

**Universidad Especializada de las Américas**

**Licenciatura en Biomédica con Especialización en Electrónica Médica**

**Programación Avanzada**

**Taller de base de datos II**

Ver:2.0 03/junio/2017

Importante: los programas para instalar o ejecutar se deben hacer desde el modo Administrador, de lo contrario puede aparecer un mensaje de error por acceso denegado, esto ocurre mucho cuando los programas se instalan en Archivos de Programas o Programs Files y se intenta modificar algún archivo.

**Prerrequisitos:**

Leer , instalar y configurar los programas explicados en los documentos:

- Herramientas de Programación usadas en clase. Parte I.
- Herramientas de Programación usadas en clase. Parte II.
- Taller de base de datos I.

En el curso usaremos la base de datos FILM.FBD descargada de :

<https://github.com/damianquijano/PythonCurso3/tree/master/Data>

junto con otros archivos en formato CSV.

Utilizaremos los softwares:

- Firebird.
- SQL Manager Lite .

**Qué es una base de datos?**

Veamos una **tabla**(de varias que contiene una base de datos) de dicha base de datos(base de datos llamada FILM.FDB) en formato Excell llamada **Film20SD**:

film_id	category_id	title	release_year	language_id	rental_duration	rental_rate	length	replacement_cost	rating
1	11	AFFAIR PREJUDICE	2010	1	5	2.99	117	26.99	G
2	11	AIRPORT POLLOCK	2016	6	6	4.99	54	15.99	R
3	11	ALABAMA DEVIL	2011	3	3	2.99	114	21.99	PG-13
4	11	ALI FOREVER	2008	1	4	4.99	150	21.99	PG
5	16	BASIC EASY	2007	3	4	2.99	90	18.99	PG-13
6	1	CAMPUS REMEMBER	2014	3	5	2.99	167	27.99	R
7	15	CARIBBEAN LIBERTY	2015	5	3	4.99	92	16.99	NC-17
8	2	CAROL TEXAS	2006	2	4	2.99	151	15.99	PG
9	1	CASUALTIES ENCINO	2013	1	3	4.99	179	16.99	G
10	9	CATCH AMISTAD	2016	5	7	0.99	183	10.99	G
11	6	DELIVERANCE MULHOLLAND	2006	4	4	0.99	100	9.99	R
12	16	DESPERATE TRAINSPOTTING	2013	3	7	4.99	81	29.99	G
13	4	DETECTIVE VISION	2013	5	4	0.99	143	16.99	PG-13
14	7	DIARY PANIC	2016	2	7	2.99	107	20.99	G
15	8	EFFECT GLADIATOR	2013	3	6	0.99	107	14.99	PG
16	6	EGG IGBY	2012	1	4	2.99	67	20.99	PG
17	12	ELF MURDER	2007	2	4	4.99	155	19.99	NC-17
18	16	ENOUGH RAGING	2006	1	7	2.99	158	16.99	NC-17
19	15	EVOLUTION ALTER	2008	3	5	0.99	174	10.99	PG-13

Pues tal como aparece arriba, también se muestra en nuestro programa manejador SQL Manager Lite.

Primero vamos a definir los conceptos de filas (o registros ) y columnas(campos o variables):

Columna o Campo 0 (film\_id)      Columna o Campo 2 (title)      Columna o Campo 9 ( rating)

Fila o Registro 0

Fila o Registro 5

Fila o Registro 14

film_id	category_id	title	release_year	language_id	rental_duration	rental_rate	length	replacement_cost	rating
1	11	AFFAIR PREJUDICE	2010	1	5	2.99	117	26.99	G
2	11	AIRPORT POLLOCK	2016	6	6	4.99	54	15.99	R
3	11	ALABAMA DEVIL	2011	3	3	2.99	114	21.99	PG-13
4	11	ALI FOREVER	2008	1	4	4.99	150	21.99	PG
5	16	BASIC EASY	2007	3	4	2.99	90	18.99	PG-13
6	1	CAMPUS REMEMBER	2014	3	5	2.99	167	27.99	R
7	15	CARIBBEAN LIBERTY	2015	5	3	4.99	92	16.99	NC-17
8	2	CAROL TEXAS	2006	2	4	2.99	151	15.99	PG
9	1	CASUALTIES ENCINO	2013	1	3	4.99	179	16.99	G
10	9	CATCH AMISTAD	2016	5	7	0.99	183	10.99	G
11	6	DELIVERANCE MULHOLLAND	2006	4	4	0.99	100	9.99	R
12	16	DESPERATE TRAINSPOTTING	2013	3	7	4.99	81	29.99	G
13	4	DETECTIVE VISION	2013	5	4	0.99	143	16.99	PG-13
14	7	DIARY PANIC	2016	2	7	2.99	107	20.99	G
15	8	EFFECT GLADIATOR	2013	3	6	0.99	107	14.99	PG
16	6	EGG IGBY	2012	1	4	2.99	67	20.99	PG
17	12	ELF MURDER	2007	2	4	4.99	155	19.99	NC-17
18	16	ENOUGH RAGING	2006	1	7	2.99	158	16.99	NC-17
19	15	EVOLUTION ALTER	2008	3	5	0.99	174	10.99	PG-13

Fila o Registro 0	1	11	AFFAIR PREJUDICE	2010	1	5	2.99	117	26.99	G
Fila o Registro 5	6	1	CAMPUS REMEMBER	2014	3	5	2.99	167	27.99	R

Columna o Campo 0 (film\_id)

film_id
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Columna o Campo 2 (title)

title
AFFAIR PREJUDICE
AIRPORT POLLOCK
ALABAMA DEVIL
ALI FOREVER
BASIC EASY
CAMPUS REMEMBER
CARIBBEAN LIBERTY
CAROL TEXAS
CASUALTIES ENCINO
CATCH AMISTAD
DELIVERANCE MULHOLLAND
DESPERATE TRAINSPOTTING
DETECTIVE VISION
DIARY PANIC
EFFECT GLADIATOR
EGG IGBY
ELF MURDER
ENOUGH RAGING
EVOLUTION ALTER

Cabe destacar que la primera columna es la 0, y la primera fila es también la 0, no empiezan a partir de la 1, esto por general produce confusiones. Por ejemplo, en el campo fil\_id, tenemos el valor 1, pues el humano tiende a etiquetar los valores iniciales a partir del 1, pero para la base de datos, el registro con el valor film\_id=1 es el 0.

Por tanto, Film20SD es **una tabla** conformada por columnas(campos) y filas(registros).

Y la base de datos? es una tabla?: no necesariamente, una base de datos está integrada por una o más tablas, es un conjunto o colección de tablas.

La base de datos FILM.FDB tiene varias tablas.

Tabla llamada **Film20SD**:

film_id	category_id	title	release_year	language_id	rental_duration	rental_rate	length	replacement_cost	rating
1	11	AFFAIR PREJUDICE	2010	1	5	2.99	117	26.99	G
2	11	AIRPORT POLLOCK	2016	6	6	4.99	54	15.99	R
3	11	ALABAMA DEVIL	2011	3	3	2.99	114	21.99	PG-13
4	11	ALI FOREVER	2008	1	4	4.99	150	21.99	PG
5	16	BASIC EASY	2007	3	4	2.99	90	18.99	PG-13
6	1	CAMPUS REMEMBER	2014	3	5	2.99	167	27.99	R
7	15	CARIBBEAN LIBERTY	2015	5	3	4.99	92	16.99	NC-17
8	2	CAROL TEXAS	2006	2	4	2.99	151	15.99	PG
9	1	CASUALTIES ENCINO	2013	1	3	4.99	179	16.99	G
10	9	CATCH AMISTAD	2016	5	7	0.99	183	10.99	G
11	6	DELIVERANCE MULHOLLAND	2006	4	4	0.99	100	9.99	R
12	16	DESPERATE TRAINSPOTTING	2013	3	7	4.99	81	29.99	G
13	4	DETECTIVE VISION	2013	5	4	0.99	143	16.99	PG-13
14	7	DIARY PANIC	2016	2	7	2.99	107	20.99	G
15	8	EFFECT GLADIATOR	2013	3	6	0.99	107	14.99	PG
16	6	EGG IGBY	2012	1	4	2.99	67	20.99	PG
17	12	ELF MURDER	2007	2	4	4.99	155	19.99	NC-17
18	16	ENOUGH RAGING	2006	1	7	2.99	158	16.99	NC-17
19	15	EVOLUTION ALTER	2008	3	5	0.99	174	10.99	PG-13

Tabla llamada **Category**:

category_id	name
1	Action
2	Animation
3	Children
4	Classics
5	Comedy
6	Documentary
7	Drama
8	Family
9	Foreign
10	Games
11	Horror
12	Music
13	New
14	Sci-Fi
15	Sports
16	Travel

