

BASE DE DATOS (1ER SESION)

“Los datos, la información es esa oportunidad que queremos para hacer el cambio, la generalidad consiste como la utilizamos para ello”. Liz Ruiz

¿QUE ES?

Data, almacenamiento guardado o un listado.

Es una colección de datos almacenados y organizados de forma que un programa o sistema pueda solucionarlos rápidamente con la capacidad de poder: recuperarlos, actualizarlos, insertarlos y eliminarlos.



PROBLEMAS:

Seguridad: cualquier persona se puede meter

Tamaño: la cantidad de información.

Precisión y redundancia: repetición de los datos.

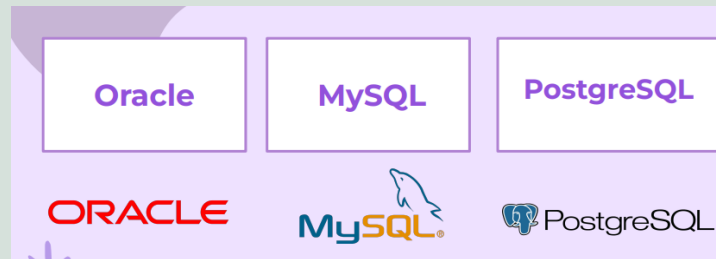
Se implementan en todo, en las aplicación y sistemas informáticos.

Fines: modelado de datos, análisis y ciencia de datos e inteligencia de negocios.



DBMS (DATABASE MANAGER SYSTEM) o SGBD (SISTEMA GESTOR DE BASE DE DATOS)

Tipo de software específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que lo utilizan.



Tipos de sistemas relacionales



Relacional: consiste en un conjunto de tablas, a cada una de las cuales se les asigna un nombre exclusivo y cada fila de la tabla representa una relación entre un conjunto de valores.

Oracle, MySQL, PostgreSQL, SQL Server y MariaDB

No relacional: es aquella que no usa el esquema tabular de filas y columnas que se encuentra en la mayoría de los sistemas de base de datos más tradicionales.

MongoDB y Casandra

OTROS:

Jerárquicos: almacenan la información de forma jerárquica almacenando los datos permitiendo crear estructuras de gran rendimiento.

De red: ofrece una solución eficiente al problema a la redundancia de datos. Utilizando en su mayoría por programas mas que por usuarios finales.

Orientados a objetos: almacenan los objetos de forma completa, de la base de datos a objetos incorpora todos los objetos del paradigma de objetos.

- Encapsulación: propiedad permite ocultar los datos al resto de los objetos, impidiendo así accesos incorrectos.
- Herencia: propiedad que a través de lo cual los objetos heredan comportamiento dentro de una jerarquía indicada a los objetos.
- Polimorfismo: propiedad de una operación mediante el cual puede ser aplicada diferentes comportamientos en distintos tipos de objetos.

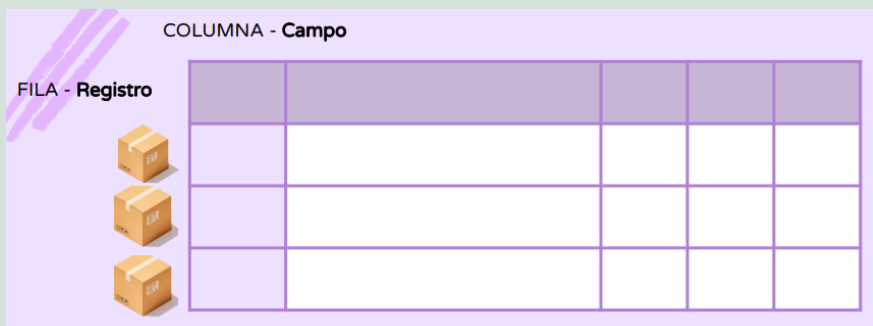
TABLAS

Tabla: es una estructura de datos donde se guardan y almacenan datos recogidos un programa o sistema.

Campos: se define por la cantidad de información en que fraccionamos la información que se guardara.

Tipo de datos: Cada campo tendrá definido **un dato que limitará** lo que podría almacenarse y también definirá la longitud de tamaño.

Registro: es la información que se guarda en la tabla por medios de filas y renglones.



