Bases de Datos

Parte 2: Modelo Relacional y el lenguaje SQL

Objetivo General

Introducir al estudiante en la implementación y consulta de bases de datos, utilizando el modelo relacional y el lenguaje SQL

Prof. Fray León Osorio Rivera

Introducción

Existen varios modelos para las bases de datos, es decir, diferentes maneras de organizar la información. Cada modelo tiene ventajas y desventajas. Las bases de datos relacionales preceden a los modelos jerárquico y en red, los cuales presentaban ciertas deficiencias como la poca flexibilidad para los cambios y un acceso más lento a la información. Las bases de datos relacionales, si bien se desarrollaron en su origen para funcionar en grandes sistemas de computadoras, han experimentado un notable auge dentro del campo de la microinformática. Una de las principales razones es que ha sido más sencilla la creación de sistemas gestores de bases de datos que soporten el modelo relacional.

TEMA 1 Conceptos básicos

El modelo de datos relacional representa la base de datos como un conjunto de tablas. La estructura de una **tabla** es la misma que se utiliza como producto del diagrama E-R.

Existe una correspondencia directa entre el concepto de tabla y el concepto matemático de una relación puesto que cada columna de una tabla representa una relación entre un conjunto de valores. De esta manera, una tabla es un conjunto de relaciones, lo cual se denota así :

$$\prod_{i=1}^{n} D_i$$

Donde n es el número de columnas de la tabla
D_i es el conjunto de valores para la columna i .

A cada línea de una tabla se le conoce como **tupla**. Para indicar un valor de un atributo en una tupla determinada se utiliza la notación :

$$t_i\,[\,atributo\,]$$

Donde i es el número de la tupla
atributo es el nombre del atributo al cual corresponden al valor

El esquema de una tabla se denotaría así :

nombre tabla = (atributo : Dominio , ...)

Ejemplo:

Para la siguiente tabla :

Tabla *Alumno*

Id Alumno	Apellidos	Nombres	Carnet	Nivel	Id Programa
1	GUTIERREZ PEREZ	ADRIANA MARIA	9019451	2	1
2	PEREZ CARDONA	JORGE ALONSO	9020678	3	1
3	GAVIRIA HENAO	CARLOS MARIO	9021023	2	2
4	CARDENAS LOPEZ	RAUL	9110345	4	1
5	OSORIO OSPINA	GLORIA ELENA	8620639	5	2
6	GALEANO RIVERA	JUAN CARLOS	8711608	6	1
7	LOPERA	FELIPE ANDRES	9011567	3	1
8	LEON CADAVID	PEDRO	9120345	2	2

Su esquema sería :

Alumno (Apellidos: cadena, Nombres : cadena, Carnet : cadena, Nivel: entero)

y una variable de tupla de la tabla sería :

t_6 (Carnet) = '8711608'

Lo cual quiere decir que el atributo ***Carnet*** en la tupla ***6*** equivale a la cadena '*8711608'*.

El usuario de un sistema de base de datos puede realizar consultas de estas tablas, inserción de nuevas tuplas o actualización de las ya existentes. La expresión de dichas operaciones se realiza mediante unos lenguajes conocidos como **Lenguajes de Consulta Relacional**, los cuales se verán más adelante.

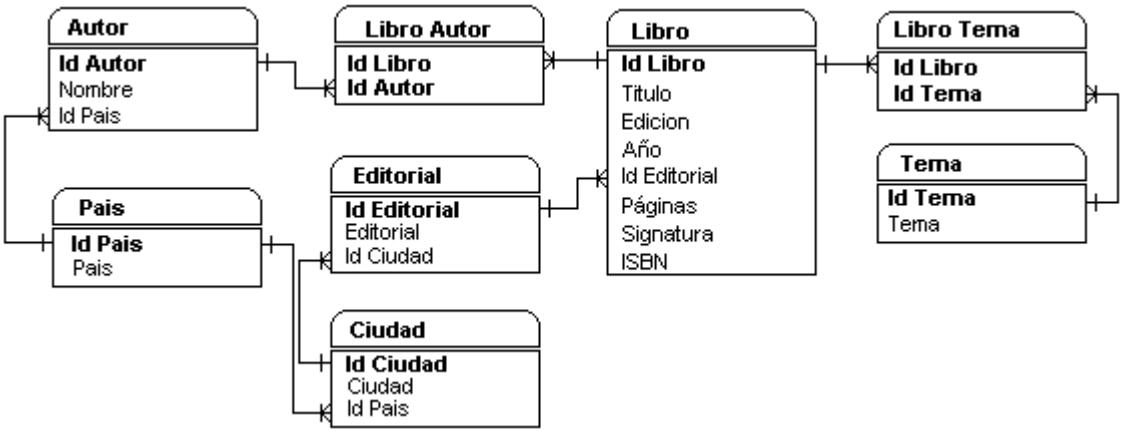
Modelo Relacional

Actualmente, todos los sistemas de gestión de bases de datos modernos almacenan y tratan la información utilizando el modelo **relacional**. El nombre relacional procede del hecho de que cada registro de la base de datos contiene información relacionada con un tema y sólo con ese tema. Además, los datos de dos clases de información (tal como libros y autores) pueden ser manipulados como una única entidad basada en los valores de los datos relacionados. Por ejemplo, sería una redundancia almacenar la información sobre el nombre y el país de un autor con todos los libros de su autoría. De modo que, en un sistema relacional, la información sobre los libros contiene algún dato que puede ser utilizado para enlazar cada libro con sus respectivos autores. En un **Sistema de Gestión de Base de Datos Relacionales** (DBMS) el sistema trata todos los datos en tablas. Cada **tabla** almacena información sobre un tema (como por ejemplo los autores) y dispone de unas columnas (**campos**) que contienen los diferentes tipos de información sobre ese tema (como por ejemplo el nombre) y de unas filas (**registros**) que describen todos los atributos de una única instancia del tema (por ejemplo los datos del autor "Gabriel García Márquez"). Incluso cuando se consulta la base de datos (**consulta**) para buscar información procedente de una o más tablas, el resultado es siempre algo semejante a otra tabla.

La forma como la información de una tabla está relacionada con la información de otra tabla se llamada **relación**. Por ejemplo, los países tienen una relación **uno a varios** con los autores ya que en de un país puede ser patria de muchos autores y una autor sólo tiene una patria. Los libros tienen una relación **varios a varios** con los autores ya que un autor lo puede ser de varios libros y un libro puede tener varios autores.

En resumen, el modelo de datos relacional representa una base de datos como un conjunto de tablas y una relaciones entre las tablas.

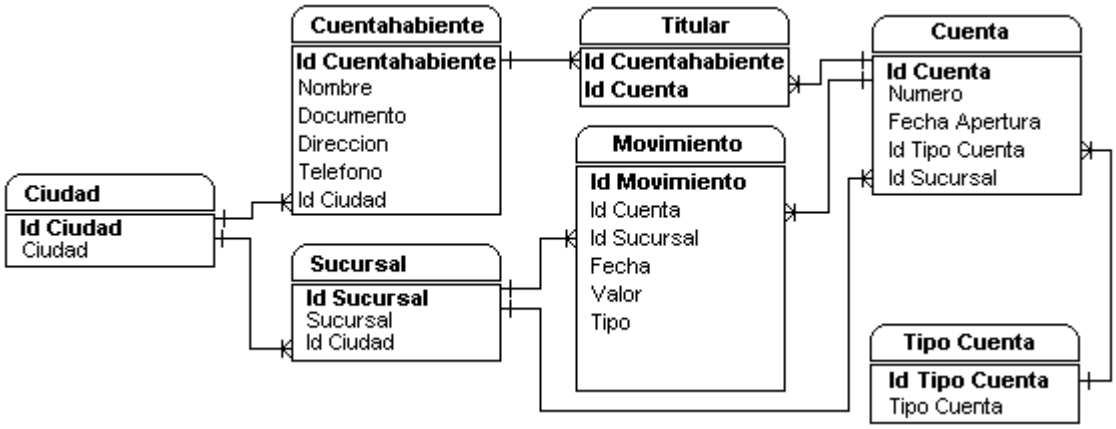
El siguiente diagrama relacional está diseñado para almacenar información sobre *bibliografías de libros*:



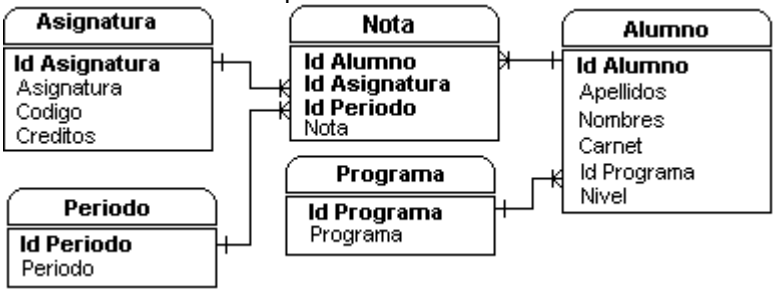
En esta ilustración, cada rectángulo representa la estructura de una tabla con su nombre en la parte superior y la lista de campos abajo. En negrilla aparecen los campos correspondientes a la **Llave primaria**, es decir, los campos que identifican cada registro en la tabla. Entre las tablas se aprecian unas líneas correspondientes a las relaciones, las cuales representan la dependencia que se da entre las tablas y que tienen asociada una **cardinalidad**:

Por ejemplo, se puede afirmar que “una editorial produce muchos libros” y que “un libro es producido por una editorial”.

El siguiente diagrama relacional está diseñado para almacenar información sobre *cuentas y movimientos bancarios*



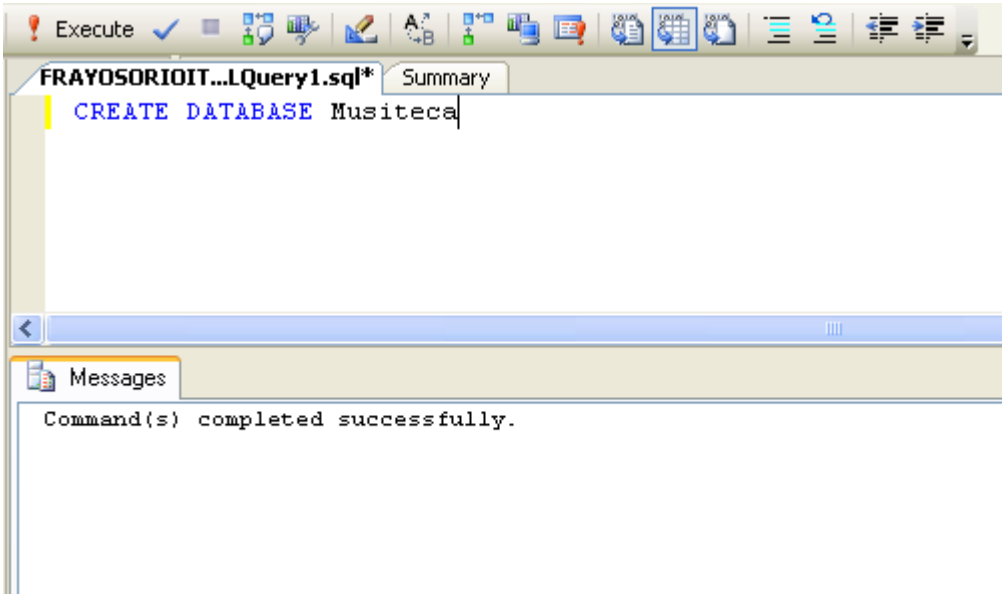
Recordemos el modelo para las *calificaciones de estudiantes universitarios*:



TEMA 2	Lenguaje de Definición de Datos (DDL por Data Definition Language)

El DDL es la parte de SQL que se usa para definir datos y objetos de una base de datos. Cuando se emplean estos comandos, se crean entradas al diccionario de datos del motor de base de datos.

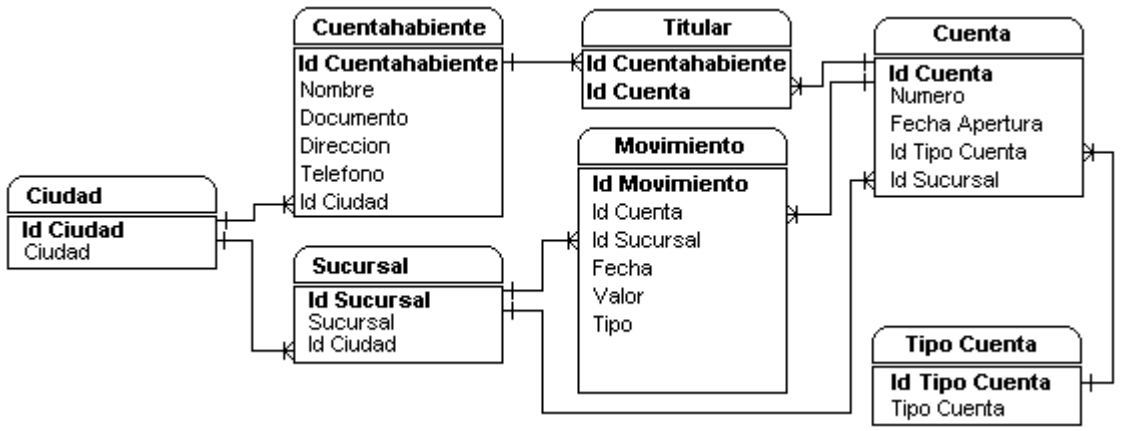
Para ejecutar estas instrucciones se debe utilizar un procesador de consultas del motor de base de datos. En el caso del *SQL Server de Microsoft* se tiene un editor de consultas el cual permite visualizar gráficamente su ejecución. Los resultados de una consulta se pueden visualizar en cuadrículas o con mensajes de ejecución.



Dentro de esta categoría del SQL (DDL) se incluyen los siguientes comandos del lenguaje *Transact-SQL*:

CREATE DATABASE

Descripción
Crea una nueva base de datos y los archivos utilizados para almacenar la base, o carga una base de datos a partir de los archivos de otra creada previamente.
Sintaxis
CREATE DATABASE nombre_base_de_datos **[ON (NAME = nombre_lógico, FILENAME = nombre_físico [, MAXSIZE = {tamaño|UNLIMITED}] [, FILEGROWTH = Incremento]) [, …]]** **[LOG ON NAME = nombre_lógico, FILENAME = nombre_físico [, MAXSIZE = {tamaño|UNLIMITED}] [, FILEGROWTH = Incremento]) [, …]]**
Ejemplo
Para los ejemplos de código DDL se utilizará el siguiente modelo relacional:



La instrucción para crear la base de datos sería

```
CREATE DATABASE Banco ON (NAME='Banco', FILENAME='E:\banco\banco.mdf')
```

ALTER DATABASE

Descripción

Modifica las definiciones de archivos y los ajustes de tamaño de una base de datos.

Sintaxis

```
ALTER DATABASE nombre_base_de_datos
    {ADD FILE (NAME= nombre_lógico, FILENAME = nombre_físico [, MAXSIZE =
    tamaño|UNLIMITED] [, FILEGROWTH = Incremento ]) [, …]
    | ADD LOG FILE (NAME = nombre_lógico, FILENAME = nombre_físico [, MAXSIZE =
    tamaño|UNLIMITED] [, FILEGROWTH = Incremento ]) [, …]
    | REMOVE FILE nombre_lógico
    | MODIFY FILE (NAME= nombre_lógico, FILENAME = nombre_físico [, MAXSIZE =
    tamaño|UNLIMITED] [, FILEGROWTH = Incremento ])
        }
```

Ejemplo

Se va a agregar un archivo de almacenamiento a la base de datos

```
ALTER DATABASE Banco ADD FILE (NAME='Banco1', FILENAME='E:\banco\banco1.mdf')
```

DROP DATABASE

Descripción

Elimina una base de datos. Sólo lo puede hacer el propietario de la base de datos y no puede estar abierta. Se recomienda antes de eliminar una base de datos sacar copia de seguridad.

