

Bases de Datos


Clase 10: Lógica en la BDD

Lógica en la BDD

- En desarrollo de software le llamamos *business logic*, lógica del negocio o simplemente lógica a todas las reglas, algoritmos, etc. que definen el sistema y que están dados por lo que quiere lograr la aplicación.
- Los motores relacionales más complejos como Postgres o MySQL tienen funcionalidades que permiten implementar parte de esa lógica directamente en la BDD.

Vistas

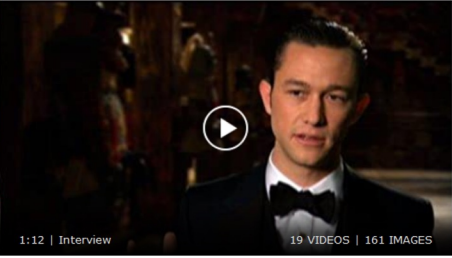

Motivación

 Menu All Search IMDb IMDbPro Watchlist

Christopher Nolan (1)

Top 500

Writer | Producer | Director

1:12 | Interview19 VIDEOS | 161 IMAGES

Best known for his cerebral, often nonlinear, storytelling, acclaimed writer-director [Christopher Nolan](#) was born on July 30, 1970, in London, England. Over the course of 15 years of filmmaking, Nolan has gone from low-budget independent films to working on some of the biggest blockbusters ever made. At 7 years old, Nolan began making short movies ... [See full bio](#) »

Born: July 30, 1970 in London, England, UK

More at IMDbPro »
[Contact Info:](#) View agent, publicist, legal on IMDbPro

Nominated for 5 Oscars. Another 139 wins & 227 nominations. [See more awards](#) »




Quick Links

Biography
Awards
Photo Gallery

Filmography (by Job)
Trailers and Videos

[Explore More](#)




Reboots & Remakes We Can't Wait to See



From "Dexter" to *The Suicide Squad*, here are our picks for the reboots and remakes we're most excited for in 2021 and beyond.

[Browse our picks](#) »

Share this page:



Motivación

The image is a screenshot of the IMDb website, overlaid with a semi-transparent version of the same site to show a transition or comparison. The top section features the IMDb logo, a navigation menu, and a search bar. Below this, the profile of Christopher Nolan is displayed, including his name, roles (Writer, Producer, Director), a "Top 500" badge, and a video player showing an interview. To the right of the profile are "Quick Links" for Biography, Awards, Photo Gallery, Filmography, and Trailers and Videos, along with an "Explore More" button. Below the profile is a section titled "Reboots & Remakes We Can't Wait to See" with movie posters for "The Matrix", "The Dark Knight", and "The Dark Knight Rises". The bottom section of the image shows the IMDb interface for the movie "The Prestige" (2006), including the full cast and crew, trivia, user reviews, IMDbPro link, more options, and a share button. The movie's rating is 8.5/10 with 1,207,788 votes, and it is categorized as PG-13, 2h 10min, Drama, Mystery, Sci-Fi, released on 20 October 2006 (USA).

IMDb Menu All Search IMDb IMDbPro Watchlist

Christopher Nolan (1)

Writer | Producer | Director

1:12 | Interview

19 VIDEOS | 161 IMAGES

Quick Links

Biography Awards Photo Gallery

Filmography (by Job) Trailers and Videos

Explore More

Reboots & Remakes We Can't Wait to See

Get a sneak peek of the new version of this page. [Check it out now](#)

Nominated

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE | SHARE

The Prestige (2006)

PG-13 | 2h 10min | Drama, Mystery, Sci-Fi | 20 October 2006 (USA)

8.5/10 1,207,788

Rate This

Motivación

The image is a screenshot of the IMDb website, showing two overlapping views. The top view is the profile page for Christopher Nolan, and the bottom view is the movie page for 'The Prestige' (2006).

Christopher Nolan Profile:

- Name:** Christopher Nolan (1)
- Roles:** Writer | Producer | Director
- Top 500:** Indicated by a red flag icon.
- Quick Links:** Biography, Awards, Photo Gallery, Filmography (by Job), Trailers and Videos.
- Explore More:** A button to explore more content.
- Reboots & Remakes We Can't Wait to See:** A section featuring movie posters for 'Inception', 'The Dark Knight', and 'The Matrix'.
- Video:** A video player showing an interview with Christopher Nolan, with a play button icon and a duration of 1:12.
- Statistics:** 19 VIDEOS | 161 IMAGES.
- Description:** Best known for his cerebral, often nonlinear, storytelling, acclaimed writer-director Christopher Nolan was born on July 30, 1970, in London, England. Over the course of 15 years of filmmaking, Nolan has directed some of the biggest blockbusters of the decade. See full bio »
- Born:** July 30, 1970
- More at IMDb:** Contact Info, Filmography, etc.

