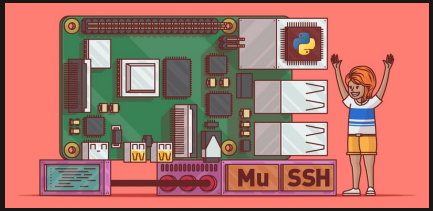
Introducción a bases de datos relacionales y SQLite

Introducción al Internet de las Cosas



Marco Teran
Universidad Sergio Arboleda

2023

Contenido

1. Introducción a bases de datos relacionales
   - Base de datos: DBMS
2. Modelo E/R
3. SQL

Introducción a bases de datos relacionales

Fichas de arcilla para llevar registro del contenido de los cargamentos

Cortes y muescas en palos de madera o nudos en cuerdas

3000 AC

Tablillas de arcilla: pictogramas, símbolos para describir venta de tierras y transacciones (pan, cerveza, ovejas, ganado y prendas de vestir)

Surgimiento de ciudades: Trueque, uso de moneda para comercio. Necesidad de llevar **registro**: datos para saber la producción.Después registro de calendarios, censos, matrimonios, contribuciones a la iglesia, etc.

1000 DC

Interés comercial: Grabado de datos en papel (contabilidad por partida doble)

1640

Máquina de Pascal: sumadora (precursora del odómetro automotriz)

Máquina tabuladora para procesamiento de datos (futuro IBM)

