



imatia
innovation

We help you to do more



- Conceptos básicos
- Tablas, columnas y claves primarias
- Relación entre tablas, claves foráneas
- Modelo Entidad-Relación
- Cardinalidad
- SQL



- Las BD son colecciones de registros interrelacionados mediante un DBMS (*Sistema Gestor de Bases de Datos*)
- Las bases de datos almacenan toda la información que quedará guardada, para que el programa pueda recogerla y trabajar con ella en el futuro
- El hecho de guardar la información de un programa en una base de datos se llama “persistencia da información”
- El lenguaje empleado para comunicarse con la BD se llama SQL

- Ventajas
 - Datos integrados
 - Poca duplicidad de datos
 - Independencia programa/datos
 - Fácil representación
- Desventajas
 - Sin estándares
 - Diferentes lenguajes según el sistema gestor

- Una base de datos consta de una serie de tablas y vistas.
- Las tablas contienen registros de información relacionada entre las diferentes columnas de una misma tabla
- Estos datos se pueden consultar, actualizar, introducir nuevos datos o eliminar datos ya existentes
- Las vistas tienen también estructura de tabla, pero solo se pueden consultar datos.

	123 DIRECTORID 🔍 ⬆️⬆️	ABC NAME 🔍 ⬆️⬆️	ABC BIRTH_YEAR 🔍 ⬆️⬆️
1	1	Frank Darabont	1959
2	2	Francis Ford Coppola	1939
3	3	Christopher Nolan	1970
4	4	Sidney Lumet	1924
5	5	Steven Spielberg	1946
6	6	Peter Jackson	1961
7	7	Quentin Tarantino	1963
8	8	Sergio Leone	1929
9	9	David Fincher	1962

	123 GENREID 🔍 ⬆️⬆️	ABC NAME 🔍 ⬆️⬆️
1	1	Drama
2	2	Crime
3	3	Action
4	4	Biography
5	5	Adventure
6	6	Western

- Ejemplo de tabla:

	123 MOVIEID	ABC NAME	123 YEAR	123 DURATION	123 GENREID	123 DIRECTORID
1	1	The Shawshank Redemption	1,994	142	1	1
2	2	The Godfather	1,972	175	2	2
3	3	The Godfather: Part II	1,974	202	2	2
4	4	The Dark Knight	2,008	152	3	3
5	5	12 Angry Men	1,957	96	1	4
6	6	Schindler's List	1,993	195	4	5
7	7	The Lord of the Rings: The Return of the King	2,003	201	5	6
8	8	Pulp Fiction	1,994	154	2	7
9	9	Il buono, il brutto, il cattivo	1,966	182	6	8
10	10	Fight Club	1,999	139	1	9

- Esta tabla contiene información sobre las 10 mejores películas (según IMDB) a comienzos del 2019
- En color rojo está señalado un registro de la BD (la información de una película)
- En color verde está señalada una columna de la DB (la misma información de todos los registros)
- En color azul está señalada la columna de la clave primaria (valor único para cada registro en esa misma tabla)

- La columna de color magenta indica las claves foráneas. Una clave foránea implica que, en dicha tabla, la columna tendrá el valor de la clave primaria de la otra tabla.

- Un ejemplo:

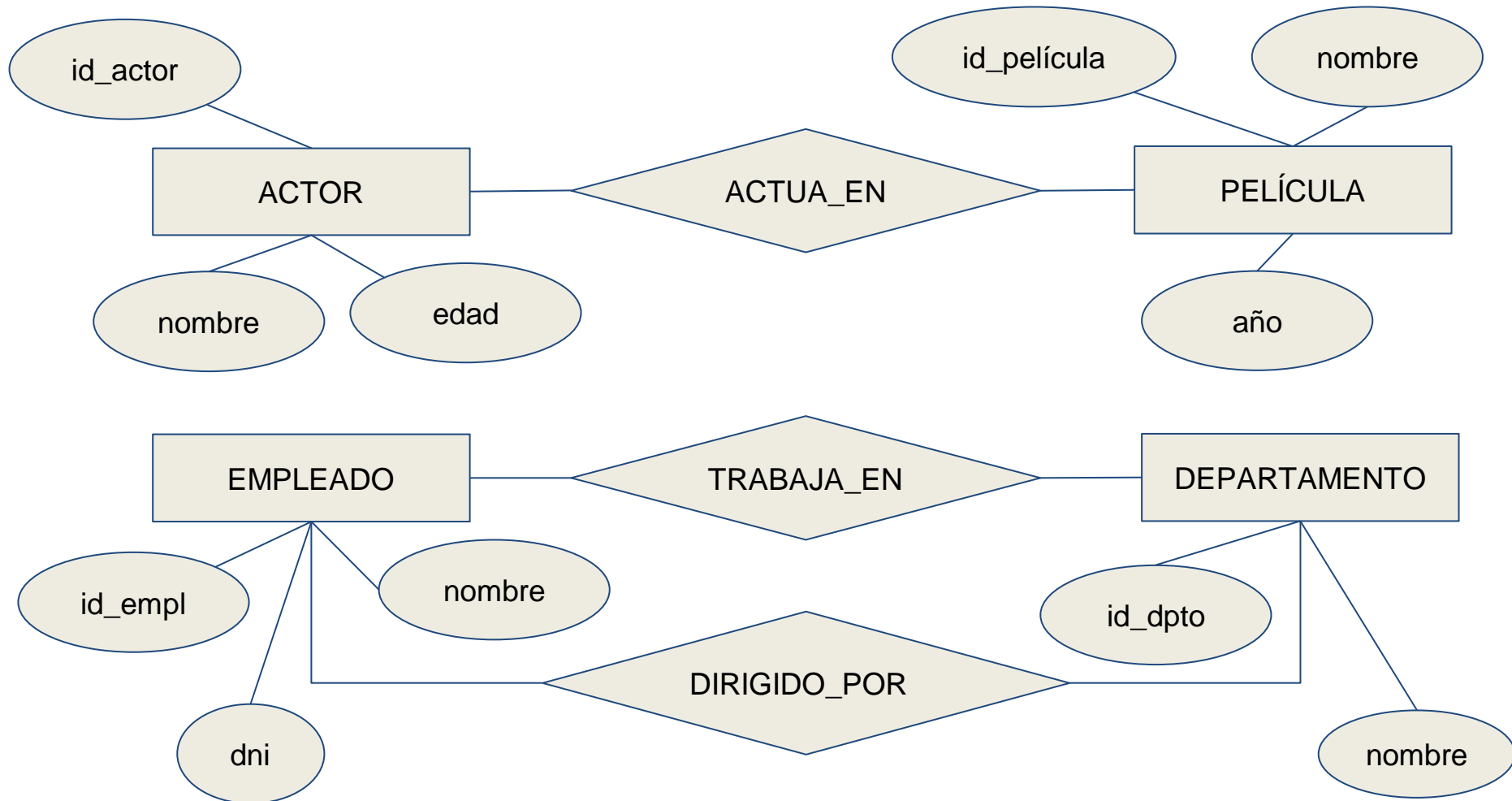
	123 MOVIEID	ABC NAME	123 YEAR	123 DURATION	123 GENREID	123 DIRECTORID
1	1	The Shawshank Redemption	1,994	142	1	1
2	2	The Godfather	1,972	175	2	2
3	3	The Godfather: Part II	1,974	202	2	2
4	4	The Dark Knight	2,008	152	3	3
5	5	12 Angry Men	1,957	96	1	4
6	6	Schindler's List	1,993	195	4	5
7	7	The Lord of the Rings: The Return of the King	2,003	201	5	6
8	8	Pulp Fiction	1,994	154	2	7
9	9	Il buono, il brutto, il cattivo	1,966	182	6	8
10	10	Fight Club	1,999	139	1	9

- Nota: Ambas columnas de las distintas tablas no tienen que tener el mismo nombre.