The Prestige Movie Page:

- Header:** FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE | SHARE
- Title:** The Prestige (2006)
- Rating:** 8.5/10 (1,207,788 votes)
- Details:** PG-13 | 2h 10min | Drama, Mystery, Sci-Fi | 20 October 2006 (USA)

Datos

Directores

<u>nombre</u>	<u>país</u>	<u>retirado</u>
Christopher Nolan	Inglaterra	false
Clint Eastwood	EE.UU	false
Ingmar Bergman	Suecia	true
John Favreau	EE.UU	false

Películas

<u>nombre</u>	<u>director</u>	<u>año</u>
The Prestige	Christopher Nolan	2006
Interstellar	Christopher Nolan	2014
Gran Torino	Clint Eastwood	2008
The Prestige	John Favreau	2021

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	<u>eval</u>
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Consulta Frecuente

Score promedio de la película "The Prestige" de "Christopher Nolan"

```
SELECT AVG(eval) AS promedio  
FROM Evaluaciones  
WHERE nombre = 'The Prestige' AND  
       director = 'Christopher Nolan'
```

promedio
8.5

Datos

Directores

<u>nombre</u>	<u>país</u>	<u>retirado</u>
Christopher Nolan	Inglaterra	false
Clint Eastwood	EE.UU	false
Ingmar Bergman	Suecia	true
John Favreau	EE.UU	false

Películas

<u>nombre</u>	<u>director</u>	<u>año</u>
The Prestige	Christopher Nolan	2006
Interstellar	Christopher Nolan	2014
Gran Torino	Clint Eastwood	2008
The Prestige	John Favreau	2021

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	<u>eval</u>
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Datos

Directores

<u>nombre</u>	<u>país</u>	<u>retirado</u>
Christopher Nolan	Inglaterra	false
Clint Eastwood	EE.UU	false
Ingmar Bergman	Suecia	true
John Favreau	EE.UU	false

Películas

<u>nombre</u>	<u>director</u>	<u>año</u>	<u>promedio</u>
The Prestige	Christopher Nolan	2006	8.5
Interstellar	Christopher Nolan	2014	8.2
Gran Torino	Clint Eastwood	2008	8.4
The Prestige	John Favreau	2021	9.8

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	<u>eval</u>
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Datos

Directores

<u>nombre</u>	<u>país</u>	<u>retirado</u>
Christopher Nolan	Inglaterra	false
Clint Eastwood	EE.UU	false
Ingmar Bergman	Suecia	true
John Favreau	EE.UU	false

Películas

<u>nombre</u>	<u>director</u>	<u>año</u>	<u>promedio</u>
The Prestige	Christopher Nolan	2006	8.5
Interstellar	Christopher Nolan	2014	8.2
Gran Torino	Clint Eastwood	2008	8.4
The Prestige	John Favreau	2021	9.8

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	<u>eval</u>
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Datos

Directores

<u>nombre</u>	<u>país</u>	<u>retirado</u>
Christopher Nolan	Inglaterra	false
Clint Eastwood	EE.UU	false
Ingmar Bergman	Suecia	true
John Favreau	EE.UU	false

Películas

<u>nombre</u>	<u>director</u>	<u>año</u>	<u>promedio</u>
The Prestige	Christopher Nolan	2006	8.6
Interstellar	Christopher Nolan	2014	8.2
Gran Torino	Clint Eastwood	2008	8.4
The Prestige	John Favreau	2021	9.8

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	<u>eval</u>
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Vistas

CREATE VIEW PelEval AS

SELECT pelicula, director,

AVG(eval) AS promedio

FROM Evaluaciones

GROUP BY pelicula, director

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

Vistas

La vista define una tabla "virtual"

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

SELECT promedio

FROM PelEval

WHERE película = 'The Prestige' AND
director = 'Christopher

Nolan'



promedio
8.5

¿Cómo funcionan las vistas?

