Programación para Analítica de Datos SQL

POR: Guillermo Andres De Mendoza Corrales

Temario

Teoría:

- Bases de datos
- SQL
- noSQL
- ACID

SQL:

- Introducción
- Select
- Join
- Insert
- Update
- Delete
- Create Table

Python:

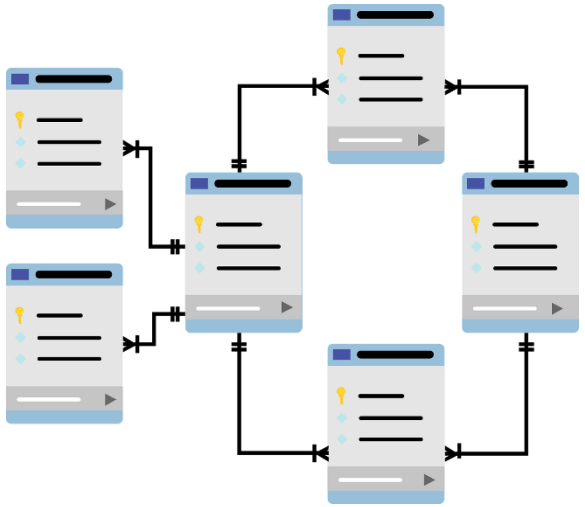
- Conectividad a sql (SQLITE)
- Execute
- DTO

Teoría

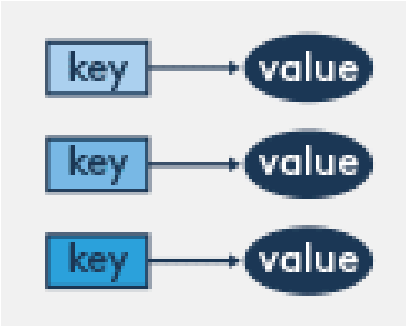
Base de datos

Una base de datos es una herramienta que permite almacenar y organizar información de manera estructurada y electrónica. Los datos se guardan en tablas, que a su vez están compuestas por filas y columnas

Relacionales



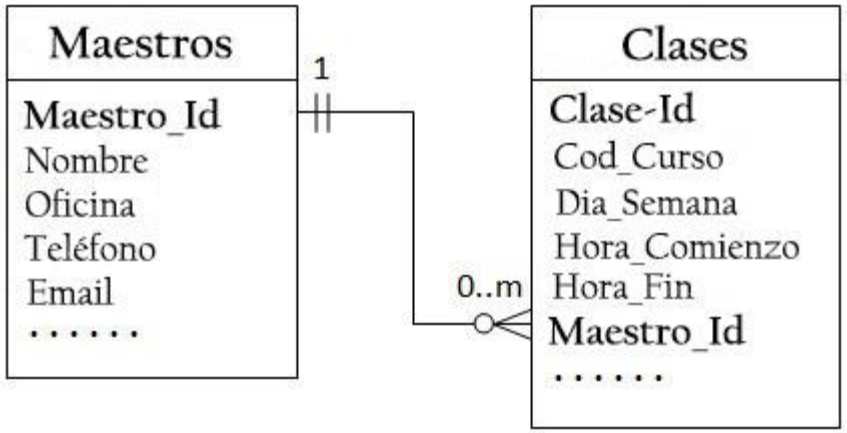
No relacionales



Base de datos - relacionales

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Basándose en el modelo relacional, una forma intuitiva y directa de representar datos en tablas.

En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.



Base de datos – relacionales

*entre otras

Base de datos – NO relacionales

Una base de datos no relacional es un sistema de almacenamiento de datos que **no** utiliza el esquema de tablas y campos de las bases de datos tradicionales. También se les conoce como bases de datos NoSQL, que significa "no solo SQL".

Las bases de datos no relacionales son más **flexibles y permiten gestionar grandes volúmenes de datos**.

id	search-document
233358	{"name": "Pacific Crest National Scenic Trail", "county": "San Diego", "elevation":1294, "location": {"type": "Point", "coordinates": [–120.802102,49.00021]}}
801970	{"name": "Lewis and Clark National Historic Trail", "county": "Richland", "elevation":584, "location": {"type": "Point", "coordinates": [–104.8546903,48.1264084]}}
1144102	{"name": "Intake Trail", "county": "Umatilla", "elevation":1076, "location": {"type": "Point", "coordinates": [–118.0468873,45.9981939]}}

Base de datos – NO relacionales

mongoDB

elastic

redis

cassandra

Amazon DynamoDB

*entre otras

¿Que requiere algo para ser una base de datos?

