Introducción
SQL y SQLITE
Bases de datos relacionales

Autores: Esp. Ing. Ernesto Gigliotti. UTN-FRA
Lic. Danilo Zecchin. UNLP

Definiciones

Base de datos

- Es una colección organizada de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

- Generalmente, los datos son almacenados y accedidos de forma electrónica a través de una computadora

Definiciones

DBMS (Database Management System)

- Conjunto integrado de componentes de software que permiten a los usuarios interactuar con una o más bases de datos y proveen acceso a todos los datos contenidos en ella.

- Interactúa con usuarios finales, aplicaciones y la base de datos para capturar y analizar los datos.

Definiciones

DBMS (Database Management System)

- La funcionalidad provista por un DBMS se clasifica en cuatro grupos funcionales:

  1)Definición de datos

  2)Actualización (altas, bajas, modificaciones)

  3)Recuperación

  4)Administración

Clasificación de las BD

Basada en el modelo de datos:

- *Bases de datos relacionales*
- Bases de datos orientadas a objetos
- Bases de datos jerárquicas
- Bases de datos de red
- Bases de datos NoSQL

Basada en la distribución de los datos:

- Bases de datos centralizadas
- Bases de datos distribuidas
  - Sistemas homogéneos
  - Sistemas heterogéneos

Bases de datos relacionales

- Basadas en el modelo relacional propuesto e introducido por E.F. Codd en 1970
- Ampliamente utilizadas durante las últimas 5 décadas
- Centralizadas o distribuidas
- Modelo cliente-servidor
- Lenguaje de consulta estructurado (SQL)
- RDBMS

