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

CENTRO DE INFORMACION Y DOCUMENTACION
"ING. BRUNO MASCANZONI"

El Centro de Información y Documentación Ing. Bruno Mascanzoni tiene por objetivo satisfacer las necesidades de actualización y proporcionar una adecuada información que permita a los ingenieros, profesores y alumnos estar al tanto del estado actual del conocimiento sobre temas específicos, enfatizando las investigaciones de vanguardia de los campos de la ingeniería, tanto nacionales como extranjeras.

Es por ello que se pone a disposición de los asistentes a los cursos de la DECFI, así como del público en general los siguientes servicios:

- **Préstamo interno.**
- **Préstamo externo.**
- **Préstamo interbibliotecario.**
- **Servicio de fotocopiado.**
- **Consulta a los bancos de datos: librunam, seriunam en cd-rom.**

Los materiales a disposición son:

- **Libros.**
- **Tesis de posgrado.**
- **Noticias técnicas.**
- **Publicaciones periódicas.**
- **Publicaciones de la Academia Mexicana de Ingeniería.**
- **Notas de los cursos que se han impartido de 1980 a la fecha.**

En las áreas de ingeniería industrial, civil, electrónica, ciencias de la tierra, computación y, mecánica y eléctrica.

El CID se encuentra ubicado en el mezzanine del Palacio de Minería, lado oriente.

El horario de servicio es de 10:00 a 19:30 horas de lunes a viernes.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

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

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

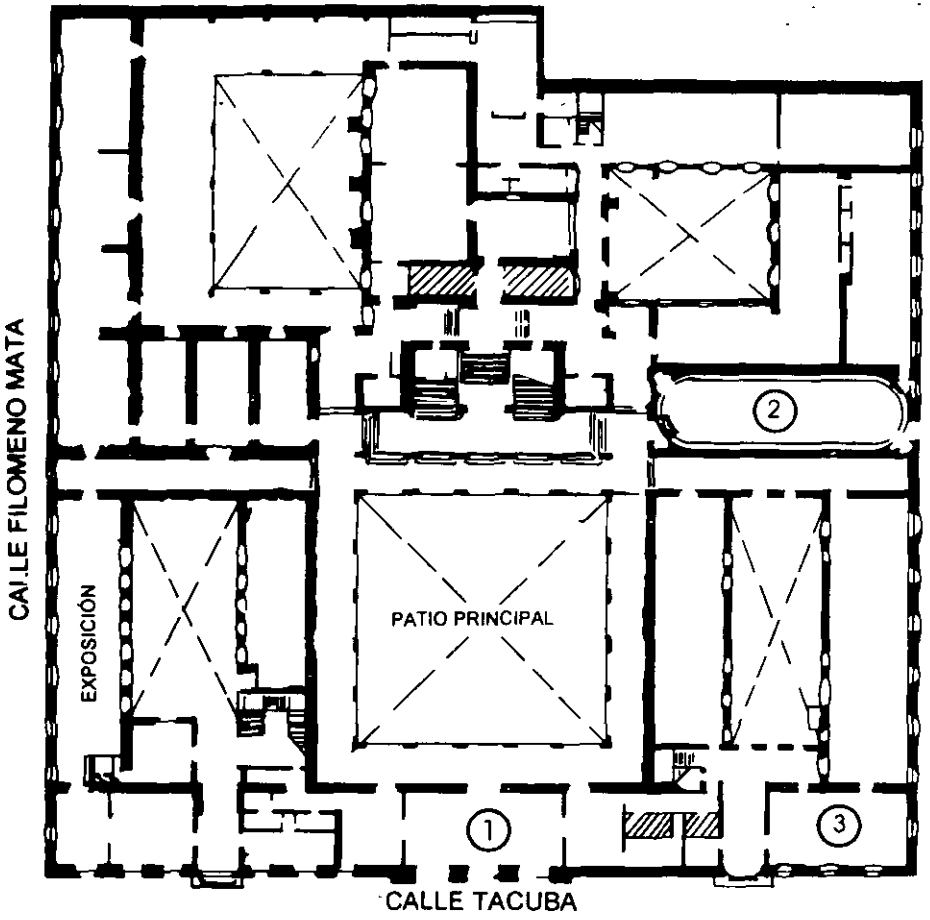
Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

