# Curso
# Estructuras de datos

UNIVERSIDAD
DE ANTIOQUIA

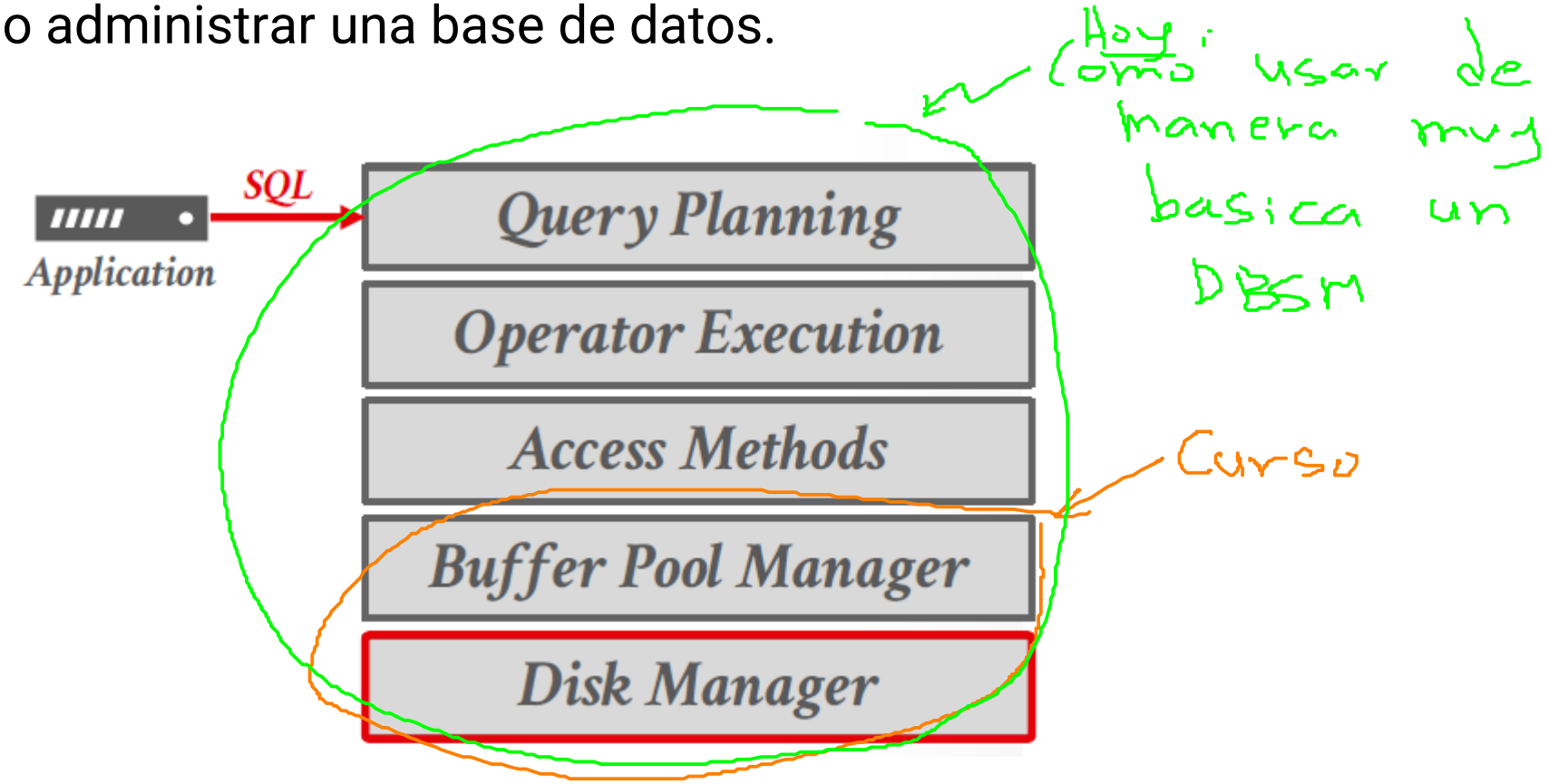
Clase 2 – Representación de los datos

20/02/2026

# Sobre el curso

UNIVERSIDAD DE ANTIOQUIA

- En este curso se abordaran algunos conceptos relacionados con el diseño y la implementación de sistemas de gestión de bases de datos orientados a disco.
- No es un curso sobre cómo usar una base de datos para construir aplicaciones ni sobre cómo administrar una base de datos.

