

lenguaje SQL II

Alex Di Genova

19/05/2022

Contenidos

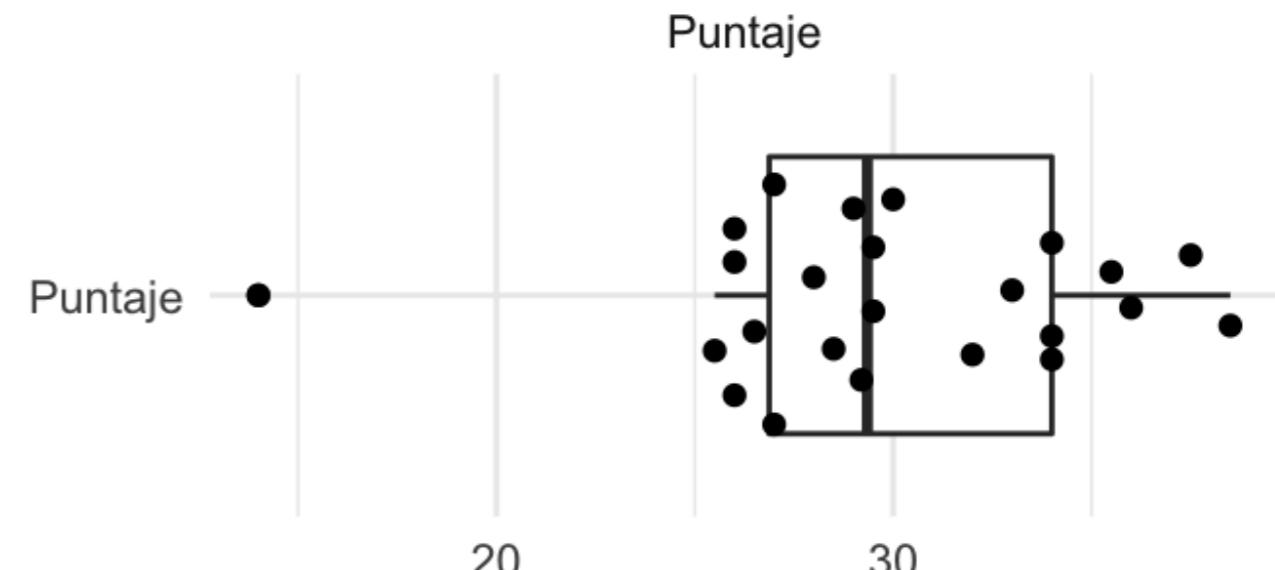
- Notas control I
- Repaso
- Algebra relacional
- SQL II

