PUCP

1INF33 Base de Datos

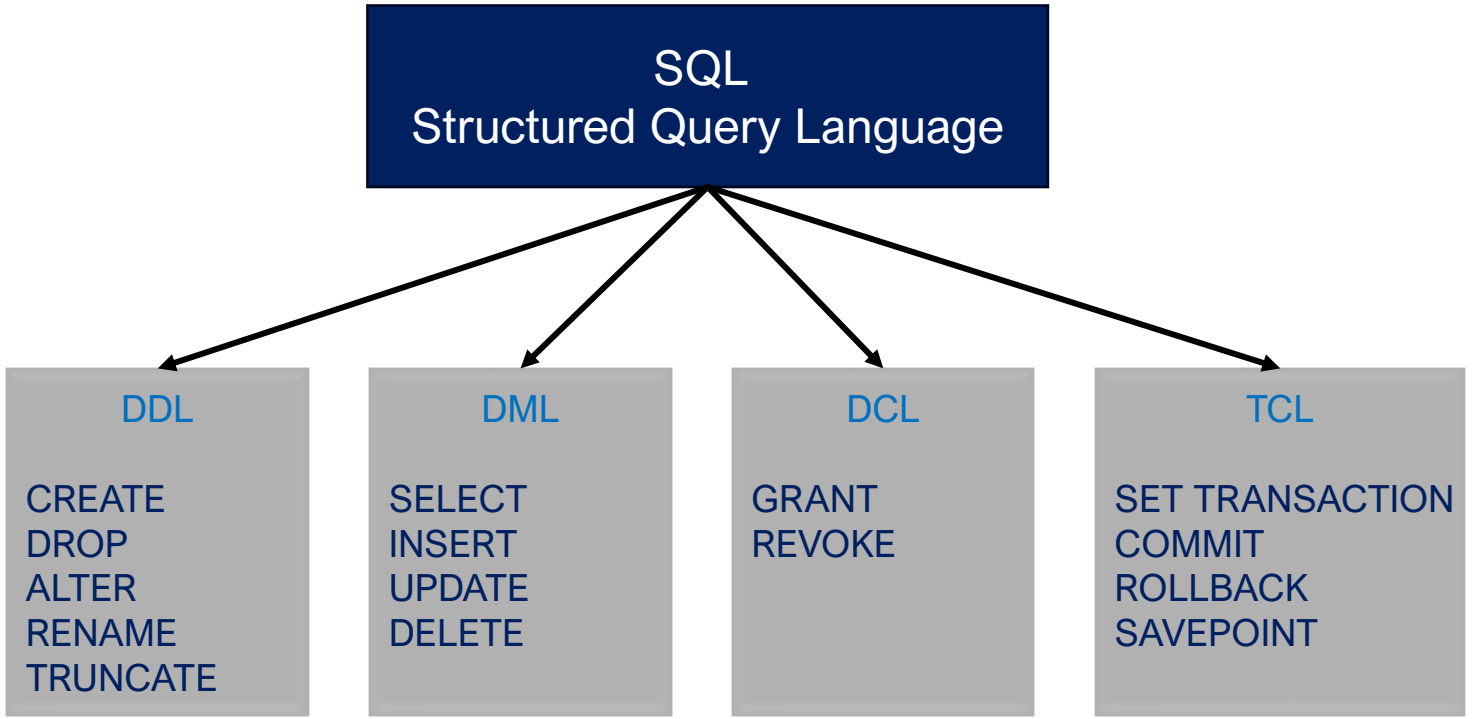
DML

2024

Profesores del curso

ÍNDICE

Antecedentes



Saberes previos

- DML, Lenguaje de manipulación de datos
- Uso del INNER JOIN y OUTER JOIN
- Agrupaciones