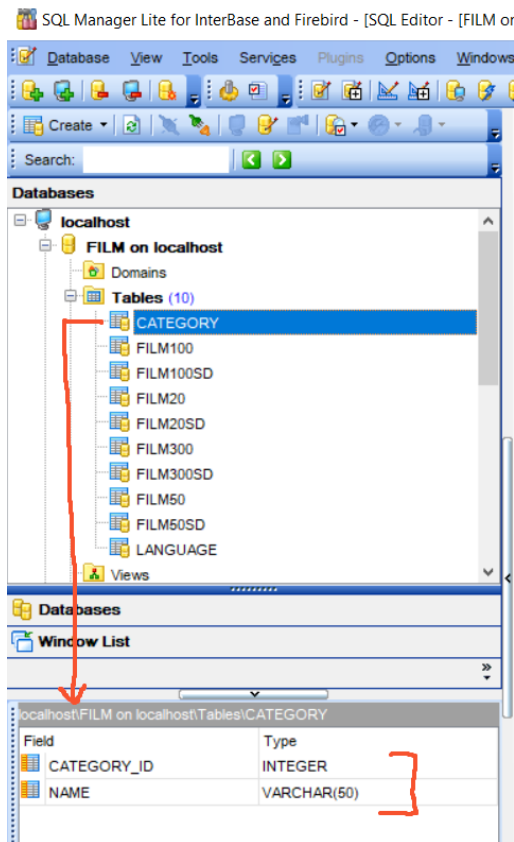
Tabla llamada **Language**:

language_id	name
1	English
2	Italian
3	Japanese
4	Mandarin
5	French
6	German

A continuación abrimos el programa SQL Manager Lite, y en la pantalla veremos configurados y conectados los servidores Host y la base de datos FILM. Una vez más recordamos que se deben leer los docs:

- Herramientas de Programación usadas en clase. Parte I.
- Herramientas de Programación usadas en clase. Parte II.
- Taller de base de datos I.

En el último documento se explica el registro y conexión del Host y de la base de datos FILM.



Al seleccionar una tabla, abajo, en Windows List, se muestran los campos de la tabla seleccionada.

Arriba, tenemos seleccionada la tabla CATEGORY, y abajo aparecen los campos que contiene dicha tabla. No aparecen aún los registros, los podremos ver usando las instrucciones del lenguaje SQL más adelante.

Pulse encima de cada tabla y vea el contenido de las columnas en la ventana de columnas, abajo.

Se dará cuenta que las tablas llamadas Film con el sufijo SD contienen las mismas columnas, la diferencia es la cantidad de registros que contienen, por ejemplo: Film20SD contiene 20 filas, y Film300SD contiene 300 filas. La diferencia entre las tablas con el prefijo SD y las que no tienen el prefijo SD, por ejemplo: Film20SD y Film20, es que Film20 contiene un campo más llamado Description.

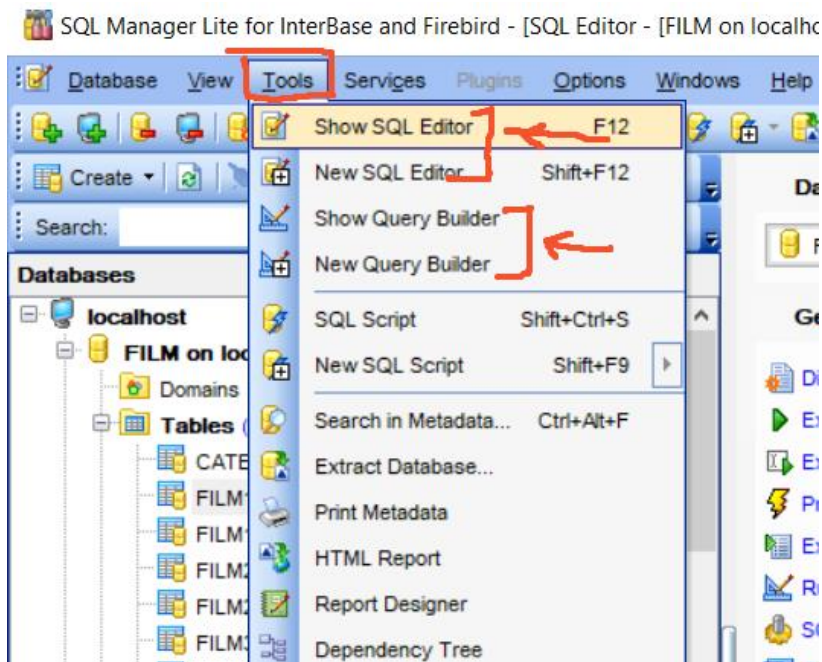
Las otras dos tablas son Language y Category con sus campos muy diferentes a los usados en las tablas del tipo Film.

La base de datos Film.fdb es utilizada por un centro de comercio dedicado a rentar o alquilar películas. La base de datos en realidad es mucho más grande, pues contiene otras tablas como: inventario, facturas, clientes, proveedores, nombres de artistas, etc, pero en esta ocasión, siendo nuestro objetivo el aprendizaje de las bases de datos en su etapa

introductoria, es suficiente con tres tablas. Por tanto FILM.FDB contiene los datos sobre películas, tales como :categoría de película: horror o ciencia ficción, entre otras, título de la película, longitud de duración de la película, rating, y otra información que es contenida en varias tablas, no solo en una.

Nos podemos percatar, que en la tabla Film20SD tenemos el campo Category\_Id, que nos indica el código de categoría al que pertenece un registro, pero no aparece el nombre de la categoría, solamente vemos el número asociado a la categoría (el código), en cambio, si vemos el contenido de la tabla Category, podemos ver el nombre de la categoría correspondiente al número de categoría en la tabla Film20SD. Más adelante, aprenderemos a relacionar dos o más tablas para que crucen su información y aparezca las columnas mezcladas de diferentes tablas.

Vamos a utilizar dos opciones muy importantes del programa SQL Manager Lite:



Las dos opciones son:

- Editor SQL
- Query Builder o constructor de queries


A continuación vamos a practicar instrucciones del lenguaje de programación llamado SQL utilizado por la mayoría de las bases de datos para consultar los datos de las tablas.

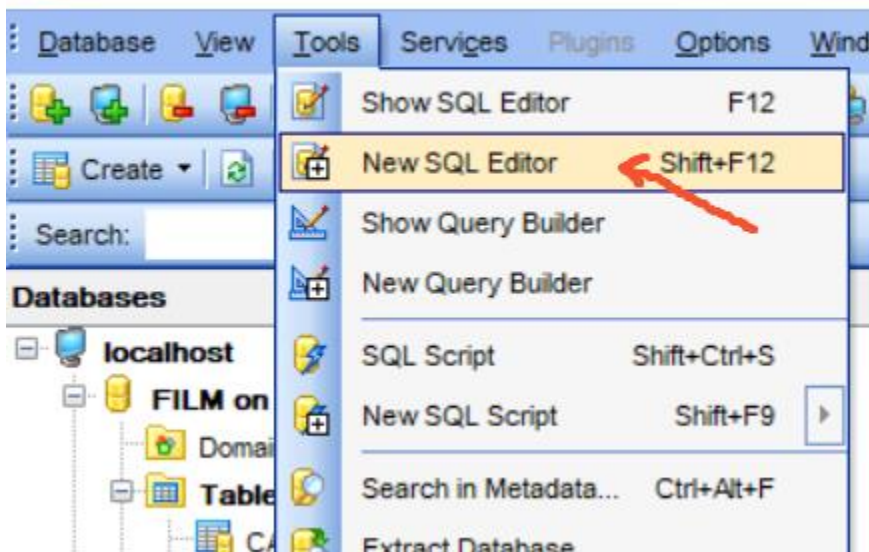
Estas consultas son llamadas Querys en el vocablo del mundo de los programadores.

La instrucciones de consulta utilizan la instrucción: Select y significa extraer o traer información de la base de datos (ya sea de una o más tablas de la base de datos). Dicha instrucción está acompañada por otras instrucciones que permiten identificar la tabla consultada con sus columnas y condiciones de selección(una condición-por ejemplo- es que la columna llamada Año cumpla que su valor sea > 2000, lo cual hará que Select nos muestre solamente aquellos registros que cumplen la condición de que en la columna Año el valor sea mayor a 2000, ignorando los restantes registros que no cumplen la condición).

