# CAPÍTULO 9º: SQL (MODIFICACIONES EN LA BASE DE DATOS)

SQL como lenguaje de programación en Bases de Datos, tiene dos formas de actuación: Como Lenguaje de Manipulación de Datos (DML) mediante el cual se consultan datos a la Base de Datos y se actualizan éstos, y como Lenguaje de Definición de los Datos (DDL) que permite definir y diseñar la estructura de los datos dentro de la Base de datos.

## 1 ***Actualizaciones en las Tablas***.-

Las actualizaciones de los datos de la Base se realizan por alguno de los tres caminos siguientes: Inserción (INSERT), Borrado (DELETE) o modificación (UPDATE).

### 1.1 ***Introducción de datos en una tabla.-***

Puede ser, a su vez, de tres formas distintas:

- *Inserción de una fila*

La expresión general de la sentencia de inserción de datos en una tabla es la siguiente:

INSERT INTO Tabla (Campo1, Campo2, Campo3, ... , CampoN)

VALUES (Valor1, Valor2, Valor3, ... , ValorN)

Conceptualmente, la sentencia INSERT construye una fila de datos que se corresponden con la estructura en columnas de la tabla. La nueva fila insertada no tiene por que ser la primera ni la última de la tabla. Es simplemente una fila mas de la tabla que el Sistema Gestor de la Base de Datos situará donde proceda.

Cuando SQL inserta una nueva fila en una tabla, automáticamente asigna el valor NULL a cualquier campo que no haya sido especificado en la lista de columnas de la sentencia INSERT. Así mismo, puede hacerse más explícita esta asignación de NULL a un campo incluyendo la columna correspondiente en la lista y especificando la palabra clave NULL en el valor correspondiente:

INSERT INTO Tabla (Campo1, Campo2, Campo3, ... , CampoN)

VALUES (Valor1, Valor2, NULL, ... , ValorN)

La sentencia anterior asigna el valor NULL al Campo3 en la nueva fila insertada

Debe tenerse en consideración que, si en la definición de un campo específico en una tabla se ha especificado que éste debe tener un valor no nulo, debe actualizarse explícitamente en la sentencia INSERT, ya que, de no hacerlo, el sistema produciría un error.

Así mismo, pueden utilizarse en la inserción constantes del sistema si el SQL utilizado lo permite:

INSERT INTO Tabla (Campo1, Campo2, Fecha3, ... , CampoN)

VALUES (Valor1, Valor2, CURRENT DATE, ... , ValorN)

La sentencia anterior introduce el valor de la fecha actual en el Campo Fecha3.

Cuando se omite la lista de columnas a actualizar SQL genera automáticamente una lista formada por todas las columnas de la tabla en secuencia de izquierda a derecha. Así:

INSERT INTO Tabla

VALUES (Valor1, Valor2, Valor3, ... , ValorN)

Actualiza todos los campos de la fila. Debe tenerse en cuenta que la palabra clave NULL debe ser utilizada en la lista de valores para asignar explícitamente valores NULL al campo que lo requiera y, además, la secuencia de valores debe corresponderse exactamente con la secuencia de columnas de la tabla.

- *Inserción multifila*

En este caso, los valores de datos para las nuevas filas a actualizar no son especificados explícitamente en el texto de la sentencia, en su lugar, la fuente de los datos de las nuevas filas es una consulta a la base de datos especificada en la sentencia:

INSERT INTO Tabla1 (Campo1, Campo2, Campo3, ... , CampoN)

SELECT Campo1, Campo2, Campo3, ... , CampoN

FROM Tabla2

WHERE Condición de Búsqueda.

Existen una serie de *restricciones lógicas* sobre la consulta que aparece dentro de una sentencia INSERT multifila, así, para SQL1:

- La consulta no puede contener la cláusula ORDER BY

- El resultado de la consulta debe contener el mismo número de columnas que hay en la lista de columnas de la sentencia INSERT (o que la tabla destino completa, si se ha omitido la lista de columnas) y los tipos de los datos deben ser compatibles columna a columna.

