



**UNJu** | Universidad  
Nacional de Jujuy



Argentina  
programa  
4.0

# Introducción a las Bases de Datos



Java

**BASES DE DATOS**



Las bases de datos surgen de la necesidad de las empresas en almacenar grandes cantidades de información de una forma rápida, sencilla y fiable, y que a su vez pudieran acceder a ella en cualquier momento sin necesidad de desplazarse a salas dedicadas a archivar documentación.





# Java | Introducción a Bases de Datos

## **Antes de comenzar a definir vamos a diferenciar Datos e Información**

Todo sistema informático tiene como objetivo principal administrar y controlar los datos de manera eficiente para la toma de decisiones. Esto nos lleva a definir a los datos como todo aquello que se puede registrar en nuestra base de datos, que siendo procesado nos devolverá información relevante.

Entonces, un dato puede ser una letra, número, símbolo o palabra que por sí solo no tiene ningún significado, mientras que la información se define como un conjunto de datos procesados que tienen significado para el usuario.



# Java | Introducción a Bases de Datos



## Ejemplo de datos e información

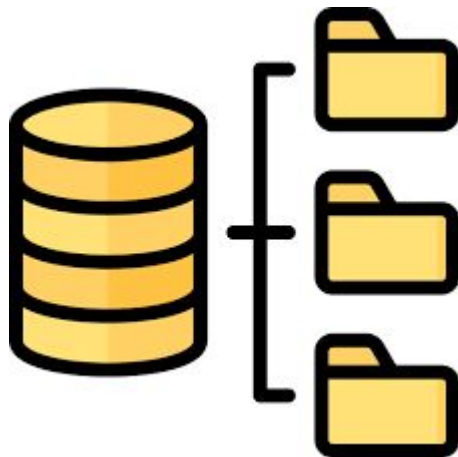


# Java | Introducción a Bases de Datos



**Y ahora ¿Qué es una Base de Datos?**

Es un programa o una herramienta capaz de almacenar gran cantidad de datos, relacionados y estructurados, que pueden ser consultados rápidamente de acuerdo con las características selectivas que se deseen.

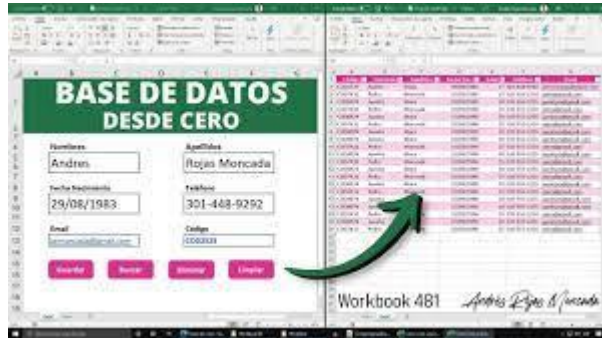




# Java | Introducción a Bases de Datos

**¿Escucharon hablar o conocen de algunas BD?**

Por ejemplo hay programas de uso común (Excel y Access) que se usan como BD pero si lo analizamos bien no son BD pero permiten almacenar y trabajar con datos.



# Java | Introducción a Bases de Datos



¿Escucharon hablar o conocen de algunas BD?

En cambio una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un **sistema de gestión de bases de datos (SGBD** o DBMS por sus siglas en inglés).



# Java | SGBD



## Y ahora ¿Qué es un Sistema Gestor de Bases de Datos?

Es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. Los usuarios acceden a la información con herramientas específicas de consulta y de generación de informes. Los SGBD, proporcionan métodos para la integridad de los datos, administrar el acceso a usuarios y recuperar la información en caso de un fallo.





# Java | SGBD



**Y ahora ¿Qué es un Sistema Gestor de Bases de Datos?**

El usuario usa un SGBD para poder consultar, extraer, insertar, modificar o borrar datos de una Base de Datos.