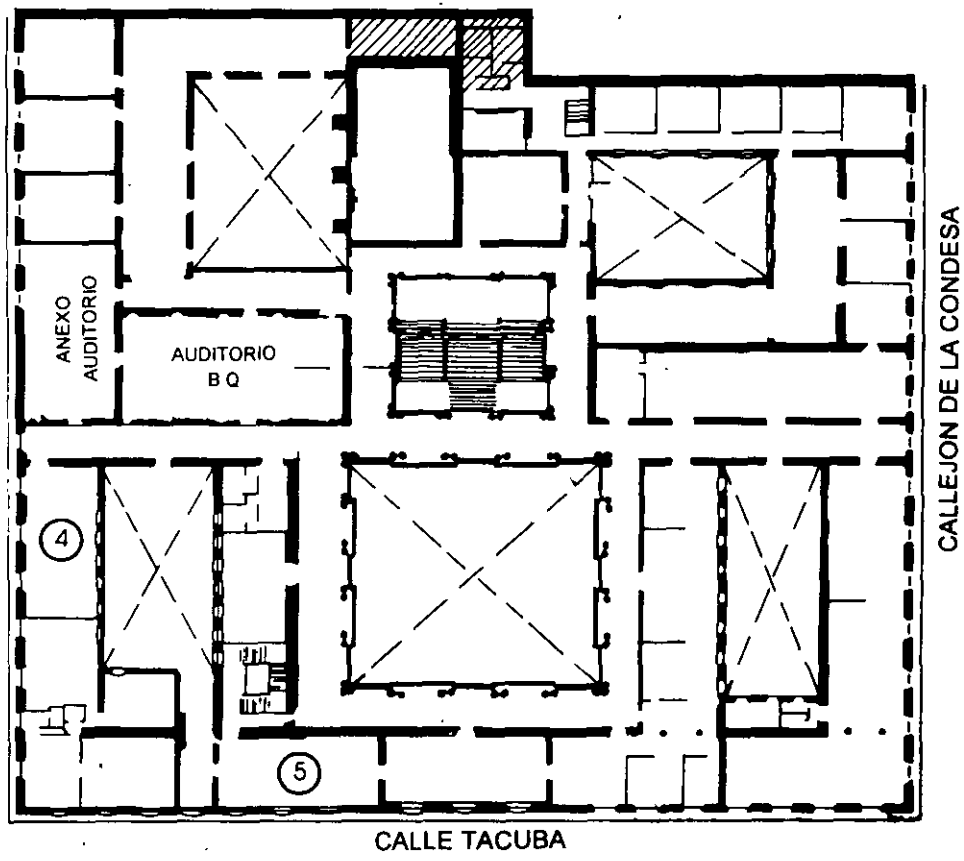
División de Educación Continua.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

