**Clase 1 al 4 de SQL**

Select de SQL corresponde con la proyección del algebra relacional

Where de SQL corresponde con la selección del algebra relacional

La idea seria leer las consultas SQL empezando por el from siguiendo por el where y luego por select.

Para ver las tablas que hay:

**show tables;**

Si esto nos tira un error es xq no estaríamos en ninguna bdd.

**Para elegir la tabla sobre la que vamos a trabajar**:

**use nombre\_archivo\_sql**

Ejemplo:

**use clasesql;**

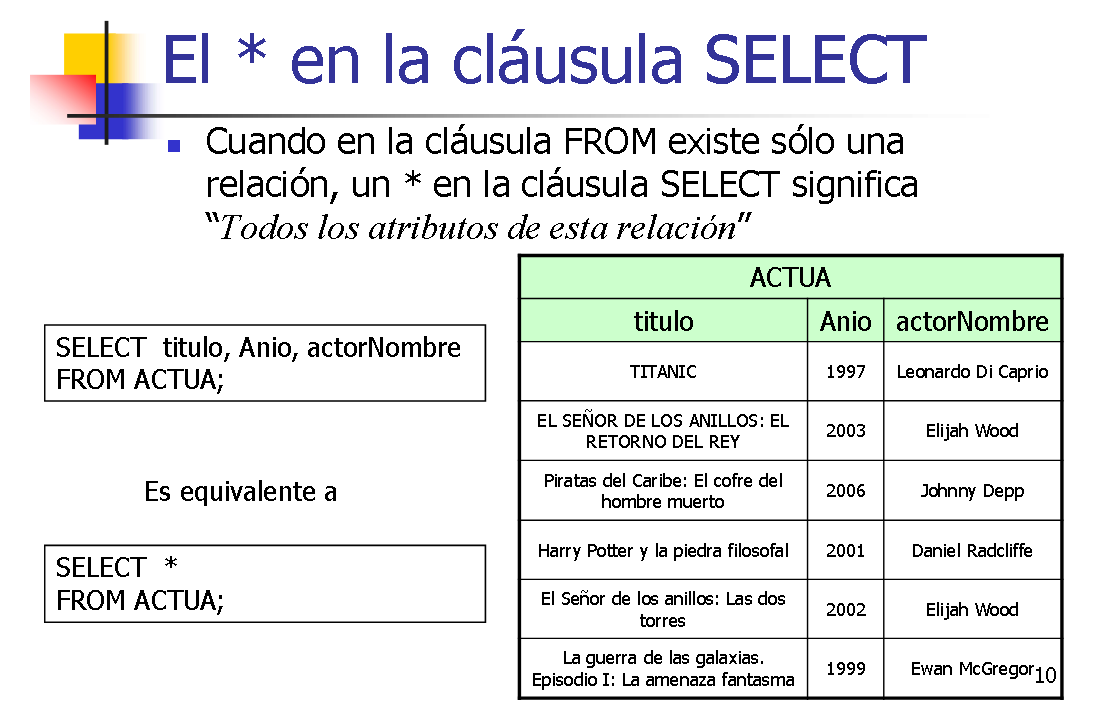
Para ver las bdd que tengamos:

**show databases;**

min 25:48

Para ver todos los atributos de una tabla podemos escribir de la siguiente forma:

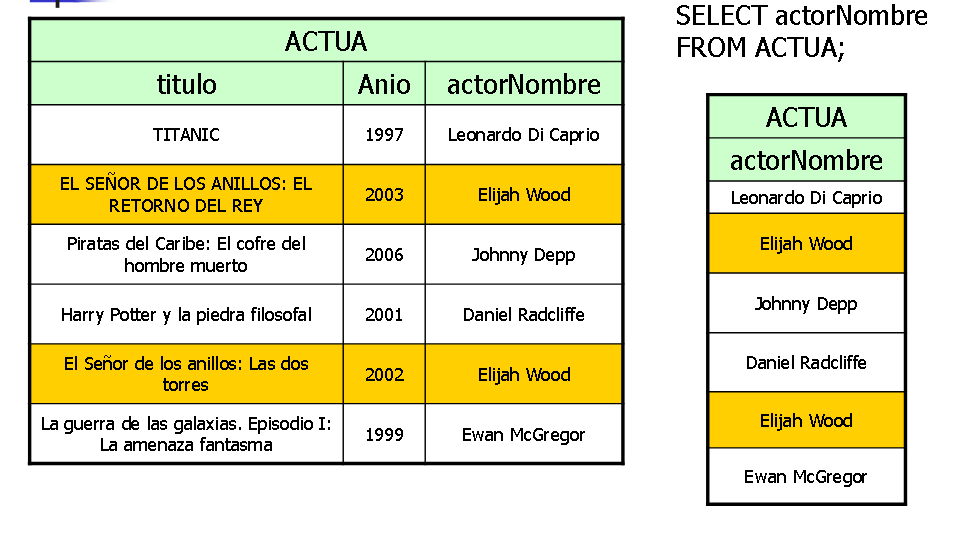
**SELECT \* FROM ACTUA;**

****

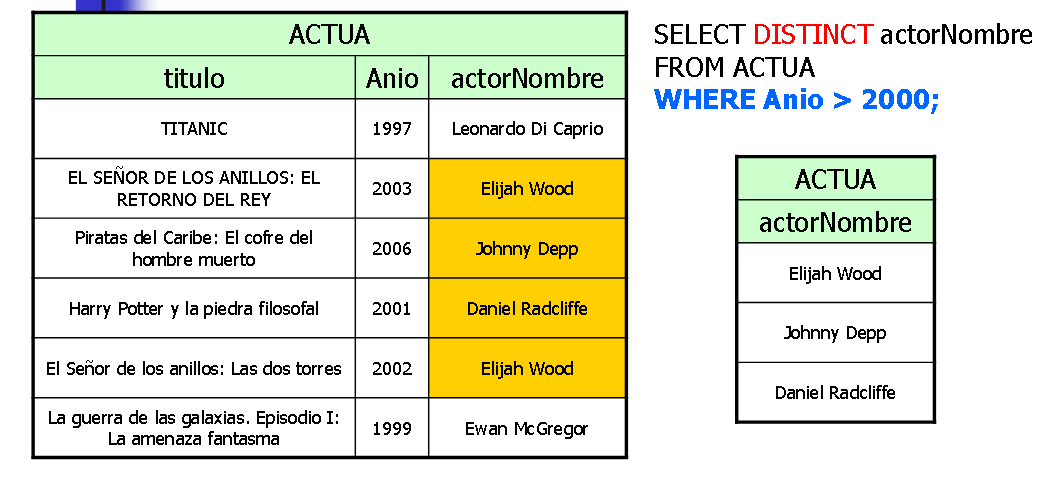
Para que no aparezcan atributos repetidos utilizamos:

**SELECT DISTINCT nombre\_Atrib FROM nombre\_tabla;**

**CONSULTA SIN DISTINCT:**

****

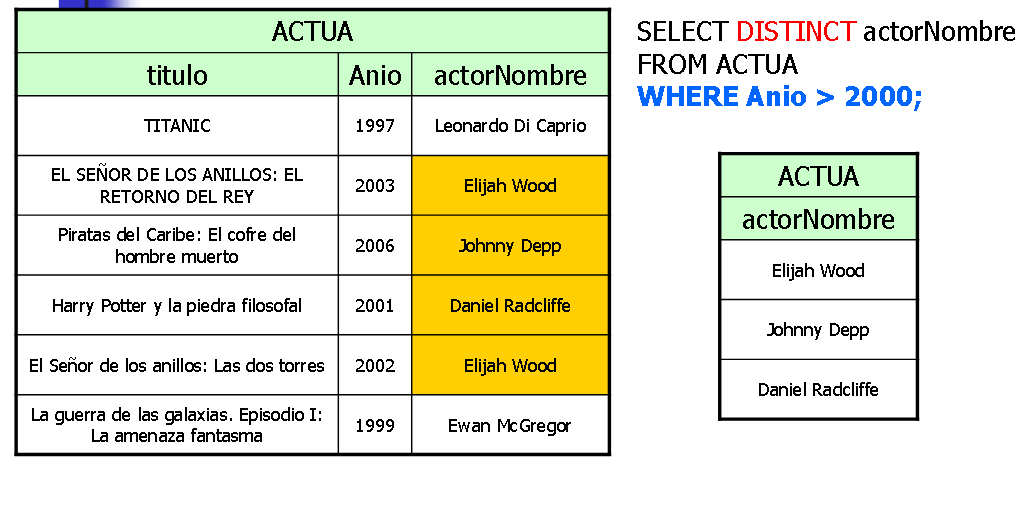
**CONSULTA CON DISTINC:**

****

**Si ponemos 2 atributos al hacer una consulta con DISTINCT va a buscar que el par de atributos sea distinto a otro par, por eso puede que el nombre de un actor aparezca 2 veces, pero como aparece en años distintos entonces las tuplas son distintas entre sí:**

****

Ejemplo de SELECT + WHERE:



**WHERE**

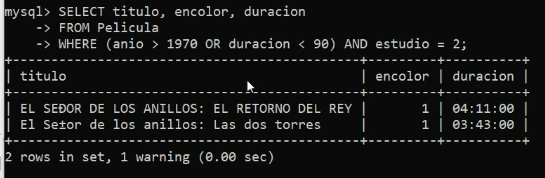
Se implementa a través de las expresiones condicionales en el WHERE.

Las expresiones se arman con los operadores de comparación **>, <, =, <>, >=, <=** y los operadores lógicos **AND, OR y NOT.**

Los valores que se comparan pueden incluir atributos de las relaciones mencionadas en el FROM y/o constantes.

Pueden usarse operadores aritméticos **(+, -, etc)** siempre que se respeten los tipos de los datos a comparar.

**Ejemplo:**



**Comparación de Strings**

Pueden realizarse comparaciones con operadores relacionales <, >, >=, <=, =, <> según el orden lexicográfico.

También comparaciones de patrones:

<Atributo> LIKE <patrón>

<Atributo> NOT LIKE <patrón>

El patrón es un string que puede contener dos caracteres especiales % y \_ que son comodines

**string LIKE patron [ESCAPE ‘<caracter>’]**

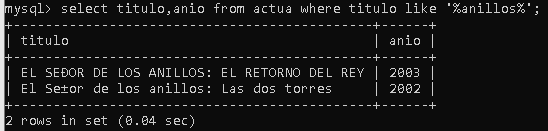
El carácter % (en s) puede corresponderse con cualquier cadena de 0 o más caracteres (en p).

El carácter \_ (en s) se corresponde con un único carácter cualquiera (en p)

**Ejemplo**:

**SELECT título, Anio FROM ACTUA WHERE título LIKE ´%anillos%´;**

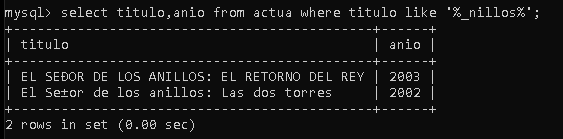
**Esto quiere decir** desde actúa buscamos títulos talque tengan el patrón **%anillos%** que **quiere** **decir** que **antes** de anillos **puede** **venir** **cualquier** **cosa** y **después** **también**, pero **tiene** **que** **estar** la **palabra** **anillos**.



**Observación**: acá busco anillos independientemente de si estaba en mayúsculas o no. Para discriminar y que busque exactamente como lo escribí habría que escribir LIKE BINARY en la consulta.

**Ejemplo** utilizando “**\_**”:

**SELECT título, Anio FROM ACTUA WHERE título LIKE ´%\_nillos%´;**



**Mejorando la Salida (ORDER BY)**

Para mostrar las tuplas resultantes ordenadas de acuerdo a ciertos atributos se puede utilizar la cláusula:

ORDER BY < lista de atributos separados por comas>

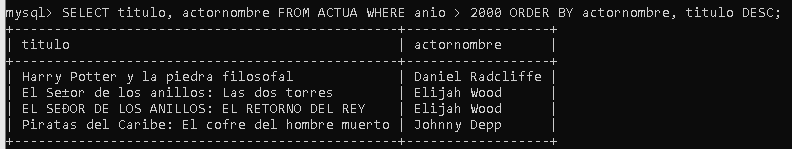
Por defecto es ascendente y con prioridad izquierda a derecha.