# Java | Tipos de BD

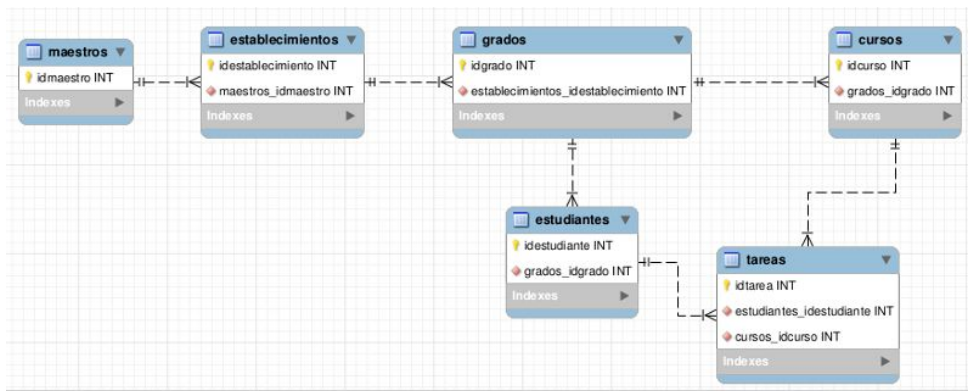


# Java | Bases de Datos Relacionales



Son las más utilizadas en aplicaciones reales y se usan para administrar datos de forma dinámica. Permiten crear todo tipo de datos y relacionarlos entre sí.

Los datos son almacenados en registros que son organizados en tablas, de esta forma pueden asociarse los elementos entre sí muy fácilmente, además se pueden cruzar sin ninguna dificultad.



# Java | Bases de Datos Relacionales



## Principales características:

- Pueden ser utilizadas por cualquier persona.
- Son de fácil gestión.
- Se pueden acceder a los datos con rapidez.
- Garantiza la total consistencia de los datos, sin posibilidad de error.
- Los datos pueden ser recuperados o almacenados mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar los datos.



# Java | Bases de Datos Relacionales



## Tablas, Filas y Columnas

Las BB Relacionales nos permiten agrupar los datos en forma de tablas que, a su vez, dentro de las tablas, los datos se organizan en filas y columnas. La intersección de un registro (fila) y un campo (columna) nos da el dato. Esto nos recuerda mucho a la forma en que trabajamos con las planillas de cálculo como Excel, LibreOffice, OpenOffice, etc.

Nombre de la tabla: Trabajo			
Código	Nombre	Posición	Salario
1	Edgardo Trujillo	Gerente	19000
2	Lidimarie Fonsi	Empleada	12000
3	Jean Piaget	Empleado	13500
4	Jerome Bruner	Empleado	14000



# Java | Bases de Datos Relacionales



## Modelo de Datos

Un modelo de datos es una serie de conceptos que se utilizan para describir un conjunto de datos y las operaciones para manipularlos.

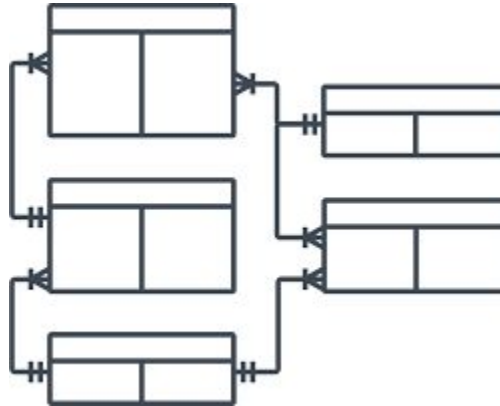
Hay dos tipos de modelos de datos: **los modelos conceptuales** y **los modelos lógicos**. Los modelos conceptuales se utilizan para **representar la realidad a un alto nivel de abstracción**. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, **las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos**.



# Java | Modelo de Entidad-Relación



Cuando empezamos a pensar o diseñar una base de datos, puede parecer simple en un principio, pero en la medida que nuestra idea de base de datos crece, el diseño también crece y necesitamos una forma de poder conceptualizar y visualizar todos los elementos que la componen, cómo se relacionan y qué características tienen.



# Java | Modelo de Entidad-Relación



## ¿Qué es un MER o DER?

Un MER o DER, es un tipo de diagrama de flujo que muestra cómo las “**entidades**”, como personas, objetos o conceptos, los cuales tienen distintos “**atributos**”, “**se relacionan**” entre sí dentro de un sistema.



