lenguaje SQL II

Alex Di Genova

02/05/2023

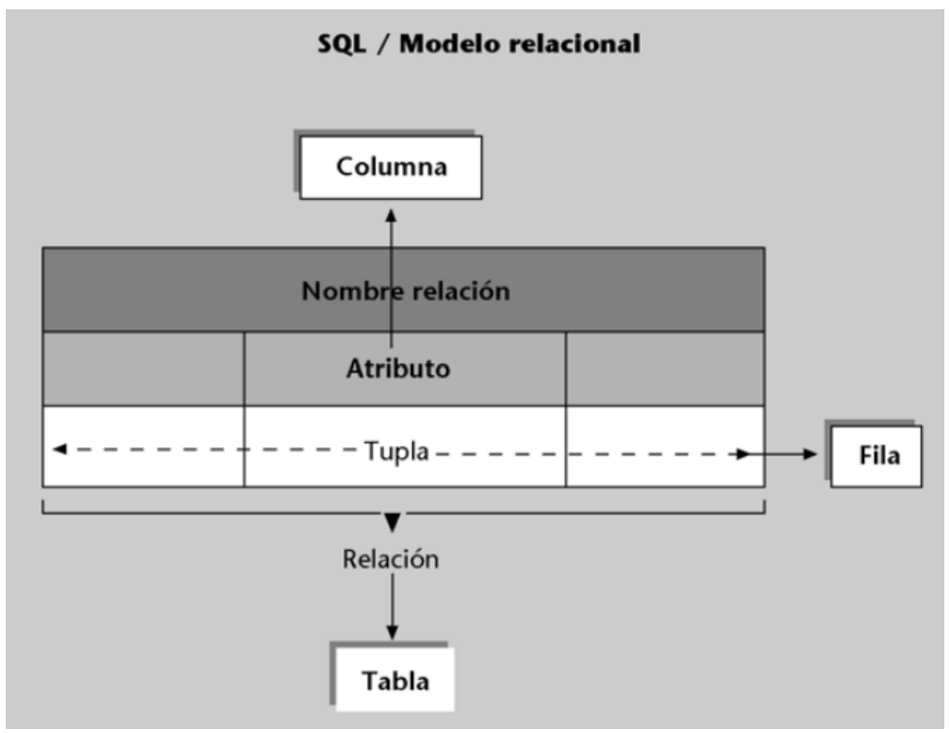
Contenidos

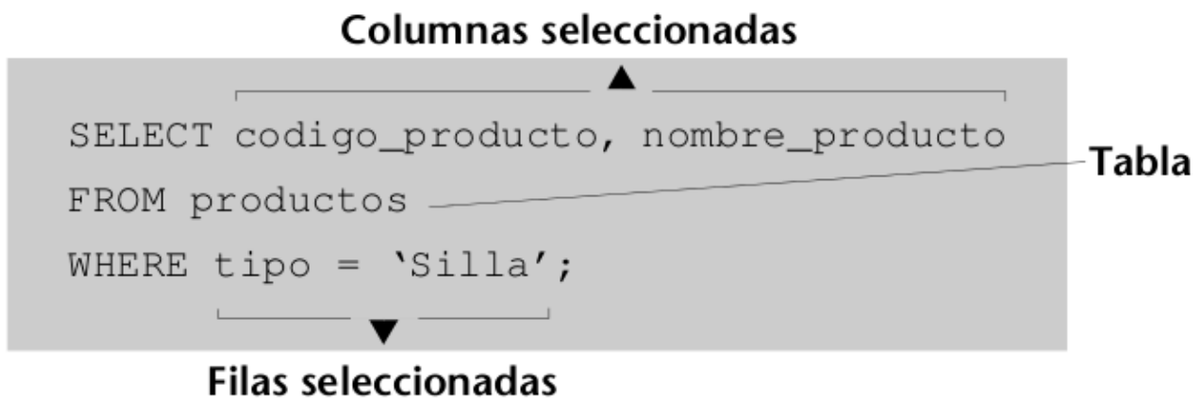
- Repaso
- Algebra relacional
- SQL II

Qué es el SQL?