Si se desea que el ordenamiento sobre un cierto atributo sea de forma descendente, se agregar la cláusula DESC a continuación de dicho atributo.

**Ejemplo de ordenamiento:**

**SELECT titulo, actornombre FROM ACTUA WHERE anio > 2000**

**ORDER BY actornombre, titulo DESC;**



Podemos mejorar la legibilidad de los atributos al mostrar los resultados podemos renombrar los atributos mediante la cláusula **“AS <nombre>”.**

**Ejemplo**:

**SELECT titulo AS 'Titulo', anio AS 'año', actornombre AS 'actor' FROM actua WHERE anio>2000;**

**Funciones**

**En el manual se pueden encontrar en la sección Functions and operators**

**Dentro del SELECT o del WHERE pueden ir funciones:**

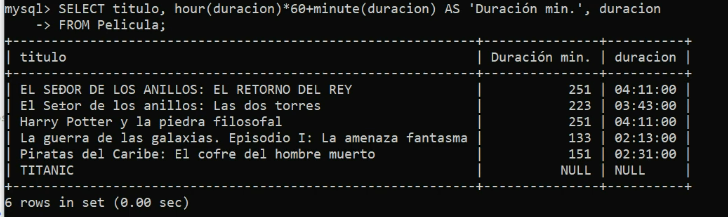
Dada la tabla: PELICULA(titulo, anio, duracion, encolor, estudio, director#)

Expresiones: Se pueden hacer funciones que se combinen con atributos para crear atributos derivados.

**SELECT titulo, hour(duracion)\*60+minute(duracion) AS**

**‘Duración min.’ FROM Pelicula;**

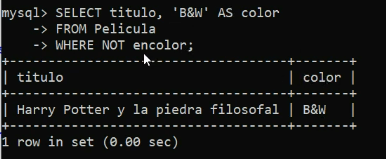
En esta función **pasamos** las **horas** **a** **minutos** **y lo sumamos a los minutos** que tenga, para **obtener** la **duración** de la **peli** en **minutos**



Constantes:

**SELECT titulo**, ‘**B&W**’ **AS color FROM Pelicula WHERE NOT encolor;**

**Acá le estamos pidiendo que devuelva los títulos de películas que son en blanco y negro le pedimos que el nombre de la columna para B&W sea “color”.**



**Consultas sobre múltiples relaciones**

Supongamos querer conocer el nombre del director de la saga “El señor de los Anillos”.

Sean las tablas:

PELICULA(titulo, anio, duracion, encolor, estudio, director)

DIRECTOR(nombre, direccion, cert)

La consulta seria:

**SELECT DISTINCT nombre as ‘Nombre Director’**

**FROM PELICULA, DIRECTOR**

**WHERE titulo LIKE ‘%anillos%’ AND director = cert;**

Hacemos el producto entre película y director y buscamos donde el titulo tenga anillos en alguna parte y director=cert, y nos quedamos con el nombre del director que no se va a repetir gracias al DISTINCT.

**Otra consulta:**

Supongamos querer conocer el nombre del director y actores que vivan juntos.

Para distinguir atributos con el mismo nombre utilizamos “<relación>. <atributo>”

**Dadas las tablas:**

ACTOR(nombre, direccion, sexo, fechaNac)

DIRECTOR(nombre, direccion, cert)

Si vamos a hacer una consulta sobre nombre como se repite en ambas tablas hay que especificar que te tabla viene:

**SELECT DISTINCT ACTOR.nombre, DIRECTOR.nombre**

**FROM ACTOR, DIRECTOR**

**WHERE ACTOR.direccion = DIRECTOR.direccion;**

**Mas consultas:**

Supongamos querer conocer el nombre de todos los pares de actores que vivan juntos.