Requiere cumplir con el modelo ACID

SQL

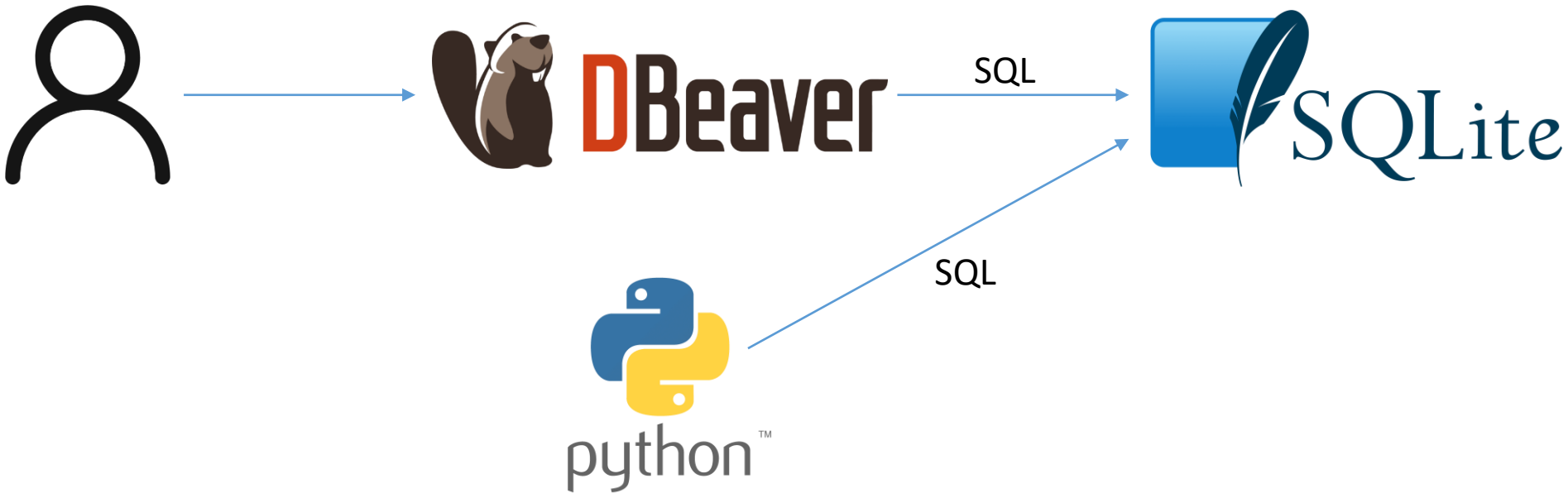
SQL

SQL (Lenguaje de Consulta Estructurada) es un lenguaje de programación que se utiliza para administrar bases de datos relacionales. Con SQL se pueden almacenar, actualizar, eliminar, buscar y recuperar información de las bases de datos.

SQL es una herramienta clave para trabajar con datos, ya que permite realizar operaciones como: Insertar datos, Consultar datos, Actualizar datos, Eliminar datos.

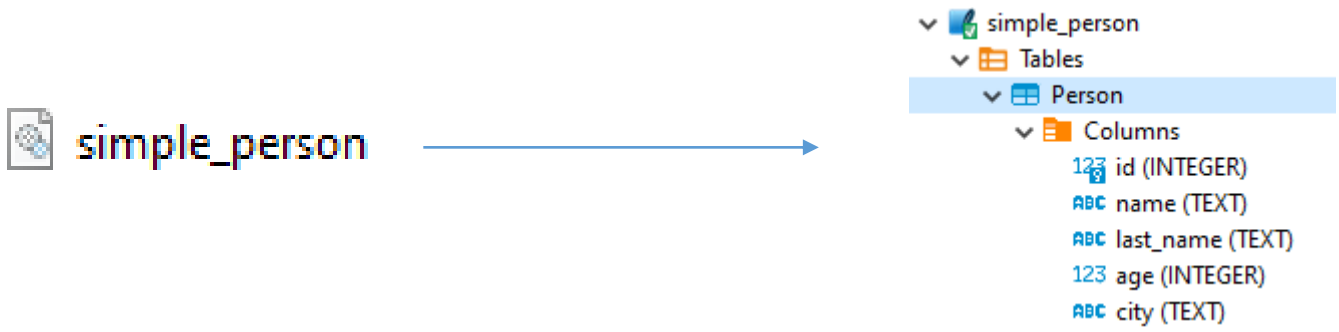
NO ES CASE SENSITIVE !