Sintaxis

```
DROP DATABASE nombre_base_de_datos
```

Ejemplo

Si se quisiera eliminar la base de datos, esta sería la instrucción:

```
DROP DATABASE Banco
```

CREATE TABLE

Descripción

Crea una tabla en la actual base de datos.

Sintaxis

```
CREATE TABLE nombre_tabla (
    nombre_campo tipo (longitud) [NULL|NOT NULL] IDENTITY [(semilla, incremento)] [,…]
    )
```

Ejemplo

Las siguientes instrucciones agregan las tablas a la base de datos

```
CREATE TABLE Ciudad (
        [Id Ciudad] int NOT NULL ,
        [Nombre Ciudad] nvarchar (30) NULL )

CREATE TABLE Cuentahabiente (
        [Id Cuentahabiente] int NOT NULL ,
        [Nombre] nvarchar (50) NULL ,
        [Documento] nvarchar (50) NULL ,
        [Direccion] nvarchar (50) NULL ,
        [Telefono] nvarchar (50) NULL ,
        [Id Ciudad] int NULL )

CREATE TABLE [Tipo Cuenta] (
        [Id Tipo Cuenta] int NOT NULL ,
        [Tipo Cuenta] nvarchar (50) NULL )

CREATE TABLE Cuenta (
        [Id Cuenta] int NOT NULL ,
        [Numero] nvarchar (50) NOT NULL ,
        [Fecha Apertura] smalldatetime NULL ,
        [Id Tipo Cuenta] int NULL ,
        [Id Sucursal] int NULL )

CREATE TABLE Sucursal (
        [Id Sucursal] int NOT NULL ,
        [Nombre Sucursal] nvarchar (30) NULL ,
        [Id Ciudad] int NULL )

CREATE TABLE Movimiento (
        [Id Movimiento] int NOT NULL ,
        [Id Cuenta] int NULL ,
        [Id Sucursal] int NULL ,
        [Fecha] smalldatetime NULL ,
        [Valor] int NULL ,
        [Tipo] int NULL )

CREATE TABLE Titular (
        [Id Cuenta] int NOT NULL ,
        [Id Cuentahabiente] int NOT NULL )
```

ALTER TABLE

Descripción

Modifica la definición de una tabla para:

- Adicionar o eliminar campos
- Adicionar o eliminar restricciones (como por ejemplo claves primarias y foráneas).

- Habilitar o deshabilitar restricciones

Sintaxis

```
ALTER TABLE nombre_tabla
    [WITH CHECK|NOCHECK]
    {ALTER COLUMN nombre_campo tipo (longitud) [NULL|NOT NULL] IDENTITY [(semilla,
    incremento)] [,…]
    | ADD nombre_campo tipo (longitud) [NULL|NOT NULL] IDENTITY [(semilla, incremento)]
    [,…]
    | ADD CONSTRAINT nombre_restricción [PRIMARY KEY|FOREIGN KEY]
    CLUSTERED|NONCLUSTERED
    (nombre_campo [,…])
    [REFERENCES nombre_tabla_referida (nombre_campo [,…])]
    | DROP CONSTRAINT nombre_restricción
    | DROP nombre_campo
    }
```

Ejemplo

La siguiente instrucción modifica la longitud del campo [Nombre Sucursal] en la tabla [Sucursal]

```
ALTER TABLE Sucursal
        ALTER COLUMN [Nombre Sucursal] nvarchar (30)
```

Las siguientes instrucciones definen las claves primarias de las tablas:

```
ALTER TABLE Ciudad
        WITH NOCHECK
        ADD CONSTRAINT [PK_Ciudad] PRIMARY KEY NONCLUSTERED ([Id Ciudad])

ALTER TABLE [Cuentahabiente]
        WITH NOCHECK
        ADD CONSTRAINT [PK_Cuentahabiente] PRIMARY KEY NONCLUSTERED ([Id
Cuentahabiente])

ALTER TABLE [Tipo Cuenta]
        WITH NOCHECK
        ADD CONSTRAINT [PK_Tipo Cuenta] PRIMARY KEY NONCLUSTERED ([Id Tipo Cuenta])

ALTER TABLE Cuenta
        WITH NOCHECK
        ADD CONSTRAINT [PK_Cuenta] PRIMARY KEY NONCLUSTERED ([Id Cuenta])

ALTER TABLE Sucursal
        WITH NOCHECK
        ADD CONSTRAINT [PK_Sucursal] PRIMARY KEY  NONCLUSTERED ([Id Sucursal])

ALTER TABLE Movimiento
        WITH NOCHECK
        ADD CONSTRAINT [PK_Movimiento] PRIMARY KEY NONCLUSTERED ([Id Movimiento])

ALTER TABLE Titular
        WITH NOCHECK
        ADD CONSTRAINT [PK_Titular] PRIMARY KEY  NONCLUSTERED ([Id Cuenta], [Id
Cuentahabiente])
```

Las siguientes instrucciones definen las claves foráneas de las tablas, es decir las relaciones entre las tablas:

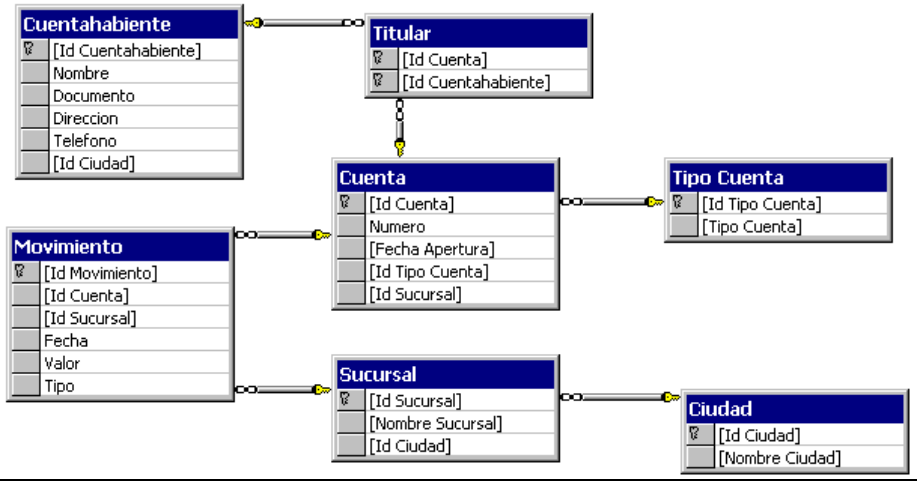
```
ALTER TABLE Cuenta
        ADD CONSTRAINT [FK_Cuenta_Tipo Cuenta] FOREIGN KEY ([Id Tipo Cuenta])
        REFERENCES [Tipo Cuenta] ([Id Tipo Cuenta])

ALTER TABLE Sucursal
        ADD CONSTRAINT [FK_Sucursal_Ciudad] FOREIGN KEY ([Id Ciudad])
        REFERENCES Ciudad ([Id Ciudad])

ALTER TABLE Movimiento
        ADD CONSTRAINT [FK_Movimiento_Cuenta] FOREIGN KEY ([Id Cuenta])
        REFERENCES Cuenta ([Id Cuenta]),
        CONSTRAINT [FK_Movimiento_Sucursal] FOREIGN KEY ([Id Sucursal])
        REFERENCES Sucursal ([Id Sucursal])

ALTER TABLE Titular
        ADD CONSTRAINT [FK_Titular_Cuenta] FOREIGN KEY ([Id Cuenta])
        REFERENCES Cuenta ([Id Cuenta]),
        CONSTRAINT [FK_Titular_Cuentahabiente] FOREIGN KEY ([Id Cuentahabiente])
        REFERENCES Cuentahabiente ([Id Cuentahabiente])
```

Después de estas instrucciones se tiene el siguiente diagrama (Que puede ser observado mediante el *Management Studio* del *SQL Server*)



DROP TABLE

Descripción
Elimina la definición de una tabla y todos los datos, índices, triggers, restricciones y especificaciones de permisos para la tabla. Cualquier vista sobre la tabla eliminada debe eliminarse usando la instrucción DROP VIEW.
Sintaxis
DROP TABLE nombre_tabla
Ejemplo

Si se quisiera eliminar la tabla [Ciudad]:

```
DROP TABLE Ciudad
```

CREATE INDEX

Descripción

Crea un índice en una tabla dada. Esta instrucción o bien cambia el ordenamiento físico de la tabla, o proporciona al optimizador un ordenamiento lógico de la tabla a fin de aumentar la eficiencia de las consultas.