ACTOR(nombre, direccion, sexo, fechaNac)

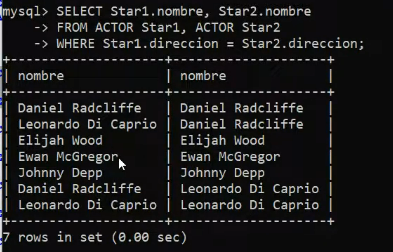
SQL permite establecer un alias para las relaciones de la clausula FROM.

**SELECT Star1.nombre, Star2.nombre**

**FROM ACTOR Star1, ACTOR Star2**

**WHERE Star1.direccion = Star2.direccion**

Se lee desde actor 1 y actor 2 donde la dirección sea igual.

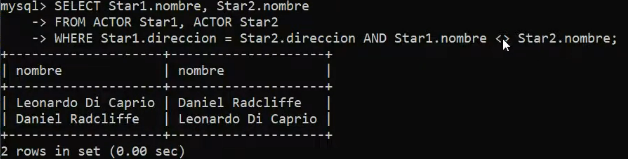


Para que no aparezca que un actor vive consigo mismo podemos agregar:

**SELECT Star1.nombre, Star2.nombre**

**FROM ACTOR Star1, ACTOR Star2**

**WHERE Star1.direccion = Star2.direccion AND Star1.nombre <> Star2.nombre;**

****

Otra solución seria pedirle que el nombre sea menor o mayor al otro nombre, o sea que no serian iguales. Esto sería modificando la última línea con **Star1.nombre < Star2.nombre;**

**Unión, Intersección, Diferencia**

Las operaciones de unión, intersección y diferencia sobre relaciones se expresan de la siguiente forma:

(subconsulta) UNION (subconsulta)

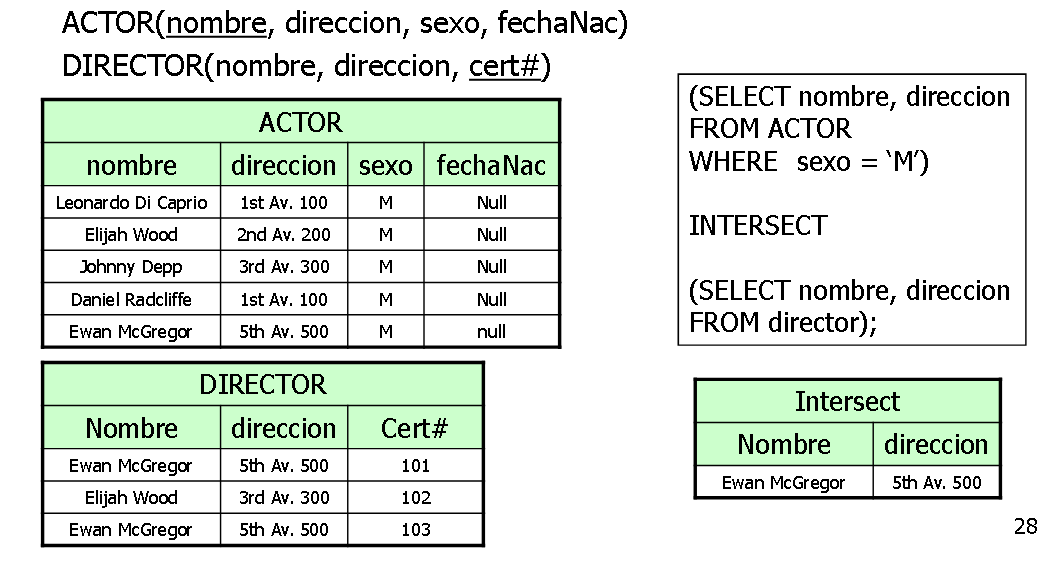
(subconsulta) INTERSECT (subconsulta)

(subconsulta) EXCEPT (subconsulta)

Las cláusulas UNION, INTERSECT y EXCEPT permiten vincular dos subconsultas con la misma estructura (esquema).

**Intersección**

Veremos la intersección entre los resultados de las siguientes consultas:



Esta intersección nos muestra el nombre y la dirección de los hombres que son actores y directores.