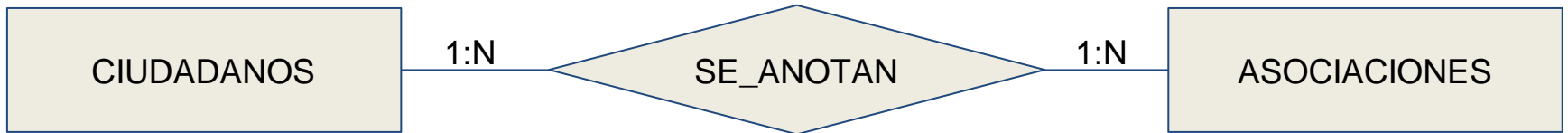
	123 GENREID	ABC NAME
1	1	Drama
2	2	Crime
3	3	Action
4	4	Biography
5	5	Adventure
6	6	Western

- Este modelo sirve exclusivamente para diseñar los esquemas de la base de datos que luego tendremos que implementar en el DBMS
- Una *entidad* representa un *objecto* o cosas, que son diferentes entre sí. Un ejemplo puede ser un **Vehículo**, un **Empleado** o un **Cliente**
- Los *atributos* definen las características de la entidad. Según el ejemplo de la entidad **Vehículo** del ejemplo anterior, los atributos podrían ser: *matrícula, color, marca, modelo*
- La entidad se representa en el interior de un rectángulo.
- Los atributos de esa entidad se representan en el interior de elipses
- La clave primaria, se indica subrayando el atributo que hará de clave primaria
- Para indicar una relación entre dos entidades, se indica con un rombo entre ellas para indicar su relación

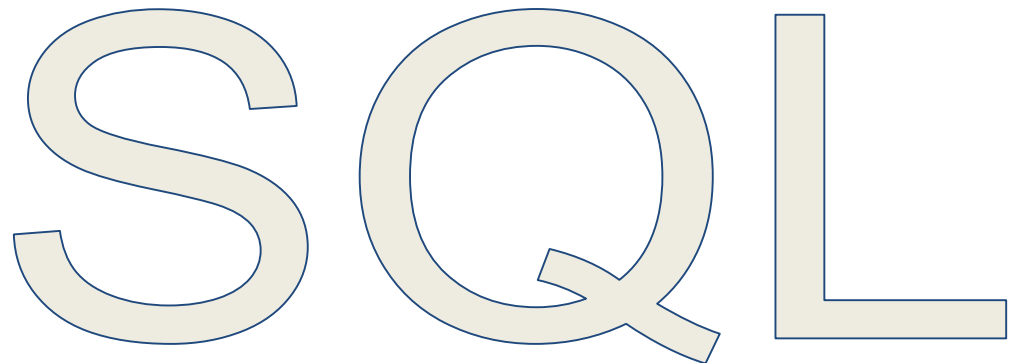
- Ejemplo de Entidad-Relación:



- Cardinalidad
- La cardinalidad indica la relación entre dos entidades. Los registros están enlazados entre ellos.
- Ejemplos:
 - Un empleado es jefe de un departamento
 - Una película está dirigida por una persona
 - Los géneros literarios de una serie de novelas
 - Los ciudadanos pueden pertenecer a varias asociaciones
- Las cardinalidades pueden ser de 3 tipos diferentes:
 - Una a una (**1:1**)
 - Una a varios (**1:N**)
 - Varios a varios (**N:M**)
- Con estas cardinalidades, los ejemplos anteriores serían:
 - Un empleado es jefe de un departamento (1:1)
 - Una película está dirigida por una persona (1:N)
 - Los géneros literarios de una serie de novelas (1:N)
 - Los ciudadanos pueden pertenecer a varias asociaciones (N:M)



- SQL es un lenguaje de consultas estructuradas(**Structured Query Language**)
- Es un lenguaje que permite administrar y recuperar información desde los DBMS
- Mediante SQL se pueden crear nuevas tablas, modificar las existentes, añadir, consultar, actualizar y eliminar datos.
- SQL puede trabajar con diferentes datos, como números, cadenas, fechas, o incluso elementos de tipo binario (según el DBMS que se emplee).
- Para los ejemplos de SQL, usaremos el DBMS **SQLite**

The word 'SQL' is rendered in a large, light beige, sans-serif font. Each letter has a thin blue outline, giving it a 3D or embossed appearance.

