

CURSO DE PROGRAMACIÓN FULL STACK

APÉNDICE C

# BASE DE DATOS



# TIPOS DE DATOS, OPERADORES Y COMANDOS EN MYSQL

## TIPOS DE DATOS

Las columnas de la base de datos almacenan valores que pueden ser de diversos tipos. A continuación, se indican los tipos de datos agrupados en tres grupos.

### Tipos de dato numéricos

Listado de cada uno de los tipos de dato numéricos en MySQL, su ocupación en disco y valores.

- *INT (INTEGER): Ocupación de 4 bytes con valores entre -2147483648 y 2147483647 o entre 0 y 4294967295.*
- *SMALLINT: Ocupación de 2 bytes con valores entre -32768 y 32767 o entre 0 y 65535.*
- *TINYINT: Ocupación de 1 bytes con valores entre -128 y 127 o entre 0 y 255.*
- *MEDIUMINT: Ocupación de 3 bytes con valores entre -8388608 y 8388607 o entre 0 y 16777215.*
- *BIGINT: Ocupación de 8 bytes con valores entre -8388608 y 8388607 o entre 0 y 16777215.*
- *DECIMAL (NUMERIC): Almacena los números de coma flotante como cadenas o string.*
- *FLOAT (m,d): Almacena números de coma flotante, donde 'm' es el número de dígitos de la parte entera y 'd' el número de decimales.*
- *DOUBLE (REAL): Almacena número de coma flotante con precisión doble. Igual que FLOAT, la diferencia es el rango de valores posibles.*
- *BIT (BOOL, BOOLEAN): Número entero con valor 0 o 1.*

### Tipos de dato con formato fecha

Listado de cada uno de los tipos de dato con formato fecha en MySQL, su ocupación en disco y valores.

- *DATE: Válido para almacenar una fecha con año, mes y día, su rango oscila entre '1000-01-01' y '9999-12-31'.*
- *DATETIME: Almacena una fecha (año-mes-día) y una hora (horas-minutos-segundos), su rango oscila entre '1000-01-01 00:00:00' y '9999-12-31 23:59:59'.*
- *TIME: Válido para almacenar una hora (horas-minutos-segundos). Su rango de horas oscila entre -838-59-59 y 838-59-59. El formato almacenado es 'HH:MM:SS'.*
- *TIMESTAMP: Almacena una fecha y hora UTC. El rango de valores oscila entre '1970-01-01 00:00:01' y '2038-01-19 03:14:07'.*
- *YEAR: Almacena un año dado con 2 o 4 dígitos de longitud, por defecto son 4. El rango de valores oscila entre 1901 y 2155 con 4 dígitos. Mientras que con 2 dígitos el rango es desde 1970 a 2069 (70-69).*

## Tipos de dato con formato string

Listado de cada uno de los tipos de dato con formato string en MySQL, su ocupación en disco y valores.

- CHAR: Ocupación fija cuya longitud se especifica entre paréntesis y comprende de 1 a 255 caracteres. El espacio no utilizado se rellena con blancos.
- VARCHAR: Ocupación variable cuya longitud comprende de 1 a 255 caracteres.
- SET: Almacena 0, uno o varios valores una lista con un máximo de 64 posibles valores.
- ENUM: Igual que SET pero solo puede almacenar un valor.
- TINYTEXT: Una longitud máxima de 255 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- TEXT: Una longitud máxima de 65.535 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- MEDIUMTEXT: Una longitud máxima de 16.777.215 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- LONGTEXT: Una longitud máxima de 4.294.967.298 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.

## COMANDOS PARA LA MANIPULACIÓN DE BASES DE DATOS

- CREATE: Sirve para crear tablas o un nuevo esquema de base de datos.
- DROP: Sirve para eliminar columnas, tablas o bases de datos.
- ALTER: Se utiliza para modificar la estructura de una tabla, éste puede ser combinado con otros comandos como: ADD, DROP, ADD PRIMARY KEY(), ADD FOREIGN KEY(), entre otros.
- DESCRIBE: Muestra la estructura de las tablas de la base de datos.
- SHOW TABLES: Muestra las tablas que pertenecen a la base de datos.

