RECUPERACIÓN DE DATOS

SELECT

SELECT → significa Seleccionar, lista de campos

FROM → es Desde, tabla o conjunto de tablas

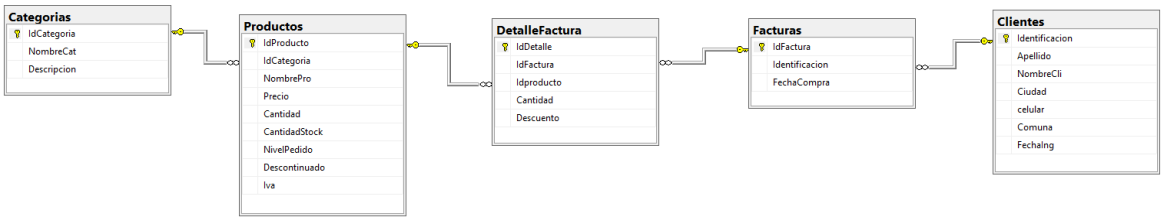
WHERE → condición o condiciones, para extraer los datos de, o las tablas.

DOCUMENTO GUIA

DOCENTE

ALBEIRO MURIEL

1 RECUPERACIÓN DE DATOS SELECT



S E L E C T

En general a las operaciones básicas de manipulación de datos que podemos realizar con SQL se les denomina operaciones **CRUD** (de **C**reate, **R**ead, **U**pdate and **D**elete, o sea, **C**rear, **L**eer, **A**ctualizar y **B**orrar, sería *CLAB en español, pero no se usa*). Lo verás utilizado de esta manera en muchos sitios, así que apréndete ese acrónimo.

1.1 Consulta de datos

Nos concentraremos en la "**R**" de C**R**UD, es decir, en cómo recuperar la información que nos interesa de dentro de una base de datos, usando para ello el lenguaje de consulta o SQL. Para realizar consultas sobre las tablas se utiliza la instrucción ***SELECT***. Con ella se puede *consultar una o varias tablas*. Es sin duda el comando ***más importante*** del lenguaje SQL.

Esta instrucción recupera las filas de una base de datos y habilita la selección de varias filas y columnas de tablas que se encuentran contenidas dentro de una base de datos. El proceso más relevante que podemos llevar a cabo en una base de datos es la consulta de los datos. No serviría de nada contar con una base de datos si no pudiéramos consultarla. Esta es una de las instrucciones más utilizadas en el mundo de las bases de datos.

Estructura básica

SELECT → significa Seleccionar, lista de campos

FROM → es Desde, tabla o conjunto de tablas

WHERE → condición o condiciones, para extraer los datos de, o las tablas.

1.2 Especificaciones de la sentencia

SELECT: Palabra clave y reservada que indica que la sentencia de SQL que se desea ejecutar es de selección.

FROM: Indica la tabla desde la cual se desean recuperar los datos. En el caso de que exista más de una tabla se hace una combinación de tablas usando la instrucción ***JOIN***. En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula *WHERE*.

WHERE: Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas **condiciones**.

1.3 Condiciones

Son expresiones lógicas por comprobar, para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

- (Mayor)
- >= (Mayor o igual)
- < (Menor)
- <= (Menor o igual)
- = (Igual)
- o != (Distinto)
- <> Distinto
- **IS [NOT] NULL** (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)

Se dice que una columna de una fila es **NULL** *si está completamente vacía.* Hay que tener en cuenta que, si se ha introducido cualquier dato, incluso en un campo alfanumérico si se introduce una cadena en blanco o un cero en un campo numérico, deja de ser NULL.

1.4 Estructura básica de un SELECT

SELECT → campo1, campo2, campo3…. campoN

FROM → Nombre de la tabla

WHERE → condición o condiciones, para extraer los datos de, o las tablas.

