



#>/<>

**HACK A BOSS**

**<CODE YOUR TALENT>**

# MANIPULACION DE DATOS

#>/<>

## HACK A BOSS

**<CODE YOUR TALENT>**

*"La tecnología, bien utilizada, es uno  
de los mayores catalizadores sociales  
que han existido nunca"*

**2020 EDITION**

# SQL

## Data Manipulation Language

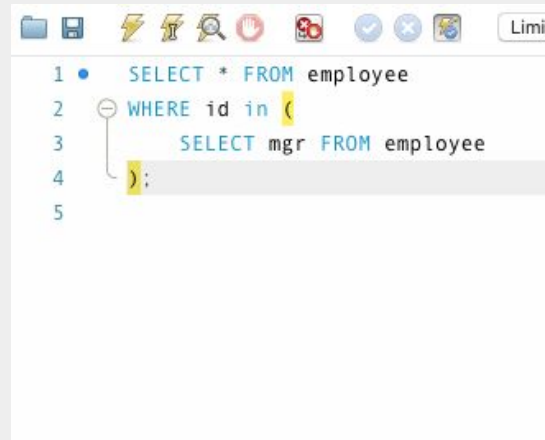
Se emplean para trabajar con los datos

- SELECT -> Consultar datos
- INSERT -> Insertar datos
- UPDATE -> Actualizar datos
- DELETE -> Eliminar datos

## Data Definition Language

Se emplean para modificar la base de datos

- CREATE -> Crear objetos
- ALTER -> Modificar objetos
- DROP -> Eliminar objetos



```
1 • SELECT * FROM employee
2 WHERE id in (
3     SELECT mgr FROM employee
4 );
5
```

#>/<>

**HACK A BOSS**

**<CODE YOUR TALENT>**

# INSERT

Nos permite añadir datos a una tabla

Para los campos en los que no queramos establecer un valor:

- Pasarle el valor NULL explícitamente
- No especificar esa columna en el INSERT

```
INSERT INTO department (id, name, location)
VALUES (50, 'MERCHANDISING', 'SAN FRANCISCO');
```

```
INSERT INTO department (id, name)
VALUES (61, NULL);
```

# INSERT

También es opcional especificar el orden de los campos

Para ello necesitamos saber en qué orden enviarlos

```
DESC dept;
```

```
INSERT INTO department  
VALUES (62, 'MERCHANDISING', 'SAN FRANCISCO');
```

# UPDATE

Nos permite modificar datos de una tabla

**Los cambios se aplican a todas las filas recuperadas.**

```
UPDATE department  
SET location = 'LOS ANGELES'  
WHERE NAME = 'SALES';
```

**Si no indicamos WHERE se modificarán todas las filas de la tabla**

```
UPDATE department  
SET name = 'ADVERTISEMENT', location = 'CALIFORNIA';
```

# UPDATE

Para modificar un único campo usaremos la PK.

```
UPDATE department  
SET name = 'ADVERTISEMENT', location = 'CALIFORNIA'  
WHERE id = 61;
```

# DELETE

Nos permite modificar datos de una tabla

**Los cambios se aplican a todas las filas recuperadas.**

```
DELETE FROM department  
WHERE NAME = 'SALES';
```

**Si no indicamos WHERE se eliminará todas las filas de la tabla**

```
DELETE FROM department
```

# DELETE

Para eliminar un único campo usaremos la PK.

```
DELETE FROM department WHERE id = 61;
```



# INTEGRIDAD REFERENCIAL

- Los valores en una fila en la tabla hija deben existir solo en una fila en la tabla referenciada.
- Una fila en la tabla hija no puede contener valores que no existen en la tabla referenciada.

	id	name	location	
►	10	ACCOUNTING	NEW YORK	
	20	RESEARCH	DALLAS	
	30	SALES	CHICAGO	
	40	OPERATIONS	BOSTON	
	NULL	NULL	NULL	

```
insert into employee (id, name, job, manager_id, hire_date, salary,  
commission, department_id)  
values (1111, 'juanito', 'SALESMAN', '7777', '2020-02-20', 5000, null, 123);
```

# INTEGRIDAD REFERENCIAL

## Acciones sobre la clave foránea

```
ALTER TABLE <nombre tabla>
  ADD [ CONSTRAINT <nombre restricción> ]
    FOREIGN KEY ( <expresión columna> [, <expresión columna>]... )
    REFERENCES <nombre tabla> [ ( <expresión columna> [, <expresión columna>]... )
]
  [ ON UPDATE <acción> ]
  [ ON DELETE <acción> ];
```

# INTEGRIDAD REFERENCIAL

## Acciones

### CASCADE

Cada vez que se eliminan o se actualizan las filas de la tabla maestra, se eliminarán o actualizarán las respectivas filas de la tabla hija.