# COMANDOS PARA LA MANIPULACIÓN DE INFORMACIÓN

- **SELECT:** Es utilizado para realizar consultas a la base de datos que cumplan con alguna condición.
- **INSERT:** Es utilizado para la inserción de información en la base de datos.
- **UPDATE:** Se utiliza para la modificación o actualización de información en la base de datos.
- **DELETE:** Elimina registros de la base de datos.

## CLÁUSULAS

- **FROM:** Se utiliza para especificar la tabla o tablas de las cuales se va a seleccionar los registros.
- **WHERE:** Es utilizado para determinar las condiciones que debe cumplir la información seleccionada.
- **GROUP BY:** Sirve para agrupar los registros seleccionados en grupos específicos.
- **HAVING:** Expresa la condición que debe satisfacer cada grupo.
- **ORDER BY:** Sirve para ordenar registros de acuerdo a algún campo específico. Esta cláusula puede realizarse de forma ascendente o descendente.

## CONSULTAS CON PREDICADO

El predicado se incluye entre la cláusula y el primer nombre del campo a recuperar. Los posibles predicados son:

<b>Predicado</b>	<b>Descripción</b>
ALL	Devuelve todos los campos de la tabla
TOP	Devuelve un determinado número de registros de la tabla
DISTINCT	Omite los registros cuyos campos seleccionados coincidan totalmente
DISTINCTROW	Omite los registros duplicados basándose en la totalidad del registro y no en los campos Seleccionados.

## OPERADORES LÓGICOS

- AND: Es el equivalente a "y", devuelve valor de verdad solo si todas las condiciones son verdaderas.
- OR: Es el equivalente a "o", devuelve valor de verdad si al menos una de las condiciones es verdadera.
- NOT: Es el equivalente a negación, devuelve valor verdadero cuando la condición es falsa y viceversa.