1805

Tarjetas perforadoras. Telar programable

1890

1950

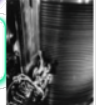
Primeros computadores electrónicos

Cinta magnética para guardado de datos

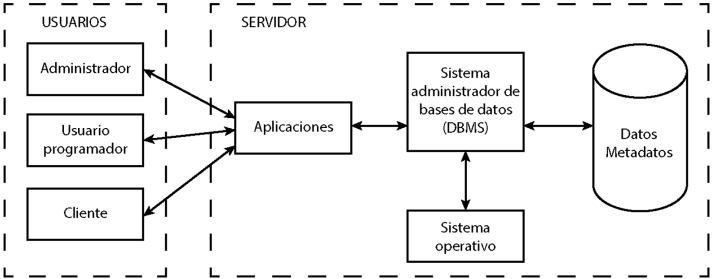
1953

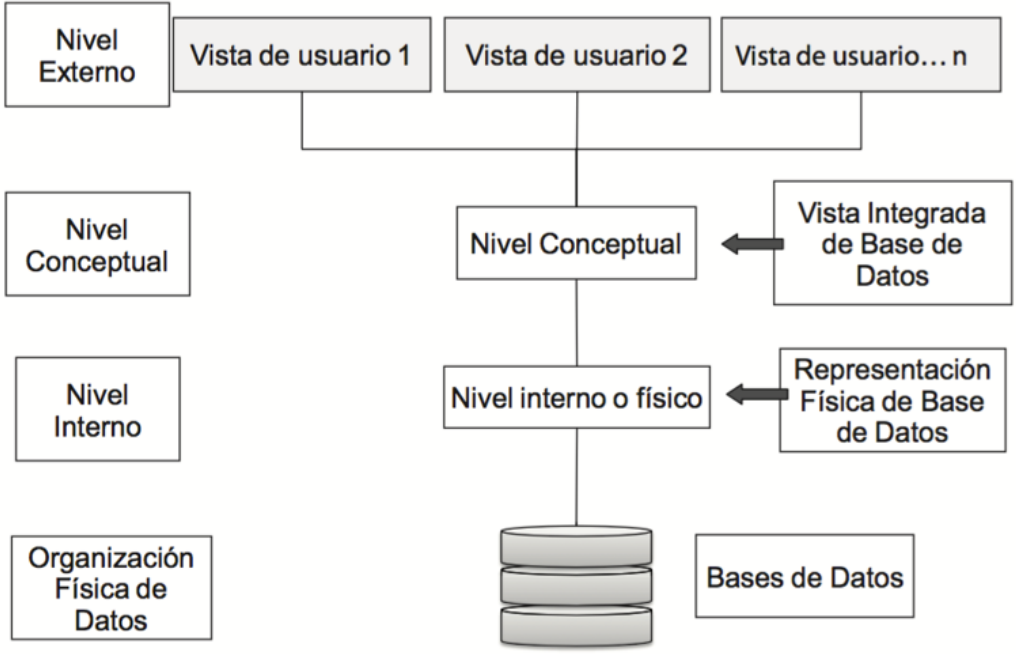
1956

Dispositivo de almacenamiento en disco



Base de datos




Nivel Externo
Vista de usuario 1
Vista de usuario 2
Vista de usuario… n
Nivel Conceptual
Nivel Conceptual
Vista Integrada de Base de Datos
Nivel Interno
Nivel interno o físico
Representación Física de Base de Datos
Organización Física de Datos
Bases de Datos

Base de datos: DBMS


MySQL®

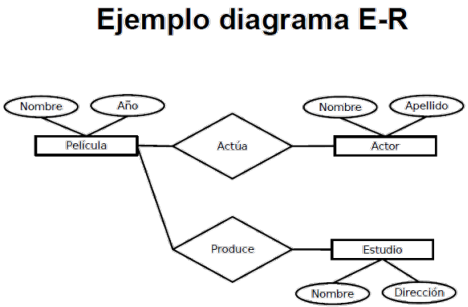

PostgreSQL


Microsoft®
SQL Server®


ORACLE
DATABASE

Modelo E/R

Modelo E/R

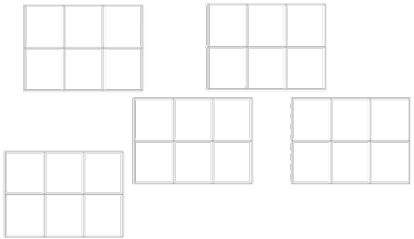


- Representación del modelado relacional de datos.
- Propuesto por Peter Chen en 1976.
- Elementos:
  - Entidad
  - Atributos
  - Relaciones
  - Restricciones

Modelo E/R y modelo Relacional

E/R **Relacional**

- Entidad → Relación (tabla)
- Relaciones → Relación/Llaves foráneas



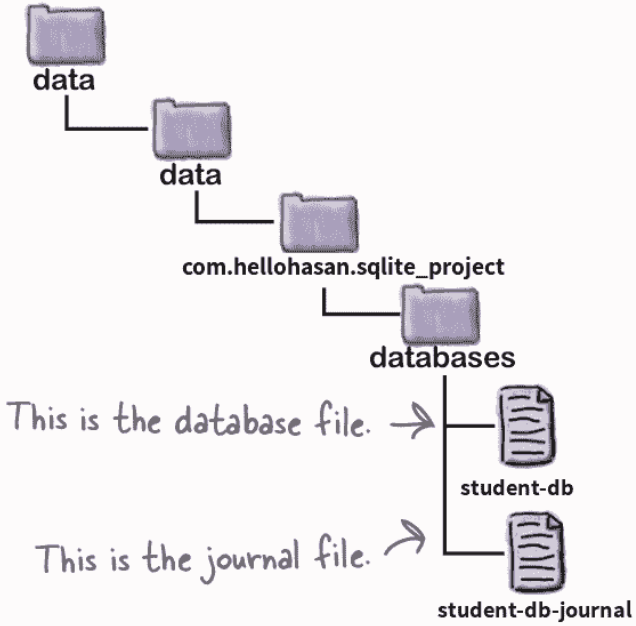
Modelo E/R y modelo Relacional

Diagrama E/R	Modelo Relacional
Relación 1 a 1	Atributos de una entidad pasan a ser atributos de otra.
Relación 1 a muchos	Llave primaria de la entidad con cardinalidad 1 pasa a ser llave foránea en la entidad con cardinalidad múltiple.
Relación muchos a muchos	Cada llave primaria de una entidad está presente como llave foránea en la otra entidad.

