

# Base de Datos

## Sesión 7: SQL DML - Subconsultas

2022-2

## Semana 7 SQL DML



Profesor del curso:  
César Aguilera  
Luis Ríos



Elaborado por:  
César Aguilera  
Luis Ríos



Revisado por:  
César Aguilera  
Rony Cueva  
Luis Ríos

## Saberes previos

- SQL DML, lenguaje de manipulación de datos

# Agenda

- SQL DML: Lenguaje para manipulación de datos
  - Indexes
  - Vistas
  - Sequence
- SQL DQL: Lenguaje para consulta de datos
  - Subconsultas
- SQL DCL: Lenguaje para control de datos
  - Role
  - User
  - Grant

## CREATE INDEX

Un índice es una estructura que sirve para agilizar las consultas, guardando el orden de la(s) columna(s) dentro de una tabla máximo puede ser 16. Una física y el resto lógicas.

Recomendaciones:

- Crear índices luego de carga de datos.
- Elegir columnas más convenientes :
  - Usadas en joins.
  - Con valores relativamente únicos
  - Con diversidad de valores
  - Si no se hacen consultas con valores nulos
  - Columnas de tipo LONG y LONG RAW no pueden indexarse
- Elegir orden conveniente en caso de índice para varias columnas

## CREATE INDEX

Para el ejemplo anterior generemos un índice sobre la tabla Persona:

```
create index In_puesto on Persona(Puesto desc)
```

Ahora definamos los siguientes índices:

```
create index In_apellido on Persona(Apellido)
```

```
create index In_departamento on Persona(IDDEP)
```

Los mismo estaran diseñados de la siguiente forma:

# INDICES

Índice por apellido

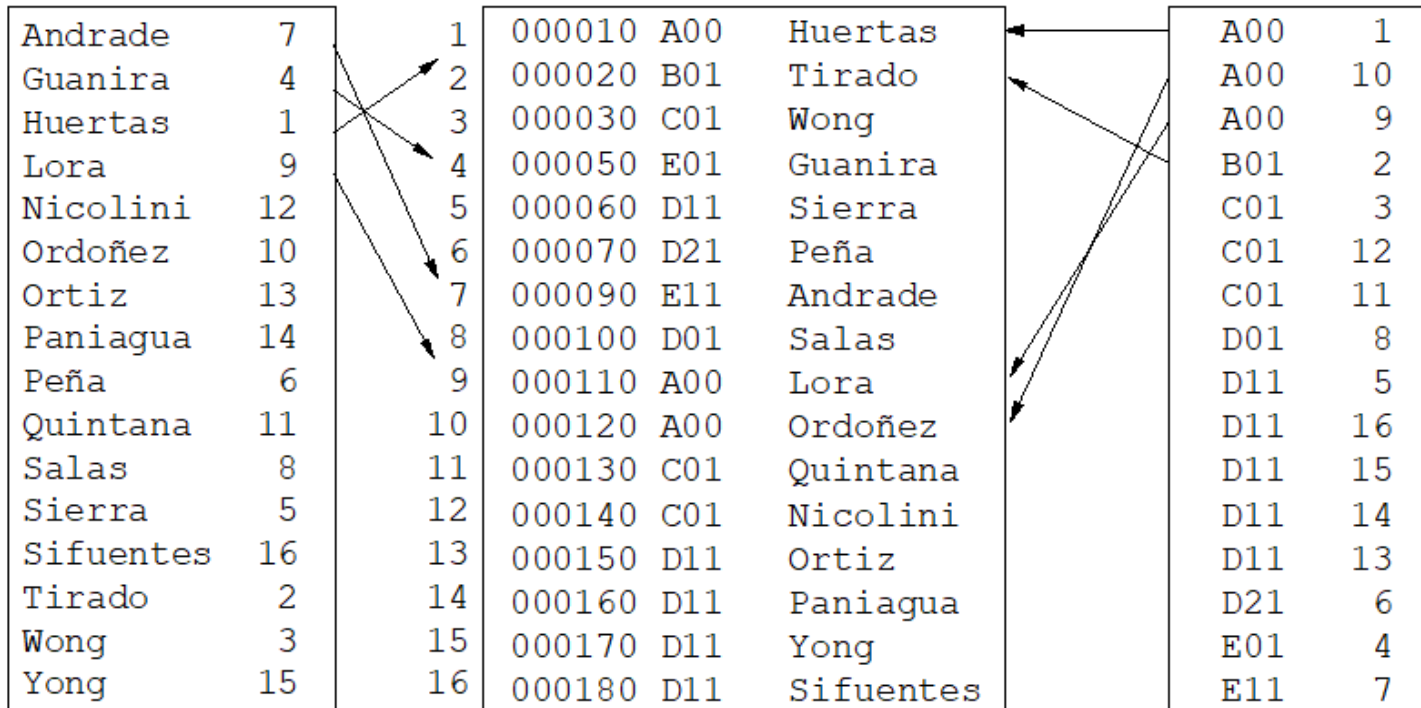
APELLIDO	RID
Andrade	7
Guanira	4
Huertas	1
Lora	9
Nicolini	12
Ordoñez	10
Ortiz	13
Paniagua	14
Peña	6
Quintana	11
Salas	8
Sierra	5
Sifuentes	16
Tirado	2
Wong	3
Yong	15