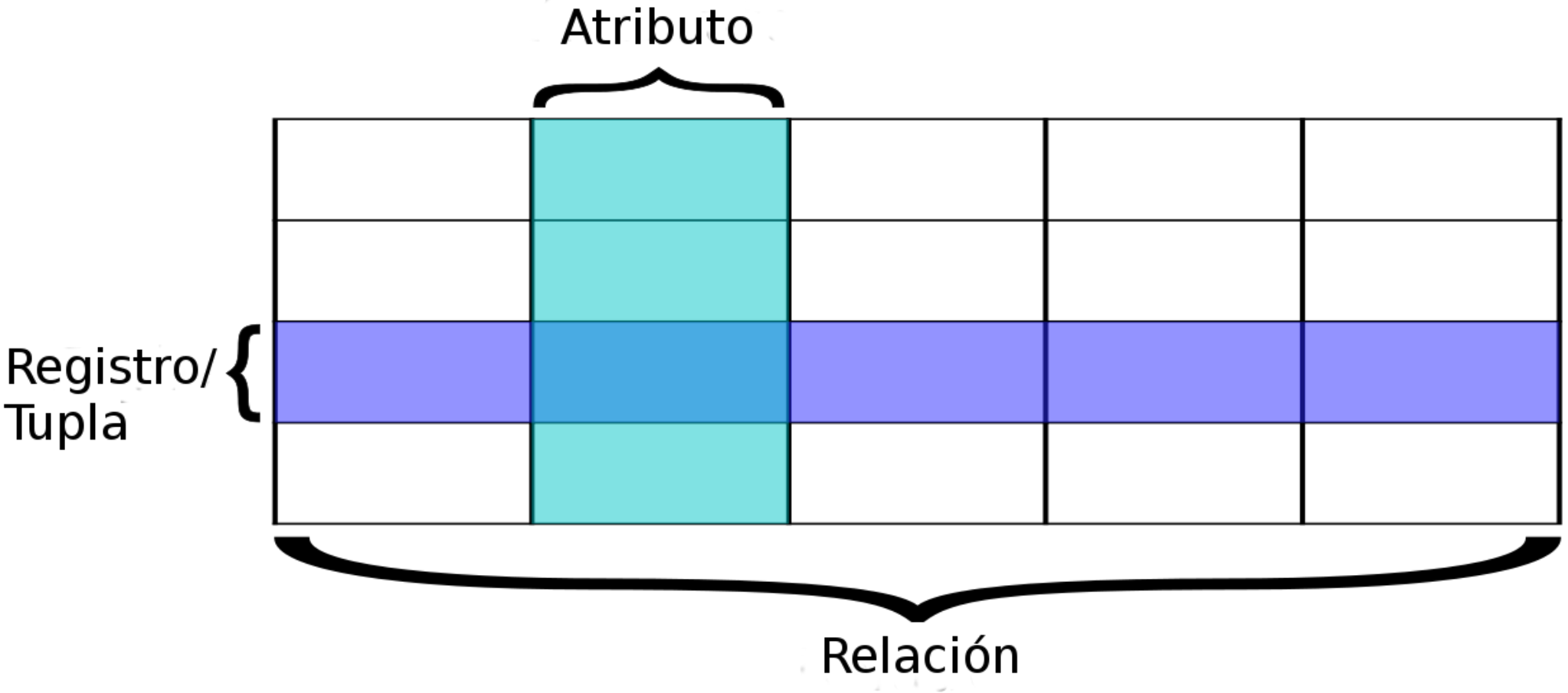
Bases de datos relacionales

Organización de los datos

- Una o más tablas (relaciones) compuestas por filas y columnas
- Cada tabla/relación representa un tipo de entidad
- Las filas usualmente son llamadas registros o tuplas y las columnas atributos
- Las filas representan instancias del tipo de entidad y las columnas representan los valores de los atributos de dicha instancia
- Cada tabla dispone una clave que permite identificar univocamente cada registro

Bases de datos relacionales

Organización de los datos



Bases de datos relacionales

Ejemplo

id-cliente	*nombre-cliente*	*calle-cliente*	*ciudad-cliente*
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
18.273.609	Abril	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Ramblas	León
01.928.374	Gómez	Carretas	Cerceda