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

```
CREATE VIEW PelEval AS
SELECT película, director,
       AVG(eval) AS promedio
FROM Evaluaciones
GROUP BY película, director

SELECT promedio
FROM PelEval
WHERE película = 'The Prestige' AND
       director = 'Christopher Nolan'
```

=

```
SELECT promedio
FROM (
  SELECT película, director,
         AVG(eval) AS promedio
  FROM Evaluaciones
  GROUP BY película, director )
WHERE película = 'The Prestige' AND
       director = 'Christopher Nolan'
```

¿Cómo funcionan las vistas?

Las vistas no son tablas físicas!

Cuando consultamos una vista:

1. Se reescribe la consulta reemplazando la vista por la consulta original en el **FROM**.
2. Se ejecuta la consulta sobre las **tablas originales**.

¿Cómo funcionan las vistas?

En la práctica solo estamos guardando una consulta frecuente en el sistema para reutilizarla después:

- No estamos guardando una tabla
- No se crea una tabla en el disco
- Se trabaja con las tablas base
- Si se actualizan los datos no hay problemas

Vistas y actualización

Evaluaciones

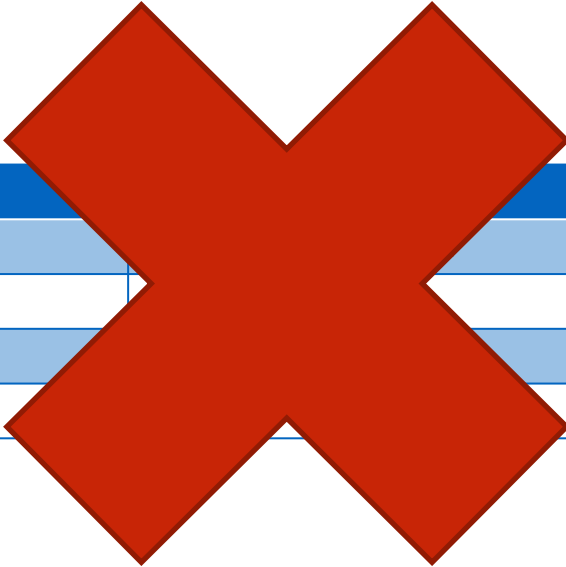
<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.6
The Prestige	John Favreau	9.8

Eliminando Vistas

PelEval



película	promedio
Interstellar	8.4
Gran Torino	8.2
The Prestige	8.6
The Prestige	9.8

DROP VIEW PelEval

SELECT *
FROM PelEval

Error: No such table PelEval

¿Para qué sirven las vistas?

Abstracción:

- Reducir la complejidad de consultas grandes.
- Evitar repetición de consultas frecuentes.

Mantenibilidad:

- Más fácil de manejar que la gestión de datos duplicados o redundantes.
- Más fácil de optimizar y mantener que una consulta repetida.
- Es más lento que tener las tablas de verdad!

Vistas materializadas

```
CREATE MATERIALIZED VIEW PelEval AS
SELECT pelicula, director,
       AVG(eval) AS promedio
FROM Evaluaciones
GROUP BY pelicula, director
```

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

Vistas materializadas

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

```
SELECT promedio  
FROM PelEval  
WHERE pelicula = 'The Prestige' AND  
director = 'Christopher Nolan'
```



promedio
8.5

Vistas materializadas

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	Uncut	9.8

Vistas materializadas

Evaluaciones

<u>película</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favreau	The Guardian	8.8

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

Vistas materializadas

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favre		

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

```
SELECT promedio
FROM PelEval
WHERE pelicula = 'The Prestige' AND
      director = 'Christopher Nolan'
```



promedio
8.5

Vistas materializadas

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favre		

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.5
The Prestige	John Favreau	9.8

REFRESH MATERIALIZED VIEW PelEval

Vistas materializadas

Evaluaciones

<u>pelicula</u>	<u>director</u>	<u>fuelle</u>	eval
Gran Torino	Clint Eastwood	The Guardian	8.1
Gran Torino	Clint Eastwood	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Observer	8.3
The Prestige	Christopher Nolan	Uncut	8.5
The Prestige	Christopher Nolan	Rolling Stone	8.7
The Prestige	Christopher Nolan	The Guardian	8.7
Interstellar	Christopher Nolan	Rolling Stone	8.2
The Prestige	John Favre		