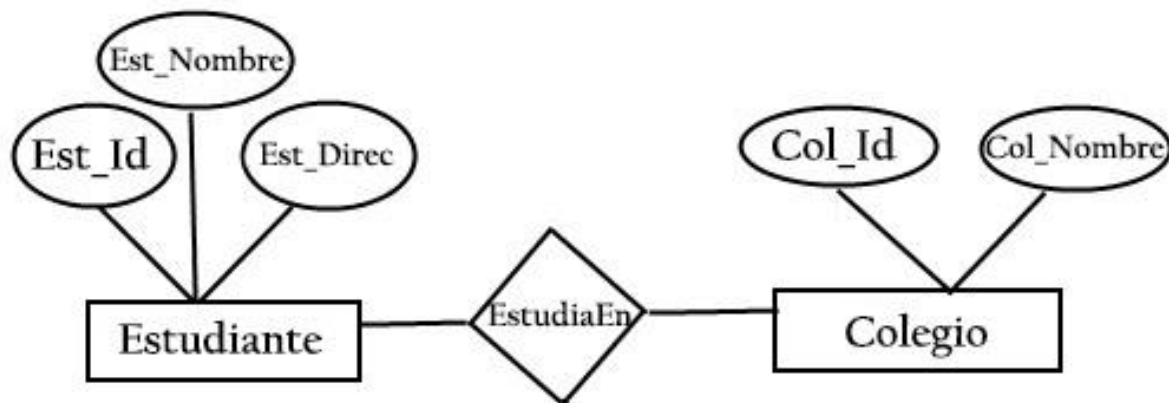


# Java | Modelo de Entidad-Relación



## Elementos básicos de un MER

1. Entidad
2. Atributo
3. Relación



# Java | Modelo de Entidad-Relación



**Entidad:** Una entidad representa una “cosa”, "objeto" o "concepto" del mundo real.

Ejemplo:

1. Una persona: se diferencia de cualquier otra persona, incluso siendo gemelos.
2. Un automóvil: aunque sean de la misma marca, el mismo modelo, etc, tendrán atributos diferentes, por ejemplo, el número de chasis o color.
3. Una casa: aunque sea exactamente igual a otra, aún se diferenciará en su dirección.



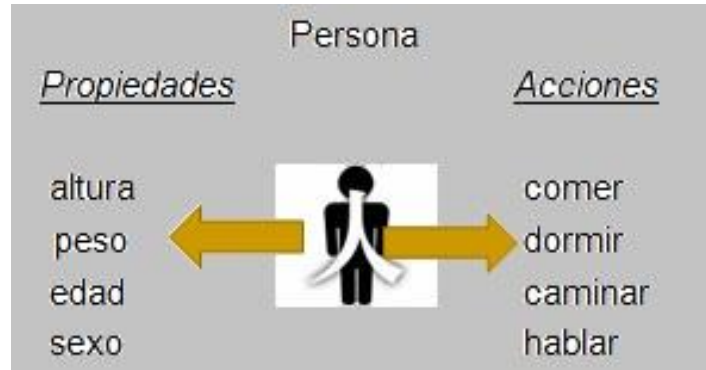
# Java | Modelo de Entidad-Relación



**Atributo:** Los atributos son las características o propiedades que definen o identifican a una entidad.

Ejemplo:

1. Para una entidad Persona tenemos: altura, peso, edad, sexo, etc.
2. Para una entidad Materia tenemos: nombre, tipo (anual o cuatrim), régimen (promocional, examen final), etc.



# Java | Modelo de Entidad-Relación



**Relaciones:** Consiste en una colección, o conjunto, de relaciones entre las entidades.

Ejemplo:

1. Dadas las entidades "Alumno" y "Materia", la relación que podemos tener es que uno o más alumnos estarán estudiando una materia. Además, a esta relación le podemos dar un título: "Estudia" o "Cursa" ya que los alumnos estudian o cursan materias.
2. Dadas las entidades "Habitación" y "Huésped", la relación que podemos tener es que uno o más huéspedes estarán alojados en una habitación. Además, a esta relación le podemos dar un título: "Se aloja" ya que los huéspedes se alojan en las habitaciones.



# Java | Modelo de Entidad-Relación



## Cardinalidad de las Relaciones

La cardinalidad de un atributo indica el número mínimo y el número máximo de valores que puede tomar para cada ocurrencia de la entidad o relación a la que pertenece. El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M".



# Java | Modelo de Entidad-Relación



## Cardinalidad de las Relaciones

Las relaciones entre las entidades se pueden agrupar en 3 tipos según su finalidad.

1. Uno a Uno (1:1) se da cuando un elemento de una entidad se puede relacionar solamente con un solo registro de otra entidad y viceversa.
2. Uno a Muchos (1:M) se da cuando un registro de una entidad A se puede relacionar con cero o muchos registros de otra entidad B y cada registro de la entidad B se relaciona con un sólo registro de la entidad A.
3. Muchos a Muchos (N:M) se da cuando un registro de una entidad se relaciona con cero o varios registros de otra entidad.



