Teórico 2

Repaso SQL

SQL

- SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos relacional, "SQL" es una abreviatura de Structured Query Language (Lenguaje de Consultas Estructurado).

Historia del Estándar

- El Lenguaje SQL fue diseñado sobre un prototipo de IBM que incluía un lenguaje de consulta a base de datos llamado SEQUEL.

- El Instituto de estandarización ANSI y más tarde la ISO adoptan SQL como el lenguaje estándar de consulta a bases de datos. SQL 1999 incorpora triggers. SQL 2005 incorpora la interacción con XML. En la actualidad el estándar es SQL2012.

- Aunque el ANSI/ISO SQL es el estándar, ORACLE presentó su propia versión comercial, y más tarde lo han hecho Microsoft (SQL SERVER) y otras empresas. En la actualidad MySQL es de libre distribución y su servidor es el más utilizado en servidores de bases de datos de Internet.

DML, DDL y DCL

- SQL está dividido en tres partes:
  - Lenguaje de Manipulación de Datos (DML siglas en Ingles): Brinda instrucciones que incluyen un lenguaje de consultas, basado en el Álgebra Relacional, además incluye sentencias para agregar modificar y eliminar datos.
  - Lenguaje de Definición de Datos (DDL siglas en Ingles): Proporciona instrucciones para la definición del esquema (Estructura) de la Base de Datos, por ej.: crear tablas, índices, etc.
  - Lenguaje de Control de Datos (DCL siglas en Ingles): contiene instrucciones que permiten manipular los permisos sobre las bases de datos y sus objetos.

DML

Comando SELECT

El comando más usado en SQL es la instrucción SELECT, se utiliza para recuperar datos de diferentes tablas de una base de datos. Consta de seis cláusulas: las dos primeras (SELECT y FROM) obligatorias y las cuatro restantes(WHERE, GROUP BY, HAVING, ORDER BY) opcionales.

Su sintaxis es:

```
SELECT [ALL|DISTINCT]
        { * | expr_columna_1 [AS c_alias_1]
          [, expr_columna_2 [AS c_alias_2][,...]]}
  FROM nombre_tabla_1 [t_alias_1][, table_name_n [t_alias_n][,...]]
  [WHERE condicionWhere]
  [GROUP BY expr_columna_group1  [,expr_columna_group2] [,...]]
  [HAVING condicionHaving]]
  [{UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]
  [ORDER BY nombre_campo_i1 [ASC|DESC]
                  [, nombre_campo_j1 [ASC|DESC]][,...]]];
```

Ejemplos

De ahora en adelante los ejemplos de consultas SQL estarán hechos sobre el siguiente esquema de base de datos.

articulos(Nart, desc, precio, cant, stock_min, stock_max)

Provee(Nprov,Nart, precio_venta)

Proveedores(Nprov, nombre, direccion)

Clientes(Ncli, nombre, direccion)

Compran(Ncli, Nart)

Con las siguientes instancias:

articulos

Nart	desc	precio	cant	stock-min	stock-max
100	Café	5.2	8	10	50
101	Azucar	1.2	30	15	45
102	Harina	1.1	35	16	40

proveedores

Nprov	nombre	dir
200	Grasano	Bs As 100
201	Atomo	Lima 85
202	VEA	Alvear 455

provee

Nprov	Nart	precio_Venta
200	100	4
200	101	1.1
201	101	1
201	102	0.5
202	102	1

clientes

Ncli	nombre	dir
300	Pedro	Bs As 90
301	Juan	Jujuy 800
302	Maria	Cordoba 24

compran

Ncli	Nart
300	100
300	102
301	101
302	101

Ejemplo

1) Devolver un listado con todos los campos de los artículos a reponer.