A diferencia de la sentencia **SELECT FROM WHERE** que utiliza la semántica de bolso (Bag), **las operaciones de unión, intersección y diferencia** utilizan la **semántica** de **conjunto** (Set). Es decir que **todas las túplas duplicadas son eliminadas**.

Para evitar eliminar duplicados es necesario escribir la palabra reservada ALL luego de UNION, INTERSECT o EXCEPT.

**Subconsultas**

Una consulta que es parte de otra consulta es llamada subconsulta.

Las subconsultas se escriben entre paréntesis

Las subconsultas pueden utilizarse en:

* UNION, INTERSECT, EXCEPT
* Cláusula WHERE
  + Retornando un valor escalar. O sea, una sola tupla con un solo atributo.
  + Retornando una relación. Muchas tuplas con muchos atributos.
* Cláusula FROM
* En algunos dialectos en la cláusula SELECT

Si una subconsulta garantiza producir una única túpla, entonces la subconsulta puede ser utilizada como un valor.

Usualmente, estamos interesados en un único atributo, que además utilizando las llaves y otra información podemos deducir que se producirá un único valor.

Si la subconsulta produce más de una túpla o ninguna, se produce un error en tiempo de ejecución.

**Ejemplo:** Supongamos querer conocer el nombre del director de la saga “El señor de los Anillos”.

**PELICULA(titulo, anio, duracion, encolor, estudio, director)**

**DIRECTOR(nombre, direccion, cert)**

**Aclaración importante: director en la tabla PELICULA es un numero**

Le vamos a pedir que el cert del director (su identificador sea igual a la consulta que esta entre parentesis)

**SELECT nombre FROM DIRECTOR**

**WHERE cert = (**

**SELECT DISTINCT director**

**FROM PELICULA**

**WHERE titulo LIKE ‘%anillos%’ );**

**Esto el profe dice que para pensar las consultas hay que dividirlas por partes y se puede pensar de adentro para afuera o viceversa.**

**Él lo piensa así: ¿qué es lo que necesito para devolver el nombre de un director? Acceder a la relación director. ¿Como identifico al director? A través del cert. ¿Y el cert como se cuál es? Lo se por el director que está en película**

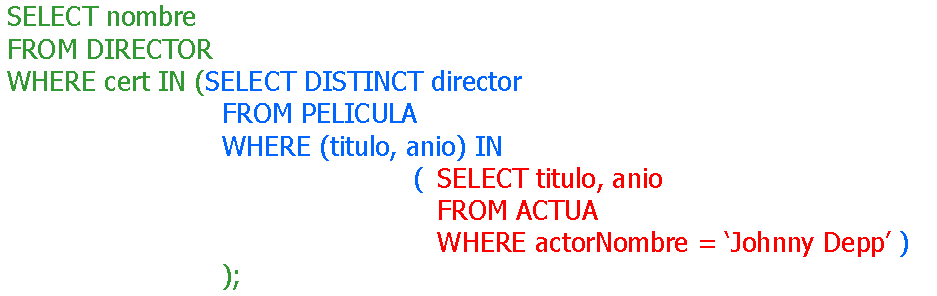
Cuando se utiliza una subconsulta en la cláusula WHERE y la subconsulta retorna una relación, SQL provee una serie de operadores para obtener un resultado booleano.

Sea s un valor escalar (valor atómico que puede aparecer como componente de una tupla) y R una relación unaria.

* **EXISTS** **R** es verdadero si R no es vacío.
* **NOT EXISTS** R es verdadero si R es vacío
* **s IN R** es verdadero si el atributo s está en R. También puede ser precedido por NOT (s NOT IN R).
* **s > ALL R** es verdadero si el atributo s es mayor que todos los elementos de R.
* Pueden utilizarse cualquiera de los otros operadores de comparación en lugar de >.
* **s > ANY R** es verdadero si el atributo s es mayor que alguno de los elementos de R. También se puede utilizar SOME.

Si s es una tupla y R una relación con la misma aridad de s entonces los operadores antes mencionados también podrán ser utilizados.

