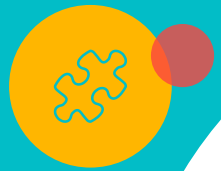
Apuntes sobre Bases de Datos

Miguel Troyano Redondo



1.
Operaciones DDL

Data Definition Language

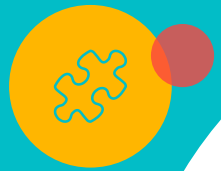
Miguel Troyano Redondo

¿Qué son las operaciones DDL?

Data Definition Language
La principal misión de estas operaciones consiste en la creación, modificación y borrado de las tablas que componen una base de datos, asi como sus indices, vistas, sinonimos, etc.

En esta sección veremos:

- Create
- Alter
- Drop



1.1
CREATE

Miguel Troyano Redondo

Create Database

Create Database

Este comando crea una base de datos vacia.

CREATE DATABASE *databasename*;

```
create database CAMPEONATO
CHARACTER SET latin1
COLLATE latin1_spanish_ci;
```

En este ejemplo vemos como se crea la base de datos campeonato estableciendo su conjunto de caracteres en latin1 y su intercalación (estos dos ultimos son opcionales)

Antes de realizar cualquier operación en una base de datos, es necesario indicarle cuál se va utilizar. Para ello, utilizamos el comando ***use*** *seguido del nombre.*

```
use CAMPEONATO;
```

Create Table

Create Table

Este comando crea una nueva tabla en la base de datos seleccionada.

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
   ....
);
```

En este ejemplo se crea la tabla hotel con dos columnas. Una se llamara codigo y otra hotel y ambas son de tipo varchar, cada una con su longitud.

Ademas, opcionalmente se le puede indicar la intercalación y el motor que va utilizar para crearla.

```
CREATE TABLE hotel (
        codigo varchar (5),
        nombre varchar (25)
        )
COLLATE=latin1_spanish_ci
ENGINE=InnoDB;
```

A cada columna es obligatorio especificarle su tipo de datos. Existen muchos, pero los más comunes son:

Char(n): de tipo texto y ocupación fija
Varchar(n): de tipo texto y ocupación variable.
Integer(n): de tipo numérico.
Date: de tipo fecha.
Decimal (n,m): de tipo numérico con decimales.
Timestamp: de tipo fecha y hora.

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

```
CREATE TABLE prueba (
    codigo varchar (5) not null,
    nombre varchar (25),
    apellidos char (25),
    capacidad integer (8),
    f_inaguracion date,
    dias_promedio decimal(5,2)
);
```

- NOT NULL – se olbiga a que la columna nunca pueda estar vacia.
- DEFAULT value – se le especifica un valor por defecto.
- AUTO INCREMENT – va incrementando de uno en uno su valor.

Cada tabla puede tener claves primarias (identificador único de la tabla) y claves ajenas o foráneas (claves primarias externas).

```
CREATE TABLE jugar (
        num_asoc varchar(5),
        cod_partida varchar(5),
        color_ficha varchar(20),
        PRIMARY KEY (num_asoc,cod_partida),
        CONSTRAINT FK_Jugar_Participantes
        FOREIGN KEY(num_asoc)
        REFERENCES jugadores(num_asoc)
        On delete restrict on update cascade
        );
```

- Campo o campos que formaran la clave primaria
- Nombre de la clave ajena
- Campo a referenciar de la tabla actual
- Tabla y campo ajeno donde buscará los valores
- Condición para actualizar y borrar en cascada

En este ejemplo vemos como se crea la tabla jugar con tres columnas de tipo varchar. Además establecemos que su clave primaria serán los campos num_asoc y cod_partida.
Su clave ajena se llamara FK_Jugar_Participantes y enlazara el campo num_asoc de esta tabla con num_asoc de la tabla jugadores.

Create View

Create View

Este comando crea una nueva vista de una o varias tablas.

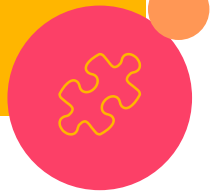
```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

En este ejemplo se crea la vista llamada vHotel, que será el resultado de seleccionar el nombre y apellidos de la tabla hotel.