COLUMNA - Campo				
FILA - Registro				

Atributo: son las columnas de la relación y describen las características particulares.

Llave primaria (primary key PK): es un valor único en un conjunto de atributos que permite identificar a una tupla de manera única en cualquier momento.

CLAVE PRIMARIA O PRIMARY KEY
VALORES UNICOS

Id Numérico	Nombre Texto	Tipo Texto	Capacidad Numerico	Dirección Numerico	Teléfono Texto
1	El Dorado	Comida mexicana	100	Jalisco México. Col 102	6666666666
2	Capitán	Marisquería	50	Sinaloa México.Col Delicias	8888888888

Relación: es una colección o grupo de objetos que tienen en común un conjunto de características o atributos.



Es una asociación entre varias entidades, se presenta mediante líneas y cambios intermedio.

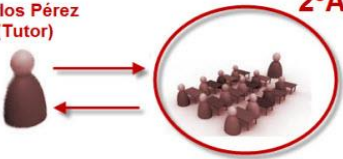
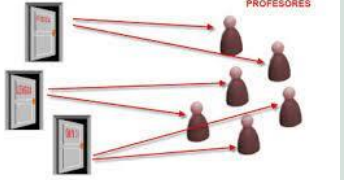
Llave foránea (foreign key FK): es un atributo que hace referencia a una llave primaria de otra tabla. Una tabla puede tener varias llaves foráneas, las cuales permite establecer las relaciones.

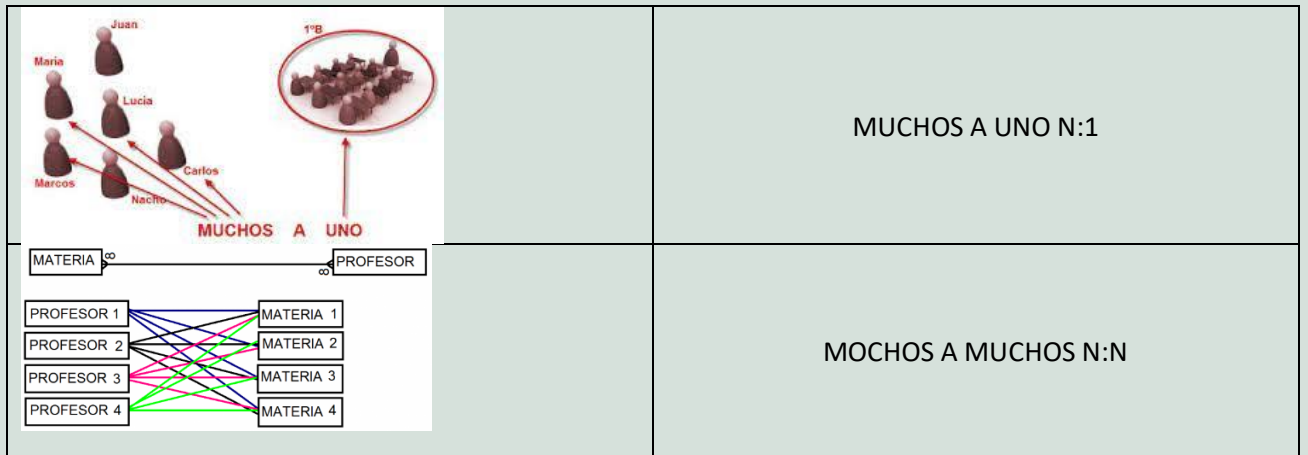
Tabla: Platillo					Tabla: Cliente				
Id Numérico	Nombre Texto	Descripción Texto	Ingredientes Texto	Precio Numérico	Id Numérico	Nombre Texto	Apellido Texto	Teléfono Texto	Dirección Texto
1	Pizza	Pizza mexicana	Champiñón, chorizo, chile jalapeño	300	1	Lizeth	Ruiz	6666666666	Sinaloa, México
2	Tacos	Tacos de asada de res	salida, guacamole	150					

Tabla: Orden					Clave foráneas o Foreign Key	
Id Numérico	Cantidad Numérico	Importe Numérico	Fecha Date		cliente_id Numérico	platillo_id Numérico
1	2	600	2021/06/01		1	1

Tipos DE RELACIONES:

TIPO	RELACIÓN	REPRESENTACIÓN
1:1	Uno a uno: La cardinalidad máxima en ambas direcciones es 1.	1  1
1:N	Uno a muchos: La cardinalidad máxima en una dirección es 1 y en la otra muchos.	1  N
N:M	Muchos a muchos: La cardinalidad máxima en ambas direcciones en muchos.	N  M

<p>Carlos Pérez (Tutor) 2ºA</p>  <p>UNO a UNO</p>		<p>UNO A UNO 1:1</p>
<p>DEPARTAMENTOS</p>  <p>UNO a MUCHOS</p>		<p>UNO A MUCHOS 1:N</p>



RESTRICCIONES

Integridad de entidad

Cada tabla debe tener una columna con valores únicos, no existen 2 filas o tuplas (cada una de las filas en una relación dentro de una tabla el mismo valor).

Integridad referencial

Los valores de una columna o una tabla deben coincidir con los valores de la columna de la tabla relacionada.

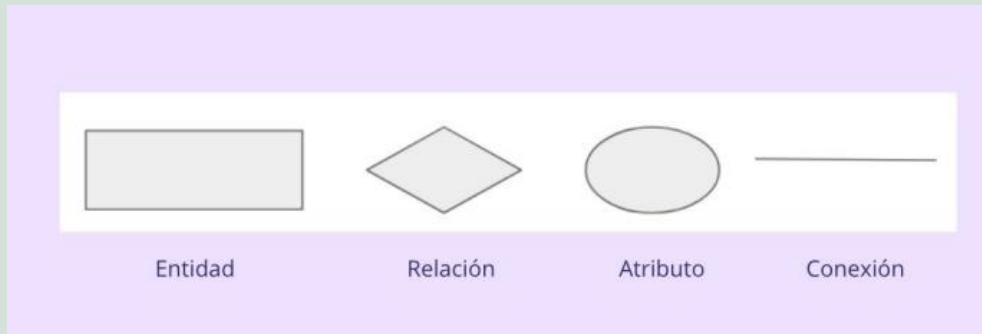


Integridad dominio

Es la validación de las entradas(valores) de una determinada columna de la tabla.

MODELO ENTIDAD- RELACION(E-R)

Visualiza los objetos que permanecen a la base de datos como entidades las cuales tienen atributos y se vinculan mediante relaciones.



Entidad: se representa mediante un rectángulo nominado y representa un conjunto de objetos.

Relación: es la asocian entre varias entidades representa mediante líneas y rombos intermedios.

Atributo: se representa un diagrama ER. Describen los elementos de una entidad.

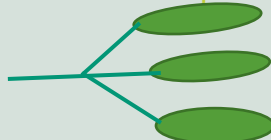
- **Atributo identificador:** son aquellos que identifican de forma única a una identidad.



- **Atributo descriptor:** el cual se representa con una eclipse y nombre sin subrayar.



- **Atributo compuesto o estructura:** es la etiqueta de una línea o árbol que se divide en tantos atributos simples como forma la estructura.



NORMALIZACIÓN

Elimina la redundancia en un diseño de tablas utilizando las restricciones o dependencias ente columnas.

Es el proceso de eliminación de redundancias en la tabla para que sea más fácil de manejar. Se ha desarrollado un sin números de formas reales para eliminar dicha redundancia.

Forma normal: regla que se emplea sobre las tablas, específicamente sobre las dependencias permisibles. Elimina cierto tipo de redundancias.

Inconsistencias: cuando dos instancias(campos) del mismo elemento no tienen valores iguales se determina que existe inconsistencia de datos.

Integridad: cuando una instancia (campo de un mandato) tiene valores raros o irreales.

PRIMERA FORMA NORMAL (1FN)

Se elimina todos los campos o atributos respectivos, el cual cada tabla deba tener una única tupla por lo cual se debe definir una llave primaria,