Tabla persona

IDPER	IDDEP	APELLIDO
000010	A00	Huertas
000020	B01	Tirado
000030	C01	Wong
000050	E01	Guanira
000060	D11	Sierra
000070	D21	Peña
000090	E11	Andrade
000100	D01	Salas
000110	A00	Lora
000120	A00	Ordoñez
000130	C01	Quintana
000140	C01	Nicolini
000150	D11	Ortiz
000160	D11	Paniagua
000170	D11	Yong
000180	D11	Sifuentes

Índice por departamento

IDDEP	RID
A00	1
A00	10
A00	9
B01	2
C01	3
C01	12
C01	11
D01	8
D11	5
D11	16
D11	15
D11	14
D11	13
D21	6
E01	4
E11	7



# CREATE INDEX

CREATE INDEX índice esquema tabla esquema (columna ASC DESC)

ON esquema



# Definición de vistas

## CREATE VIEW

Define una vista (tabla lógica) en base a una o más tablas o vistas ya existentes. No contiene datos.

Las vistas son usadas para :

- Nivel adicional de seguridad
- Ocultar complejidad
- Mostrar desde otro punto de vista la misma información.

## Definición de vistas

```
CREATE VIEW view-Name  
    [ ( Simple-column-Name [, Simple-column-Name] * ) ]  
AS Query [ ORDER BY clause ]
```

### Ejemplos:

```
CREATE VIEW V1 (COL_SUM, COL_DIFF)  
    AS SELECT COMM + BONUS, COMM - BONUS  
    FROM EMPLOYEE;
```

```
CREATE VIEW PROJ_COMBO  
    (PROJNO, PRENDATE, PRSTAFF, MAJPROJ)  
    AS SELECT PROJNO, PRENDATE, PRSTAFF, MAJPROJ  
    FROM PROJECT  
    UNION ALL  
    SELECT PROJNO, EMSTDATE, EMPTIME, EMPNO  
    FROM EMP_ACT  
    WHERE EMPNO IS NOT NULL;
```

## Secuencias (Sequence)

Una secuencia (*sequence*) se emplea para generar valores enteros secuenciales únicos y asignárselos a campos numéricos; se utilizan generalmente para las claves primarias de las tablas garantizando que sus valores no se repitan.

```
CREATE SEQUENCE NOMBRESECUENCIA  
  start with VALORENTERO  
  increment by VALORENTERO  
  maxvalue VALORENTERO  
  minvalue VALORENTERO  
  cycle | nocycle;
```

- **"start with"** indica el valor desde el cual comenzará la generación de números secuenciales. Si no se especifica, se inicia con el valor que indique "minvalue".

## Secuencias (Sequence)

- **"increment by"** especifica el incremento, es decir, la diferencia entre los números de la secuencia; debe ser un valor numérico entero positivo o negativo diferente de 0. Si no se indica, por defecto es 1.
- **"maxvalue"** define el valor máximo para la secuencia. Si se omite, por defecto es 99999999999999999999999999999999.
- **"minvalue"** establece el valor mínimo de la secuencia. Si se omite será 1.
- **"cycle"** indica que, cuando la secuencia llegue a máximo valor (valor de "maxvalue") se reinicie, comenzando con el mínimo valor ("minvalue") nuevamente, es decir, la secuencia vuelve a utilizar los números. Si se omite, por defecto la secuencia se crea "nocycle".