SQL

Bases de datos en SQL

- SQL (Structured Query Language) es un lenguaje estándar para manipular datos almacenados en bases de datos relacionales.
- Las bases de datos y las tablas deben ser creadas cuidadosamente para garantizar la eficiencia, la seguridad de los datos y la facilidad de uso.


data
data
com.hellohasan.sqlite_project
databases
This is the database file.
student-db
This is the journal file.
student-db-journal

Introducción a SQLite


SQLite

Introducción a SQLite

- SQLite es una biblioteca en lenguaje C que proporciona una base de datos relacional ligera en disco.
- No se necesita un servidor independiente para su funcionamiento, lo que la hace ideal para dispositivos integrados y para aplicaciones con necesidades de almacenamiento de datos simples.
- La estructura de una consulta SQLite es similar a SQL estándar, con el comando básico para crear una tabla como sigue:

```
CREATE TABLE nombre_tabla (columna1
tipo_dato, columna2 tipo_dato, ...);
```

Instalación de SQLite en la Raspberry Pi

- Para instalar SQLite en una Raspberry Pi, se utiliza el administrador de paquetes `apt-get`.
- Antes de la instalación, es recomendable actualizar el sistema con los siguientes comandos:
  ```
  sudo apt-get update
  sudo apt-get upgrade
  ```
- Posteriormente, se instala SQLite con el siguiente comando:
  ```
  sudo apt-get install sqlite3
  ```
- Tras la instalación, se puede verificar que SQLite esté instalado correctamente ejecutando:
  ```
  sqlite3 --version
  ```
- Este comando devolverá la versión instalada de SQLite, confirmado así su correcta instalación.
- SQLite ahora está listo para ser utilizado en la Raspberry Pi.

Creación de bases de datos en SQL

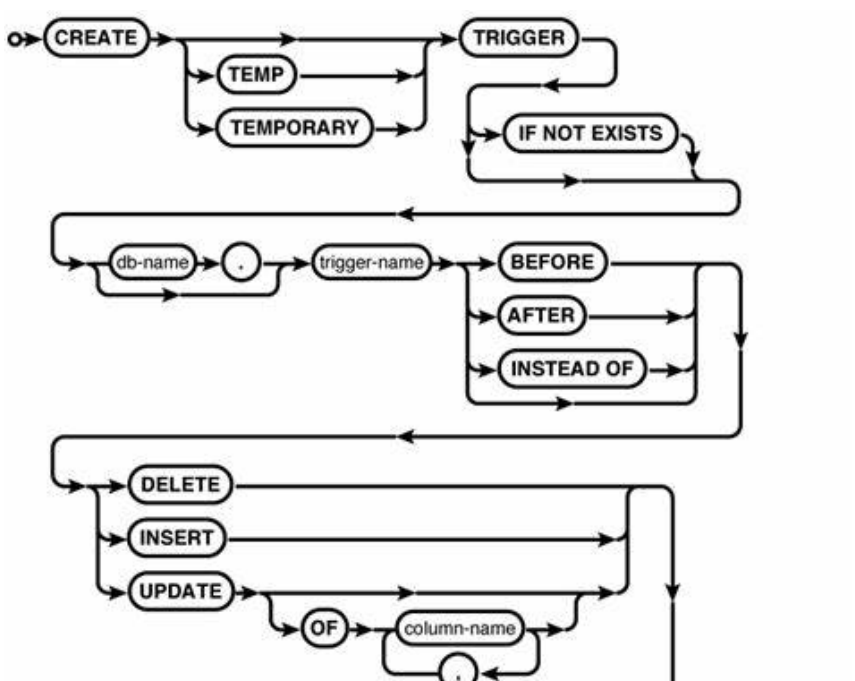
- Para crear una nueva base de datos en SQL, se utiliza el comando CREATE DATABASE seguido del nombre de la base de datos deseado.
- **Ejemplo:** `CREATE DATABASE MyDatabase;`

