superentidades y subentidades.

- Identificará relaciones recursivas y N-arias.

- Conocerá los DEA como herramienta gráfica para representar un modelo de Bases de Datos.

El análisis de Bases de Datos Relacionales está basado en la percepción de un mundo real que se compone de un conjunto de objetos básicos llamados **Entidades**, y de **Asociaciones** entre esos objetos.

Ejemplo:

Cuando una Empresa maneja PRODUCTOS que son ENTREGADOS por un DISTRIBUIDOR, cada uno de esos productores o distribuidores es una Entidad, y la entrega de un PRODUCTO por un DISTRIBUIDOR representa una Asociación.

El conjunto de todos los productos que maneja la empresa es entonces un "conjunto de Entidades", y todas las Asociaciones forman un "conjunto de Asociaciones".

Sin embargo, por simplicidad, se denomina Entidad a la representación del conjunto de Entidades y Asociación a la representación del conjunto de Asociaciones.

El procedimiento de análisis es el siguiente:

1. Seleccionar las Entidades (como cliente, parte, o proveedor) y el tipo de entidad.

2. Seleccionar las Asociaciones entre las entidades (como orden de partes por clientes, partes proporcionadas por el proveedor, etc.) que están dentro del Alcance de Integración de la Empresa.

3. Determinar el tipo de relación entre las entidades.

4. Determinar el grado de asociación entre las entidades.

5. Asignar atributos a esas entidades y Asociaciones.

ENTIDADES

Es un objeto que existe y que es distinguible de otros objetos.

Es algo (persona, lugar, objeto, concepto) a lo que la Empresa le reconoce poder existir en forma independiente y que puede ser definido en forma única.

Ejemplo: película, cliente, empleado, departamento, máquina, etc.

Una instancia de una entidad es un elemento de ese tipo de entidad, por ejemplo: "Los gritos del silencio" (película), "Sistemas" (departamento), etc.

Al conjunto de todas las instancias de una entidad se le denomina "conjunto de Entidades" o "tipo de entidad".

Por simplicidad se denomina entidad al "conjunto de entidades".

Las entidades representan tablas en el diseño relacional.

¿Como obtener las entidades?

Las entidades se obtienen de un análisis de los requerimientos de la empresa.

Generalmente las entidades son sustantivos; pero no todos los sustantivos representan entidades.

Se deberán ignorar los sustantivos que:

- No tengan importancia para la empresa.

- Aquellos que denoten documentos que contienen información proveniente de otras entidades, por ejemplo: recibo de nomina, credencial de socio, etc.

Determinación de la llave primaria

Por definición, toda ocurrencia de una Entidad debe ser identificable en forma única. Es por lo anterior que se debe encontrar un identificador de la Entidad o llave.

Existen varios tipos de llaves:

Súper llave. Es un conjunto de uno o más atributos que, tomados en conjunto, permiten identificar en forma única una Entidad dentro del conjunto de Entidades.

Llave candidato. El concepto de Súper llave no es suficiente, puesto que puede contener atributos extraños. Las Súper llaves con el menor conjunto de atributos se conocen como llaves candidato y es posible que existan diferentes conjuntos de atributos que puedan servir como llaves candidato.

Llave primaria. Una llave primaria es una llave candidato que ha sido seleccionada por el diseñador de la base de datos como el medio de identificar Entidades.

Las características necesarias para una llave primaria son las siguientes:

- Unica
- Conocible en cualquier tiempo

Las características deseables de una llave primaria son las siguientes:

- Estable
- No descriptiva
- Pequeña y simple

Documentación de las entidades

Una vez que se han determinado las entidades, es necesario asignarles un nombre, para lo cual se pueden tomar en cuenta los siguientes puntos:

- Utilizar sustantivos en singular (EMPLEADO, DEPARTAMENTO, PELICULA, etc.).

- Eliminar homónimos.

- Describir el significado de la entidad en un Diccionario, utilizando ejemplos y contra ejemplos.

Ejemplo:

EMPLEADO

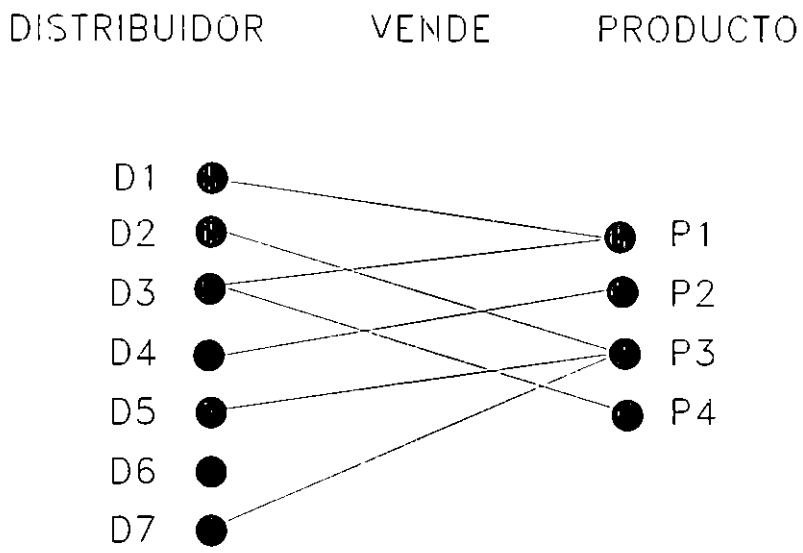
Un empleado es para la empresa X toda persona que tiene una relación laboral mediante un contrato suscrito mediante la Ley Laboral del Trabajo, por ejemplo: empleado de confianza, de medio tiempo, jubilados. No se consideran empleados a aquellas personas que presten sus servicios bajo pago por honorarios o bien empleados temporales o contratados por otra empresa.

Asociaciones

Es el vínculo que existe entre dos o más Entidades.

Por ejemplo, la Entidad departamento puede asociarse con la Entidad empleado vía la Asociación emplea.

Ejemplo:



Una instancia de una asociación es un elemento de esa relación entre entidades, por ejemplo: Juan Pérez TRABAJA Sistemas.

¿Como obtener las asociaciones?

Al conjunto de todas las instancias de una asociación se le denomina "conjunto de Asociaciones" o "tipo de asociación".

Por simplicidad se denomina asociación o relación al "conjunto de asociaciones".

Las asociaciones o relaciones se obtienen de un análisis de los requerimientos de la empresa.

Generalmente las entidades son verbos; pero no todos los verbos representan relaciones.

Se deberán ignorar los verbos que:

- No tengan importancia para la empresa.
- Aquellos que denoten procesos, por ejemplo, "generar nomina".

Grado de asociación

El grado de asociación o cardinalidad, representa en forma cualitativa, el número de ocurrencias de una entidad con las que puede estar asociada una instancia de otra entidad.

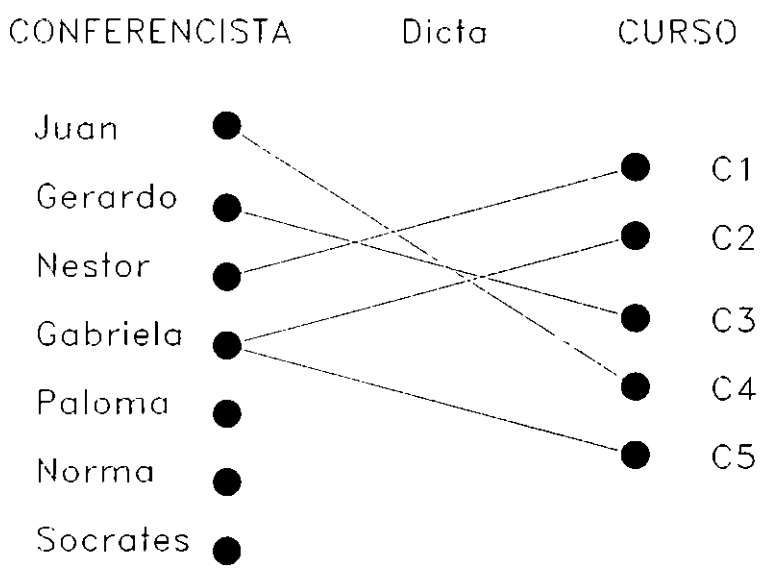
Una asociación puede tener tres tipos de grados:

TIPO		PUEDE INCLUIR				
1:1	(uno a uno)	1:0	0:1	1:1		
1:N	(uno a muchos)	1:0	0:1	1:1	1:N	
M:N	(muchos a muchos)	1:0	0:1	1:1	1:N	N:1M:N

El grado de asociación puede ser obtenido de las reglas de empresa.

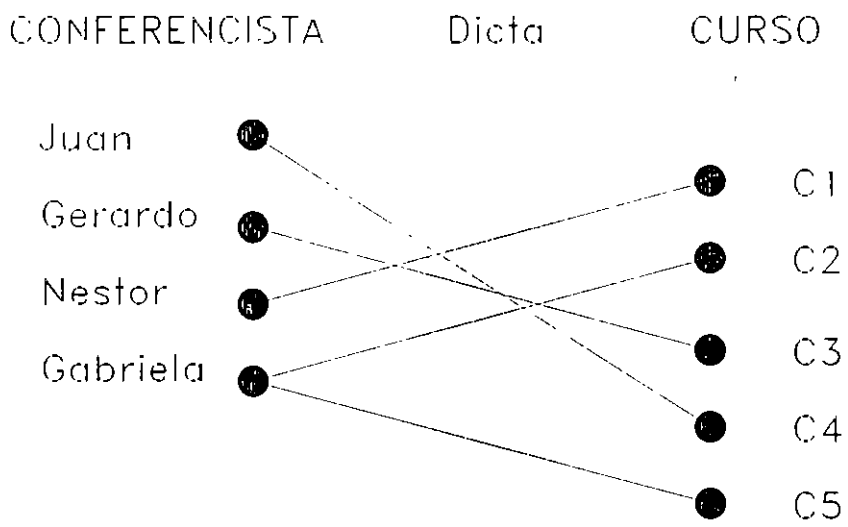
Las reglas de empresa son definiciones que se obtienen del análisis de datos. Por ejemplo:

"Un conferencista puede dictar muchos cursos"
"Un curso sólo puede ser dictado por un conferencista"



Los grados que incluye una categoría de asociación pueden ser reducidos con reglas de empresa más estrictas, como:

"Un conferencista debe ofrecer varios cursos"
"Un curso debe ser ofrecido por un conferencista"



Cardinalidad mínima

Las reglas de empresa que excluyen Asociaciones 1:0 se traducen como la obligatoriedad de que una entidad participe en una Asociación con otra Entidad.