Se puede hacer vistas de una o varias tablas.

```
CREATE VIEW vHotel AS
SELECT nombre, apellidos
FROM hotel ;
```

Una vista nunca almacena datos, solo guarda la estructura de la tabla. Por tanto, no se pueden hacer inserts directamente sobre la vista.



Create Or Replace view

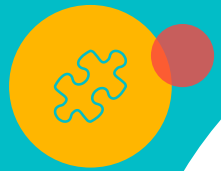
Create o Replace View

Puedes actualizar una vista usando este comando.

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

En este ejemplo se modifica la vista vHotel y ahora tiene la condicion de que la localidad sea igual a Lleida.

```
CREATE OR REPLACE VIEW vHotel  AS
SELECT h.*
FROM hotel h
WHERE Localidad = 'Lleida';
```



1.2
ALTER

Miguel Troyano Redondo

Alter

Alter table

Este comando modifica una tabla que ya ha sido creada.

Se pueden modificar las tablas agregando o borrando columnas, claves primarias o ajenas o incluso modificar columnas.

Cuando la tabla ya esta creada es posible crear, borrar o modificar columnas.

Añadir columna:

```
ALTER TABLE table_name
ADD column_name datatype;
```

```
ALTER TABLE jugar
ADD partidos integer(6);
```

Borrar columna:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

```
ALTER TABLE jugar
DROP COLUMN partidos ;
```

Modificar columna:

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

```
ALTER TABLE jugar
MODIFY COLUMN partidos integer(10) ;
```

Normalmente las claves primarias y ajenas se suelen definir en la creación de la tabla, no obstante, también es posible definirlas o borrarlas cuando la tabla ya este creada.

Clave primaria:

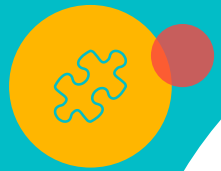
```
ALTER TABLE jugar ADD PRIMARY KEY (num_asoc);
```
→ Crea clave primaria

```
ALTER TABLE jugar DROP PRIMARY KEY;
```
→ Borra clave primaria

Clave ajena:

```
ALTER TABLE jugar
ADD CONSTRAINT FK_Jugar_Partida
FOREIGN KEY (num_asoc)
REFERENCES jugadores(num_asoc)
ON DELETE RESTRICT ON UPDATE CASCADE;
```
→ Crea clave ajena

```
ALTER TABLE jugar  DROP FOREIGN KEY FK_Jugar_Partida;
```
→ Borra clave ajena



1.3
DROP

Miguel Troyano Redondo

Drop

Drop Database

Este comando borra una base de datos.

DROP DATABASE *databasename*;

DROP DATABASE campeonato;

Drop Table

Este comando borra una tabla.

DROP TABLE *table_name*;

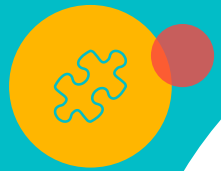
DROP TABLE hotel;

Drop View

Este comando borra una vista.

DROP VIEW view_name;

DROP VIEW vHotel;



2.
Operaciones DML

Data Manipulation Language

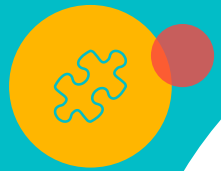
Miguel Troyano Redondo

¿Qué son las operaciones DML?

Data Manipulation Language
La principal misión de estas operaciones consiste en la consulta, insercción, modificación de los datos de una tabla.

En esta sección veremos:

- Select / From
  - Alias
  - Funciones
  - Joins
  - Subconsultas
- Insert
- Update
- Delete
- Truncate
- Transaction



2.1
SELECT / FROM

Miguel Troyano Redondo

SELECT

Select

Esta instrucción se utiliza para indicar que columnas se quieren mostrar.

```
SELECT column1, column2, ...
FROM table_name;
```

```
SELECT *
FROM pais;
```

cod_pais	nombre	continente	clubes
p001	Rusia	Europa	10
p002	Francia	Europa	3
p003	Guayana Francesa	America	1
p004	Uzbekistan	Asia	8
p005	Nigeria	Africa	14