- La consulta no puede ser la UNION de varias sentencias SELECT diferentes (Esto es, únicamente puede especificarse una única sentencia SELECT)

- La tabla destino de la sentencia INSERT no puede aparecer en la cláusula FROM de la consulta, o de ninguna subconsulta que ésta contenga. Esto prohibe insertar parte de una tabla sobre sí misma.

SQL2 es, sin embargo más flexible en las dos últimas restricciones ya que permite en la consulta expresiones y operaciones de unión y composición además de permitir la autoinserción.

*- Inserción por carga masiva*

En general, todos los SGBD comerciales incluyen en sus prestaciones la capacidad de carga masiva de datos en una tabla procedentes de un archivo. El estándar SQL de ANSI/ISO no considera esta función y suele ser suministrada como un programa de utilidad autónomo (Importar datos) en lugar de formar parte del lenguaje SQL.

1.2 ***Supresión de datos en una tabla.-***

Una fila de datos se suprime de una tabla de una Base de Datos cuando la entidad representada por la fila desaparece del mundo exterior, esto es, la fila se suprime para mantener la Base de Datos como un modelo preciso del mundo real. La unidad más pequeña de datos que puede ser suprimida de una Base de Datos relacional es una única fila.

La sentencia DELETE elimina las filas seleccionadas de una tabla en función del criterio de selección indicado:

DELETE FROM Tabla

WHERE Criterio de Selección

La cláusula WHERE actúa en la sentencia DELETE de forma exactamente igual a como lo hace en la sentencia SELECT, por lo que las condiciones de búsqueda en ambos casos tienen el mismo formato. Como consecuencia de ello es preciso tener en consideración lo siguiente:

- La cláusula WHERE puede especificar una sola fila o un conjunto de filas. En ambos casos, al actuar la sentencia DELETE, la fila o filas seleccionadas serán eliminadas de la tabla.

- Si no se especifica la cláusula WHERE, esto es:

DELETE FROM Tabla

La sentencia DELETE borra todas las filas las filas de la tabla, elimina todos los datos pero mantiene la estructura de la tabla, es decir, desaparece el contenido de la tabla pero se mantiene ésta en la Base de Datos.

1.3 ***Modificación de datos en una tabla.-***

Tiene por fin actualizar los valores de la Base de Datos para mantener a ésta como un modelo preciso del mundo real. La unidad mínima que puede modificarse en una Base de Datos es una única columna de una única fila.

El formato general de la sentencia de actualización es:

UPDATE Tabla

SET Campo1 = Valor1, Campo2 = Valor2, ... , CampoN = ValorN

WHERE Condición de Selección

La cláusula SET asigna los valores Valor1, Valor2, ... , ValorN respectivamente a las columnas Campo1, Campo2, ... , CampoN y actualiza con dichos valores a la fila o filas de la Tabla que cumplan la Condición de Selección indicada en la sentencia.

La cláusula WHERE actúa en la sentencia UPDATE de forma exactamente igual a como lo hace en la sentencia SELECT, por lo que las condiciones de búsqueda en ambos casos tienen el mismo formato. Esto implica, en particular, que si no se especifica la cláusula WHERE, se actualizarán todas las filas de la tabla. Ello permite una actualización masiva de la tabla destino.

2 ***Actualizaciones en la Base de Datos***.-

Como se ha indicado en varias ocasiones, las sentencias SELECT, INSERT, DELETE, UPDATE y sentencias de transacciones se refieren a la manipulación de los datos en una Base de Datos. Estas sentencias de denominan colectivamente *Lenguaje de Manipulación de Datos (Data Manipulation Language)* o DML. Estas sentencias no pueden alterar la estructura de la Base de Datos, esto es, no pueden crear o destruir tablas o columnas en una Base de Datos.