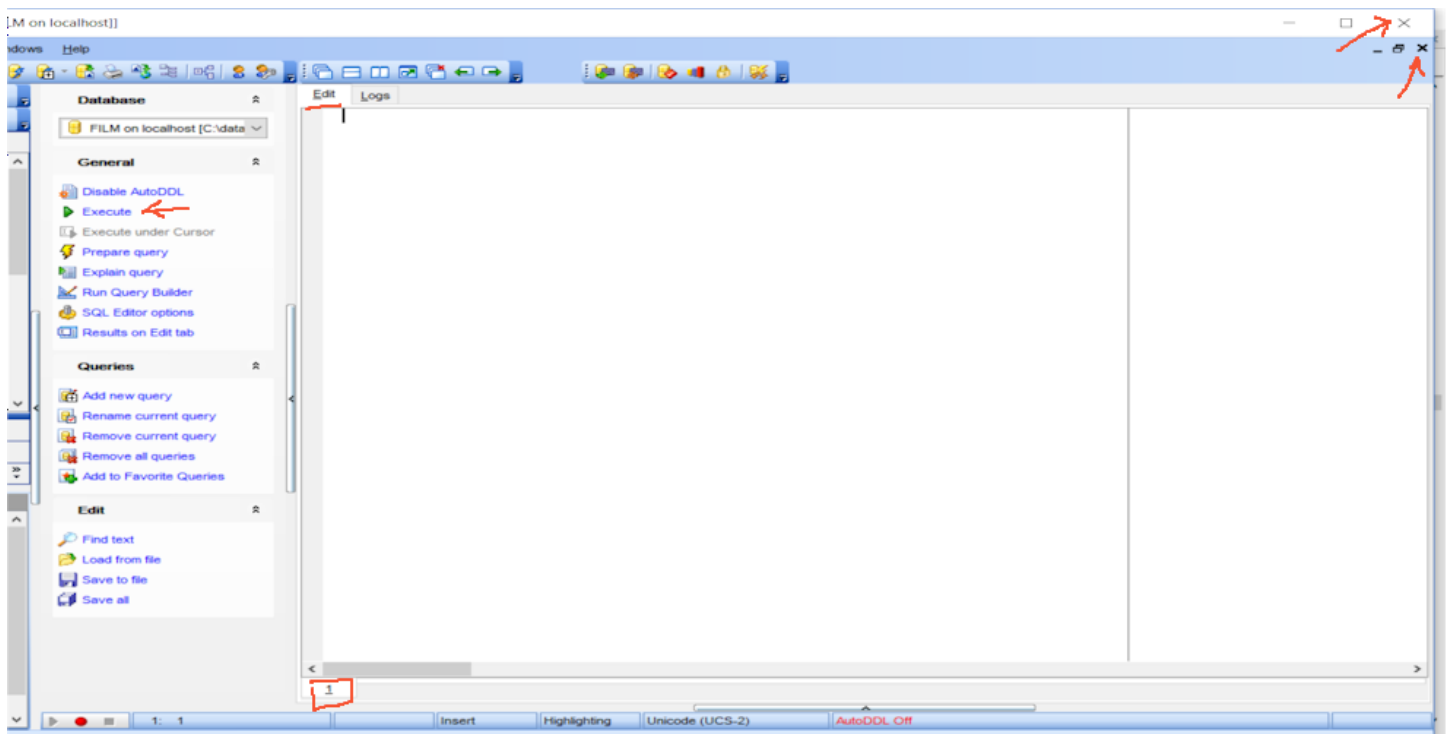
Debemos poner mucha atención a las palabras: columnas o campos, registros o filas, tablas, bases de datos, consultas, queries, para lograr una imagen mental correcta sobre los resultados que deseamos lograr.

Veamos nuestra primera consulta , para lo cual primero debes activar la opción Editor SQL (new):

 SQL Manager Lite for InterBase and Firebird



Abajo aparece la pantalla para escribir nuestros Querys, en este caso consultas del tipo Select:



Arriba vemos la pantalla para escribir nuestras instrucciones. es importante observar las dos cruces arriba a la derecha: la cruz más oscura y debajo de la otra, es la que cierra la pantalla de edición de SQL, y se usa mucho cuando ya no se desea continuar y queremos escribir otra diferente y no deseamos tener tantos querys en la pantalla. La otra cruz cierra el programa , y hay que tener cuidado con no confundir ambas cruces.

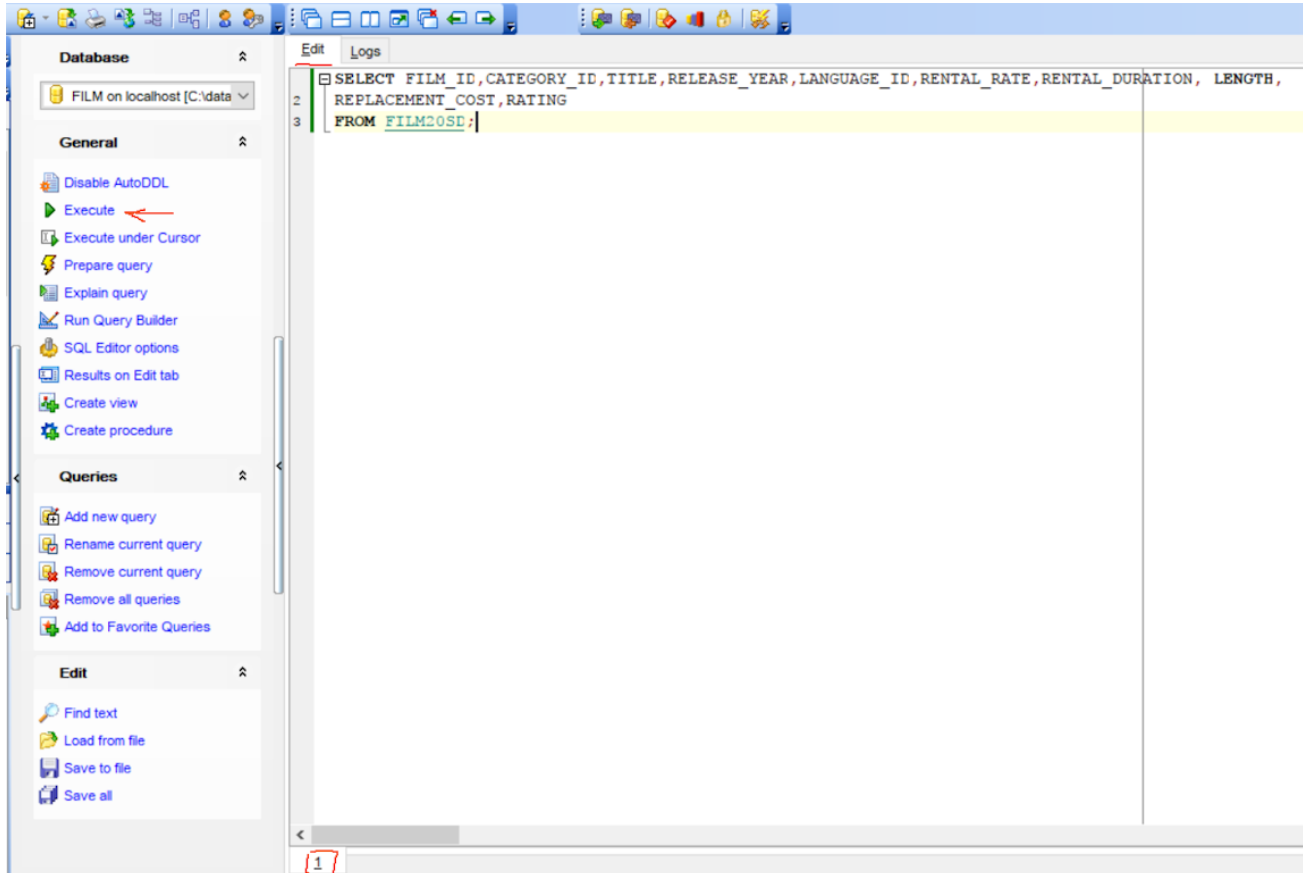
En la ventana de edición de SQL, hay dos pestañas o tabs, una es Edit y otra es Log, pero a medida que seguimos trabajando con la ventana, irán apareciendo otras pestañas. Abajo tenemos una pestaña que nos presenta el 1, esto significa que es el Query 1. A medida que agreguemos más querys que deseamos escribir, aparecerán los números 2, 3 y tantos como Querys escribamos, por tanto, si deseo saltar del query 10 al 1, solamente tengo que pulsar en la pestaña 1 y podré ver las instrucciones sql construidas para ese caso. Todo esto se entenderá mejor a medida que practiquemos.

Escribamos nuestra primera consulta. Usted puede copiar y pegar directamente desde este documento al programa SQL Manager Lite, pero es mejor que intente escribir las instrucciones para que con las equivocaciones pueda aprender en vez de repetir. Seguramente se equivocará al escribir una coma que sobre o que falta, o bien al escribir el nombre de un campo de forma incorrecta. Es necesario que se equivoque al momento de practicar, que al momento de resolver un examen.



Veamos la primera consulta:

```
SELECT FILM_ID,CATEGORY_ID,TITLE,RELEASE_YEAR,LANGUAGE_ID,RENTAL_RATE,RENTAL_DURATION,  
LENGTH,REPLACEMENT_COST,RATING  
  
FROM FILM20SD;
```



La instrucción **Select** nos permite seleccionar los campos que deseamos visualizar, a continuación se enumeran los campos, en esta ocasión hemos decidido ver todos. Es importante escribir correctamente los nombres de los campos. Los campos están separados por comas. El campo final no lleva coma después de dicho campo. La instrucción **From** permite que indiquemos la tabla a la que pertenecen los campos seleccionados mediante Select. Se termina con punto y coma.