**Ejemplo:**



Leyendo de adentro hacia afuera:

De ACTÚA dame el título y año donde actuó Johnny Depp.

De PELICULA dame el director donde el título y año se encuentren en la consulta anterior.

Como en DIRECTOR tenemos el nombre del director a diferencia de las otras tablas que nomas tienen su identificador, entonces acá sacamos de DIRECTOR el nombre del director de las películas en las que actuó Johnny Deep.

**Expresiones JOIN**

Utilizan semántica bag y no semántica set.

**Theta Join: R JOIN S ON <condición>**

**Ejemplo:**

**SELECT DISTINCT director**

**FROM PELICULA P JOIN ACTUA A ON P.titulo = A.titulo**

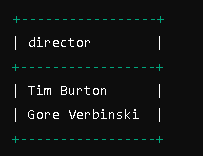
**AND P.anio = A.anio**

**WHERE actorNombre = ‘Johnny Depp’;**

**Explicación de la parte del JOIN:**

FROM PELICULA P JOIN ACTUA A ON P.titulo = A.titulo AND P.anio = A.anio

* Se realiza una combinación o *join* entre dos tablas: PELICULA (con el alias P) y ACTUA (con el alias A).
* El *join* se basa en que el título y el año de las películas en ambas tablas coincidan (P.titulo = A.titulo y P.anio = A.anio). Es decir, se asocian las películas con los actores que actuaron en ellas mediante estos dos campos.



**JOIN Natural**

Este join difiere del Theta-join en:

1. La condición del join es implícita, igualando todos los atributos de ambas relaciones que tengan el mismo nombre (por eso no se especifican los atributos creo).

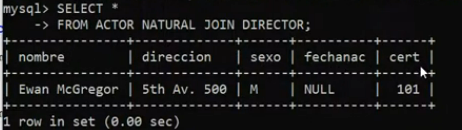
2. Sólo uno de los atributos igualados es dejado como resultado.

**Ejemplo Join natural:**

DIRECTOR(nombre, direccion, cert)

ACTOR(nombre, direccion, sexo, fechaNac)