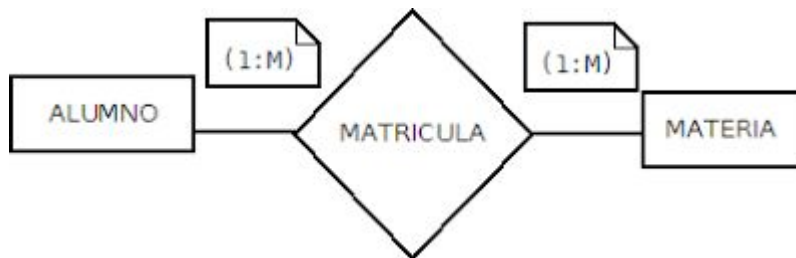
# Java | Modelo de Entidad-Relación



## Cardinalidad de las Relaciones

Ejemplo: Si tuviéramos el caso de una base de datos de una escuela podríamos pensar en las siguientes relaciones:

1. Un alumno de una escuela puede realizar una sola materia (1:1)
2. Un alumno de una escuela puede realizar varias materias (1:N)
3. Un alumno de una escuela puede realizar varias materias y las materias pueden ser cursadas por varios alumnos (N:N)



# Java | Modelo de Entidad-Relación



## Símbolos y notaciones de Diagrama E-R

Las entidades, sus atributos y las relaciones se pueden representar de forma gráfica a través de algunos elementos visuales. A continuación, una imagen que nos describe en primer término cómo se visualiza una entidad, un atributo y una relación:



Entity



Attribute









Relationship



# Java | Modelo de Entidad-Relación



## Símbolos de cardinalidad

Símbolo de cardinalidad	nombre
	One
	Many
	One (and only one)
	Zero or one
	One or many
	Zero or many



# Java | Modelo de Entidad-Relación

## Herramientas para crear MER - DER

Los DER pueden verse confusos, pero no son tan complejos como parecen. Los diagramas entidad-relación se pueden generar fácilmente, y existen muchas herramientas que permiten facilitarnos la tarea como:

1. <https://app.creately.com/>
2. <https://www.lucidchart.com/>
3. <https://app.diagrams.net/>



# Java | Modelo de Entidad-Relación



**Ejemplo: Abrir un programa de los mencionados anteriormente y crear un DER conceptual de un sistema simple en el cual un estudiante se registra para un curso que es impartido por un profesor.**

1. Determinar entidades.
2. Identificar relaciones.
3. Agregar atributos.
4. Identificar cardinalidad de las relaciones.



# Java | Modelo de Entidad-Relación



## **Tipos de Datos**

Conocer y definir los tipos de datos en el diseño físico es muy importante porque forman parte de las restricciones que sirven para establecer las reglas de una tabla. Especificando qué tipos de datos se pueden añadir en un registro evita que se inserten datos incorrectos. Es necesario conocer los tipos de datos de SQL.



# Java | Modelo de Entidad-Relación



## Tipos de Datos

A continuación, una sencilla tabla de tipos de datos que manipulan las bases de datos.

Tipo de Datos	Longitud	Descripción
BIT	1 byte	Valores Si/No - True/False - Verdadero/Falso
DATETIME	8 bytes	Fecha u hora entre los años 100 y 9999.
DOUBLE	8 bytes	Un valor en punto flotante de precisión doble con un rango de - 1.79769313486232*10308 a -4.94065645841247*10-324 para valores <b>negativos</b> siendo para valores <b>positivos</b> entre 4.94065645841247*10-324 a 1.79769313486232*10308 y 0.
LONG	4 bytes	Valor entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por carácter	De cero a un máximo de 1.2 gigabytes.
TEXT	1 byte por carácter	De cero a 255 caracteres.

# Java | Modelo de Entidad-Relación



## Índice

Para facilitar la obtención de información de una tabla se utilizan índices. El índice de una tabla desempeña la misma función que el índice de un libro: permite encontrar datos rápidamente. En el caso de las tablas, localiza registros.

Sin índice, se debe recorrer secuencialmente toda la tabla para encontrar un registro consumiendo más tiempo y recursos. Entonces el objetivo de un índice es acelerar la recuperación de información.