Al pulsar **Execute** veremos los resultados- en caso de error, revise bien la sintaxis.

Abajo vemos una nueva pestaña llamada Results (subrayada en rojo) , la cual muestra los registros de la consulta, en este caso aparecen todas las columnas y todos los registros, o sea, la tabla completa. Si no logra ver todas las columnas, puede mover la barra scroll horizontal abajo señalada para ver el resto de los campos o columnas de la tabla.

Database: FILM on localhost [C:\data]

General

- Disable AutoDDL
- Execute
- Execute under Cursor
- Prepare query
- Explain query
- Run Query Builder
- SQL Editor options
- Results on Edit tab
- Create view
- Create procedure

Queries

- Add new query
- Rename current query
- Remove current query
- Remove all queries
- Add to Favorite Queries

Data Management

- Commit transaction
- Rollback transaction
- Export data
- Export as SQL script
- Import data

Results

Drag a column header here to group by that column

FILM_ID	CATEGORY	TITLE	RELEASE_YEAR	LANGUAGE	RENTAL_RATE	RENTAL_DURATION	LENGTH	REPLACEMENT_COST
1	11	AFFAIR PREJUDICE	2,010	1	2.99	5	117	
2	11	AIRPORT POLLOCK	2,016	6	4.99	6	54	
3	11	ALABAMA DEVIL	2,011	3	2.99	3	114	
4	11	ALI FOREVER	2,008	1	4.99	4	150	
5	16	BASIC EASY	2,007	3	2.99	4	90	
6	1	CAMPUS REMEMBER	2,014	3	2.99	5	167	
7	15	CARIBBEAN LIBERTY	2,015	5	4.99	3	92	
8	2	CAROL TEXAS	2,006	2	2.99	4	151	
9	1	CASUALTIES ENCINO	2,013	1	4.99	3	179	
10	9	CATCH AMISTAD	2,016	5	0.99	7	183	
11	6	DELIVERANCE MULHOLLAND	2,006	4	0.99	4	100	
12	16	DESPERATE TRAINSPOTTING	2,013	3	4.99	7	81	
13	4	DETECTIVE VISION	2,013	5	0.99	4	143	
14	7	DIARY PANIC	2,016	2	2.99	7	107	
15	8	EFFECT GLADIATOR	2,013	3	0.99	6	107	
16	6	EGG IGBY	2,012	1	2.99	4	67	
17	12	ELF MURDER	2,007	2	4.99	4	155	
18	16	ENOUGH RAGING	2,006	1	2.99	7	158	
19	15	EVOLUTION ALTER	2,008	3	0.99	5	174	

Grid View Form View Print Data

----- QUERY PERFORMANCE -----  
 Prepare : 0 ms  
 Execute : 0 ms  
 Avg fetch time: 0 ms

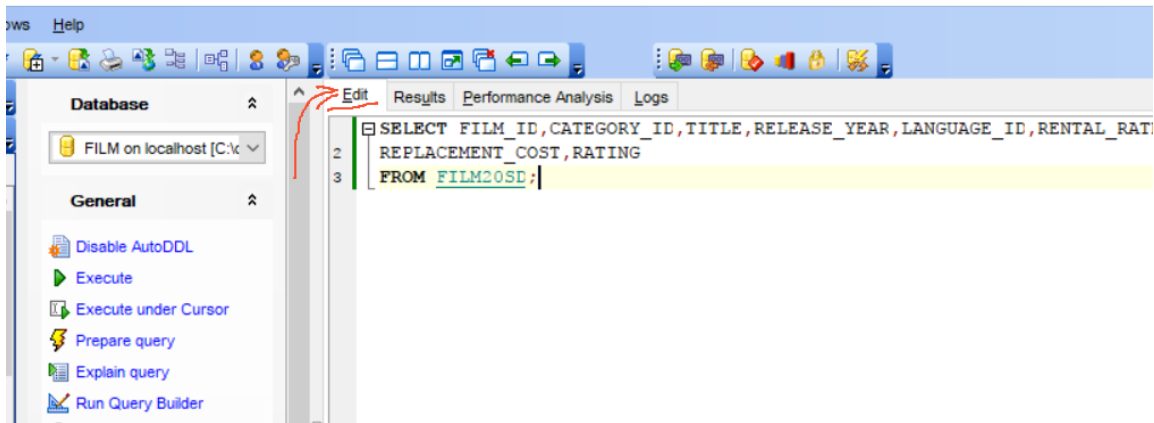
----- MEMORY -----

Fetches: 19/19

Unicode (UCS-2)

Los resultados se pueden presentar como una tabla (grid), o como una formulario o como una vista para imprimir. Abajo están las tres formas que usted puede probar.

Si queremos regresar a la pestaña de las instrucciones SQL, solo hay que pulsar en la pestaña Edit nuevamente:

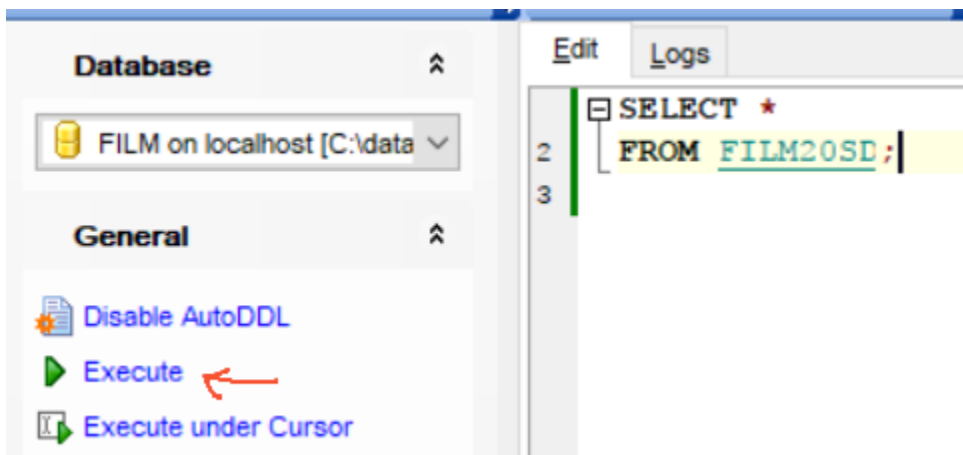
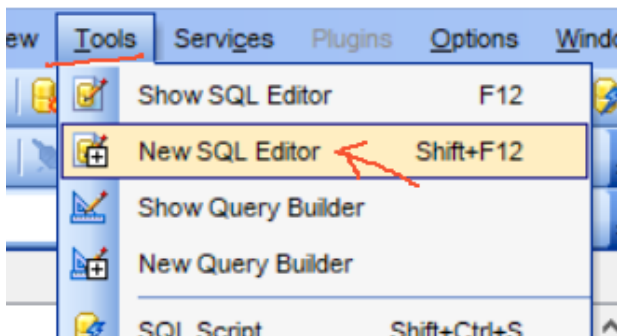


Y si queremos ver los Resultados, pulsamos nuevamente la pestaña Results.

Con la siguiente instrucción provocamos el mismo resultado que el query anterior, pero de forma más abreviada:

```
SELECT *  
FROM FILM20SD;
```

El \* indica : todo. En este caso, se refiere a todas las columnas. En otro escenarios, se refiere a todos los registros de toda la tabla, y en otras ocasiones se refiere a todos los registros de un resultado producto de una condición.



Edit Results Performance Analysis Logs								
Drag a column header here to group by that column								
FILM_ID	CATEGORY	TITLE	RELEASE_YEAR	LANGUAGE	RENTAL	RENTAL_RATE	LENGTH	
1	11	AFFAIR PREJUDICE	2,010	1	5	2.99	1	
2	11	AIRPORT POLLOCK	2,016	6	6	4.99		
3	11	ALABAMA DEVIL	2,011	3	3	2.99	1	
4	11	ALI FOREVER	2,008	1	4	4.99	1	
5	16	BASIC EASY	2,007	3	4	2.99		
6	1	CAMPUS REMEMBER	2,014	3	5	2.99	1	
7	15	CARIBBEAN LIBERTY	2,015	5	3	4.99		
8	2	CAROL TEXAS	2,006	2	4	2.99	1	
9	1	CASUALTIES ENCINO	2,013	1	3	4.99	1	
10	9	CATCH AMISTAD	2,016	5	7	0.99	1	
11	6	DELIVERANCE MULHOLLAND	2,006	4	4	0.99	1	
12	16	DESPERATE TRAINSPOTTING	2,013	3	7	4.99		
13	4	DETECTIVE VISION	2,013	5	4	0.99	1	

Arriba los resultados también muestran todas las columnas y todos los registros.