SELECT * FROM articulos WHERE stock_min >cant

Resultado:

Nart	desc	precio	cant	stock-min	stock-max
100	Café	5.2	8	10	50

Ejemplos (sigue)

2) Dar un listado con los proveedores y los artículos que provee con su precio de venta

```
SELECT Nart, precio_venta,p.Nprov, nombre, dir
FROM provee p, proveedores pro WHERE p.Nprov = pro.Nprov ORDER
BY nArt
```

Resultado

Nart	precio_Venta	Nprov	nombre	dir
100	4	200	Grasano	Bs As 100
101	1.1	200	Grasano	Bs As 100
101	1	201	Atomo	Lima 85
102	0.5	201	Atomo	Lima 85
102	1	202	VEA	Alvear 455

Expresiones

En una expresión se pueden cotejar con operadores de comparación, columnas con otras columnas o con expresiones del tipo de la columna, las expresiones se pueden conectar con otras por medio de operadores lógicos

Operadores Lógicos

- **AND**
- **OR**
- **NOT**
- **XOR**

Operadores de Comparación

- **< , >, <>, <=, >=, =**
- **BETWEEN:** Utilizado para especificar un intervalo de valores. Utilizado en tipos String, numéricos y de fechas.
- **LIKE:** Utilizado en la comparación de un patrón. El carácter '%' se utiliza como comodín
- **In:** Utilizado para saber si el valor de un campo se encuentra en una subconsulta de selección de una columna.
- **EXIST:** Devuelve verdadero si una subconsulta es vacía, si no devuelve falso.

Ejemplos

a)

SELECT * FROM articulos WHERE desc like "A%"

Listados de artículos cuya descripción empieza con la letra A

b)

SELECT * FROM articulos WHERE desc like "A%" AND desc like “%K"

Listados de artículos cuya descripción empieza con la letra A y termina con la letra K

c)

SELECT * FROM articulos WHERE Nart IN (SELECT Nart FROM provee);

Devuelve los artículos que son provistos por algún proveedor.

Cláusula ORDER BY

La cláusula ORDER BY ordena los resultados de la consulta en base a los datos de una o más columnas. Si se omite, los resultados serán ordenados por el primer campo que sea clave en el índice que se haya utilizado.

Su sintaxis es:

ORDER BY *expresión_orden1* [ASC|DESC]

[, *expresión_orden2* [ASC|DESC]][,...];

Argumento	**Descripción**
Expresión_ordenN	Puede ser el nombre de un campo, expresión o el número de posición que ocupa la expresión de columna en la cláusula SELECT.
ASC	Produce el ordenamiento ascendente de los valores de la columna. Esta opción es la elegida por defecto.
DESC	Produce el ordenamiento descendente de los valores de la columna.

Ejemplo

1) Dar un listado con los artículos, que proveedores lo suministran y su respectivo precio de venta. El listado debe estar ordenado ascendentemente por el número de artículo.

```
SELECT DISTINCT Nart AS Numero_Art, Pv.Nprov AS Numero_prov,
Nombre, Precio_Venta
FROM Proveedores Pd, Provee Pv
WHERE Pd.Nprov =Pv.Nprov
ORDER BY Nart
```

Nota: se puede observar la utilización de alias tanto en el nombre de tablas como en el de columnas.

Resultado

numero_Art	nu		
100			
101			
101			

Expresiones

Funciones Agregadas

SQL proporciona operadores agregados que toman el nombre de una columna como argumento. El valor del operador agregado se calcula sobre todos los valores de la columna especificada en la tabla completa. Estas funciones se pueden combinar con la cláusula GROUP BY que se vera mas adelante.

Las funciones agregadas básicas del Estándar son:

- AVG (Calcula el promedio)
- COUNT (Cuenta la cantidad de registros)
- SUM (Suma los valores de una columna)
- MIN (Calcula el mínimo valor de una columna)
- MAX (Calcula el máximo valor de una columna)

AVG

Devuelve el promedio de los valores de una expresión de columna.