Dbeaver & python



Database : simple_person.db

La siguiente información va a referenciar a la siguiente base de datos



Tablas

Nombre

PK

Columnas

Person | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

Text

	id	name	last_name	age	city
1	1	Mario	Bross	50	Bogota
2	2	Luigi	Bross	40	Medellin
3	3	Donkey	Kong	20	Cartagena
4	4	Bowser	Evil	70	Bogota
5	5	Peach	Princess	25	Barranquilla
6	6	Warrio	Bross	45	Santa Marta

Filas

Tablas

Descripción

Table Name: Person

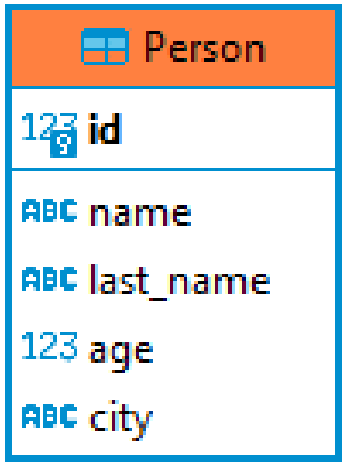
Table Type: TABLE

Table Description:

Columns
Keys
Foreign Keys
Indexes
References
Triggers
DDL
Virtual

Column Name	#	Data Type	Length	Not Null	Auto Increment	Default	Description
id	1	INTEGER		[v]	[v]		
name	2	TEXT		[v]	[ ]		
last_name	3	TEXT		[v]	[ ]		
age	4	INTEGER		[ ]	[ ]		
city	5	TEXT		[ ]	[ ]		

Diagrama



Llave principal (PK)

Valor único que referencia a cada fila de una tabla

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta

SELECT (all)

Tabla

Alias

```
SELECT *
FROM Person p;
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta

SELECT (some)

```sql
SELECT p.id, p.name
FROM Person p;
```

id	name
1	Mario
2	Luigi
3	Donkey
4	Bowser
5	Peach
6	Warrio

SELECT (alias)

```sql
SELECT p.id as 'identificador', p.name as 'nombre'
FROM Person p;
```

identificador	nombre
1	Mario
2	Luigi
3	Donkey
4	Bowser
5	Peach
6	Warrio

SELECT (where)

```sql
SELECT *
FROM Person p
WHERE p.city = "Bogota";
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
4	Bowser	Evil	70	Bogota

SELECT (where + and + or)

```
SELECT *
FROM Person p
WHERE p.city = "Bogota" AND p.age > 60;
```

id	name	last_name	age	city
4	Bowser	Evil	70	Bogota

```
SELECT *
FROM Person p
WHERE p.city = "Bogota" OR p.age > 40;
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
4	Bowser	Evil	70	Bogota
6	Warrio	Bross	45	Santa Marta

SELECT (in - notin)

```sql
SELECT *
FROM Person p
WHERE p.city IN("Medellin","Santa Marta");
```

id	name	last_name	age	city
2	Luigi	Bross	40	Medellin
6	Warrio	Bross	45	Santa Marta

```sql
SELECT *
FROM Person p
WHERE p.city NOT IN ("Medellin","Santa Marta");
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla

SELECT (like)

Inicia con

```
SELECT *
FROM Person p
WHERE p.name LIKE "M%";
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota

termina con

```
SELECT *
FROM Person p
WHERE p.name LIKE "%o";
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
6	Warrio	Bross	45	Santa Marta

contiene

```
SELECT *
FROM Person p
WHERE p.name LIKE "%ar%";
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
6	Warrio	Bross	45	Santa Marta

SELECT (order by)

ascendente

```sql
SELECT *
FROM Person p
ORDER BY p.age ASC;
```

id	name	last_name	age	city
3	Donkey	Kong	20	Cartagena
5	Peach	Princess	25	Barranquilla
2	Luigi	Bross	40	Medellin
6	Warrio	Bross	45	Santa Marta
1	Mario	Bross	50	Bogota
4	Bowser	Evil	70	Bogota

desendente

```sql
SELECT *
FROM Person p
ORDER BY p.age DESC;
```

id	name	last_name	age	city
4	Bowser	Evil	70	Bogota
1	Mario	Bross	50	Bogota
6	Warrio	Bross	45	Santa Marta
2	Luigi	Bross	40	Medellin
5	Peach	Princess	25	Barranquilla
3	Donkey	Kong	20	Cartagena

SELECT (limit)

```sql
SELECT *
FROM Person p
ORDER BY p.age DESC
LIMIT 3;
```

id	name	last_name	age	city
4	Bowser	Evil	70	Bogota
1	Mario	Bross	50	Bogota
6	Warrio	Bross	45	Santa Marta

SELECT (distinct)

```
SELECT DISTINCT p.city
FROM Person p;
```

ABC city
Bogota
Medellin
Cartagena
Barranquilla
Santa Marta

SELECT (count, sum, average)

```
SELECT COUNT(*)
FROM Person p
WHERE p.city = "Bogota";
```

123 COUNT(*)
2

```
SELECT SUM(p.age)
FROM Person p
WHERE p.city = "Bogota";
```

123 SUM(p.age)
120