- Creación de una tabla
- Para la creación de una tabla, utilizamos el comando CREATE TABLE, y a continuación, el nombre de la tabla.
- Entre paréntesis, añadimos las columnas de la tabla separadas por comas, indicando el tipo de columna y su nombre. Podemos añadir las restricciones de clave primaria desde la misma columna, el resto de las restricciones se tendrán que añadir al final de la declaración de las columnas

```
CREATE TABLE MOVIES (  
    MOVIE_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    MOVIE_NAME TEXT NOT NULL,  
    MOVIE_YEAR INTEGER,  
    MOVIE_DURATION INTEGER,  
    GENRE_ID INTEGER,  
    DIRECTOR_ID INTEGER  
);
```

- Creación de una tabla
- Para añadir más restricciones a una tabla ya creada, como una clave foránea o una restricción única (como la clave primaria, el valor no puede repetirse en la misma columna para cualquier registro de la tabla) o mismamente una clave principal, se puede añadir mediante el comando ALTER TABLE, indicando la tabla que queremos modificar y la columna a la que queramos añadir una restricción.

```
ALTER TABLE MOVIES  
ADD CONSTRAINT FK_MOVIES_GENRES  
FOREIGN KEY (MOVIES.GENRE_ID)  
REFERENCES GENRES(GENRES.GENRE_ID);
```


- Sin embargo, algunos DBMS no soportan este comando (Como SQLite), y se tienen que añadir las claves foráneas en el momento de la creación de la tabla

- Ejemplo de creación en SQLite

```
CREATE TABLE MOVIES (  
    MOVIE_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    MOVIE_NAME TEXT NOT NULL,  
    MOVIE_YEAR INTEGER,  
    MOVIE_DURATION INTEGER,  
    GENRE_ID INTEGER,  
    DIRECTOR_ID INTEGER,  
    FOREIGN KEY (GENRE_ID) REFERENCES GENRES(GENRE_ID),  
    FOREIGN KEY (DIRECTOR_ID) REFERENCES DIRECTORS(DIRECTOR_ID)  
);
```

- Instrucción SELECT
- La instrucción SELECT permite recoger la información guardada en la BD
- Se escribe la palabra SELECT, seguida de los nombres de las columnas que queramos obtener. Si escribimos un asterisco en vez de escribir las columnas, obtendremos todas las columnas de la tabla. A continuación, la cláusula FROM indicará la tabla de la cual queremos recoger la información

```
SELECT GENRE_DESCRIPTION FROM GENRES;
```

	ABC GENRE_DESCRIPTION 
1	Drama
2	Crime
3	Action
4	Biography
5	Adventure
6	Western

- Instrucción WHERE
- La palabra WHERE se utiliza para filtrar una consulta. Si el campo a filtrar es una cadena de texto, se utilizan las comillas y los números sin ellas
- El operador igual (=) comparará el valor del campo con el valor que se le asigna en la consulta

```
SELECT * FROM DIRECTORS WHERE DIRECTOR_ID=5;
```

123 DIRECTOR_ID	ABC DIRECTOR_NAME	123 DIRECTOR_BIRTH_YEAR
5	Steven Spielberg	1.946

- O operador de distinto (<>) funciona igual que el anterior, pero muestra los que no sean iguales

```
SELECT * FROM GENRES WHERE GENRE_DESCRIPTION <> 5;
```


- De la misma manera que existen los operadores igual y distinto, existen los operadores mayor que, menor que, mayor o igual que, menor o igual que. En este caso, solo se aceptan valores numéricos

```
SELECT * FROM DIRECTORS WHERE DIRECTOR_BIRTH_YEAR <= 1950
```