Select distinct

Esta instrucción se utiliza para indicar que columnas se quieren mostrar, pero solo mostrara los valores diferentes.

```
SELECT distinct nombre
FROM pais
```

FROM

From

Esta instrucción se utiliza para indicar de que tabla o tablas se deben utilizar para obtener los datos.

```
SELECT column1, column2, ...
FROM table_name;
```

```
SELECT *
FROM pais;
```

```
SELECT *
FROM pais, hotel;
```

WHERE

Where

Esta instrucción se utiliza para filtrar o relacionar tablas.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
SELECT *
FROM pais
where nombre = 'rusia'
```

Como norma general, los textos van entre comillas simples y los números sin ellas.

Operador igual (=): nos permite comparar el campo con el valor que le indiquemos.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition = 'texto';
```

```
Select *
From pais
Where nombre = 'Rusia';
```

cod_pais	nombre	continente	clubes
p001	Rusia	Europa	10

Operador distinto (<>): nos permite comparar el campo con el valor que le indiquemos, mostrando los que no son iguales.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition <> 'texto';
```

```
Select *
From pais
Where nombre <> 'Rusia';
```

cod_pais	nombre	continente	clubes
p002	Francia	Europa	3
p003	Guayana Francesa	America	1
p004	Uzbekistan	Asia	8
p005	Nigeria	Africa	14

Operador menor, mayor, menor que o mayor que (<,>,<=,>=): nos permite comparar el campo con el valor que le indiquemos.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition < X;
```

```
Select *
From pais
Where clubes > 8;
```

cod_pais	nombre	continente	clubes
p001	Rusia	Europa	10
p005	Nigeria	Africa	14

Operador like / not like: nos permite mostrar los valores que cumplan o no la condición.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition like '%texto%';
```

```
Select *
From pais
Where nombre like '%e%';
```

cod_pais	nombre	continente	clubes
p003	guayana francesa	america	1
p004	uzbekistan	asia	8
p005	nigeria	africa	14

El operador like / not like admite el comodin % para indicarle si puede tener algun otro caracter por delante o por detras .

Operador AND: nos permite comparar por varios campos. Se tienen que cumplir todas las condiciones.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition > X and condition < X ;
```

```
Select *
From pais
Where clubes > 1 and clubes < 11;
```

cod_pais	nombre	continente	clubes
p001	rusia	europa	10
p002	francia	europa	3
p004	uzbekistan	asia	8

Operador OR: nos permite comparar por varios campos.
Se tienen que cumplir una o varias condiciones

```
SELECT column1, column2, ...
FROM table_name
WHERE condition > X OR condition < X ;
```

```
Select *
From pais
Where nombre = 'rusia' OR nombre = 'francia';
```

cod_pais	nombre	continente	clubes
p001	rusia	europa	10
p002	francia	europa	3

GROUP BY

Group by

Esta clausula se utiliza para agrupar filas, normalmente se utiliza con funciones de resumen.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

```
select medio, count(*)
from salas
group by medio;
```

medio	count(capacidad)
audio	1
todos	3
video	1

HAVING

Having

Esta clausula se utiliza para poder utilizar funciones de resumen como condición de filtro.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
```

```
select medio, sum(capacidad)
from salas
where capacidad > 16
group by medio
having count(medio) >= 2;
```

medio	sum(capacidad)
todos	170

ORDER BY

Order By

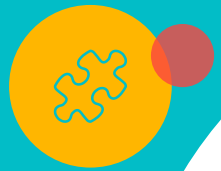
Utilizamos order by para poder ordenar los registros que nos van aparecer en pantalla.

```
SELECT column_name(s)
FROM table_name
WHERE condition
ORDER BY column_name(s) asc;
```

```
select *
from pais
where nombre like '%e%'
Order by nombre asc;
```

cod_pais	nombre	continente	clubes
p003	guayana francesa	america	1
p005	nigeria	africa	14
p004	uzbekistan	asia	8

Si se quiere ordenar de forma descendente se debe cambiar asc por desc.



2.1.1
ALIAS

Miguel Troyano Redondo

ALIAS

¿Qué es un alias?

Un alias es un nombre virtual que se le asigna a una columna, tabla o consulta.

Se utilizan para acortar los nombres de las tablas, cambiar el nombre de las columnas que se muestran, utilizar una misma tabla varias veces o para darle nombre a una consulta.

Alias de columna: nos permite renombrar virtualmente la columna.