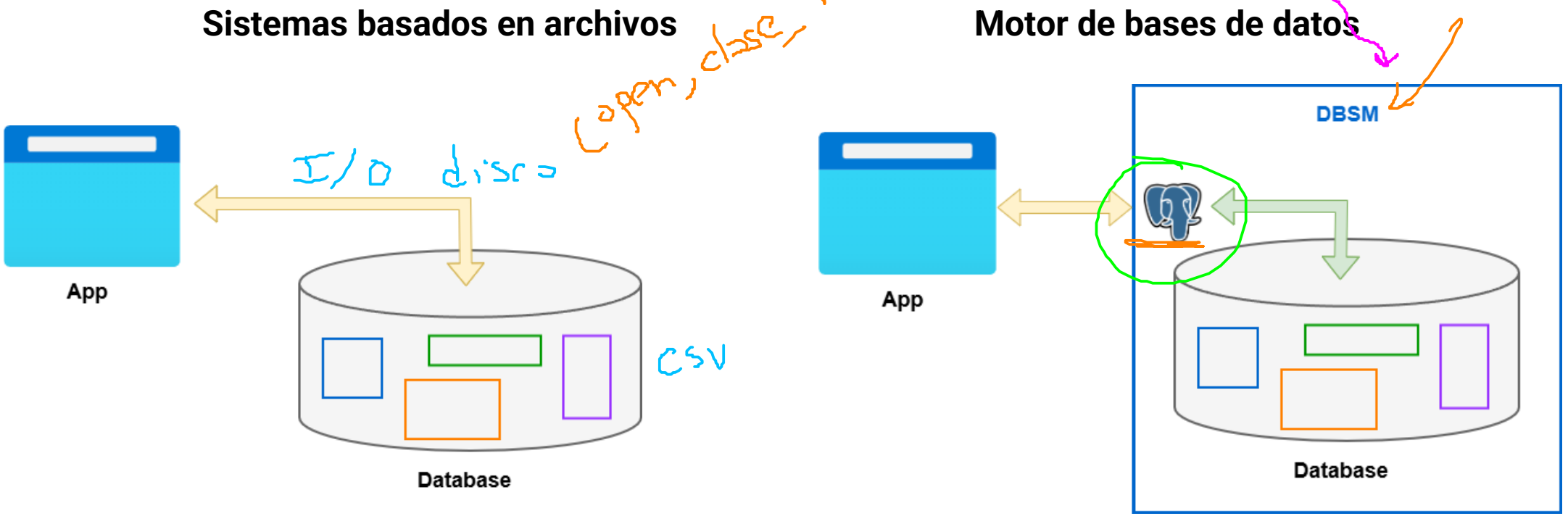


# Conceptos previos

UNIVERSIDAD DE ANTIOQUIA

La clase anterior tratamos algunos de los siguientes conceptos

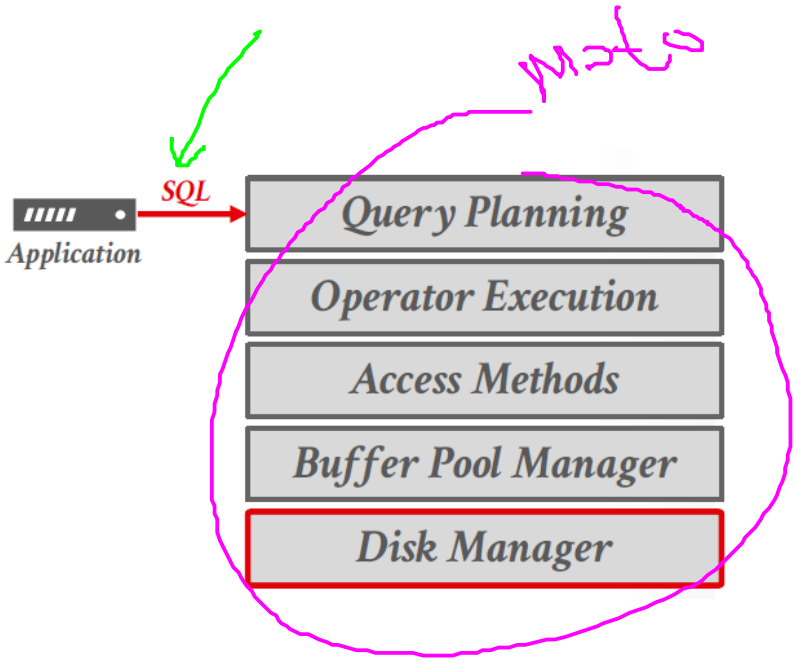
- Base de datos
- Manejo de bases de datos:
  - **Forma antigua**: Sistemas basados en archivos
  - **Actual**: Motor de bases de datos (DBSM)

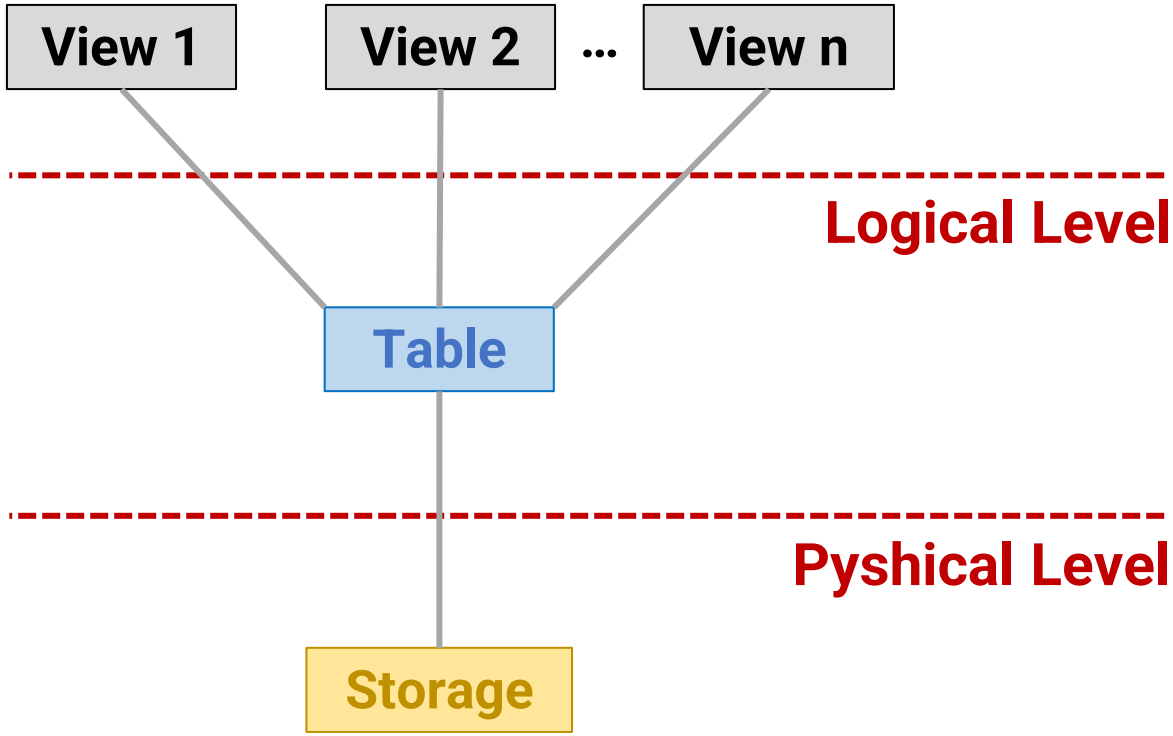


# Conceptos previos

UNIVERSIDAD DE ANTIOQUIA

- Motor de bases de datos (DBSM)
  - Arquitectura de tres capas.
  - Tipos de usuarios.
  - Componentes de un motor de base de datos.



# Sistema basado en archivos

UNIVERSIDAD DE ANTIOQUIA

**Crear una base de datos que modele una aplicación de música digital para llevar el registro de artistas y álbumes.**

- Información que necesitamos llevar en nuestra tienda:
  - Información sobre los artistas
  - Los álbumes que esos artistas han publicado

```
Artist(name, year, country)
"Billie Holiday", 1933, "USA"
"Ella Fitzgerald", 1917, "USA"
"Louis Armstrong", 1922, "USA"
```

```
Album(name, artist, year)
"Lady in Satin", "Billie Holiday", 1958
"Ella and Louis", "Ella Fitzgerald", 1957
"Ella and Louis", "Louis Armstrong", 1957
"What a Wonderful World", "Louis Armstrong", 1967
```

- **Ejemplo**: Obtener el año en que Louis inició su carrera como solista.

```
Artist(name, year, country)
"Billie Holiday", 1933, "USA"
"Ella Fitzgerald", 1917, "USA
"Louis Armstrong", 1922, "USA"
```

```
for line in file.readlines():
    record = parse(line)
    if record[0] == "Louis Armstrong":
        print(int(record[1]))
```

# Vista de los datos

UNIVERSIDAD DE ANTIOQUIA

- Recordemos que un sistema de base de datos es una colección de datos interrelacionados y un conjunto de programas que permiten a los **usuarios** acceder a dichos datos y modificarlos.
- Un propósito principal de un sistema de base de datos es proporcionar a los usuarios una visión abstracta de los datos.