Sintaxis

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED]
    INDEX nombre_indice ON nombre_tabla (nombre_campo [,...])
```

Ejemplo

Las siguientes instrucciones crean los índices de las tablas:

```
CREATE INDEX [IX_Ciudad] ON Ciudad ([Nombre Ciudad])

CREATE INDEX [IX_Cuentahabiente] ON Cuentahabiente([Nombre])

CREATE INDEX [IX_Cuentahabiente_Documento] ON Cuentahabiente ([Documento])

CREATE INDEX [IX_Tipo Cuenta] ON [Tipo Cuenta] ([Tipo Cuenta])

CREATE INDEX [IX_Cuenta] ON Cuenta ([Numero])

CREATE INDEX [IX_Sucursal] ON Sucursal ([Nombre Sucursal])
```

DROP INDEX

Descripción

Elimina uno o más índices de la bases de datos en uso. No aplica a índices creados mediante restricciones de llave primaria.

Sintaxis

```
DROP INDEX nombre_tabla.nombre_indice
```

Ejemplo

Las siguientes instrucciones eliminan los índices de [IX_Cuentahabiente_Documento], [IX_Tipo Cuenta] y [IX_Cuenta] los cuales deben tener valores únicos, razón por la cual vuelven a ser creados:

```
DROP INDEX Cuentahabiente.[IX_Cuentahabiente_Documento], [Tipo Cuenta].[IX_Tipo Cuenta],
Cuenta.[IX_Cuenta]

CREATE UNIQUE INDEX [IX_Cuentahabiente_Documento] ON Cuentahabiente ([Documento])

CREATE UNIQUE INDEX [IX_Tipo Cuenta] ON [Tipo Cuenta] ([Tipo Cuenta])

CREATE UNIQUE INDEX [IX_Cuenta] ON Cuenta ([Numero])
```

GRANT

Descripción

Crea una entrada en el sistema de seguridad que le permite a un usuario de la base de datos en uso trabajar con datos o ejecutar instrucciones Transact-SQL específicas..

Sintaxis

```
GRANT {ALL|permiso [,...]}
```

```
ON {nombre_tabla|nombre_vista|procedimiento_almacenado}
TO cuenta [,...]
```

Ejemplo

La siguiente instrucción concede a la cuenta 'Cajero' el permiso de consultar la tabla [Tipo Cuenta]

```
GRANT SELECT ON [Tipo Cuenta] TO Cajero
```

EXISTS

Descripción

Especifica un subconsulta para probar la existencia de registros.

Sintaxis

```
EXISTS (subconsulta)
```

Ejemplo

El siguiente código Transact-SQL elimina el índice [IX_Cuenta] de la tabla [Cuenta] siempre y cuando el índice exista:

```
IF EXISTS(SELECT name FROM sysindexes WHERE name=' IX_Cuenta')
        DROP INDEX [Cuenta]. [IX_Cuenta]
```

CREATE VIEW

Descripción

Crea una tabla virtual que representa una manera alternativa e mirar los datos en una o varias tablas. Las vistas pueden utilizarse como mecanismos de seguridad que otorgan permiso en una vista, pero no en las tablas implicadas de las bases de datos.

Sintaxis

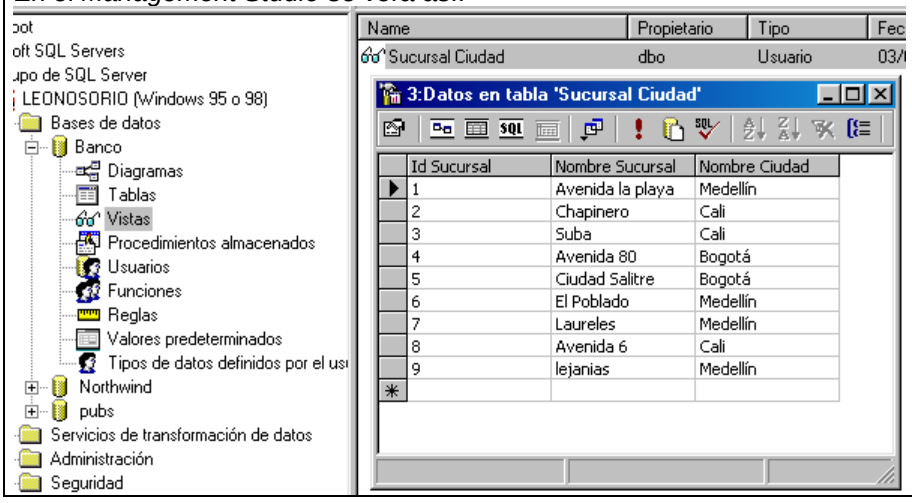
```
CREATE VIEW nombre_vista
    AS instrucción_select
```

Ejemplo

La siguiente vista define una consulta de todas las sucursales con sus respectivas ciudades:

```
CREATE VIEW [Sucursal Ciudad]
        AS SELECT [Id Sucursal], [Nombre Sucursal], [Nombre Ciudad]
        FROM Sucursal, Ciudad
        WHERE Sucursal.[Id Ciudad]=Ciudad.[Id Ciudad]
```

En el *Management Studio* se verá así:



ALTER VIEW

Descripción
Modifica una vista creada con la instrucción CREATE VIEW.
Sintaxis
ALTER VIEW nombre_vista **AS instrucción_select**
Ejemplo
A la anterior vista le agregaremos el campo [Id Ciudad] **ALTER VIEW [Sucursal Ciudad]** **AS SELECT [Id Sucursal], [Nombre Sucursal], [Nombre Ciudad], Ciudad.[Id Ciudad]** **FROM Sucursal, Ciudad** **WHERE Sucursal.[Id Ciudad]=Ciudad.[Id Ciudad]**

DROP VIEW

Descripción
Elimina una o más vistas.
Sintaxis
DROP VIEW nombre_vista [,...]
Ejemplo
La siguiente instrucción elimina la vista [Sucursal Ciudad] **DROP VIEW [Sucursal Ciudad]**

TEMA 3 | Lenguajes de Consulta Formales

Un **lenguaje de Consulta Formal** sirve para que el usuario solicite información de las bases de datos. Estos lenguajes son normalmente de alto nivel, mayor que los lenguajes estándar de programación.

Los lenguajes de consulta pueden ser **procedimentales** o **no procedimentales**. En un lenguaje procedimental el usuario debe especificar las operaciones necesarias para obtener un resultado, mientras que en uno no procedimental, el usuario describe la información que desea sin indicar procedimiento alguno para obtenerla. La mayor parte de los sistemas de bases de datos ofrecen lenguajes de consulta que incluye elementos de los dos enfoques: procedimental y no procedimental.

Álgebra Relacional

El **álgebra relacional** es un lenguaje de consulta procedimental. Comprende cinco operaciones fundamentales que son : **elegir, proyectar, producto cartesiano, unión y diferencia de conjuntos**. Todas ellas producen como resultado una nueva tabla.

Además de estas operaciones, existen otras que se definen en términos de las operaciones fundamentales por lo cual simplifican la notación. Son : **intersección de conjuntos, producto theta, producto natural y división**.

1. Operaciones Fundamentales.

Las operaciones **elegir** y **proyectar** se denominan operaciones **unarias** porque sólo actúan sobre una tabla. Las otras 3 operaciones operan sobre parejas de tablas.

1.1. La operación **Elegir** permite extraer las tuplas que satisfagan cierto predicado y se denota :

$\sigma_{predicado}$ (*tabla*)

Donde σ es el operador elegir.
predicado es una expresión dada en términos de comparaciones con atributos utilizando los **operadores =, <>, <, <=, >, >=** y los **conectivos lógicos** $\vee$ y $\wedge$.
tabla es la tabla de la cual se extraerán las tuplas.

Ejemplo :
Con base en la tabla ***Alumno*** obtener las tuplas de los estudiantes que están en el segundo Nivel.

$\sigma_{nivel = 2}$ (*Alumno*)

El resultado sería :

Tabla *Alumno*

Id Alumno	Apellidos	Nombres	Carnet	Nivel	Id Programa
1	GUTIERREZ PEREZ	ADRIANA MARIA	9019451	2	1
3	GAVIRIA HENAO	CARLOS MARIO	9021023	2	2
8	LEON CADAVID	PEDRO	9120345	2	2

1.2. La operación **Proyectar** permite extraer de una tabla algunas de sus columnas y se denota :

$\Pi_{lista\ de\ atributos}$ (*Tabla*)

Ejemplo :
Para la anterior consulta, supóngase que sólo se desea conocer el atributo *carnet*.

Π_{Carnet} ($\sigma_{nivel = 2}$ (*Alumno*))

El resultado sería :

Carnet
9019451
9021023
9120345

1.3. La operación **Producto Cartesiano** obtiene la combinación de dos tablas y se denota :

tabla 1 X ***tabla 2***

Si *tabla 1* contiene *n1* tuplas y *tabla 2* contiene *n2,* el resultado será *n1 por* n2 tuplas, producto de la combinación de cada tupla de <*tabla 1*> con todas las tuplas de <*tabla 2*>.

Ejemplo:
Se tienen las siguientes tablas:

Alumno

Id Alumno	Apellidos	Nombres	Carnet	Nivel	Id Programa
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2	1
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3	1
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3	1
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4	2
5	DIAZ HINESTROSA	ANA MARIA	8620639	8	2
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7	2

Programa

Id Programa	Programa
1	INGENIERIA CIVIL
2	CONTADURIA

Se desea obtener una tabla cuyas tuplas estén compuestas por el nombre del estudiante, el carnet y el programa académico en el que está matriculado.

En primer lugar obtendremos el producto cartesiano de las tablas ***Alumno*** y ***Programa***:

Alumno X Programa

Alumno. Id Alumno	Alumno. Apellidos	Alumno. Nombres	Alumno. Carnet	Alumno. Nivel	Alumno. Id Programa	Programa. Id Programa	Programa. Programa
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2	1	1	INGENIERIA CIVIL
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2	1	2	CONTADURIA
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3	1	1	INGENIERIA CIVIL
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3	1	2	CONTADURIA
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3	1	1	INGENIERIA CIVIL
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3	1	2	CONTADURIA
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4	2	1	INGENIERIA CIVIL
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4	2	2	CONTADURIA
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8	2	1	INGENIERIA CIVIL
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8	2	2	CONTADURIA
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7	2	1	INGENIERIA CIVIL
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7	2	2	CONTADURIA

A la tabla obtenida mediante este producto cartesiano le extraemos las tuplas que cumplan con la condición ***Alumno.Id programa = Programa..Id programa***:

$\sigma_{\textit{Alumno.Id Programa = Programa.Id Programa}}$(***Alumno X Programa***)

Alumno. Id Alumno	Alumno. Apellidos	Alumno. Nombres	Alumno. Carnet	Alumno. Nivel	Alumno. Id Programa	Programa. Id Programa	Programa. Programa
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2	1	1	INGENIERIA CIVIL
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3	1	1	INGENIERIA CIVIL
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3	1	1	INGENIERIA CIVIL
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4	2	2	CONTADURIA
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8	2	2	CONTADURIA
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7	2	2	CONTADURIA

Por último, se seleccionan los atributos solicitados, y la expresión definitiva sería :

$\Pi_{\textit{Alumno.Nombres, Alumno.Apellidos, Alumno-Carnet, Programa. Programa}}$
($\sigma_{\textit{Alumno.Id Programa = Programa.Id Programa}}$ (***Alumno X Programa***)

Alumno. Apellidos	Alumno. Nombres	Alumno. Carnet	Programa. Programa
GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	INGENIERIA CIVIL
MENESES AGUIRRE	OLGA LUCIA	9020678	INGENIERIA CIVIL
TOBON VELASQUEZ	ELKIN DARIO	9021023	INGENIERIA CIVIL
MUNOZ PEREZ	JORGE ELIECER	9110345	CONTADURIA
DÍAZ HINESTROSA	ANA MARIA	8620639	CONTADURIA
IDARRAGA GOMEZ	JAIME ELADIO	8711608	CONTADURIA

1.4. La operación **Unión** recopila en una tabla las tuplas que aparezcan en dos tablas. Se debe ser cuidadoso de que las tablas involucradas sean compatibles. Por tanto, para que una unión sea legal se requiere que :

- Las tablas involucradas deben tener el mismo número de atributos.
- Los dominios de los atributos *i-ésimo* de cada una de las relaciones deben ser los mismos.

Se denota :

<tabla 1>* U *<tabla 2>

Ejemplo:

En una base de datos de los empleados de una empresa se tienen las siguientes tablas:

Tabla *Personal*

Cedula	Nombre
43'520.356	GERARDO MOLINA
71'890.567	GABRIEL JAIME SALAZAR
98'789.890	GONZALO ANGULO
70'345.278	JAVIER BOTERO

Tabla *Cesantias*

Cedula	Fecha - Inicial	Fecha-Final	Valor
43'520.356	01/05/92	01/12/95	1'000.000
71'890.567	15/09/93	15/05/95	550.000

Tabla *Vacaciones*

Cedula	Fecha - Inicial	Fecha-Final	Valor
43'520.356	15/05/93	30/05/93	100.000
43'520.356	01/06/94	15/06/94	125.000
43'520.356	01/05/95	15/05/95	175.000
71'890.567	01/10/94	15/10/94	300.000
71'890.567	01/10/95	15/10/94	379.000
98'789.890	15/08/95	30/08/95	200.000

Se desea obtener una tabla con cedula, nombre y valores pagados a los empleados por concepto de cesantías y vacaciones en la empresa.

Como se puede observar, las tablas ***Cesantias*** y ***Vacaciones*** son compatibles por lo que se puede realizar una unión entre ellas para atender la solicitud.

Cesantias* U *Vacaciones

Cedula	Fecha - Inicial	Fecha-Final	Valor
43'520.356	01/05/92	01/12/95	1'000.000
71'890.567	15/09/93	15/05/95	550.000
43'520.356	15/05/93	30/05/93	100.000
43'520.356	01/06/94	15/06/94	125.000
43'520.356	01/05/95	15/05/95	175.000
71'890.567	01/10/94	15/10/94	300.000
71'890.567	01/10/95	15/10/94	379.000
98'789.890	15/08/95	30/08/95	200.000

Ahora podemos obtener el producto cartesiano de esta unión con la tabla ***personal***.

***Personal* X (*Cesantias* U *Vacaciones*)**

Personal. Cedula	Personal. Nombre	Vacaciones U Cesantias. Cedula	Vacaciones U Cesantias. Fecha - Inicial	Vacaciones U Cesantias. Fecha-Final	Vacaciones U Cesantias. Valor
43'520.356	GERARDO MOLINA	43'520.356	01/05/92	01/12/95	1'000.000
43'520.356	GERARDO MOLINA	71'890.567	15/09/93	15/05/95	550.000
43'520.356	GERARDO MOLINA	43'520.356	15/05/93	30/05/93	100.000
43'520.356	GERARDO MOLINA	43'520.356	01/06/94	15/06/94	125.000
43'520.356	GERARDO MOLINA	43'520.356	01/05/95	15/05/95	175.000
43'520.356	GERARDO MOLINA	71'890.567	01/10/94	15/10/94	300.000
43'520.356	GERARDO MOLINA	71'890.567	01/10/95	15/10/94	379.000
43'520.356	GERARDO MOLINA	98'789.890	15/08/95	30/08/95	200.000
71'890.567	GABRIEL JAIME SALAZAR	43'520.356	01/05/92	01/12/95	1'000.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	15/09/93	15/05/95	550.000
71'890.567	GABRIEL JAIME SALAZAR	43'520.356	15/05/93	30/05/93	100.000
71'890.567	GABRIEL JAIME SALAZAR	43'520.356	01/06/94	15/06/94	125.000