Creación de bases de datos en SQL

- Una vez creada la base de datos, se pueden crear tablas dentro de ella utilizando el comando CREATE TABLE seguido de la definición de la tabla.
- **Ejemplo:** `CREATE TABLE Students (ID INT PRIMARY KEY NOT NULL, Name TEXT NOT NULL, Age INT NOT NULL, Address CHAR(50));`

```
CREATE TABLE nombre_tabla (
        nombre_columna1 datatype NULL,
        nombre_columna2 datatype NOT NULL,
        nombre_columna3 datatype NULL,
        …
);
```

```
CREATE TABLE Cliente (
        cedula      int(10)          NOT NULL,
        nombre      varchar(30)      NOT NULL,
        apellido    varchar(30)      NOT NULL,
        email       varchar(30)      NULL,
        cargo       varchar(15)      NOT NULL
);
```


CREATE
TEMP
TEMPORARY
TRIGGER
IF NOT EXISTS
db-name
.
trigger-name
BEFORE
AFTER
INSTEAD OF
DELETE
INSERT
UPDATE
OF
column-name
,

Uso de PRIMARY KEY en SQL

- En SQL, PRIMARY KEY es un tipo de restricción que se utiliza para identificar de manera única cada fila en una tabla.
- Cada tabla en una base de datos SQL debe tener una columna (o conjunto de columnas) designada como PRIMARY KEY.

Uso de PRIMARY KEY en SQL

- **Ejemplo:** `CREATE TABLE Students (ID INT PRIMARY KEY NOT NULL, Name TEXT NOT NULL, Age INT NOT NULL, Address CHAR(50));`
- En este ejemplo, ID es la PRIMARY KEY. Esto significa que cada estudiante debe tener un ID único.

Uso de PRIMARY KEY en SQL

- La restricción PRIMARY KEY evita que dos filas tengan el mismo valor en la columna clave.
- Además, una tabla no puede tener más de una PRIMARY KEY.
- SQLite soporta el uso de PRIMARY KEYs al igual que otros sistemas de bases de datos SQL.
- El uso de PRIMARY KEYs es esencial para mantener la integridad de los datos en una base de datos SQL.

```
CREATE TABLE Cliente (
        cedula      int(10)          NOT NULL    PRIMARY KEY,
        nombre      varchar(30)      NOT NULL,
        apellido    varchar(30)      NOT NULL,
        email       varchar(30)      NULL,
        cargo       varchar(15)      NOT NULL
);
```

```
CREATE TABLE Cliente (
        cedula      int(10)          NOT NULL,
        nombre      varchar(30)      NOT NULL,
        apellido    varchar(30)      NOT NULL,
        email       varchar(30)      NULL,
        cargo       varchar(15)      NOT NULL,
        PRIMARY KEY (cedula, nombre, cargo)
);
```

Valores por defecto y listas de valores en SQLite

- En SQLite, al definir las columnas de una tabla, se puede especificar un valor por defecto para una columna utilizando la cláusula DEFAULT.

Valores por defecto y listas de valores en SQLite

- **Ejemplo:** `CREATE TABLE Students (ID INT PRIMARY KEY NOT NULL, Name TEXT NOT NULL, Age INT NOT NULL DEFAULT 18, Address CHAR(50));`
- En este ejemplo, si no se especifica un valor para la columna Age al insertar una nueva fila, SQLite automáticamente asignará el valor 18.

```
CREATE TABLE pais (
        id              int(10)         AUTOINCREMENT           PRIMARY KEY,
        nombre          varchar(30)             NOT NULL    DEFAULT '',
        continente      enum('Asia', 'Europa`,`Oceania`,`America`,`Antartica`)
                        NOT NULL    DEFAULT  'Asia',
);
```

Valores por defecto y listas de valores en SQLite

- Además, SQLite permite definir una lista de valores posibles para una columna mediante la cláusula CHECK.