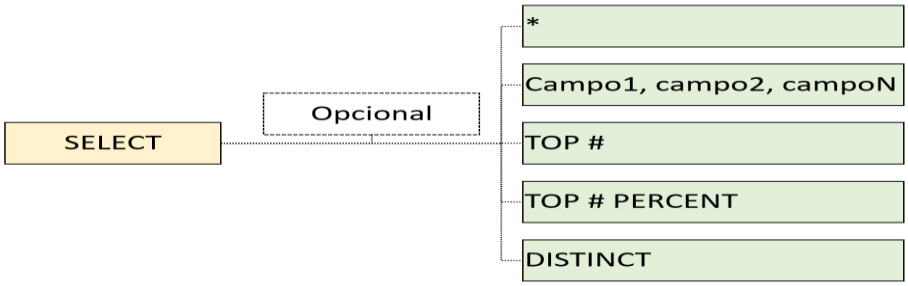
El resultado de una consulta **SELECT** devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene *filas/registros*, con una serie de *campos/columnas.* Igual que cualquier *tabla* de la base de datos. Sin embargo, esta ***tabla está en memoria mientras la utilicemos, y luego se descarta.*** Cada vez que ejecutamos la consulta se vuelve a calcular el resultado

Las instrucciones mínimas para crear una consulta SELECT son:

- SELECT
- FROM

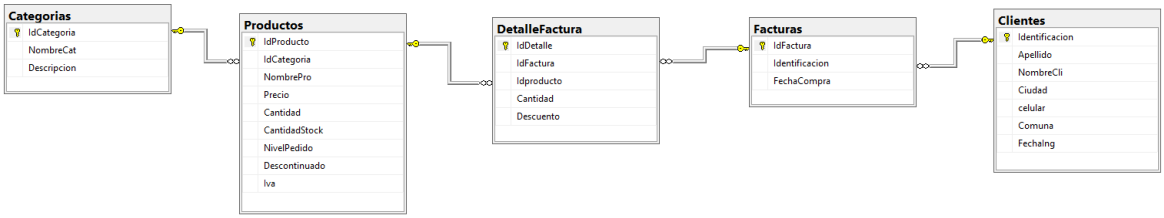
1.4.1 Argumentos opcionales de un SELECT.

- * devuelve todos los datos de la tabla
- Campo1, campo2,…. campoN muesta los nombres solo de esos campos
- TOP # devuelve un número de registros específicos.
- TOP # PERCENT devuelve un porcentaje específico del total de registros de la tabla.
- DISTINCT muestra registros únicos que se especifican en la línea del SELECT



Ejemplos

Para los siguientes ejemplos se utiliza la siguiente base de datos.



SELECT * -- Mostrar todos los datos de la tabla categorías. (10 registros) **FROM** *categorias*
SELECT NombrePro, precio, cantidad -- tabla productos. (59 registros) **FROM** *productos*
SELECT TOP 15 * -- Mostrar los 15 primeros registros. **FROM** *productos*
SELECT TOP 15 PERCENT * -- Mostrar el 15% de los registros de la tabla (9 registros) **FROM** *clientes*
SELECT DISTINCT ciudad -- Mostrar solo la columna ciudad sin repetirlas, registros únicos (8 registros) **FROM** *clientes*

1.4.2 ALIAS… AS

Permite renombrar el nombre de la columna en la cláusula SELECT, o renombrar tablas si se utiliza en la cláusula FROM. Esta cláusula es opcional. Ejemplo.

SELECT o FROM	Campo1 **AS** ‘nombreColumna’, campo2 ‘nombreColumna’…
SELECT	Identificacion **AS** ‘Cédula cliente’
FROM	clientes **AS** socios

2 WHERE

La cláusula **WHERE** puede comparar valores de columnas, expresiones, funciones, listas de valores, constantes, etc.

La condición **WHERE** especifica el filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

Para limitar los registros recuperados por una consulta SQL, se usa los operadores de comparación o los operadores lógicos.

Operadores de comparación

Suponiendo que la variable a tiene un valor de 5 y la variable b tiene un valor de 10.