# Java | Modelo de Entidad-Relación



## Índice

Hay distintos índices, a saber:

1. **Index:** Puede haber varios varios por tabla. Es un índice común. Los valores no necesariamente son únicos y aceptan valores nulos. Podemos darle un nombre, si no se lo damos, se coloca uno por defecto. "key" es sinónimo de "index".
2. **Unique:** Es un índice para las cuales los valores deben ser únicos y diferentes, aparece un mensaje error si intentamos agregar un registro con un valor ya existente. Permite valores nulos y pueden definirse varios por tabla.
3. **Clave Primaria o Primary Key (PK).**

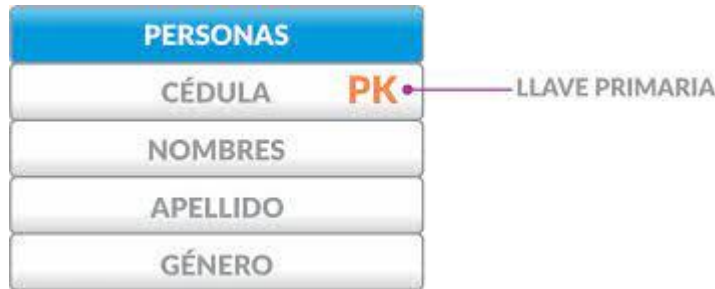


# Java | Modelo de Entidad-Relación



## Clave Primaria (PK)

Una clave primaria es la columna o colección de columnas que nos permiten identificar de forma única a una fila determinada en una tabla. La clave primaria proporciona una forma importante de distinguir una fila de otra. Subrayar las columnas o la colección de columnas que componen la clave primaria usualmente es la mejor forma de representar la clave primaria de cada tabla de la base de datos.





# Java | Modelo de Entidad-Relación



## Clave Primaria

Veamos un ejemplo, si tenemos una tabla con datos de personas, el número de documento puede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo, por ejemplo), pero su documento será siempre distinto.



DNI	Nombre	Apellido	Edad
01886962	Franklin	García	25
04264489	David	Sánchez	32
02679366	Pedro	Hernández	48

# Java | Modelo de Entidad-Relación



## Clave Foránea (FK)

Una clave foránea es un campo o colección de campos de una tabla cuyos valores deben coincidir con los valores de la clave primaria de una segunda tabla.



# Java | SQL - Structured Query Language



El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o **Lenguaje Estructurado de Consultas**, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

.



# Java | Bases de Datos SQL



Si queremos crear, insertar o consultar datos desde una base de datos será necesario interactuar con el motor de la base de datos mediante lenguaje de consulta estructurado o SQL.

## ¿Qué es SQL?

SQL es un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos. Los programas de bases de datos relacionales, como SQL Server, MySQL, MariaDB, etc. usan SQL para trabajar con datos. El lenguaje SQL



# Java | Bases de Datos SQL



## Operaciones SQL:

### 1. Crear una base de datos

```
CREATE DATABASE MI_BASE_DE_DATOS ;
```

### 2. Crear tabla

```
CREATE TABLE Nombre_Tabla(  
  Nombre_Campo tipo_valor opciones,  
  Nombre_Campo_2 tipo_valor opciones  
);
```

```
CREATE TABLE Clientes(  
  Id_Cliente int NOT NULL,  
  Nombre varchar(50) NOT NULL,  
  Apellido varchar(50) NOT NULL,  
  Id_Localidad int NOT NULL  
);
```



# Java | Bases de Datos SQL



## Operaciones SQL:

### 3. Crear clave primaria (PK)

```
ALTER TABLE Nombre_Tabla  
ADD PRIMARY KEY (Nombre_campo);
```

```
ALTER TABLE Clientes  
ADD PRIMARY KEY (Id_Cliente);
```

### 4. Crear clave foránea (FK)

```
ALTER TABLE Tabla_origen ADD FOREIGN KEY(campo_tabla_origen)  
REFERENCES Tabla_destino (campo_clave_primaria_destino)
```

```
ALTER TABLE Clientes ADD FOREIGN KEY(Id_Localidad)  
REFERENCES Localidades (Id_Localidad)
```



# Java | Bases de Datos SQL



## Operaciones SQL:

**5. Seleccionar datos:** el SELECT que sirve para obtener los datos de una tabla.

```
SELECT * FROM Nombre_tabla
```

```
SELECT campo_1,campo_2,... FROM Nombre_tabla
```

```
SELECT Nombre, Apellido FROM Clientes
```



# Java | Bases de Datos SQL



## Operaciones SQL:

**6. Insertar datos:** El comando para insertar datos en una tabla es INSERT. Deben respetar el orden de las columnas con los datos que se van a guardar y deben estar separados por comas las columnas y es posible ignorar los campos que permiten valores nulos.

```
INSERT INTO nombre_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