## OPERADORES DE COMPARACIÓN

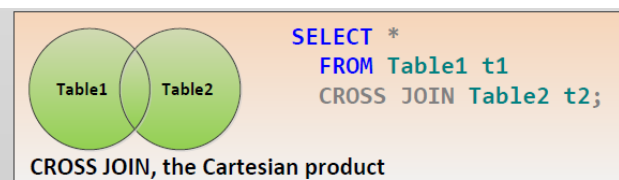
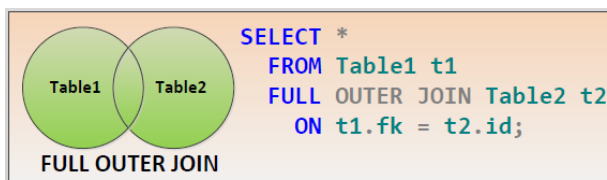
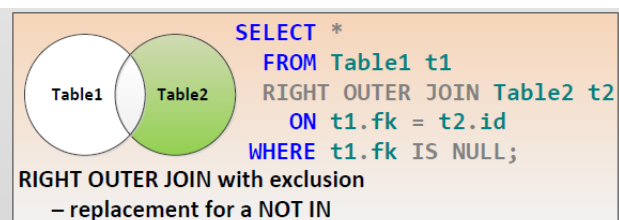
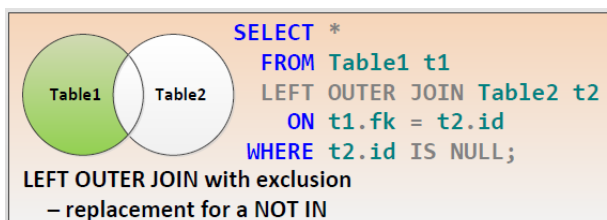
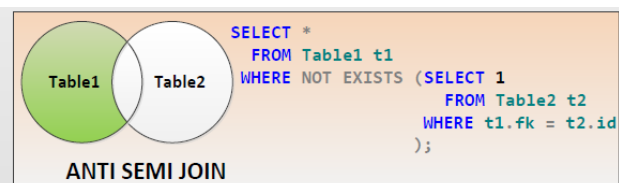
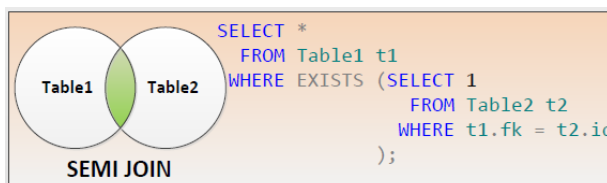
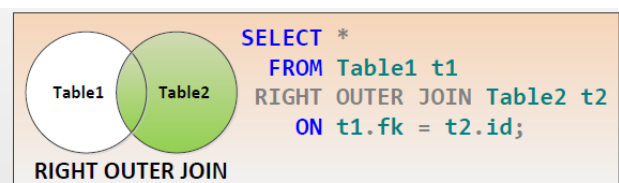
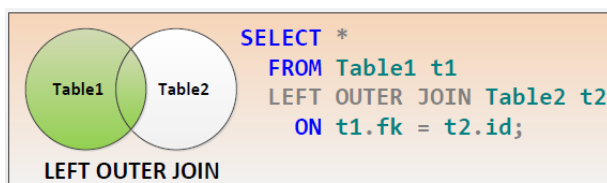
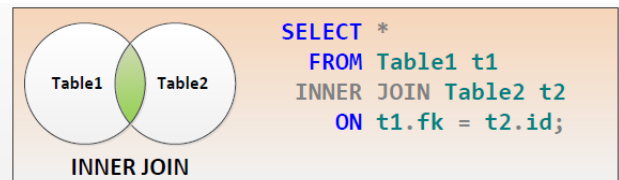
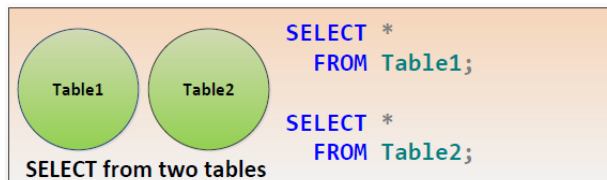
- <: *Menor que.*
- >: *Mayor que.*
- <>: *Diferente.*
- <=: *Menor o igual que.*
- >=: *Mayor o igual que.*
- =: *Igual que.*
- BETWEEN: *Se utiliza para especificar un rango de valores.*
- IN: *Se utiliza para especificar registros de una base de datos.*
- LIKE: *Se utiliza para comparar patrones de texto pudiendo incluir comodines como los siguientes:*

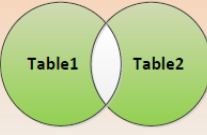
<b>Comodín</b>	<b>Descripción</b>
%	<i>Sustituto para cero o más caracteres.</i>
—	<i>Sustituto para exactamente un carácter</i>
[lista caracteres]	Cualquier carácter de la lista
[^lista caracteres]	Cualquier carácter que no esté en la lista
[!lista caracteres]	Cualquier carácter que no esté en la lista

## **FUNCIONES DE COLUMNA**

- Avg: Devuelve como resultado el promedio de un campo determinado.
- Count: Devuelve como resultado el número de registros que cumplen una condición.
- Sum: Devuelve como resultado la suma de los valores de un campo determinado.
- Max: Devuelve el valor más alto de un campo determinado.
- Min: Devuelve el valor más bajo de un campo determinado.