Operador	Descripción	Ejemplo
=	Compara dos valores	(a = b) es falso.
!=	Compara si son diferentes	(a != b) es verdadero.
<>	Diferente a	(a <> b) es verdadero
>	Mayor que	(a > b) es falso.
<	Menor que	(a < b) es verdadero.
>=	Mayor o igual que	(a >= b) es falso.
<=	Menor o igual que	(a <= b) es verdadero.

Operadores lógicos

Los Operadores lógicos en SQL Server son los siguientes:

Operador	Descripción
AND	Usado para múltiples en expresiones lógicas con más de una condición, para que el resultado final sea verdadero, todas las condiciones deben de ser verdaderas.
BETWEEN	Compara un dato en un rango de valores
EXISTS	Comprueba si existe un dato en un select
IN	Comprueba si un valor está en un conjunto de datos.
LIKE	Permite comparar datos de tipo caracter no exactos.
NOT	Niega el resultado de una o más condiciones.
OR	Usado para múltiples condiciones en la cláusula Where, para que el resultado final sea verdadero, basta que una de las condiciones sea verdadera.
IS NULL	Compara si el valor del campo es Null
UNIQUE	Lista valores sin incluir duplicados

ORDER BY

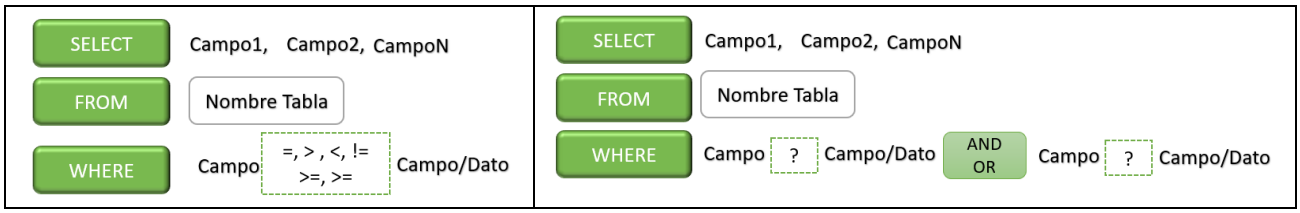
Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

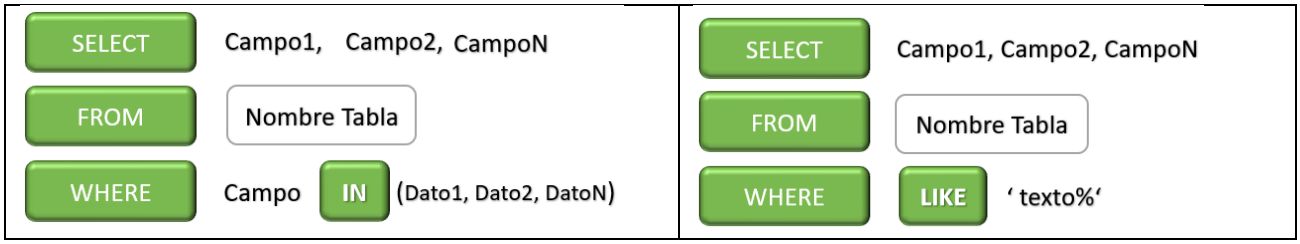
ASC es el valor predeterminado, especifica que la columna indicad en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica **DESC** se ordenará de forma descendente (de mayor a menor).