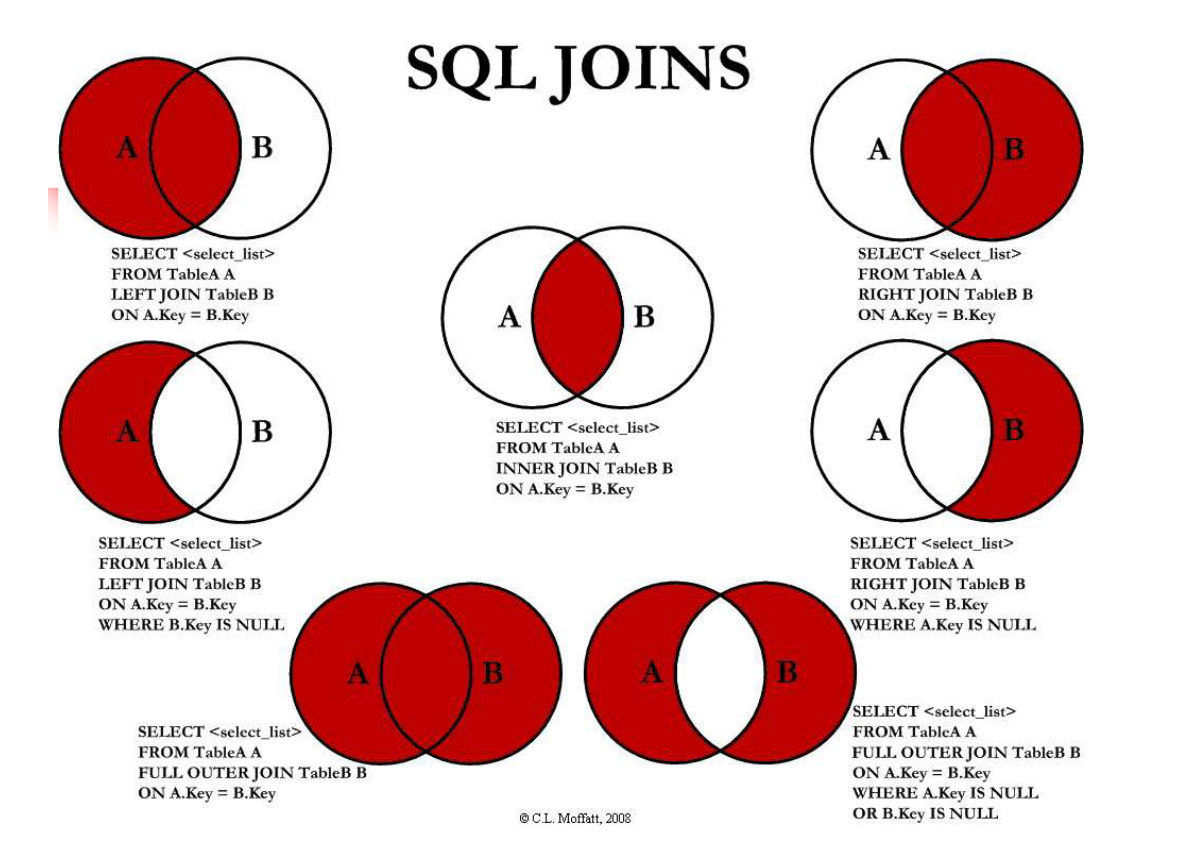
**SELECT \* FROM ACTOR NATURAL JOIN DIRECTOR;**

****

**Outerjoins**

**R OUTER JOIN S** incorpora además las tuplas que no satisfacen la condición (dangling tuples) o que tienen valores null.

**Otros joins:**



**Operadores de Agregación**

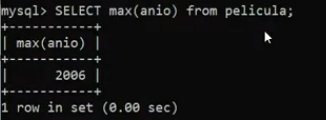
**Los siguientes operadores se aplican generalmente sobre atributos o sea sobre columnas.**

* **AVG** (atributo) o promedio. Se aplica sobre columnas numéricas.
* **COUNT** (atributo) retorna el número de valores no nulos.
* **MAX** (atributo) retorna el valor máximo.
* **MIN** (atributo) retorna el valor mínimo.
* **SUM** (atributo) retorna la suma.
* **COUNT(\*)** retorna el número de tuplas de la relación.
* Se pueden combinar con DISTINCT u ALL (por defecto).
  + **COUNT (DISTINCT nombres)** contar los nombres una sola vez aunque se repitan.
* No se pueden componer funciones agregadas, por ej. max(avg()).

Estas funciones se aplican sobre todas las tuplas de la relación y retornan un resultado.

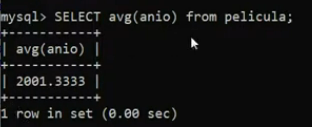
**Ejemplo:**

**SELECT max(anio) FROM película;**



**Ejemplo:**

**SELECT avg(anio) FROM película;**



**Ejemplo**: Promedio de duración de las películas a color.

**SELECT AVG(duracion)**

**FROM PELICULA**

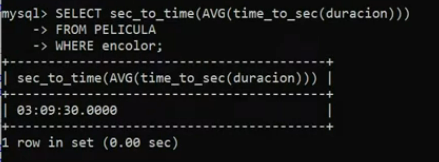
**WHERE encolor;**

**Ejemplo** utilizando funciones:

**SELECT sec\_to\_time(AVG(time\_to\_sec(duracion)))**

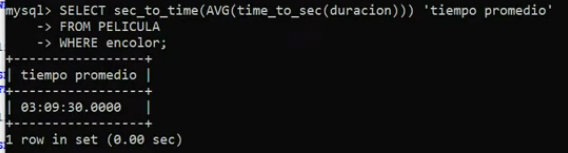
**FROM película**

**WHERE encolor;**



Esto es las duraciones las pasamos a segundos, calculamos el promedio de las duraciones en segundos de las películas y dsps lo volvemos a pasar a tiempo.

Mismo ejemplo, pero cambiándole el nombre al campo de la tabla:



Clase 4 47:30