# TIPOS DE JOIN REPRESENTADOS A TRAVÉS DE DIAGRAMAS DE VENN



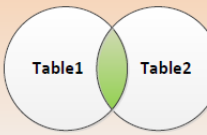


```

SELECT *
FROM Table1 t1
FULL OUTER JOIN Table2 t2
ON t1.fk = t2.id
WHERE t1.fk IS NULL
OR t2.id IS NULL;

```

**FULL OUTER JOIN with exclusion**

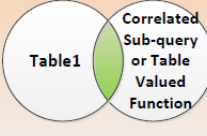


```

SELECT *
FROM Table1 t1
INNER JOIN Table2 t2
ON t1.fk >= t2.id;

```

**NON-EQUI INNER JOIN**

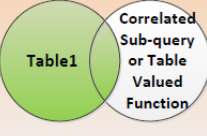


```

SELECT *
FROM Table1 t1
CROSS APPLY
[dbo].[someTVF](t1.fk)
AS t;

```

**CROSS APPLY**

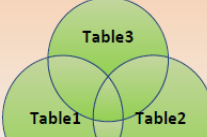


```

SELECT *
FROM Table1 t1
OUTER APPLY
[dbo].[someTVF](t1.fk)
AS t;

```

**OUTER APPLY**

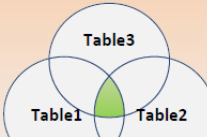


```

SELECT *
FROM Table1 t1
FULL OUTER JOIN Table2 t2
ON t1.fk = t2.id
FULL OUTER JOIN Table3 t3
ON t1.fk_table3 = t3.id;

```

**Two FULL OUTER JOINS**

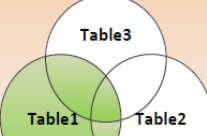


```

SELECT *
FROM Table1 t1
INNER JOIN Table2 t2
ON t1.fk = t2.id
INNER JOIN Table3 t3
ON t1.fk_table3 = t3.id;

```

**Two INNER JOINS**

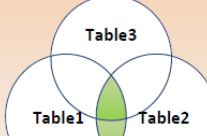


```

SELECT *
FROM Table1 t1
LEFT OUTER JOIN Table2 t2
ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
ON t1.fk_table3 = t3.id;

```

**Two LEFT OUTER JOINS**

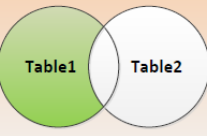


```

SELECT *
FROM Table1 t1
INNER JOIN Table2 t2
ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
ON t1.fk_table3 = t3.id;

```

**INNER JOIN and a LEFT OUTER JOIN**

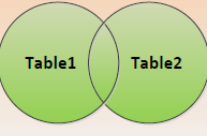


```

SELECT fk as id
FROM Table1
EXCEPT
SELECT ID
FROM Table2;

```

**EXCEPT**

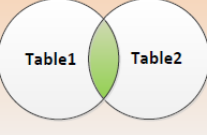


```

SELECT fk as id
FROM Table1
UNION
SELECT ID
FROM Table2;

```

**UNION**



```

SELECT fk as id
FROM Table1
INTERSECT
SELECT ID
FROM Table2;

```

**INTERSECT**

### Sample Schema

**Table 1  
(People)**

	id	Name	fk	fk_table3
1	1	Steve	1	NULL
2	2	Aaron	3	NULL
3	3	Mary	2	NULL
4	4	Fred	1	NULL
5	5	Anne	5	NULL
6	6	Beth	8	1
7	7	Johnny	NULL	1
8	8	Karen	NULL	2

**Table 2  
(Favorite Colors)**

	id	FavoriteColor
1	1	red
2	2	green
3	3	blue
4	4	pink
5	5	purple
6	6	mauve
7	7	orange
8	8	yellow
9	1	indigo

**Table 3  
(Favorite Foods)**

	id	dataValue
1	1	Pizza
2	2	Burger
3	3	Sushi

Note: Column names are very generic to simplify the sample queries.  
 Foreign keys are  
 Table1.fk -> Table2.id  
 Table2.fk\_table3 -> Table3.id