```
INSERT INTO Clientes (Nombre, Apellido, Id_Localidad) VALUES ('Juan',
'Gomez', 15)
```





# Java | Bases de Datos SQL



## Operaciones SQL:

**7. Actualizar datos (modificar datos):** La instrucción UPDATE sirve para actualizar los registros existentes. Es necesario definir una condición que permita identificar los registros que se quieren actualizar:

```
UPDATE Nombre_tabla  
SET columna1 = valor1, columna2 = valor2, ...  
WHERE condicion;
```

```
UPDATE Clientes SET Nombre = 'Juan Carlos', Apellido = 'Gómez' WHERE  
Id_Cliente = 10
```



# Java | Bases de Datos SQL



## Operaciones SQL:

**8. Borrar datos:** Para borrar registros necesitamos la instrucción DELETE. Nuevamente es necesario y muy importante tener una condición que nos permita identificar los registros a eliminar. En caso de que no se defina esta condición es posible que perdamos todos los registros de nuestra tabla o base de datos. La instrucción sería:

```
DELETE FROM Nombre_tabla WHERE condicion;
```

```
DELETE FROM Clientes WHERE Id_Cliente = 10
```



# Java | Bases de Datos SQL



## Operaciones SQL:

**9. Ordenación de datos:** Para esa acción tenemos la instrucción ORDER BY. Se pueden especificar una o más columnas. La sentencia es la siguiente:

```
SELECT columna1, columna2, ...  
FROM Nombre_tabla  
ORDER BY columna1, columna2, ... ASC|DESC;
```

```
SELECT * FROM Clientes ORDER BY Apellido ASC
```



# Java | Bases de Datos SQL



## Operadores

Además de los operadores lógicos que ya conocemos (=,!=,<,>,<>,<=, >=) tenemos otros como por ejemplo:

- IS NULL para seleccionar aquellos campos que sean nulos
- IS NOT NULL para seleccionar aquellos campos que no sean nulos
- IN para seleccionar aquellos campos que estén dentro de una lista
- LIKE para buscar con algún patrón.

También tenemos la posibilidad de unir condiciones bajo los operadores AND, OR y NOT



# Java | Bases de Datos SQL



## Funciones de agregación

Una función de agregación es un cálculo sobre un conjunto de valores, devolviendo un sólo valor, excepto por la función COUNT. Estas funciones ignoran los valores NULL. Por lo general, debemos combinar las funciones de agregado con la cláusula GROUP BY. Algunas de ellas son MAX (Máximo), MIN (Mínimo), AVG (Promedio), SUM (Suma), COUNT (Cantidad) entre otras.

Por ejemplo, si tuviéramos una tabla Transferencias con un campo monto y quisiéramos obtener el monto máximo, nuestra consulta sería la siguiente:

```
SELECT MAX(Monto) FROM Transferencias;
```



# Java | Bases de Datos SQL



## Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de otra instrucción SELECT, las cuales pueden contener condiciones como las mencionadas anteriormente. Para generar una subconsulta debemos tener presente ciertas reglas como: escribir la subconsulta entre paréntesis, especificar en ella sólo una columna (si no se utiliza IN, ANY, ALL o EXISTS), que no contenga la cláusula GROUP BY.



# Java | Bases de Datos SQL



## Subconsultas

Por ejemplo, si quisiéramos obtener los Clientes que tienen en el campo Id\_Localidad cualquier ID que esté en la tabla Provincias siendo Córdoba el nombre de la misma, nuestra consulta sería la siguiente:

```
SELECT * FROM Clientes WHERE Id_Localidad IN(SELECT Id_Localidad FROM Provincias WHERE Nombre = 'Córdoba')
```

La subconsulta obtendrá todos los valores de Id\_Localidad que contenga la provincia Córdoba y la consulta obtendrá los Clientes que tengan el valor de Id\_Localidad que estén en la lista de la subconsulta.



# Java | Bases de Datos SQL



## Manos a la obra

Para comenzar a trabajar consultas a bases de datos SQL, primeramente debemos instalar:

XAMPP (Apache, MySQL, PHP, PERL), además te instala PHPMyAdmin que nos posibilita trabajar las consultas SQL mediante una interfaz:  
<https://www.apachefriends.org/es/index.html>



O MySQL Workbench que también nos permite trabajar consultas SQL:  
<https://dev.mysql.com/downloads/>



En nuestro caso vamos a trabajar con el primero XAMPP