Una Entidad con obligatoriedad se puede considerar como una Entidad débil puesto que para existir depende de la existencia de otra Entidad.

Documentación de las relaciones

Una vez que se han determinado las entidades y relaciones entre ellas, es necesario asignar un nombre a las relaciones, para lo cual se pueden tomar en cuenta los siguientes puntos:

- Utilizar verbos en voz activa (TRABAJA, PERTENECE, RENTA, CURSA, etc.).

- Eliminar homónimos.

- Utilizar sinónimos cuando sea necesario.

- Describir el significado de la relación entre las entidades participantes en un Diccionario, utilizando ejemplos y contra ejemplos.

Atributos

Es una propiedad de una Entidad.

Por ejemplo, número, nombre y RFC pueden ser atributos de la Entidad cliente.

Los atributos generalmente describen a una entidad.

Hay que decidir el tipo asociado a los atributos, por ejemplo: entero, real, carácter, etc.

Hay que determinar cuales son atributos indirectos en una relación, por ejemplo, en la relación EMPLEADO TRABAJA DEPARTAMENTO, el atributo número de sucursal es un atributo indirecto de EMPLEADO en la relación.

Los atributos derivables se deben eliminar en esta etapa. Por ejemplo el salario neto de los empleados obtenido de la suma del salario base más comisiones más prestaciones, no se debe considerar como atributo directo.

En el Diccionario hay que documentar los atributos de cada entidad, indicando nombre, tipo y descripción.

Diagramas de Entidades y Asociaciones

El análisis de Entidades y Asociaciones cuenta con una herramienta gráfica para cumplir sus objetivos.

El proceso se realiza dibujando diagramas conocidos como Diagramas de Entidades y Asociaciones (DEA).

Las convenciones al dibujar DEA son:

1. Las Entidades serán representadas por rectángulos.

2. Las Asociaciones serán rombos.

3. Las líneas de conexión mostrarán qué Entidades son vinculadas por cuál Asociación.

4. Los atributos de las Entidades y las Asociaciones se muestran como círculos o elipses conectados al rombo o rectángulo correspondiente. Generalmente no se dibujan.

5. El grado de la Asociación será representado por 1, M ó N, sobre las líneas de conexión.

6. La obligatoriedad de una entidad débil se indicará terminando la correspondiente línea de conexión dentro de un pequeño rectángulo que forme parte de la Entidad.

Generalización y especialización

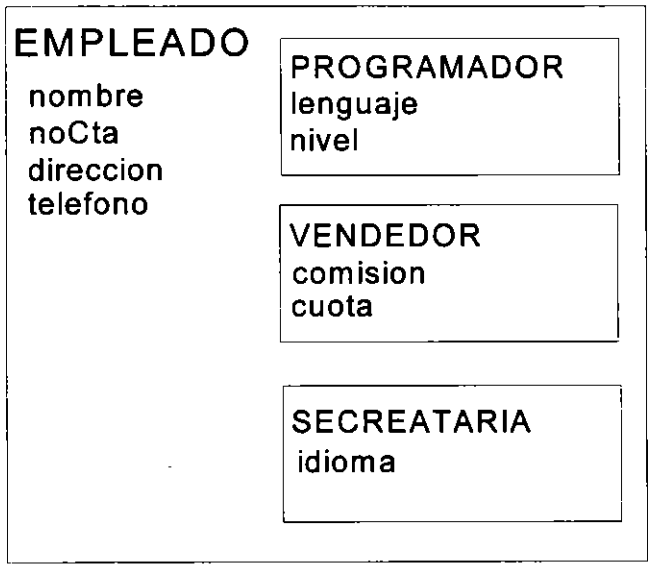
La generalización es el proceso que permite reunir a un conjunto de entidades con atributos comunes en una entidad de nivel superior llamada superentidad.

A cada una de las entidades agrupadas se les conoce como subentidades.

La especialización consiste en generar entidades con atributos particulares a partir de una entidad. En este caso la entidad de la cual parte la especialización también es llamada superentidad, y las entidades generadas subentidades.

La llave primaria, atributos y relaciones de una superentidad también lo son de las subentidades. Lo contrario no aplica.

Ejemplo:


EMPLEADO
nombre
noCta
direccion
telefono
PROGRAMADOR
lenguaje
nivel
VENDEDOR
comision
cuota
SECREATARIA
idioma

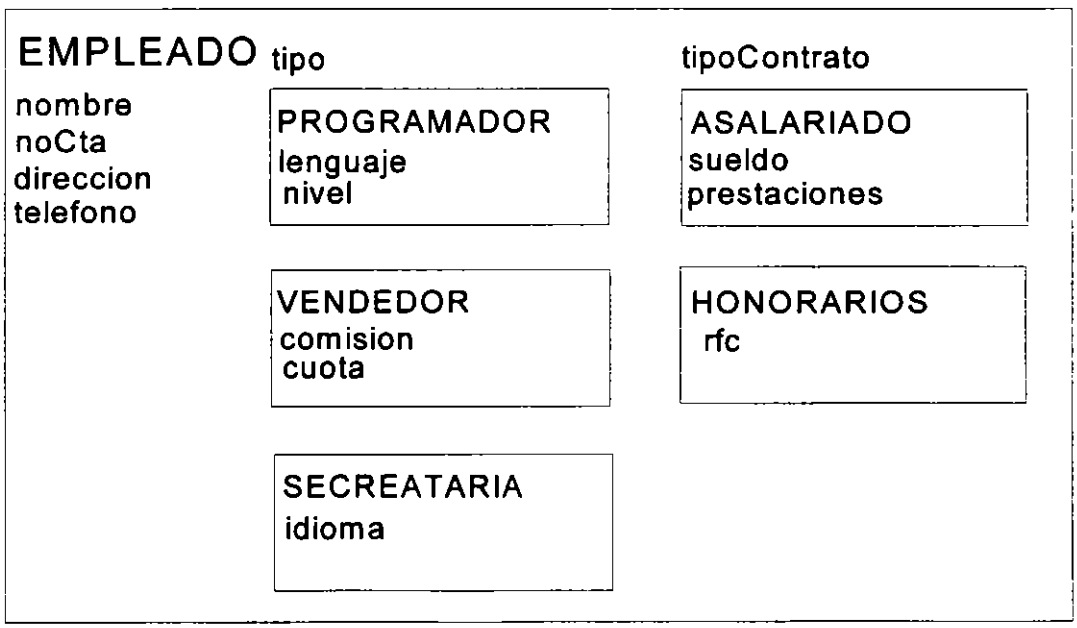
Clases

Una clase es un conjunto de subentidades mutuamente excluyentes.

Cada clase tiene asociado un atributo de clasificación en la superentidad.

Una superentidad puede tener n clases.

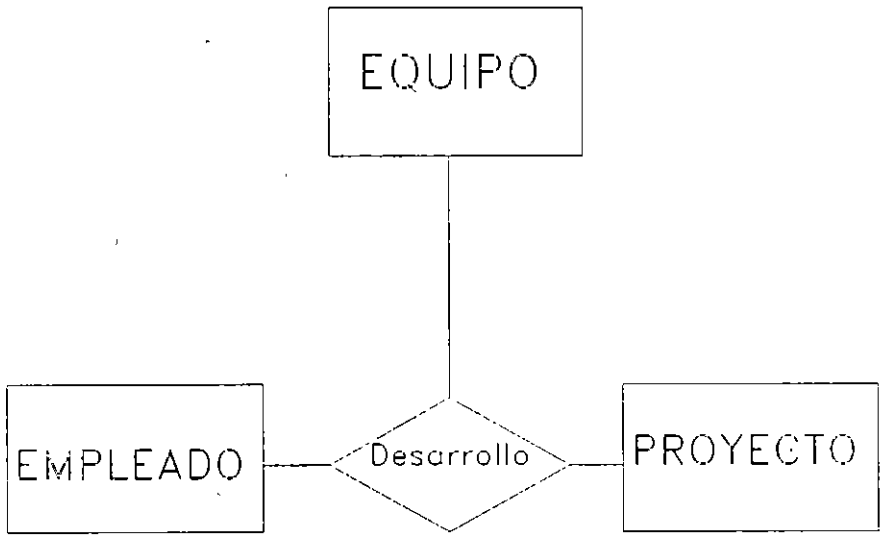
Ejemplo:



Relaciones N-arias

Las relaciones N-arias, son aquellas que existen entre más de dos Entidades.

El Modelo de Entidades y Asociaciones representa estos casos relacionando las Entidades involucradas con un mismo rombo, que representa la relación.



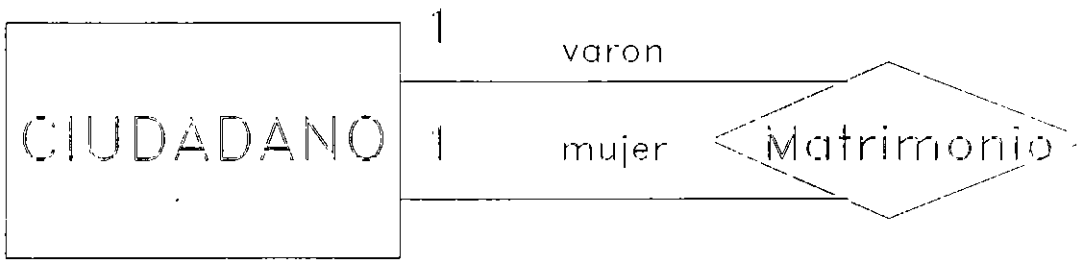
En las relaciones N-arias no es de mucha utilidad especificar el grado de asociación y la cardinalidad mínima.

Relaciones recursivas

Son las que se dan entre Entidades del mismo conjunto de Entidades.

Un ejemplo lo podemos ver considerando al conjunto de Entidades CIUDADANO y la Asociación MATRIMONIO, que debe ser entre dos CIUDADANOs.

En estos casos el modelo se realiza exactamente igual a los casos no-recursivos, pero se debe indicar el papel que juega cada Entidad en la relación.