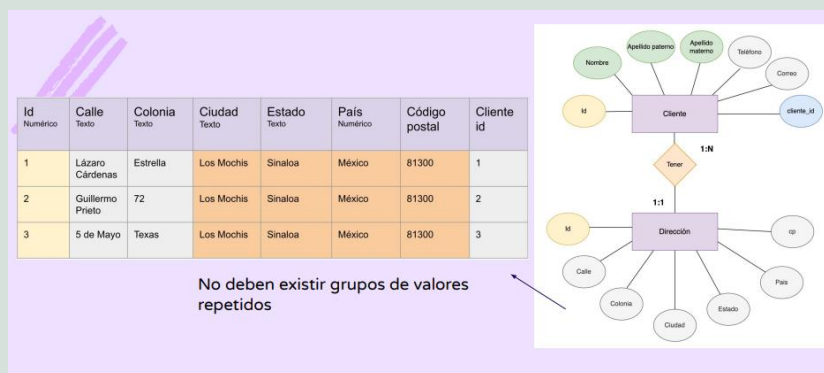
Campo Pk

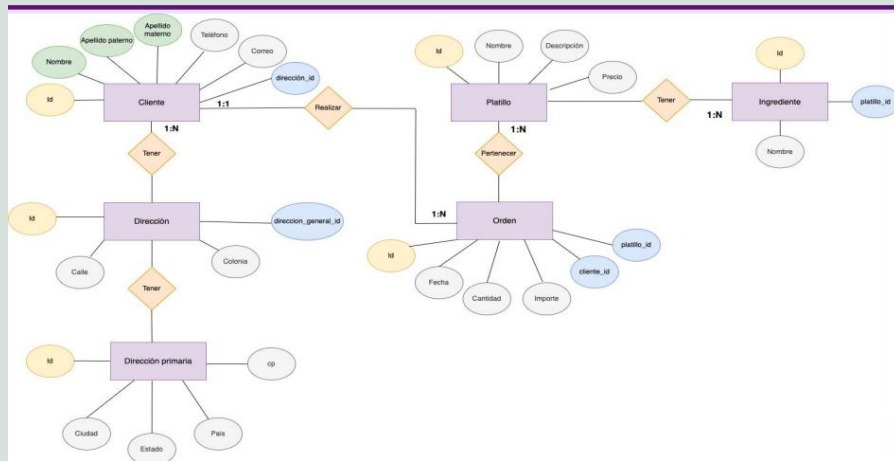
Dependencia funcional

Campo (solo tienen un valor dominio)

Primera Forma Normal

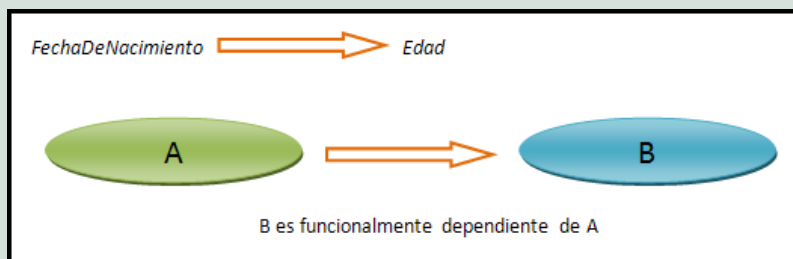
1. Todos los atributos, deben ser indivisibles
2. No deben existir grupos de valores repetidos





Dependencia funcional y transitiva

Funcional: representa que B es funcionalmente dependiente de A para el valor de A siempre pertenece un valor B



Transitiva: $A \rightarrow B \rightarrow C$

Si $A \rightarrow B$ (Depende B a A)

$B \rightarrow C$ (depende C a B)

C depende de forma transitiva de A



SEGUNDA FORMA FORMAL (2FN)

- Debe estar en primera forma normal

- No debe existir dependencias parciales. Los campos no llaves deben depender solo de la llave primaria.

Primera forma normal (1FN)

No deben existir dependencias parciales.

Segunda Forma Normal

1. Cualquier campo no clave debe ser dependiente de toda la clave primaria

Cuso	Fecha	Título	Aula	Capacidad
SQL101	2020/05/01	Fundamentos SQL	4A	12
DB202	2020/07/01	Diseño de base de datos	7B	14
SQL101	2020/06/01	Fundamentos SQL	7B	14
SQL101	2020/07/01	Fundamentos SQL	12A	8
POO200	2020/07/20	Programación Orientada a Objetos	5A	12

Fecha	Aula	Capacidad	curso_id	curso	título
2020/05/01	4A	12	SQL101	SQL101	Fundamentos SQL
2020/07/01	7B	14	DB202	DB202	Diseño de base de datos
2020/06/01	7B	14	SQL101		
2020/07/01	12A	8	SQL101		
2020/07/20	5A	12	POO200	POO200	Programación orientada a objetos

Dependencia parcial: es un termino que describe aquellos datos que no depende de la llave primaria de una tabla para poder identificar.