## Base de datos de una universidad

UNIVERSIDAD DE ANTIOQUIA
1803

**Los datos consisten en información sobre**:

- Estudiantes
- Instructores
- Clases

**Ejemplos de programas de aplicación:**

- Agregar nuevos estudiantes, instructores y cursos
- Registrar estudiantes en cursos y generar listas de clase
- Asignar calificaciones a los estudiantes, calcular promedios de calificaciones (GPA) y generar historiales académicos

# Modelos de datos (Data Models)

UNIVERSIDAD DE ANTIOQUIA

Como meto los Datos al motor?

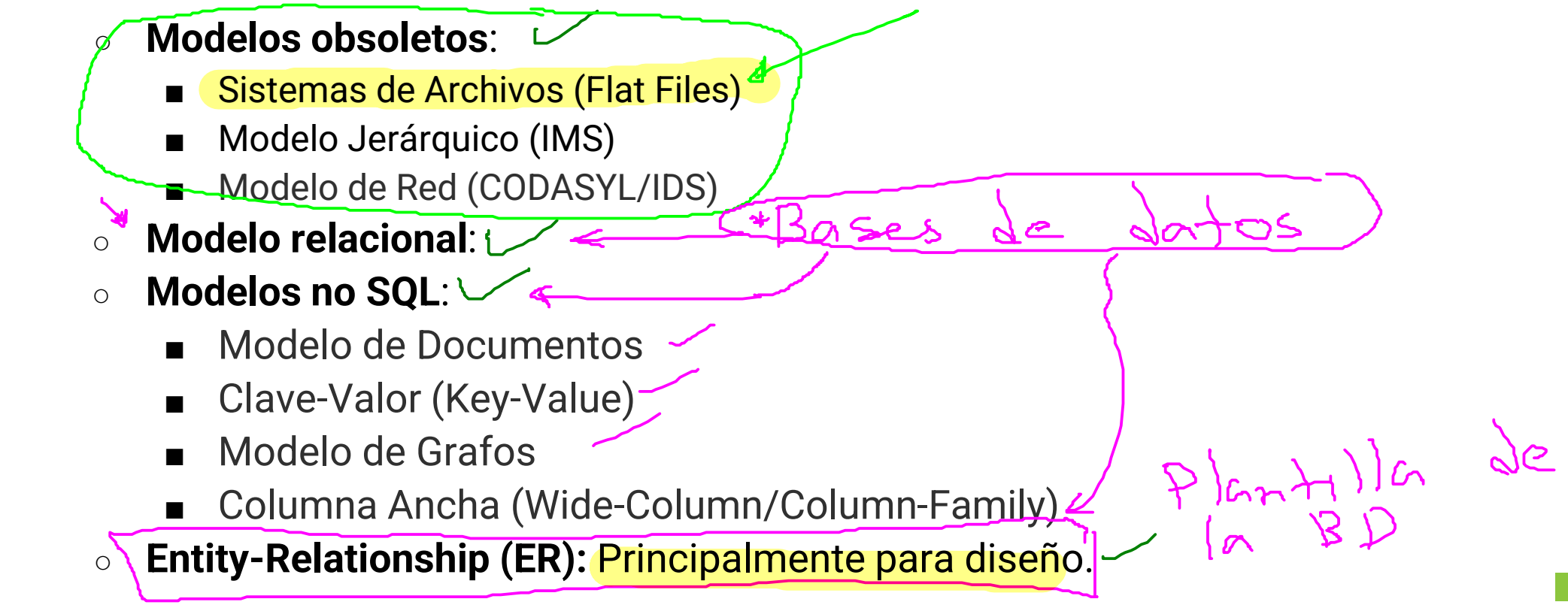
- Un **modelo de datos** es un conjunto de conceptos y reglas utilizados para describir la estructura de una base de datos.
- Para que un modelo sea completo, debe describir cuatro cosas:
  - **Data (Datos)**: Qué vamos a guardar (ej. Artistas y Álbumes).
  - **Relationships (Relaciones)**: Cómo se conectan (ej. Un artista crea un álbum).
  - **Semantics (Semántica)**: El significado de los datos (ej. Que el campo "año" se refiere a la fecha de lanzamiento y no a la edad del cantante).
  - **Constraints (Restricciones)**: Las reglas de seguridad (ej. "El año no puede estar vacío" o "El nombre del artista debe ser texto").
- Elegir el modelo de datos adecuado es crucial, porque afecta la eficiencia con la que podemos almacenar, acceder y gestionar los datos.

# Modelos de datos (Data Models)

UNIVERSIDAD DE ANTIOQUIA

## Tipos de modelos

- Elegir el modelo de datos adecuado es crucial, porque afecta la eficiencia con la que podemos almacenar, acceder y gestionar los datos.
- Existen diferentes modelos para describir la estructura de la información y sus relaciones
  - **Modelos obsoletos**:
    - Sistemas de Archivos (Flat Files)
    - Modelo Jerárquico (IMS)
    - Modelo de Red (CODASYL/IDS)
  - **Modelo relacional**:
  - **Modelos no SQL**:
    - Modelo de Documentos
    - Clave-Valor (Key-Value)
    - Modelo de Grafos
    - Columna Ancha (Wide-Column/Column-Family)
  - **Entity-Relationship (ER):** Principalmente para diseño.



# Modelos obsoletos

UNIVERSIDAD DE ANTIOQUIA

- A finales de la década de 1960, los primeros DBMS requerían que los desarrolladores escribieran las consultas usando código procedimental.
  - Ejemplos: IDS, IMS, CODASYL
- El desarrollador debía elegir explícitamente las rutas de acceso y el orden de ejecución, basándose en el contenido actual de la base de datos.
  - Si la base de datos cambiaba, entonces el desarrollador debía reescribir el código de la consulta.