71'890.567	GABRIEL JAIME SALAZAR	43'520.356	01/05/95	15/05/95	175.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	01/10/94	15/10/94	300.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	01/10/95	15/10/94	379.000
71'890.567	GABRIEL JAIME SALAZAR	98'789.890	15/08/95	30/08/95	200.000
98'789.890	GONZALO ANGULO	43'520.356	01/05/92	01/12/95	1'000.000
98'789.890	GONZALO ANGULO	71'890.567	15/09/93	15/05/95	550.000
98'789.890	GONZALO ANGULO	43'520.356	15/05/93	30/05/93	100.000
98'789.890	GONZALO ANGULO	43'520.356	01/06/94	15/06/94	125.000
98'789.890	GONZALO ANGULO	43'520.356	01/05/95	15/05/95	175.000
98'789.890	GONZALO ANGULO	71'890.567	01/10/94	15/10/94	300.000
98'789.890	GONZALO ANGULO	71'890.567	01/10/95	15/10/94	379.000
98'789.890	GONZALO ANGULO	98'789.890	15/08/95	30/08/95	200.000
70'345.278	JAVIER BOTERO	43'520.356	01/05/92	01/12/95	1'000.000
70'345.278	JAVIER BOTERO	71'890.567	15/09/93	15/05/95	550.000
70'345.278	JAVIER BOTERO	43'520.356	15/05/93	30/05/93	100.000
70'345.278	JAVIER BOTERO	43'520.356	01/06/94	15/06/94	125.000
70'345.278	JAVIER BOTERO	43'520.356	01/05/95	15/05/95	175.000
70'345.278	JAVIER BOTERO	71'890.567	01/10/94	15/10/94	300.000
70'345.278	JAVIER BOTERO	71'890.567	01/10/95	15/10/94	379.000
70'345.278	JAVIER BOTERO	98'789.890	15/08/95	30/08/95	200.000

De este producto extraemos las tuplas que cumplan la condición ***Personal.Cedula = Vacaciones U Cesantias.Cedula***, es decir, las vacaciones y cesantías que realmente le correspondan a un empleado :

$\sigma_{Personal.cedula = Vacaciones\text{-}Cesantias.cedula}$ (***Personal*** X (***Cesantias*** U ***Vacaciones***))

Personal. Cedula	Personal. Nombre	Vacaciones U Cesantias. Cedula	Vacaciones U Cesantias. Fecha-Inicial	Vacaciones U Cesantias. Fecha-Final	Vacaciones U Cesantias. Valor
43'520.356	GERARDO MOLINA	43'520.356	01/05/92	01/12/95	1'000.000
43'520.356	GERARDO MOLINA	43'520.356	15/05/93	30/05/93	100.000
43'520.356	GERARDO MOLINA	43'520.356	01/06/94	15/06/94	125.000
43'520.356	GERARDO MOLINA	43'520.356	01/05/95	15/05/95	175.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	15/09/93	15/05/95	550.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	01/10/94	15/10/94	300.000
71'890.567	GABRIEL JAIME SALAZAR	71'890.567	01/10/95	15/10/94	379.000
98'789.890	GONZALO ANGULO	98'789.890	15/08/95	30/08/95	200.000

Por último, seleccionamos los atributos ***Personal.Cedula***, ***Personal.Nombre*** y ***Vacaciones U Cesantias.Valor*** :

$\Pi_{Personal.Cedula,\ Personal.Nombre,\ Vacaciones\text{-}Cesantias.Valor}$
($\sigma_{Personal.cedula = Vacaciones\text{-}Cesantias.cedula}$ (***Personal*** X (***Cesantias*** U ***Vacaciones***)))

Personal. Cedula	Personal. Nombre	Vacaciones U Cesantias. Valor
43'520.356	GERARDO MOLINA	1'000.000
43'520.356	GERARDO MOLINA	100.000
43'520.356	GERARDO MOLINA	125.000
43'520.356	GERARDO MOLINA	175.000
71'890.567	GABRIEL JAIME SALAZAR	550.000
71'890.567	GABRIEL JAIME SALAZAR	300.000
71'890.567	GABRIEL JAIME SALAZAR	379.000
98'789.890	GONZALO ANGULO	200.000

1.5. La operación **Diferencia** permite encontrar las tuplas que estén en una relación pero no en otra. Se denota :

<tabla 1> - <tabla 2>

Ejemplo:

En el ejemplo anterior, para hallar cedula y nombre de los empleados que hayan recibido vacaciones pero no cesantías, tenemos :

Para ello seleccionamos en primera instancia, el atributo ***Cedula*** en las tablas ***Vacaciones*** y ***Cesantias*** y aplicamos diferencia de conjuntos :

$\Pi_{Vacaciones.Cedula}$ (***Vacaciones***) – $\Pi_{Cesantias.Cedula}$ (***Cesantias***)

Cedula
43'520.356
43'520.356
43'520.356
71'890.567
71'890.567
98'789.890

Cedula
43'520.356
71'890.567

⟹

Cedula
98'789.890

Aplicamos ahora el producto cartesiano de esta diferencia con la tabla ***personal***

Personal X ($\Pi_{Vacaciones.Cedula}$ (***Vacaciones***) - $\Pi_{Cesantias.Cedula}$ (***Cesantias***))

Personal. Cedula	Personal. Nombre	Vacaciones - Cesantias Cedula
43'520.356	GERARDO MOLINA	98'789.890
71'890.567	GABRIEL JAIME SALAZAR	98'789.890
98'789.890	GONZALO ANGULO	98'789.890
70'345.278	JAVIER BOTERO	98'789.890

Por último, se extraen las tuplas donde ***Personal.cedula = Vacaciones-Cesantias.Cedula*** y se seleccionan los campos ***Personal.Cedula*** y ***Personal.Nombre***.

$\Pi_{Personal.Cedula,\ Personal.Nombre}\ (\sigma_{Personal.cedula\ =\ Vacaciones\text{-}Cesantias.cedula}\ (Personal\ X\ (\Pi_{Vacaciones.Cedula}\ (Vacaciones)\ -\ \Pi_{Cesantias.Cedula}\ (Cesantias))))$

Personal. Cedula	Personal. Nombre
98'789.890	GONZALO ANGULO

2. Operaciones Adicionales.

Ya se han visto las cinco operaciones fundamentales del álgebra relacional **σ, Π, X, U** y **-** , las cuales son suficientes para expresar cualquier consulta. Sin embargo, algunas consultas comunes van a producir expresiones muy largas. Es por esto que se definen otros operadores, para simplificar las consultas. Cada uno de estos operadores tiene su respectiva expresión en operadores básicos.

2.1. La **Intersección** permite obtener las tuplas que están tanto en una tabla como en otra.

$$<tabla\ 1> \cap <Tabla\ 2> = <tabla\ 1> - (<tabla\ 1> - <tabla\ 2>)$$

2.2. El **Producto Theta** es una operación binaria que permite combinar la elección y el producto cartesiano en una sola operación.

$$<tabla\ 1> \Theta_{<predicado>} <Tabla\ 2> = \sigma_{<predicado>} (<tabla\ 1> X <tabla\ 2>)$$

2.3. El **Producto Natural** es una operación binaria une dos esquemas de dos tablas y sólo tiene en cuenta aquellas parejas de tuplas que tengan igual valor en algún atributo en común.

$$<tabla\ 1>\ x\ <Tabla\ 2> = \Pi_{<tabla\ 1>\ U <Tabla\ 2>} (<tabla\ 1> \Theta_{(<tabla\ 1>.<atributo\ 1> = <tabla\ 2>.<atributo\ 1> \ldots <tabla\ 1>.<atributo\ n> = <tabla\ 2>.<atributo\ n>)} <tabla\ 2>)$$

Donde {*<atributo 1>,...,<atributo n>* } son los atributos comunes a ambas tablas.

2.4. La **División** es una operación binaria que extrae de una tabla los atributos que no tiene otra tabla.

$$<tabla\ 1> \div <Tabla\ 2> = \Pi_{<tabla\ 1> - <tabla\ 2>} (<tabla\ 1>) - \Pi_{<tabla\ 1> - <tabla\ 2>} ((\Pi_{<tabla\ 1> - <tabla\ 2>} (<tabla\ 1>)\ X\ <tabla\ 2>) - <tabla\ 1>$$

TEMA 4 | Lenguajes de Consulta Comerciales

Los sistemas comerciales de bases de datos ofrecen un lenguaje de consulta más "amigable con el usuario", eliminando el carácter simbólico de los lenguajes formales. Estos lenguajes, además de consultas, ofrecen muchas otras funciones.

Se tratará aquí, de una manera no exhaustiva, el lenguaje **SQL**. Este lenguaje se introdujo como lenguaje de consulta del **Sistema R**. La sigla SQL significa ***Structured Query Language*** (*Lenguaje de Consulta Estructurado)*.

Consultas en SQL

1. Estructura Básica.

La estructura básica de una expresión de consulta en SQL se compone de tres cláusulas :

- La cláusula **Select** que corresponde a la operación de proyección del álgebra relacional. Sirve para listar los atributos que se desean en el resultado de una consulta.
- La cláusula **From** es una lista de las tablas que se van a examinar durante la ejecución de la expresión.
- La cláusula **Where** corresponde al predicado de selección del álgebra relacional. Se compone de un predicado que incluye atributos de las tablas que aparecen en la cláusula ***From***.

Una consulta común en SQL tiene la forma :

Select ***<atributo 1>,<atributo 2>,... ,<atributo n>***
from ***<tabla 1>,<tabla 2>,... ,<tabla n>***
Where ***<predicado>***

Esta consulta es equivalente a la expresión del álgebra relacional:

Π ***<atributo 1>,<atributo 2>,... ,<atributo n>***$(\sigma$***<predicado>***$($ ***<tabla 1>*** X ***<tabla 2>*** $X...X$ ***<tabla n>***$))$

El SQL forma el producto cartesiano de las tablas que se nombran en la cláusula ***From***; realiza una selección del álgebra relacional utilizando el predicado de la cláusula ***Where*** y proyecta el resultado a los atributos de la cláusula ***Select***.

Si se omite la cláusula ***where*** el predicado *<predicado>* es verdadero.
La lista de atributos *<atributo 1>,<atributo 2>,... ,<atributo n>* puede sustutuirse por un asterisco (*) para elegir todos los atributos de las tablas que aparecen en la cláusula ***From***.

El resultado de una consulta en SQL es una tabla.

Para la eliminación de duplicados de tuplas, se inserta la palabra reservada **distinct** después de ***Select*** :

Select distinct ***<atributo 1>,<atributo 2>,... ,<atributo n>***
from ***<tabla 1>,<tabla 2>,... ,<tabla n>***
Where ***<predicado>***

El SQL incluye las operaciones **Union** (Unión), **Intersect** (Intersección) y **Minus** (Diferencia) que operan sobre las tablas.

El SQL utiliza los operadores lógicos **and** (y), **or** (o) y **not** (no) en vez de los símbolos $\wedge$, $\vee$ y $\neg$.

Ejemplo:

Con las siguientes tablas:

Alumno

Id Alumno	Apellidos	Nombres	Carnet	Nivel
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7

Asignatura

Id Asignatura	Codigo	Asignatura	Creditos
1	MAT051	MATEMATICAS OPERATIVAS	4
2	MAT020	MATEMATICAS DISCRETAS	4
3	MAT021	GEOMETRIA VECTORIAL	4
4	MAT025	CALCULO I	6
5	MAT026	CALCULO II	4

Nota

Id Alumno	Id Asignatura	Id Periodo	Calificacion
1	1	1	**3,5**
1	3	1	**4,5**
1	4	2	**5,0**
2	1	1	**2,5**
2	3	1	**3,0**
3	1	1	**4,0**
3	3	1	**2,7**
4	1	2	**3,8**
4	3	2	**3,2**
5	1	2	**3,0**
6	1	2	**4,0**

Periodo

Id Periodo	Periodo
1	1991 - 1
2	1991 - 2
3	1992 - 1

Se desea obtener el ***Carnet***, ***Apellidos***, ***Nombres***, ***Codigo***, ***Asignatura***, ***Calificacion*** y ***Periodo*** de los estudiantes que perdieron asignaturas.

En álgebra relacional esta consulta se expresa así:

$$\Pi_{Alumno.Nombre,\ Asignatura.Codigo\ Asignatura.Asignatura,\ Nota.Calificación,\ Periodo.Periodo}\ (\sigma_{Alumno.Id\ Alumno\ =\ Nota.Id\ Alumno\ \wedge\ Asignatura.Id\ Asignatura=Nota.Id\ Asignatura\ \wedge\ Periodo.Id\ Periodo=Nota.Id\ Periodo}\ (Alumno\ X\ Nota\ X\ Asignatura\ X\ Periodo))$$

En SQL, esta consulta quedaría expresa así utilizando variables de tupla:

```
Select Alumno.Carnet, Alumno.Apellidos, Alumno.Nombres, Asignatura.Codigo,
       Asignatura.Asignatura, Nota.Calificacion, Periodo.Periodo
       From Alumno, Asignatura, Nota, Periodo
       Where Alumno.Id Alumno = Nota.Id Alumno And S.Id Asignatura = Nota.Id Asignatura
       And Periodo.Id Periodo = Nota.Id Periodo And Nota.Calificacion < 3
```

Apellidos	Nombres	Carnet	Codigo	Asignatura	Calificacion	Periodo
MENESES AGUIRRE	OLGA LUCIA	9020678	MAT051	MATEMATICAS OPERATIVAS	**2,5**	1991 - 1
TOBON VELASQUEZ	ELKIN DARIO	9021023	MAT021	GEOMETRIA VECTORIAL	**2,7**	1991 - 1

2. Variables de Tupla.

SQL emplea el concepto de variables de tupla del cálculo relacional de tuplas. Una variable tupla en SQL debe estar asociada a una tabla determinada y se define en la cláusula ***From*** colocándola después del nombre de la tabla a la cual está asociada utilizando el parámetro ***As***.

Ejemplo:

La consulta anterior quedaría así:

```
Select T.Carnet, T.Apellidos, T.Nombres, S.Codigo, S.Asignatura, U.Calificación, W.Periodo
        From Alumno As T, Asignatura As S, Nota As U, Periodo As W
        Where T.Id Alumno = U.Id Alumno And S.Id Asignatura = U.Id Asignatura
        And W.Id Periodo = U.Id Periodo And U.Calificacion < 3
```

3. Operaciones de Conjunto.

SQL también permite los operadores :

```
> any, >= any, = any, <= any, < any y <> any
> all, >= all, = all, <= all, < all y <> all
in  y not in
contains y not contains
```

para comparar un atributo con un conjunto, donde el conjunto es un grupo de valores producidos por la cláusula ***Select***.

- La cláusula ***any*** quiere decir que por lo menos hay una tupla en el conjunto que cumpla con la condición.
- La cláusula ***all*** indica que todas las tuplas en el conjunto cumplen la condición.
- La cláusula ***in*** pregunta si hay pertenencia en el conjunto.
- La cláusula ***contains*** pregunta si un conjunto contiene a otro.

Ejemplo:

Para las tablas del ejemplo anterior, se requiere solo los nombres de quienes perdieron la asignaturas.