## Secuencias (Sequence)

En este caso se crea una secuencia llamada ***sec\_codigoclientes***.

```
create sequence sec_codigoclientes  
start with 1  
increment by 1  
maxvalue 99999  
minvalue 1;
```

La secuencia comienza en 1, se incrementa en 1, sus valores estarán entre 1 y 99999, por defecto, será no cycle.

Se utilizarán generalmente para una tabla específica, por lo tanto, es conveniente darle un nombre que referencie a la misma. Las secuencias son independientes de las tablas.

## Secuencias (Sequence)

Para recuperar los valores de una secuencia empleamos las pseudocolumnas "*currval*" y "*nextval*".

- Primero, debe inicializar la secuencia con "*nextval*". La primera vez que se referencia "*nextval*" retorna el valor de inicio de la secuencia; las siguientes veces, incrementa la secuencia y nos retorna el nuevo valor:

NOMBRESECUENCIA.nextval;

- Para recuperar el valor actual de una secuencia usamos:

NOMBRESECUENCIA.currval;

Los valores retornados por "*currval*" y "*nextval*" pueden usarse en sentencias "INSERT" y "UPDATE".

## Secuencias (Sequence)

Inicializamos la secuencia

```
SELECT sec_codigoclientes.nextval FROM dual;
```

La primera vez que se reference la secuencia debe emplearse "nextval" para inicializarla.

Ingresamos un registro en "clientes", almacenando en el campo "codigo" el valor actual de la secuencia:

```
INSERT INTO clientes VALUES  
(sec_codigoclientes.currval, 'Kinos S.A', 'Av. Universitaria 1802');
```

Luego ingresamos otro registro en "clientes":

```
INSERT INTO clientes VALUES  
(sec_codigoclientes.nextval, 'ABC S.A', 'Jr. Morro de Arica 257');
```

# Anidamiento (Subconsultas)

Una subconsulta es un comando SELECT colocado “dentro de” otro comando SQL (llamado “comando padre”), para que este último utilice las filas resultado de la selección.

Una subconsulta es evaluado sólo una vez para el comando padre.

Puede ser:

- Subconsulta escalar
- Subconsulta correlacionada



# Anidamiento (Subconsultas)

```
SELECT select_list  
FROM table  
WHERE expr operator (SELECT select_list  
FROM table);
```

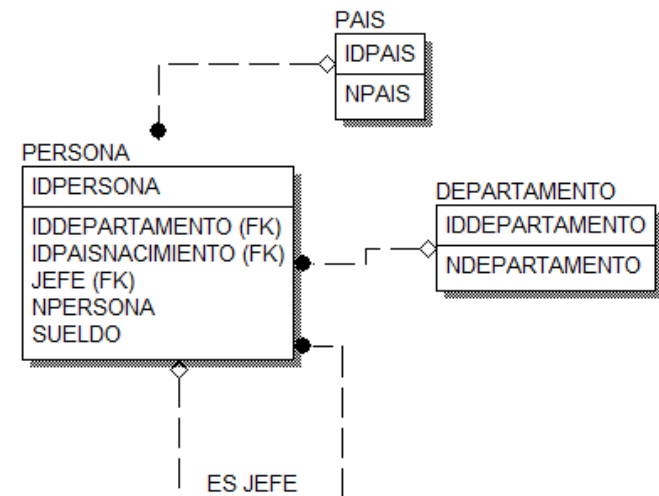
- La subconsulta (consulta interna) se ejecuta una vez antes de la consulta principal
- El resultado de la subconsulta es usado por la consulta principal