O también se pude:

***ORDER BY** 2 DESC, 1*

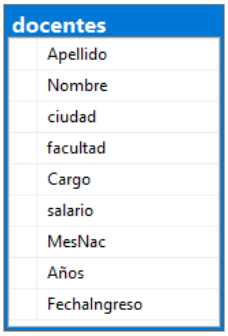
2.1 Sintaxis de la instrucción WHERE





Para los siguientes ejercicios se utilizará la base de datos, DOCENTES

2.2 Ejemplos operadores lógicos



-- Mostrar los datos de los personas que el salario sea mayor o igual a 3 millones

SELECT apellido, nombre, cargo, salario
FROM docentes
WHERE salario >=3000000

-- Mostrar los datos de los personas que viven en Medellín
SELECT apellido, nombre, facultad, salario
FROM docentes
WHERE ciudad =’Medellín’

2.3 Ejemplos operadores lógicos AND - OR

AND → debe cumplir con todas las condiciones

OR → debe cumplir al menos una condición

-- Mostrar los datos de los personas que estén en el rango de los 40 y 50 años
SELECT apellido, nombre, facultad, salario
FROM *docentes*
WHERE años >=40 **AND** años <=50

-- Mostrar los datos de los personas que viven en los ciudades de Bello o Sabaneta
SELECT apellido, nombre, ciudad, cargo, salario
FROM *docentes*
WHERE ciudad = ‘Bello´ **OR** ciudad = ‘Sabaneta’

-- Mostrar los datos de los personas que estén en el rango de salarios entre 3 millones y 5 millones

```
SELECT  apellido, nombre, ciudad, facultad, salario
FROM  docentes
WHERE  salario >= 3000000  AND  salario <= 5000000
```

2.4 Ejemplos con BETWEEN:

Se utiliza para un intervalo de valores. Por ejemplo:

```
-- Mostrar los datos de los personas que estén en el rango de los 40 y 50 años
SELECT  apellido, nombre, ciudad, cargo
FROM  docentes
WHERE  años  BETWEEN 40  AND  50


-- Mostrar los datos de los personas que estén su Fecha Ingreso se encuentre entre el 1/1/95 y 31/12/05
SELECT  apellido, nombre, facultad, cargo
FROM  docentes
WHERE  años  BETWEEN 40  AND  50
```

2.5 IN( )

Seleccionar varios datos en la misma columna.

para especificar una relación de valores concretos. Por ejemplo:

```
SELECT apellido, nombre, ciudad, salario
FROM docentes
WHERE ciudad IN('bello','medellín')
```

La negación de IN

```
SELECT ciudad, salario
FROM docentes
WHERE ciudad NOT IN('bello','medellín')
```

2.6 Like

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL.

LIKE: para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: "%" y "_". Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda. Por ejemplo:

- El nombre empieza por A: Nombre LIKE 'A%'
- El nombre acaba por A: Nombre LIKE '%A'
- El nombre contiene la letra A: Nombre LIKE '%A%'
- El nombre empieza por A y después contiene un solo carácter cualquiera: Nombre LIKE 'A_'
- El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: Nombre LIKE 'A_E%'

```
-- Mostrar los datos de las personas que su apellido comience con la letra A
SELECT  apellido, nombre, ciudad, cargo, salario
FROM  docentes
WHERE  apellido LIKE 'a%'
```

```
SELECT  apellido, nombre, ciudad, cargo, salario
FROM  docentes
ORDER BY apellido, Nombre DESC
```

2.7 Columnas calculadas

Una columna calculada es una columna virtual que no está almacenada físicamente en la tabla, a menos que la columna esté marcada con PERSISTED. Las expresiones de columnas calculadas pueden utilizar datos de otras columnas al calcular un valor para la columna a la que pertenecen.

Limitaciones y restricciones

- Una columna calculada no puede utilizarse como definición de restricción DEFAULT o FOREIGN KEY ni como NOT NULL.
- Una columna calculada no puede ser el destino de una instrucción INSERT o UPDATE.