123 DIRECTOR_ID 🔍	ABC DIRECTOR_NAME 🔍	123 DIRECTOR_BIRTH_YEAR 🔍
2	Francis Ford Coppola	1,939
4	Sidney Lumet	1,924
5	Steven Spielberg	1,946
8	Sergio Leone	1,929

- El operador LIKE y NOT LIKE permite buscar dentro del contenido de la cadena de un campo
- Si se emplea el carácter ‘%’ como comodín, este puede ser substituído como si existiese cualquier número de todos los caracteres a partir de su posición. Por ejemplo, de escribir ‘%erg’ indicaría que fuese cualquier cadena acabada en las letras ‘erg’.

```
SELECT MOVIE_NAME FROM MOVIES WHERE MOVIE_NAME LIKE '%the%';
```

ABC MOVIE_NAME
The Shawshank Redemption
The Godfather
The Godfather: Part II
The Dark Knight
The Lord Of The Ring: The Return Of The King

- El funcionamiento del operador NOT LIKE es justo el contrario.

```
SELECT MOVIE_NAME FROM MOVIES WHERE MOVIE_NAME NOT LIKE '%the%';
```

- En el SELECT se pueden utilizar una serie de funciones predefinidas, que permiten obtener otros datos a mayores de los que ya contienen las propias tablas, como valores medios, sumas, conteos.. etc..
- Cuando se usan funciones de este tipo, si mostrase más de un campo, debe utilizarse la cláusula GROUP BY. (En el GROUP BY se escribirán todos los campos que no sean los de las funciones)
- A función COUNT() es un contador:
 - COUNT(*) cuenta las filas totales
 - COUNT(CAMPO) cuenta las veces que este campo aparece
 - COUNT (DISTINCT CAMPO) cuenta las veces que aparece el valor sin repetirse

SELECT COUNT(*) FROM AWARDS

123 COUNT(*)	10
--------------	----

SELECT COUNT(GENRE_ID) FROM MOVIES


123 COUNT(GENRE_ID)	10
---------------------	----

SELECT COUNT(DISTINCT GENRE_ID) FROM MOVIES

123 COUNT(DISTINCT GENRE_ID)	6
------------------------------	---








- Las funciones SUM() / AVG() / MIN() / MAX() se utilizan del mismo modo, escribiendo entre paréntesis el campo a utilizar. La condición es que solo pueden usarse sobre campos numéricos

```
SELECT AVG(AWARD_NOMINATIONS) FROM AWARDS
```

123 AVG(AWARD_NOMINATIONS) 
53.4




- La cláusula GROUP BY se utiliza para agrupar las filas de manera que se pueden utilizar las funciones de agregación con sum(), min(), max()...

```
SELECT DIRECTOR_ID, COUNT(MOVIE_ID)
FROM MOVIES GROUP BY DIRECTOR_ID;
```

123 DIRECTOR_ID 	123 COUNT(MOVIE_ID) 
1 	1
2 	2
3 	1
4 	1
5 	1









- La cláusula HAVING se usa para poder utilizar las funciones como filtro de la tabla

```
SELECT DIRECTOR_ID, COUNT(MOVIE_ID)
FROM MOVIES GROUP BY DIRECTOR_ID
HAVING COUNT(MOVIE_ID)>1;
```

123 DIRECTOR_ID 	123 COUNT(MOVIE_ID) 
2 	2

- La cláusula ORDER BY se usa para ordenar los registros que devuelve una tabla según unos campos de manera ascendente o descendente. El ORDER BY va después del WHERE

```
SELECT * FROM AWARDS WHERE AWARD_WIN > 50 ORDER BY AWARD_WIN DESC
```

123 AWARD_ID 	123 MOVIE_ID 	123 AWARD_NOMINATIONS 	123 AWARD_WIN 
6	7 	122	198
4	4 	155	154
5	6 	49	89
7	8 	69	64

- Los alias son nombres virtuales que pueden aplicarse tanto a una tabla como a una columna.
- Al aplicarse a las tablas, se usa para acortar el nombre y que sea más fácil su manejo. Una vez establecido se deberá usar siempre
- Al aplicarse a las columnas, modifica el nombre con el que se devolverá la columna