PelEval

película	director	promedio
Interstellar	Christopher Nolan	8.4
Gran Torino	Clint Eastwood	8.2
The Prestige	Christopher Nolan	8.6
The Prestige	John Favreau	9.8

```
SELECT promedio
FROM PelEval
WHERE pelicula = 'The Prestige' AND
      director = 'Christopher Nolan'
```



promedio
8.6

Vistas materializadas vs tablas

```
CREATE MATERIALIZED VIEW PelEval AS
```

```
    SELECT pelicula, director,  
           AVG(eval) AS promedio  
    FROM Evaluaciones  
    GROUP BY pelicula, director
```

```
REFRESH MATERIALIZED VIEW  
PelEval
```

```
CREATE TABLE PelEval AS
```

```
    SELECT pelicula, director,  
           AVG(eval) AS promedio  
    FROM Evaluaciones  
    GROUP BY pelicula, director
```

- Las vistas materializadas generan la misma abstracción que las normales pero son más rápidas.
- Por otro lado, requieren espacio adicional e introducen una redundancia al esquema que nos debemos preocupar de mantener consistente, por ejemplo ejecutando **REFRESH MATERIALIZED VIEW** según sea necesario.
- Podemos automatizar eso en la misma DB? Si, mediante **TRIGGERS**.

Triggers

Triggers

- Procedimientos en la base de datos que se *gatillan* cada vez que se ejecuta algún evento.
- Contribuyen a forzar ciertas restricciones más complejas y a mantener la consistencia de la BD.
- Por ejemplo: Queremos disminuir el stock de un producto cada vez que se le crea una venta .

Triggers

Sintaxis (parte de)

- El evento puede ser {BEFORE | AFTER | INSTEAD OF} {CREATE| DELETE | UPDATE}. (ej: BEFORE CREATE)
- Podemos ejecutar el trigger para cada fila afectada por el evento (FOR EACH ROW), o 1 vez por evento (FOR EACH STATEMENT).

Triggers

Ejemplo

La sintaxis cambia mucho entre sistemas pero es más o menos así:

```
CREATE TRIGGER reducir_stock
```

```
AFTER CREATE ON Ventas
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE Productos
```

```
    SET stock = Productos.stock - 1
```

```
    WHERE NEW.id_productos = Productos.id
```

```
END
```

Triggers

Ejemplo

```
CREATE TRIGGER reducir_stock
```

```
AFTER CREATE ON Ventas
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE Productos
```

```
    SET stock = Productos.stock - 1
```

```
    WHERE NEW.id_productos = Productos.id
```

```
END
```

Ejecutar cada vez que
se cree una venta.

Triggers

Ejemplo

```
CREATE TRIGGER reducir_stock  
AFTER CREATE ON Ventas  
FOR EACH ROW  
BEGIN  
    UPDATE Productos  
    SET stock = Productos.stock - 1  
    WHERE NEW.id_productos = Productos.id  
END
```

Para cada fila creada

Triggers

Ejemplo

```
CREATE TRIGGER reducir_stock
```

```
AFTER CREATE ON Ventas
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE Productos
```

```
    SET stock = Productos.stock - 1
```

```
    WHERE NEW.id_productos = Productos.id
```

```
END
```

Actualizamos el stock en la tabla
Productos

Triggers

Ejemplo

```
CREATE TRIGGER reducir_stock
```

```
AFTER CREATE ON Ventas
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE Productos
```

```
    SET stock = Productos.stock - 1
```

```
    WHERE NEW.id_productos = Productos.id
```

```
END
```

En que el id del producto corresponda con la venta que se creó.

Triggers

Ejemplo

Para lo que comentábamos antes:

```
CREATE TRIGGER refresh_eval  
AFTER CREATE ON Evaluaciones  
FOR EACH STATEMENT  
BEGIN  
    REFRESH MATERIALIZED VIEW PelEval  
END
```

Stored Procedures

Stored Procedures

- Son funciones definidas mediante SQL, que quedan guardadas en el mismo DBMS y por lo tanto se pueden usar en consultas.
- Permiten ejecutar lógica compleja y repetitiva directamente en el DBMS.
- En un sólo proceso podemos ejecutar todas las consultas que queramos, o también hacer control de flujo con IFs o loops.