Contenido del Diccionario

Para cada entidad:

- Nombre
- Descripción
- Llave primaria
- En el caso de subentidades:

  Superentidad
  Clase
  Atributo de clasificación

- Número aproximado de instancias

Para cada relación:

- Nombre (entidad - relación - entidad)
- Descripción
- Grado de asociación
- Cardinalidad mínima
- En el caso de relaciones recursivas, rol de cada entidad

Para cada atributo (entidad o relación)

- Nombre
- Descripción
- Tipo de dato
- Si es atributo derivable, indicar fórmula

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

MATERIAL DIDACTICO

ANALISIS, DISEÑO E IMPLEMENTACION

DE SISTEMAS DE BASES DE DATOS

M A Y O

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg Cuauhtemoc 06000 Mexico, D F APDO. Postal M-2285
Telefonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

Diseño de Bases de Datos Relacionales

OBJETIVO:

En este capítulo se estudiaran las técnicas para la obtención de tablas a partir del modelado representado por medio de un Diagrama de Entidades y Relaciones.

En este capítulo el asistente:

- Aprenderá a obtener las tablas que componen la Base de Datos a partir de entidades.

- Aprenderá a diseñar las relaciones entre tablas por medio de llaves foráneas o tablas de relación.

- Enetenderá el concepto de redundancia.

- Comprenderá el proceso de normalización de un diseño de Bases de Datos.

Llave primaria

Una llave primaria es una llave candidato que ha sido seleccionada por el diseñador de la base de datos como el medio de identificar Entidades.

Las características necesarias para una llave primaria son las siguientes:

- Unica
- Conocible en cualquier tiempo

Las características deseables de una llave primaria son las siguientes:

- Estable
- No descriptiva
- Pequeña y simple

Llave foránea

Una llave foránea en una tabla son las columnas que conforman la llave primaria de otra tabla.

Es conveniente que la llave foránea tenga el mismo nombre y tipo de la llave primaria de la que proviene.

La llave primaria de una tabla puede estar formada en parte por una llave foránea.

Obtención de tablas

La primera etapa de Diseño es la obtención de esqueletos de las tablas que componen el modelo de Bases de Datos.

El esqueleto de una tabla se compone de: el nombre de la tabla, que usualmente es el nombre de la Entidad o Asociación; una lista de atributos mínimos que debe contener esa tabla, que por lo regular son una llave candidato y las llaves foráneas necesarias para mantener el vínculo con otras tablas; y grupos de tres puntos, que indican la futura presencia de otros atributos de la tabla.

La llave candidato se coloca al principio de la lista y se subraya para indicar su calidad de identificador de la Entidad.

La segunda etapa es la asignación del resto de atributos, colocándolos en la tabla que les corresponde, y cumpliendo siempre con las reglas de normalización.

Es un hecho que el conjunto de tablas resultantes puede ser implantado directamente dentro del ambiente de un manejador de bases de datos relacionales, puesto que cada tabla será una relación bien normalizada.

Representación de Entidades

Cada entidad independiente representa una tabla.

La llave primaria de tablas independientes no contiene llaves foráneas.

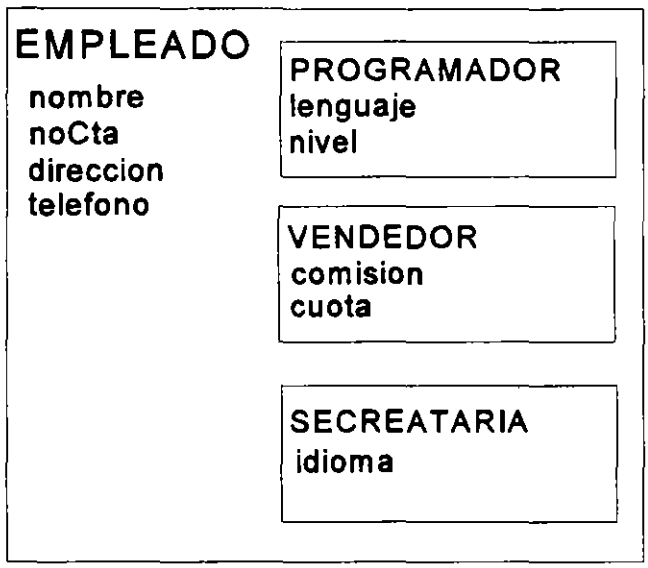
Representación de superentidades y subenetidades

La llave primaria, atributos y relaciones de una superentidad también lo son de las subentidades indirectamente. Lo contrario no aplica.

La superentidad es una tabla que contiene la llave primaria y atributos comunes.

Cada una de las subentidades son entidades dependientes que se representan mediante una tabla que contiene la llave primaria de la superentidad y los atributos propios de la subentidad:

Ejemplo:



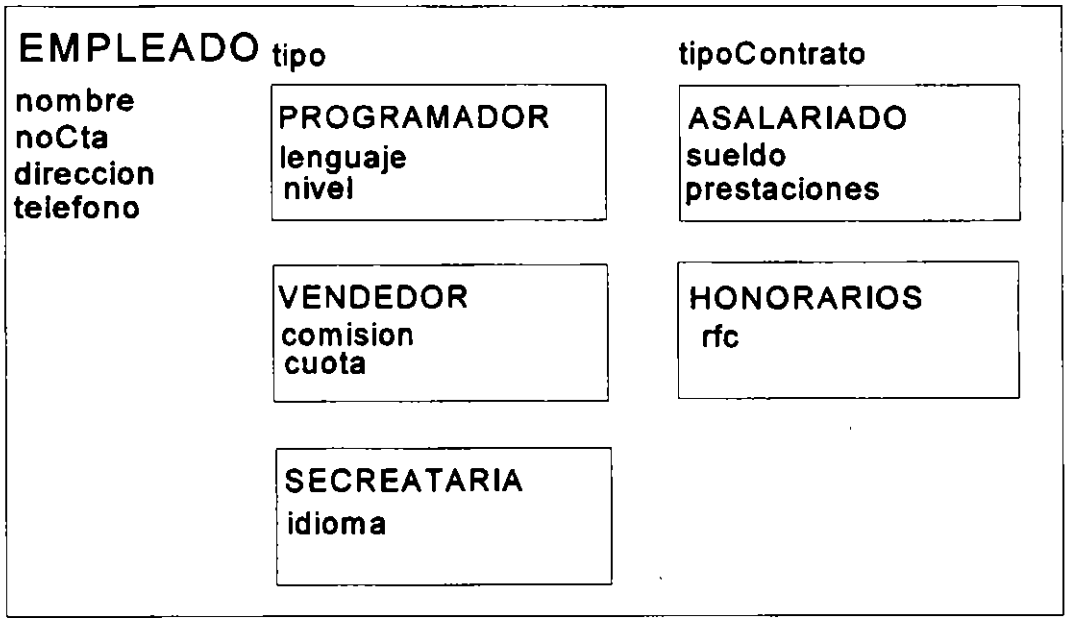
EMPLEADO(numCta, nombre, direccion, edad, ...)
VENDEDOR(numCta, comision, pasajes, ...)
PROGRAMADOR(numCta, nivel, lenguaje, ...)
SECRETARIA(numCta, tipo, idioma, ...)

Clases

Una clase es un conjunto de subentidades mutuamente excluyentes.

Cada atributo de clasificación es un atributo de la superentidad.

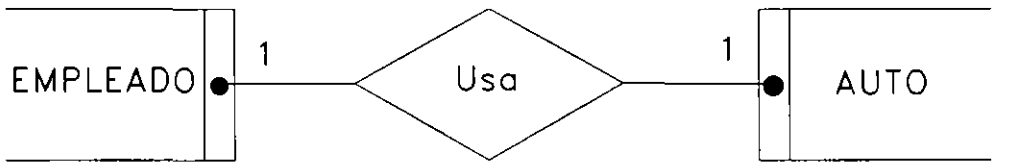
Ejemplo:



EMPLEADO(numCta, nombre, direccion, edad, tipo, contrato, ...)
VENDEDOR(numCta, comision, pasajes, ...)
PROGRAMADOR(numCta, nivel, lenguaje, ...)
SECRETARIA(numCta, tipo, idioma, ...)
ASALARIADO(numCta, salario, prestaciones, ...)
TEMPORAL(numCta, pagoHora, ...)
HONORARIOS(numCta, rfc, ...)

Representación de Realaciones 1:1

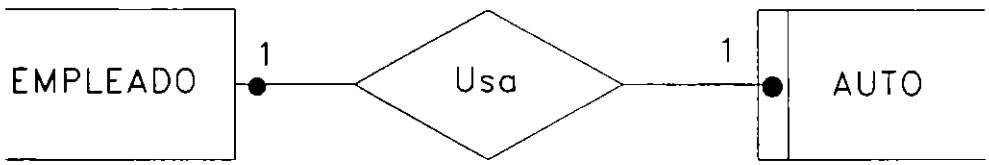
Obligatoriedad para ambas Entidades:



EMPLEADO(#empleado, ... ,#auto, ...)

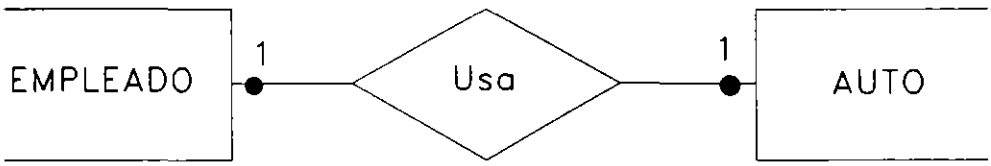
#empleado y #auto son las llaves primarias de cada Entidad.

Obligatoriedad sólo para una Entidad:



EMPLEADO(#empleado, ...)
AUTO(#auto, ... , #empleado)

No obligatoriedad para las dos Entidades:

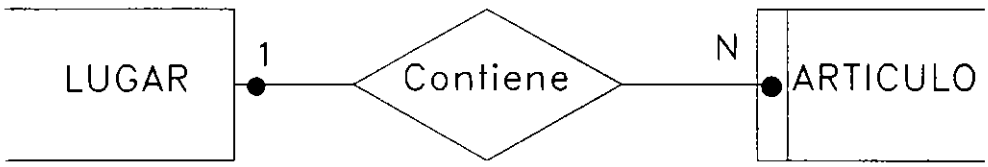


Se crea una tabla para la Relación:

EMPLEADO(#empleado, ...)
AUTO(#auto, ...)
USA(#empleado,#auto, ...)

