Bases de Datos

Clase 3: SQL

Hasta ahora

- Tenemos un lenguaje teórico para realizar consultas a relaciones
- Queremos un programa con tablas y un lenguaje de consultas para utilizar en la práctica.

Relational

Data

Base

Modeling

System

Cómo funciona un RDBMS ?

DBMS Relacional

- Un DBMS relacional es un programa que se instala en un computador (**servidor**)
- Este programa se mantiene escuchando conexiones
- El usuario (generalmente otro programa) se conecta (**cliente**) al programa y le puede entregar instrucciones.

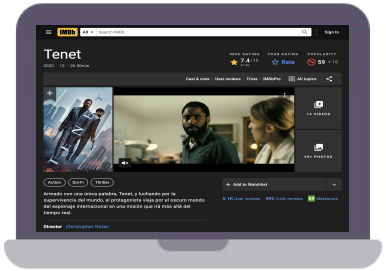
DBMS Relacional

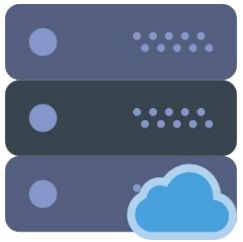
DBMS Relacional

https://www.imdb.com/title/tt6723592/

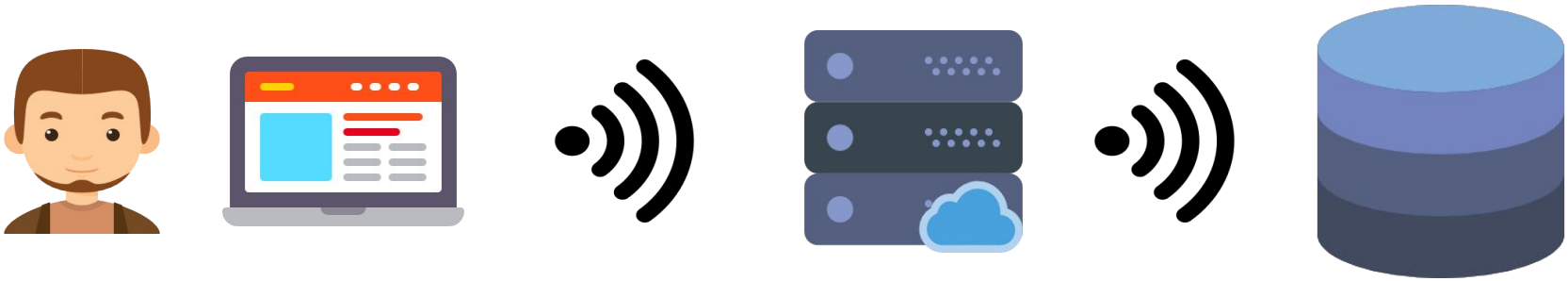
DBMS Relacional

DBMS Relacional

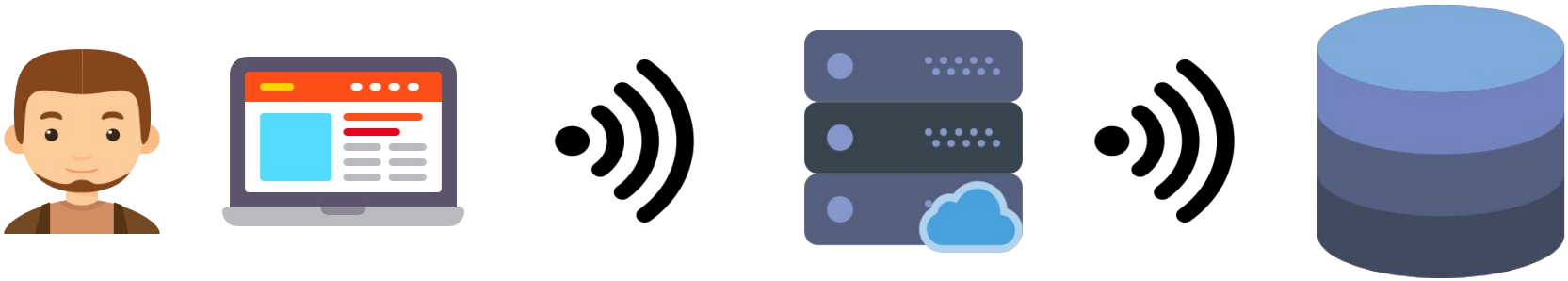




DBMS Relacional

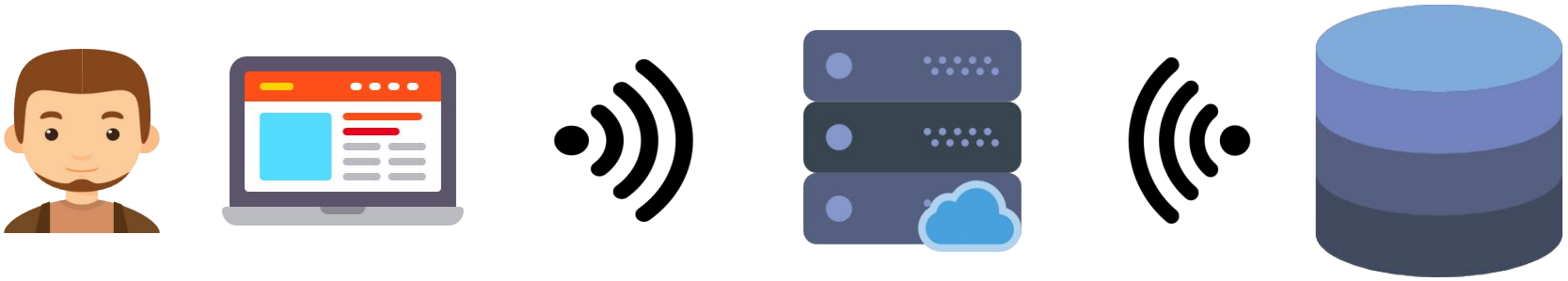


DBMS Relacional



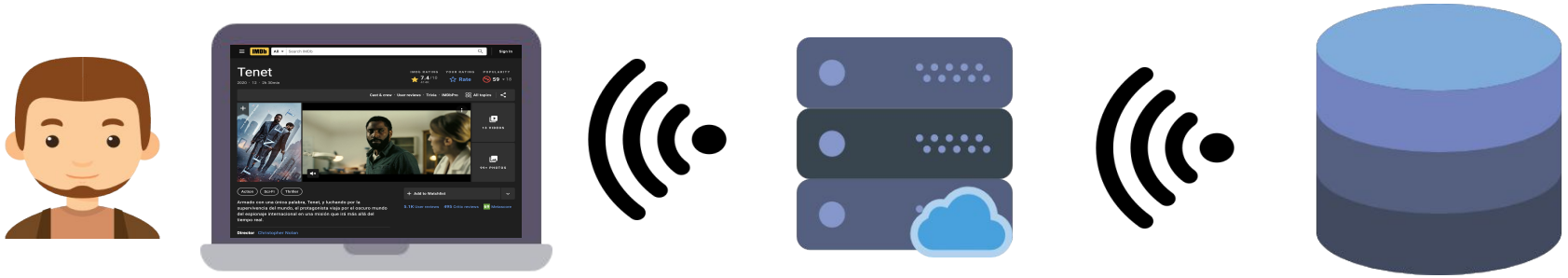
```
SELECT name, score
FROM titles
WHERE title.id='tt6723592'
```

DBMS Relacional



name	score
Tenet	7.4

DBMS Relacional



name	score
Tenet	7.4

SQL

Structured Query Language

- Usado para todas las comunicaciones con bases de datos relacionales.
- Aplicaciones web, locales, móviles, análisis de datos, etc.
- Hasta algunas arquitecturas serverless lo requieren o usan lenguajes basados en SQL.

SQL

Structured Query Language

- Último estándar SQL99 (SQL3)
- Softwares implementan “subconjunto” del estándar (cada uno tiene diferencias sutiles)