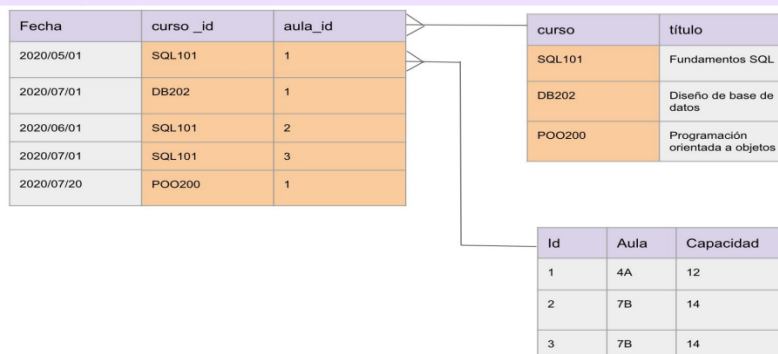
TERCERA FORMA NORMAL (3FN)

Debe estar en segunda forma normal no debe existir dependencias transitivas (ningún campo debe depender de no. Llaves).

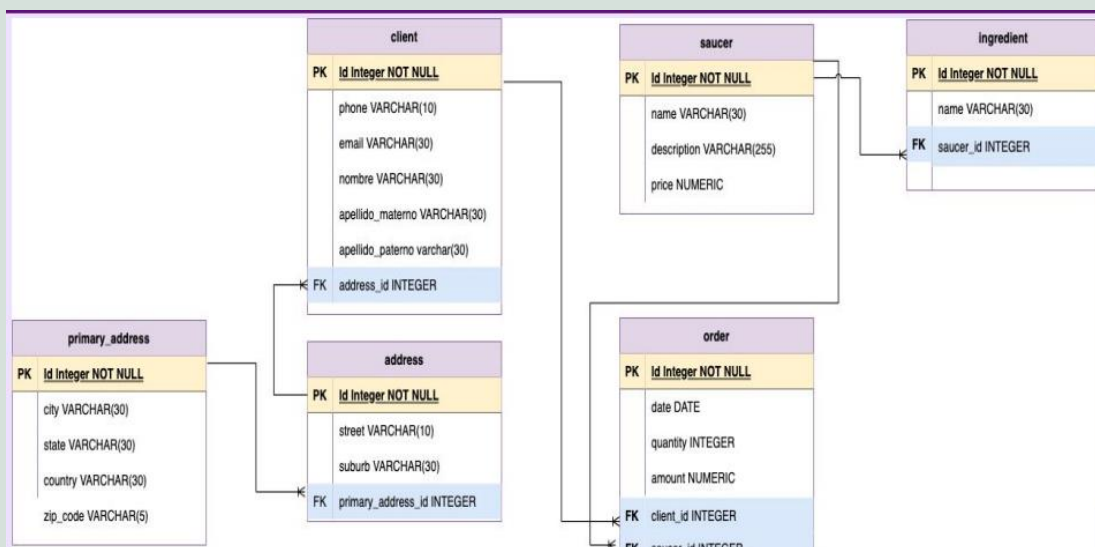
Tercera Forma Normal

1. Ningún campo no clave depende de ningún otro clave

Fecha	Aula	Capacidad	curso_id
2020/05/01	4A	12	SQL101
2020/07/01	7B	14	DB202
2020/06/01	7B	14	SQL101
2020/07/01	12A	8	SQL101
2020/07/20	5A	12	POO200



Diccionario de datos: informaciones referentes a la descripción de la estructura de base de datos.



BASE DE DATOS (2ª SESION) SQL

Lenguaje descriptivo que nos permitirá administrar y recuperar información de un sistema gestor de base de datos relacional



MANIPULACION DE DATOS

CONSULTAS:

Las consultas se utilizan para recuperar información en una o varias tablas obtenidas en una base de datos la estructura básica de una consulta está formada por las siguientes instrucciones.