Notas Control I

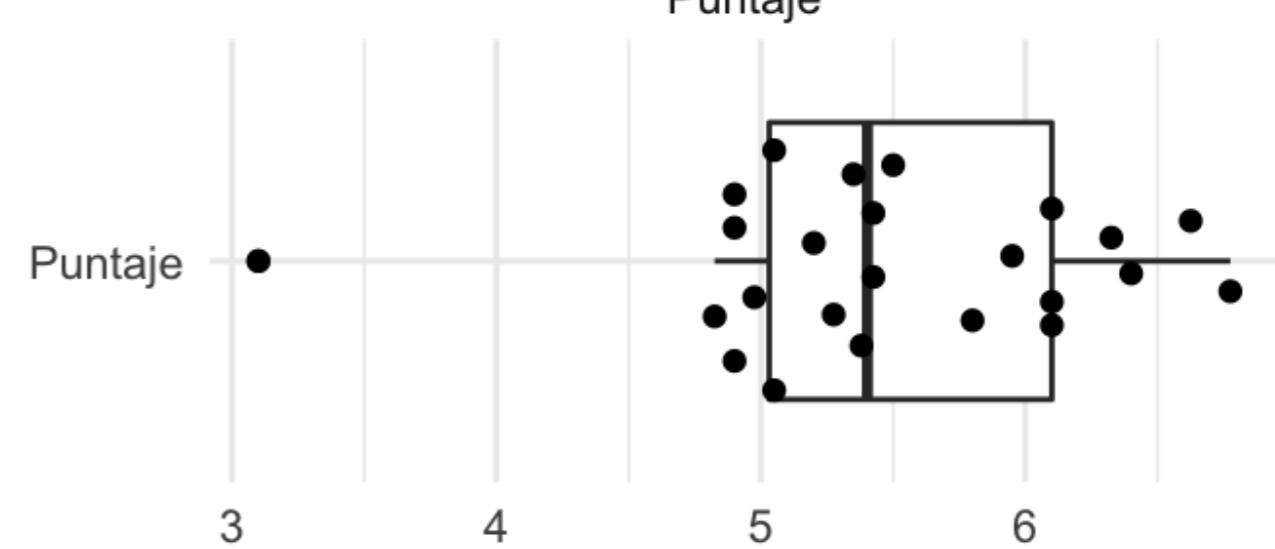
Preguntas



Puntaje



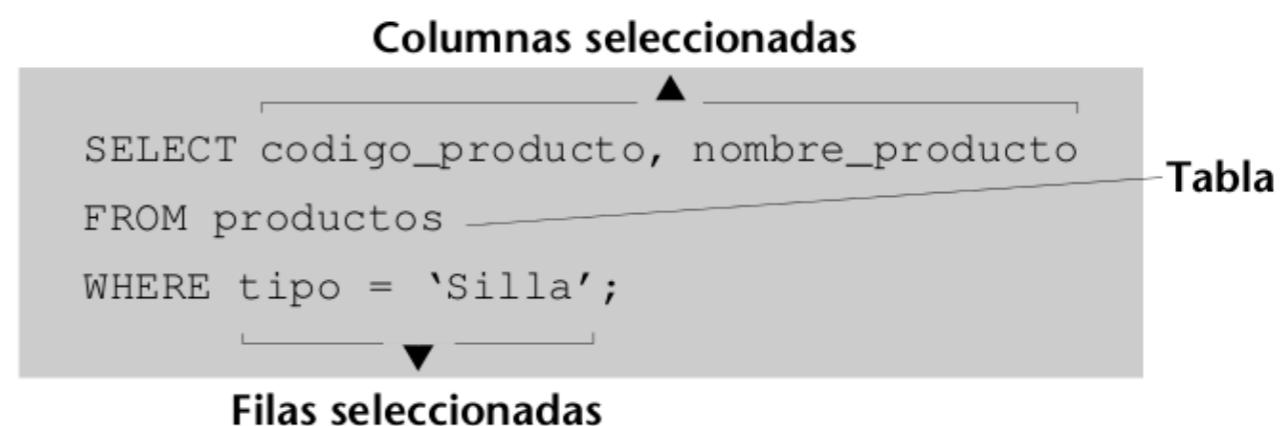
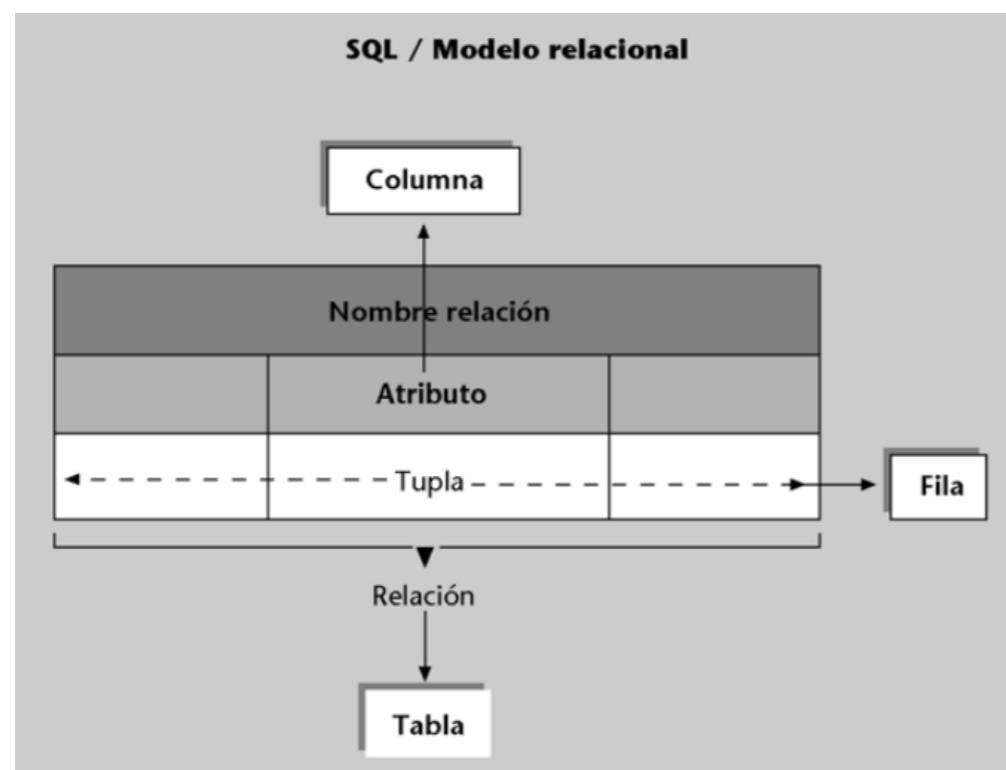
Notas