Ejemplo:

```
SELECT AVG(Precio) as promedio FROM articulos
```

Devolverá el promedio de precios de todos los artículos.

Resultado

promedio
2,5

Cláusula GROUP BY

La cláusula GROUP BY especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la base de datos, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria de resultados para cada grupo.
Su sintaxis:

[GROUP BY *expr_columna_group1*
[,*expr_columna_group2*] [,...]]

Donde:
expr_columna_groupN deben coincidir con las expresión de columna utilizada en la cláusula SELECT.

Ejemplo.

1) Retornar un listado con los números de artículos y el precio promedio de venta de los proveedores para cada artículo.

```
SELECT Nart, AVG(precio_venta) as promedio_p_v
FROM Provee
GROUP BY Nart;
```

Resultado

Nart	promedio_p_v
100	4
101	1.05
102	0.75

Cláusula Having

La cláusula HAVING trabaja muy similarmente a la cláusula WHERE, y se utiliza para considerar sólo aquellos grupos que satisfagan una condición dada.

Su sintaxis es:

HAVING *condicionHaving*

Donde *condicionHaving* es la condición que deben cumplir los grupos. Las expresiones permitidas en la cláusula HAVING deben involucrar funciones agregadas. Cada expresión que utilice sólo atributos planos deberá ubicarse en la cláusula WHERE. Por otro lado, toda expresión que involucre funciones agregadas debe aparecer en la cláusula HAVING.

Ejemplo

Retornar un listado con los números de artículos y el precio promedio de venta de los artículos cuyo precio promedio de venta de los proveedores es menor a $2.

```
SELECT Nart, AVG(precio_venta)
FROM Provee
GROUP BY Nart
Having AVG(precio_venta) < 2;
```

Resultado

Nart	avg
101	1.05
102	0.75

Comando INSERT

Este comando se utiliza para agregar uno o mas registros (filas). Este comando se puede utilizar para efectuar dos tipos de operaciones: a) Insertar un único registro ó b) Insertar en una tabla los registros contenidos en otra.

Comando INSERT para un registro:

INSERT INTO *destino* [(*campo1*[, *campo2*[, ...]])]VALUES (*valor1*[, *valor2*[, ...])

Comando INSERT para insertar más de un registro:

INSERT INTO *destino* [(*campo1*[, *campo2*[, ...]])] *selección*

Argumento	Descripción
Destino	El nombre de la tabla donde se van a añadir registros.
CampoN	Los nombres de los campos donde se van a añadir los datos, si no se ponen se asume que son todos atributos de la tabla destino.
Selección	Es una expresión de selección que debe tener el mismo número de columnas y tipos que las columnas a insertar. Los registros que de cómo resultado la expresión SELECT serán agregados en la tabla destino (ver comando SELECT). Esto permite insertar en una tabla varios registros.
ValorN	Los valores que se van a insertar en los campos específicos del nuevo registro. Cada valor se inserta en el campo que corresponde a la posición del valor en la lista: valor1 se inserta en campo1 del nuevo registro, valor2 dentro de campo2, y así sucesivamente. Debe separar los valores con una coma y escribir los campos de texto entre comillas (' ') y las fechas indicarlas en formato dd/mm/aa y entre comillas.

Ejemplo

1) INSERT INTO articulos (Nart,Descripcion,precio,Stock_max,Stock_min) VALUES (5, 'Vino x1L', 1.5,90,20);

En este ejemplo se agrega un nuevo registro a la tabla artículos, pero en el campo cantidad no se pone ningún valor (ver definición tabla artículos)

2) INSERT INTO proveedores SELECT * FROM clientes

En este ejemplo se agrega a la tabla proveedores todos los clientes

Comando UPDATE

Para cambiar uno o más valores de campos de registros en una tabla, se utiliza el comando UPDATE.