Introducción

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





SQL

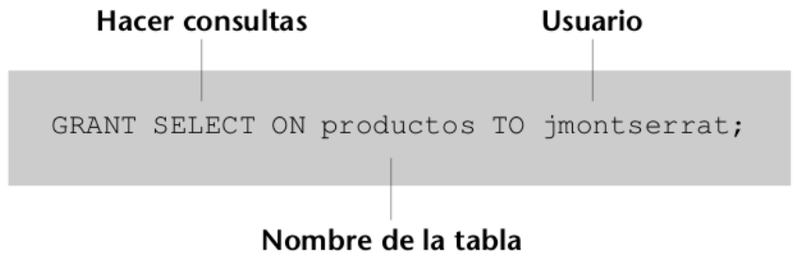
DDL- Data Definition Language

- **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)
```

```
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}]
```

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.

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 algebra de conjuntos.
- Cada operador toma uno o más relaciones como entradas y su salida es una nueva relacion.
    - Podemos "encadenar" operadores para crear operaciones más complejas.
- Operadores:
    - ó [select], ∪ [union], ∩ [intersection], - [Diference], x [Product], ⋈ [join]

Algebra relacional

Ejemplos select

- Seleccionar un subconjunto de las tuplas de una relación que satisface unos criterios de seleccion (predicado)
- El predicado actúa como un filtro para selectionar 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

$\sigma_{a\_id='a1'}(R)$

a_id	b
a1	10

$\sigma_{a\_id='a2' \wedge b>30}(R)$

a_id	b
a2	31

```
SELECT * FROM R
WHERE a_id =‘a2’ and B>30
```

Algebra relacional

UNION, INTERSECTION, DIFFERENCE, PRODUCT

```
%%sql
drop table if exists R;
drop table if exists S;
-- Relación R --
CREATE TABLE R (
a_id INTEGER,
b INTEGER);
-- Relación S --
CREATE TABLE S (
a_id INTEGER,
b INTEGER);

-- valores en R --
INSERT INTO R(a_id,b) VALUES
(1,10),
(2,20),
(2,31),
(4,32);
-- valores en R --
INSERT INTO S(a_id,b) VALUES
(4,32),
(5,50),
(7,51);
```

$$R \cup S$$

```
%%sql
select * from R
UNION ALL
select * from S;
```

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

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

$$R \cap S$$

```
%%sql
select * from R
INTERSECT
select * from S;
```

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

a_id	b
4	32

$$R - S$$

```
%%sql
select * from R
EXCEPT
select * from S;
```

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

a_id	b
1	10
2	20
2	31

$$R \; x \; S$$