Qué es el SQL?

Introducción

- Lenguaje estándar ANSI(1986) de **definición, manipulación y control** de bases de datos relacionales.
- Estándar de facto en la actualidad, aunque cada fabricante agrega sus propias extensiones.



SQL

Introducción



SQL

Tipos de datos

- **CHARACTER** (largo) : Cadenas de caracteres de longitud fija.
- **INTEGER** : Números enteros.
- **SMALLINT** : Números enteros pequeños.
- **DECIMAL** (precisión,escala) : Número de decimales con tantos dígitos como indique la precisión y tantos decimales como indique la escala.
- **DATE**: Fechas. Están compuestas de año (YEAR), mes (MONTH) y día (DAY)
- **BOOLEAN**: Valores verdadero o falso.

TYPE	SQLite3
INT	
INTEGER	INTEGER
TINYINT	
SMALLINT	
MEDIUMINT	
BIGINT	
UNSIGNED BIG INT	
INT2	
INT8	
CHARACTER(20)	
VARCHAR(255)	
VARYING CHARACTER(255)	
NCHAR(55)	TEXT
NATIVE CHARACTER(70)	
NVARCHAR(100)	
TEXT	
CLOB	
BLOB	BLOB
REAL	
DOUBLE	
DOUBLE PRECISION	
FLOAT	
NUMERIC	
DECIMAL(10,5)	
BOOLEAN	
DATE	
DATETIME	NUMERIC

SQL

DDL- Data Definition Language

- **DDL (lenguaje de definición de datos)**
 - Las sentencias DDL son aquellas utilizadas para la creacion de una base de datos y todos sus componentes.
 - Tablas, indices, relaciones, restricciones...
- **Sentencias**
 - CREATE -> crear tablas y vistas.
 - DROP -> para borrar tablas y vistas.
 - ALTER -> para modificar tablas.
 - ALTER TABLE EMPLEADO ADD COLUMN sueldo INT;
 - ALTER TABLE EMPLEADO DROP COLUMN mail CASCADE;
 - ALTER TABLE DEPARTAMENTO ALTER COLUMN id_director SET DEFAULT '00000000';

SQL

Sentencia CREATE

Restricciones de columna	
Restricción	Descripción
NOT NULL	La columna no puede tener valores nulos.
UNIQUE	La columna no puede tener valores repetidos. Es una clave alternativa.
PRIMARY KEY	La columna no puede tener valores repetidos ni nulos. Es la clave primaria.
REFERENCES tabla [(columna)]	La columna es la clave foránea de la columna de la tabla especificada.
CHECK (condiciones)	La columna debe cumplir las condiciones especificadas.

FOREIGN KEY(trackartist) REFERENCES artist(artistid)

Restricciones de tabla	
Restricción	Descripción
PRIMARY KEY (columna [, columna...])	El conjunto de las columnas especificadas no puede tener valores nulos ni repetidos. Es una clave primaria.
FOREIGN KEY (columna [, columna...]) REFERENCES tabla [(columna2 [, columna2...])]	El conjunto de las columnas especificadas es una clave foránea que referencia la clave primaria formada por el conjunto de las columnas2 de la tabla dada. Si las columnas y las columnas2 se denominan exactamente igual, entonces no sería necesario poner columnas2.
CHECK (condiciones)	La tabla debe cumplir las condiciones especificadas.