```
Select Nombres, Apellidos
  from Alumno
  Where Id Alumno in (  Select Id Alumno
                        from Nota
                        Where Calificacion < 3 )
```

Esta consulta se traduciría "*encontrar los nombres de los alumnos cuyo Id Alumno esté en el conjunto de Id Alumno de alumnos que perdieron asignaturas*".

4. Ordenamiento de consultas.

SQL ofrece al usuario cierto control sobre el orden en que se van a mostrar las tuplas. La cláusula **Order by** permite que las tuplas en el resultado aparezcan ordenadas por determinados atributos.

Ejemplo:

Supóngase que la consulta anterior se requiere ordenada por el nombre del estudiante.
La expresión SQL sería :

```
Select Nombres, Apellidos
  from Alumno
  Where Id Alumno in (  Select Id Alumno
                        from Nota
                        Where Calificacion < 3 )
  Order by Apellidos, Nombres
```

5. Funciones y Grupos en las consultas .

SQL incluye funciones para calcular :

- Promedio: **avg**
- Mínimo: **min**
- Máximo: **max**
- Total: **sum**
- numero de tuplas: **count**

Estas funciones se denominan **operaciones de agregados** ya que operan sobre tuplas.
Para operar estas funciones sobre grupos de tuplas se utiliza la cláusula **Group by**. El atributo que se da en esta cláusula sirve para formar grupos.

Ejemplo:
Con base en las tablas de la consulta anterior, se desea saber el promedio de ***Calificacion*** que obtuvo cada estudiante en una tabla que contenga además el ***Carnet***, *los* ***Apellidos*** y ***Nombres***, y además, esté ordenada por el ***Carnet***.
La expresión SQL sería :

```
Select Alumno.Carnet, Alumno.Apellidos, Alumno.Nombres, avg (Nota.Calificacion)
  From Alumno, Nota
  Where Alumno.Id Alumno= Nota.Id Alumno
  Group by Alumno.Carnet, Alumno.Apellidos, Alumno.Nombres
  Order by Alumno.Carnet
```

SQL emplea la cláusula **Having** para operar sobre grupos ya formados, pudiendo utilizar también los operadores de agregados.

Ejemplo:
Supóngase que en la consulta anterior se desea obtener solamente los alumnos que obtuvieron promedio menor a 3.
La expresión SQL sería :

```
Select Alumno.Carnet, Alumno.Apellidos, Alumno.Nombres, avg (Nota.Calificacion)
  From Alumno, Nota
  Where Alumno.Id Alumno= Nota.Id Alumno
  Group by Alumno.Carnet, Alumno.Apellidos, Alumno.Nombres
  Having avg(Nota.Calificacion) < 3
  Order by Alumno.Carnet
```

6. El predicado Exist.
Este predicado tiene como argumento una proposición ***Select*** y es verdadero a menos que el ***Select*** resulte en una tabla vacía.

Actualización en SQL

Hasta esta instancia sólo se ha hecho énfasis en la extracción de información de la base de datos, pero no se ha mencionado ninguna operación que permita añadir, modificar o eliminar información, operaciones conocidas como de **actualización**.

1. Inserción.

Para insertar datos en una tabla debe especificarse la tupla que se va a insertar o una consulta cuyo resultado sea el conjunto de tuplas que se insertará. Esto se denota:

```
Insert into <tabla>
    (<atributo 1>, < atributo 2>, ..., < atributo n>)
    values (<valor 1>, <valor 2>, ..., <valor n>)
```

ó

```
Insert into <tabla>
    Select ...
```

Ejemplo:

En la siguiente tabla de estudiantes

Alumno

Id Alumno	Apellidos	Nombres	Carnet	Nivel
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7

Se desean añadir los datos de *HECTOR DARIO GOMEZ* con carnet *9234567*, el cual se matricula para el primer semestre:

```
Insert into Alumno
    (Apellidos, Nombres, Carnet, Semestre)
    values ( 'GOMEZ', 'HECTOR DARIO', '9234567', 1)
```

Alumno

Id Alumno	Apellidos	Nombres	Carnet	Nivel
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4
5	DÍAZ HINESTROSA	ANA MARIA	8620639	8
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7
7	GOMEZ	HECTOR DARIO	9234567	1

2. Modificación.

Permite cambiar los valores de los atributos de todas o algunas tuplas de una tabla. Se denota :

```
Update <tabla>
    Set <atributo> = <expresion>
    where <predicado>
```

Ejemplo:

Supóngase que en la tabla de la consulta anterior se desea modificar el apellido del último alumno agregado:

Update *Alumno*
Set *Apellidos='GOMEZ CUARTAS'*
Where *[Id Alumno]=7*

Alumno

Id Alumno	Apellidos	Nombres	Carnet	Nivel
1	GUTIERREZ LOPEZ	ADRIANA MARIA	9019451	2
2	MENESES AGUIRRE	OLGA LUCIA	9020678	3
3	TOBON VELASQUEZ	ELKIN DARIO	9021023	3
4	MUNOZ PEREZ	JORGE ELIECER	9110345	4
5	DIAZ HINESTROSA	ANA MARIA	8620639	8
6	IDARRAGA GOMEZ	JAIME ELADIO	8711608	7
7	GOMEZ CUARTAS	HECTOR DARIO	9234567	1

3. Eliminación.

Funciona de manera similar a una consulta, sólo que el resultado es la desaparición de las tuplas elegidas:

Delete From *<tabla>*
where *<predicado>*

Es de anotar que este comando, como los anteriores, sólo operan sobre una tabla.

Ejemplo:

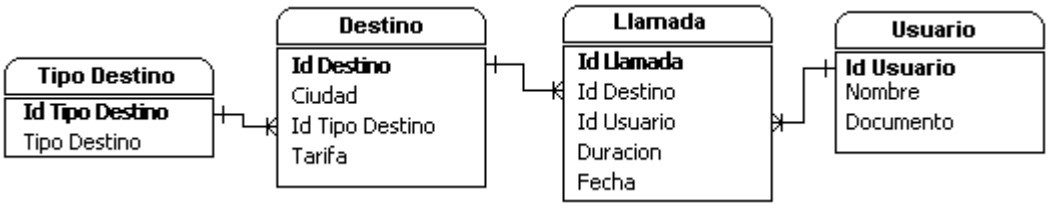
Para la tabla de la consulta anterior, eliminar los estudiantes que estén en nivel:

Delete From *Alumno*
Where *Nivel= 3*

TEMA 5 Ejercicios de aplicación

1. Para un **Sistema de Información Discográfico**, diseñar el respectivo modelo relacional
   Tenga en cuenta:
   - Los intérpretes con su respectivo tipo (Grupo, Cuarteto, Trío, Dueto, Solista, etc.)
   - Las canciones y sus distintas interpretaciones en los ritmos que haya sido interpretada (Por ejemplo la canción “Lluvia” es una “balada” interpretada por “Luis Angel” y es una “Salsa Romántica” interpretada por “Eddie Santiago”. Si la canción es interpretada en otro idioma se dice que es otra canción.
   - Las canciones tienen compositor.
   - Los álbumes son un conjunto de interpretaciones de canciones que pueden ser de uno o varios intérpretes y pueden venir en varios medios (Cassette, VHS, CD, DVD, etc.). A su vez, la interpretación en un álbum puede estar varias veces en distintas formatos (Pista, MP3, Video, etc.)

2. Con base en el siguiente diagrama relacional, el cual está diseñado para almacenar información sobre llamadas telefónicas



Codifique las respectivas instrucciones SQL para las siguientes consultas:

a. Total de llamadas hechas a cada destino.
b. Valor total de las llamadas hechas por cada usuario entre las *Fechas* **F1** y **F2**
c. Cambiar todas las llamadas hechas por el usuario con *Nombre* **N** por el usuario con *Id Usuario* igual a **X**