Representación de Realaciones 1:N

Obligatoriedad en la Entidad de grado N:



LUGAR(nombre, ...)
ARTICULO(#artículo, ..., nombre)

Nótese que la llave primaria de la Entidad de grado 1 se convierte en llave foránea de la tabla que representa la Entidad de grado N, **en ese orden**. De otra forma la llave de ARTICULO podría tomar valores nulos.

Sin obligatoriedad en la Entidad de grado N:

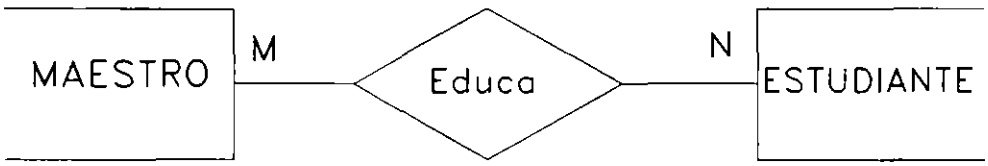
LUGAR(nombre, ...)
ARTICULO(#artículo, ...)
CONTIENE(#artículo, nombre, ...)

La llave primaria de la Entidad de grado 1 se convierte en la llave foránea de la Tabla que representa a la Asociación, **en ese orden.**

Los casos en que existe o no obligatoriedad en la Entidad de grado 1 no modifican nada.

Representación de Realaciones M:N

No importa la existencia o no de obligatoriedad en las Entidades:

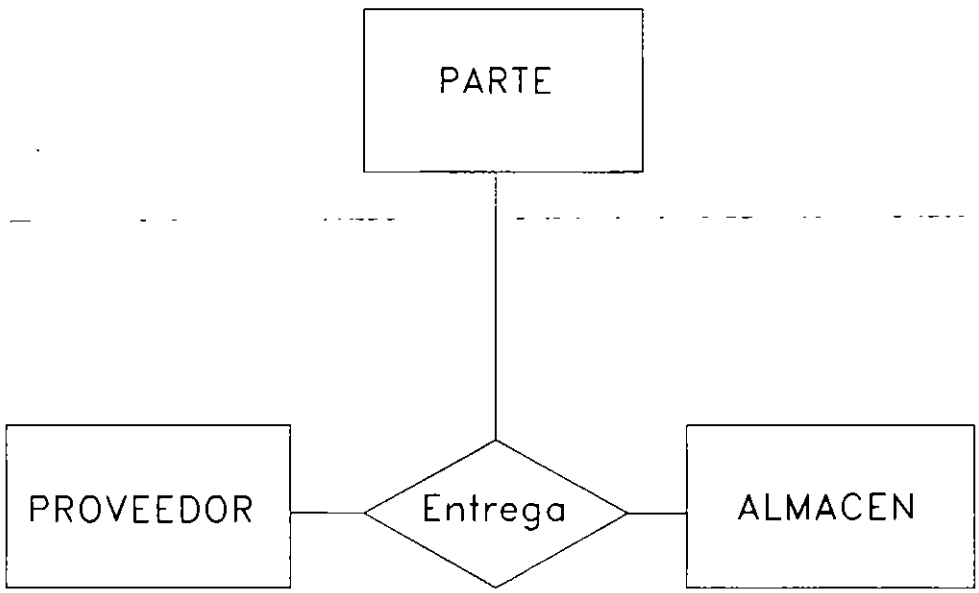


MAESTRO(nombre, ...)
ESTUDIANTE(#estudiante, ...)
EDUCA(nombre, #estudiante, ...)

La tabla que representa la Asociación debe tener como llave primaria la concatenación de las llaves de las Entidades.

Representación de Realaciones N-arias

No importan mucho los grados de asociación o de pertenencia, de cualquier forma se debe representar la Asociación y formar una llave primaria con la concatenación de los identificadores de las Entidades que participan.



PARTE(#parte, ...)
ALMACEN(#almacén, ...)
PROVEEDOR(#proveedor, ...)
ENTREGA(#proveedor,#parte,#almacén, ...)

Normalización

Normalización es una técnica desarrollada para asegurar que las estructuras de datos sean eficientes. Los beneficios de la Normalización son:

- Libera de dependencias indeseables de inserción, borrado y actualización.
- Minimiza la reestructuración de datos cuando se introduce algo nuevo. Se mejora la independencia de datos, permitiendo que las extensiones a la base de datos tengan poco o ningún efecto sobre los programas o aplicaciones que tienen acceso a ella.
- No se introducen restricciones artificiales a las estructuras de datos.

Aunque se han definido más estados de normalización, sólo se han aceptado ampliamente tres. Estos se conocen como **primera, segunda, y tercera formas normales**, o 1NF, 2NF y 3NF respectivamente.

El normalizar es un proceso ascendente, en el que se parte de un universo de relaciones y atributos, y se avanza de forma en forma hasta llegar a la tercera forma normal.

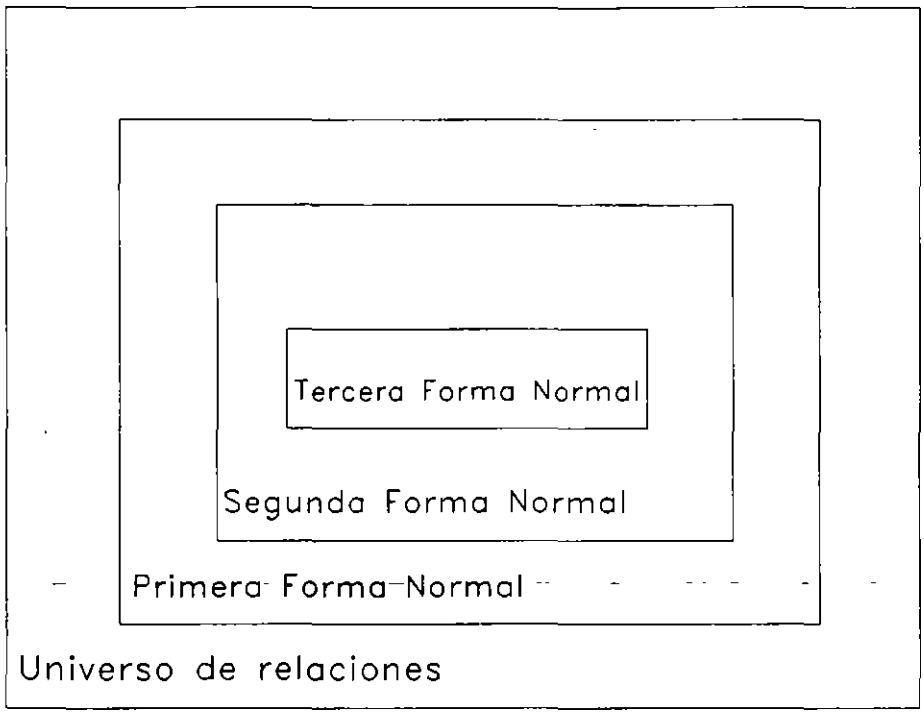


Figura B.1 El universo de las relaciones.

Las etapas de normalización se muestran adelante con un ejemplo, dada la relación no normalizada:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, #producto, descripción_producto, cantidad_producto, precio_total_producto, precio_total_orden)

Primera Forma Normal

Un registro en primera forma normal no incluye grupos repetidos. Es decir cada uno de sus campos debe tener un solo valor.

En la relación ORDEN, se observa que para una misma orden habrá varios productos, por lo que #producto y otros atributos serán grupos repetidos. En 1NF habría que separar:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, precio_total_orden)

PRODUCTO_ORDENADO (#orden, #producto, descripción_producto, precio_producto, cantidad_producto, precio_total_producto)

Segunda Forma Normal

Cada atributo depende de la totalidad de la llave, y no de sólo de una parte de ella.

Se puede observar en PRODUCTO_ORDENADO que descripción_producto depende sólo de #producto, y no tiene que ver con #orden. En 2NF quedaría:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, precio_total_orden)

PRODUCTO (#producto, descripción_producto, precio_producto)

PRODUCTO_ORDENADO (#orden, #producto, cantidad_producto, precio_total_producto)

Tercera Forma Normal

Todos los atributos dependen solamente de la llave y no de otros atributos no llave.

En la relación ORDEN se presenta este problema:

nombre_proveedor y direccion_proveedor dependen de #proveedor

De modo que las tablas en 3NF quedarían como:

ORDEN (#orden, fecha, #proveedor, precio_total_orden)

PROVEEDOR (#proveedor, nombre_proveedor, dirección_proveedor)

PRODUCTO (#producto, descripción_producto, precio_producto)

PRODUCTO_ORDENADO (#orden, #producto, cantidad_producto, precio_total_producto)

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

ANALISIS DISEÑO E IMPLEMENTACION DE SISTEMAS DE BASES DE DATOS

MATERIAL DIDACTICO

MAYO 1995

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg Cuauhtemoc 06000 Mexico, D F. APDO. Postal M-2285
Telefonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

Consideraciones de Performance en el Diseño de Bases de Datos Relacionales

OBJETIVO:

En este capítulo se estudiaran las técnicas para la obtención de un buen Diseño de Bases de Datos Relacionales.

En este capítulo el asistente:

- Conocerá el concepto de redundancia.

- Aprenderá a determinar las características que debe cumplir un buen Diseño de Bases de Datos.

- Aprenderá a aplicar técnicas de desnormalización en el Diseño de Bases de Datos para mejorar el rendimiento de las aplicaciones.

REDUNDANCIA

La repetición de información no implica necesariamente redundancia. La redundancia se puede definir como la repetición de hechos.

Suponga los dos casos siguientes:

1.

EMPLEADO

no_empleado	nombre	departamento
10	Antonio Chávez Flores	Sistemas
15	Edwin Navarro Pliego	Administración
20	Jéssica Briseño C.	Sistemas
25	Ramón Ramírez Hdez.	Sistemas

2.

EMPLEADO

no_empleado	nombre	num_depto	nombre_depto
10	Antonio Chávez Flores	5	Sistemas
15	Edwin Navarro Pliego	7	Administración
20	Jéssica Briseño C.	5	Sistemas
25	Ramón Ramírez Hdez.	5	Sistemas

En el primer caso existe repetición de información (Sistemas) pero no redundancia.

En el segundo caso se repite el hecho de que el departamento 5 es el departamento llamado Sistemas.

Un Diseño de Bases de Datos con tablas normalizadas minimiza la redundancia de datos, pero, ¿que sucede con el rendimiento?