FOREIGN KEY clave_secundaria REFERENCES tabla [(clave_primaria)]
[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

SQL

DDL- Data Definition Language

The screenshot shows a GitHub repository page for [adigenova/uohdb](#). The repository is public and contains several Jupyter Notebooks:

File	Description	Last Commit
Basic_SQL.ipynb	moving code	2 months ago
Chinook_db.ipynb	Created using Colaboratory	2 months ago
ONG.ipynb	Created using Colaboratory	12 days ago
PYME_database.ipynb	Created using Colaboratory	19 days ago

At the bottom of the page, there is a footer with links to GitHub's Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About pages.

<https://github.com/adigenova/uohdb/commit/8c3b5fff59328db531f2ce0ef00e6211113bfa5e>

SQL

DML- Data Manipulation Language

- DML (Lenguaje de manipulación de datos):
 - Utilizadas para **INSERT**, borrar **DELETE**, modificar **UPDATE** y consultar **SELECT FROM** los datos de una base de datos.

```
SELECT nombre_columnas_a_seleccionar  
FROM tabla_a_consultar  
WHERE condiciones;
```

SQL

SELECT

```
SELECT nombre_columnas_a_seleccionar  
FROM tabla_a_consultar  
WHERE condiciones;
```

- select * from Jugador;
- select nombre, edad from Jugador;
- select nombre from Jugador where edad > 30;
- select nombre from Jugador where edad > 30 or id_j < 10;

SQL

Funciones de agregación

- **COUNT**
 - Select count(id_j) from Jugador where edad > 31;
- **SUM**
 - select sum(edad) from Jugador where edad > 31;
- **AVG**
 - select avg(edad) from Jugador where edad > 31;
- **MIN y MAX**
 - select max(edad) from Jugador where edad > 31;

SQL

BETWEEN y IN

- Select * from Jugador where edad **between** 20 and 30;
- Select * from Jugador where id_j **in** (1,20,35);
- Select * from Jugador where id_j **not in** (1,20,35);

Algebra relacional

Algebra relacional

Operadores

- Operaciones fundamentales para recuperar y manipular tuplas en una relación.
 - Basado en álgebra de conjuntos.
- Cada operador toma uno o más relaciones como entradas y su salida es una nueva relación.
 - Podemos "encadenar" operadores para crear operaciones más complejas.
- Operadores:
 - ó [select], \cup [union], \cap [intersection], - [Diference], \times [Product], \bowtie [join]

Algebra relacional

Ejemplos select

- Seleccionar un subconjunto de las tuplas de una relación que satisface unos criterios de selección (predicado)
 - El predicado actúa como un filtro para seleccionar solo tuplas que cumplen su condición.
 - Podemos combinar múltiples predicados (AND, OR..)

R(a_id, b)

a_id	b
a1	10
a2	20
a2	31
a4	32

$$\sigma_{a_id='a1'}(R) \quad \sigma_{a_id='a2' \wedge b>30}(R)$$

a_id	b
a1	10

a_id	b
a2	31

```
SELECT * FROM R  
WHERE a_id = 'a2' AND B > 30
```

Algebra relacional

UNION, INTERSECTION, DIFFERENCE, PRODUCT

```

1 %%sql
2 drop table if exists R;
3 drop table if exists S;
4 -- Relación R --
5 CREATE TABLE R (
6 a_id INTEGER,
7 b INTEGER);
8 -- Relación S --
9 CREATE TABLE S (
10 a_id INTEGER,
11 b INTEGER);
12
13 -- valores en R --
14 INSERT INTO R(a_id,b) VALUES
15 (1,10),
16 (2,20),
17 (2,31),
18 (4,32);
19 -- valores en R --
20 INSERT INTO S(a_id,b) VALUES
21 (4,32),
22 (5,50),
23 (7,51);

```

 $R \cup S$

```

1 %%sql
2 select * from R
3 UNION ALL
4 select * from S;

```

* sqlite:///test.db
Done.

a_id **b**

1	10
2	20
2	31
4	32
4	32
5	50
7	51

 $R \cap S$

```

1 %%sql
2 select * from R
3 INTERSECT
4 select * from S;

```

* sqlite:///test.db
Done.

a_id **b**

4	32
---	----

 $R \times S$

```

1 %%sql
2 select * from R,S;

```

* sqlite:///test.db
Done.

a_id **b** **a_id_1** **b_1**

1	10	4	32
1	10	5	50
1	10	7	51
2	20	4	32
2	20	5	50
2	20	7	51
2	31	4	32
2	31	5	50
2	31	7	51
4	32	4	32
4	32	5	50
4	32	7	51