Índices

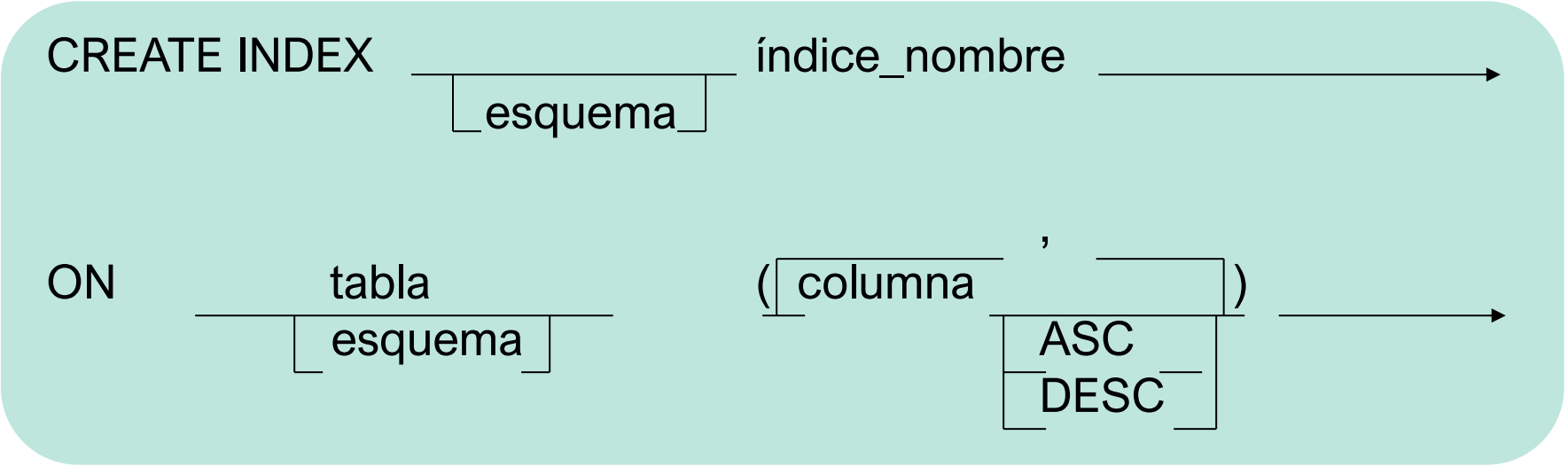
Definición

Un índice es un objeto de esquema que contiene una entrada para cada valor que aparece en las columnas indexadas de la tabla [o el clúster] y proporciona acceso directo y rápido a las filas.

Recomendaciones:

- Crear índices luego de carga de datos
- Elegir columnas más convenientes como:
  - Usadas en JOINS
  - Con valores relativamente únicos
  - Con diversidad de valores
  - Que no contengan valores nulos
  - Columnas de tipo LONG y LONG RAW no pueden indexarse
- Elegir orden conveniente en caso de índice para varias columnas

Cláusula INDEX



Para eliminar un índice

DROP INDEX idx_name;

Tipos

[Oracle] Admite varios tipos de índices:

- *Normal index**, de manera predeterminada crea índice B-tree (árbol balanceado)
- *Bitmap index*, es adecuado para columnas con un conjunto limitado de valores distintos
- *Partitioned index*, divide el índice en particiones según criterio (rango, lista o hash)
- *Function_based index*, permite crear un índice en el resultado de una función o expresión aplicada a una o más columnas
- *Domain index*, es un índice definido por el usuario para datos complejos que no son fácilmente indexables mediante los índices estándares

*Haremos énfasis en este tipo de índice

Cláusula INDEX

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

```
CREATE INDEX idx_puesto ON PERSONA(puesto DESC)
```

Ahora definamos los siguientes índices:

```
CREATE INDEX idx_apellido ON PERSONA(apellido)
CREATE INDEX idx_departamento ON PERSONA(iddep)
```

Los mismos estarán diseñados de la siguiente manera (ver Figura. Estructura INDEX tabla PERSONA)