Los cambios de estructura se realizan con el conjunto de sentencias denominadas conjuntamente *Lenguaje de Definición de Datos (Data Definition Language)* o DDL que permiten:

- Crear o suprimir una Base de Datos
- Definir y crear una nueva tabla

- Suprimir una tabla que ya no se necesita
- Cambiar la definición de una tabla existente
- Definir una tabla virtual (o Vista) de datos
- Establecer controles de seguridad para una Base de Datos
- Construir índices para hacer más rápido el acceso a la tabla
- Controlar el almacenamiento físico de los datos por parte del SGBD.

El núcleo del Lenguaje de Definición de Datos está basado en las sentencias siguientes:

- CREATE, que define y crea un objeto en la Base de Datos.
- DROP, que elimina un objeto existente en la Base de Datos
- ALTER, que modifica la definición de un objeto en la Base de Datos

En general, los principales productos comerciales de SGBD basados en SQL permiten utilizar el DDL mientras el SGBD está ejecutándose. La estructura de la Base de Datos es por tanto dinámica, permitiendo crear, eliminar o modificar la definición de las tablas de la base de datos mientras simultáneamente proporciona acceso a la base de datos a sus usuarios. Además, aunque el DDL y el DML son dos partes distintas del lenguaje SQL en la mayoría de los SGBD basados es SQL, la división es solamente conceptual. Las sentencias de DDL y de DML se remiten al SGBD de forma exactamente igual y pueden ser libremente entremezcladas en distintas sesiones de SQL.

Es preciso tener en consideración que el estándar de SQL de ANSI/ISO no exige la exige el soporte del DDL. Esto implica que las sentencias que se describen a continuación no están estandarizadas.

2.1 ***Creación de una Base de Datos***.-

Al no existir un estándar para la creación de una Base de Datos, cada SGBD comercial adopta un planteamiento en la creación de una base ligeramente diferente:

- Oracle crea una base de datos como parte del proceso de la instalación del software Oracle. Normalmente las tablas de usuario se colocan en esta base de datos única global.
- Ingres incluye la utilidad especial CREATEDB que crea una nueva base de datos Ingres. El programa adicional DESTROYDB suprime una base de datos no necesaria.

- SQL Server y OS/2 Extended Edition incluyen la sentencia:

CREATE DATABASE Base

Como parte de su DDL. La sentencia adicional:

DROP DATABASE Base

destruye la base de datos creada previamente.

- SQLBase utiliza la orden de MS-DOS *COPY* para crear una nueva Base de Datos. El usuario simplemente hace una copia de una plantilla de base de datos vacía suministrada con el software SQLBase. Para suprimir una base de datos existente se utiliza la orden de MS-DOS *DEL*.

2.2 ***Definiciones de Tablas***.-

La sentencia CREATE TABLE define una nueva tabla en la Base de datos y la prepara para aceptar datos.. Las diferentes cláusulas de la sentencia especifican los elementos de la definición de la tabla.

Cuando se ejecuta una sentencia CREATE TABLE, el usuario que la realiza (si tiene autorización para ello) se convierte en *propietario* de la tabla recién creada, a la cual se le da el nombre especificado en la sentencia , que debe ser un nombre SQL legal y no entrar en conflicto con el nombre de alguna tabla ya existente. La tabla recién creada está vacía pero el SGBD la prepara para aceptar datos añadidos con la sentencia INSERT.

Las columnas de la tabla se definen en el cuerpo de la sentencia CREATE TABLE y aparecen en una lista separada por comas e incluida entre paréntesis. El orden de la lista de definiciones de columnas determina el orden de izquierda a derecha de las columnas de la tabla. Cada definición de columna especifica:

* *El nombre de la columna*. Cada tabla debe tener un nombre único, pero los nombres de las columnas pueden ser iguales a los nombres de columnas de otras tablas.

* *El tipo de datos de la columna*. Establece la clase de datos que la columna puede almacenar. Algunos tipos de datos, como VARCHAR o DECIMAL requieren información adicional, como la longitud o el número de decimales de los datos. Esta información adicional se incluye entre paréntesis a continuación de la palabra clave que especifica el tipo de datos.

* *El requerimiento de los datos*. La cláusula NOT NULL impide que aparezcan valores NULL en la columna. En caso contrario se permiten valores nulos.