Ventajas

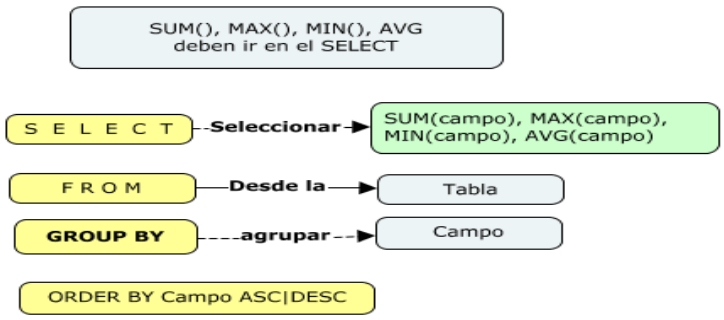
- Las fórmulas en las columnas calculadas son muy similares a las fórmulas en Excel.
- Una columna calculada está basada en datos ya incluidos en una tabla existente, o que se crean mediante una fórmula. Por ejemplo, podría decidir concatenar los valores, realizar la multiplicación, división, suma, resta, etc. extraer las subcadenas o comparar los valores de otros campos.

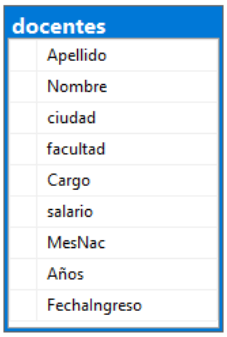
Los valores de las columnas calculadas se "calculan" cada vez que se consulta la tabla, a no ser que se utilice la cláusula **PERSISTED** consiguiendo de esta manera almacenar físicamente el valor de la columna calculada. Así pues, *usaremos columnas calculadas sin almacenamiento físico* cuando el número de consultas sea *muy limitado* y utilizaremos *PERSISTED cuando la información vaya a ser muy consultada*, aunque nos penalice el aumento de espacio físico.

Ejercicios

```
-- Unir las columnas Apellido y Nombre, para que se visualicen en una sola columna.
SELECT CONCAT(Apellidos,' ',Nombres) AS nombre_completo
FROM Alumno

-- Otro método con el mismo resultado
SELECT (apellido + '  ' + Nombre) AS 'Nombre Completo'
FROM docentes

-- Calcular el 4% del salario del docente para pago de EPS
SELECT apellido, nombre, salario, salario * 4/100 AS ‘Descuento EPS’
FROM docentes

-- colocar formato de numero a las columnas
SELECT FORMAT(vlrmat, '#,###') AS 'Vlr Matricula' , FORMAT((VlrMat * 10/100),'$, #,###') AS 'recargo'
FROM Matricula
```

3 CONSULTAS DE AGRUPACION





Para los siguientes ejercicios se utilizará la base de datos, DOCENTES

3.1 GROUP BY

La instrucción **GROUP BY** se usa a menudo con funciones agregadas (**COUNT, MAX, MIN, SUM, AVG**) para agrupar el conjunto de resultados por una o más columnas.

ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con **ASC** "orden ascendente" y **DESC** "orden descendente". El valor predeterminado es ASC.

Funciones Agregadas que devuelven un único valor

- **AVG**: función utilizada para calcular el promedio de los valores de un campo determinado
- **COUNT**: función utilizada para devolver el número de registros de la selección
- **SUM**: función utilizada para devolver la suma de todos los valores de un campo determinado
- **MAX**: función utilizada para devolver el valor más alto de un campo especificado
- **MIN**: función utilizada para devolver el valor más bajo de un campo especificado

Ejemplo

SELECT ciudad, **COUNT**(ciudad) **AS** cantidad, **SUM**(salario) **AS** salario, **MAX**(salario) **AS** 'sal Máximo', **MIN**(salario) **AS** 'sal mínimo'
FROM docentes
GROUP BY ciudad
ORDER BY ciudad

	ciudad	Cantidad Docentes	salario	sal Máximo	sal Mínimo
1	Medellín	77	204140000	5700000	1100000
2	Sabaneta	21	74700000	4700000	3200000
3	itagui	19	58110000	5600000	980000
4	Bello	18	63050000	6500000	1200000
5	Envigado	17	54300000	4500000	1500000