```
SELECT column_name AS alias_name
FROM table_name;
```

```
Select nombre as PAIS
From pais;
```

PAIS
rusia
francia

Alias de tabla: nos permite renombrar virtualmente una tabla.

```
SELECT column_name(s)
FROM table_name AS alias_name
WHERE condition;
```

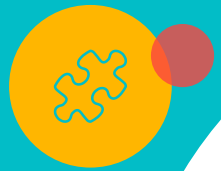
```
Select nombre as PAIS
From pais
where h.nombre='hotel';
```

```
select h.nombre, s.nombre
from hotel h inner join salas s on h.codigo=s.codigo
where h.nombre='hotel central'
```

Alias de tabla: nos permite renombrar virtualmente una tabla.

```
select h.nombre, s.capacidad, T2.capacidad
from hotel h, salas s, (select codigo, capacidad+1 capacidad
                        from salas) T2
where h.codigo = T2.codigo
and h.codigo=s.codigo
group by h.codigo
```

nombre	capacidad	capacidad
hotel central	32	33
hotel ilerna	16	17
hotel real	80	81



2.1.2
FUNCIONES

Miguel Troyano Redondo

FUNCIONES

¿Qué es una función?

Es una rutina creada para tomar unos parámetros, procesarlos y retornar en un salida.

Las funciones de resumen más comunes son min, max, count, avg, sum, pero existe otras funciones como truncate o round.

COUNT

La función count() es simplemente un contador. Si escribimos count(*) nos contara todas las filas. Si por el contrario escribimos count(campo) nos contara cuantas veces aparece y por ultimo, si escribimos count (distinct campo) nos mostrara las veces que aparece sin repetirse.

```
SELECT count(column_name)
FROM table1;
```

```
SELECT column_name(s), count(column_name)
FROM table1
GROUP BY column_name(s);
```

```
SELECT count(distinct continente)
FROM pais
```

count(distinct continente)
4

```
SELECT count(*)
FROM pais
```

count(*)
5

SUM / AVG / MIN / MAX

Todo este tipo de funciones se utilizan del mismo modo. Se escribe el nombre de la función seguido del campo a utilizar entre paréntesis. Siempre sobre campos numéricos.

```
SELECT sum(column_name)
FROM table1;
```

```
SELECT column_name(s), sum(column_name)
FROM table1
GROUP BY column_name(s);
```

```
SELECT sum(clubes)
FROM pais
```

sum(clubes)
36

```
SELECT max(clubes)
FROM pais
```

max(clubes)
14

TRUNCATE / ROUND

Ambas funciones se utilizan para quitar decimales, pero lo hace de forma muy distinta.
La función truncate elimina los decimales de un campo mostrando solo la parte entera, sin embargo, la función round muestra la parte entera pero aproximando hacia arriba o hacia abajo.

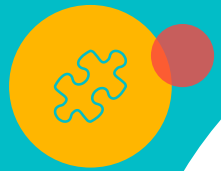
```
SELECT truncate(column_name,n)
FROM table1;
```

```
SELECT round(column_name)
FROM table1;
```

```
select numero1, truncate(numero1,0), round(numero1)
from numeros
```

numero1	truncate(numero1,0)	round(numero1)
5,50	5	6
5,20	5	5
5,60	5	6

Si se utiliza una función y se muestra cualquier otro campo se debe agrupar por todos los campos del select (excepto el de la función) para no repetir información.



2.1.3
JOINS

Miguel Troyano Redondo

JOINS

Joins

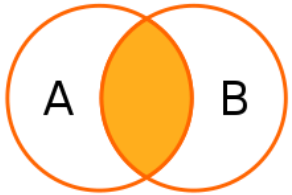
Los joins se utilizan para unir varias tablas en una misma consulta.

Existen diferentes tipos de joins, pero los más habituales son inner join, left join y right join.

Se pueden combinar entre ellos o utilizar varios del mismo tipo en una misma consulta.

INNER JOIN

Este tipo de join se utiliza para unir una o varias tablas a otra de la consulta. El resultado de este join serán todos aquellos registros que coincidan entre las tablas. En el grafico se utilizan las tablas A y B, el resultado será todo lo que coincida.

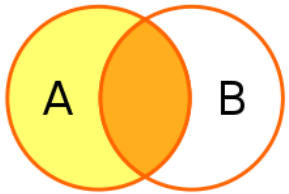