* *El valor por omisión*. Opcional para la columna, indica el valor que el SGBD debe asignar a la columna cuando en una sentencia INSERT aplicada a la tabla no se especifica un valor para la columna.

Además el SQL2 suministra varias partes diferentes para la definición de una columna, como que contenga valores únicos, que sea una clave primaria o una clave ajena o restringir los valores que puede contener.

La forma general de la sentencia CREATE TABLE es la siguiente:

```
CREATE TABLE Tabla

(Campo1     INTEGER              NOT NULL,

Campo2      VARCHAR(XX) NOT NULL,

Campo3      DATE                          ,

Campo4      MONEY                NOT NULL)
```

La sentencia CREATE TABLE ofrece ligeras variantes según el SGBD utilizado ya que cada SGBD utiliza sus propias palabras clave para identificar los distintos tipos de datos en las definiciones de columna. Además, Sybase y SQL Server difieren mucho de otros productos SGBD y del estándar ANSI/ISO en el manejo de los valores NULL, ya que el estándar indica que una columna puede contener valores nulos a menos que específicamente se declare NOT NULL. Sybase y SQL Server utilizan el criterio opuesto, suponiendo que los valores NULL no son permitidos a menos que la columna se declare explícitamente como NULL.

En cuanto a los "valores por omisión", tanto el estándar ANSI/ISO como productos SQL soportados por IBM soportan este tipo de valores para las columnas, pero lo hacen de forma diferente: El estándar ANSI/ISO permite especificar un valor por omisión para cada columna. Así:

Estándar ANSI/ISO:

```
CREATE TABLE Tabla

(Campo1     INTEGER              NOT NULL     DEFAULT 106  ,

Campo2      VARCHAR(XX) NOT NULL     DEFAULT 'Nombre',

Campo3      DATE                                          ,

Campo4      MONEY                NOT NULL)
```

Los productos SQL de IBM no permiten especificar un valor diferente para cada columna sino que proporcionan un valor específico de omisión para cada tipo de dato utilizado, así el valor de omisión para datos numéricos es 0, para datos de cadena (VARCHAR) es la cadena vacía, para datos de caracteres (CHAR) es el blanco y para datos de fecha y hora es la fecha y hora actual:

Sintaxis IBM:

```
CREATE TABLE Tabla
(Campo1    INTEGER              NOT NULL    WITH DEFAULT      ,
Campo2     VARCHAR(XX) NOT NULL    WITH DEFAULT   ,
Campo3     DATE                                               ,
Campo4     MONEY                NOT NULL)
```

2.2.1 ***Definiciones de clave primaria y ajena y Restricciones de Unicidad.-***

Además de la definición de las columnas de una tabla, la sentencia CREATE TABLE identifica la clave primaria de la tabla y las relaciones de la tabla con otras tablas de la Base de datos mediante las cláusulas PRIMARY KEY y FOREIGN KEY respectivamente.

La cláusula PRIMARY KEY especifica la columna o columnas que forman la clave primaria de la tabla. Esta columna (o combinación de columnas) sirve como identificador único para cada fila de la tabla. El SGBD requiere automáticamente que el valor de clave primaria sea único para cada fila de la tabla. Además, la definición de columna para todas y cada una de las columnas que forman la clave primaria debe especificar que la columna es NOT NULL.

La cláusula FOREIGN KEY especifica la clave ajena de la tabla y la relación que crea con otra tabla (Tabla Padre) de la Base de Datos. La cláusula especifica:

* La columna o columnas que forman la clave ajena, todas las cuales son columnas de la tabla que está siendo creada. Esto es, la columna de la tabla padre y la columna de la tabla hijo que se relaciona con la anterior deben tener el mismo nombre.

* La tabla que es referenciada por la clave ajena. Esta es la tabla padre en la relación, la tabla que se está definiendo es la tabla hija.

* Un nombre opcional para la Relación entre las dos tablas. Este nombre es opcional, no se utiliza en ninguna sentencia SQL pero es necesario si se desea poder suprimir la clave ajena posteriormente y puede aparecer en los mensajes de error.