1. **SELECT** se usa para listar los campos que se requieren en una consulta para obtener todos los campos de una tabla se coloca un *.
2. **FROM** Lista las relaciones que deben ser analizadas en la consulta. Con esta instrucción se indica la o las tablas de donde se obtendrá la información.
3. **WHERE** Esta instrucción permite filtrar o condicionar la información que se desea presentar.

Si se involucra más de la tabla o el nombre de la tabla es demasiado largo, se pueden utilizar identificadores que hagan diferencia a cada objeto (tabla) involucrado-

SINTAXIS

SELECT <Lista campo>

FROM <Lista -tablas>

WHERE [<condición>]

<p>Ejemplo 1</p> <pre>SELECT * FROM ALUMNO;</pre>	<p>Ejemplo 2</p> <pre>SELECT MATRICULA, NON-ALU FROM ALUMNO</pre>
<p>Ejemplo 3</p> <pre>SELECT <u>ALU</u>.MATRICULA, <u>ALU</u>.NOM_ALU FROM ALUMNO <u>ALU</u>;</pre>	<p>Ejemplo 4</p> <pre>SELECT * FROM ALUMNO <u>ALU</u>, MATERIA <u>ALU</u> WHERE <u>ALU</u> MATRICULA= 1760010022;</pre>

INSERTAR:

Para **INSERTAR** datos en una tabla en **BDR**, se debe especificar el nombre de la table en la que se desea agregar información la inserción puede ser individual (renglón por renglón) o se puede formular una consulta cuyo resultado sean los datos que se requieren agregar a dicha tabla.

Pue poder insertar registros en una table existen 2 formas básicas para hacerlo:

1. **INSERCIÓN** indicando los atributos o campo de la tabla. Indicando los campos de la tabla

Sintaxis

```
INSERT INTO <nom tablas>(<nom_campos>) VALUES (<valores_camps>);
```

Ejemplo

```
INSERT INTO ALUMNO (MATRICULA, APEPAT-ALU, APEMAT_ALU, NOM_ALU
VALUES (2000001, 'PEREZ','GARCIA','PABLO');
```

2. **INSERCIÓN DE REGISTROS** donde los valores se especifican en el mismo orden de la tabla que aparecen en la tabla. Mismo orden de la tabla.

Sintaxis

```
INSERT INTO <nom tabla > VALUES (<valores_camps>);
```

EJEMPLO:

INSERT INTO ALUMNO.

VALUES (201700002,' IRENE',' AGUILAR', 'LOPEZ');

MODIFICACIÓN

Para actualizar o modificar registros, se utilizan las instrucciones UPDATE y SET. La instrucción SET es utilizada para indicar el nuevo valor que contendrá un campo.

Sintaxis:

UPDATE <nombre_tabla>

SET <nom_campo>= <nuevo_valor>

[WHERE <condicion>];

Ejemplo

UPDATE ALUMNO

SET NOM_ALU = 'GUADALUPE'

WHERE MATRICULA = 201700002;

ELIMINACION

Para borrar registros en una tabla se utiliza la instrucción. DELETE, indicando la tabla con la instrucción FROM, también se puede indicar un filtro a condición para eliminar solo un conjunto de registros utilizando la instrucción WHERE.

Sintaxis.

DELETE FROM <nom tablas> [WHERE <Condicion>];

Ejemplo 1

DELETE FROM ALUMNO;

Ejemplo 2

DELETE FROM ALUMNO WHERE MATRICULA = 2017000032;

Ejemplo 3

```
DELETE FROM ALUMNO
```

```
WHERE MATRICULA >=201700001
```

```
AND MATRICULA <= 2017600009;
```

DEFINICIÓN DE DATOS

CREATE:

Este comando crea un objeto dentro del gestor de base de datos. Puede ser una base de datos, tabla, índice, procedimiento almacenado o vista.

Creamos tablas

```
CREATE TABLE nombre_tabla (nombre_campo tipo_dato);
```

ALTER:

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.

Agregamos Foreign Keys

```
ALTER TABLE nombre_tabla ADD CONSTRAINT nombre_constraint  
FOREIGN KEY columna REFERENCES tabla;
```

DROP:

Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

```
ALTER TABLE nombre_tabla DROP COLUMN  
nombre_columna;
```

CONTROL DE DATOS

GRANT:

Para otorgar privilegios a un usuario o un grupo. El sistema añade estos privilegios a cualquier otro privilegio que el usuario o el grupo tenga ya.

