1. **SELECT:**

##### Seleccionamos x datos de una base de datos

SELECT *column1*, *column2, ...*

FROM *table\_name*;

SELECT \* FROM *table\_name*;

## SELECT DISTINCT:

##### Seleccionará x datos, pero no seleccionará los duplicados.

SELECT DISTINCT *column1*, *column2, ...*

FROM *table\_name*;

SELECT *column* FROM *table\_name*;

## INSERT INTO:

##### Agregara valores a las columnas correspondientes

INSERT INTO *table\_name* (*column1*, *column2*, *column3*, ...) VALUES (*value1*, *value2*, *value3*, ...);

## Create database:

CREATE DATABASE *databasename*;

## CREATE TABLE:

##### Lo usaremos para crear una nueva tabla en la base de datos.

CREATE TABLE *table\_name* ( *column1 datatype*, *column2 datatype*, *column3 datatype*,

....

);

## DESC

##### Muestra las columnas sin atributos Embudo N con circulo 0

Para foreing keys tenemos que utilizar una de la que hereda

## DROP TABLE:

##### Lo usaremos para borrar una tabla.

DROP TABLE *table\_name*;

## TRUNCATE TABLE:

##### Borra los elementos dentro de la tabla, pero no la tabla en sí.

TRUNCATE TABLE *table\_name*;

## ALTER TABLE:

##### Se usa para añadir/eliminar o modificar columnas de una tabla.

ALTER TABLE *table\_name*

ADD *column\_name datatype*;

ALTER TABLE *table\_name*

DROP COLUMN *column\_name*;

ALTER TABLE *table\_name*

MODIFY COLUMN *column\_name datatype*;

**Añadir clave Primaria**

ALTER TABLE *table\_name*

ADD PRIMARY KEY (*column\_name*);

ALTER TABLE *table\_name*

DROP PRIMARY KEY;

#### Añadir clave foránea

ALTER TABLE *table\_name*

ADD CONSTRAINT FK\_keyname

ADD FOREIGN KEY (*column\_name*) REFERENCES *table\_name*

(*column\_name*);

ALTER TABLE *table\_name*

DROP FOREIGN KEY key\_name;

## WHERE:

##### Lo usaremos para seleccionar filas y columnas en los que se cumpla una condición podremos usar condiciones lógicas como: and or;

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition*;

SELECT \* FROM *table\_name*

WHERE Country='Mexico';

## Operaciones logicas

**AND** Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* AND *condition2* AND *condition3 ...*;

**OR** Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* OR *condition2* OR *condition3 ...*;

**NOT** Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE NOT *condition*;

## BETWEEN Operator

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* BETWEEN *value1* AND *value2;*

SELECT \* FROM Products

WHERE Price BETWEEN 10 AND 20;

Valor IS NULL devolverá a true si es NULL;

## Order By

SELECT *column1*, *column2, ...*

FROM *table\_name*

ORDER BY *column1, column2,*

*...* ASC(ascendiente)|DESC(descendiente)s;

## Group By

Podremos agrupar distintos **(COUNT(), MAX(), MIN(), SUM(), AVG())**

SELECT *column\_name(s)* FROM *table\_name* WHERE *condition*

GROUP BY *column\_name(s)*

ORDER BY *column\_name(s);*

## COUNT(), MAX(), MIN(), SUM(), AVG()

#### Nos contará

SELECT COUNT(*column\_name*) FROM *table\_name*

WHERE *condition*;

#### Hará la media

SELECT AVG(*column\_name*) FROM *table\_name*

WHERE *condition*;

#### Sumará

SELECT SUM(*column\_name*) FROM *table\_name*

WHERE *condition*;

#### Devuelve el mínimo

SELECT MIN(*column\_name*) FROM *table\_name*

WHERE *condition*;

**Devuelve el máximo**

SELECT MAX(*column\_name*) FROM *table\_name*

WHERE *condition*;

## Having

##### La cláusula HAVING se usa porque la palabra clave WHERE no se puede usar con funciones agregadas.

SELECT *column\_name(s)* FROM *table\_name* WHERE *condition*

GROUP BY *column\_name(s)*

HAVING *condition*

ORDER BY *column\_name(s);*

SELECT COUNT(CustomerID), Country FROM Customers

GROUP BY Country

HAVING COUNT(CustomerID) > 5;

## Aliases

SELECT *column\_name(s)*

FROM *table\_name* AS *alias\_name;*

SELECT o.OrderID, o.OrderDate, c.CustomerName FROM Customers AS c, Orders AS o

WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;

## Update

UPDATE *table\_name*

SET *column1* = *value1*, *column2* = *value2*, ... WHERE *condition*;

## Union

##### La Unión ya nos asegura que no se repiten

SELECT *column\_name(s)* FROM *table1*

UNION

SELECT *column\_name(s)* FROM *table2*;