(a) La tabla *cliente*

número-cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

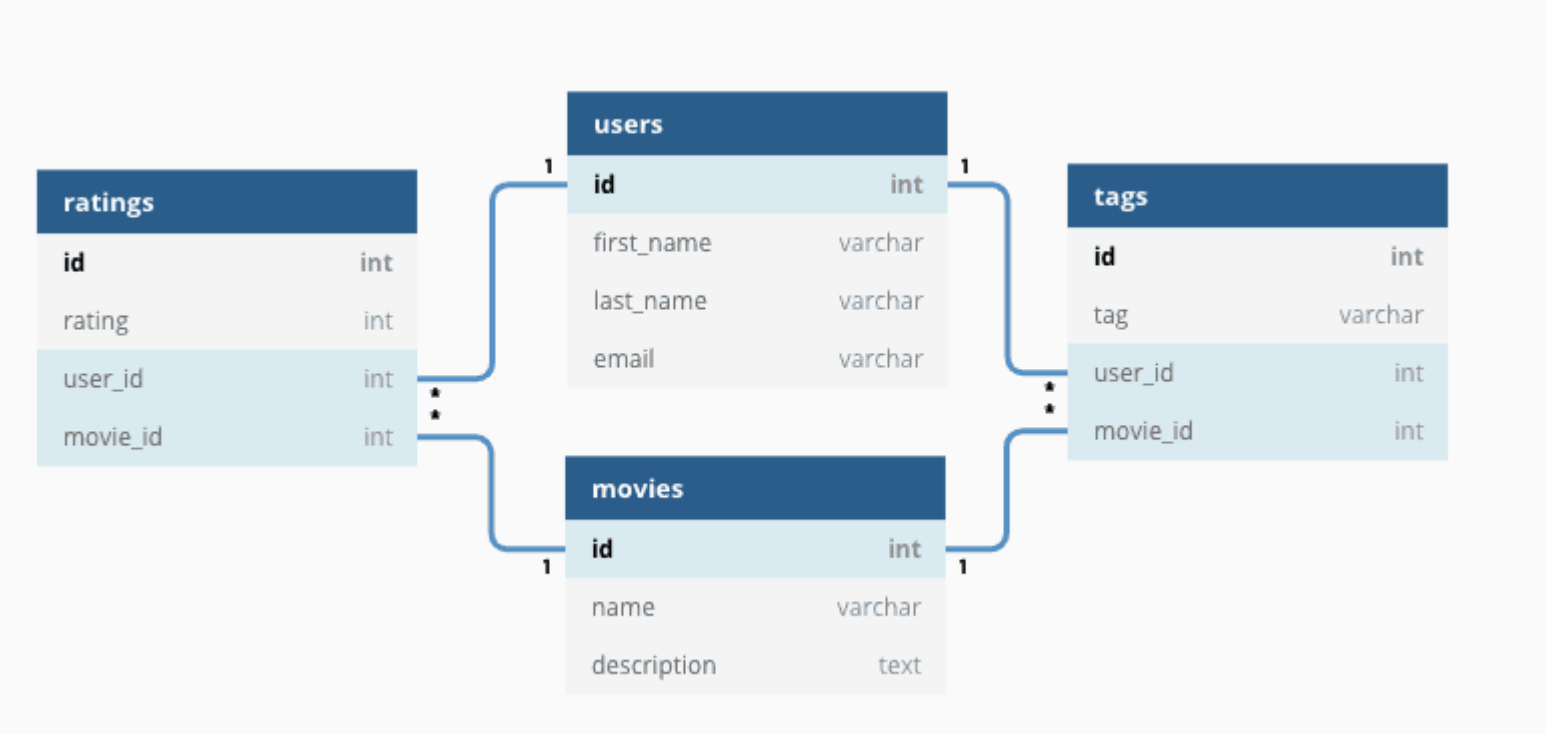
(b) La tabla *cuenta*

id-cliente	*número-cuenta*
19.283.746	C-101
19.283.746	C-201
01.928.374	C-215
67.789.901	C-102
18.273.609	C-305
32.112.312	C-217
33.666.999	C-222
01.928.374	C-201

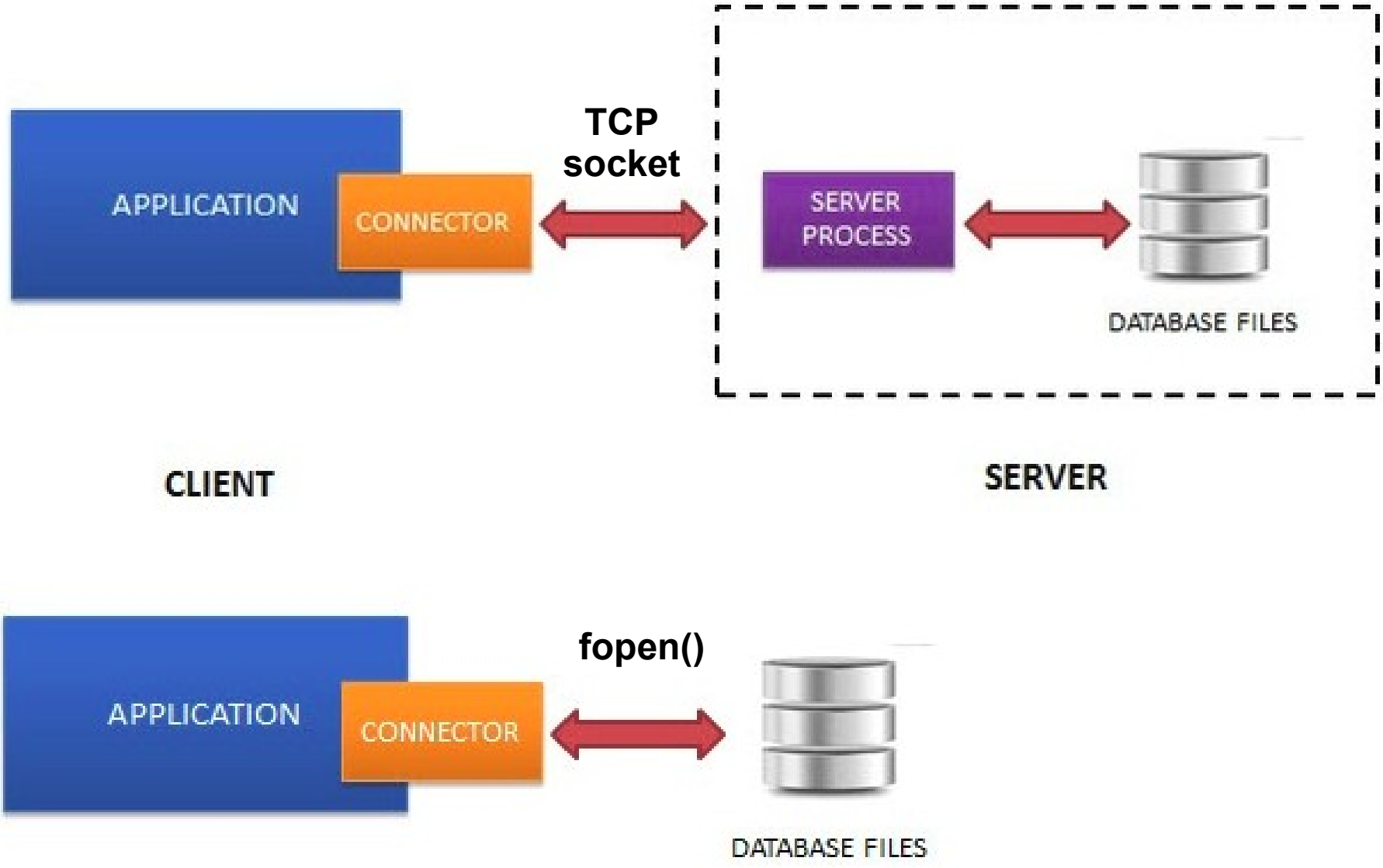
(b) La tabla *impositor*

Bases de datos relacionales

Ejemplo



- Diferencias MySQL y SQLite



SQLiTE

- **No utiliza una conexión cliente-servidor** (generalmente TCP) como otros motores de base de datos.
- El motor de **SQLite se enlaza con el programa** pasando a ser parte integral del mismo.
- El programa utiliza la funcionalidad de SQLite a través de **llamadas simples** a subrutinas y funciones.
- El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como **un sólo archivo** estándar en la máquina host.