Su sintaxis es:

UPDATE *tabla* SET *campo1* = *valor1* [, *campo2* = *valor2* [, ...]] [WHERE *condición*];

Argumento	**Descripción**
Tabla	Nombre de la tabla cuyos datos desea modificar.
CampoN	Nombre del campo cuyo valor se actualizará.
ValorN	Expresión cuyo valor tomará el campoN. La expresión debe ser del tipo del campo.
Condición	Una expresión que determina qué registros se actualizarán. Sólo se actualizan los registros que satisfacen la expresión.

Ejemplos

1)
UPDATE articulos SET precio = precio * 1.2;

Esta instrucción incrementa los precios de todos los artículos en un 20 por ciento.

2)
UPDATE articulos SET precio = precio * 1.2
WHERE precio <2;

La ejecución de este comando incrementa los precios de los artículos cuyo precio es menor a $2, en un 20 por ciento.

Comando DELETE

El comando DELETE se utiliza para borrar uno o varios registros de una tabla particular.

Su sintaxis es:

DELETE FROM *tabla* [WHERE *condición*];

Argumento	**Descripción**
Tabla	Nombre de la tabla cuyos registros se van a eliminar.
Condición	Expresión que determina qué registros se van a eliminar.

Nota:
Si se omite la cláusula WHERE se eliminan todos los registros de la tabla.

Ejemplos

1)
DELETE FROM proveedores;

La ejecución de este comando producirá que la tabla proveedores quede vacía.

2)
DELETE FROM proveedores
WHERE Nprov=200;

Este comando elimina el registro correspondiente al proveedor número 200.

Reuniones (JOIN)

Se utilizan dentro de la cláusula from y su sintaxis es:

tabla1 [NATURAL] { {LEFT|RIGHT|FULL} OUTER | INNER} JOIN *tabla2* [ON *condicionJoin*][using (*columna1[, columna2[, ...]])]

Donde *tablaN* son las tablas involucradas en la reunión , *condicionJoin* es la condición(θ) del join y *columnaN* son las columnas involucradas en el join

Su comportamiento es similar al visto en el álgebra relacional, vista en Bases de Datos I.

DDL

Tipo de Datos

Los tipos de datos SQL se clasifican en 13 categorías primarias, las cuales pueden tener sinónimos reconocidos por dichos tipos de datos.

Cabe aclarar que dependiendo del manejador de base de datos, puede haber algunas variaciones en los tipos de datos de los atributos o campos.

En SQL-92, además de los tipos de datos del SQL-89 (INTEGER, SMALLINT, CHARACTER, DECIMAL, NUMERIC, REAL, FLOAT y DOUBLE PRECISION) también se admiten los siguientes: CHARACTER VARYING, DATE, TIME, BIT, TIMESTAMP, INTERVAL y BIT VARYING.

SQL 99, extiende SQL con características de base de datos objeto-relacionales.

Tipos de datos de Mysql

TinyInt	Es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255.
Bit ó Bool	Un número entero que puede ser 0 ó 1.
SmallInt	Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.
MediumInt	Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a 16777215.
Integer, Int:	Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295.
BigInt	Número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.
Float	Número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38 .
xReal, Double	Número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308 .
Decimal, Dec, Numeric	Número en coma flotante desempaquetado. El número se almacena como una cadena.

Date	Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia.
DateTime	Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-dia horas:minutos:segundos.
TimeStamp	Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo.
Char(n)	Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.
VarChar(n)	Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.
TinyText y TinyBlob	Columna con una longitud máxima de 255 bytes (caracteres en text)..
Blob y Text	Columna con un máximo de 65535 bytes (caracteres en text)..
MediumBlob y MediumText	Columna con un máximo de 16.777.215 bytes (caracteres en text).
LongBlob y LongText	Columna con un máximo de caracteres 4.294.967.295 bytes (caracteres en text).