SELECT *column\_name(s)* FROM *table1*

UNION ALL

SELECT *column\_name(s)* FROM *table2*;

SELECT *column\_name(s)* FROM *table1*

UNION

SELECT *column\_name(s)* FROM *table2*

ORDER BY *column\_name(s)* ;

SELECT City, Country FROM Customers WHERE Country='Germany'

UNION

SELECT City, Country FROM Suppliers WHERE Country='Germany'

ORDER BY City;

## Join

##### Una JOIN cláusula se utiliza para combinar filas de dos o más tablas, según una columna relacionada entre ellas.

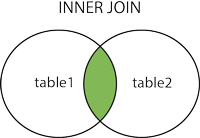
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.Customeri D;

## Inner join

##### La INNER JOIN palabra clave selecciona registros que tienen valores coincidentes en ambas tablas.



SELECT *column\_name(s)*

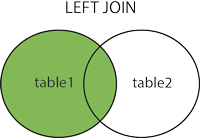
FROM *table1*

INNER JOIN *table2*

ON *table1.column\_name* = *table2.column\_name*;

## LEFT JOIN

##### La LEFT JOIN palabra clave devuelve todos los registros de la tabla izquierda (tabla1) y los registros coincidentes de la tabla derecha (tabla2). El resultado es 0 registros del lado derecho, si no hay coincidencia.



SELECT *column\_name(s)*

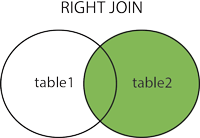
FROM *table1*

LEFT JOIN *table2*

ON *table1.column\_name* = *table2.column\_name*;

## RIGHT JOIN

##### La RIGHT JOIN palabra clave devuelve todos los registros de la tabla derecha (tabla2) y los registros coincidentes de la tabla izquierda (tabla1). El resultado es 0 registros del lado izquierdo, si no hay coincidencia.



SELECT *column\_name(s)*

FROM *table1*

RIGHT JOIN *table2*

ON *table1.column\_name* = *table2.column\_name*;

# VISTAS

1. **Create View**

CREATE VIEW *view\_name* AS SELECT *column1*, *column2*, ... FROM *table\_name*

WHERE *condition*; **25.Insert,Delete,Update** UPDATE *officeInfo*

SET

*phone = '+33 14 723 5555'*

WHERE

*officeCode = 4;*

## Drop View

DROP VIEW *view\_name*;

## Show full tables

SHOW FULL TABLES

WHERE table\_type = 'VIEW';

## Rename Table

RENAME TABLE original\_view\_name TO new\_view\_name

# Índices

1. **Create Index**

CREATE INDEX *index\_name*

ON *table\_name* (*column1*, *column2*, ...);

## Drop Index

DROP INDEX *index\_name* ON *table\_name*;

## Show Indexes

SHOW INDEXES FROM **table\_name;**

# Procedimientos almacenados

## Create Procedure

CREATE PROCEDURE procedure\_name(parameter\_list) BEGIN

statements; END //

## Delimiter

DELIMITER $$

CREATE PROCEDURE sp\_name() BEGIN

-- statements END $$

DELIMITER ;

## Drop Procedure

Drop Procedure procedureName**;**

## 35.IN

DELIMITER //

CREATE PROCEDURE GetOfficeByCountry( IN countryName VARCHAR(255)

) BEGIN

SELECT \*

FROM offices

WHERE country = countryName;

END //

DELIMITER ;

## Out

DELIMITER $$

CREATE PROCEDURE GetOrderCountByStatus ( IN orderStatus VARCHAR(25),

OUT total INT

) BEGIN

END$$

SELECT COUNT(orderNumber) INTO total

FROM orders

WHERE status = orderStatus;

DELIMITER ;

## Variables

DECLARE variable\_name datatype(size) [DEFAULT default\_value];

## Cambios

[https://www.mysqltutorial.org/mysql-stored-procedure/alter-stored-](https://www.mysqltutorial.org/mysql-stored-procedure/alter-stored-procedure/) [procedure**/**](https://www.mysqltutorial.org/mysql-stored-procedure/alter-stored-procedure/)

## Listing

SHOW PROCEDURE STATUS;

## If state

### IF condition THEN statements;

END IF;

## Case Statement

CASE case\_value

WHEN when\_value1 THEN ... WHEN when\_value2 THEN ...

END CASE;

## While

WHILE search\_condition DO statement\_list

END WHILE;

## Repeat

REPEAT

statement

UNTIL search\_condition END REPEAT

## Cursor

DECLARE cursor\_name CURSOR FOR SELECT\_statement;

# TRIGGER

## Create trigger

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE| DELETE }

ON table\_name FOR EACH ROW

trigger\_body;

# Error handling

DELIMITER //

CREATE TRIGGER before\_phone\_insert BEFORE INSERT ON offices FOR EACH ROW BEGIN

IF new.phone IS null THEN SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = "No puede contener valor

nulo";

END IF;

END // DELIMITER