- Lenguaje declarativo

Declarativo vs. Procedural

- SQL es declarativo, decimos lo que queremos, pero sin dar detalles de cómo lo computamos
- El DBMS transforma la consulta SQL en en un algoritmo ejecutado sobre un lenguaje procedural
- Un lenguaje como Java es procedural: para hacer algo debemos indicar paso a paso el procedimiento

SQL

Structured Query Language

- DDL: Lenguaje de definición de datos
  - Crear y modificar tablas, atributos y llaves
- DML: Lenguaje de manipulación de datos
  - Consultar una o más tablas
  - Insertar, eliminar, modificar tuplas

En este curso

Durante el curso vamos a usar un motor RDBMS:

- PostgreSQL (PSQL)

PSQL

PSQL es un sistema relacional *open source*

Tiene varias funcionalidades avanzadas, como por ejemplo el uso de procedimientos almacenados o el almacenamiento de JSON

PSQL

PSQL va a ser usado en el proyecto y parte de la cátedra

Cada grupo tendrá acceso a un servidor en el que dispondrán de una base de datos a la que su aplicación se va a conectar

Los datos son almacenados en múltiples archivos en carpetas ocultas del computador

SQL

SQL

Tipos de datos

- Caracteres (Strings)
  - char(20) - Largo fijo
  - varchar(20) - Largo variable
- Números
  - int, smallint, float, …
- Tiempo y fecha
  - time - hora formato 24 hrs.
  - date - fecha
  - timestamp - fecha + hora

- Y varios otros! Dependen del RDBMS que se esté usando.