Create Table

Es el comando fundamental para definir datos, crea una nueva relación (una nueva tabla). La sintaxis del comando CREATE TABLE es:

CREATE TABLE *tabla* (*campo1 tipo* [(*tamaño*)] [NOT NULL] [*índice1*]
[, *campo2 tipo* [(*tamaño*)] [NOT NULL] [*índice2*] [, ...]]
[, *campoN tipo* [(*tamaño*)] [NOT NULL] [*índiceN*] [, ...]]
[, *ligaduraIntegridad1* [, ...]]
[, *ligaduraIntegridadN* [, ...]])

Argumento	Descripción
Tabla	Nombre de la tabla a crear.
CampoN	Nombre del campo o de los campos componentes de la nueva tabla. La nueva tabla debe contener, al menos, un campo.
Tipo	Es el tipo de datos del campo en la nueva tabla. Anteriormente se vieron los tipos permitidos de un campo. También puede ser un dominio definido por el usuario.
Tamaño	Tamaño del campo, sólo se aplica para algunos tipos, ej.: varchar.
ÍndiceN	Es una cláusula CONSTRAINT que define el tipo de restricción a crear. Más adelante se describe esta cláusula.
LigaduraIntegridadN	Es una cláusula CONSTRAINT que define el tipo de restricción a crear.
NOT NULL	Indica que el campo no puede recibir valores nulos.

Ejemplo

```
CREATE TABLE Proveedores
(Nprov  INTEGER NOT NULL CONSTRAINT pkproveedores PRIMARY KEY,
 Nombre VARCHAR(50) NOT NULL CONSTRAINT uniconombre UNIQUE,
 Direccion VARCHAR(50));
```

La ejecución de este comando produce como resultado la creación de una tabla Proveedores con tres campos; Nprov: de tipo entero y clave primaria de la tabla, Nombre: de tipo cadena de caracteres de 50 e índice único o clave secundaria, y por último el campo Dirección de tipo cadena de caracteres de 50.

Ligaduras de Integridad

- Claves primarias (Primary key)
- Claves secundarias o candidatas (Unique)
- Cardinalidades de relaciones (en E/R)
- Ligaduras de dominios (check).
- Integridad referencial (Foreing key)
- Aserción (Assertion)
- Disparadores (Trigger)

Cláusula CONSTRAINT

Se utiliza la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar restricciones.

Existen dos sintaxis para esta cláusula dependiendo si se quiere crear un índice de un único campo o si se trata de un índice multi-campo.

Para los índices de campos únicos:

CONSTRAINT *nombre*
{PRIMARY KEY | UNIQUE | NOT NULL | REFERENCES *tablaexterna* [(*campoexterno1*]}

Para los índices de múltiples campos:

CONSTRAINT *nombre*
{
PRIMARY KEY (*primario1*[,*primario2* [, ...]])
| UNIQUE (*único1*[,*único2* [, ...]])
| NOT NULL (*nonulo1* [, *nonulo2* [, ...]])
| CHECK *condicion*
| FOREIGN KEY (*referencia1* [, *referencia2* [, ...]])
REFERENCES *tablaexterna [*(*campoexterno1*
[,*campoexterno2* [, ...]])]
[ON DELETE accion][ON UPDATE accion]
}

Tipo de Índice	Descripción
PRIMARY KEY	Genera un índice primario el campo o los campos especificados. Todos los campos de la clave primaria deben ser únicos y no nulos, cada tabla sólo puede contener una única clave primaria.
UNIQUE	Genera un índice de clave única, de esta forma se generan las claves secundarias o candidatas.
CHECK	Es para especificar una condición (condición) que deben cumplir los datos de una tabla.
FOREIGN KEY	Genera un índice externo (toma como valor del índice campos contenidos en otras tablas), de esta forma se incorporan restricciones de **integridad referencial**.