Stored Procedures

Sintaxis (parte de)

CREATE or REPLACE Function <nombre_funcion> (<argumentos>) RETURNS

<tipo_retorno> AS

\$\$

DECLARE

<declaracion de variables>

BEGIN

<sentencias SQL>

END

\$\$ language plpgsql

Stored Procedures

Ejemplo

Procedimiento para insertar una fila a tabla personas:

```
CREATE OR REPLACE FUNCTION insertar_persona (rut varchar, nombre varchar, apellido  
varchar) RETURNS void AS  
$$  
BEGIN  
INSERT INTO personas VALUES (rut,nombre,apellido);  
END  
$$ language plpgsql
```

Stored Procedures

Ejemplo

Ahora para usarla hacemos:

```
SELECT insertar_persona('11.111.111-1', 'pepito', 'los palotes')
```

Stored Procedures

Ejemplo

Podemos iterar sobre resultados de consultas y usar eso para procesar e insertar datos a otras tablas:


```
CREATE OR REPLACE FUNCTION transferencia_nombres() RETURNS void AS $$  
DECLARE  
    tupla RECORD;  
    concat varchar;  
BEGIN  
    FOR tupla IN SELECT * FROM Personas LOOP  
        concat = tupla.nombre || tupla.apellido;  
        insert into personascompleto values (tupla.rut, concat);  
    END LOOP;  
END  
$$ language plpgsql
```

Stored Procedures

Consultas dinámicas

Podemos retornar consultas completas y además usar los argumentos de la función para generarlas de forma dinámica.

```
CREATE OR REPLACE FUNCTION vuelos_desde (c_origen varchar)
RETURNS TABLE (ciudad_destino varchar(50), horas integer) AS $$
BEGIN
RETURN QUERY EXECUTE '
    SELECT ciudad_destino, horas
    FROM Vuelo
    WHERE ciudad_origen = $1'
    USING c_origen;
RETURN; END
$$ language plpgsql
```



Lógica en la BDD

Conclusiones

- Escribir lógica compleja en SQL se puede volver inmanejable muy rápido. Además, triggers, vistas y procedimientos ya guardados en la DB requieren de ejecutar un `CREATE` para modificarse.
- Por otro lado, los frameworks web modernos, de manera opinionada presentan escaso soporte para este tipo de funcionalidades. Estas dos cosas hacen que usar views, procedures y triggers no se vean mucho en aplicaciones modernas.
- De todas formas, estas cosas aún se ven en sistemas legados (antiguos) o en sistemas de *Data engineering*, como *data warehouses* o *data lakes* (hablaremos más de lo que es eso en el futuro) y en ciertos servicios en la nube.

Programando con SQL

Programación y SQL

- Hasta ahora hemos visto a la Base de Datos como un componente aislado.
- Pero una Base de datos no tiene sentido si no podemos conectarla a una aplicación.
- En esta clase veremos dos formas de programar conectado a una base de datos.

SQL en Python

Conexión a la Base de Datos

- La forma más simple de usar una base de datos desde un programa es usando una librería especializada para el motor que estemos usando.
- Por ejemplo en Python existen `sqlite3` y `psycopg2` (para postgres).

Conexión a la Base de Datos

Python

Por ejemplo, para sqlite importamos la librería correspondiente e iniciamos una conexión a la db:

```
import sqlite3  
db_connection = sqlite3.connect('some_db.db')  
# hacer cosas con la db  
db_connection.close() # cerramos la conexion
```

La conexión significa que la base de datos está esperando instrucciones por parte de nuestro programa.

Cursores

Python

Para ejecutar comandos SQL desde Python, nos falta instanciar un **cursor**.

Un objeto de la clase cursor nos permite ejecutar un comando SQL en Python y representa un “puntero” al output del comando.