SQL

Creando un esquema

Consideremos el esquema de ejemplo:

Peliculas(id, nombre, año, categoria, calificacion, director)

Actores(id, nombre, edad)

Actuo_en(id_actor, id_pelicula)

SQL

Crear Tablas

```
CREATE TABLE Peliculas(
    id int,
    nombre varchar(30),
    año int,
    categoria varchar(30),
    calificacion float,
    director varchar(30)
)
```

SQL

Crear Tablas

```sql
CREATE TABLE Peliculas(
    id int,
    nombre varchar(30),
    año int,
    categoria varchar(30),
    calificacion float,
    director varchar(30)
)
```

```sql
CREATE TABLE Actores(
    id int,
    nombre varchar(30),
    edad int
)
```

```sql
CREATE TABLE Actuo_en(
    id_actor int,
    id_pelicula int,
)
```

SQL

Crear Tablas con llaves

```
CREATE TABLE Peliculas(
    id int PRIMARY KEY,
    nombre varchar(30),
    año int,
    categoria varchar(30),
    calificacion float,
    director varchar(30)
)
```

```
CREATE TABLE Actores(
    id int PRIMARY KEY,
    nombre varchar(30),
    edad int
)
```

```
CREATE TABLE Actuo_en(
    id_actor int,
    id_pelicula int,
    PRIMARY KEY (id_pelicula, id_actor)
)
```

SQL

Valores Default

Sintaxis general:

```
CREATE TABLE <Nombre> (...<atr> tipo DEFAULT <valor>...)
```

Ejemplo:

```
CREATE TABLE Peliculas(
    id int PRIMARY KEY,
    nombre varchar(30),
    año int,
    categoria varchar(30) DEFAULT 'Acción',
    calificacion float DEFAULT 0,
    director varchar(30)
)
```

SQL

Modificar Tablas

Eliminar tabla:

```
DROP TABLE Peliculas
```

Eliminar atributo:

```
ALTER TABLE Peliculas DROP COLUMN director
```

Agregar atributo:

```
ALTER TABLE Peliculas ADD COLUMN productor varchar(30)
```

SQL

Insertar Datos

Sintaxis general:

```
INSERT INTO R(at_1, ..., at_n) VALUES (v_1, ..., v_n)
```

Ejemplo:

```
INSERT INTO Peliculas(id, nombre, año, categoria, calificacion,
director) VALUES (321351, 'V for Vendetta', 2005,'Action', 8.2
,'James McTeigue')
```

Ejemplo abreviado (asume orden de creación):

```
INSERT INTO Peliculas VALUES (321351, 'V for Vendetta',
2005,'Action', 8.2 ,'James McTeigue')
```

Consultando con SQL

SQL

Forma básica

Las consultas en general se ven:

SQL

Forma básica

Las consultas en general se ven:

```
SELECT atributos
FROM relaciones
WHERE condiciones
```

SQL

Forma básica

Para ver todo de una tabla (en este caso película):

```
SELECT * FROM Peliculas
```

Para ver nombre y calificación de todas las películas dirigidas por Nolan:

```
SELECT nombre, calificacion
FROM Peliculas
WHERE director = 'C. Nolan'
```

SQL

Forma básica

Para las películas estrenadas desde el 2010:

```
SELECT *
FROM Peliculas
WHERE año >= 2010
```

El `WHERE` permite =, <>, !=, >, <, <=, >=, AND, OR, NOT, IN, BETWEEN, etc...