```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

```
Select distinct h.nombre
from hotel h inner join alojar a on h.codigo = a.codigo
             inner join participantes p on a.num_asoc = p.num_asoc;
```

LEFT JOIN

Este tipo de join se utiliza para unir una o varias tablas a otra de la consulta. El resultado de este join serán todos aquellos registros que coincidan entre las tablas y todos los de la tabla izquierda. En el grafico se utilizan las tablas A y B, el resultado será todo lo que coincida y todo lo de la tabla A.

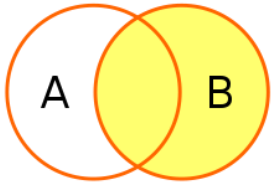



```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

```
Select distinct h.nombre
from hotel h left join alojar a on h.codigo = a.codigo
             left join participantes p on a.num_asoc = p.num_asoc;
```

RIGHT JOIN

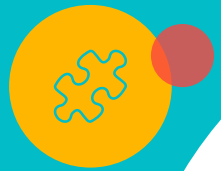
Este tipo de join se utiliza para unir una o varias tablas a otra de la consulta. El resultado de este join serán todos aquellos registros que coincidan entre las tablas y todos los de la tabla derecha. En el grafico se utilizan las tablas A y B, el resultado será todo lo que coincida y todo lo de la tabla B.




```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

```
Select distinct h.nombre
from hotel h right join alojar a on h.codigo = a.codigo
                right join participantes p on a.num_asoc = p.num_asoc;
```

Si se utilizan más de una tabla en una consulta, TODAS deben estar relacionadas al menos con otra tabla. No puede haber tablas sin relacionar.



2.1.4
SUBCONSULTAS

Miguel Troyano Redondo

SUBCONSULTAS

¿Qué es una subconsulta?

Una subconsulta es una o varias consultas dentro de otra consulta.

Se pueden utilizar tanto en el *from* como en el *where*.

En este último se acompañan de diferentes operadores y solo pueden devolver un valor.

SUBCONSULTA - IGUAL

En este tipo de subconsulta se iguala un campo a otro.

```
SELECT column_name
FROM table1
WHERE column_name = (SELECT column_name
                     FROM table1
                     WHERE column_name);
```

```
select nombre
from pais
where nombre = (select nombre
                from pais
                where nombre like '%ria');
```

nombre
Nigeria

SUBCONSULTA - IN

En este tipo de subconsulta se compara un campo con el resultado de otra consulta.

```
SELECT column_name
FROM table1
WHERE column_name in (SELECT column_name
                      FROM table1
                      WHERE column_name);
```

```
select nombre
from pais
where nombre IN (select nombre
                 from pais);
```

nombre
Rusia
Francia
Guayana Francesa
Uzbekistan
Nigeria

SUBCONSULTA – EXISTS / NOT EXISTS

En este tipo de subconsulta se comprueba si existe con el resultado de otra consulta.

```
SELECT column_name
FROM table1 t1
WHERE EXISTS (SELECT column_name
              FROM table2 t2
              WHERE t1.column_name=t2.column_name);
```

```
select nombre
from pais p
where exists  (select pais
               from participantes pa
               where p.cod_pais = pa.pais
                and apellidos like '%a')
```