Argumento	Descripción
Nombre	Es el nombre del índice que se va a crear.
PrimarioN	Es el nombre del campo o de los campos que forman el índice primario.
UnicoN	Es el nombre del campo o de los campos que forman el índice de clave única.
NonuloN	Es el nombre del campo o de los campos que no admiten valores nulos.
ReferenciaN	Es el nombre del campo o de los campos que forman el índice externo (hacen referencia a campos de otra tabla). Se utiliza para introducir restricciones de integridad referencial.
Tabla externa	Es el nombre de la tabla que contiene el campo o los campos referenciados en refN.
CampoexternoN	Es el nombre del campo o de los campos de la tabla externa especificados por ref1, ref2, ..., refN.
Accion	Puede ser: Cascade, set null, set Default y restricted o no action

Ejemplos

1)CREATE TABLE articulos (

Nart INTEGER NOT NULL PRIMARY KEY,

Descr varchar(100), Precio FLOAT, Cant INTEGER,

Stock_min INTEGER, Stock_max INTEGER,

CONSTRAINT control_min_max CHECK (stock_min<stock_max),

CONSTRAINT precio_positivo CHECK (precio>0));

La ejecución de este comando produce como resultado la creación de la tabla artículos con seis campos; Nart de tipo entero y clave primaria de la tabla, descr: de tipo cadena de caracteres de 100, precio de tipo FLOAT y por último los campos cant, stock_min y stock_max de tipo INTEGER. Además se agregan dos restricciones CHECK, una para controlar que el stock_min sea menor que el stock_max y la otra para controlar que el precio sea positivo.

2)

```
CREATE TABLE PROVEE
(Nprov  INTEGER NOT NULL,
 Nart  INTEGER NOT NULL,
 Precio_venta FLOAT,
 Constraint pkprovee primary key (Nart,Nprov),
 Constraint fkproveedores foreign key (Nprov) references proveedores,
 Constraint fkarticulos foreign key (Nart)
                references articulos on delete cascade)
```

Este comando crea la tabla PROVEE con los campos Nart, Nprov y precio_venta, además tenemos como clave primaria a Nart y Nprov. La tabla PROVEE posee además dos claves Foráneas (integridad referencial), una, el atributo Nprov que hace referencia a la tabla PROVEEDORES y la otra, el atributo Nart que hace referencia a la tabla ARTICULOS y el borrado será en cascada.

Alter Table

Comando utilizado para modificar la Estructura de una tabla existente, su sintaxis es:

ALTER TABLE tabla
{
 ADD {COLUMN *campo* *tipo* [(*tamaño*)] [NOT NULL]
      | *clausulaConstraint*}
      | DROP {COLUMN *campo* | CONSTRAINT *clausulaEliminar*}
}

Argumento	Descripción
Tabla	Es el nombre de la tabla que se desea modificar.
Campo	Es el nombre del campo que se va a agregar o eliminar.
Tipo	Tipo del campo que se va a agregar.
Tamaño	Tamaño del campo que se va a agregar, depende del tipo de dato.
clausulaConstraint	Es una cláusula CONSTRAINT que define el tipo de restricción a crear.
clausulaEliminar	Nombre de la cláusula constraint a eliminar.

1) ALTER TABLE Proveedores ADD Telefono VARCHAR (20) NOT NULL;

La ejecución de este comando provoca que la tabla Proveedores, ya existente, incorpore una nueva columna llamada “Telefono” de tipo cadena de caracteres de 20 de longitud máxima.

2) ALTER TABLE Proveedores DROP Direccion;

Elimina la columna Dirección de la tabla Proveedores

3) ALTER TABLE Proveedores ADD CONSTRAINT telefonounico UNIQUE Telefono;

Agrega una restricción de Unique para el campo telefono de la tabla Proveedores

4) ALTER TABLE Proveedores DROP CONSTRAINT telefonounico;

Elimina la restricción con nombre telefonounico sobre el campo teléfono de la tabla Proveedores

Dominios

Un dominio es un tipo de datos definido por el usuario en función de un tipo existente. Su sintaxis es:

CREATE DOMAIN nombreDominio [AS] *tipoDatos*

[DEFAULT valorPorDefecto]

[NOT NULL] [CHECK (condición)];

Donde :

Argumento	**Descripción**
nombreDominio	Nombre del nuevo dominio, el cual se hará referencia en la definición.
tipoDatos	Es un tipo de datos de SQL.
valorPorDefecto	Es el valor por defecto que tendrán las columna de este dominio
condición	Condición que deben cumplir los valores del dominio.

