Bases de datos

UNIDAD 3

EL LENGUAJE SQL DE LAS BASES DE DATOS RELACIONALES

SQL

SQL (Structured Query Language) es el **lenguaje estándar** de las bases de datos relacionales.

Es un lenguaje declarativo que permite especificar diversos tipos de operaciones sobre éstas.

Es capaz de conjugar las operaciones del **álgebra** y el cálculo **relacional** con operadores adicionales, y *definir* así *consultas* para *recuperar* o *modificar* información de bases de datos, así como hacer cambios en ellas.

Pero no sólo incluye consulta. SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. En conjunto, disponemos de instrucciones para **definir** (*crear* y *modificar* el esquema), **mantener** (*insertar*, *actualizar*, *eliminar*) y **consultar** BBDD relacionales.

Comandos

Los tipos de comandos en SQL se agrupan en dos categorías o sub-lenguajes:

DDL (Definition Data Language): permite definir el esquema de bases de datos, creando relaciones (tablas), campos e índices, o modificando las definiciones existentes.

DML (Data Manipulation Language): permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos, así como insertar, modificar y eliminar registros de las tablas.

Componentes del Lenguaje

- Sentencias DDL

  (Lenguaje de Definición de Datos)
  - Create
  - Drop
  - Alter

- Sentencias DML

  (Lenguaje de Manipulación de Datos)
  - Insert
  - Update
  - Delete
  - Select

- Control de transacciones
  - Begin Transaction
  - Commit
  - Rollback

- SQL almacenado
  - View
  - Store Procedures
  - Function
  - Trigger

Comandos DDL

- CREATE: crea objetos de la base
- ALTER: modifica objetos de la base
- DROP: elimina objetos de la base

SENTENCIA CREATE

```
CREATE DATABASE “Nombre_BD” ON ( NAME =
“Archivo_datos”, FILENAME = ‘Path_mdf', SIZE = 10,
MAXSIZE = 50, FILEGROWTH = 5 )
LOG ON ( NAME = “Archivo_log”, FILENAME =
‘Path_log', SIZE = 5MB, MAXSIZE = 25MB,
FILEGROWTH = 5MB ) ;
```

CREATE TABLE

```
CREATE TABLE tabla (
        definición de campos,
        claves y restricciones
);
```

CREATE TABLE

Definición:

```
nombre_campo tipo marcadores
```

Tipos:

- `CHAR(n)` Cadenas longitud fija hasta *n* caracteres.
- `VARCHAR(n)` Cadenas longitud variable hasta *n* caracteres.
- `INTEGER`, `BIGINT`, `SMALLINT`, . . . Enteros.. . .
- `REAL` Número real
- `DATE` Fechas. Están compuestas de: **YEAR**, **MONTH** y **DAY**.
- `TIME` Horas. Están compuestas de **HOUR**, **MINUTE** y **SECOND**.
- . . . y muchos más (varían según el SGBD).

Tipos de datos

- Numeros
  - int, bigint
  - float
  - bit
  - Numeric, decimal, money
- Cadenas
  - char
  - varchar
- Cadenas Binarias
  - binary
  - image
  - varbinary
- Caracteres unicode
  - nchar
  - nvarchar
  - ntext

- Fechas
  - date
  - datetime
  - shortdatetime
  - time

- Otros tipos de objetos
  - cursores
  - sqlVariant
  - table
  - Timestamp
  - Xml
  - uniqueidentifiquer

Marcadores y restricciones

- `AUTO_INCREMENT` - Autonumérico, secuencial que va asignando el entero siguiente al máximo valor almacenado para el campo.
- `DEFAULT val` - Establece un valor por defecto al campo.
- `NOT NULL` - No puede contener valores nulos.
- `PRIMARY KEY` - El campo es la clave primaria no compuesta.
- `REFERENCES tabla [(campo)]` - Clave ajena no compuesta.
- `CHECK (cond.)` - El campo debe cumplir una condición.[2]

Claves compuestas y restricciones

- `PRIMARY KEY (campos)` - Clave primaria (compuesta o no).
- `FOREIGN KEY (campos) REFERENCES tabla [(campos)]` - Clave ajena (compuesta o no).
- `CHECK (cond.)` - El campo debe cumplir una condición.[3]

Tanto a las claves primarias y ajenas como a los chequeos de condición se les puede anteponer la partícula `CONSTRAINT nombre`, para nombrar la restricción.

Ejemplo genérico básico

```
CREATE TABLE PRODUCTOS (
  codigo_producto INTEGER AUTO_INCREMENT,
  nombre_producto VARCHAR(20) UNIQUE NOT NULL,
  tipo VARCHAR(20),
  descripcion VARCHAR(50),
  precio REAL DEFAULT 1.0,
  fabrica INTEGER DEFAULT NULL,
  PRIMARY KEY (codigo_producto),
  FOREIGN KEY (fabrica) REFERENCES FABRICAS(id_fabrica)
        ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT precio CHECK (precio>5)
);
```

IDENTITY

Auto increment en SQL Server se reemplaza por IDENTITY.