Pueden existir consultas críticas que involucren la obtención de datos de varias tablas, en este caso el costo de los joins puede ser muy alto. Por ejemplo, obtener el estado de cuenta de un cliente a partir de las tablas CLIENTE, MOVIMIENTOS, SALDO, CUENTA.

Existen algunas aplicaciones que requieren el cálculo de datos en base a ciertas columnas de una o varias tablas. El cálculo de estos datos implica tiempo. Por ejemplo, el cálculo del salario mensual de un empleado en base a los campos prestaciones, salario_gravable, bono, insentivo.

En muchas ocasiones solamente una parte de los datos de una tabla se utiliza constantemente.

CARACTERISTICAS DE UN BUEN DISEÑO

Un buen diseño debe:

- **Minimizar la redundancia de información.**

- **Fácil de comprender**

  Las tablas que componen el diseño no deben de estar demasiado fragmentadas, ni tampoco deben contener demasiada información, ya que sería difícil determinar la entidad que representan.

  Es necesario que las tablas tengan una estructura que permitan fácilmente determinar la información que representan.

- **Ayudar a que el performance de las consultas y movimientos en la BD sea aceptable**

  La información no utilizada debe separarse.

  La información que se accesa junta debe permanecer como tal.

Un Diseño de Bases de Datos difícilmente puede satisfacer todos estos requerimientos.

ASPECTOS A EVALUAR

En un ambiente de operación real pueden existir aplicaciones de cualquier tipo sobre la Base de Datos, por lo que hay que determinar cuales son las consultas o movimientos más críticos e importantes. Para estos hay que determinar:

- La frecuencia con que se llevan a cabo
- El número de usuarios concurrentes que las ejecutan
- El volúmen de información procesada
- Procesos en ejecución cuando estos se realizan
- El tipo de operación (en línea o batch)
- El usuario que las ejecuta
- El tiempo de respuesta requerido

VENTAJAS Y DESVENTAJAS DE LA NORMALIZACION

Reduce la redundancia de datos.

Las tablas son más pequeñas, lo que implica que la Base de datos ocupa menos espacio de disco.

Como las tablas son más pequeñas los accesos a disco para leer una tabla se reducen.

Las modificaciones son más eficientes, por ejemplo, suponga los dos esquemas siguientes:

1. Tablas normalizadas

    EMPLEADO(noCta, nombre, dirección, salario, noDepto)

    DEPTO(noDepto, nombre)

2. Tabla no normalizada

    EMPLEADO(noCta, nombre, dirección, salario, noDepto, nombreDepto)

    suponga que existen 12000 empleados asignados a 50 departamentos, y que el departamento de "Sistemas" cambiará de nombre, ahora se llamará departamento de "Informática y Sistemas". ¿Bajo que esquema es más fácil hacer el cambio conservando la consistencia de la Base de Datos?

Debido a que el número de tablas se incrementa, para las consultas que requieren de mucha información, muy probablemente se tengan que llevar a cabo joins entre varias tablas.

Una consulta que involucra joins tiene, generalmente, un tiempo de respuesta mayor a una consulta sobre una sola tabla.

Suponga los siguientes esquemas:

1. Tablas Normalizadas

   PELICULA(noPel, nombre, ...)
   RENTA(noPel, cveSocio, noCopia)
   SOCIOS(cveSocio, nombre, direccion, ...)

2. Tablas no normalizadas

   PELICULA(noPel, nombre, ...)
   RENTA(noPel, cveSocio, nomPel, nomSocio, noCopia)
   SOCIOS(cveSocio, nombre, direccion, ...)

   Suponga que se desea hacer un reporte que muestre las películas rentadas y el nombre del socio que las rento, ¿bajo que esquema sería más eficiente la consulta?

DESNORMALIZACION

La desnormalización tiene como objetivo obtener un mejor performance de cierto tipo de aplicaciones sobre la Base de Datos.

Cuando se desnormaliza se debe partir de que existe un diseño normalizado.

La desnormalización se da a nivel de columna o tabla.

La desnormalización mejora el performance de cierto tipo de aplicaciones, por lo que hay que tener conocimiento de como se van ha utilizar los datos.

La desnormalización reduce el número de joins, el uso de llaves foráneas y se puede reducir el número de tablas.

El costo de desnormalizar se refleja al momento de hacer actualizaciones y mantener la integridad de la Base de Datos.

Tácticas de Desnormalización

- Añadir datos redundantes.
- Redefinición de columnas.
- Redefinición de tablas.

Duplicación de columnas

Las columnas que se van a duplicar deben de cumplir con:

- ser del mismo tipo
- tener el mismo dominio
- tener el mismo nombre

Las columnas a duplicar no deben ser muy volátiles, es decir, no deben cambiar constantemente, ya que de lo contrario se tendría que requerir un control más estricto para mantener la integridad de la Base de Datos.

La duplicación puede darse mediante una copia exacta o como una columna calculada en base a otras.

Duplicación de columnas, copias exactas

Ejemplo:

PELICULA(cvePel, nomPel, ...)

RENTA(cvePel, cveSocio, **nomPel**, **nomSocio,** noCopia)

SOCIOS(cveSocio, nomSocio, direccion, ...)

En este caso el reporte de películas rentadas indicando el nombre del socio tiene un mejor tiempo de respuesta.

Las columnas repetidas no son volátiles, ¿Cambia el nombre del socio respecto a su clave de socio?

Ejemplo:

PELICULA(cvePel, nomPel,precio ...)
COPIA(cvePel, noCopia,**precio**)

En este caso la columna precio es muy volátil, por lo que para mantener la integridad de la Base de Datos, cada vez que cambie el costo de la renta de una pelicula, se debería de hacer el cambio en las dos tablas. En el caso de un esquema normalizado, ¿como se mantiene la integridad en este caso?

Duplicación de columnas, columnas derivadas

Una columna derivada es aquella cuyos valores son obtenidos de un cálculo en base a los registros o columnas de una o varias tablas:

- Suma de varias columnas
- Conteo de registros
- etc.

Ejemplos:

PELICULA(cvePel, nomPel, genero, **numCopias**)

COPIA(cvePel, noCopia)

ALUMNO(noCta, nombre, **promedio,** ...)

CURSO(noCta, cveMat, calificación, ...)

MATERIA(cveMat, nombre, ...)

EMPLEADO(noEmp, nombre, salBase, comisión, prestaciones, **salTotal)**

Redefinición de columnas

La redefinición de columnas se da sobre llaves primarias cuando estas son muy grandes.

Se redefine la llave primaria para que esta sea más pequeña, lo que provoca que:

- Las llaves foráneas en otras tablas sean más pequeñas.

- Los joins sean más rápidos, ya que se hacen con campos de relación más pequeños.

Ejemplo:

Suponga el siguiente esquema:

PRODUCTO(***tipo, marca***, precio, ...)
VENDE(*numTienda, tipo, marca*, cantidad, fecha, ...)
TIENDA(***numTienda***, dirección, tel, ...)

Redefiniendo la llave primaria:

PRODUCTO(***cvePro***,tipo, marca, precio, ...)
VENDE(*numTienda, cvePro*, cantidad, fecha, ...)
TIENDA(***numTienda***, dirección, tel, ...)

Redefinición de tablas

El performance de las aplicaciones de la Base de Datos se puede ver afectado por alguna de las siguientes razones:

- Los renglones contienen más atributos de los que se necesitan, de modo que el tamaño del registro es muy grande y el tiempo de lectura de n registros se ve afectado.

- La tabla contiene renglones que son muy poco utilizados, lo que provoca que la lectura de datos sea más lenta.

Existen dos métodos para redefinir tablas:

- Duplicación (por renglón o por columna)

- Segmentación (por renglón o por columna)

Para elegir entre un método y otro se deben considerar los siguientes aspectos.

- Integridad de la Base de Datos.

- Redundancia.

- Espacio en disco adicional.

Duplicación/segmentación de tablas por columna

Ejemplo:

La tabla de empleados contiene 20 columnas, por lo que cada registro ocupa 350 bytes. Las aplicaciones más comunes solamente requieren del nombre, número de cuenta, salario y número de departamento del empleado; estos cuatro campos ocupan en total 40 bytes. La llave primaria de la tabla es el número de cuenta:

EMPLEADO(noCta, nombre, direccion, tel, edad, sexo, salario, noDepto, fechaContrato, nomEsposa, numHijos, ...)

Duplicación:

EMPLEADO(noCta, **nombre, salario, noDepto**, direccion, tel, edad, sexo, fechaContrato, nomEsposa, numHijos, ...)

EMP_DUP(noCta, **nombre, salario, noDepto**)

Ventajas:

- En el caso de que se requiera hacer una consulta que involucre a todas las columnas, solamente se definirá sobre la tabla EMPLEADO. No se requieren joins.

Desventajas:

- Requiere más espacio en disco

- Se requiere de mayor esfuerzo para mantener la integridad. Cuando se cambie el salario o el número de departamento de un empleado, el cambio se deberá realizar en ambas tablas.

Segmentación:

EMP_PERSONAL(noCta, direccion, tel, edad, sexo, fechaContrato, nomEsposa, numHijos, ...)

EMP_NOMINA(noCta, **nombre, salario, noDepto**)

Ventajas:

- Requiere menos espacio en disco.
- Mantener la integridad no requiere de esfuerzo adicional.

Desventajas:

- Una consulta que involucre la información de todas las columnas requiere de un join.

Duplicación/segmentación de tablas por renglón

Ejemplo:

La empresa X mantiene una tabla de empleados en donde se tienen registrados todos los empleados que alguna vez han trabajado en dicha empresa. La tabla contiene 100000 registros; sin embargo, las aplicaciones de nómina y asistencia que son las más importantes y que requieren un buen tiempo de respuesta, solamente utilizan la información de los 5000 empleados que actualmente se encuentran contratados.

Duplicación:

EMPLEADO(noCta, nombre, ...) 100000 registros

EMP_ACTIVO(noCta, nombre, ...) 5000 registros

Ventajas:

- En el caso de que se requiera hacer una consulta que involucre a todos los registros, solamente se definirá sobre la tabla EMPLEADO. No se requieren uniones.

Desventajas:

- Requiere más espacio en disco

- Se requiere de mayor esfuerzo para mantener la integridad. Cuando se cambie los datos de un empleado activo, el cambio se deberá realizar en ambas tablas.