```
%%sql
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

Algebra relacional

JOIN

```
%%sql
drop table if exists R;
drop table if exists S;
-- Relación R --
CREATE TABLE R (
a_id INTEGER,
b INTEGER);
-- Relación S --
CREATE TABLE S (
a_id INTEGER,
b INTEGER);

-- valores en R --
INSERT INTO R(a_id,b) VALUES
(1,10),
(2,20),
(2,31),
(4,32);
-- valores en R --
INSERT INTO S(a_id,b) VALUES
(4,32),
(5,50),
(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.

$$R \bowtie S$$

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

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

a_id	b
4	32

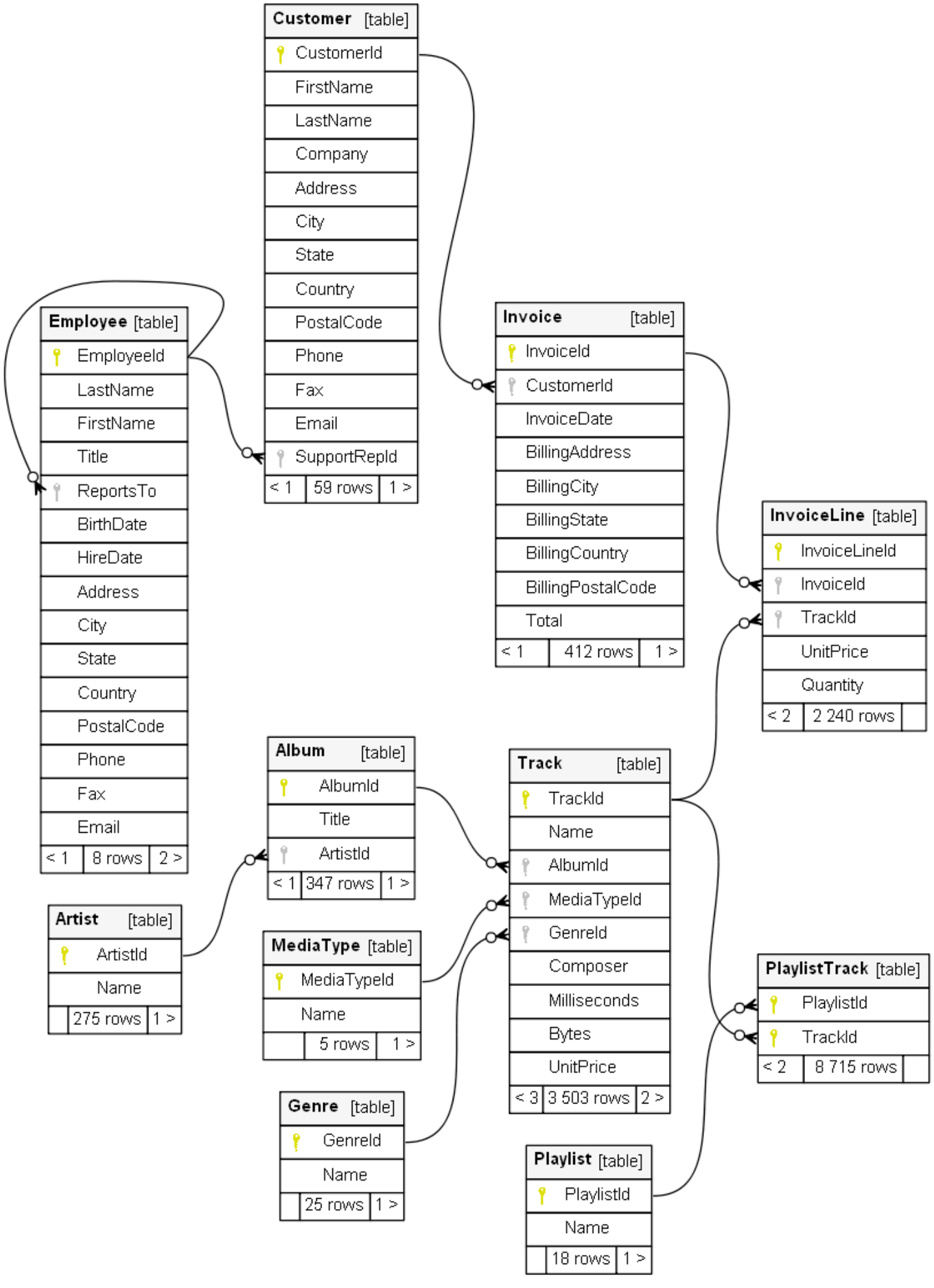
- RENAME
- SORTING
- AGGREGATION

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
1	3290
3	213
5	1477
8	3290
9	1
10	213
11	39
12	75
13	25
14	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;
```

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

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;
```

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

- 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
AC/DC	Let There Be Rock

Operaciones con strings

LIKE

- **LIKE** es usado para realizar busquedas en texto.
  - Operadores:
    - ‘%’ -> acepta cualquier substring.

```
select distinct(composer) from track where composer like
"%Brian";
select distinct(composer) from track where composer like
"%Brian";
select distinct(composer) from track where composer like
“%Brian%";
```

    - ‘_’ -> cualquier caracter

```
select distinct(composer) from track where composer like
"%Br_an%";
```

```
1 %%sql
2 select distinct(composer) from track where co
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
```

Composer
May, Brian
Deacon, John/May, Brian

```
[86] 1 %%sql
     2 select distinct(composer) from track where co
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
```

Composer
Brian Holland/Freddie Gorman/Georgia Dobbins/Robert Batemar
Brian May
Brian Holland/Eddie Holland/Lamont Dozier
Brian Eno, Bono, Adam Clayton, The Edge & Larry Mullen Jnr.
Brian Eno/U2

```
[84] 1 %%sql
     2 select distinct(composer) from track where co
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
```

Composer
Angus Young, Malcolm Young, Brian Johnson
Brian Holland/Freddie Gorman/Georgia Dobbins/Robert Batemar
Brian May
Brian Holland/Eddie Holland/Lamont Dozier
Brian Eno, Bono, Adam Clayton, The Edge & Larry Mullen Jnr.
May, Brian
Deacon, John/May, Brian
Brian Eno/U2

Operaciones con strings

Otras funciones

- Length, upper, lower, substr, replace, ||

```
select length(distinct(composer)) from track where
composer like "%Br_an%" limit 2;
```

```
select upper(distinct(composer)) from track where
composer like "%Br_an%" limit 2;
```