A continuación , iremos explicando a medida que probemos los queries siguientes:

```
SELECT FILM_ID, TITLE,RELEASE_YEAR
FROM FILM20SD;
```

Muestra todos los registros pero solo las columnas FILM\_ID, TITLE,RELEASE\_YEAR . Podemos traducir la instrucción : seleccionar(select) las columnas FILM\_ID, TITLE,RELEASE\_YEAR desde (From) la tabla FILM20SD.

```
SELECT FIRST 10 FILM_ID, TITLE from FILM20SD;
```

En este caso seleccionar ordena dos cosas:

1.Seleccionar los primeros 10 registros. Dado que no se sabe a partir de cuál registro, el programa establece por defecto que es a partir del registro 0( el primer registro). First significa: los primeros X registros a partir de....Si First es 5, se refiere los primeros 5 registros a partir de...o de los resultados de...Si First es 50, se refiere a los 50 primeros registros a partir de ...o los resultados de...

2.Seleccionar las columnas FILM\_ID, TITLE desde la tabla FILM20SD.

FILM_ID	TITLE
1	AFFAIR PREJUDICE
2	AIRPORT POLLOCK
3	ALABAMA DEVIL
4	ALI FOREVER
5	BASIC EASY
6	CAMPUS REMEMBER
7	CARIBBEAN LIBERTY
8	CAROL TEXAS
9	CASUALTIES ENCINO
10	CATCH AMISTAD

```
SELECT skip 10 FILM_ID, TITLE from FILM20SD;
```

En este caso seleccionar ordena dos cosas:

1.La selección (select) ordena saltar(skip) 10 registros a partir del primer registro, siempre es a partir del primer registro. Imprime todos los registros a partir de 10.

2.Seleccionar las columnas FILM\_ID, TITLE desde la tabla FILM20SD.

FILM_ID	TITLE
11	DELIVERANCE MULHOLLAND
12	DESPERATE TRAINSPOTTING
13	DETECTIVE VISION
14	DIARY PANIC
15	EFFECT GLADIATOR
16	EGG IGBY
17	ELF MURDER
18	ENOUGH RAGING
19	EVOLUTION ALTER

```
SELECT FIRST 10 SKIP 5 FILM_ID, TITLE from FILM20SD;
```

En este query tenemos la combinación de la instrucción First y Skip.

La selección(select) ordena que a partir del registro 5 (skip 5) se vean los primeros 10 registros( First 10) desde(from) la tabla FILM20SD.

FILM_ID	TITLE
6	CAMPUS REMEMBER
7	CARIBBEAN LIBERTY
8	CAROL TEXAS
9	CASUALTIES ENCINO
10	CATCH AMISTAD
11	DELIVERANCE MULHOLLAND
12	DESPERATE TRAINSPOTTING
13	DETECTIVE VISION
14	DIARY PANIC
15	EFFECT GLADIATOR

Por tanto, con First y Skip, podemos establecer intervalos de registros basados en su índice de posición en la tabla recordando que los índices inician a partir de 0.

Si usamos First y Skip combinados, es importante el orden. Primero First y luego Skip.

**Ahora pasamos a las opciones de ordenar los datos, lo cual es muy simple y solo es cuestión de ejecutar el query y ver los resultados.**

```
SELECT FILM_ID, TITLE from FILM20SD ORDER BY TITLE ASC;
```

En este query, se ordenarán los registros por orden ascendente de Title, o sea, por títulos, empezando por los que inician por la letra A. Si la columna fuera numérica, entonces se ordenaría de menor a mayor.

```
SELECT FILM_ID,CATEGORY_ID,TITLE,RELEASE_YEAR
FROM FILM20SD ORDER BY RELEASE_YEAR ASC, TITLE DESC, CATEGORY_ID ASC;
```

Entramos a una opción muy importante de la consulta de base de datos, se refiere a la selección condicionada mediante la instrucción **where** y los operadores de comparación: =,>,< etc y los de lógica: and, or ,not etc...

```
SELECT FILM_ID,CATEGORY_ID,TITLE,RELEASE_YEAR
FROM FILM20SD WHERE RELEASE_YEAR =2006;
```

En este query se traen los registros que cumplen que su columna RELEASE\_YEAR contenga el valor igual a 2006.

```
SELECT FILM_ID,CATEGORY_ID,TITLE,RELEASE_YEAR
FROM FILM20SD WHERE RELEASE_YEAR =2006 AND CATEGORY_ID > 3;
```

En este query se traen los registros que cumplen que su columna RELEASE\_YEAR contenga el valor igual a 2006 y además cuya columna CATEGORY\_ID sea mayor que 3.

**Ahora prueba todas estas combinaciones:**

```
SELECT FILM_ID,CATEGORY_ID,TITLE,RELEASE_YEAR ,LANGUAGE_ID,RENTAL_RATE,LENGTH
FROM FILM20SD WHERE (CATEGORY_ID =4 AND RELEASE_YEAR> 2012)OR(LANGUAGE_ID =1 AND LENGTH >=150)
ORDER BY RELEASE_YEAR, TITLE;
```

```
SELECT FILM_ID, TITLE,RELEASE_YEAR
FROM FILM20SD WHERE TITLE LIKE '%DET%';
```

```
SELECT FILM_ID, TITLE,RELEASE_YEAR
FROM FILM20SD WHERE TITLE containing 'DE' ORDER BY RELEASE_YEAR, FILM_ID;
```

```
SELECT FILM_ID, TITLE,RELEASE_YEAR
FROM FILM20SD WHERE NOT RELEASE_YEAR =2016;
```

```
SELECT FILM_ID, TITLE,RELEASE_YEAR
FROM FILM20SD WHERE TITLE IS NULL;
```

En este caso, aplica para registros que contienen valores nulos(no hay).

```
SELECT *
FROM FILM20SD WHERE TITLE starting WITH 'D';
```

## Agrupamiento y operaciones de Agregación .

El agrupamiento permite particionar en grupos de filas o registros y hacer operaciones sobre cada grupo de registros. Veamos un ejemplo:

```
SELECT CATEGORY_ID, COUNT(*) AS Cant
FROM FILM20SD GROUP BY CATEGORY_ID;
```

El resultado es:

CATEGORY_ID	CANT
1	2
2	1
4	1
6	2

7	1
8	1
9	1
11	4
12	1
15	2
16	3

Arriba lo que ha ocurrido es que se cuentan la cantidad de filas que existen para cada categoría. Por ejemplo, para la categoría 1, hay 2 registros, para la categoría 11 hay 4 registros, y esto lo podemos comprobarlo con el siguiente select para el caso de categoría 11:

**SELECT \***

**FROM** FILM20SD **where** CATEGORY\_ID=11;

FILM_ID	CATEGORY	TITLE	RELEASE_Y	LANGUAGE	RENTAL_D	RENTAL_RA
1	11	AFFAIR PREJUDICE	2,010	1	5	
2	11	AIRPORT POLLOCK	2,016	6	6	
3	11	ALABAMA DEVIL	2,011	3	3	
4	11	ALI FOREVER	2,008	1	4	

Arriba vemos que aparecen 4 registros que cumplen con la condición que pertenecen a la categoría= 11.

Qué significa **AS**? permite dar un nombre a una columna para que no aparezca el nombre por defecto que asigna el programa a las columnas que son producto de operaciones o cálculos.

La instrucción COUNT(\*) cuenta la cantidad de registros, en este caso se aplica a cada grupo de registros que están agrupados por categorías. Para el caso de categoría 11 , el Count se aplica a todos los registros de dicha categoría que son 4, por tanto el Count arroja el resultado de 4.

El \* significa todos los registros, y en caso de agrupamientos, se refiere a todos los registros de cada agrupamiento, en este caso, por cada categoría.

Ahora bien, vemos que al ejecutar :

**SELECT** CATEGORY\_ID, COUNT(\*) **AS** Cant

**FROM** FILM20SD **GROUP BY** CATEGORY\_ID;

aparecen solo dos columnas, la columna de Categoría y la columna procesada producto de Count. Esto es porque solo deben agregarse al select aquellas columnas que se usan para agrupar, en este caso CATEGORY\_ID , y el resto de las columnas son aquellas producto de Agregación, o sea, son columnas producto de operaciones como sum() o count() entre otras tantas.

Veamos el siguiente ejemplo:

**SELECT** CATEGORY\_ID, COUNT(\*) **AS** CANT, SUM(Length) **AS** SUMLONGITUD, AVG(RENTAL\_DURATION) **AS** PROMDURACON

**FROM** FILM20SD **GROUP BY** CATEGORY\_ID;

El resultado es:

CATEGORY_ID	CANT	SUMLONGITUD	PROMDURACION
1	2	346	4
2	1	151	4
4	1	143	4
6	2	167	4
7	1	107	7
8	1	107	6
9	1	183	7
11	4	435	4
12	1	155	4
15	2	266	4
16	3	329	6

Aparece la columna CATEGORY\_ID, pero hay tres columnas más que no son de la tabla, pero son las columnas agregadas producto de operaciones aplicadas al conjunto de filas o registros para cada grupo de categoría.