```
SELECT AVG(p.age)
FROM Person p
WHERE p.city = "Bogota";
```

123 AVG(p.age)
60

SELECT (min, max)

```
SELECT MIN(p.age)
FROM Person p
WHERE p.city = "Bogota";
```

123 MIN(p.age)
50

```
SELECT MAX(p.age)
FROM Person p
WHERE p.city = "Bogota";
```

123 MAX(p.age)
70

SELECT (group by)

```
SELECT p.city , sum(p.age)
FROM Person p
GROUP BY p.city;
```

city	sum(p.age)
Barranquilla	25
Bogota	120
Cartagena	20
Medellin	40
Santa Marta	45

```
SELECT p.city , avg(p.age)
FROM Person p
GROUP BY p.city;
```

city	avg(p.age)
Barranquilla	25
Bogota	60
Cartagena	20
Medellin	40
Santa Marta	45

```
SELECT p.city , count(p.age)
FROM Person p
GROUP BY p.city;
```

city	count(p.age)
Barranquilla	1
Bogota	2
Cartagena	1
Medellin	1
Santa Marta	1

SELECT (having)

Genera condiciones después de agrupar (GROUP BY)

```
SELECT p.city , count(p.age)
FROM Person p
GROUP BY p.city
HAVING count(p.age) >= 2;
```

city	count(p.age)
Bogota	2

HAVING: aplica después de agrupamiento

WHERE: aplica antes de agrupamiento

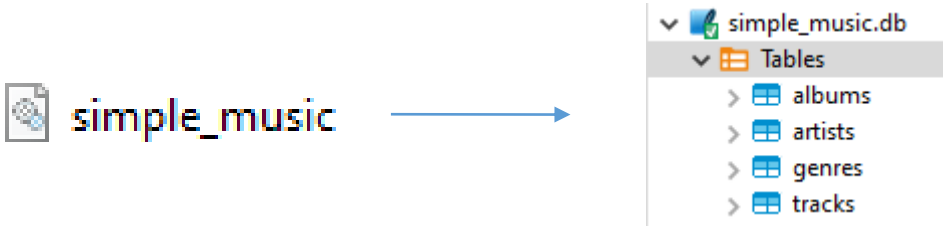
SELECT (case)