Modificando la DB

Python

```
cur = db_connection.cursor() # instanciamos el cursor
create_query = "CREATE TABLE Peliculas(id INT, titulo
VARCHAR(100));"
insert_query = "INSERT INTO Peliculas VALUES (1, 'Minions');"
try:
    cur.execute(create_query)
    cur.execute(insert_query)
except sqlite3.OperationalError as e:
    print(e)

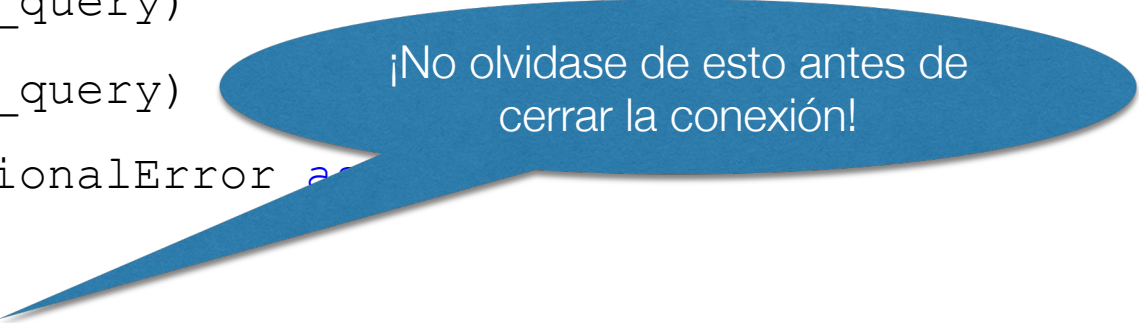
db_connection.commit() # guardamos los cambios en disco.
db_connection.close() # cerramos la conexion
```

Modificando la DB

Python

```
cur = db_connection.cursor() # instanciamos el cursor
create_query = "CREATE TABLE Peliculas(id INT, titulo
VARCHAR(100));"
insert_query = "INSERT INTO Peliculas VALUES (1, 'Minions');"
try:
    cur.execute(create_query)
    cur.execute(insert_query)
except sqlite3.OperationalError as e:
    print(e)

db_connection.commit() # guardamos los cambios en disco.
db_connection.close() # cerramos la conexion
```



¡No olvidase de esto antes de cerrar la conexión!

Consultas básicas

Python

```
select_query = "SELECT * FROM Peliculas"  
try:  
    cur.execute(select_query)  
except sqlite3.OperationalError as e:  
    print(e)
```

... y cómo accedemos al resultado? 🤔

Cursores

El **Cursor** representa un puntero que va recorriendo los resultados. Podemos usar el método `fetchone()` para traer uno o `fetchall()` para traerlos todos (Ojo con la memoria! 🤯).

Adicionalmente en Python el Cursor está implementado como un iterable.

Cursores

Obteniendo resultados de consultas

```
select_query = "SELECT * FROM Peliculas"
```

```
cur.execute(select_query)
```

```
first_row = cur.fetchone()
```

```
second_row = cur.fetchone()
```

```
print(first_row)
```

```
print(second_row)
```

```
cur.execute(select_query)
```

```
rows = cur.fetchall()
```

```
print(rows)
```

```
cur.execute(select_query)
```

```
for row in cur:
```


```
    print(row)
```

Ahora lo importante

- Usuarios de mi aplicación ingresan datos
- Basado en el input se hace una consulta a mi base de datos y retorno algo.

Un ejemplo inocente...

```
id = input("id de la película?")
select_query = "SELECT * FROM Peliculas WHERE id="+id
cur.execute(select_query)
print(cur.fetchone())
print("Exitó!")
```

 <https://www.psycopg.org/docs/usage.html>

Psycopg can automatically convert Python objects to and from SQL literals: using this feature your code will be more robust and reliable. We must stress this point:

Warning: Never, **never**, **NEVER** use Python string concatenation (+) or string parameters interpolation (%) to pass variables to a SQL query string. Not even at gunpoint.

The correct way to pass variables in a SQL command is using the second argument of the `execute()` method:

```
>>> SQL = "INSERT INTO authors (name) VALUES (%s);" # Note: no quotes
>>> data = ("O'Reilly", )
>>> cur.execute(SQL, data) # Note: no % operator
```

Pero profesor,
¿qué tiene de malo concatenar
strings para generar consultas?