11. Obtener los empleados cuyos sueldos sean mayores al promedio o al de su jefe

```

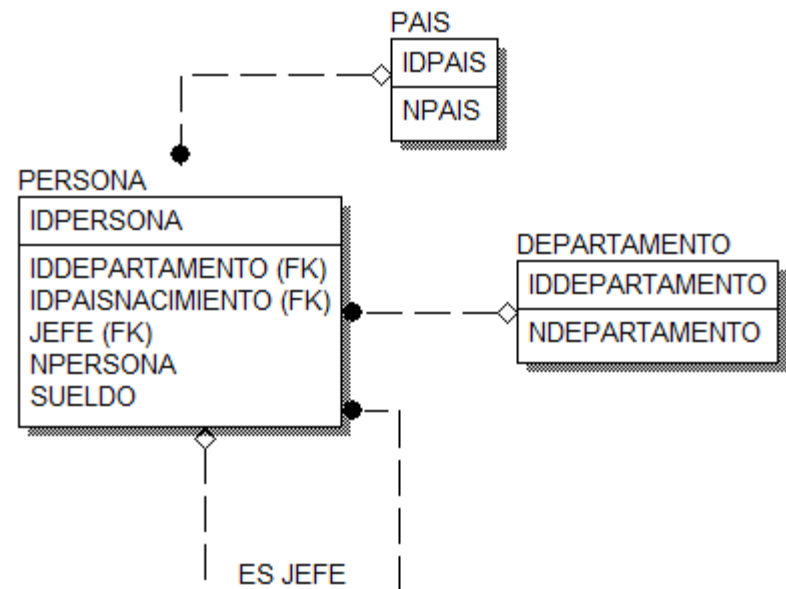
SELECT      P.NPersona, P.Sueldo, J.NPersona "Jefe",
            J.Sueldo "Sueldo del Jefe"
FROM        Persona P, Persona J
WHERE       P.Jefe = J.IdPersona
AND
(P.Sueldo > ( SELECT  AVG(Sueldo) FROM Persona)
OR
P.Sueldo > J.Sueldo )
  
```

Subconsulta.  
Devuelve un  
valor



12. Obtener los departamentos que tenga un salario promedio menor que el salario promedio de la Cía.

```
SELECT IdDepartamento, AVG(Sueldo)
FROM Persona
GROUP BY IdDepartamento
HAVING AVG(Sueldo) < (SELECT AVG(Sueldo) FROM Persona)
```



# SELECT

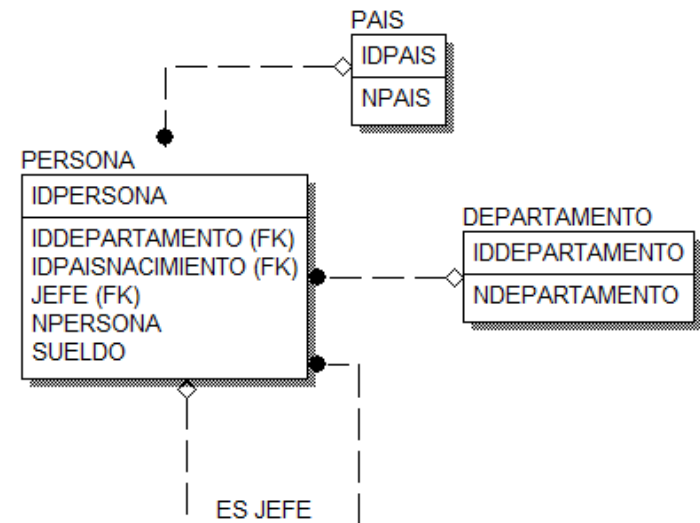
Pertenencia ( $\epsilon$ )

```
SELECT a1
FROM R1
WHERE a1 IN (SELECT a2
               FROM R2 ... )
```

13. Obtener los compañeros de "FIESTAS" (del mismo departamento que éste)

```
SELECT IdDepartamento, Npersona  
FROM Persona  
WHERE IdDepartamento IN (SELECT IdDepartamento  
FROM Persona  
WHERE NPersona LIKE "FIESTAS%")
```