Recuperar los nombres de los artistas que aparecen en el mixtape “DJ Mooshoo Tribute”.

```
PROCEDURE GET_ARTISTS_FOR_ALBUM;
BEGIN
    /* Declare variables */
    DECLARE ARTIST_RECORD ARTIST;
    DECLARE APPEARS_RECORD APPEARS;
    DECLARE ALBUM_RECORD ALBUM;

    /* Start navigation */
    FIND ALBUM USING ALBUM.NAME = "Mooshoo Tribute"
        ON ERROR DISPLAY "Album not found" AND EXIT;

    /* For each appearance on the album */
    FIND FIRST APPEARS WITHIN APPEARS.ALBUM OF ALBUM_RECORD
        ON ERROR DISPLAY "No artists found for this album" AND EXIT;

    /* Loop through the set of APPEARS */
    REPEAT
        /* Navigate to the corresponding artist */
        FIND OWNER WITHIN ARTIST.APPEARS OF APPEARS_RECORD
            ON ERROR DISPLAY "Error finding artist";

        /* Display artist name */
        DISPLAY ARTIST_RECORD.NAME;

        /* Move to the next APPEARS record in the set */
        FIND NEXT APPEARS WITHIN APPEARS.ALBUM OF ALBUM_RECORD
            ON ERROR EXIT;
    END REPEAT;
END PROCEDURE;
```

# Modelos obsoletos

UNIVERSIDAD DE ANTIOQUIA

Recuperar los nombres de los artistas que aparecen en el mixtape “DJ Mooshoo Tribute”.

```
PROCEDURE GET_ARTISTS_FOR_ALBUM;
BEGIN
    /* Declare variables */
    DECLARE ARTIST_RECORD ARTIST;
    DECLARE APPEARS_RECOR
    DECLARE ALBUM_RECOR

    /* Start navigati
    FIND ALBUM USING                    Tribute"
        ON ERROR DISP                   AND EXIT;

    /* For each appear nce on  album *
    FIND FIRST APPEAR ITHIN APP RS.AL  OF ALBUM_RECORD
        ON ERROR DISPLA rtis  for this album" AND
EXIT;

    /* Loop  e set o  EARS */
    REPEAT
        /* Navigate to  rrespo  artist */
        FIND OWNER WITHIN A  EARS OF APPEARS_RECORD
            ON ERROR DISPLAY  finding artist";

        /* Displa  name */
        DISP  RECORD.NAME;

        /* Move  the next APPEARS record in  set */
        FIND NEXT APPEARS WITHIN APPEARS.ALBUM OF ALBUM_RECORD
            ON ERROR EXIT;
    END REPEAT;
END PROCEDURE;
```

SQL

```
SELECT ARTIST.NAME
FROM ARTIST, APPEARS, ALBUM
WHERE ARTIST.ID = APPEARS.ARTIST_ID
  AND APPEARS.ALBUM_ID = ALBUM.ID
  AND ALBUM.NAME = "Mooshoo Tribute"
```

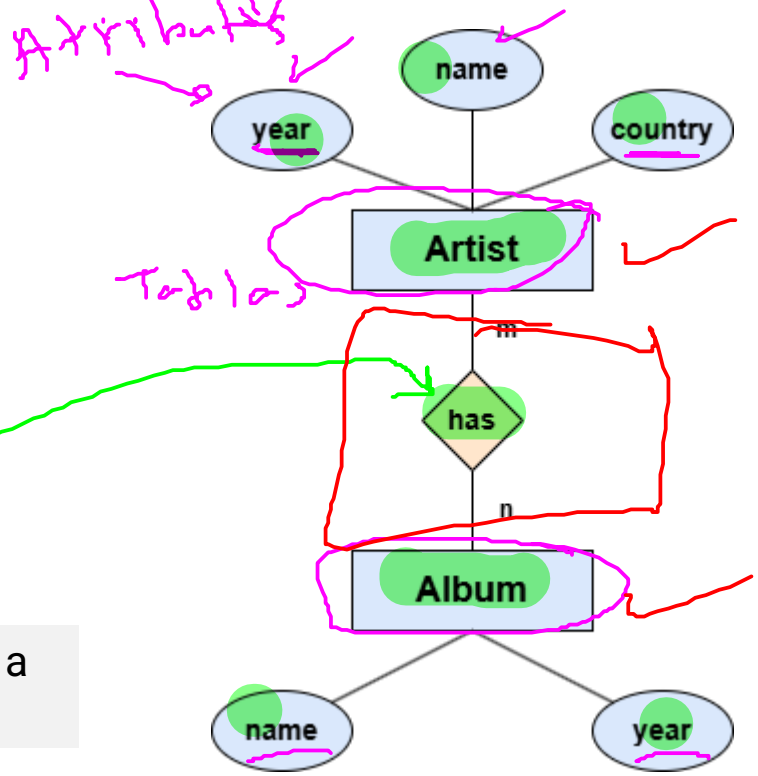
# Modelo Entidad - Relación

UNIVERSIDAD DE ANTIOQUIA

- Propuesto originalmente por Chen en 1976, se utiliza fundamentalmente durante la fase de **diseño conceptual** de una base de datos.
- Define la arquitectura de la base de datos.

**Crear una base de datos que modele una aplicación de música digital para llevar el registro de artistas y álbumes.**

- **Entidades**:
  - Los **Artistas** tienen nombre, el año en que iniciaron y su país de origen.
  - Los **Álbumes** tienen nombre y año de lanzamiento.
- **Relaciones**:
  - Un **Álbum** tiene uno o más **Artistas**.
  - Un **Álbum** tiene múltiples **Pistas**.




**Importante**: El modelo ER no es la base de datos. Es el diseño conceptual previo a crear las tablas.

# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Base de datos relacional

- Todos los datos se almacenan en varias **tablas** relacionadas (también conocidas como **relaciones**)
- **Relación**: Conjunto de filas o tupas
  - Todas las filas son distintas.
  - No hay orden entre las filas.

| name | year | Country |
|---|---|---|
| Billie Holiday | 1933 | USA |
| Ella Fitzgerald | 1917 | USA |
| Louis Armstrong | 1922 | USA |

**Tabla Artist**

| name | artist | year |
|---|---|---|
| Lady in Satin | Billie Holiday | 1958 |
| Ella and Louis | Ella Fitzgerald | 1957 |
| Ella and Louis | Louis Armstrong | 1957 |
| What a Wonderful World | Louis Armstrong | 1922 |