PALACIO DE MINERIA

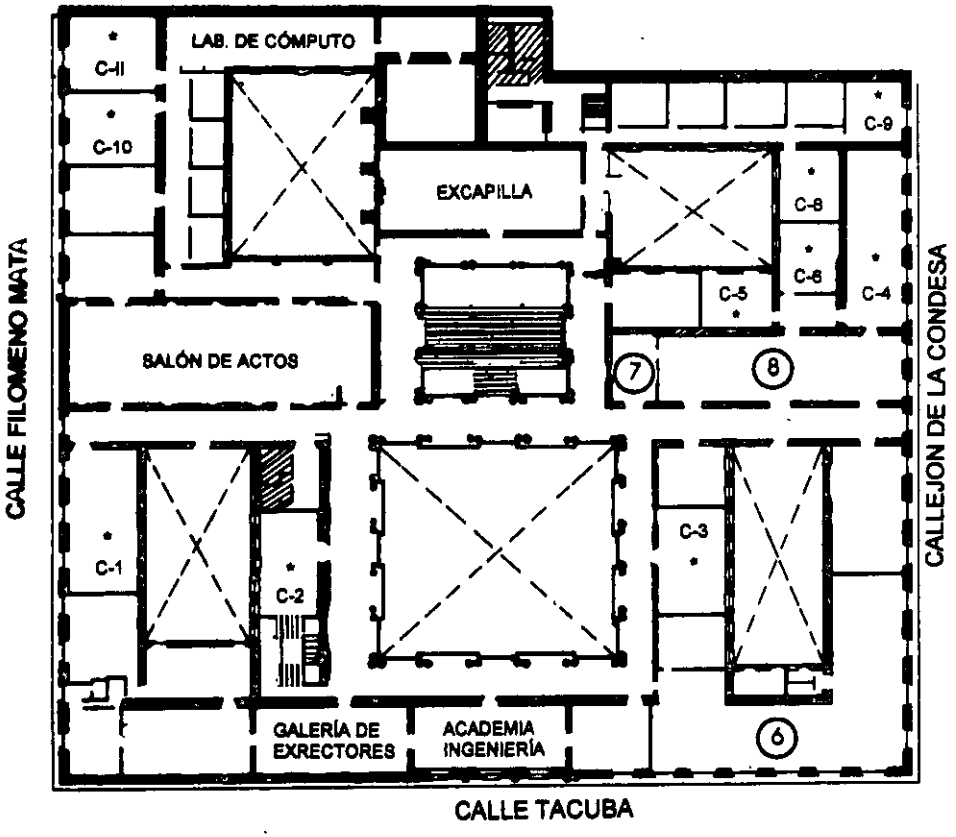


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



Ier. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA
DEC

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

SQL (STRUCTURED QUERY LANGUAGE)

Introducción al ambiente de Bases de Datos

JUNIO 1998

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

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 eficiéntes. 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

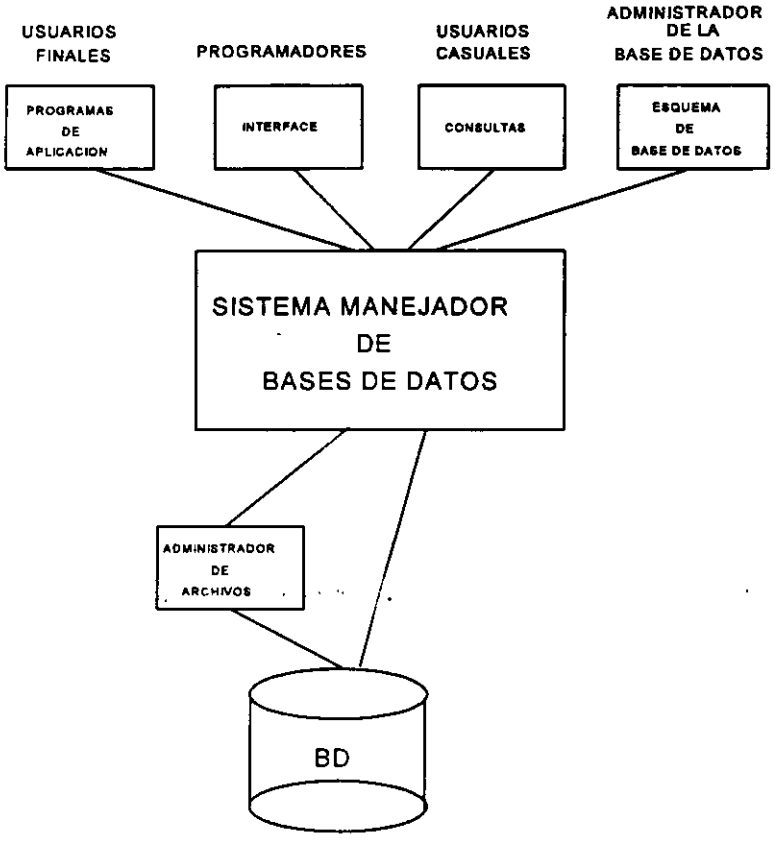
BASE DE DATOS

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 nuèstra 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.