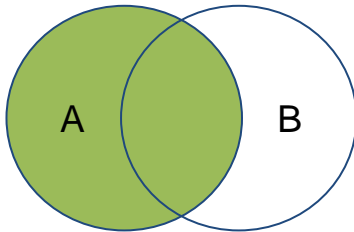
```
SELECT M.MOVIE_NAME FROM MOVIES M WHERE M.MOVIE_ID = 7
```

ABC MOVIE_NAME	↑↓
The Lord Of The Ring: The Return Of The King	

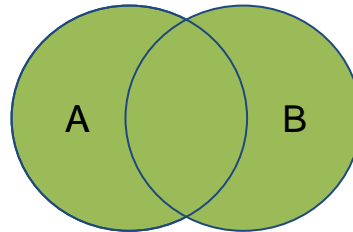
```
SELECT M.MOVIE_NAME AS "Título" FROM MOVIES M WHERE M.MOVIE_ID = 7
```

ABC Título	↑↓
The Lord Of The Ring: The Return Of The King	

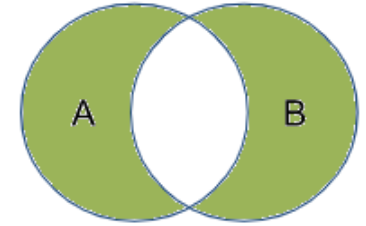
- La unión de tablas (o JOINS) son todos os datos que coinciden según los siguientes diagramas



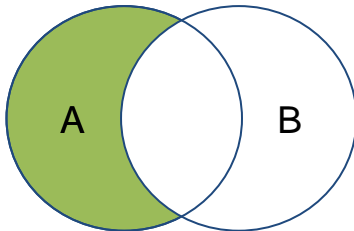
SELECT * FROM A LEFT
OUTER JOIN B ON A.name =
B.name



SELECT * FROM A
FULL OUTER JOIN
B ON A.name =
B.name



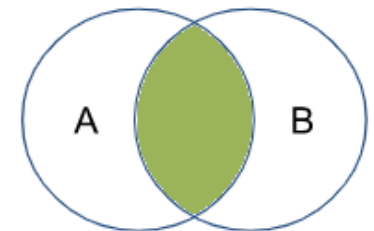
SELECT * FROM A FULL
OUTER JOIN B ON A.name =
B.name WHERE A.id IS null OR
B.id IS null



SELECT * FROM A LEFT OUTER
JOIN B ON A.name = B.name
WHERE TableB.id IS null

Produto Cartesiano:
5 filas A x 5 filas B = 25
filas

SELECT * FROM A CROSS
JOIN B



SELECT * FROM A INNER
JOIN B ON A.name = B.name

- Una subconsulta sirve para cambiar el valor de un campo con resultado de otra consulta

SELECT

MOVIE_NAME

FROM

MOVIES

WHERE

MOVIE_ID = (

SELECT

MOVIE_ID

FROM

MOVIES

WHERE

MOVIE_YEAR = 2008);

ABC MOVIE_NAME	
The Dark Knight	

- Nesta ocasión úsase a subconsulta para poder recoller un valor de filtrado no where. Cando úsase un igual, a subconsulta só pode devolver un valor.

SELECT

COUNT (*)

FROM

MOVIES

WHERE

MOVIE_YEAR = 2008;

123 COUNT (*)	
1	

- Si la subconsulta devuelve una lista de valores, se podría usar el operador IN

```
SELECT
    MOVIE_NAME
FROM
    MOVIES
WHERE
    MOVIE_ID IN (
        SELECT
            MOVIE_ID
        FROM
            MOVIES
        WHERE
            MOVIE_YEAR > 1995);
```