**Tabla Album**

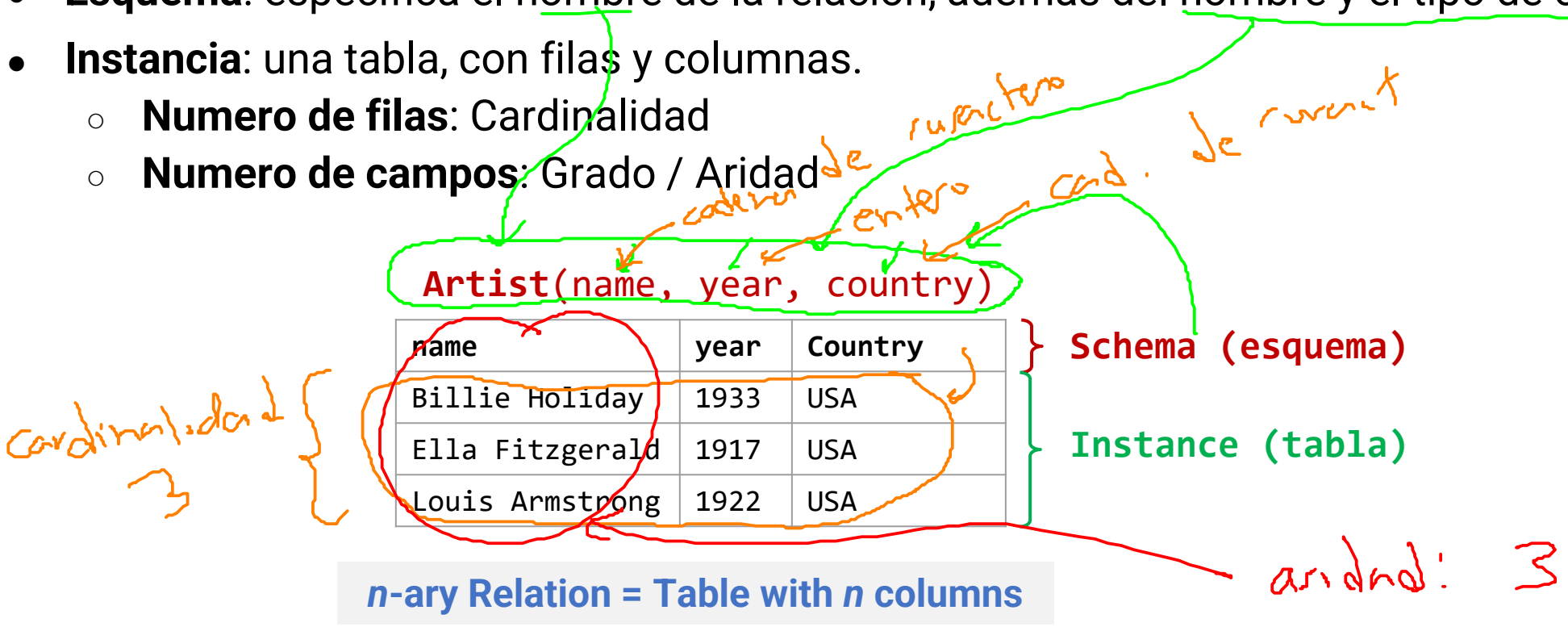
# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Estructura de una tabla (Relación)

Esta compuesta de dos partes

- **Esquema**: especifica el nombre de la relación, además del nombre y el tipo de cada columna.
- **Instancia**: una tabla, con filas y columnas.
  - **Numero de filas**: Cardinalidad
  - **Numero de campos**: Grado / Aridad




*n*-ary Relation = Table with *n* columns

# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Clave Primaria (Primary Key - PK)

- La **clave primaria** es un campo o conjunto de campos que identifica de forma **única** a cada registro de una tabla.
- En una tabla, no pueden existir dos filas con el mismo valor en la clave primaria, y bajo ninguna circunstancia se permiten valores NULL en ella
- El DBMS crea automáticamente un índice único sobre la clave primaria para facilitar búsquedas rápidas y validar la integridad en cada inserción.

**Artist**(name, year, country)

| name | year | Country |
|---|---|---|
| Billie Holiday | 1933 | USA |
| Ella Fitzgerald | 1917 | USA |
| Louis Armstrong | 1922 | USA |

**Artist**(id, name, year, country)

| id | Name | year | Country |
|---|---|---|---|
| 101 | Billie Holiday | 1933 | USA |
| 102 | Ella Fitzgerald | 1917 | USA |
| 103 | Louis Armstrong | 1922 | USA |

# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Clave Foránea (Foreign Key - FK)

- Una **clave foránea** es una copia de la clave primaria de una tabla "padre" (tabla referenciada por otra tabla) que se coloca en una tabla "hijo" (tabla que depende de la tabla padre) para crear una relación lógica entre ambas
- A diferencia de la PK, una clave foránea puede contener duplicados (permitiendo que un padre tenga muchos hijos) y también puede aceptar valores NULL si la relación entre las tablas es opcional.
- La finalidad principal de la FK es garantizar la integridad referencial, asegurando que cualquier valor introducido en la tabla hijo corresponda efectivamente a un registro válido en la tabla padre

**Artist_Album**(artist_id, album_id)

| artist_id | album_id |
|---|---|
| 101 | 11 |
| 102 | 22 |
| 103 | 22 |
| 103 | 33 |

**Artist**(id, name, year, country)

| id | Name | year | Country |
|---|---|---|---|
| 101 | Billie Holiday | 1933 | USA |
| 102 | Ella Fitzgerald | 1917 | USA |
| 103 | Louis Armstrong | 1922 | USA |