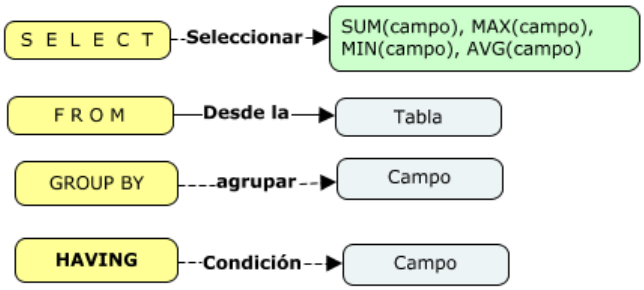
	ciudad	facultad	salario
1	Bello	Comunicación	30700000
2	Bello	Ingeniería	32350000
3	Envigado	Administración	25200000
4	Envigado	Diseño	29100000
5	itagui	Administración	22380000
6	itagui	Ingeniería	35730000
7	Medellín	Administración	37750000
8	Medellín	Diseño	65610000
9	Medellín	Ingeniería	55450000
10	Medellín	Medicina	45330000
11	Sabaneta	Derecho	45600000
12	Sabaneta	Publicidad	29100000

SELECT ciudad, facultad, **SUM**(salario) as salario
FROM docentes
GROUP BY facultad, ciudad

	ciudad	promedio
1	Bello	3502777
2	Envigado	3194117
3	itagui	3058421
4	Medellín	2651168
5	Sabaneta	3557142

SELECT ciudad, AVG(salario) as 'promedio'
FROM docentes
GROUP BY ciudad

3.2 HAVING



Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de **WHERE** pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a *GROUP BY* y la condición debe estar referida a los campos contenidos en ella.

- **La cláusula HAVING es diseñada para ser utilizada con la cláusula GROUP BY para restringir los grupos que aparecen en la tabla final.**
- **HAVING es similar a WHERE, pero WHERE filtra filas individuales mientras que HAVING filtra grupos.**
- **Los nombres de la columna en la cláusula HAVING deben también aparecer en la lista GROUP BY o estar contenido dentro de una función agregada.**

- Con **HAVING** podemos establecer una condición sobre un grupo de registros, algo muy importante es que HAVING acostumbra a ir acompañado de la cláusula GROUP BY. Esto último es así dado que HAVING opera sobre los grupos que nos "retorna" GROUP BY.

	ciudad	promedio
1	Bello	3502777
2	Sabaneta	3557142

```
SELECT ciudad, AVG(salario) as 'Promedio'
FROM docentes
GROUP BY ciudad
HAVING AVG(salario) >3200000
```

Ejercicios

Crear una consulta de agrupación por facultad y mostrar la suma, el promedio, el máximo y el mínimo de la columna salario. Ordenar la columna Total Salario.

Crear una consulta de agrupación que visualice la cantidad de docentes por facultad.

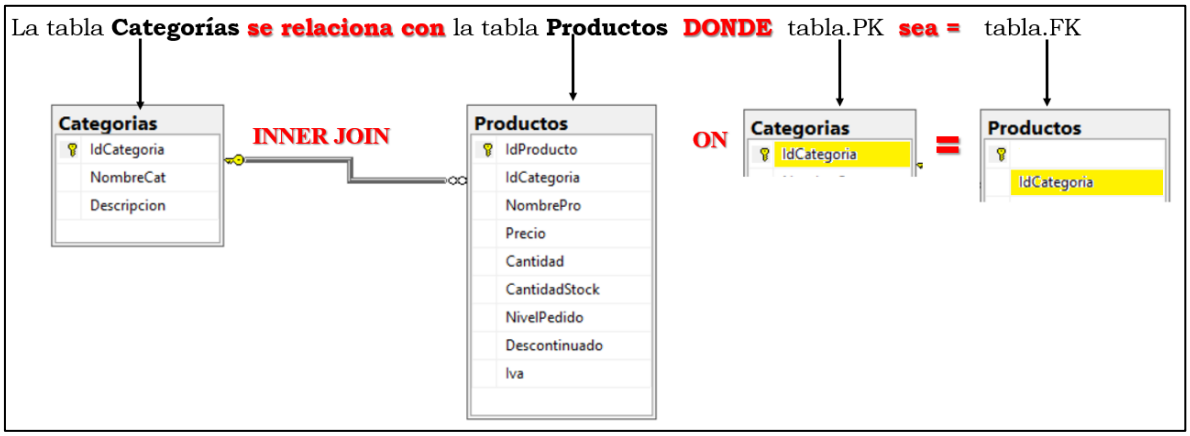
Crear una consulta de agrupación que visualice la suma de salario por comunas.