nombre
Guayana Francesa
Uzbekistan

SUBCONSULTA – ANY

En este tipo de subconsulta se comprueba si un campo es igual a algun valor del resultado de otra consulta.

```
SELECT column_name
FROM table1 t1
WHERE column_name = ANY (SELECT column_name
                         FROM table2 t2);
```

```
select distinct  p.nombre, p.apellidos
from participantes p inner join jugar j on p.num_asoc=j.num_asoc
where p.num_asoc = ANY (select num_asoc
                        from jugadores);
```

nombre	apellidos
JOSE LUIS	LOPEZ VAZQUEZ
SERGIO	RAMON GARCIA
MICKEY	MOUSE LOPEZ

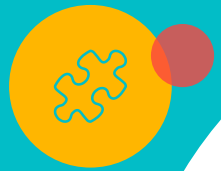
SUBCONSULTA – ALL

En este tipo de subconsulta se comprueba si un campo es igual a todos los valores del resultado de otra consulta.

```
SELECT column_name
FROM table1 t1
WHERE column_name = ALL (SELECT column_name
                         FROM table2 t2);
```

```
select distinct  p.nombre, p.apellidos
from participantes p inner join jugar j on p.num_asoc=j.num_asoc
where p.num_asoc = ALL (select num_asoc
from jugadores
where num_asoc = '001');
```

nombre	apellidos
JOSE LUIS	LOPEZ VAZQUEZ



2.2
INSERT

Miguel Troyano Redondo

INSERT

¿Qué es un insert?

Este comando se utiliza para insertar datos en tablas que ya estan creadas indiferentemente de si tienen ya datos o estan vacias.

INSERT INTO

Se pueden insertar datos en una tabla especificando sus campos o sin especificar si se van a insertar en todos.

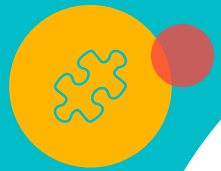
```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

```
insert into partida
values('par01','004','s001'),
      ('par02','004','s002'),
      ('par03','005','s001'),
      ('par04','005','s005'),
      ('par05','004','s003');
```

partida (3×5)

cod_partida	arbitro	cod_salas
par01	004	s001
par02	004	s002
par03	005	s001
par04	005	s005
par05	004	s003



2.3
UPDATE

Miguel Troyano Redondo

UPDATE

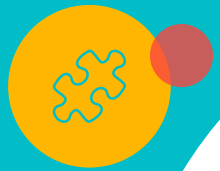
¿Qué es un update?

Este comando se utiliza para actualizar registros que ya estan creados dentro de una tabla.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
update arbitros
set sueldo = 15000;
```

```
update arbitros
set sueldo = 15000
where sueldo = 100 ;
```



2.4
DELETE

Miguel Troyano Redondo

DELETE

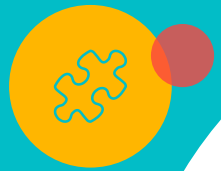
¿Qué es el delete?

Esta operación nos permite borrar valores de una tabla que cumplen una condición.

```
DELETE FROM table1
WHERE column_name  > 30;
```

```
DELETE FROM table1
WHERE column_name  = 'Texto';
```

```
DELETE FROM salas
WHERE capacidad > 30;
```



2.5
TRUNCATE

Miguel Troyano Redondo

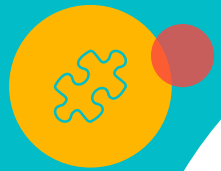
TRUNCATE

¿Qué es el truncate?

Esta operación nos permite borrar todos los valores de una tabla, pero sin borrar su estructura.

```
TRUNCATE table1;

TRUNCATE hotel ;
```



2.6
TRANSACTION

Miguel Troyano Redondo

TRANSACTION

¿Qué es una transacción?

Es un conjunto de operaciones DML que deben realizarse de una vez.

Se utiliza junto a commit o rollback para confirmar o deshacer los cambios.

TRANSACTION

Se definen de la siguiente forma:

```
start transaction;
use table_name;
...
commit;
Rollback;
```

En el siguiente ejemplo se usa la base de datos campeonato y se borra la tabla hotel. Después, se confirman los cambios.

```
start transaction;
use campeonato;
drop table hotel;
commit;
```

¡GRACIAS!

¿Alguna pregunta?

Puedes encontrarme en LinkedIN