```
select substr(distinct(composer),1,6) from track
where composer like "%Br_an%" limit 2;
```

```
select replace(distinct(composer),"Angus","Alex")
from track where composer like "%Br_an%" limit 2;
```

```
select composer || Name as merge_composer_name from
track where composer like "%Br_an%" limit 2;
```

```
1 %%sql
2 select length(distinct(composer)) from track where composer like
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/
Done.
```

length(distinct(composer))
41
41

```
1 %%sql
2 select upper(distinct(composer)) from track where composer like
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/
Done.
```

upper(distinct(composer))
ANGUS YOUNG, MALCOLM YOUNG, BRIAN JOHNSON
ANGUS YOUNG, MALCOLM YOUNG, BRIAN JOHNSON

```
1 %%sql
2 select substr(distinct(composer),1,6) from track where composer
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/
Done.
```

substr(distinct(composer),1,6)
Angus
Angus

```
1 %%sql
2 select replace(distinct(composer),"Angus","Alex") from track whe
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/
Done.
```

replace(distinct(composer),"Angus","Alex")
Alex Young, Malcolm Young, Brian Johnson
Alex Young, Malcolm Young, Brian Johnson

```
1 %%sql
2 select composer || Name as merge_composer_name from track where
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/
Done.
```

merge_composer_name
Angus Young, Malcolm Young, Brian JohnsonFor Those About To Rock (We Salute You)
Angus Young, Malcolm Young, Brian JohnsonPut The Finger On You

Controlando los resultados

ORDER BY, LIMIT

- ORDER BY <atributo> [ASC|DESC]
  - Ordena las tuplas por uno o más valores de sus columnas

```
select Name,Milliseconds from track
order by Milliseconds ASC limit 3;
```

```
select Name,Milliseconds from track
order by Milliseconds DESC limit 3;
```

```
select * from track
order by MediaTypeId DESC, GenreId ASC, AlbumId DESC
limit 10;
```

- LIMIT X [offset] Y

```
select * from track limit 5 offset 10
```

```
1 %%sql
2 select Name,Milliseconds from track
3 order by Milliseconds ASC limit 3;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources
Done.
```

Name	Milliseconds
É Uma Partida De Futebol	1071
Now Sports	4884
A Statistic	6373

```
1 %%sql
2 select Name,Milliseconds from track
3 order by Milliseconds DESC limit 3;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources
Done.
```

Name	Milliseconds
Occupation / Precipice	5286953
Through a Looking Glass	5088838
Greetings from Earth, Pt. 1	2960293

```
1 %%sql
2 select * from track
3 order by MediaTypeId DESC, GenreId ASC, AlbumId DESC limit 10;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources
Done.
```

TrackId	Name	AlbumId	MediaTypeId	GenreId	
3355	Love Comes	265	5	1	Darius "Take One" Min
3353	I Guess You're Right	265	5	1	Darius "Take One" Min
3357	OAM's Blues	267	5	2	Aaron Goldberg
3350	Despertar	262	5	2	Andrea Dulbecco
3349	Amanda	262	5	2	Luca Gusella
3356	Muita Bobeira	266	5	7	Luciana Souza
3358	One Step Beyond	264	5	15	Karsh Kale
3352	Distance	264	5	15	Karsh Kale/Vishal Vaid
3354	I Ka Barra (Your Work)	263	5	16	Habib Koité
3351	Din Din Wo (Little Child)	263	5	16	Habib Koité

```
1 %%sql
2 select * from track limit 5 offset 10
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources
Done.
```

TrackId	Name	AlbumId	MediaTypeId	GenreId	Con
11	C.O.D.	1	1	1	Angus Young, Malcol
12	Breaking The Rules	1	1	1	Angus Young, Malcol
13	Night Of The Long Knives	1	1	1	Angus Young, Malcol
14	Spellbound	1	1	1	Angus Young, Malcol
15	Go Down	4	1	1	AC/DC

Manipulando fechas y horas

strftime

- Atributos tipo DATETIME
  - TEXTO “YYYY-MM-DD HH:MM:SS.SSS"
- Función strftime permite formatear fechas y realizar calculos con fechas.

Format	Explanation
%d	Day of the month (1-31)
%f	Seconds with fractional seconds (*SS.sss*)
%H	Hour on 24-hour clock (00-23)
%j	Day of the year (001-366)
%J	Julian day number (*DDDDDDD.ddddddd*)
%m	Month (01-12)
%M	Minute (00-59)
%s	Seconds since 1970-01-01
%S	Seconds (00-59)
%w	Weekday (0-6) (0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday)
%W	Week number in the year (00-53) The first Monday is the beginning of week 1.
%Y	Year with century (*yyyy*)
%%	% as a literal

Manipulando fechas y horas

Strftime

- Atributos tipo DATETIME
  - Función strftime permite retornar fechas con formato