Crear una consulta de agrupación que visualice el promedio de salarios por facultad.

4 RELACIONES

Para realizar una relación entre dos tablas, se debe tener en cuenta las claves primarias y secundarias. SQL utiliza la instrucción **INNER JOIN** para seleccionar las tablas a relacionar y la instrucción **ON** para establecer dicha relación ***mediante las claves***.

4.1 Relacionar dos tablas



Ejemplo: Para relacionar la tabla categorías con la tabla productos.

ELECT Tabla.campo1, Tabla.campo2……

FROM Tabla1 INNER JOIN Tabla2 ON Tabla1.**IdPK** = Tabla2.**IdFK**

SELECT categorias.NombreCat, Productos.NombrePro, Productos.Precio

FROM categorías **INNER JOIN** productos **ON** categorías.IdCategoria = productos.Idcategoria.

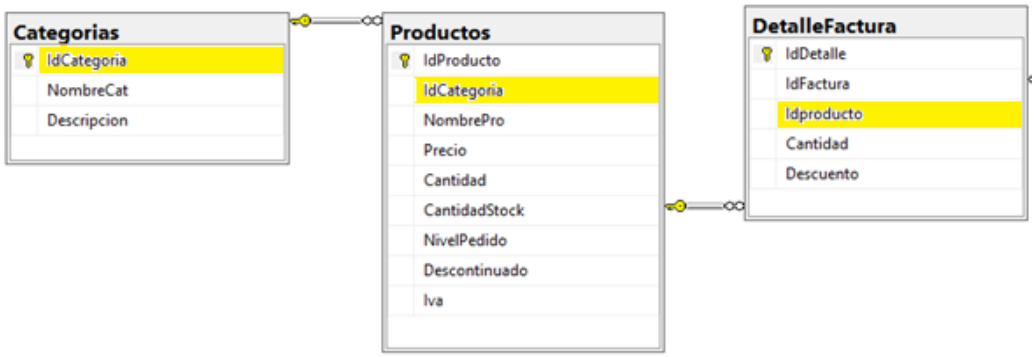
Recuerde.

INNER JOIN… selecciona las tablas a relacionar

ON … Establece la relación mediante las claves primarias (Primary Key) y secundarias (Foreing Key)

4.2 Relaciones tres tablas

Ejemplo: Para relacionar tres tablas, siga la secuencia categorías con la tabla productos con la tabla DetalleFactura.




```
SELECT Tabla.campo1, Tabla.campo2……
FROM Tabla1 INNER JOIN Tabla2 ON Tabla1.IdPK = Tabla2.IdFK
            INNER JOIN Tabla3  ON  Tabla2.IdPK = Tabla3.IdFK
```

```
SELECT categorias.NombreCat, Productos.NombrePro, Productos.Precio, DetalleFactura.Cantidad
FROM categorías INNER JOIN productos ON categorías.IdCategoria = productos.Idcategoria
            INNER JOIN DetalleFactura  ON  productos.Idproducto = DetalleFactura.Idproducto
```

Recuerde.