Valores por defecto y listas de valores en SQLite

- **Ejemplo:** `CREATE TABLE Students (ID INT PRIMARY KEY NOT NULL, Name TEXT NOT NULL, Age INT CHECK(Age>=18 AND Age<=120) NOT NULL, Address CHAR(50));`
- En este ejemplo, SQLite sólo permitirá insertar valores entre 18 y 120 para la columna Age.
- Tanto la cláusula DEFAULT como la cláusula CHECK son útiles para mantener la integridad de los datos en una base de datos SQLite.

Borrar tablas en SQLite

- En SQLite, la eliminación de una tabla existente se realiza utilizando la sentencia DROP TABLE.
- **Ejemplo de sintaxis:** `DROP TABLE table_name;`
- **Ejemplo:** Si se desea eliminar la tabla **Students**, se utilizaría la sentencia: `DROP TABLE Students;`

Borrar tablas en SQLite

- Es importante tener en cuenta que esta operación es irreversible y eliminará todos los datos almacenados en la tabla.
- Por ende, se recomienda siempre hacer una copia de seguridad de los datos importantes antes de ejecutar la sentencia DROP TABLE.
- Adicionalmente, SQLite proporciona la sentencia DROP TABLE IF EXISTS, que elimina una tabla sólo si existe. Esto puede ser útil para evitar errores si se intenta eliminar una tabla que no existe.
- **Ejemplo:** `DROP TABLE IF EXISTS Students;`

Adicionar Columnas en SQLite

- SQLite permite añadir nuevas columnas a una tabla existente utilizando la sentencia ALTER TABLE.
- La sintaxis general para añadir una columna es: `ALTER TABLE table_name ADD COLUMN column_name column_type;`
- Por ejemplo, para añadir una columna de **Email** de tipo `TEXT` a la tabla **Students**, la sentencia sería: `ALTER TABLE Students ADD COLUMN Email TEXT;`

Agregar una columna:

```
ALTER TABLE nombre_tabla
Add nombre_columna datatype NULL
```

Agregar varias columnas:

```
ALTER TABLE nombre_tabla
Add nombre_columna     datatype NULL,
      nombre_columna2  datatype NULL
```

Adicionar Columnas en SQLite

- Hay que tener en cuenta que esta operación modifica la estructura de la tabla y puede afectar a las consultas existentes.
- SQLite permite agregar una columna que no permita valores NULL, siempre y cuando se especifique un valor por defecto. Por ejemplo: `ALTER TABLE Students ADD COLUMN BirthDate TEXT NOT NULL DEFAULT 'Unknown';`
- A diferencia de otros sistemas de gestión de bases de datos, SQLite no permite eliminar o modificar columnas existentes con la sentencia ALTER TABLE.

Inserción de Datos en SQLite

- Para insertar datos en una tabla en SQLite, se utiliza la sentencia INSERT INTO.
- La sintaxis general para insertar datos es: `INSERT INTO table_name(column1, column2, column3, ...) VALUES (value1, value2, value3, ...);`

Inserción de Datos en SQLite

- Por ejemplo, para insertar una fila en la tabla "Students"con los valores "John Doe", "johndoe@example.com" "1990-01-01"para las columnas "Name", "Email" "BirthDaterespectivamente, la sentencia sería: `INSERT INTO Students(Name, Email, BirthDate) VALUES ('John Doe', 'johndoe@example.com', '1990-01-01');`
- Esta sentencia insertará una nueva fila en la tabla "Students". Los valores proporcionados se asignarán a las columnas

```
INSERT INTO nombre_tabla (nombre_atributo1, nombre_atributo2, …)
        VALUES (valor_atributo1, valor_atributo2, …);
```

Esta sintaxis se puede usar cuando los valores de los atributos se pasan en el orden en que están en la tabla.

```
INSERT INTO nombre_tabla
        VALUES (valor_atributo1, valor_atributo2, …);
```

Inserción de Datos en SQLite

- Cabe destacar que si no se especifican las columnas (solo se usa `INSERT INTO table_name VALUES (...);`), los valores se deben proporcionar en el mismo orden que las columnas en la definición de la tabla.