 $R - S$

```

1 %%sql
2 select * from R
3 EXCEPT
4 select * from S;

```

* sqlite:///test.db
Done.

a_id **b**

1	10
2	20
2	31

Algebra relacional

JOIN

```
1 %%sql
2 drop table if exists R;
3 drop table if exists S;
4 -- Relación R --
5 CREATE TABLE R (
6 a_id INTEGER,
7 b INTEGER);
8 -- Relación S --
9 CREATE TABLE S (
10 a_id INTEGER,
11 b INTEGER);
12
13 -- valores en R --
14 INSERT INTO R(a_id,b) VALUES
15 (1,10),
16 (2,20),
17 (2,31),
18 (4,32);
19 -- valores en R --
20 INSERT INTO S(a_id,b) VALUES
21 (4,32),
22 (5,50),
23 (7,51);
```

Generar una relación que contenga todas las tuplas que son una combinación de dos tuplas (una de cada relación de entrada) con un valor(es) común(es) para uno o más atributos.

$$R \bowtie S$$

```
1 %%sql
2 select * from R NATURAL JOIN S;
```

```
* sqlite:///test.db
Done.
```

a_id	b
4	32

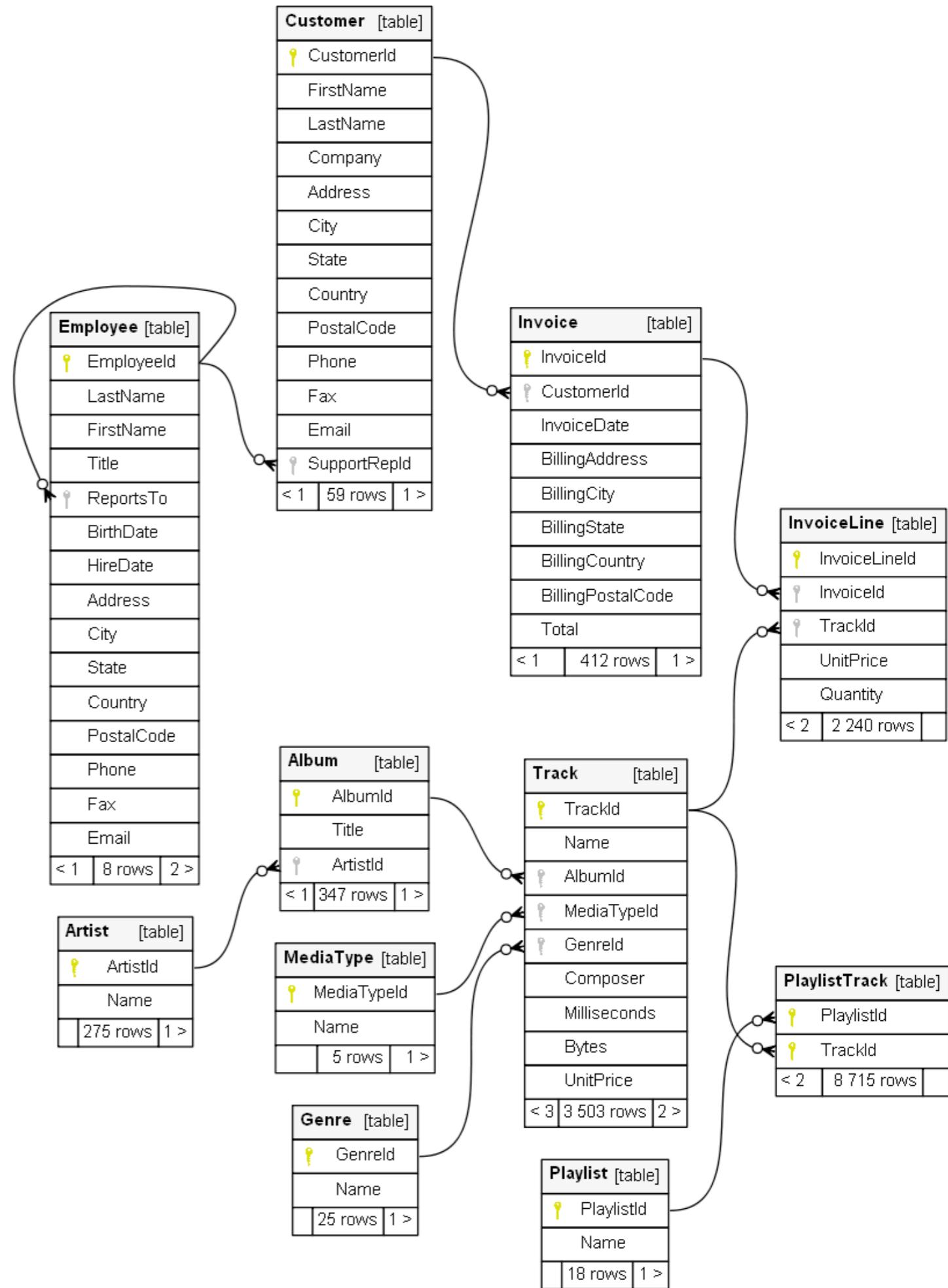
- RENAME
- SORTING
- AGGREGATION

SQL II

SQL II

Consultas combinando 2 o más relaciones

- Base de datos **Chinook**
- La base de datos Chinook modela una tienda de medios digitales, que incluye tablas para artistas, álbumes, canciones, facturas y clientes.



SQL II

as, DISTINCT, GROUP BY, and HAVING

- Contar el número de canciones?

```
select COUNT(TrackID) as cnt from Track;  
cnt 3503
```

- Mostrar las listas distintas desde PlaylistTrack?

```
select DISTINCT PlaylistId from PlaylistTrack;  
  
select COUNT(DISTINCT PlaylistId) as cnt from PlaylistTrack;  
cnt 14
```

- Contar el número de canciones por lista?

```
select PlaylistId,COUNT(TrackId) as cnt  
from PlaylistTrack GROUP BY PlaylistId;
```

PlaylistId	cnt	PlaylistId	cnt
1	3290	1	3290
3	213	3	213
5	1477	5	1477
8	3290	8	3290
9	1	10	213
10	213	11	39
12	75	12	75
13	25	13	25
14	25	14	25
15	25	15	25

- Seleccionar las listas con más de 100 canciones?

```
select PlaylistId,COUNT(TrackId) as cnt  