SQL

En General

La consulta:

```
SELECT a_1, ..., a_n
FROM T_1, ..., T_m
WHERE <condicion>
```

Se traduce al álgebra relacional como:

$$\pi_{a_1,\ldots,a_n}(\sigma_{condiciones}(T_1 \times \cdots \times T_m))$$

Update

Para actualizar valores de una tabla:

```
UPDATE Peliculas
SET calificacion = 0
WHERE name = 'Sharknado 6'
```

Update

Forma general

```
UPDATE R
SET <Nuevos valores>
WHERE <Condición sobre R>
```

<Nuevos valores> → (atributo$_1$ = nuevoValor$_1$, …, atributo$_n$ = nuevoValor$_n$)

Delete

Para borrar filas que cumplan una condición:

```
DELETE FROM R
WHERE <Condición sobre R>
```

¿Qué pasa si se nos olvida el `WHERE` en un `UPDATE` o `DELETE FROM`?

```
UPDATE Users
SET password = 'contraseña'
```

```
DELETE FROM Tweets
```

… ¿O si se nos pasa un ‘;’ entre medio?

```
UPDATE Users
SET password = 'contraseña';
WHERE email = 'some@email.com'

DELETE FROM Tweets;
WHERE user_name ='realDonaldTrump'
```

¿Qué pasa si se nos olvida el `WHERE` en un `UPDATE` o `DELETE FROM`?

… ¿O si se nos pasa un '`;`' entre medio?

Se borran / actualizan todas las filas!!

Producto cruz

Si pedimos datos de más de una tabla la base de datos va hacer un producto cruz y entregará nxm filas.

```
SELECT  *
FROM Peliculas, Actuo_en
```

SQL

Joins

Podemos hacer un join agregando un `WHERE`

Por ejemplo, para obtener todas las películas junto a los ids de los actores que participaron en ella:

```
SELECT *
FROM Peliculas, Actuo_en
WHERE id = id_pelicula
```

Observación: id es atributo de Peliculas, mientras que id_pelicula es atributo de Actuo_en

SQL

Joins - Desambiguando atributos

Entregue todas las películas junto a los id de los actores que participaron en ella:

```
SELECT *
FROM Peliculas, Actuo_en
WHERE Peliculas.id = Actuo_en.id_pelicula
```

Sirve cuando tenemos atributos en distintas tablas con el mismo nombre y para agregarle claridad a la consulta.

SQL

Joins

¿Y si queremos los nombres de los actores en vez de los ids?

```
SELECT Peliculas.nombre, Actores.nombre
FROM Peliculas, Actuo_en, Actores
WHERE Peliculas.id = Actuo_en.id_pelicula
AND Actores.id = Actuo_en.id_actor
```

SQL

Alias

Podemos acortar la consulta anterior:

```
SELECT p.nombre, a.nombre
FROM Peliculas as p, Actuo_en as ae, Actores as a
WHERE p.id = ae.id_pelicula
AND a.id = ae.id_actor
```

Ese tipo de alias no es muy recomendable

SQL

Alias

Podemos hacer operaciones y nombrar la columna:

```
SELECT (nombre || ' dirigida por ' || director) as creditos, año
FROM Peliculas
```

creditos	año
V for Vendetta dirigida por James McTeigue	2005
Dunkirk dirigida por C. Nolan	2017

SQL

Ordenando

Entregue el nombre y la calificación de todas las películas (orden ascendente):

```
SELECT nombre, calificacion
FROM Peliculas
ORDER BY nombre, calificacion
```

SQL

Ordenando

Entregue el nombre y la calificación de todas las películas (orden ascendente):

```
SELECT nombre, calificacion
FROM Peliculas
ORDER BY nombre, calificacion
```

El i-ésimo atributo del ORDER BY resuelve un empate en el atributo i-1

SQL

Ordenando

Entregue el nombre y la calificación de todas las películas (orden descendente):

```
SELECT nombre, calificacion
FROM Peliculas
ORDER BY nombre DESC, calificacion
```

SQL

Union

Entregue el nombre de todos actores y directores:

```
SELECT nombre
FROM Actores
UNION
SELECT director
FROM Peliculas
```

SQL

Operadores de conjuntos

- `EXCEPT`: diferencia del álgebra
- `UNION`: unión del álgebra
- `INTERSECT`: intersección del álgebra
- `UNION ALL`: unión que admite duplicados

SQL

Matching de patrones con LIKE

`s LIKE p`: string s es como p, donde p es un patrón definido mediante:

- `%` Cualquier secuencia de caracteres
- `_` Cualquier caracter (solamente uno)

```
SELECT *
FROM Peliculas
WHERE name LIKE '%Spiderman%'
```

SQL

Eliminando duplicados

Entregue todos los nombres distintos de las películas:

```
SELECT DISTINCT nombre
FROM Peliculas
```

OJO: `DISTINCT` es un operador en sí mismo.