ABC MOVIE_NAME



The Dark Knight

The Lord Of The Ring: The Return Of The King

Il buono, il brutto, il cattivo

Fight Club

```
SELECT
    MOVIE_ID
FROM
    MOVIES
WHERE
    MOVIE_YEAR > 1995
```

123 MOVIE_ID



4

7

9

10

- Una subconsulta puede no utilizarse solo en la cláusula WHERE, si no también en el FROM. De esta forma, se utiliza el resultado de la subconsulta como tabla.

```

SELECT
    MOVIE_NAME,
    MOVIE_DURATION
FROM
    (
        SELECT
            *
        FROM
            MOVIES
        WHERE
            MOVIE_DURATION >= 180)
WHERE
    MOVIE_YEAR > 2000;

```

ABC MOVIE_NAME	123 MOVIE_DURATION
The Lord Of The Ring: The Return Of The King	201

- Para insertar datos, usaremos la sentencia INSERT INTO

```
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(1, 1, 39, 19);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(2, 2, 28, 27);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(3, 3, 20, 17);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(4, 4, 155, 154);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(5, 6, 49, 89);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(6, 7, 122, 198);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(7, 8, 69, 64);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(8, 9, 3, 1);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(9, 10, 38, 10);
INSERT INTO AWARDS (AWARD_ID, MOVIE_ID, AWARD_NOMINATIONS, AWARD_WIN) VALUES(10, 5, 11, 16);
```

- Para actualizar, la sentencia UPDATE, indicando en el WHERE una condición. Se actualizarán todos los registros que cumplan la condición

```
UPDATE AWARDS SET MOVIE_ID=1, AWARD_NOMINATIONS=39, AWARD_WIN=19 WHERE AWARD_ID=1;
UPDATE AWARDS SET MOVIE_ID=2, AWARD_NOMINATIONS=28, AWARD_WIN=27 WHERE AWARD_ID=2;
```

- Para eliminar registros de una tabla, se emplea la sentencia DELETE FROM. Se empleará SEMPRE con la condición WHERE, se no, borrará todos los datos de la tabla

```
DELETE FROM AWARDS WHERE AWARD_ID=2;
```

- Para eliminar una tabla, se utiliza la sentencia DROP TABLE

```
DROP TABLE TEST;
```

- Para crear una vista a partir de una consulta compleja, podremos envolver a consulta coa sentencia CREATE VIEW

```
CREATE  
  VIEW VMOVIESDIRECTOR AS SELECT  
    MOVIES.*,  
    DIRECTORS.DIRECTOR_NAME,  
    DIRECTORS.DIRECTOR_BIRTH_YEAR  
FROM  
  MOVIES  
INNER JOIN DIRECTORS ON  
  MOVIES.DIRECTOR_ID = DIRECTORS.DIRECTOR_ID;
```

- Se puede usar una vista como tabla para otras consultas.

123 MOVIE_ID ↑	ABC MOVIE_NAME	123 MOVIE_YEAR ↑	123 MOVIE_LENGTH ↑	123 GENRE_ID ↑	123 GENRE_NAME ↑	ABC DIRECTOR_NAME	123 DIRECTOR_YEAR ↓
5	12 Angry Men	1,957	96	1	4	Sidney Lumet	1,924
9	Il buono, il brutto, il cattivo	1,996	182	6	8	Sergio Leone	1,929
3	The Godfather: Part II	1,974	202	2	2	Francis Ford Coppola	1,939
2	The Godfather	1,972	175	2	2	Francis Ford Coppola	1,939
6	Schindler's List	1,993	195	4	5	Steven Spielberg	1,946
1	The Shawshank Redemption	1,994	142	1	1	Frank Darabont	1,959
7	The Lord Of The Ring: The Return Of The King	2,003	201	5	6	Peter Jackson	1,961
10	Fight Club	1,999	139	1	9	David Fincher	1,962
8	Pulp Fiction	1,994	154	2	7	Quentin Tarantino	1,963
4	The Dark Knight	2,008	152	3	3	Christopher Nolan	1,970

SELECT

MOVIE_NAME,
DIRECTOR_NAME

FROM

VMOVIESDIRECTOR

WHERE

MOVIE_YEAR = 2003;

ABC MOVIE_NAME	ABC DIRECTOR_NAME
The Lord Of The Ring: The Return Of The King	Peter Jackson