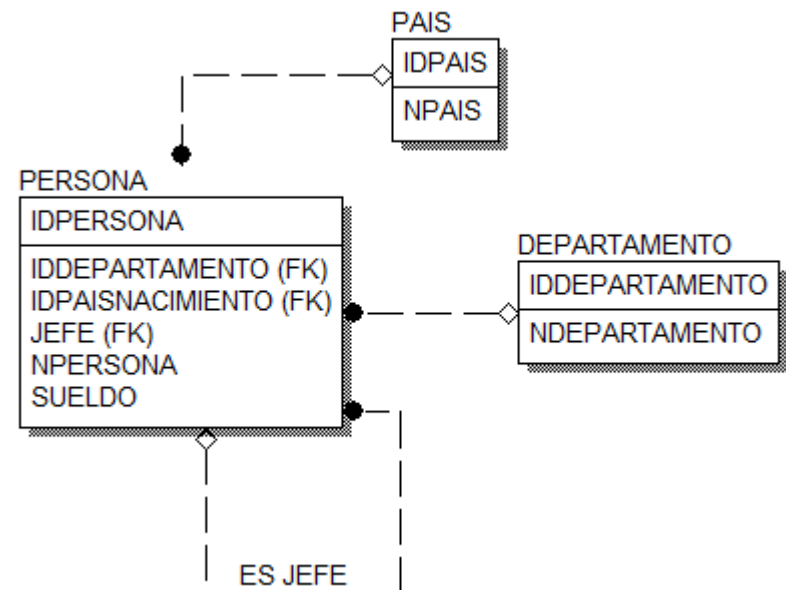
Devuelve más  
de un valor



14. Obtener los nombres de los trabajadores que no son Gerentes ni Subgerentes

```
SELECT NPersona  
FROM   Persona  
WHERE  Puesto NOT IN ("GERENTE","SUBGERENTE")
```

Es una lista  
de valores



## Usando una subconsulta en la cláusula FROM


Una subconsulta en la cláusula FROM de una sentencia SELECT es también llamada una vista en línea. Una subconsulta en una cláusula FROM de una sentencia SELECT define un origen de datos para esa sentencia SELECT en particular, y solo esa sentencia SELECT.

```
SELECT  a.last_name, a.salary,  
        a.department_id, b.salavg  
FROM    employees a, (SELECT  department_id,  
                        AVG(salary) salavg  
                        FROM    employees  
                        GROUP BY department_id) b  
WHERE   a.department_id = b.department_id  
AND     a.salary > b.salavg;
```

Es como  
una tabla  
“virtual”

## Subconsulta Escalar

Una subconsulta que obtiene exactamente un valor de una columna de una fila es también llamada subconsulta escalar.

```
SELECT employee_id, last_name,  
       (CASE  
         WHEN department_id =  20  
         (SELECT department_id FROM departments  
          WHERE location_id = 1800)  
         THEN 'Canada' ELSE 'USA' END) location  
FROM   employees;
```

Subconsultas de múltiples columnas escritas para comparar dos o más columnas, usando una cláusula WHERE compuesta y operadores lógicos, no pueden ser calificadas como subconsultas escalares.



# Anidamiento Correlacionado

*(Correlated subqueries)*

Un subconsulta correlacionada es un comando SELECT que es evaluado para cada fila procesada por el “comando padre”, el que puede ser un comando SELECT, UPDATE o DELETE.

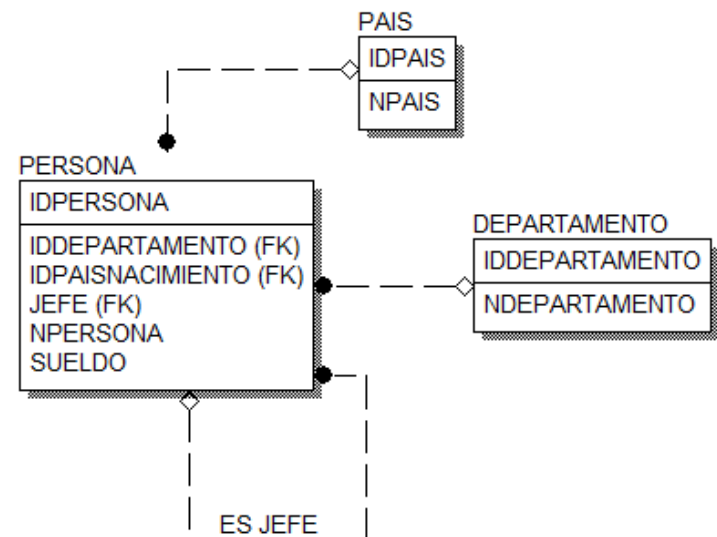
Se ejecuta cada vez que la subconsulta hace referencia a una columna de una tabla del comando padre.