INNER JOIN… selecciona las tablas a relacionar

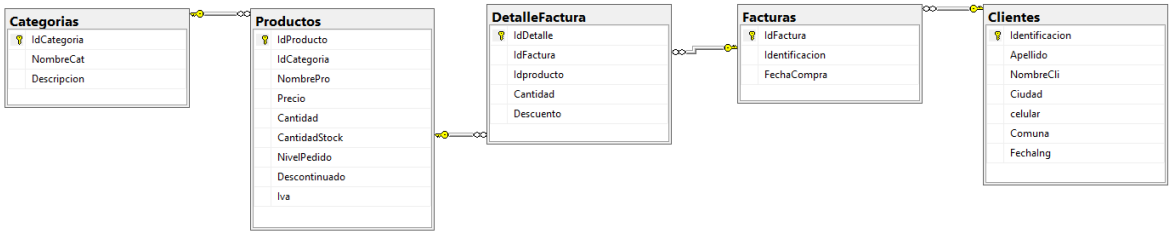
categorías INNER JOIN productos

INNER JOIN DetalleFactura

ON … Establece la relación mediante las claves primarias (Primary Key) y secundarias (Foreing Key)

```
ON categorías.IdCategoria = productos.Idcategoria
ON  productos.Idproducto = DetalleFactura.Idproducto
```

Ejercicios



1. Realizar una consulta SQL, que visualice la Categoría, descripción de la Categoría, Nombre del producto y el precio del producto. Ordenar la lista por Categoría y mostrar los precios de mayor a menor
2. Realizar una consulta SQL, que visualice la categoría y el nombre de los productos que se encuentren descontinuados (=1)
3. Realizar una consulta SQL, que visualice la categoría y el nombre de los productos que se encuentren con nivel de pedido (=1)
4. Realizar una consulta SQL, que visualice la categoría, descripción de la Categoría, Nombre del producto y el precio del producto; de los productos que tengan una cantidad superior a 30
5. Realizar una consulta SQL, que calcule: **Subtotal** = precio * cantidad, mostrar el nombre de la categoría, el nombre del producto, precio, cantidad y el Subtotal, colocar nombre a la columna Subtotal.
6. Realizar una consulta SQL, que muestre los siguientes datos: concatenar nombre y apellido, ciudad, Identificación del cliente, fecha de la factura y el IdFactura. Ordenar por ciudad y nombre.
7. Realizar una consulta SQL, que Liste los datos principales del cliente con Identificación = 24 por cada factura.
8. La tabla DETALLE FACTURA, es la tabla más importante de la base de datos, pues en ella se encuentra el resumen de los clientes y los productos. Por lo tanto, la información que se requiere, por lo general involucras varias tablas. Por Ejemplo, se desea saber del cliente con identificación 69, su nombre completo y el nombre de los productos, el precio, la cantidad, el descuento, así como los números de las facturas y la fecha de compra.

5 REFERENCIAS

https://www.aulaclic.es/sqlserver/t_2_3.htm

http://sql.11sql.com/sql-select.htm

http://deletesql.com/viewtopic.php?f=5&t=5

http://basededatos.umh.es/sql/sql02.htm

http://sql-principiantes.blogspot.com/

https://www.ibm.com/support/knowledgecenter/es/SSEPGG_11.1.0/com.ibm.db2.luw.sql.ref.doc/doc/r0059224.html

https://www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Como-realizar-consultas-simples-con-SELECT.aspx

https://sqltrainning.wordpress.com/2014/12/21/que-es-select/

http://www.mug-it.org.ar/332353-Como-manejar-las-fechas-en-Sql-Server.note.aspx

https://www.lawebdelprogramador.com/cursos/archivos/ManualPracticoSQL.pdf

https://www.manualsqlserver.com/?p=185

http://www.tutorialesprogramacionya.com/sqlserverya/temarios/descripcion.php?cod=33&punto=&inicio=

http://m.sql.11sql.com/sql-funcion-having.htm

http://carmoreno.github.io/sql/2017/02/09/Diferencia-entre-having-y-where/

https://www.1keydata.com/es/sql/sql-having.php

https://es.khanacademy.org/computing/computer-programming/sql

Imagen para talleres, tomada de la pagina:

https://pngtree.com/freepng/researching-new-technology-men_4595616.html