**Album**(id, name, artist, year)

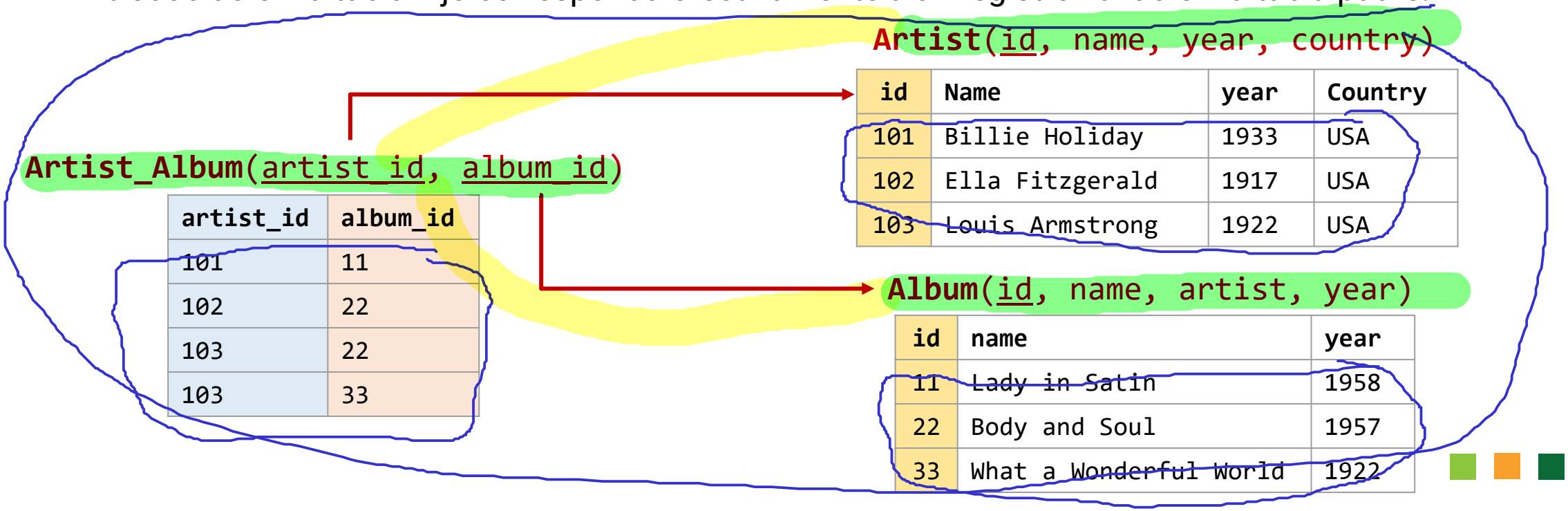
| id | name | artist | year |
|---|---|---|---|
| 11 | Lady in Satin | 101 | 1958 |
| 22 | Ella and Louis | ??? | 1957 |
| 33 | What a Wonderful World | 103 | 1922 |

# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Clave Foránea (Foreign Key - FK)

- Una **clave foránea** es una copia de la clave primaria de una tabla "padre“ (tabla referenciada por otra tabla) que se coloca en una tabla "hijo" (tabla que depende de la tabla padre) para crear una relación lógica entre ambas
- A diferencia de la PK, una clave foránea puede contener duplicados (permitiendo que un padre tenga muchos hijos) y también puede aceptar valores NULL si la relación entre las tablas es opcional.
- La finalidad principal de la FK es garantizar la **integridad referencial**, asegurando que cualquier valor introducido en la tabla hijo corresponda efectivamente a un registro válido en la tabla padre.

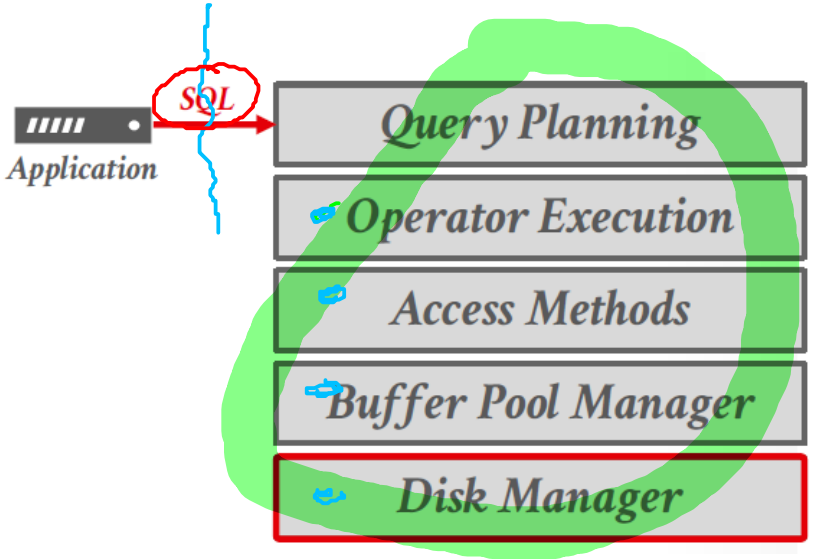


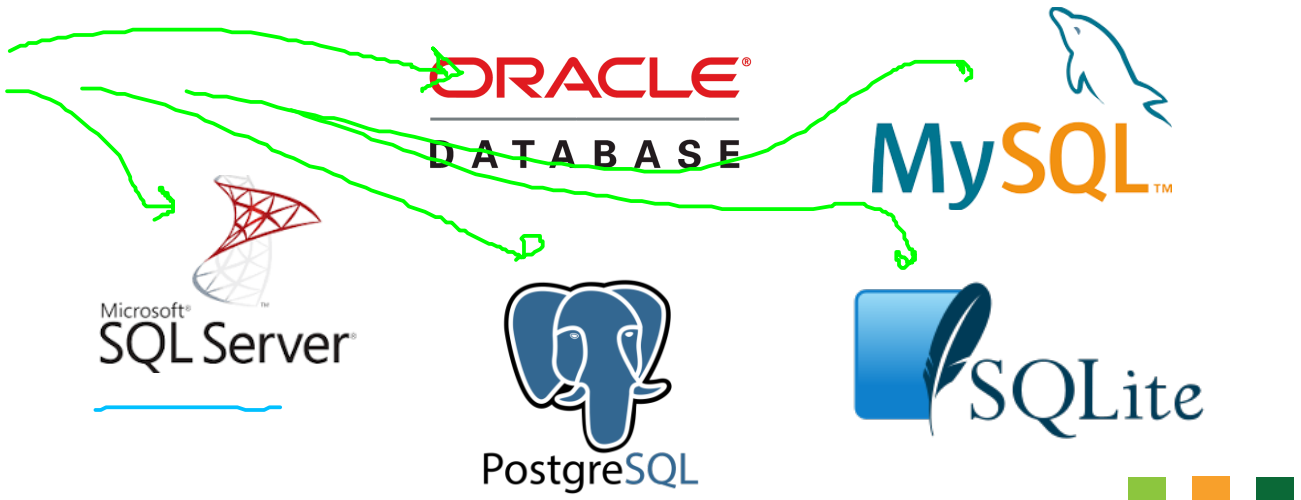
# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## Lenguaje SQL

- Es el lenguaje estándar y más utilizado para crear, manipular y consultar bases de datos relacionales.
- Es un lenguaje declarativo o no procedimental.
- Algunos motores SQL: SQLite, MySQL, PostgreSQL, Oracle SQL, SQL Sever.
- El SQL se divide en varios sublenguajes especializados según la tarea:
  - **DDL (Data Definition Language)**: Definicion y modificación de la estructura de la base de datos.
  - **DML (Data Manipulation Language)**: Consulta y actualización de la base de datos.
  - **DCL (Data Control Language)**: Controla el acceso y la seguridad de la base de datos.
  - **TCL (Transaction Control Language)**: Gestiona las transacciones en la base de datos.



# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## DDL (Data Definition Language)

Notación empleada para definir el esquema de la base de datos. Permite:

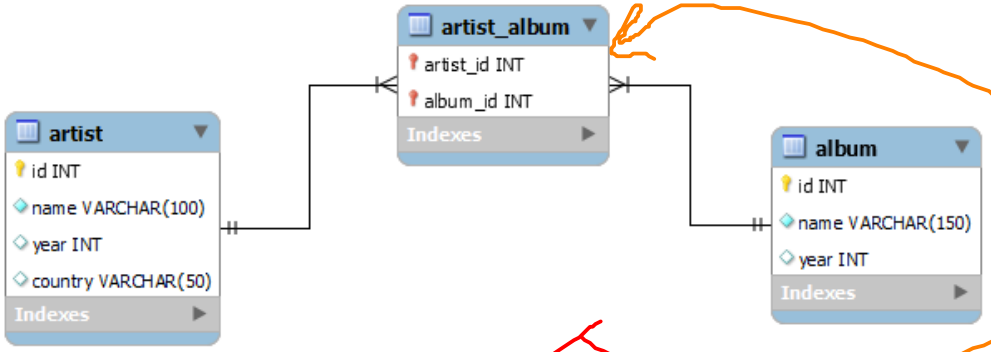
- Crear, modificar, eliminar relaciones
- Especificar restricciones
- Administrar usuarios, seguridad, etc.

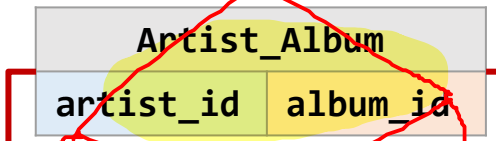
```
CREATE DATABASE music_db;
USE music_db;

CREATE TABLE Artist (
  id INT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  year INT,
  country VARCHAR(50)
);

CREATE TABLE Album (
  id INT PRIMARY KEY,
  name VARCHAR(150) NOT NULL,
  year INT
);

CREATE TABLE Artist_Album (
  artist_id INT NOT NULL,
  album_id INT NOT NULL,
  PRIMARY KEY (artist_id, album_id),
  FOREIGN KEY (artist_id) REFERENCES Artist(id),
  FOREIGN KEY (album_id) REFERENCES Album(id)
);
```






# Modelo relacional

UNIVERSIDAD DE ANTIOQUIA

## DML (Data Manipulation Language)

También conocido como lenguaje de consulta, permite acceder y actualizar los datos:

- Especificar consultas para encontrar tuplas que satisfacen ciertos criterios
- Agregar, modificar, eliminar tuplas

```
INSERT INTO Artist (id, name, year, country) VALUES
(101, 'Billie Holiday', 1933, 'USA'),
(102, 'Ella Fitzgerald', 1917, 'USA'),
(103, 'Louis Armstrong', 1922, 'USA');

INSERT INTO Album (id, name, year) VALUES
(11, 'Lady in Satin', 1958),
(22, 'Body and Soul', 1957),
(33, 'what a wonderful world', 1922);

INSERT INTO Artist_Album (artist_id, album_id) VALUES
(101, 11),
(102, 22),
(103, 22),
(103, 33);
```

**Artist**

| id | Name | year | Country |
|---|---|---|---|
| 101 | Billie Holiday | 1933 | USA |
| 102 | Ella Fitzgerald | 1917 | USA |
| 103 | Louis Armstrong | 1922 | USA |

**Album**

| id | name | year |
|---|---|---|
| 11 | Lady in Satin | 1958 |
| 22 | Body and Soul | 1957 |
| 33 | What a Wonderful World | 1922 |

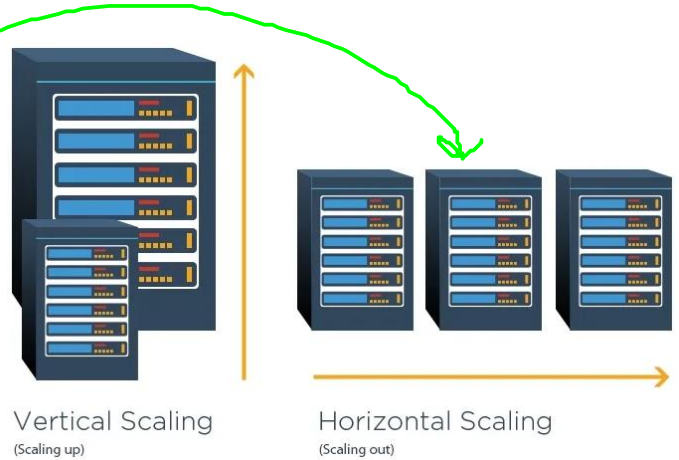
**Artist_Album**

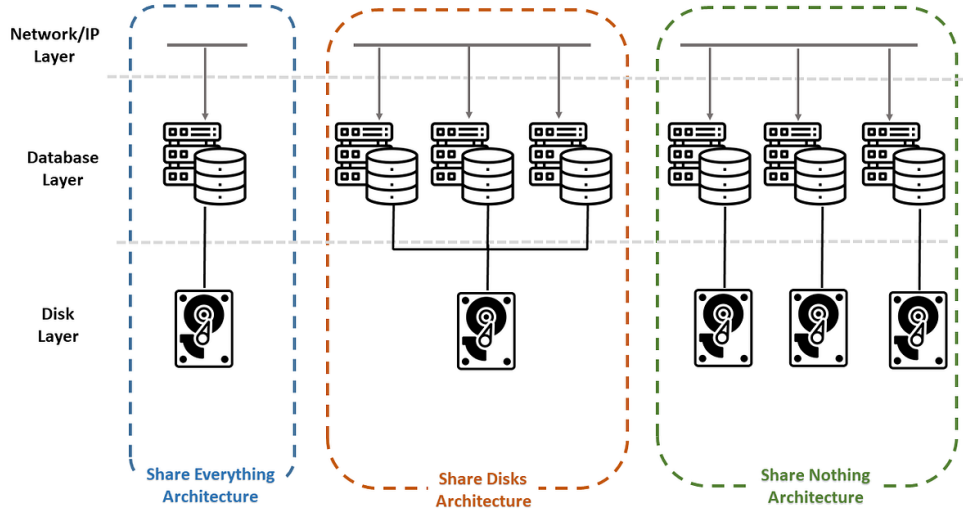
| artist_id | album_id |
|---|---|
| 101 | 11 |
| 102 | 22 |
| 103 | 22 |
| 103 | 33 |