Si se tiene a la misma tabla en la subconsulta y en comando padre, es necesario el alias. En otros casos siempre es recomendable el uso de alias para calificar a las columnas.

15. Obtener las personas cuyos salarios excedan al promedio de su respectivo departamento.

```
SELECT IdDepartamento, NPersona, Sueldo  
FROM Persona P  
WHERE Sueldo > (SELECT AVG (Sueldo)  
                FROM Persona R  
                WHERE P.IdDepartamento = R.IdDepartamento)
```

Se relacionan en la condición



# EXISTS

Existencia ( $\exists$ )

```
SELECT  a1, ... an
FROM    R1, ... Rn
WHERE EXISTS (SELECT a1 ...
                FROM  Rm ... )
```

```
SELECT  a1, ... an
FROM    R1, ... Rn
WHERE NOT EXISTS (SELECT a2
                    FROM  R2 ... )
```

## EXISTS

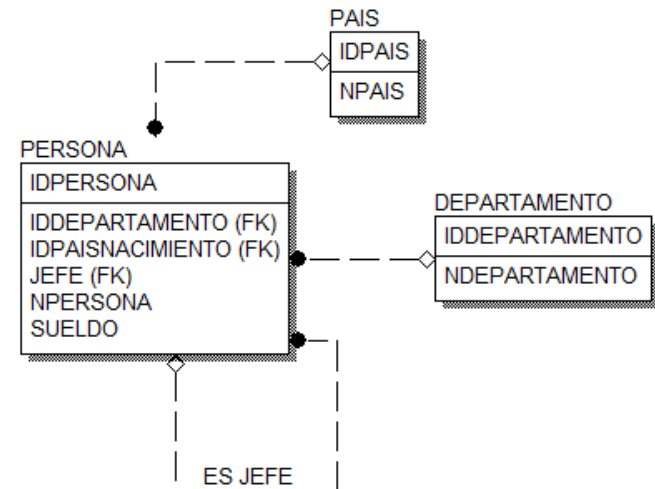
Este operador es frecuentemente usado en subconsultas correlacionadas para verificar cuando un valor recuperado por la consulta externa existe en el conjunto de resultados obtenidos por la consulta interna. Si la subconsulta obtiene al menos una fila, el operador obtiene el valor TRUE. Si el valor no existe, se obtiene el valor FALSE.

Consecuentemente, NOT EXISTS verifica cuando un valor recuperado por la consulta externa no es parte del conjunto de resultados obtenidos por la consulta interna.

16. Obtener los nombres de los departamentos que cuenten con algún empleado

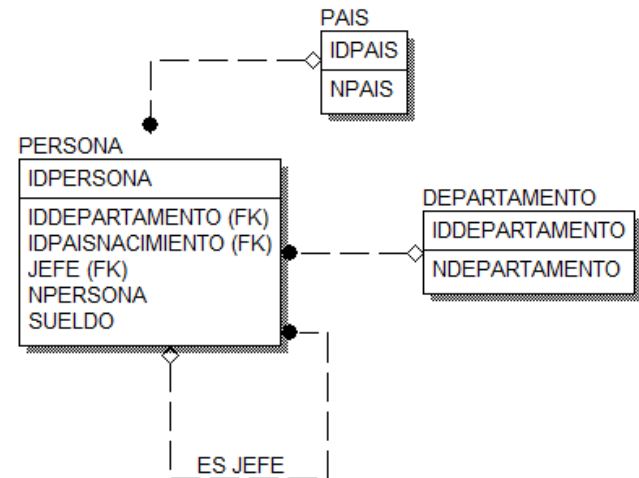
```
SELECT nDepartamento
FROM   Departamento
WHERE  EXISTS (SELECT *
                FROM   Persona
                WHERE  Departamento.IdDepartamento =
                       Persona.IdDepartamento)
```