Segmentación:

EMP_INACTIVO(noCta, nombre, ...) 95000 registros

EMP_ACTIVO(noCta, nombre, ...) 5000 registros

Ventajas:

- Requiere menos espacio en disco.
- Mantener la integridad no requiere de esfuerzo adicional.

Desventajas:

- Una consulta que involucre la información de todos los registros requiere de la union de las dos tablas.

En general es más recomendable la segmentación en ambos casos.

SQL
Structured Query Language

OBJETIVO:

En este capítulo se estudiará la forma de utilizar el lenguaje SQL para manipular la información de una Base de Datos.

En este capítulo el asistente:

- Conocerá las instrucciones de SQL necesarias para modificar la información de la Base de Datos

- Lleverá a cabo consultas de selección y proyeccción

- Utilizará funciones agregadas en sus consultas

- Realizará consultas que involucren información de más de una tabla

- Se ayudará del concepto de subconsultas para llevar a cabo consultas más complejas sobre la Base de Datos

EL LENGUAJE SQL

SQL (*Structure Query Languaje*) es un lenguaje estandar que podemos definir en tres categorias:

1. Lenguaje de definición de datos (DDL)

2. Lenguaje de control de datos (DCL)

3. Lenguaje de manipulación de datos (DML)

Características:

- Es un 4GL.

- Es un estandar.

- No incluye ninguna referencia física a los datos que accesa.

- Es un camino sencillo para obtener y manipular información de una Base de Datos.

- Puede ser utilizado interactivamente.

- Puede ser utilizado desde un lenguaje de tercera generación.

- Todas las herramientas de explotación de la Base de Datos utilizan una interface de SQL..

- Incluye instrucciones para administrar y definir la Base de Datos.

CREACION DE LA BASE DE DATOS

La creación de la Base de Datos, también se puede llevar a cabo desde SQL. El comando utilizado es:

```
CREATE DATABASE nombre
```

La Base de Datos se creará con un tamaño por dedault asignado por el DBMS de que se trate. La opción para indicar un tamaño en especial, es sintáxis propia de los diferentes DBMS´s.

CREACION DE TABLAS

Las tablas también se pueden crear desde SQL:

```
CREATE TABLE nombre (
        column_name datatype [NOT NULL] [UNIQUE],
        ...)
```

- Por default las columnas se crean con NULL como valor por default.

- UNIQUE indica que en la columna no pueden existir valores repetidos.

Ejemplos:

```
CREATE TABLE clientes (
cve_cli             integer not null unique,
nombre              char(25),
dir                 char(25),
tel                 char(10)
)

CREATE TABLE peliculas (
cve_pel             integer not null unique,
titulo        char(25),
clas                char(5),
genero              char(12)
cve_costo     char,
precio              integer
)

CREATE TABLE copias (
cve_copia           integer not null unique,
cve_pel             integer not null,
formato             char(5),
)

CREATE TABLE costos (
cve_costo     char not null unique,
costo         integer
)

CREATE TABLE renta (
cve_copia           integer,
cve_pel             integer,
cve_cli             integer,
fecha         date not null,
fecha_dev           date,
estado              char
)
```

CREACION DE INDICES

Sobre una tabla se pueden crear indices, los cuales pueden ser:

CLUSTERED	fisicamente, los registros son ordenados de acuerdo al orden que establecen las columnas sobre las que se crea el índice.
NONCLUSTERED	el orden lógico que determina el índice no corresponde al orden físico de los datos.

La sintáxis es :

```
CREATE [UNIQUE] [CLUSTER] INDEX index_name
        ON nombre_tabla (nombre_columna [ASC | DESC], ...)
```

- Se pueden incluir hasta 8 columnas en un indice compuesto

- La longitud de las columnas indexadas no puede exceder los 120 bytes

INSERCION DE DATOS

La instrucción INSERT permite insertar datos en una tabla ya existente.

- Los valores en las columnas se deben especificar en el orden en como se definieron las columnas al crear la tabla.

- Si solamente se especifican algunos valores, los no especificados toman valores nulos.

- Por la consideración anterior, para todas las columnas que no acepten nulos se debe indicar un valor.

- Los valores indicados deben ser del mismo tipo de la columna que afectan.

- Los valores de tipo CHAR y DATE deben ser encerrados entre comillas.

Sintáxis:

```
INSERT [into] nombre_table
        [(lista_de_columnas)]
        { values (lista_valores) | bloque_select}
```

Ejemplos:

```
INSERT into clientes
        values(100, 'J. Antonio Chavez', 'Almaraz 2-1','6437707')

INSERT into clientes (nombre, cve_cli, dir, tel)
        values('C. Jessica Briseño C.', 101, 'Norte 86-B 87',
                '7519256')

INSERT into clientes
        values(102, 'Edwin Navarro Pliego', NULL, NULL)

INSERT into clientes (cve_cli, nombre)
        values(103, 'Norberto Arrieta M.')
```

LABORATORIO

1. Defina las tablas que conforman la Base de Datos EMPLEADOS.

2. Inserte la información de las tablas de la Base de Datos de EMPLEADOS.

PROYECCION DE COLUMNAS

La instrucción SELECT es la más utilizada para llevar a cabo consultas sobre la Base de Datos. Una de las funciones básicas que lleva a cabo es la proyección de columnas de una tabla.

La sintáxis simplificada de SELECT que permite la proyección de columnas es:

```
SELECT lista_columnas
        FROM nombre_tabla
```

- El orden de las columnas en la instrucción determina el orden de estas en el resultado

Ejemplos:

```
SELECT * from clientes     /* obtiene toda la información de los clientes */
```

Resultado:

cve_cli	nombre	dir	tel
100	J. Antonio Chavez	Almaraz 2-1	6437707
101	C. Jessica Briseño C.	Norte 86-B 87	7519256
102	Edwin Navarro Pliego	Sur 33	5272178
103	Norberto Arrieta M.	Taxqueña 127	6282919
104	Aarón Arcos Tapia	Contreras 11	5202128

...

```
SELECT nombre, dir                  /* Obtiene el nombre y   */
       FROM clientes                /* dirección de clientes */
```

Resultado:

```
nombre                dir
---------------       -------------
J. Antonio Chavez     Almaraz 2-1
C. Jessica Briseño C. Norte 86-B 87
Edwin Navarro Pliego  Sur 33
Norberto Arrieta M.   Taxqueña 127
Aarón Arcos Tapia     Contreras 11

...
```

REGISTROS DUPLICADOS

La palabra DISTINCT permite eliminar registros repetidos

Ejemplos:

```
SELECT  genero                  /* Obtiene los generos    */
       FROM peliculas           /* de todas las peliculas */


SELECT  distinct genero         /* Obtiene los generos de */
       FROM peliculas           /* peliculas              */
```

SELECCION DE REGISTROS

Con ayuda de la palabra WHERE dentro de la instrucción SELECT, podemos condicionar los registros que se desean obtener como resultado.

Sintáxis simplificada:

```
SELECT lista_columnas
FROM nombre_tabla
WHERE expresion
```

- La expresión puede involucrar:

  · Operadores de comparación

  · Rangos (BETWEEN y NOT BETWEEN)

  · Patrones de caracteres (LIKE y NOT LIKE)

  · Valores desconocidos (NULL y NOT NULL)

  · Listas de valores (IN y NOT IN)

  · Operadores lógicos (AND y OR)

- El operador NOT niega una expresión lógica

OPERADORES DE COMPARACION

Los operadores de comparación son los siguientes:

=	igual
!= o <>	diferente
>	mayor que
<	menor que
>=	mayor o igual
<=	menor o igual

- En campos tipo CHAR los operadores >, >=, < y <= comparan lexicográficamente las cadenas

- En campos tipo DATE los operadores comparan tiempos

Ejemplos:

```
SELECT titulo                    /* obtiene las peliculas */
       FROM peliculas            /* comedia               */
       WHERE genero='comedia'
```

```
SELECT titulo                    /* obtiene las peliculas */
       FROM peliculas            /* no comedia            */
       WHERE genero!='comedia'
```

```
SELECT nombre, dir               /* obtiene el nombre y la    */
       FROM clientes             /* direccion de los clientes */
       WHERE cve_cli>120         /* con clave mayor a 120     */
```

RANGOS

La palabra BETWEEN en la instrucción SELECT permite especificar rangos de valores.

Ejemplos:

```
/* Obtiene el nombre y dirección de los clientes con clave */
/* entre 120 y 150                                         */

SELECT nombre, dir
        FROM clientes
        WHERE cve_cli between 120 and 150


/* Obtiene el nombre y dirección de los clientes cuya clave */
/* no esta entre 120 y 150                                  */

SELECT nombre, dir
        FROM clientes
        WHERE cve_cli not between 120 and 150
```

PATRONES DE CARACTERES

La palabra LIKE en la instrucción SELECT permite especificar valores que cumplen con un patrón. Se utilizan en campos tipo CHAR o DATE.

Los metacaracteres o *wildcars* para LIKE son:

%	especifica cero o más caracteres
_	especifica un caracter
[..]	especifica un caracter en un rango [a-z] un caracter de la **a** a la **z**

Ejemplos:

```
SELECT nombre                          /* clientes cuyo nombre  */
      FROM clientes                    /* comience con A, B O C */
      WHERE nombre like '[ABC]%'
```

```
SELECT nombre                          /* clientes cuyo nombre no */
      FROM clientes                    /* comience con A, B O C   */
      WHERE nombre not like '[ABC]%'
```

LISTAS DE VALORES

La palabra IN en la instrucción SELECT permite especificar una lista de valores para una columna.

Ejemplos:

```
SELECT nombre                          /* Obtiene las peliculas  */
    FROM peliculas                     /* comedia y de terror    */
    WHERE genero IN ('comedia', 'terror')
```

OPERADORES LOGICOS

Los operadores lógicos sirven para unir expresiones.

- Con AND la expresión es verdadera si las expresiones que involucra lo son.

- Con OR la expresión es verdadera si alguna de las expresiones que involucra es verdadera.

- La presedencia de los operadores es:

AND
OR
NOT

- La presedencia se puede cambiar incluyendo parentesis.