# Modelo NoSQL

UNIVERSIDAD DE ANTIOQUIA

- **Por que surgen**
  - Crecimiento masivo de datos (Big Data)
  - Necesidad de escalabilidad horizontal (scale-out)
  - Sistemas distribuidos y alta disponibilidad
- **Características**
  - En NoSQL no es necesario definir una estructura rígida antes de guardar los datos. La forma de los datos se interpreta al ser consultada.
  - La responsabilidad de entender los datos pasa de la Base de Datos al Programador (Schema-on-read).
  - La base de datos es solo un almacén veloz; el código de la aplicación es el que debe saber cómo leer lo que hay dentro.
  - Diseñadas para distribuirse en múltiples nodos independientes (arquitectura shared-nothing).



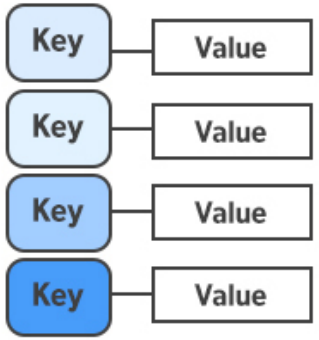



# Modelo NoSQL

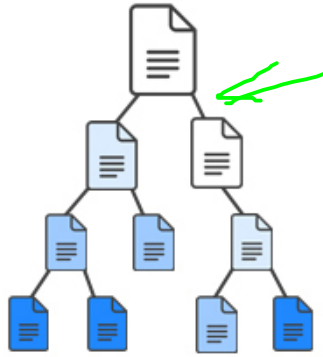
UNIVERSIDAD DE ANTIOQUIA

## Tipos

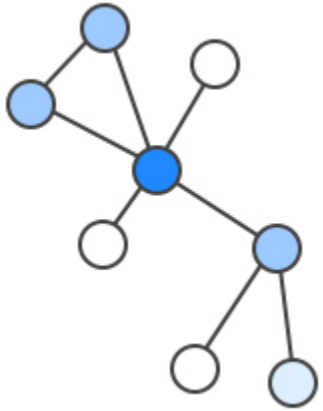
**Clave-Valor (key-value)**: El más simple y escalable; recupera registros mediante una clave única (ej. Redis, Riak, Voldemort).



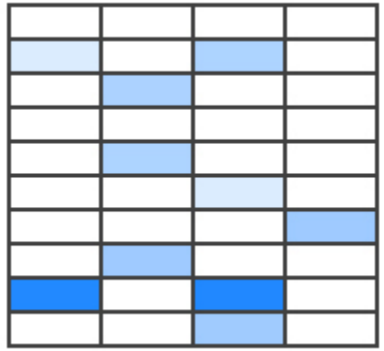

**Documentos (Document)**: Almacena datos en objetos anidados autocontenidos, usualmente JSON o BSON (ej. MongoDB, CouchDB).



**Grafos (Graph)**: Enfocados en las conexiones; representan datos mediante vértices y aristas para analizar interconectividad compleja (ej. Neo4j, AllegroGraph).



**Columna Ancha (Wide-Column)**: Mapas multidimensionales ordenados, optimizados para escaneos masivos de datos y alta escritura (ej. Cassandra, HBase, BigTable).



# Modelo NoSQL

UNIVERSIDAD DE ANTIOQUIA

**Artist**(name, year, country)

```
"Billie Holiday", 1933, "USA"
"Ella Fitzgerald", 1917, "USA"
"Louis Armstrong", 1922, "USA"
```

**Album**(name, artist, year)

```
"Lady in Satin", "Billie Holiday", 1958
"Ella and Louis", "Ella Fitzgerald", 1957
"Ella and Louis", "Louis Armstrong", 1957
"What a Wonderful World", "Louis Armstrong", 1967
```

```
[
  {
    "_id": 101,
    "name": "Billie Holiday",
    "year": 1933,
    "country": "USA",
    "albumsId": [11]
  },
  {
    "_id": 102,
    "name": "Ella Fitzgerald",
    "year": 1917,
    "country": "USA",
    "albumsId": [22]
  },
  {
    "_id": 103,
    "name": Louis Armstrong",
    "year": 1922,
    "country": "USA",
    "albumsId": [22, 33]
  },
]
```

```
[
  {
    "_id": 11,
    "name": "Lady in Satin",
    "year": 1958,
    "artistId": [101]
  },
  {
    "_id": 22,
    "name": "Ella and Louis",
    "year": 1957,
    "artistId": [102, 103]
  },
  {
    "_id": 33,
    "name": "What a Wonderful World",
    "year": 1922,
    "artistId": [103]
  },
]
```

# Modelo NoSQL

UNIVERSIDAD DE ANTIOQUIA

```
use music_db

// (opcional) limpiar para evitar duplicados
db.artists.deleteMany({})
db.albums.deleteMany({})

// Insertar artistas
db.artists.insertMany([
  { _id: 101, name: "Billie Holiday", year: 1933, country: "USA", albumIds: [11] },
  { _id: 102, name: "Ella Fitzgerald", year: 1917, country: "USA", albumIds: [22] },
  { _id: 103, name: "Louis Armstrong", year: 1922, country: "USA", albumIds: [22, 33] }
])

// Insertar álbumes
db.albums.insertMany([
  { _id: 11, name: "Lady in Satin", year: 1958, artistIds: [101] },
  { _id: 22, name: "Ella and Louis", year: 1957, artistIds: [102, 103] },
  { _id: 33, name: "What a Wonderful World", year: 1967, artistIds: [103] }
])
```

# Referencias

UNIVERSIDAD DE ANTIOQUIA


- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). **Database System Concepts (7th Edition)**. McGraw-Hill – Capitulo 1
- https://15445.courses.cs.cmu.edu/spring2026/
- https://www.mongodb.com/resources/basics/databases/nosql-explained
- https://blog.bytebytego.com/p/sql-vs-nosql-choosing-the-right-database
- https://cs186berkeley.net/notes/note17/
- https://www.datacamp.com/tutorial/beginners-introduction-postgresql
- https://github.com/Wathon/data_engineering_with_python-track-datacamp/tree/main

# UNIVERSIDAD DE ANTIOQUIA

Curso de Estructuras de datos

Clase 2 – Representación de los datos