SENTENCIA ALTER

ALTER TABLE nombre_tabla ADD campo [tipo datos]

ALTER TABLE nombre_tabla DROP COLUMN campo

ALTER TABLE nombre_tabla ALTER COLUMN campo [tipo datos]

SENTENCIA ALTER - Ejemplo

```
ALTER TABLE Empleados ADD SegundoNombre varchar(50)

ALTER TABLE nombre_tabla DROP COLUMN Fecha_Nacimiento

ALTER TABLE nombre_tabla ALTER COLUMN Nombre
  varchar(50)
```

SENTENCIA DROP

DROP DATABASE Nombre_base_datos

DROP TABLE Nombre_tabla

SENTENCIA DROP- Ejemplo

```
DROP DATABASE rrhh

DROP TABLE Empleados
```

Comandos DML

- Comandos DML
  - Select: seleccionar registros
  - Insert: agregar registros
  - Update: modificar registros
  - Delete: eliminar registros
- Clausulas
  - from: especifica tablas
  - Where: especifica condiciones
  - Group by: separación por grupo
  - Having: filtro por grupo
  - Order by: ordena registros
- Operadores lógicos DML
  - and: “y” lógico
  - or: “o” lógico
  - not: negación
- Operadores de Comparación
  - between: intervalo de valores
  - like: comparación patrones (%)
  - in: especifica registros
  - >, <, =, >=, <=, <>

Comandos DML

SQL define cuatro sentencias de manipulación de datos principales:

- `INSERT`, para insertar registros en la base de datos
- `UPDATE` Encargado de modificar los valores de los campos indicados, en los registros que cumplan cierta condición
- `DELETE` Encargado de eliminar los registros de una tabla que cumplan una condición
- `SELECT` Encargado de consultar registros de la base de datos que satisfagan una condición determinada

Modificadores

- El uso de `UPDATE`, `DELETE` y `SELECT` necesita modificadores que indiquen a qué tuplas afectan.
- Ciertos modificadores (cláusulas) nos permiten generar criterios para definir los datos a manipular o selecionar.
- `FROM`: establece la tabla o tablas de la/s que seleccionar los registros
- `WHERE`: condiciones que los registros a seleccionar deben cumplir
- `GROUP BY`: criterio para agrupar los registros seleccionados
- `HAVING`: establece condiciones sobre datos calculados para los grupos generados por GROUP BY
- `ORDER BY`: ordena los registros seleccionados según el orden indicado

Operadores lógicos

Estos operadores permiten formar condiciones más complejas a partir de otras pasadas como operandos. Así, dadas tres condiciones (expresiones lógicas) *a*, *b* y *c*:

- *a* `AND` *b*: conjunción, evalúa las condiciones *a* y *b* y devuelve *verdad* si ambas son ciertas.
- *a* `OR` *b*: disyunción lógica, evalúa *a* y *b* y devuelve *verdad* si alguna de las dos es cierta.
- `NOT` *c*: negación lógica, devuelve el valor lógico contrario a la evaluación de *a*.

Operadores de comparación

Para cada condición simple, antes de agruparlas mediante los operadores lógicos anteriores, podemos emplear operadores como:

- = Igual que, <> Distinto de
- < Menor que, > Mayor que
- <= Menor o igual que (>= Mayor o igual que)
- `x BETWEEN a AND b` Devuelve los registros en los que el valor de campo *a* esté entre *a* y *b*, ambos inclusive.
- *LIKE* Utilizado en la comparación de un modelo
- *IN* Utilizado para especificar registros de una base de datos

SENTENCIA INSERT

```
INSERT INTO [nombre tabla] (columna1,columna2, …)
  VALUES (valor1, valor2, …) ;
```

SENTENCIA INSERT - Ejemplos

Tabla Empleados que tiene los campos Codigo, nombre, sueldo, ciudad, departamento

INSERT INTO empleados VALUES (1, ‘JAIME’, 40000, ‘Barcelona’, 1)

INSERT INTO empleados (codigo, departamento, nombre) VALUES (2, 1, ‘RAUL’)

Sintaxis UPDATE

```
UPDATE [nombre tabla]
SET [campo1]=[valor1], [campo2]=[valor2], …
WHERE [condición del registro];
```

Sintaxis UPDATE - Ejemplo

```
UPDATE empleado SET sueldo = sueldo + 10000
  WHERE departamento= 1;

UPDATE empleado SET sueldo = sueldo + 10000,
  ciudad = 'Madrid' WHERE codigo= 2
```

Sintaxis DELETE

```
DELETE FROM [tabla]
  WHERE [condición]
```

Sintaxis DELETE - Ejemplo

```
DELETE FROM empleado WHERE departamento=2;

DELETE FROM empleado WHERE sueldo <= 10000
```