Ejemplos:

```
/* Obtiene el nombre y dirección de los clientes con clave */
/* entre 120 y 150                                         */

SELECT nombre, dir
        FROM clientes
        WHERE cve_cl >= 120 and cve_cli <= 150


/* Obtiene las peliculas cómicas de costo A */

SELECT nombre
        FROM peliculas
        WHERE genero = 'comedia' AND cve_costo = 'A'
```

```
/* Obtiene las peliculas cómicas o de costo A */

SELECT nombre
        FROM peliculas
        WHERE genero = 'comedia' OR cve_costo = 'A'
```

/* Obtiene las peliculas cómicas cuya clave este entre 315 y 400 o las peliculas de vaqueros */

```
SELECT nombre
        FROM peliculas
        WHERE genero = 'comedia' AND
                clave_pel >= 315 AND clave_pel <= 400
                OR genero = 'western'
```

/* Obtiene las peliculas cómicas o aquellas cuya clave este entre 315 y 400 y las peliculas de vaqueros */

```
SELECT nombre
        FROM peliculas
        WHERE (genero = 'comedia' OR
                clave_pel >= 315 AND clave_pel <= 400)
                AND genero = 'western'
```

RENOMBRANDO COLUMNAS

En los resultados se permite utilizar otro nombre para las columnas listadas en la instrucción SELECT.

```
SELECT nombre 'Nombre Cliente', dir 'Direccion'
        FROM clientes
```

Resultado:

Nombre Cliente	Direccion
J. Antonio Chavez	Almaraz 2-1
C. Jessica Briseño C.	Norte 86-B 87
Edwin Navarro Pliego	Sur 33
Norberto Arrieta M.	Taxqueña 127
Aarón Arcos Tapia	Contreras 11

...

```
SELECT 'Cliente', nombre, dir
        FROM clientes
```

Resultado:

	Nombre Cliente	Direccion
Cliente	J. Antonio Chavez	Almaraz 2-1
Cliente	C. Jessica Briseño C.	Norte 86-B 87
Cliente	Edwin Navarro Pliego	Sur 33
Cliente	Norberto Arrieta M.	Taxqueña 127
Cliente	Aarón Arcos Tapia	Contreras 11

...

OPERADORES NUMERICOS

Los operadores aritméticos permitidos son:

+
-
*
/

- Pueden ser utilizados en expresiones númericas en la lista de columnas del SELECT o bien en WHERE

VALORES NULOS

Un valor NULL implica un valor desconocido:

- Un valor NULL no implica cero o una cadena vacía.

- Un valor NULL no es igual a otro valor NULL.

- Para seleccionar registros con valores NULL en algúna columna se utiliza IS NULL (también = NULL).

- Operaciones que involucran NULL dan como resultado NULL.

Ejemplos:

```
SELECT 1500 + NULL
```

Resultado:

```
NULL
```

```
SELECT nombre                    /* Obtiene peliculas     */
    FROM peliculas               /* de genero desconocido */
    WHERE genero IS NULL
```

LABORATORIO

1. Liste toda la información de empleados.

2. Liste el nombre y salario de los empleados.

3. Genere un listado que tenga como encabezados "EMPLEADO" y "PERCEPCION TOTAL" y contenga el nombre del empleado y su percepción total(salario más comisión).

4. Liste los diferentes tipos de puestos que existen.

5. Liste la información de todos los empleados del departamento 30.

6. Liste el nombre y salario de los "Clerck".

7. ¿Qué empleados perciben más de comisión que de salario?

8. Liste los "Salesman" del departamento 30 que tengan un salario mayor a 1500.

9. Liste todos los empleados que son "Manager" o que ganan más de 3000.

10. Proporcione el nombre de los "Manager" o bien de los "Clerck", estos últimos del departamento 10.

11. ¿Qué empleados del departamento 10 no son ni "Manager" ni "Clerck".

12. ¿Qué empleados ganan entre 1200 y 1400?

SELECT/ORDER BY

La claúsula ORDER BY en la instrucción SELECT permite ordenar el resultado de la consulta.

- El ordenamiento es ascendente por default.

- Los null se consideran al principio.

Sintáxis simplificada:

```
SELECT [DISTINCT] lista_columnas
        FROM nombre_tabla
        [WHERE expresion]
        [ORDER BY {columna | expresion} [asc | desc] [,...]]
```

Ejemplos:

```
SELECT nombre                       /* Lista de clientes ordenada */
        FROM clientes               /* nombre                     */
        ORDER BY nombre

SELECT nombre                       /* Obtiene una lista de       */
        FROM peliculas              /* peliculas comedia ordenada*/
        WHERE genero='comedia'
        ORDER BY nombre
```

FUNCIONES AGREGADAS

Las funciones agregadas permitidas son:

SUM	calcula la suma de los valores en una columna
AVG	calcula el promedio de los valores de una columna
MAX	obtiene el valor máximo en una columna
MIN	obtiene el valor mínimo en una columna
COUNT	calcula el número de registros

- Las funciones agregadas ignoran los valores NULL (excepto count(*)).

- Sum y avg sólo trabajan con valores númericos.

- Solamente se regresa un valor como resultado.

- Pueden aplicarse a todos los registros de una tabla o a un subconjunto con ayuda de WHERE.

- La palabra DISTINCT es permitida en las funciones sum, avg y count.

- La palabra DISTINCT es utilizada solamente con nombres de columnas.

- Se puede utilizar más de una función agregada en una instrucción SELECT

Ejemplos:

```
SELECT count(*)              /* Obtiene el número de */
    FROM peliculas           /* peliculas           */

SELECT max(precio)                /* Obtiene la pelicula  */
    FROM peliculas                /* más cara             */

SELECT min(costo)                 /* Obtiene el precio más */
    FROM costos                   /* alto por una renta    */

SELECT avg(precio)                /* Obtiene el precio     */
    FROM peliculas                /* promedio de las peliculas*/

SELECT avg(costo)                 /* Obtiene el precio     */
    FROM costos                   /* promedio de una renta */

SELECT max(precio)                /* Obtiene la pelicula   */
    FROM peliculas                /* de comedia más cara   */
    WHERE genero='comedia'

SELECT min(costo), max(costo)     / * Obtiene el precio más */
    FROM costos                   /* alto y el más bajo    */
                                  /* por una renta         */

SELECT count(*)              /* Obtiene el número de  */
    FROM copias                   /* copias de la película */
    WHERE cve_pel = 312           /* con clave 312         */
```

VALORES NULL EN FUNCIONES AGREGADAS

Las funciones agregadas ignoran los valores NULL:

- Para asignar un valor a los NULL se utiliza:

isnull(expresion, valor)

Ejemplos:

```
SELECT avg(isnull(price, 120))
        FROM peliculas
```

```
SELECT isnull(price, 0) * 1.1
        FROM peliculas
```

AGRUPACION DE REGISTROS

La clausula GROUP BY en la instrucción SELECT permite agrupar registros.

- Generalmente es utilizada en combinación con una función agregada.

- Todos los valores NULL son tratados como un grupo.

- La clausula WHERE se lleva a cabo antes de formar los grupos.

Sintáxis simplificada:

```
SELECT [DISTINCT] lista_columnas
        FROM nombre_tabla
        [WHERE  expresión]
        [GROUP BY  expresión]
        [ORDER BY ...]
```

Ejemplos:

```
SELECT genero, avg(precio)          /* Mal uso. Todas los  */
      FROM peliculas                /* generos aparecen con*/
                                    /* el mismo precio     */
                                    /* promedio            */

SELECT genero, avg(precio)          /* Uso correcto        */
      FROM peliculas
      GROUP BY genero

SELECT genero, avg(precio)          /* Lista ordenada de   */
      FROM peliculas                /* promedios de peliculas*/
      GROUP BY genero               /* por genero          */
      ORDER BY avg(precio)
```

SELECT GROUP BY/HAVING

La clausula HAVING condiciona a los grupos que se generan como resultado. La condición se aplica después de que se han formado los grupos.

Sintáxis simplificada:

```
SELECT [DISTINCT] lista_columnas
        FROM nombre_tabla
        [WHERE  expresión]
        [GROUP BY  expresión]
        [HAVING expresion]
        [ORDER BY ...]
```

Ejemplos:

```
SELECT genero, avg(precio)          /* Obtiene una lista  */
        FROM peliculas              /* del promedio de    */
        GROUP BY genero             /* precio de las peli-*/
        HAVING avg(precio) > 100    /* culas por genero   */
                                    /* para los generos   */
                                    /* cuyo promedio es   */
                                    /* mayor a 100        */

SELECT genero, avg(precio)          /* Igual que la anterior */
        FROM peliculas              /* pero la lista es      */
        GROUP BY genero             /* ordenada              */
        HAVING avg(precio) > 100
        ORDER BY avg(precio)
```

LABORATORIO

1. Liste a los empleados agrupados por departamento.

2. ¿Cuál es el salario promedio de los empleados?

3. ¿Cuá es el salario más alto que se paga a un empleado?

4. Liste el salario promedio por tipo de trabajo.

5. ¿Cuál es el puesto en donde en promedio se gana un mayor salario?

6. Listar el promedio de salario por tipo de trabajo en cada departamento, indicando cuantos empleados tienen ese puesto en cada departamento.

7. Listar el salario promedio de los puestos que tienen más de dos empleados.

JOINS

El JOIN es la operación más importante en el modelo relacional de Bases de Datos.

Un JOIN permite obtener información de más de una tabla.

El JOIN es una selección sobre un producto cruz.

Para poder resolver una consulta mediante un JOIN se debe determinar:

- Tablas involucradas
- Columnas de relación
- Condición de selección

El resultado de un producto cruz contiene el nombre de todas las columnas de ambas tablas identificadas. El nombre de las columnas es de la forma: TABLA.NOMBRE_COLUMNA.

El resultado del producto cruz son nxm registros, donde m es el número de resgistros de la primera tabla y n el de la segunda.

Ejemplo:

Listar las peliculas y su costo de renta.

Para este caso las tablas involucradas son: PELICULA y COSTO

El producto cruz de PELICULA y COSTO considerando las tablas siguientes se muestra a continuación.

PELICULA

CVE_PELICULA	TITULO	CVE_COSTO
1	RAMBO I	A
15	RAMBO II	A
39	DUMBO	B
47	RAMBO III	C

COSTO

CVE_COSTO	COSTO
A	5.00
B	6.00
C	6.50
D	7.50

PELICULA X COSTO

CVE_PELICULA	TITULO	CVE_COSTO	CVE_COSTO	COSTO
1	RAMBO I	A	A	5.00
1	RAMBO I	A	B	6.00
1	RAMBO I	A	C	6.50
1	RAMBO I	A	D	7.50
15	RAMBO II	A	A	5.00
15	RAMBO II	A	B	6.00
15	RAMBO II	A	C	6.50
15	RAMBO II	A	D	7.50
39	DUMBO	B	A	5.00
39	DUMBO	B	B	6.00
39	DUMBO	B	C	6.50
39	DUMBO	B	D	7.50
47	RAMBO III	C	A	5.00
47	RAMBO III	C	B	6.00
47	RAMBO III	C	C	6.50
47	RAMBO III	C	D	7.50

Las columnas de relación son CVE_COSTO en PELICULA y CVE_COSTO en COSTO.