```sql
SELECT
    p.name ,
    p.age,
    CASE
        WHEN p.age > 60 THEN "HI"
        WHEN p.age <= 60 AND p.age > 30 THEN "MID"
        ELSE "LOW"
    END as "age_category"
FROM Person p;
```

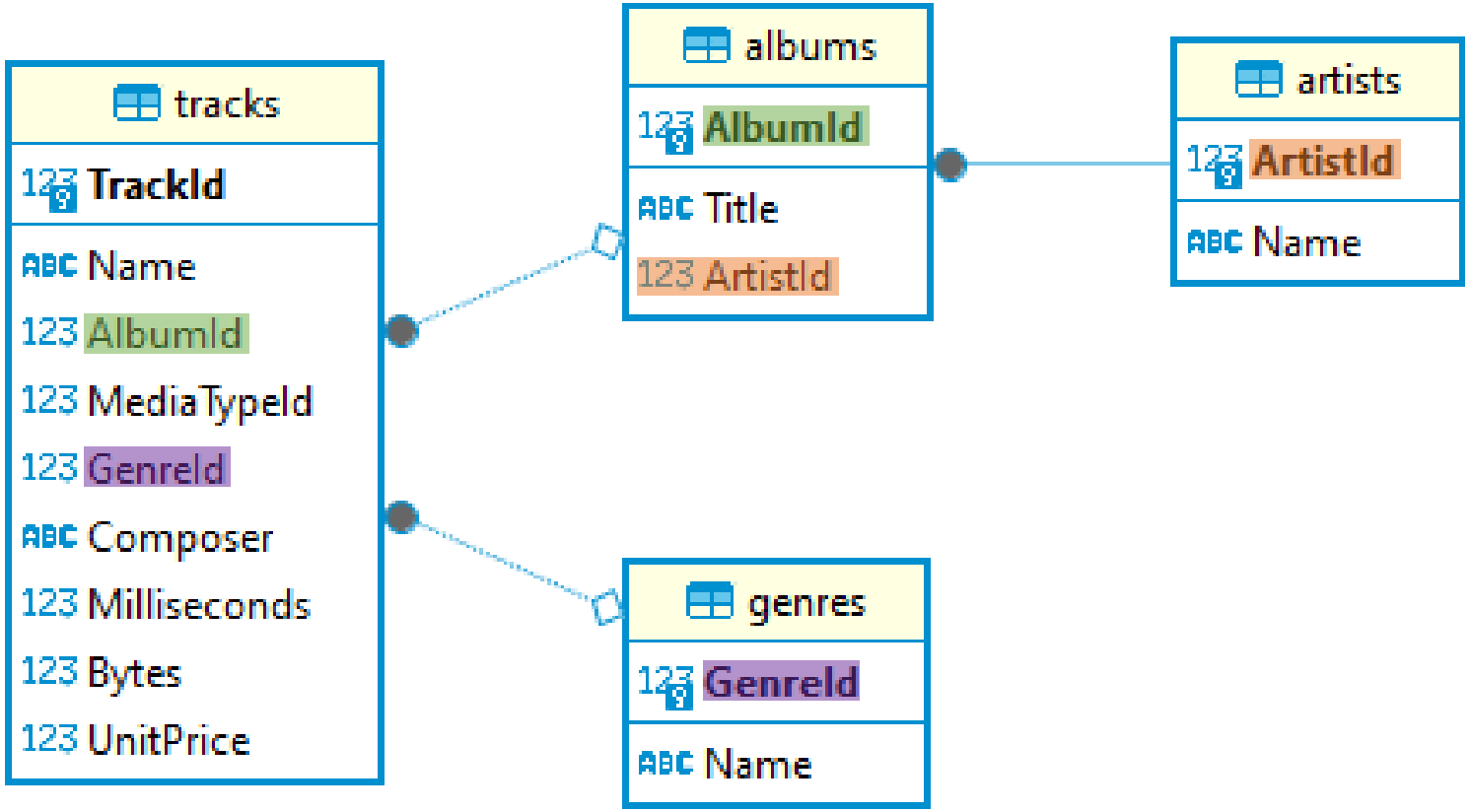
name	age	age_category
Mario	50	MID
Luigi	40	MID
Donkey	20	LOW
Bowser	70	HI
Peach	25	LOW
Warrio	45	MID

Database : simple_music.db

La siguiente información va a referenciar a la siguiente base de datos



Llaves foraneas



Tablas relacionadas y llaves foraneas

Artist

ArtistId	Name
1	AC/DC
2	Accept
3	Aerosmith
4	Alanis Morissette
5	Alice In Chains
6	Antônio Carlos Jobim

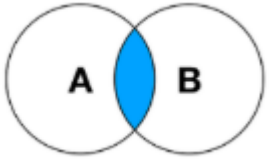
Albums

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
4	Let There Be Rock	1
2	Balls to the Wall	2
3	Restless and Wild	2
5	Big Ones	3
6	Jagged Little Pill	4
7	Facelift	5
8	Warner 25 Anos	6
34	Chill: Brazil (Disc 2)	6
9	Plays Metallica By Four Cellos	7
10	Audioslave	8
11	Out Of Exile	8
271	Revelations	8
12	BackBeat Soundtrack	9
13	The Best Of Billy Cobham	10

Tracks

TrackId	Name	AlbumId
1	For Those About To Rock (We Salute You)	1
2	Balls to the Wall	2
3	Fast As a Shark	3
4	Restless and Wild	3
5	Princess of the Dawn	3
6	Put The Finger On You	1
7	Let's Get It Up	1
8	Inject The Venom	1
9	Snowballed	1
10	Evil Walks	1
11	C.O.D.	1
12	Breaking The Rules	1
13	Night Of The Long Knives	1
14	Spellbound	1
15	Go Down	4
16	Dog Eat Dog	4
17	Let There Be Rock	4
18	Bad Boy Boogie	4

INNER JOIN




```
SELECT *
FROM artists art
INNER JOIN albums alm ON art.ArtistId = alm.ArtistId;
```

ArtistId	Name	AlbumId	Title	ArtistId
1	AC/DC	1	For Those About To Rock We Salute You	1
1	AC/DC	4	Let There Be Rock	1
2	Accept	2	Balls to the Wall	2
2	Accept	3	Restless and Wild	2
3	Aerosmith	5	Big Ones	3
4	Alanis Morissette	6	Jagged Little Pill	4
5	Alice In Chains	7	Facelift	5
6	Antônio Carlos Jc	8	Warner 25 Anos	6
6	Antônio Carlos Jc	34	Chill: Brazil (Disc 2)	6
7	Apocalyptica	9	Plays Metallica By Four Cellos	7
8	Audioslave	10	Audioslave	8
8	Audioslave	11	Out Of Exile	8
8	Audioslave	271	Revelations	8
9	BackBeat	12	BackBeat Soundtrack	9
10	Billy Cobham	13	The Best Of Billy Cobham	10
11	Black Label Socie	14	Alcohol Fueled Brewtality Live! [Disc 1]	11
11	Black Label Socie	15	Alcohol Fueled Brewtality Live! [Disc 2]	11
12	Black Sabbath	16	Black Sabbath	12
12	Black Sabbath	17	Black Sabbath Vol. 4 (Remaster)	12
13	Body Count	18	Body Count	13
14	Bruce Dickinson	19	Chemical Wedding	14

Artists **Albums**

INNER JOIN example

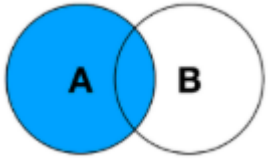
```
SELECT *
FROM artists art
INNER JOIN albums alm ON art.ArtistId = alm.ArtistId
WHERE art.Name = "Audioslave"
```

ArtistId	Name	AlbumId	Title	ArtistId
8	Audioslave	10	Audioslave	8
8	Audioslave	11	Out Of Exile	8
8	Audioslave	271	Revelations	8