- SQLITE asigna **un tipo de dato para cada campo**. Se puede insertar un String en una columna de tipo entero.

  - **NULL:** Vacío.
  - **INTEGER:** Número entero, puede ocupar 1, 2, 3, 4, 6, o 8 bytes dependiendo de la magnitud del valor.
  - **REAL:** Número de punto flotante, el tamaño es 8bytes siguiendo el standard IEEE.
  - **TEXT:** Información en formato texto codificado UTF-8, UTF-16BE o UTF-16LE.
  - **BLOB:** Información almacenada como Bytes sin interpretación (por ejemplo una imagen).

Lenguaje SQL

- Es un lenguaje de acceso a bases de datos.
- Permite especificar diversos tipos de operaciones.
- Permite efectuar consultas para recuperar información.
- Permite hacer cambios sobre la base.
- Permite ingresar nueva información a la base
- Este lenguaje esta dividido en dos partes:

- **DDL :** Permite manipular la estructura de la base de datos (crear,borrar,modificar tablas,etc.)

- **DML :** Permite manipular la información que se encuentra almacenada dentro de al base de datos.

- Sentencia **SELECT :** Permite realizar consultas en una o más tablas bajo una condición dada.

Formato:

```
SELECT nombre de las columnas separados por coma
FROM tabla
```

Ejemplo:

```
SELECT nombre,apellido
FROM Empleados
```

- Para filtrar la búsqueda bajo algún criterio, se utiliza la clausula **WHERE**:

Ejemplo:

```
SELECT nombre,apellido
FROM Empleados
WHERE nombre = "Lucas"
```

- Sentencia **INSERT**: Una sentencia INSERT de SQL agrega uno o más registros (filas) a una (y sólo una) tabla en una base de datos relacional.

Formato:

```
INSERT INTO tabla (columna1, [columna2,...])
VALUES (valor1, [valor2,...])
```

Ejemplo:

```
INSERT INTO Empleados (Nombre, Edad)
VALUES ('Juan', 25);
```

- Sentencia **UPDATE**: Una sentencia UPDATE de SQL es utilizada para actualizar los valores de un conjunto de registros existentes en una tabla.

Formato:

```
UPDATE tabla
SET columna = valor
WHERE condición;
```

Ejemplo:

```
UPDATE Empleados
SET Edad = 25
WHERE Nombre = 'Juan';
```

NOTA : Si no se coloca la sentencia WHERE, se modificarán todas las filas de la tabla.

- Sentencia **DELETE**: Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.

Formato:

```
DELETE FROM tabla
WHERE condición
```

Ejemplo:

```
DELETE FROM Empleados
WHERE Edad<=18
```

IMPORTANTE: Si no se coloca la sentencia WHERE, se eliminarán todas las filas de la tabla.

- Sentencia **ORDER BY:** Se utiliza para ordenar de forma ascendente o descendente por alguna columna.

- Luego del nombre de la columna puede colocarse DESC o ASC.

Formato:

```
SELECT * FROM tabla
WHERE condición
ORDER BY col0 DESC
```

- Sentencia **LIMIT:** Se utiliza para limitar la cantidad de resultados que se obtienen de la consulta.
- Puede tener en forma opcional un offset para que no devuelva desde el primer valor. (comienza de cero)

Formato:

```
SELECT * FROM tabla
WHERE condición
ORDER BY col0 DESC
LIMIT N
```

Formato:

```
SELECT *
FROM tabla
WHERE condición
ORDER BY col0 DESC
LIMIT N
```

- Sentencia **INNER JOIN:** Se utiliza para combinar dos o más tablas mediante alguna condición que relaciona registros de las mismas.

- Solo permanecen como resultado de la consulta, los registros que satisfacen la condición que se define mediante “ON”.