* Como debe tratar el SGBD un valor NULL en una o mas columnas de la clave ajena, cuando compare filas con la tabla padre.

* Una regla de supresión opcional en la Relación (CASCADE, SET NULL, RESTRICT, SET DEFAULT o NO ACTION) que determina la acción que se debe realizar cuando se suprime una fila en la tabla padre

* Una regla de actualización opcional en la Relación (CASCADE, SET NULL, RESTRICT, SET DEFAULT o NO ACTION) que determina la acción que se debe realizar cuando se actualiza una parte de la clave primaria de la tabla padre.

Deben tenerse en cuenta las siguientes precisiones:

- La sintaxis definida no es estándar en ANSI/ISO

- No puede definirse una clave ajena que relacione la tabla que se est{a creando con una que no se ha creado todavía, es decir, debe crearse previamente la tabla padre antes de crear la tabla hijo. De no ser así, debe crearse la tabla hijo sin definición de clave ajena y posteriormente añadir dicha clave ajena utilizando la sentencia ALTER TABLE.

El estándar ANSI/ISO permite establecer en la sentencia CREATE TABLE restricciones de unicidad mediante la cláusula UNIQUE que exige que determinada columna o columnas tomen valores únicos. Sin embargo, prácticamente todos los SGBD mas populares hacen que las restricciones de unicidad formen parte de la sentencia CREATE INDEX que se describirá posteriormente.

La sintaxis general de creación de una tabla con claves primaria y ajena es:

```
CREATE TABLE Tabla1

(Campo1     INTEGER              NOT NULL,

Campo2      INTEGER              NOT NULL,

Campo3      VARCHAR(X)   NOT NULL,

Campo4      VARCHAR(X)   NOT NULL,

PRIMARY KEY (Campo1),

FOREIGN KEY SeRelacionaCon (Campo3)

REFERENCES Tabla2 ON DELETE SET NULL ON UPDATE CASCADE,
```

UNIQUE (Campo4))

Por último, si una clave primaria, una clave ajena o una restricción de unicidad afecta a una sola columna respectivamente, el estandar ANSI/ISO permite una forma abreviada de la sentencia cuya sintaxis es la siguiente:

```
CREATE TABLE Tabla1

(Campo1    INTEGER      NOT NULL    PRIMARY KEY,

Campo2     VARCHAR(X)   NOT NULL    UNIQUE,

Campo3     INTEGER      REFERENCES Tabla2,

Campo4     MONEY)
```

2.3 ***Eliminación de una Tabla***.-

Para eliminar una tabla existente en una Base de Datos se utiliza la sentencia DROP TABLE, seguida del nombre de la tabla a eliminar:

```
DROP TABLE Tabla
```

Cuando esta sentencia suprime una tabla de la Base de Datos, su definición y todos sus contenidos se pierden y no es posible recuperarlos. Habría que crear de nuevo la tabla y posteriormente introducir sus datos. Debido a estas consecuencias, la sentencia DROP TABLE debe utilizarse con extremo cuidado.

El estándar SQL2 establece que una sentencia DROP TABLE incluye una sentencia CASCADE o una sentencia RESTRICT que especifica el impacto que tiene la eliminación de una determinada tabla sobre otros elementos de la base de datos que dependen de ella. No obstante la mayoría de los SGBD aceptan la sentencia DROP TABLE sin ninguna opción.

2.4 ***Modificación de una Definición de Tabla***.-

La modificación de la definición de una tabla conllevaría alguna de las siguientes posibilidades:

- Añadir una definición de columna a una tabla
- Suprimir una columna de una tabla
- Cambiar el valor por omisión de una columna

- Añadir o eliminar una clave primaria para una tabla
- Añadir o eliminar una clave ajena para una tabla
- Añadir o eliminar una restricción de unicidad para una tabla
- Añadir o eliminar una restricción de comprobación para una tabla