```
SELECT art.Name , alm.Title
FROM artists art
INNER JOIN albums alm ON art.ArtistId = alm.ArtistId
WHERE art.Name = "Audioslave";
```

Name	Title
Audioslave	Audioslave
Audioslave	Out Of Exile
Audioslave	Revelations

LEFT JOIN




```sql
SELECT t.TrackId , t.Name  , g.Name
FROM tracks t
LEFT JOIN genres g ON t.GenreId = g.GenreId
ORDER BY t.TrackId DESC;
```

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice	GenreId	Name
3,504	Misterius Track	[NULL]	1	[NULL]	[NULL]	0	0	0	[NULL]	[NULL]
3,503	Koyaanisqatsi	347	2	10	Philip Glass	206,005	3,305,164	0.99	10	Soundtrack
3,502	Quintet for Horn, Violin, 2 Violas, and Cello in E Flat Major, K. 407/386c: III. Allegro	346	2	24	Wolfgang Amadeus Mozart	221,331	3,665,114	0.99	24	Classical
3,501	L'orfeo, Act 3, Sinfonia (Orchestra)	345	2	24	Claudio Monteverdi	66,639	1,189,062	0.99	24	Classical
3,500	String Quartet No. 12 in C Minor, D. 703 "Quartettsatz": II. Andante - Allegro assai	344	2	24	Franz Schubert	139,200	2,283,131	0.99	24	Classical
3,499	Pini Di Roma (Pinien Von Rom) \ I Pini Della Via Appia	343	2	24	[NULL]	286,741	4,718,950	0.99	24	Classical
3,498	Concerto for Violin, Strings and Continuo in G Major, Op. 3, No. 9: I. Allegro	342	4	24	Pietro Antonio Locatelli	493,573	16,454,937	0.99	24	Classical
3,497	Erlkonig, D.328	341	2	24	[NULL]	261,849	4,307,907	0.99	24	Classical
3,496	Étude 1, In C Major - Preludio (Presto) - Liszt	340	4	24	[NULL]	51,780	2,229,617	0.99	24	Classical
3,495	24 Caprices, Op. 1, No. 24, for Solo Violin, in A Minor	339	2	24	Niccolò Paganini	265,541	4,371,533	0.99	24	Classical
3,494	Symphony No. 2, Op. 16 - "The Four Temperaments": II. Allegro Comodo e Flemmati	338	2	24	Carl Nielsen	286,998	4,834,785	0.99	24	Classical
3,493	Metopes, Op. 29: Calypso	337	2	24	Karol Szymanowski	333,669	5,548,755	0.99	24	Classical

Tracks **Genres**

INSERT

Agregar una nueva fila en la tabla person de la bd *simple_person

```sql
INSERT INTO Person (name,last_name,age,city)
VALUES ('Yosgi','Dino',5,'Mushroom kingdom');
```

Statistics 1

Name	Value
Updated Rows	1
Query	INSERT INTO Person (name,last_name,age,city)
	VALUES ('Yosgi','Dino','5','Mushroom kingdom')
Start time	Sat Jan 18 23:18:38 COT 2025
Finish time	Sat Jan 18 23:18:38 COT 2025

Person | Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	last_name	age	city
1	1	Mario	Bross	50	Bogota
2	2	Luigi	Bross	40	Medellin
3	3	Donkey	Kong	20	Cartagena
4	4	Bowser	Evil	70	Bogota
5	5	Peach	Princess	25	Barranquilla
6	6	Warrio	Bross	45	Santa Marta
7	7	Yosgi	Dino	5	Mushroom kingdom

UPDATE

Actualiza un campo en la tabla person en la bd *simple_person

```sql
UPDATE Person
    SET age=500
    WHERE id=7;
```

Statistics 1

Name	Value
Updated Rows	1
Query	UPDATE Person
	SET age=500
	WHERE id=7
Start time	Sat Jan 18 23:22:54 COT 2025
Finish time	Sat Jan 18 23:22:54 COT 2025

Person | Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	last_name	age	city
1	1	Mario	Bross	50	Bogota
2	2	Luigi	Bross	40	Medellin
3	3	Donkey	Kong	20	Cartagena
4	4	Bowser	Evil	70	Bogota
5	5	Peach	Princess	25	Barranquilla
6	6	Warrio	Bross	45	Santa Marta
7	7	Yosgi	Dino	500	Mushroom kingdom

DELETE

Elimina una fila de una tabla person de la bd *simple_person