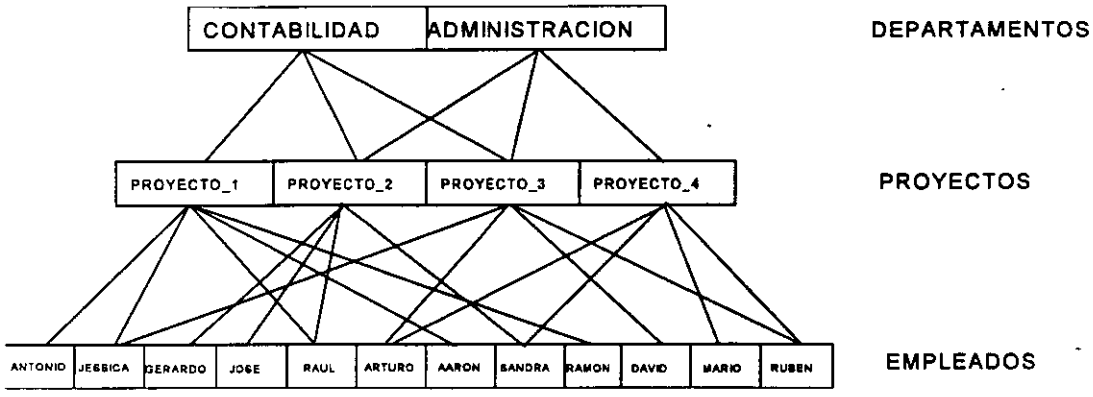
Los modelos lógicos basados en registros, no incluyen un mecanismo para la representación directa de código en la Base de Datos. En cambio, tienen asociados lenguajes que permiten expresar consultas y actualizaciones sobre la Base de Datos.

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

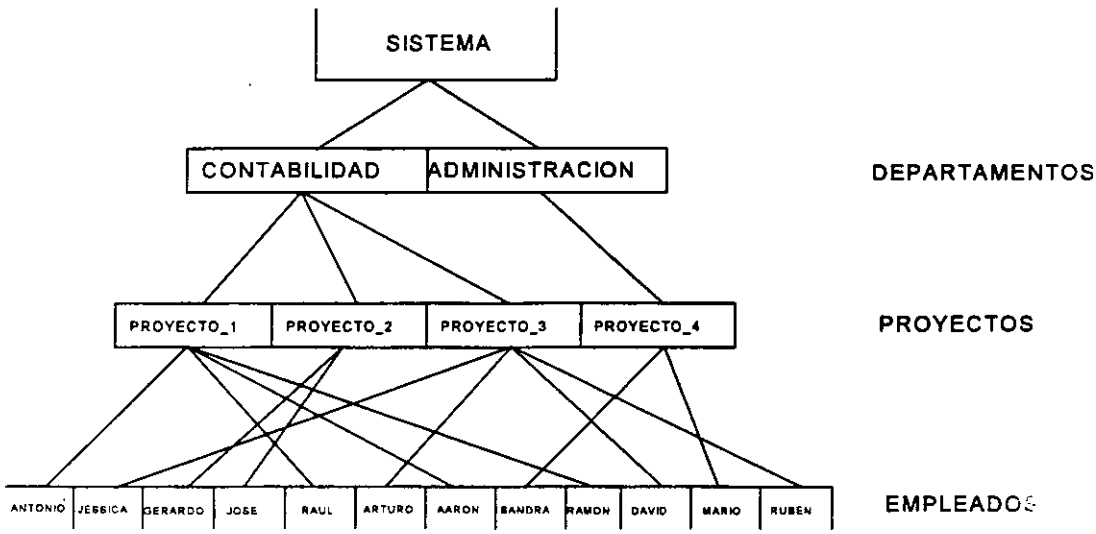
- Modificar la estructura de la Base de datos implica redefinir todo el esquema

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.

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

SQL (STRUCTURED QUERY LANGUAGE)

El modelo Relacional de Bases de Datos

JUNIO 1998

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg Cuauhtémoc 06000 Mexico, D F APDO Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 25

El Modelo Relacional de Bases de Datos

OBJETIVO:

En este capítulo se estudiará la terminología asociada al Modelo Relacional de Bases de Datos.

En este capítulo el asistente:

- Definirá el concepto de Base de Datos Relacional.

- Conocerá el significado del valor nulo.

- Definirá los conceptos Llave Primaria y Llave Foránea.

- Definirá lo que es Integridad de la Base de Datos.

EL MODELO RELACIONAL

El modelo relacional fué originalmente propuesto por Codd en 1970 en un escrito titulado "A Relational Model of Data for Large Shared Data Banks".

El primer producto relacional que fué desarrollado en base a esta teoría se llamo System R, un DBMS desarrollado por IBM en los 70´s.