```
PRAGMA table_info([invoice]);
```

```
select  InvoiceDate,
strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as "Dia",
strftime('%H',InvoiceDate) as "Horas",
strftime('%M',InvoiceDate) as "Minutos"
from invoice limit 5;
```

```
select  InvoiceDate,
strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as Dia
from invoice
where Dia in ("01","02","11") limit 5;
```

```
%%sql
PRAGMA table_info([invoice]);
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSo
Done.
```

cid	name	type	notnull	dflt_value	pk
0	InvoiceId	INTEGER	1	None	1
1	CustomerId	INTEGER	1	None	0
2	InvoiceDate	DATETIME	1	None	0
3	BillingAddress	NVARCHAR(70)	0	None	0
4	BillingCity	NVARCHAR(40)	0	None	0
5	BillingState	NVARCHAR(40)	0	None	0
6	BillingCountry	NVARCHAR(40)	0	None	0
7	BillingPostalCode	NVARCHAR(10)	0	None	0
8	Total	NUMERIC(10,2)	1	None	0

```
%%sql
select  InvoiceDate,strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as "Dia",
strftime('%H',InvoiceDate) as "Horas",
strftime('%M',InvoiceDate) as "Minutos"
from invoice limit 5;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSo
Done.
```

InvoiceDate	Año	Mes	Dia	Horas	Minutos
2009-01-01 00:00:00	2009	01	01	00	00
2009-01-02 00:00:00	2009	01	02	00	00
2009-01-03 00:00:00	2009	01	03	00	00
2009-01-06 00:00:00	2009	01	06	00	00
2009-01-11 00:00:00	2009	01	11	00	00

```
%%sql
select  InvoiceDate,strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as Dia
from invoice where Dia in ("01","02","11") limit 5;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSo
Done.
```

InvoiceDate	Año	Mes	Dia
2009-01-01 00:00:00	2009	01	01
2009-01-02 00:00:00	2009	01	02
2009-01-11 00:00:00	2009	01	11
2009-02-01 00:00:00	2009	02	01
2009-02-01 00:00:00	2009	02	01

Almacenando resultados en tablas

CREATE TABLE from SELECT

- Almacenar resultados de consultas en tablas
  - Tablas no deben estar definidas.
  - Tablas deben tener el mismo numero y tipo de columnas

```
create table Fecha as
select strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as "Dia"
from invoice limit 10;
```

```
CREATE TABLE new_table AS
SELECT expressions
FROM existing_tables
[WHERE conditions];
```

```
%%sql
select strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as "Dia"
from invoice limit 5;
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
```

Año	Mes	Dia
2009	01	01
2009	01	02
2009	01	03
2009	01	06
2009	01	11

```
%%sql
create table Fecha as
select strftime('%Y',InvoiceDate) as "Año",
strftime('%m',InvoiceDate) as "Mes",
strftime('%d',InvoiceDate) as "Dia"
from invoice limit 10;
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
[]
```

```
%%sql
select * from Fecha limit 10;
```

```
 * sqlite:////content/chinook-database/ChinookDa
Done.
```

Año	Mes	Dia
2009	01	01
2009	01	02
2009	01	03
2009	01	06
2009	01	11
2009	01	19
2009	02	01

Subconsultas en SQL

SELECT (SELECT)

- Consultas conteniendo sub-consultas
  - Subconsultas pueden ser usadas casi en cualquier lugar de las consultas.
- Select .. from where x IN (select ...)