Por ejemplo, la categoría 11 tenemos que en su columna Cant aparece el valor 4, esto es porque la categoría 11 tiene 4 filas o registros que cumplen que son de categoría 11. Luego tenemos la columna SUMLONGITUD, lo que hace es sumar los valores de la columna Length de las 4 filas que cumplen que son de las categoría 11. La columna PROMDURACION calcula el promedio de la columna Rental\_Duration para las 4 filas que cumplen que son de las categoría 11.

Lo de arriba se aplica para cada categoría.

Veamos el análisis de las 4 filas para la categoría 11 para comparar con el resultado del query de agrupamiento (subrayado en celeste arriba en la imagen de la tabla).

**SELECT** CATEGORY\_ID, LENGTH, RENTAL\_DURATION

**FROM** FILM20SD **where** CATEGORY\_ID=11;

CATEGORY_ID	LENGTH	RENTAL_DURATION
11	117	5
11	54	6
11	114	3
11	150	4
	435	4.5

Arriba copiamos a excell la tabla de resultados producto del select para CATEGORY\_ID=11 e hicimos la sumatoria para los 4 registros de la columna LENGTH y el promedio de los 4 registros de la columna RENTAL\_DURATION.

Y comparemos con:

CATEGORY_ID	CANTIDADREGISTROS	SUMLONGITUD	PROMDURACION
11	4	435	4

Vemos que la suma de la columna LENGTH es 435 y coincide con el registro producto del agrupamiento, en cambio el promedio de las 4 filas en la columna RENTAL\_DURATION nos da 4.5, pero en la tabla de agrupamiento aparece 4 ¿por qué? . Esto es debido a que la columna RENTAL\_DURATION es del tipo Integer(número entero), y al aplicar una operación de agregación, nos producirá una columna también del tipo entero, por tanto lo que hace la operación es truncar la parte decimal y solo muestra la entera. Para que esto no suceda, debo indicar a la operación de AVG que sea del tipo decimal, de esta manera:



```

SELECT CATEGORY_ID, COUNT(*) AS CantidadRegistros, SUM(Length) AS SumLongitud,
AVG(CAST(RENTAL_DURATION AS decimal(5, 2))) AS PromDuracion
FROM FILM20SD GROUP BY CATEGORY_ID;

```

Hemos introducido la siguiente diferencia:

**AVG(CAST(RENTAL\_DURATION AS decimal(5, 2)))** la instrucción **CAST(RENTAL\_DURATION AS decimal(5, 2))** convierte el valor de RENTAL\_DURATION a decimal de 5 dígitos de los cuales 2 son para decimales, por ejemplo: 234.60 , son 5 dígitos de los cuales dos son decimales.

Después de convertir el contenido mediante CAST, se aplica la operación AVG, y el resultado esta vez será como un decimal de 5 dígitos de los cuales 2 son para decimales.

Las operaciones de agregación están directamente vinculadas con el agrupamiento o instrucción GROUP BY. Son aquellas que se refieren a: sum(),count(),avg()->promedio, min() o max() entre otras tantas aplicadas a un grupo de filas o registros que son una parte de un agrupamientos

```

SELECT RENTAL_DURATION,CATEGORY_ID, COUNT(*) Cant,SUM(RENTAL_RATE) AS Rate,AVG(RENTAL_DURATION) AS
Duration

```

```

FROM FILM20SD GROUP BY RENTAL_DURATION, CATEGORY_ID;

```

Arriba vemos un agrupamiento, que al referirse para dos columnas, estas deben aparecer también en el select. A partir de las dos columnas (una agrupada dentro de la otra) realizas cálculos de agregamiento, como: cantidad, sumas o promedios, aplicados a la cantidad de registros o filas que pertenecen a cada agrupamiento más interno, en este caso CATEGORY\_ID. Si por ejemplo tenemos un agrupamiento para CATEGORY\_ID = 6, y sabemos que hay 2 registros que cumplen esa condición, entonces aplica el count a esos 2 dando 2 por supuesto, y también la suma de esos 2 y el promedio de esos 2. A esas operaciones, les puedes poner un nombre en su columna, es un alias definido por la instrucción AS.

También puedes agregar WHERE pero después del FROM y antes del GROUP BY, y si quieres aplicar un where a los resultados de GROUP, entonces usas HAVING, que es un WHERE pero exclusivamente para GROUP, y va después de GROUP.

También puedes aplicar por supuesto ORDER BY.

```

SELECT RENTAL_DURATION,CATEGORY_ID, COUNT(*) Cant,SUM(RENTAL_RATE) AS SumRate,AVG(RENTAL_DURATION)
AS Duration

```

```

FROM FILM20SD GROUP BY RENTAL_DURATION, CATEGORY_ID ORDER BY CATEGORY_ID;

```

```

SELECT SELECT RENTAL_DURATION,CATEGORY_ID, COUNT(*) Cant,SUM(RENTAL_RATE) AS
SumRate,AVG(RENTAL_DURATION) AS Duration,MIN(RENTAL_DURATION) AS minDuration, MAX(RENTAL_DURATION)
AS maxDuration
FROM FILM20SD GROUP BY RENTAL_DURATION, CATEGORY_ID ORDER BY CATEGORY_ID;

```

## Agregar nuevo registro o fila a la tabla de la base de datos.

```
INSERT INTO CATEGORY (CATEGORY_ID, NAME)
VALUES (17, 'Neuro');
```

Se puede traducir de la siguiente manera: insertar(**insert**) valores dentro(**into**) de la tabla Category en los campos que están dentro de los paréntesis: CATEGORY\_ID y NAME, los valores(**values**) que aparecen en el paréntesis en el mismo orden .

Al final de la tabla, veremos una nueva fila o registro

CATEGORY\_ID= 17

NAME= 'NEURO'

CATEGORY_ID	NAME	
0	name	
1	Action	
2	Animation	
3	Children	
4	Classics	
5	Comedy	
6	Documentary	
7	Drama	
8	Family	
9	Foreign	
10	Games	
11	Horror	
12	Music	
13	New	
14	Sci-Fi	
15	Sports	
16	Travel	
17	NEURO	←-----

Podemos verificar al correr: **SELECT** CATEGORY\_ID, NAME **FROM** CATEGORY ;

**Veamos el siguiente modo, que además de insertar, te devuelve los valores de la última fila insertada, algo muy usado en programación para conocer los valores de aquellas columnas(por ejemplo auto numéricas) que se asignan valores desde el Firebird en vez de las agregadas por nosotros.**

**INSERT INTO**

CATEGORY

(

CATEGORY\_ID,

NAME

)

**VALUES (**

17,

'NEURO'

)

**RETURNING** CATEGORY\_ID, NAME;

### Eliminar un registro:

Para eliminar, es parecido, pero debemos seleccionar la fila que deseamos eliminar usando la columna cuyos valores identifican de forma única al registro, de tal forma, que se elimine un registro y no otros.

En este caso vamos a eliminar el mismo registro que hemos agregado nuevo:

**DELETE FROM**

CATEGORY

**WHERE** CATEGORY\_ID= 17;

La instrucción ordena borrar desde la tabla CATEGORY todas las filas donde su campo CATEGORY\_ID sea igual a 17. Como solo existe una sola fila que cumple dicha condición, quedará eliminada.

Podemos verificarlo al correr :

**SELECT** CATEGORY\_ID, NAME **FROM** CATEGORY ;

A continuación mostramos otras formas de **DELETE** pero con otras tablas, solamente con la idea que se observen las posibilidades de trabajar con **DELETE**:

DELETE FROM Cities where name starting 'Alex';

DELETE FROM Cities where Cities.name starting 'Alex';

DELETE FROM Cities C where C.name starting 'Alex';

Aquí podemos observar un técnica muy usada, se asigna un alias a la tabla, en este caso asignamos la C en sustitución del nombre de la tabla llamada Cities, y usamos C para utilizar los campos de dicha tabla. Es como cambiar Cities por C.

DELETE FROM People


WHERE firstname <> 'Boris' AND lastname <> 'Johnson';

**Se pueden eliminar más de un registro?** si, por ejemplo, eliminar todos aquellos registros cuyo campo CATEGORY\_ID es 1, si hay 5 de estos registros que cumplen que CATEGORY\_ID es 1, quedarán eliminados. Pero hay que ser muy cuidadoso, si usamos un Where mal pensado o dejamos algo así:

DELETE CATEGORY sin un where, se eliminarán todos los registros de la tabla, quedando vacía.

## **Modificar** los datos existente de una fila o registro, en todas o en algunas de sus columnas.

Veamos el contenido del registro cuyo CATEGORY\_ID=17 :

11	Horror
12	Music
13	New
14	Sci-Fi
15	Sports
16	Travel
17	NEURO 

Vamos a modificar el valor de su columna Name.