El modelo relacional esta definido en base a las estructuras de datos que permiten representarlo y a las operaciones que se pueden realizar sobre los datos.

ESTRUCTURA DEL MODELO RELACIONAL

En el modelo relacional, los datos y las relaciones entre los datos se representan por medio de **tablas**, a las cuales se les asigna un nombre único.

Cada una de las tablas esta compuesta por **atributos** con nombres únicos, las cuales tienen asociado un **tipo de dato**.

El contenido de una tabla es un conjunto de **renglones**, los cuales tienen un valor para cada uno de los atributos de la tabla. Los valores permitidos para cada atributo estan limitados al tipo de dato del atributo.

Ejemplos:

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

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	Edwin	Cozumel 11-3	731-82-83	40
84324	Jorge	Fco. Sosa 266	527-73-12	20

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

El modelo relacional tiene un fuerte fundamento matemático.

Un atributo de una tabla representa una relación entre un conjunto de valores.

Un **dominio** es un conjunto de valores. Un atributo tiene asociado un conjunto de valores permitidos, es decir, un dominio.

Los matemáticos definen una **relación** como un subconjunto de un producto cartesiano de una lista de dominios, por lo tanto, las tablas son esencialmente relaciones.

Una relación es un conjunto de **tuplas** con valores para cada uno de los atributos.

Características de una relación

Una relación en el modelo relacional tiene las siguientes características:

- Los dominios de los atributos deben ser atómicos, es decir los elementos del dominio son unidades indivisibles.
- Cada atributo tiene un nombre único en la relación.
- Los valores de una tupla corresponden a los dominios de los atributos.
- El orden de los atributos no tiene importancia.
- Cada tupla es única, no existen tuplas duplicadas.
- El orden de las tuplas no es importante.

OPERACIONES

Las operaciones que se pueden llevar a cabo en el modelo relacional tienen un fundamento matemático conocido como **Algebra Relacional**.

El Algebra Relacional es un lenguaje de consulta abstracto procedural. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones.

Las operaciones definidas en el Algebra Relacional tienen la propiedad de cerradura, ya que el resultado de la operación es una relación.

Existen cinco operaciones fundamentales:

- Proyección π
- Elección σ
- Producto Cartesiano X
- Unión **U**
- Resta de conjuntos -
- Intersección de conjuntos $\cap$

Para ejemplificar las operaciones, se utilizaran las siguientes relaciones:

ALUMNO

no_cta	nom_alum	cve_carr
105	Sandra	027
107	José	032
101	Luis	032
103	Lourdes	027
106	Juan	029
102	Santiago	029
108	Véronica	027

PROFESOR

cve_prof	nom_prof
acf000	Antonio
rrh000	Ramón
enp000	Edwin
jbc000	Jéssica

CARRERA

cve_carr	carrera
027	Industrial
032	Computación
039	Electrónica

Proyección

Es una operación unitaria, ya que actúa sobre una sola relación. Se representa con la letra griega π. Se utiliza para proyectar atributos, de una relación, su sintaxis es la siguiente:

$$\pi_{\text{atributo 1, atributo 2, ..., atributo n}} (\text{Relación})$$

Ejemplo: Listar a los alumnos y su número de cuenta

$$\pi_{\text{nom\_alum, no\_cta}} (\text{ALUMNO})$$

nom_alum	no_cta
Sandra	105
José	107
Luis	101
Lourdes	103
Juan	106
Santiago	102
Verónica	108

Ejemplo: obtener el nombre de los profesores:

$$\pi_{nom\_prof}(PROFESOR)$$

nom_prof
Antonio
Ramón
Edwin
Jéssica

Elección

Al igual que la proyección la elección es una operación unitaria. Se representa con la letra griega σ, y su función es elegir aquellas tuplas que cumplan con la condición especificada. Su sintaxis es la siguiente:

$$\sigma_{condición}(Relación)$$

donde:

condición: es una expresión la cual puede tener operadores de relación (>, <, >=, <=, =, !=) y operadores lógicos (AND, OR, NOT).

Ejemplo: ¿Qué alumnos cursan la carrera cuya clave de carrera es 027?

$$\sigma_{cve\_carr=027}(ALUMNOS)$$

nom_alum	cve_carr
Sandra	027
Lourdes	027
Véronica	027

Las operaciones del Algebra Relacional tienen la propiedad de cerradura, por lo que el resultado de una operación se puede utilizar como argumento para otra.