from PlaylistTrack  
GROUP BY PlaylistId  
HAVING cnt > 100;
```

SQL II

Combinando relaciones

- Seleccionar el nombre del artista y el nombre del album

```
select Name, Title as album_name  
from album as al, Artist as a  
where al.Artistid = a.Artistid limit 10;
```

- Mostrar los album de AC/DC

```
select Name, Title as album_name  
from album as al, Artist as a  
where al.Artistid = a.Artistid and a.Name = "AC/DC";
```

Name	album_name
AC/DC	For Those About To Rock We Salute You
Accept	Balls to the Wall
Accept	Restless and Wild
AC/DC	Let There Be Rock
Aerosmith	Big Ones
Alanis Morissette	Jagged Little Pill
Alice In Chains	Facelift
Antônio Carlos Jobim	Warner 25 Anos
Apocalyptica	Plays Metallica By Four Cellos
Audioslave	Audioslave

Name	album_name
AC/DC	For Those About To Rock We Salute You
AC/DC	Let There Be Rock

Practicar SQL

Google Colab

The screenshot shows a Google Colab notebook titled "SQL_II_chinook_db.ipynb". The notebook contains two code cells. The top cell runs a query to find playlists with more than 100 tracks:

```
[31] 1 %%sql
2 select PlaylistId,COUNT(TrackId) as cnt
3 from PlaylistTrack
4 GROUP BY PlaylistId
5 HAVING cnt > 100;
```

The output shows five playlists with their respective track counts:

PlaylistId	cnt
1	3290
3	213
5	1477
8	3290
10	213

The bottom cell runs a query to list albums by artist:

```
1 %%sql
2 select Name, Title as album_name |
3 from album as al, Artist as a
4 where al.Artistid = a.Artistid limit 10;
```

The output shows ten albums with their artists:

Name	album_name
AC/DC	For Those About To Rock We Salute You
Accept	Balls to the Wall
Accept	Restless and Wild
AC/DC	Let There Be Rock
Aerosmith	Big Ones
Alanis Morissette	Jagged Little Pill
Alice In Chains	Facelift

Both cells completed successfully at 9:56 AM.

Consultas?

Consultas o comentarios?

Muchas gracias