Un ejemplo inocente...

```
id = input("id de la película?")
select_query = "SELECT * FROM Peliculas WHERE id="+id
cur.execute(select_query)
print(cur.fetchone())
print("Exitó!")
```

```
>> id de la película? 1; DROP TABLE Peliculas; --
```

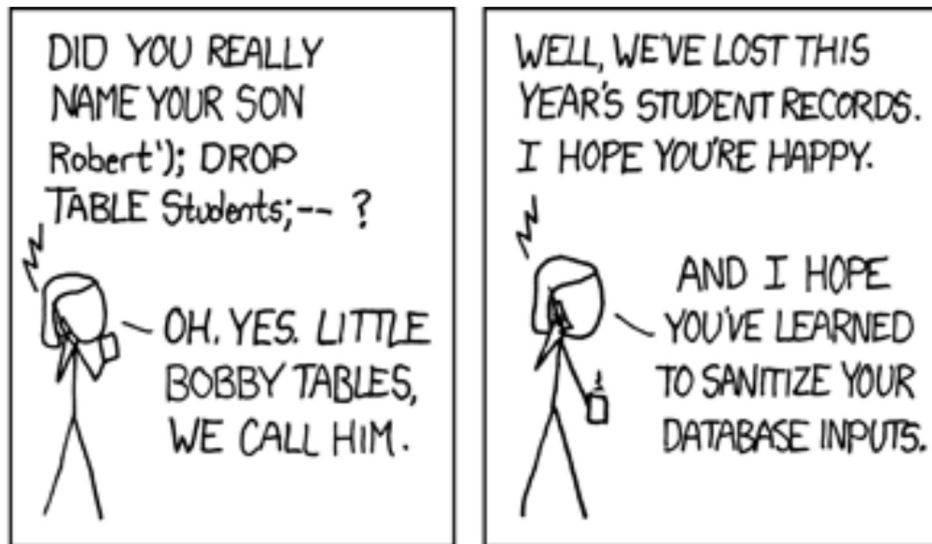
Un ejemplo inocente...

```
id = input("id de la película?")
select_query = "SELECT * FROM Peliculas WHERE id="+id
cur.execute(select_query)
print(cur.fetchone())
print("Exitó!")
```

```
>> id de la película? 1; DROP TABLE Peliculas; --
```

Por qué no concatenar

```
query = "SELECT * FROM Students WHERE name = " + nombre  
  
cursor.execute(query)
```



Nos exponemos a una forma de ataque: [SQL Injection](#)

Por qué no concatenar

Good luck speed cameras.



SQL Parametrizado

Python

Si van a usar parámetros para una consulta provenientes del input de usuario, hay que hacerlo así:

```
query = "SELECT * FROM Students \
        WHERE edad =%(edad)d  AND nombre = %(nombre)s"

cursor.execute(query, {"nombre": "Rivera", "edad": 11 })
```

A esto le llamamos “sanitizar” el input, dejamos que la librería parsee y asegure los parámetros antes de pasarlos a la DB.

ORMs

ORMs

- Del inglés *Object Relational Mapping*. Es un patrón de diseño altamente adoptado en desarrollo web, a través de librerías que vienen incluidas en los *frameworks*.
- Fundamentalmente consiste en tener para cada tabla de la DB una Clase (OOP) en el código de la aplicación.
- Nacen de la necesidad de sacar provecho a las ventajas de la programación orientada a objetos, manteniendo la estructura del esquema relacional que por debajo guarda los datos.
- Proporcionan una capa de abstracción sobre la base de datos.

ORMs

Ejemplo

```
SELECT sname  
FROM Reserves NATURAL JOIN Sailors  
WHERE bid = 100 AND rating > 5
```

Compilación

Colección de datos

Colección de datos

sailors

.join(reserves).on((s,r) => s.sid===r.sid)

JOIN

.filter((s,r) => r.bid ===100 && r.rating >5)

WHERE

.map((s,r) => s.sname)

SELECT

ORMs

Ejemplos

- [ActiveRecord](#) del framework de Ruby, Ruby on Rails.
- El Framework web de Python: Django, también tiene el [suyo](#).
- En Javascript (Node) se suele usar [Sequelize](#).
- Hay disponibles para la mayoría de lenguajes / frameworks.

Pero profesor,
¿por qué aprendimos SQL si en la
práctica se usa un ORM?

- Los ORMs agregan una capa por encima de la DB. Por debajo se sigue escribiendo SQL y eso es lo que finalmente se ejecuta.
- El uso de la base de datos es un componente clave de cada aplicación, si no entendemos las consultas que el ORM está ejecutando en verdad estamos ciegos a lo que estamos programando.
- Las consultas simples hacen parecer que el ORM te facilita la vida. Pero apenas se pone un poco complejo dan ganas de volver a SQL.
- La verdad es que estamos agregando una capa más de complejidad al sistema!!

Conclusión: Usar un ORM aporta mucho valor, pero no podemos olvidarnos de SQL.