No necesita especificar columnas



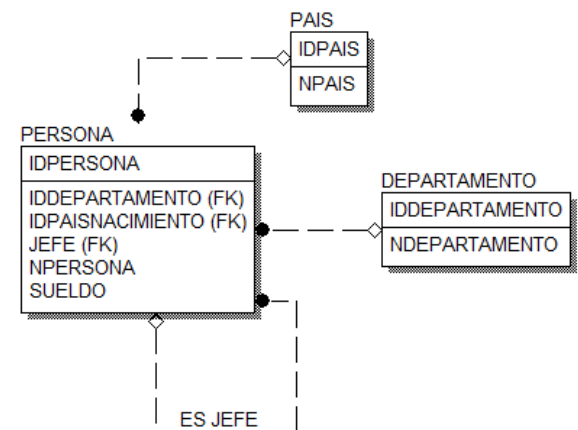
17. Obtener los nombres de los departamentos que no tengan ningún empleado

```
SELECT nDepartamento
FROM   Departamento
WHERE  NOT EXISTS (SELECT *
                   FROM Persona
                   WHERE Departamento.IdDepartamento =
                   Persona.IdDepartamento)
```



18. Obtener los nombres de los departamentos que no tengan ningún empleado  
(Usando COUNT(\*))

```
SELECT nDepartamento
FROM  Departamento
WHERE
(SELECT COUNT(*)
FROM  Persona
WHERE Departamento.IdDepartamento = Persona.IdDepartamento) = 0
```



# Seguridad: Data Control Language

## CREATE USER

Crea un usuario de base de datos o una “cuenta” a través de la que podrá darse “log in” a la base de datos. Mediante el mismo comando pueden asignársele recursos.

## CREATE ROLE

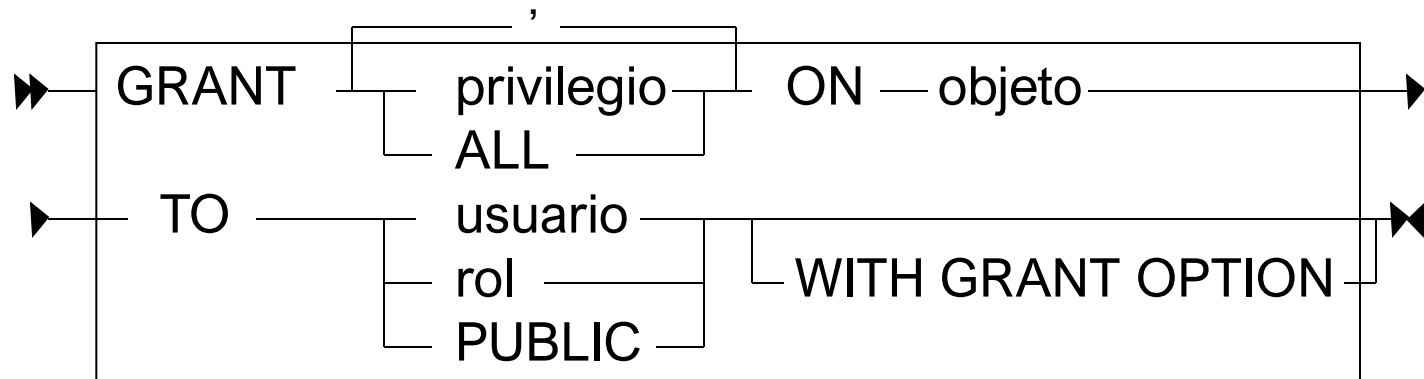
Define un rol, que es un conjunto de privilegios a otorgarse a usuarios u otros roles.



## Seguridad: Data Control Language

- GRANT (1)** Otorga privilegios de sistema a usuarios y roles.
- GRANT (2)** Otorga privilegios sobre un determinado objeto (tabla, vista, sinónimo, paquete, procedimiento, etc.) a usuarios o roles.
- REVOKE** Revoca privilegios otorgados a usuarios o roles. (revierte el resultado del comando GRANT)

# GRANT



Privilegio	Tablas	Vistas	Secuencias	Proced. Funcs.	Snapshots
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	
INDEX	X				
INSERT	X	X			
REFERENCES	X				
SELECT	X	X	X		X
UPDATE	X	X			

## Ejemplos

```
CREATE ROLE rl_prueba
```

```
GRANT CREATE session to rl_prueba
```

```
GRANT ALL on Persona to rl_prueba
```

```
CREATE USER prueba identified by prueba
```

```
GRANT rl_prueba to prueba
```