Ejemplo: ¿Cual es la clave de la carrera de Computación?

$$\pi_{\text{cve\_carr}}(\sigma_{\text{carrera= "Computación"}}(\text{CARRERA}))$$

cve_carr
027

Producto cruz

Es una operación binaria ya que actúa sobres dos relaciones, se representa por una cruz X. Esta operación permite combinar ó relacionar informacion de dos o mas relaciones. Su sintaxis es la siguiente:

(Relación1) X (Relación2)

Esta operación nos da como resultado una relación que tiene **nXm** tuplas y todos los atributos de cada relación.

donde:

n= número de tuplas de Relación1
m= número de tuplas de Relación2

Los atributos de la relación resultado son los atributos de las relaciones sobre las que se realiza la operación, el nombre de estos esta precedido por el nombre de la relación de la que provienen, de esta forma se sigue cumpliendo el que los nombres de los atributos sean únicos.

Ejemplo:

(ALUMNO X CARRERA)

ALUMNO. no_cta	ALUMNO. nom_alum	ALUMNO. cve_carr	CARRERA. cve_carr	CARRERA. carrera
105	Sandra	027	027	Industrial
105	Sandra	027	032	Computación
105	Sandra	027	029	Electrónica
107	José	032	027	Industrial
107	José	032	032	Computación
107	José	032	029	Electrónica
101	Luis	032	027	Industrial
101	Luis	032	032	Computación
101	Luis	032	029	Electrónica
103	Lourdes	027	027	Industrial
103	Lourdes	027	032	Computación
103	Lourdes	027	029	Electrónica
106	Juan	029	027	Industrial
106	Juan	029	032	Computación
106	Juan	029	029	Electrónica
102	Santiago	029	027	Industrial
102	Santiago	029	032	Computación
102	Santiago	029	029	Electrónica
108	Véronica	027	027	Industrial
108	Véronica	027	032	Computación
108	Véronica	027	029	Electrónica

Ejemplo: ¿Qué alumnos cursan la carrera de Computación?

$$\pi_{\text{nom\_alum}}(\sigma_{\text{carrera = "Computacion"}} (\sigma_{\text{ALUMNO.cve\_carr= CARRERA.cve\_carr}} (\text{ALUMNO X CARRERA})))$$

nom_alum
José
Luis

Unión

Es una operación binaria, con la cual podemos obtener todos las tuplas de las relaciones involucradas. Su sintaxies es la siguiente:

$$\text{Relación1} \cup \text{Relación2}$$

Resta

Es una operación binaria, con la cual obtenemos todos las tuplas que se encuentran en la primera relación, pero que no se encuentran en la segunda. Su sintaxies es la siguiente:

$$\text{Relación1} - \text{Relación2}$$

Intersección

Esta operación es adicional ya que se puede obtener de las operaciones anteriores. El resultado son aquellas tuplas que se encuentran en ambas relaciones.

$$\text{Relación1} \cap \text{Relación2}$$
$$=$$
$$\text{Relación1} - (\text{Relación1} - \text{Relación2})$$

Esta operación no es conmutativa: $A - B \neq B - A$

Las operaciones Union, Resta e Intersección, para que se puedan realizar deben cumplir con dos condiciones:

- Las relaciones involucradas deben tener el mismo múmero de atributos.
- Los dominios de los atributos de las relaciones involucradas deben ser los mismos.

VALORES NULOS

El símbolo **NULL** representa un valor no conocido en la base de datos, un valor que no existe.

Un valor nulo no lo representa un valor de cero o la cadena "", ya que estos valores si son conocidos, son cero y "".

Todas las comparaciones que impliquen valores nulos en las operaciones relacionales son falsas por definición.

Las operaciones aritmeticas que impliquen valores nulos, dan como resultado un valor desconocido, es decir NULL.

Llave primaria(PK)

Una **llave primaria** es un atributo o conjunto de atributos que identifican a las tuplas de una relación. Cada relación debe contar con una llave primaria.

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

Cuando la llave primaria esta formada por más de un atributo, se denomina **llave primaria compuesta**.