La condición es que CVE_COSTO en PELICULA sea igual a CVE_COSTO en COSTO.

La consulta se expresaría como:

```
select titulo, costo
        from PELICULA, COSTO
        where PELICULA.cve_costo = COSTO.cve_costo
```

CONDICIONES PARA JOINS

El JOIN puede involucrar n tablas.

El nombre de la columna en el SELECT debe ir precedido por el nombre de la tabla si existe ambiguedad, por ejemplo, para dos columnas con el mismo nombre de diferentes tablas.

Las columnas involucradas en la condición del JOIN deben de ser de tipos compatibles.

Los valores NULL no participan en un JOIN.

Las columnas participantes en la condición del JOIN no necesariamente deben de aparecer en el resultado.

Se pueden incluir más condiciones de selección u otros comandos, por ejemplo ORDER BY, GROUP BY, etc.

La condición del JOIN no necesariamente debe de ser de igualdad, puede involucrar cualquier operador: !=, >, <, etc.

Ejemplos:

Listar las peliculas cuyo costo de renta sea mayor a 6.00:

```
select titulo, costo, PELICULA.cve_costo
        from PELICULA, COSTO
        where PELICULA.cve_costo = COSTO.cve_costo
        and costo > 6.00
```

Listar las peliculas que tienen más de cinco copias:

```
select titulo, count(*)
        from PELICULA, COPIAS
        where PELICULA.cve_pel = COPIAS.cve_pel
        group by titulo
        having count(*) > 5
```

Listar las películas rentadas por cliente indicando la fecha de renta.

```
select cve_cli, titulo, fecha
        from CLIENTES, PELICULA, RENTA
        where CLIENTES.cve_cli = RENTA.cve_cli
        and   RENTA.cve_pel = PELICULA.cve_pel
```

ALIAS Y SELFT JOIN

Para abreviar la escritura se pueden utilizar alias para las tablas participantes en un JOIN:

```
select cve_cli, p.cve_pel, titulo, fecha
        from CLIENTES c, PELICULA p, RENTA r
        where c.cve_cli = r.cve_cli
        and   r.cve_pel = p.cve_pel
```

Un SELFT JOIN es un JOIN en donde solamente participa una tabla.

En un SELFT JOIN es necesario utilizar alias.

Ejemplo:

Considere la siguiente tabla:

CIUDADANO(cve, nombre, direccion, tel, cve_conyuge)

Para listar el nombre de los ciudadanos y el de su conyuge:

```
select a.nombre, b.nombre
        from CIUDADANO a, CIUDADANO b
        where a.cve_conyuge = b.cve
```

LABORATORIO

1. ¿En donde trabaja Allen?

2. Listar el nombre de los empleados y de su departamento.

3. Listar todos los empleados que trabajan en Chicago.

4. ¿Qué empleados ganan más que Jones?

5. Listar el nombre de los empleados y el de su jefe.

6. ¿Cuantos empleados tiene a su cargo cada empleado?

SUBCONSULTAS

Muchas consultas intuitivamente se pueden resolver como subconsultas en lugar de resolverse por JOIN.

Es más fácil pensar en una solución mediante una subconsulta que con un JOIN.

Ejemplo:

¿Qué peliculas tienen un costo de 6.00?

Utilizando JOINS:

```
select titulo
        from PELICULAS, COSTOS
        where PELICULAS.cve_costo = COSTOS.cve_costo
        and costo = 6.00
```

Utilizando subconsultas:

```
select titulo
        from PELICULAS
        where cve_costo =
                (select cve_costo
                        from COSTOS
                        where costo = 6.00)
```

Una subconsulta es una sentencia SELECT utilizada en una expresión como parte de otra sentencia SELECT.

La subconsulta es resuelta y el resultado es sustituido en la consulta externa.

La columna involucrada en la selección de la consulta externa debe de ser de un tipo similar a la columna proyectada en la subconsulta.

La subconsulta no puede proyectar como resultado más de una columna.

El resultado de una subconsulta puede provenir de una función agregada.

El único resultado que aparece como salida es el que genera la consulta externa.

Ejemplos:

¿En que fechas ha rentado peliculas Jéssica Briseño?

```
select fecha
        from RENTA
        where cve_cli =
        (select cve_cli
                from CLIENTES
                where nombre = "Jéssica Briseño")
```

¿Cuál es la película más cara de terror?

```
select titulo
        from PELICULAS
        where precio =
                (select max(precio)
                        from PELICULAS
                        where genero = "terror")
```

¿Qué películas cómicas tienen un precio mayor que cualquier película de terror?

```
select titulo
        from PELICULAS
        where precio =
                (select max(precio)
                        from PELICULAS
                        where genero = "terror")
        and genero = "comedia"
```

¿Cuál es la clave de la última película rentada?

```
select cve_pel
        from RENTA
        where fecha =
        ( select max(fecha)
                from RENTA )
```

Una consulta puede contener varios niveles de subconsultas.

Cuando una subconsulta genera un sólo resultado, se pueden utilizar los siguientes operadores en la consulta externa:

=, !=, >, >=, <, <=

Si se utilizan los operadores anteriores y la subconsulta genera como resultado varios registros, existe un error

Cuando la subconsulta genera varios registros, se pueden utilizar los operadores "in" y "not in" en la consulta externa.

Ejemplos:

¿Qué películas tienen copias en formato beta?

```
select titulo
        from PELICULAS
        where cve_pel in
        ( select cve_pel
                from COPIAS
                where formato = "beta")
```

¿Cuales son las películas para las que no hay copias en formato beta?

```
select titulo
        from PELICULAS
        where cve_pel not in
        ( select cve_pel
                from COPIAS
                where formato = "beta")
```

¿Qué peliculas se rentaron en el pasado mes de febrero?

```
select distinct titulo
        from PELICULAS
        where cve_pel in
        (select cve_pel
                from RENTA
                where fecha >= "1/2/1994"
                        and fecha <= "28/2/1994")
```

¿Qué peliculas ha rentado Jéssica Briseño?

```
select distinct titulo
        from PELICULAS
        where cve_pel in
        (select cve_pel
                from RENTA
                where cve_cli =
                (select cve_cli
                        from CLIENTES
                        where nombre = "Jéssica Briseño"))
```

¿En que fecha debe devolver Jéssica Briseño la película "Los gritos del silencio"?

```
select fecha_dev
        from RENTA
        where cve_pel =
        (select cve_pel
                from PELICULAS
                where titulo = "Los gritos del silencio")
        and cve_cli =
        (select cve_cli
                from CLIENTES
                where nombre = "Jéssica Briseño")
```

¿En que formato rento la película "Los gritos del silencio" Jéssica Briseño el 8/12/93?

```
select formato
        from COPIAS
        where cve_copia =
        ( select cve_copia
                from RENTA
                where cve_pel =
                (select cve_pel
                        from PELICULAS
                        where titulo = "Los gritos del silencio")
                and cve_cli =
                (select cve_cli
                        from CLIENTES
                        where nombre = "Jéssica Briseño")
                and fecha = "8/12/93")
        and cve_pel =
        (select cve_pel
                from PELICULAS
                where titulo = "Los gritos del silencio")
```

LABORATORIO

1. ¿Qué empleados tienen el mismo puesto de JONES?

2. ¿Qué empleados ganan más que algun ampleado del departamento 30?

3. ¿Qué empleados ganan más que cualquier empleado del departamento de SALES?

4. ¿Qué empleados tienen el mismo puesto y salario que FORD?

5. ¿Qué empleados tienen el mismo puesto que JONES o un salario mayor o igual al de FORD?

6. ¿Qué empleados del departamento 10 tienen el mismo puesto que algún empleado del departamento de SALES?

7. ¿Qué empleados tienen el mismo puesto que algún empleado de Chicago?

8. ¿Qué empleado es el que gana más?

9. ¿Quién es el empleado más joven?

INSERCION DE DATOS

La instrucción INSERT sirve para insertar datos en una tabla ya existente.

La sintáxis es la siguiente:

```
insert into table_name [(lista_columnas)]
{values(expresiones_constantes)}
```

Si no se indican las columnas para las cuales se va ha insertar información, se deben proporcionar los valores en el orden en como se definio la tabla, indicando NULL para aquellas columnas que no se proporcione dato (estas columnas deberán estar definidas para aceptar NULL):

```
insert into CLIENTES
values (67, "Jéssica Briseño C.", NULL, NULL)

insert into PELICULAS
values (84, "Batman", NULL, "acción", A, 100)
```

Si se indican las columnas, aquellas que no se indiquen deberán aceptar el valor NULL:

```
insert into PELICULAS (cve_pel, titulo)
values (49, "Nico")
```

MODIFICACION DE DATOS

La instrucción UPDATE permite la modificación de las columnas de los renglones de una tabla.

La sintáxis es la siguiente:

```
update table_name
set columna = { expresion }
[, columna = {expresion }] ...
    [ where condiciones ]
```

No se pueden modificar varias tablas con una sola instrucción UPDATE.

Ejemplos:

```
UPDATE emp set salary=2000
   where ename = "SMITH"

UPDATE COSTOS set costo = costo * 1.2

UPDATE CLIENTES set direccion = "Av. Centenario 12543",
                    tel = "367-82-82",
                        where cve_cli = 123
```

BORRADO DE DATOS

La instrucción DELETE permite el borrado de registros dentro de una tabla.

La sintáxis es la siguiente:

```
delete from table_name
[where condiciones]
```

Si no se indica la claúsula where se borran todos los registros de la tabla.

Ejemplos:

```
DELETE from RENTA
    where fecha_dev < "1 Mar 1994"

DELETE from COPIAS
    where cve_pel = 38

DELETE from CLIENTES   /* Cuidado borra toda la información */
```

BORRADO DE TABLAS

Para borrar una tabla (definición y datos) se utiliza la instrucción DROP TABLE.

La sintáxis es la siguiente:

```
DROP TABLE table_name [, table_name ]
```