ID	Nombre	Dirección
1	Juan Días	Calle 2 # 3-4
2	Daniel Pardo	Calle 5 # 7-8
3	Stephen King	Calle 1 # 1-2

```
INSERT INTO clientes (ID, Dirección, Nombre)
        VALUES (4, “Cra 90 # 1-1’, ‘Diego Vega’ );
```

```
INSERT INTO clientes
        VALUES (5, ‘Peter Parker’, ‘Cra 9 # 2-2’);
```

ID	Nombre	Dirección
1	Juan Días	Calle 2 # 3-4
2	Daniel Pardo	Calle 5 # 7-8
3	Stephen King	Calle 1 # 1-2
4	Diego Vega	Cra 90 # 1-1
5	Peter Parker	Cra 9 # 2-2

Borrar Datos en SQLite

- Para borrar datos de una tabla en SQLite, se utiliza la sentencia DELETE.
- La sintaxis general para borrar datos es: `DELETE FROM table_name WHERE condition;`

Borrar Datos en SQLite

- Por ejemplo, para borrar una fila en la tabla **Students** donde el **Name** es **John Doe**, la sentencia sería: `DELETE FROM Students WHERE Name = 'John Doe';`
- Esta sentencia eliminará todas las filas en la tabla **Students** donde el **Name** es **John Doe**. Si se omite la cláusula `WHERE`, todas las filas de la tabla serán eliminadas.

Borrar Datos en SQLite

- Es importante tener precaución al usar la sentencia `DELETE` sin la cláusula `WHERE`, ya que puede resultar en la pérdida de todos los datos en la tabla.

SQL para borrar una tabla:

```
DELETE FROM nombre_tabla;
```

SQL para borrar todos los registros de una tabla que cumplen con una condición:

```
DELETE FROM cliente WHERE país = 'Francia';
```

Actualizar Datos en SQLite

- Por ejemplo, para actualizar la edad de un estudiante llamado **John Doe** en la tabla **Students**, la sentencia sería: `UPDATE Students SET Age = 20 WHERE Name = 'John Doe';`
- Esta sentencia modificará el valor de la columna **Age** para todas las filas en la tabla **Students** donde el **Name** es **John Doe**. Si se omite la cláusula `WHERE`, todos los registros de la tabla serán actualizados.
- Es crucial ser preciso al usar la sentencia `UPDATE`, especialmente con la cláusula `WHERE`, para evitar cambios no deseados en los datos.

Actualizar Datos en SQLite

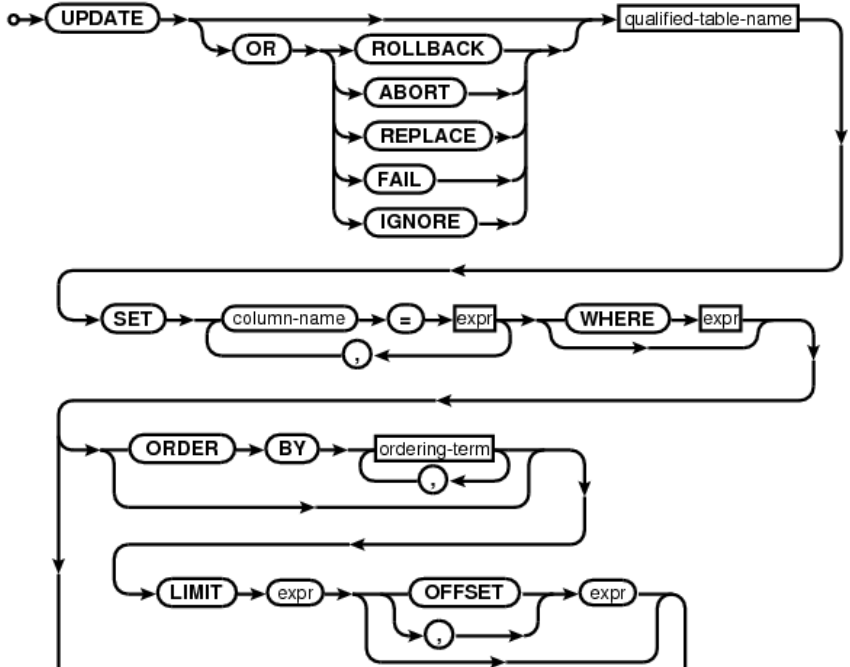
- Para actualizar datos en una tabla en SQLite, se utiliza la sentencia `UPDATE`.
- La sintaxis general para actualizar datos es: `UPDATE table_name SET column1 = value1, column2 = value2,... WHERE condition;`