2.4.1 ***Adición de una columna.-***

El uso mas común de la sentencia ALTER TABLE es el de añadir una columna a una tabla existente. La nueva columna se añade al final de las definiciones de columnas de la tabla y se sitúa mas a la derecha de la tabla. La sentencia ALTER TABLE es prácticamente igual a la sentencia CREATE TABLE y su sintaxis es:

ALTER TABLE Tabla

ADD CampoN VARCHAR(XX) NOT NULL

2.4.2 ***Supresión de una columna.-***

La sentencia ALTER TABLE no puede ser utilizada para eliminar una columna de una tabla. Si realmente se quiere eliminar una columna de una tabla, los pasos a seguir son:

1° Descargar los datos de la tabla (por ejemplo mediante una Vista)

2° Utilizar la sentencia DROP TABLE para eliminar la definición de la tabla

3° Utilizar la sentencia CREATE TABLE para redefinir la tabla sin la columna no deseada

4° Volver a cargar los datos anteriormente descargados.

2.4.3 ***Modificación de Claves primaria y ajena.-***

La sentencia ALTER TABLE también se utiliza para cambiar o añadir definiciones de claves primaria y ajena a una tabla. A diferencia de las definiciones de columna, las definiciones de clave primaria y ajena pueden ser añadidas y suprimidas con la sentencia ALTER TABLE. Las cláusulas que añaden definiciones de claves son exactamente las mismas que las contempladas en la sentencia CREATE TABLE y las cláusulas que suprimen las claves van antecedidas por la palabra reservada DROP:

Creación de una clave primaria:

ALTER TABLE Tabla

PRIMARY KEY (Campo1)

Creación de una clave ajena:

ALTER TABLE Tabla1

FOREIGN KEY NombreRelación (Campo2)

REFERENCES Tabla2

Eliminación de una clave primaria:

ALTER TABLE Tabla

DROP PRIMARY KEY

Eliminación de una clave ajena:

ALTER TABLE Tabla1

DROP FOREIGN KEY NombreRelación

La destrucción y creación de una clave puede hacerse en la misma sentencia SQL:

ALTER TABLE Tabla

DROP PRIMARY KEY

PRIMARY KEY (Campo2)

2.5 ***Definiciones de Restricción***.-

En el estándar SQL2 la definición de la estructura de la Base de Datos se ha ampliado para incluir un nuevo área: *"Restricciones sobre los datos que se pueden introducir en la Base de Datos"*. Estas restricciones, denominadas *constraints*, son fundamentalmente las siguientes:

- Restricción de unicidad.
- Restricción de clave primaria
- Restricción de clave ajena
- Restricciones de comprobación

- Aserciones
- Definiciones de Dominio

Las tres primeras ya han sido tratadas anteriormente y se aplican en la sentencia CREATE TABLE y pueden ser modificadas por la sentencia ALTER TABLE.

2.5.1 ***Restricciones de comprobación.-***

La restricción de comprobación (CHECK CONSTRAINT) limita el contenido de una tabla particular. En el estándar SQL2 una restricción de comprobación se especifica como una condición de búsqueda y aparece como parte de la definición de la tabla:

```
CREATE TABLE Tabla

(Campo1     INTEGER       NOT NULL,

Campo2      VARCHAR(X)    NOT NULL,

...  ...  ...

Fecha1      DATETIME      ,

CHECK (( Fecha1 < "01-Ene-2001) OR (Campo1 <= 3000)))
```

La restricción de comprobación se verifica cada vez que una sentencia SQL intenta actualizar o insertar datos en la tabla.

2.5.2 ***Aserciones.-***

Una aserción (ASSERTION) es una restricción de la base de datos que restringe los contenidos de la base de datos en su conjunto. Se especifica como una condición de búsqueda, pero, a diferencia de la restricción de comprobación, la condición de búsqueda de una aserción puede restringir el contenido de múltiples tablas y los de los datos de las relaciones entre ellas. Se especifica, por tanto, como parte de la definición de la base de datos completa utilizando la sentencia CREATE ASSERTION. Un ejemplo sería:

```
CREATE ASSERTION ValorMaximo

CHECK ((Tabla1.Campo1 = Tabla2.Campo3) AND

(SUM(Tabla1.Campo4) <= ValorMaximo))
```