### RESTRICT

El valor de una columna de la tabla maestra no puede ser actualizado o borrado cuando existe una fila en una tabla hija que hace referencia al valor de la columna de la tabla maestra.

Del mismo modo, una fila no se puede eliminar, siempre que haya una referencia a la misma a partir de una tabla hija.

# INTEGRIDAD REFERENCIAL

## Acciones

### NO ACTION

La principal diferencia entre el NO ACTION y RESTRICT es que NO ACTION realiza la comprobación de integridad referencial después de tratar de modificar la tabla mientras que RESTRICT hace la comprobación antes de intentar ejecutar la sentencia UPDATE o DELETE.

Ambas acciones referenciales actúan de la misma forma si la comprobación de integridad referencial falla: la sentencia UPDATE o DELETE dará lugar a un error.

### SET DEFAULT, SET NULL

El valor de la referencia afectada se cambia a NULL para SET NULL, y con el valor predeterminado especificado por SET DEFAULT .

Con SET NULL, cada vez que se elimina o actualiza el registro en la tabla padre, establece a NULL las columnas de la FK en la tabla hija, para ello las columnas de la tabla hija NO han de haber sido definidas como NOT NULL.

# PROPIEDADES DE BBDD

## ACID

Atomicity, Consistency, Isolation and Durability

Atomicidad, Consistencia, Aislamiento y Durabilidad

- **(A) Atomicidad:** Si cuando una operación consiste en una serie de pasos, de los que o bien se ejecutan todos o ninguno, es decir, las transacciones son completas.
- **(C) Consistencia:** (*Integridad*). Es la propiedad que asegura que sólo se empiece aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de *Integridad* de la base de datos.

# PROPIEDADES DE BBDD

## ACID

Atomicity, Consistency, Isolation and Durability

Atomicidad, Consistencia, Aislamiento y Durabilidad

- (I) **Aislamiento:** una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- (D) **Durabilidad:** (*Persistencia*). Esta propiedad asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan.

# TRANSACCIONES

- Una transacción es una unidad lógica que consta de 1 o más instrucciones SQL.
- Son atómicas.
- Se puede aceptar la transacción (COMMIT) o volver atrás (ROLLBACK)
- Nos garantiza que una serie de instrucciones se ejecuten con éxito o ninguna se ejecute, volviendo al estado inicial antes de ejecutar la transacción.

# TRANSACCIONES

## Transferencia bancaria

- Retirar fondos de la cuenta de origen
- Añadir fondos a la cuenta de destino
- Registrar la transferencia, con su origen, destino, cantidad y fecha
- Aceptar la transacción



# TRANSACCIONES

## Transferencia bancaria

- Retirar fondos de la cuenta de origen
- Error: no hay fondos suficientes
- Volver atrás

# TRANSACCIONES

## Transferencia bancaria

- Retirar fondos de la cuenta de origen
- Añadir fondos a la cuenta de destino
- Registrar la transferencia, con su origen, destino, cantidad y fecha
- Error (hay un campo mal, etc)
- Volver atrás

# TRANSACCIONES

START TRANSACTION

```
transaction_characteristic [, transaction_characteristic] ...]
```

BEGIN [WORK]

COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]

ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]

SET autocommit = {0 | 1}



#>/<>

**HACK A BOSS**

**<CODE YOUR TALENT>**

# #<THANX!>

#>/<>

## HACK A BOSS

### <CODE YOUR TALENT>

+34 919 04 23 63

[www.hackaboss.com](http://www.hackaboss.com)

Av.Linares Rivas 50-51, 15005, A Coruña

**2020 EDITION**