SQL para actualizar un registro de una tabla:

```
UPDATE nombre_tabla
SET atributo1 = valor1, atributo2 = valor 2, …
WHERE atributoN = valorN;
```

Ejemplo: Actualizar en la base de datos de clientes, aquellos que tenían en nacionalidad ‘Holanda’ por ‘Paises Bajos’.

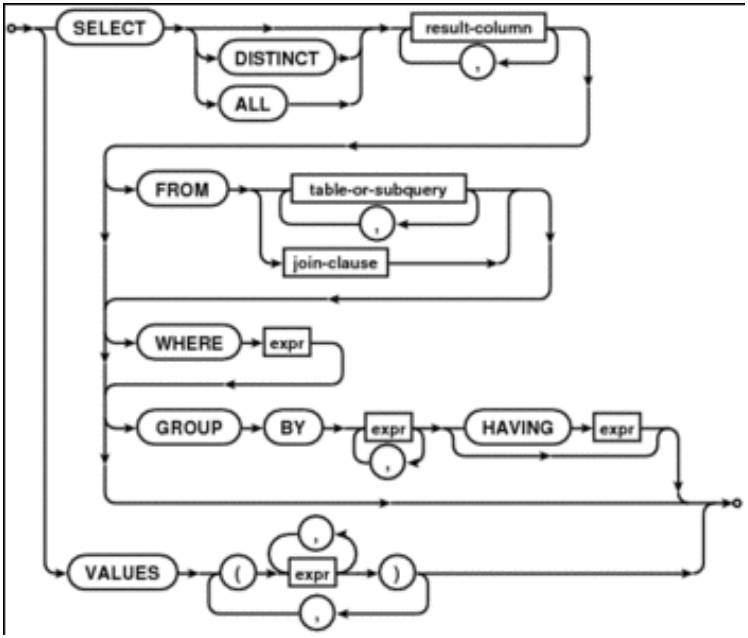
```
UPDATE clientes
SET nacionalidad = ‘Paises Bajos’
WHERE nacionalidad = ‘Holanda’;
```


UPDATE
OR
ROLLBACK
ABORT
REPLACE
FAIL
IGNORE
qualified-table-name
SET
column-name
=
expr
,
WHERE
expr
ORDER
BY
ordering-term
,
LIMIT
expr
OFFSET
,
expr

Realizar el siguiente tutorial

Una excelente forma de recordar las principales operaciones del lenguaje SQL es realizar el siguiente tutorial

https://www.khanacademy.org/computing/computer-programming/sql/


SELECT
DISTINCT
ALL
result-column
,
FROM
table-or-subquery
,
join-clause
WHERE
expr
GROUP
BY
expr
,
HAVING
expr
VALUES
(
,
expr
)
,

¡Muchas gracias por su atención!

¿Preguntas?

Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: marco.teran@usa.edu.co

UNIVERSIDAD
SERGIO ARBOLEDA