2.5.3 ***Definiciones de Dominio.-***

Existe una restricción a priori en las columnas de una tabla, establecida por el tipo de los datos (y la extensión si procede) de la columna. Sin embargo, cuando se quiere especificar un dominio mas concreto para una determinada columna de una tabla, SQL2 implementa el concepto formal de dominio como una parte de la definición de la base de datos. Según SQL, un dominio es un conjunto nominado de valores de datos que realmente funciona como un tipo de datos adicional para utilizarlo en la definición de la base de datos. Se crea con la sentencia CREATE DOMAIN y puede ser referenciado dentro de la definición de la Tabla como si fuese un tipo de dato.

2.6 ***Indices.-***

Un índice es una estructura que proporciona un acceso rápido a las filas de una tabla en función de los valores de una o mas columnas.

El SGBD utiliza el índice de la misma manera que se usa el índice de un libro: El índice almacena los valores y punteros a las filas en las que los valores se producen. En el índice los valores de datos están ordenados de forma ascendente o descendente para que el SGBD pueda buscar rápidamente el índice para encontrar un valor particular; posteriormente puede seguir el puntero para localizar la fila que tiene el valor.

La presencia o ausencia de índice es completamente transparente al usuario de SQL que acceda a una tabla. El funcionamiento de los índices es el siguiente:

Si el usuario realiza una consulta a la base de datos como:

SELECT Campo1, Campo2

FROM Tabla

WHERE Campo3 = 'Condición de Búsqueda'

Si no existe un índice asociado a la columna Campo3 el SGBD realizará la consulta exista o no exista información en ella. Procesará la consulta recorriendo secuencialmente la tabla fila a fila, para todas y cada una de las filas, verificando si se cumple 'Condición de Búsqueda' en Campo3. Si la tabla es muy grande la exploración puede llevar minutos, incluso horas.

Si existiera un índice asociado a la columna Campo3, El SGBD examina el índice para encontrar el valor solicitado ('Condición de Búsqueda') y luego sigue el puntero para encontrar la fila o filas solicitadas de la tabla. La búsqueda en índices es mucho mas rápida ya que el índice está ordenado y sus filas son muy pequeñas. Pasar del índice a la fila correspondiente es también muy rápido ya que el índice informa al SGBD el lugar del disco en el que está localizada la fila.

La gran ventaja de la utilización de índices se centra en que se acelera enormemente la ejecución de sentencias SQL con condiciones de búsqueda que se refieren a columnas indexadas.

Los inconvenientes se refieren a que consumen un espacio de disco adicional y a que deben ser actualizados cada vez que se añada a la tabla una fila nueva o se actualice el contenido de una fila existente.

Para establecer un índice a una columna de una tabla deben tenerse en cuenta las consideraciones siguientes:

- La indexación es apropiada a columnas que son utilizadas frecuentemente en condiciones de búsqueda.

- La indexación es aconsejable cuando las consultas de una tabla son mas frecuentes que las inserciones y las actualizaciones.

- El SGBD siempre establece un índice para la clave primaria ya que presupone que el acceso a la tabla se efectuará mas frecuentemente a través de dicha clave.

Para crear un índice se utiliza la sentencia CREATE INDEX mediante la cual se asigna un nombre al índice y se especifica la tabla para la cual se crea el índice, así como la columna o columnas a indexar y el orden, ascendente o descendente en que debe hacerse la indexación:

```
CREATE INDEX Campo3IDX

ON Tabla (Campo3)
```

La palabra clave UNIQUE que se utiliza para indicar que la columna a indexar debe tener valores únicos. el estándar ANSI/ISO la asocia a la sentencia CREATE TABLE, sin embargo el SQL basado en DB2 la asocia con el índice:

```
CREATE UNIQUE INDEX Campo3IDX

ON Tabla (Campo3)
```

Por último, la sentencia:

```
DROP INDEX Campo3IDX
```

Suprime un índice creado anteriormente.

---