Formato:

```
SELECT *
FROM tabla INNER JOIN tabla2
ON (condición que relaciona registros de ambas tablas)
WHERE condición
```

- Sentencia **INNER JOIN**

Ejemplo:

```
SELECT *
FROM Empleados e INNER JOIN Puesto p
ON (e.id_puesto=p.id_Puesto)
WHERE Edad<=18
```

- La consulta arrojará los valores de los registros para cada empleado junto con la información del puesto que corresponde a cada empleado según el id_puesto que éstos posean.

- **Funciones:** Permiten hacer cálculos con las columnas de las tablas.

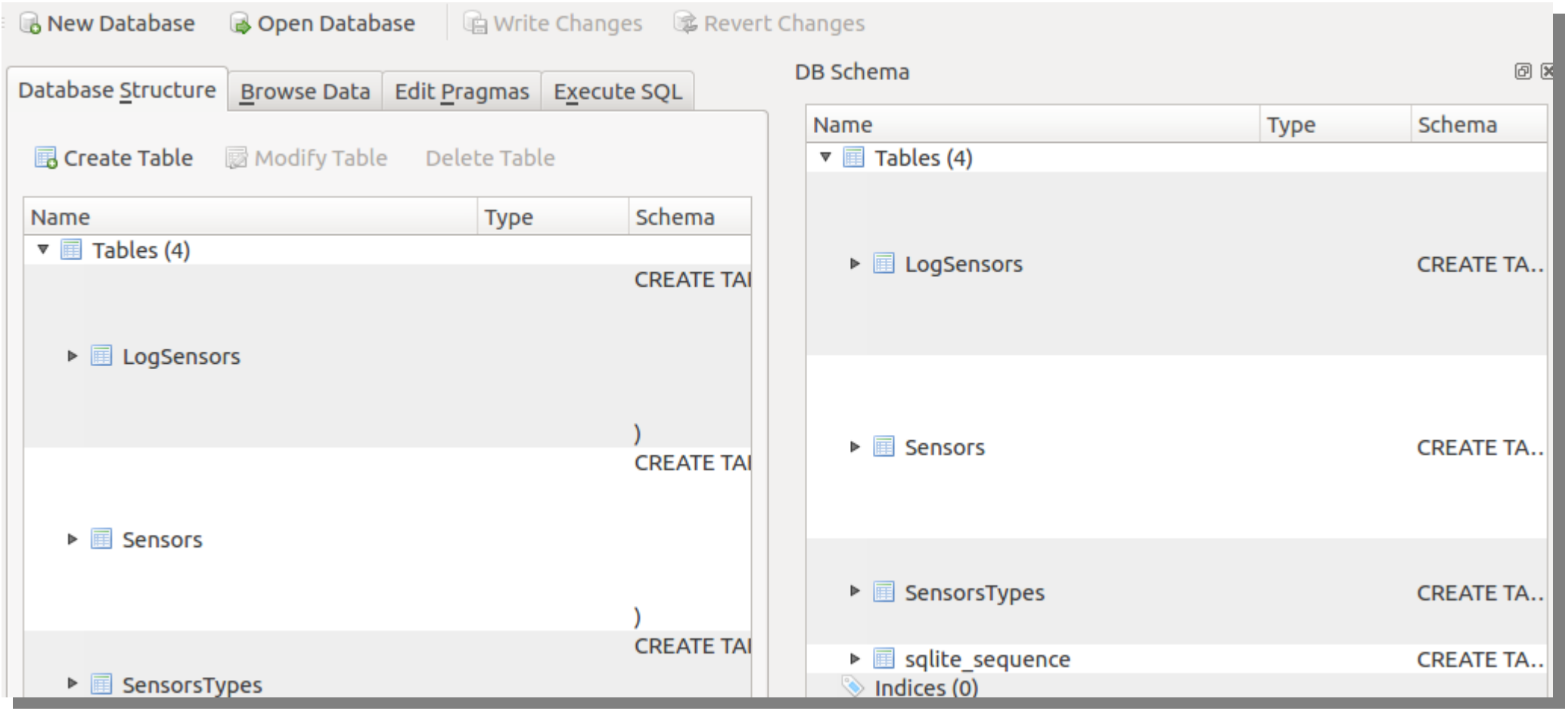
  - count(*columna*)
  - avg(*columna*)
  - max(*columna*)
  - min(*columna*)
  - datetime(*columna*)

- Ejemplo:

```
SELECT avg(Edad) as prom_edad
FROM Empleados
```

- DB Browser for SQLITE

```
sudo aptitude install sqlitebrowser
```



Bibliografía

- https://www.sqlite.org/index.html
- https://docs.python.org/2/library/sqlite3.html
- https://www.sqlite.org/lang_datefunc.html
- Grant Allen and Mike Owens. The Definitive Guide to SQLite.2010. Apress.