Una relación puede tener más de un atributo o combinación de atributos que pueden actuar como llave primaria. A cada una de estas combinaciones o atributos se les llama **llave candidato**.

Solamente puede existir una llave primaria por relación, las llaves candidatos restantes se denominan **llave alterna**.

Ejemplos:

EMPLEADO

numeroCta	nombre	rfc	direccion	salario	noDepto

La llave primaria es numeroCta
rfc es una llave candidato, por lo tanto también es una llave alterna

CUENTA

banco	noCta	saldo	fechaApertura

La llave primaria es banco, noCta, la cuál es una llave primaria compuesta

Llave foránea(FK)

Una llave foránea en una relación es un atributo o conjunto de atributos que forman la llave primaria de otra o la misma relación.

Las llaves foráneas permiten llevar a cabo operaciones binarias para obtener información de un conjunto de relaciones.

Los valores de los atributos que conforman una llave foránea deben corresponder a valores existentes en la llave foránea, o, tener valor nulo.

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 relación puede estar formada en parte por una llave foránea, en este caso, la llave foránea no puede tener valores nulos.

Ejemplos:

DEPARTAMENTO

noDepto	nombreDepto	**numeroCta**

EMPLEADO

numeroCta	nombre	rfc	direccion	salario	**noDepto**

numeroCta es llave foránea en DEPARTAMENTO
noDepto es llave foránea en EMPLEADO

BANCO

banco	nombre	direccion

CUENTA

banco	noCta	saldo	fechaApertura

La llave primaria de CUENTA esta formada en parte por una llave foránea

INTEGRIDAD

El concepto de Integridad es sencillo: los datos deben ser válidos.

Ejemplos:

- No pueden existir renglones repetidos en una tabla.

- El número de departamento de un empleado debe de pertenecer a un departamento válido.

- En una Base de Datos de Reservación de líneas aéreas, el número de reservaciones no debe ser mayor que el cupo en un vuelo.

- La edad de un empleado debe estar entre N$ 1,000 y N$ 8,000.

Se definen los siguientes tipos de Integridad:

Integridad de Entidades: cada tupla en una realción debe ser única.

Integridad de Dominio: cada valor de un atributo pertenece a un dominio previamente definido. Si la llave primaria es única, se garantiza que una tupla es única en una relación.

Integridad Referencial: cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor nulo.

Integridad del Negocio: Los datos de la Base de Datos deben cumplir con las reglas del negocio.

Integridad Referencial

Cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor nulo.

Para poder mantener esta relación se pueden considerar las siguientes reglas:

Para la llave foránea:

- Insert/Update

  - No permitir el insert/update si no existe una llave primaria asociada (RESTRICT).

  - Asignar valor nulo a la llave foránea sino existe la correspondiente llave primaria(NULLIFY).

  - Asignar valor default a la llave foránea sino existe la correspondiente llave primaria(DEFAULT).

  - Si la llave primaria asociada a la llave foránea no existe, dar de alta la tupla asociada a la llave primaria.

- Para Delete no se debe considerar acción alguna.

Para la llave primaria:

- Para Update:

  - Modificar todas las llaves foráneas asociadas (UPDATE CASCADE).

  - Evitar la modificación si existen llaves foráneas asociadas (RESTRICT).

- Para Delete:

  - Borrar todas las llaves foráneas asociadas (DELETE CASCADE).

  - Evitar el borrado si existen llaves foráneas asociadas (RESTRICT).

  - Asignar NULL a las llaves foráneas asociadas (NULLIFY).

  - Asignar un valor de default a las llaves foráneas asociadas (DEFAULT).

- Para Insert no se debe considerar acción alguna.

REGLAS DE CODD

pag. 208 DATABASE SYSTEMS

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

SQL (STRUCTURED QUERY LANGUAGE)

SQL
Structured Query Language

JUNIO 1998

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

SQL
Structure 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 como NULL

- 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:

UNIQUE	no permite valores duplicados en una columna
CLUSTER	fisicamente, los registros son ordenados en la misma secuencia que los indices

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 definierón 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	52[illegible]78
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?

13. Liste los empleados cuyos nombres comiencen con M.

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 se 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 peliculas 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?

9. ¿Qué empleado es el que gana más?

10. ¿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 ]
```