```
select name , AlbumId
from track
where AlbumId in (
  select AlbumId
  from album
  where title = "Californication")
limit 10;
```

```
#Obtener los nombres de todas las canciones del álbum "Californication".
%%sql
select name , AlbumId
from track
where AlbumId in (
  select AlbumId
  from album
  where title = "Californication")
limit 5;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/Chinook_S
Done.
```

Name	AlbumId
Around The World	195
Parallel Universe	195
Scar Tissue	195
Otherside	195
Get On Top	195

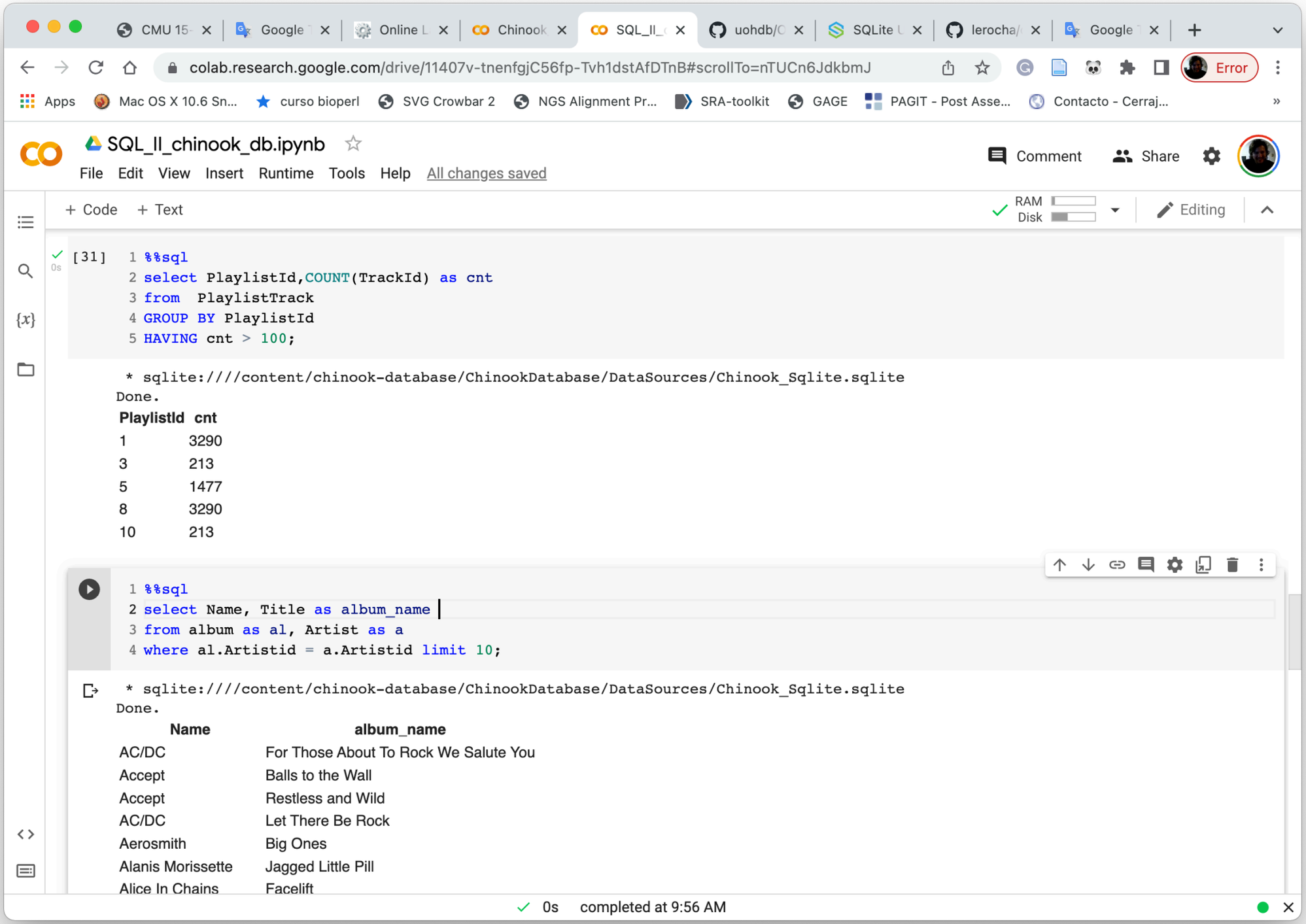
```
%%sql
select name , AlbumId
from track
where AlbumId not in (
  select AlbumId
  from album
  where title = "Californication")
limit 5;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/Chinook_S
Done.
```

Name	AlbumId
For Those About To Rock (We Salute You)	1
Balls to the Wall	2
Fast As a Shark	3
Restless and Wild	3
Princess of the Dawn	3

Practicar SQL

Google Colab



Consultas?

Consultas o comentarios?

Muchas gracias