Para otorgar privilegios solo a unas pocas columnas, cree una vista que tenga esas columnas y después otorgue privilegios a esa vista.

Sintaxis:

GRANT ALL PRIVILEGES

ON TABLE <tabla1>, <tabla2>

[WITH GRANT OPTION] TO <usuario1>, <usuario2> ;

REVOKE:

Para revocar privilegios de un usuario o un grupo o para revocar la capacidad de un usuario o un grupo de conceder privilegios a otros usuarios.

Sintaxis:

REVOKE <derecho1>, <derecho2>, ...

ON TABLE <nombre tabla>

FROM <usuario1>, <usuario2> ...;

UNIONES (joins)

La operación o instrucción JOINS (combinación) permite mostrar columnas de columnas o varias tablas como si se tratase de sola combinando entre si los registros relacionadas a través de llaves foráneas.

Las tablas solamente se indican en la cláusula a instrucción FROM, además hay que hacer coincidir los valores que relacionan las columnas en cada tabla. Para evitar que se produzca como resultado in producto cartesiano entre dos o más tablas se debe indicar dentro de la instrucción WHERE o JOIN.


```

SELECT ALU.MATRICULA
        ALU.NOM_ALU
        ALU.APEPAT_ALU
        ALU.APEMAT_ALU
FROM    ALUMNO ALU
        PASATIEMPOS PAS
WHERE ALU. MATRICULA = PAS MATRICULA
        AND PAS.NOM_PAS = 'LECTURU';

```

INNER JOIN

Las combinaciones internas se realizan mediante las instrucciones INNER JOIN devolviendo solamente aquellos registros o filas que tienen valores idénticos en los dos campos que se comparan -para unir ambas tablas.

TABLA 1:

```

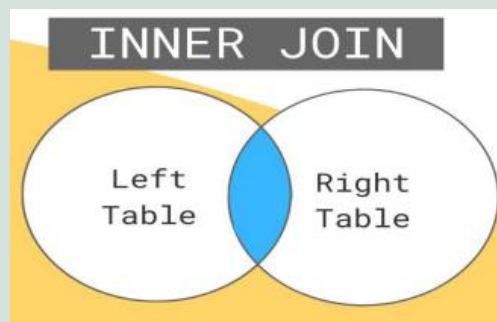
SELECT    ALU. MATRICULA
          ALU. NOM_ALU
          ALU APEPAT_ALU
          ALU. APEMAT ALU

```

```

FROM      ALUMNO ALU
INNER JOINS PASATIEMPOS PAS
          ALU.MATRICULA=PAS. MATRICULA;
          PAS.NON_PAS = "LECTURA";

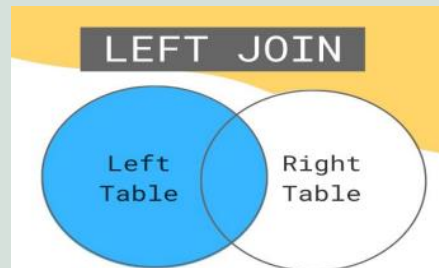
```



LEFT JOIN

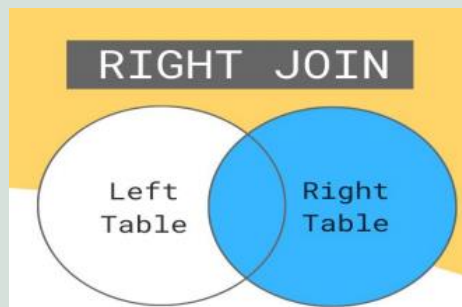
Se obtienen todas las filas colocadas a la izquierda junto, con la intersección.

Los datos obtenidos de la tabla de la izquierda se muestran, aunque no tenga correspondencia con la tabla de la derecha



RIGHT JOIN

Con esta instrucción se obtiene todos los registros de la tabla de la derecha junto con los datos de intersección.



FULL JOIN

Ejemplo:

```
SELECT ALU.NOM-ALU, ALU. MATRICULA PAS.NOM_PAS  
FROM ALUMNO ALU  
FULL JOIN PASATIEMPO PAS  
ON ALU MATRICULA =PAS.MATRICULA;
```