### **UPDATE**

CATEGORY

### **SET**

NAME = 'Manga'

**WHERE** CATEGORY\_ID=17;

La instrucción nos manda actualizar(update) o modificar(editar) la tabla CATEGORY asignando (set) el valor "Manga" a la columna Name, donde(where) la fila tenga su campo CATEGORY\_ID= 17. Usa apóstrofe ( ' ) en vez de comillas ( " ). También mucho cuidado con las apóstrofes o comillas de estilos diferentes, pues no las reconoce el lenguaje SQL en el programa.

Queda así:

CATEG	NAME
1	Action
2	Animation
3	Children
4	Classics
5	Comedy
6	Documentary
7	Drama
8	Family
9	Foreign
10	Games
11	Horror
12	Music
13	New
14	Sci-Fi
15	Sports
16	Travel
17	Manga

Otros ejemplos pero de otras tablas:

```
UPDATE addresses
SET city = 'Saint Petersburg', citycode = 'PET'
WHERE city = 'Leningrad';
```

```
UPDATE employees
SET salary = 2.5 * salary
WHERE title = 'CEO';
```

```
UPDATE employees
SET salary = salary + 50
ORDER BY salary ASC
ROWS 20;
```

Arriba limita a 20 filas la cantidad procesada por la instrucción, Si por ejemplo, el update afecta a 100 registros, con ROWS limita a 20 afectados, y como existen un ORDER que ordena de menos a mayor salarios, indica que los primeros 20 son los afectados según ese orden.

```
UPDATE Scholars
SET firstname = 'Hugh', lastname = 'Pickering'
WHERE firstname = 'Henry' and lastname = 'Higgins'
RETURNING id, old.lastname, new.lastname;
```

## La mezcla de columnas mediante JOIN:

Observemos la imagen de abajo :

FILM_ID	CATEGORY_ID	TITLE	RELEASE_YEAR	CATEGORY_ID	NAME
1	11	AFFAIR PREJUDICE	2010	1	Action
2	11	AIRPORT POLLOCK	2016	2	Animation
3	11	ALABAMA DEVIL	2011	3	Children
4	11	ALI FOREVER	2008	4	Classics
5	16	BASIC EASY	2007	5	Comedy
6	1	CAMPUS REMEMBER	2014	6	Documentary
7	15	CARIBBEAN LIBERTY	2015	7	Drama
8	2	CAROL TEXAS	2006	8	Family
9	1	CASUALTIES ENCINO	2013	9	Foreign
10	9	CATCH AMISTAD	2016	10	Games
11	6	DELIVERANCE MULHOLLAND	2006	11	Horror
12	16	DESPERATE TRAINSPOTTING	2013	12	Music
13	4	DETECTIVE VISION	2013	13	New
14	7	DIARY PANIC	2016	14	Sci-Fi
15	8	EFFECT GLADIATOR	2013	15	Sports
16	6	EGG IGBY	2012	16	Travel
17	12	ELF MURDER	2007	17	Manga
18	16	ENOUGH RAGING	2006		
19	15	EVOLUTION ALTER	2008		

FILM_ID	CATEGORY_ID	NAME	TITLE	RELEASE_YEAR
1	11	Horror	AIR PREJU	2010
2	11	Horror	ORT POLL	2016
3	11	Horror	BAMA DE	2011
4	11	Horror	LI FOREVE	2008
5	16	Travel	SIC EAS	2007
6	1	Action	PUS REME	2014
7	15	Sports	BBEAN LIB	2015
8	2	Animation	AROL TEXA	2006
9	1	Action	ALTIES EN	2013
10	9	Foreign	ICH AMIST	2016

La tabla a la izquierda (viendo de usted hacia el documento) es FILM20SD, y observamos que existe la columna CATEGORY\_ID que contiene un número que identifica la categoría de la película, pero no podemos conocer el nombre en lenguaje humano de dicha categoría. En cambio, en la tabla a la derecha, la tabla CATEGORY, nos muestra el número de la categoría y el nombre de la categoría asignado a dicho número. Por tanto, la idea es producir otra tabla que incluya el nombre de la categoría, lo cual se puede ver en la tabla debajo de las otras dos. Vemos que hay campos de la tabla FILM20SD y un campo de la tabla CATEGORY.

Esta mezcla se hace mediante la instrucción JOIN, y tenemos que considerar varias cosas.

Puede ser que en la tabla FILM20SD tengamos un valor de CATEGORY\_ID que no existe en la tabla CATEGORY, por ejemplo CATEGORY\_ID=25. O al revés, puede que exista una categoría en la tabla CATEGORY que no exista en la tabla FILM20SD. Todo esto nos obliga a escoger el tipo de JOIN que deseamos.

Veamos lo que podemos pedir:

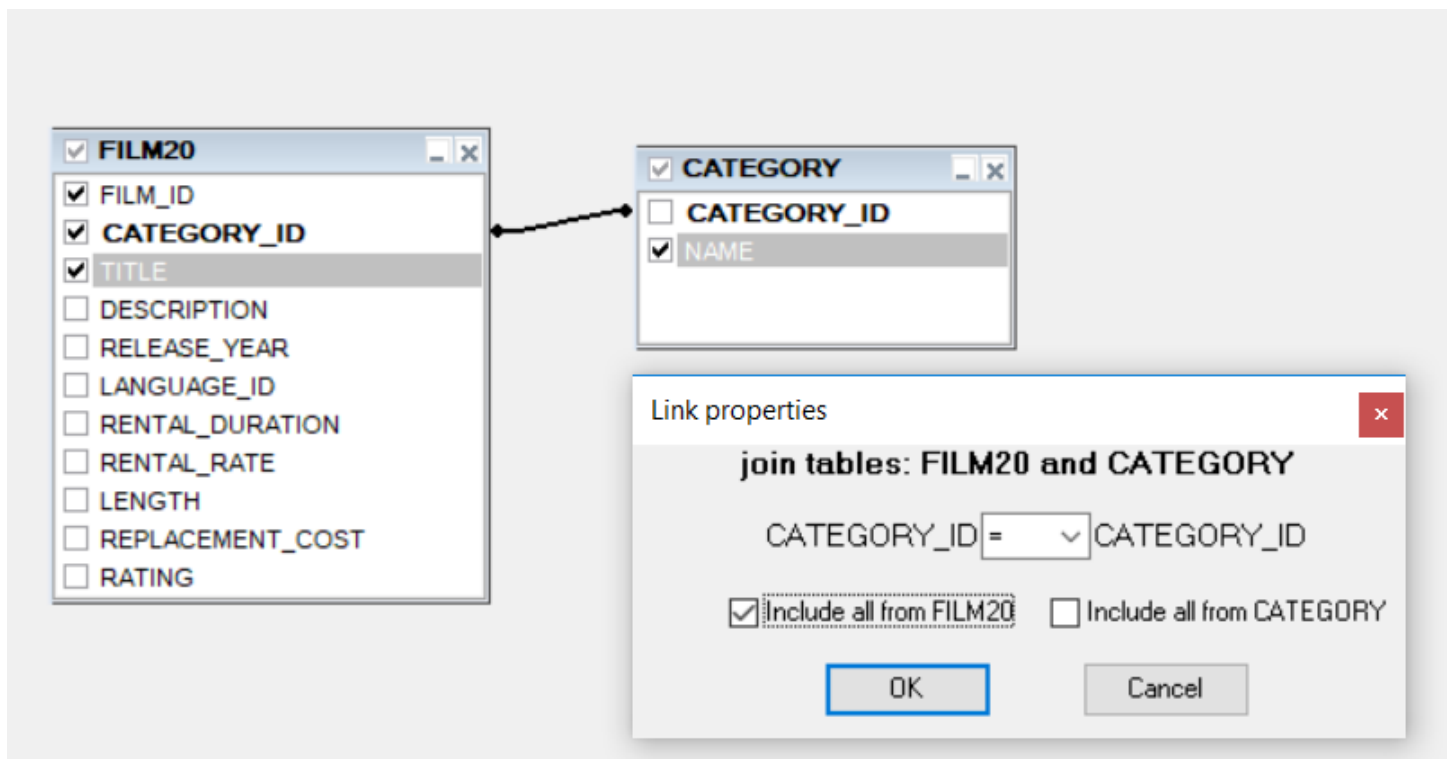
Que muestre aquellos registros de FILM20SD y CATEGORY que se pueden relacionar. O sea, aquellos registros de FILM20SD que no se les puede asignar un nombre de categoría ... no aparecerán, o sea, son todos aquellos valores de la

columna CATEGORY\_ID de FILM20SD que no aparecen en la misma columna CATEGORY\_ID de CATEGORY., por ejemplo CATEGORY\_ID de FILM20SD = 100 no existe ese valor en la columna CATEGORY\_ID de la tabla CATEGORY. Y aquellos registros de CATEGORY cuyo valor de CATEGORY\_ID no existe en FILM20SD... tampoco aparecerán en los resultados, por ejemplo que exista el valor CATEGORY\_ID= 250 de la tabla CATEGORY, no existe dicho valor en la columna CATEGORY\_ID de la tabla FILM20SD. Esto garantiza que los resultados mezclados (joinados) , o sea, una nueva tabla con todas o algunas columnas de las dos tablas, no contengan registros con columnas que contengan valores nulos.