```sql
DELETE FROM Person
    WHERE id=7;
```

Statistics 1

Name	Value
Updated Rows	1
Query	DELETE FROM Person WHERE id=7
Start time	Sat Jan 18 23:26:04 COT 2025
Finish time	Sat Jan 18 23:26:04 COT 2025

Person | Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	last_name	age	city
1	1	Mario	Bross	50	Bogota
2	2	Luigi	Bross	40	Medellin
3	3	Donkey	Kong	20	Cartagena
4	4	Bowser	Evil	70	Bogota
5	5	Peach	Princess	25	Barranquilla
6	6	Warrio	Bross	45	Santa Marta

CREATE TABLE

Como crear la tabla person simple, en una tabla en alguna base de datos

```sql
CREATE TABLE Person (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    age INTEGER,
    city TEXT
);
```

- Nombre columna
- Tipo de dato
- Especificaciones

CREATE TABLE

Como crear la tabla track simple, en una tabla en alguna base de datos

```sql
CREATE TABLE "albums"
(
    [AlbumId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    [Title] NVARCHAR(160)  NOT NULL,
    [ArtistId] INTEGER  NOT NULL,
    FOREIGN KEY ([ArtistId]) REFERENCES "artists" ([ArtistId])
        ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE INDEX [IFK_AlbumArtistId] ON "albums" ([ArtistId]);
```

Nombre columna

Tipo de dato

Especificaciones

Llaves foraneas

Python

Python
execute sql – get all persons

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)

    def get_all_persons(self):
        try:
            query = f"""
                    SELECT *
                    FROM {self.table} p
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            results = cursor.fetchall()
            return results
        except Exception as e:
            print(f"Error selecting data: {e}")
```

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

```python
person_results = person_repository.get_all_persons()
person_results
```

```
[(1, 'Mario', 'Bross', 50, 'Bogota'),
 (2, 'Luigi', 'Bross', 40, 'Medellin'),
 (3, 'Donkey', 'Kong', 20, 'Cartagena'),
 (4, 'Bowser', 'Evil', 70, 'Bogota'),
 (5, 'Peach', 'Princess', 25, 'Barranquilla'),
 (6, 'Warrio', 'Bross', 45, 'Santa Marta')]
```

Python
execute sql – get one person

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)

    def get_person_by_id(self,person_id):
        try:
            query = f"""
                    SELECT *
                    FROM {self.table} p
                    WHERE p.id = {person_id}
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            results = cursor.fetchone()
            return results
        except Exception as e:
            print(f"Error selecting data: {e}")
```

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

```python
person = person_repository.get_person_by_id(3)
person
```

```
(3, 'Donkey', 'Kong', 20, 'Cartagena')
```

Python
execute sql – create person

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)

    def create_person(self,name,last_name,age,city):
        try:
            query = f"""
                    INSERT INTO {self.table}
                        (name,last_name,age,city)
                    VALUES('{name}','{last_name}',{age},'{city}')
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            self.__connection.commit()
        except Exception as e:
            print(f"Error creating data: {e}")
```

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

```python
person_repository.create_person("Guillermo","DeMendoza",36,"Monteria")
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta
9	Guillermo	DeMendoza	36	Monteria

Python
execute sql – update person

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)

    def update_person_name_by_id(self,person_id,name):
        try:
            query = f"""
                    UPDATE {self.table}
                    SET name = '{name}'
                    WHERE id = {person_id}
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            self.__connection.commit()
        except Exception as e:
            print(f"Error updating data: {e}")
```

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

```python
person_repository.update_person_name_by_id(1,"Boo")
```

123 id	ABC name	ABC last_name	123 age	ABC city
1	Boo	Bross	50	Bogota
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta
9	Guillermo	DeMendoza	36	Monteria

Python
execute sql – delete person

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)