Ejemplo

```
CREATE DOMAIN enteroPositivo AS INTEGER
        DEFAULT 2
        CHECK (value>1)
                NOT NULL;
```

Esta instrucción crea un dominio de valores enteros mayores que 1, no acepta valores nulos y tiene por defecto el valor 2.

Ejemplo de utilización

```
CREATE TABLE PROVEE
( Nprov enteroPositivo,
  Nart enteroPositivo,
 Precio_venta FLOAT,
 Constraint pkprovee primary key (Nart,Nprov),
 Constraint fkproveedores foreign key (Nprov) references proveedores,
 Constraint fkarticulos foreign key (Nart) references articulos)
```

Este comando crea la tabla PROVEE utilizando el dominio definido para los campos Nart, Nprov.

Índices

Un índice esta basado en una o más columnas de una tabla, su función es ordenar el contenido de las columnas especificadas y almacenar esta información ordenada en disco, para tener acceso rápido y eficiente a estas columnas.

Conforme los registros se anexan, cambian o eliminan, el sistema de administración de base de datos actualiza automáticamente el índice para reflejar los cambios.

Creación de Índices

CREATE [UNIQUE] INDEX *índice* ON *tabla* (*campo1* [ASC|DESC] [, *campo2*[ASC|DESC], ...])
  [WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

Argumento	Descripción
Índice	Nombre del índice a crear.
Tabla	Nombre de una tabla existente en la que se creará el índice.
CampoN	Nombre del campo o lista de campos que constituyen el índice.
ASC | DESC	Indica el orden de los valores de los campos ASC indica un orden ascendente (valor predeterminado) y DESC un orden descendente.
UNIQUE	Indica que el índice no puede contener valores duplicados.
DISALLOW NULL	Prohíbe valores nulos en el índice
IGNORE NULL	Excluye del índice los valores nulos incluidos en los campos que lo componen.
PRIMARY	Asigna al índice la categoría de clave principal, en cada tabla sólo puede existir un único índice que sea Clave Primaria.

Ejemplo

CREATE INDEX IndiceTelefono ON Proveedores (Telefono);

Crea un índice llamado IndiceTelefono en la tabla proveedores con el campo Telefono.

Vistas

- Son relaciones que no forman parte del modelo lógico de la base de datos que son visibles a los usuarios como tablas virtuales.
- Son utilizadas por razones de seguridad, para ocultar cierta información a los usuarios.
- La actualización sobre las vistas conlleva varios problemas.

Sintaxis

Create view <nombre_vista> as <expresion_de_consulta>

Ejemplo:

Create view nombre_proveedores as

select nombre from proveedores;

Borrado de Estructuras

El comando DROP elimina una tabla existente de una base de datos o elimina un índice existente de una tabla. Su sintaxis es:

DROP {TABLE tabla | INDEX índice ON tabla}

Donde:

Argumento	**Descripción**
Tabla	Nombre de la tabla que se va a eliminar o la tabla de la cual se va a eliminar un índice.
Índice	Nombre del índice que se va a eliminar de tabla.

EJEMPLOS

1) DROP TABLE provee;

Elimina de la base de datos la tabla provee.

2)DROP TABLE proveedores;

Elimina de la base de datos la tabla proveedores.

3) DROP INDEX IndiceTelefono ON proveedores;

Elimina el índice IndiceTelefono de la tabla proveedores.