Que muestre todos los registros de la tabla FILM20SD tengan o no categorías en la tabla CATEGORY (por tanto el campo NAME aparecerá con valor Nulo en la nueva tabla mezclada) y que aparezcan solo los registros de CATEGORY que se pueden relacionar (o aportar información) con la tabla FILM20SD. En este caso entonces, aparecerán todas las filas de FILM20SD y solo aquellas filas de CATEGORY cuyo CATEGORY\_ID existe en FILM20SD. Esto implica que pueden salir valores nulos en la columna NAME.

Que muestre todos los registros de tabla FILM20SD y todos de CATEGORY, tengan valores o no que los ligen a ambos. Saldrán posiblemente valores nulos si hay valores en uno diferentes a las del otro en la columna CATEGORY\_ID.

Veamos abajo de forma gráfica las posibilidades de un JOIN:



Arriba vemos que relacionamos la tabla mediante un campo común, en este caso CATEGORY\_ID. Y el programa nos da la posibilidad de establecer el tipo de relación del JOIN: incluir todos los registros de FILM20S solamente lo cual implica que se incluyen solamente los registros que aportan datos de CATEGORY, o se incluyen todos los registros de CATEGORY pero de FILM20SD lo cual significa que se incluyen en los resultados solo aquellos que pueden ser llenados por CATEGORY y los demás los excluye. O Se hace check en ambas tablas, y por tanto se incluyen todos los registros de ambos aporten datos o no uno al otro, o bien dejamos ambos en blanco lo cual significa que en los resultados se incluirán los registros de ambas tablas que pueden ser llenados con la información del otro, por tanto aquel registro de FILM20SD que no se le puede asignar un nombre porque un valor de su campo CATEGORY\_ID no existe en la tabla CATEGORY no será incluido dicho registro de FILM20SD en los resultados, y aquel registro de CATEGORY cuyo valor del campo CATEGORY\_ID no aparece en la columna CATEGORY\_ID de la tabla FILM20SD tampoco será incluido en los resultados.

## Sobre Join:

<https://firebird21.wordpress.com/2013/07/21/entendiendo-a-los-join/>

<https://firebird21.wordpress.com/2013/12/02/inner-join-y-outer-join/>

<https://firebird21.wordpress.com/2014/04/13/join-implicit-y-join-explicit/>

**Aclaración:** los resultados no son una tabla, son una vista, que se puede explicar como una tabla que solo existe en memoria, y después que se apaga la computadora, desaparece y no se conserva en el disco duro para su uso posterior.

La sintaxis del JOIN arriba explicado es:

### SELECT

```
FILM20.FILM_ID,  
FILM20.CATEGORY_ID,  
CATEGORY.NAME,  
FILM20.TITLE
```

### FROM

```
FILM20
```

```
LEFT OUTER JOIN CATEGORY ON (FILM20.CATEGORY_ID = CATEGORY.CATEGORY_ID)
```

Podemos traducir a lenguaje humano de la siguiente forma: seleccionar(**select**) los campos: FILM\_ID de la tabla FILM20, Category\_ID de la tabla FILM20, NAME de la tabla CATEGORY y TITLE de la tabla FILM20 desde(**from**) la tabla FILM20 (es la que está a la izquierda y a la que relaciones las demás) haciendo un JOIN por la izquierda (**LEFT OUTER JOIN**) con la tabla CATEGORY en una relación(**ON**) que iguale FILM20.CATEGORY\_ID = CATEGORY.CATEGORY\_ID

Lo anterior implica que debemos escoger con cuidado el campo que hará la relación entre ambas tablas, ambos campos deben ser del mismo tipo, o sea, si es entero uno, debe ser entero otro, no puede ser que uno sea entero y otro cadena. Ambos campos deben referirse a lo mismo, y seguramente debe coincidir sus valores en ambas tablas.

Otros ejemplos:

### SELECT

```
FILM20SD.FILM_ID,  
FILM20SD.CATEGORY_ID,  
CATEGORY.NAME,  
FILM20SD.TITLE,  
FILM20SD.RELEASE_YEAR,  
FILM20SD.LANGUAGE_ID,  
FILM20SD.RENTAL_DURATION,  
FILM20SD.RENTAL_RATE,  
FILM20SD."LENGTH",
```



```
FILM20SD.REPLACEMENT_COST,  
FILM20SD.RATING  
  
FROM  
  
FILM20SD  
  
LEFT OUTER JOIN CATEGORY ON (FILM20SD.CATEGORY_ID = CATEGORY.CATEGORY_ID)
```

A continuación utilizamos una técnica que asigna una alias al nombre de una tabla. En nuestro caso asignamos la letra F a FILM20SD, de tal modo , que podemos usar F en vez de FILM20SD.

```
SELECT  
F.FILM_ID,F.CATEGORY_ID,C."NAME",F.TITLE,F.RELEASE_YEAR,F.LANGUAGE_ID,F.RENTAL_DURATION,F.RENTAL_R  
ATE,  
F.REPLACEMENT_COST  
  
FROM FILM20SD F  
  
LEFT JOIN CATEGORY C  
  
ON F.CATEGORY_ID=C.CATEGORY_ID;
```

## **Referencias de estudio:**

### **Blog Firebird SQL:**

Teoría y Práctica sobre Firebird

<https://firebird21.wordpress.com>

Autor: Walter R. Ojeda Valiente

<https://firebird21.wordpress.com/about/>

Sitio completo:

<https://firebird21.wordpress.com/2013/06/16/el-indice-del-blog-firebird21/>

En este sitio, se muestra con mucho detalles y muchos ejemplos en español, operaciones con Firebird. Las secciones que deben leer al menos por encima y tener en cuenta como una fuente de consulta y aprendizaje autónomo, son:

Ejemplos de SELECTs para principiantes

<https://firebird21.wordpress.com/category/principiantes/ejemplos-de-selects-para-principiantes/>

Ejemplos de SELECTs para intermedios

<https://firebird21.wordpress.com/category/ejemplos-de-selects-para-intermedios/>

Funciones agregadas

<https://firebird21.wordpress.com/category/funciones/funciones-agregadas/>

INSERTs y UPDATEs

<https://firebird21.wordpress.com/category/inserts-y-updates/>

### **Sobre Join:**

<https://firebird21.wordpress.com/2013/07/21/entendiendo-a-los-join/>

<https://firebird21.wordpress.com/2013/12/02/inner-join-y-outer-join/>

<https://firebird21.wordpress.com/2014/04/13/join-implicit-y-join-explicito/>

### **Otros ejemplos para prácticas:**

Ejemplo N° 005 - Usando LEFT JOIN

<https://firebird21.wordpress.com/2013/05/08/ejemplo-no-005-usando-left-join/>

Ejemplo N° 006 - Usando LEFT JOIN e INNER JOIN

<https://firebird21.wordpress.com/2013/05/08/ejemplo-no-006-usando-left-join-e-inner-join/>

Los predicados de comparación

<https://firebird21.wordpress.com/2014/04/27/los-predicados-de-comparacion/>

Usando SIMILAR TO

<https://firebird21.wordpress.com/2014/04/27/usando-similar-to/>

Optimizando los JOIN

<https://firebird21.wordpress.com/2014/05/14/optimizando-los-join/>

Paginando un SELECT

<https://firebird21.wordpress.com/2014/05/29/paginando-un-select/>

Un error de concepto en la cláusula WHERE

<https://firebird21.wordpress.com/2014/09/14/un-error-de-concepto-en-la-clausula-where/>

Poniendo los JOIN en el orden correcto

<https://firebird21.wordpress.com/2015/07/31/poniendo-los-join-en-el-orden-correcto/>

Consultando datos que NO EXISTEN en una tabla

<https://firebird21.wordpress.com/2013/11/29/consultando-datos-que-no-existen-en-una-tabla/>

Entendiendo la cláusula GROUP BY: agrupando datos

<https://firebird21.wordpress.com/2015/09/01/entendiendo-la-clausula-group-by-agrupando-datos/>

La cláusula GROUP BY requiere estar ordenada

<https://firebird21.wordpress.com/2015/09/02/la-clausula-group-by-requiere-estar-ordenada/>

La cláusula HAVING: filtrando las filas agrupadas

<https://firebird21.wordpress.com/2015/09/03/la-clausula-having-filtrando-las-filas-agrupadas/>

Obteniendo la primera fila de cada grupo

<https://firebird21.wordpress.com/2015/12/27/obteniendo-la-primer-fila-de-cada-grupo/>

Hallando la primera palabra

<https://firebird21.wordpress.com/2016/05/12/hallando-la-primer-palabra/>

Hallando la última palabra

<https://firebird21.wordpress.com/2016/05/13/hallando-la-ultima-palabra/>