    def delete_person_by_id(self,person_id):
        try:
            query = f"""
                    DELETE FROM {self.table}
                    WHERE id = {person_id}
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            self.__connection.commit()
        except Exception as e:
            print(f"Error updating data: {e}")
```

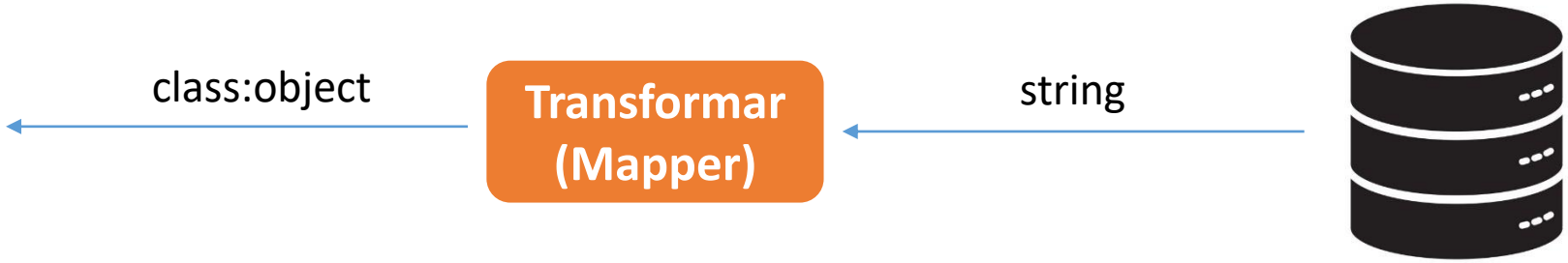
```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

```python
person_repository.delete_person_by_id(1)
```

id	name	last_name	age	city
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta
9	Guillermo	DeMendoza	36	Monteria

DTO

Python DTO



Python DTO

```python
class Person:

    person_id = None
    name = None
    last_name = None
    age = None
    city = None

    def __str__(self):
        return f"PersonDTO(id={self.person_id}, name={self.name}, last_name={self.last_name}, age={self.age}, city={self.city})"

    def __repr__(self):
        return self.__str__()
```

```python
class PersonMapper:

    def toDTO(self,sqlResult):
        person = Person()
        person.person_id = sqlResult[0]
        person.name = sqlResult[1]
        person.last_name = sqlResult[2]
        person.age = sqlResult[3]
        person.city = sqlResult[4]
        return person
```

Python
execute sql – get all persons

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)
        self.mapper = PersonMapper()

    def get_person_by_id(self,person_id):
        try:
            query = f"""
                    SELECT *
                    FROM {self.table} p
                    WHERE p.id = {person_id}
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            result = cursor.fetchone()
            person = self.mapper.toDTO(result)
            return person
        except Exception as e:
            print(f"Error selecting data: {e}")
```

connection

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

Get all

```python
person_repository.get_all_persons()
```

```
[PersonDTO(id=1, name=Mario, last_name=Bross, age=50, city=Bogota),
 PersonDTO(id=2, name=Luigi, last_name=Bross, age=40, city=Medellin),
 PersonDTO(id=3, name=Donkey, last_name=Kong, age=20, city=Cartagena),
 PersonDTO(id=4, name=Bowser, last_name=Evil, age=70, city=Bogota),
 PersonDTO(id=5, name=Peach, last_name=Princess, age=25, city=Barranquilla),
 PersonDTO(id=6, name=Warrio, last_name=Bross, age=45, city=Santa Marta)]
```

Python
execute sql – get all persons

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)
        self.mapper = PersonMapper()

    def get_person_by_id(self,person_id):
        try:
            query = f"""
                    SELECT *
                    FROM {self.table} p
                    WHERE p.id = {person_id}
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            result = cursor.fetchone()
            person = self.mapper.toDTO(result)
            return person
        except Exception as e:
            print(f"Error selecting data: {e}")
```

connection

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

Get one

```python
person_repository.get_person_by_id(1)
```

```
(1, 'Mario', 'Bross', 50, 'Bogota')
```

Python
execute sql – create person

```python
import sqlite3

class PersonRepository:

    table = "person"

    def __init__(self,url):
        self.__connection = sqlite3.connect(url)
        self.mapper = PersonMapper()

    def create_person(self,person):
        try:
            query = f"""
                    INSERT INTO {self.table}
                        (name,last_name,age,city)
                    VALUES('{person.name}','{person.last_name}',{person.age},'{person.city}')
                    """
            cursor = self.__connection.cursor()
            cursor.execute(query)
            self.__connection.commit()
        except Exception as e:
            print(f"Error creating data: {e}")
```

connection

```python
database_path = "C:\\Users\\Memo\\Dropbox\\maestria - programacion analitica de datos\\clases\\p3 - SQL"
database_name = "simple_person.db"
person_repository = PersonRepository(f"{database_path}\\{database_name}")
```

Create

```python
person = Person()
person.name = "Guillermo"
person.last_name = "De Mendoza"
person.age = 36
person.city = "Monteria"

person_repository.create_person(person)
```

id	name	last_name	age	city
1	Mario	Bross	50	Bogota
2	Luigi	Bross	40	Medellin
3	Donkey	Kong	20	Cartagena
4	Bowser	Evil	70	Bogota
5	Peach	Princess	25	Barranquilla
6	Warrio	Bross	45	Santa Marta
10	Guillermo	De Mendoza	36	Monteria