Estructura INDEX

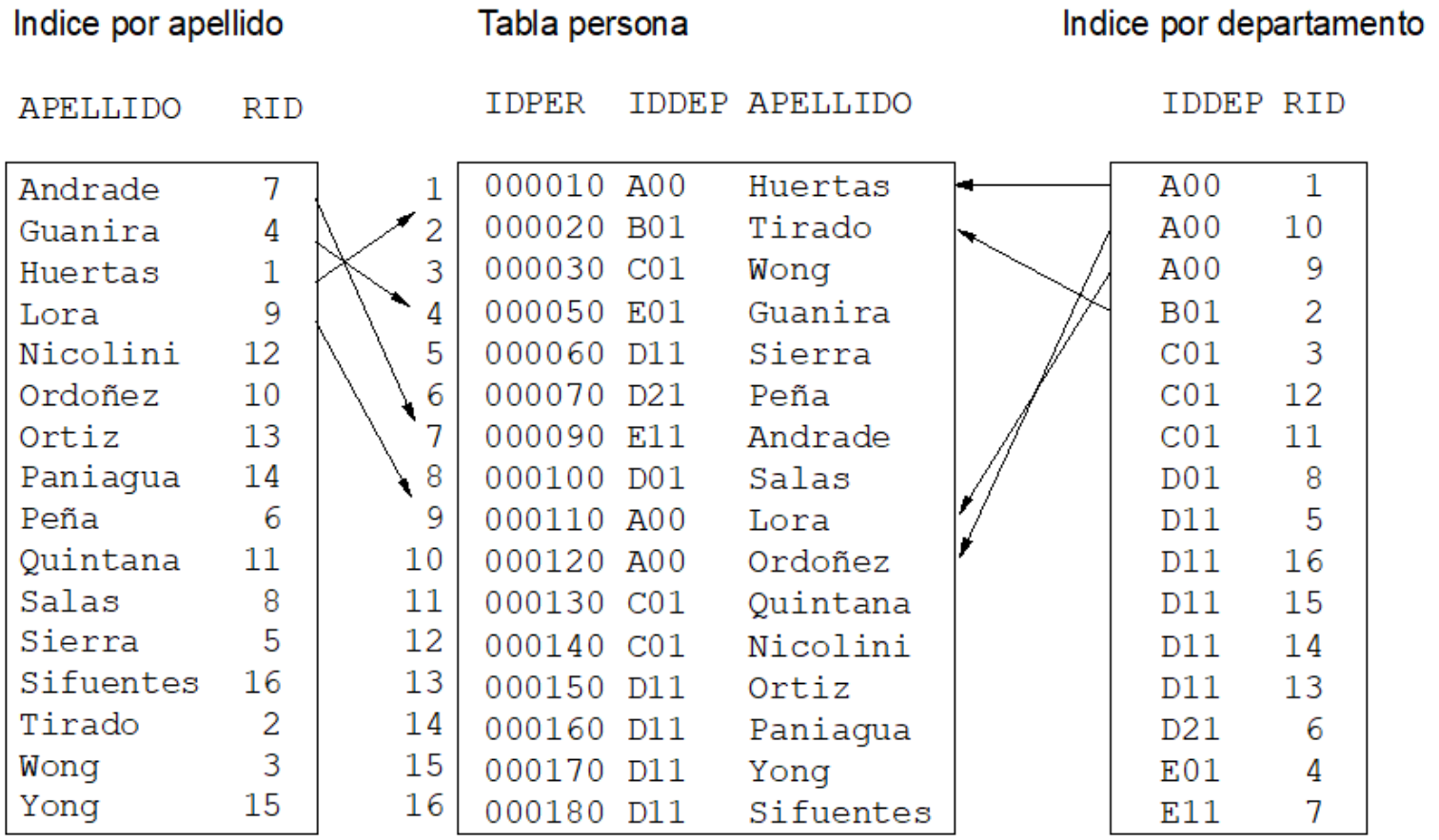


Figura. Estructura INDEX tabla PERSONA

Sintaxis

Para el ejemplo:

CREATE INDEX idx_empleado_apellido ON EMPLOYEES (LAST_NAME);

ÍNDICE NORMAL de una columna B-tree

CREATE INDEX idx_departamento_salario
ON EMPLOYEES (department_id, salary);

ÍNDICE NORMAL de dos columnas B-tree

*CREATE BITMAP INDEX idx_empleado_genero ON EMPLOYEES (gender);

ÍNDICE BIPMAP

CREATE INDEX idx_empleado_apellido_upper
ON EMPLOYEES (**UPPER**(last_name));

ÍNDICE DE FUNCIÓN

* No habilitado en Oracle XE

Sintaxis

ÍNDICE PARTICIONADO

Para el ejemplo:

```
* CREATE INDEX idx_empleado_particionado ON EMPLOYEES (hire_date)
GLOBAL PARTITION BY RANGE (hire_date) (
PARTITION  p1 VALUES LESS THAN(TO_DATE('01-01-1999','DD-MM-YYYY')),
PARTITION p2 VALUES LESS THAN(TO_DATE('01-01-2000','DD-MM-YYYY')),
PARTITION p3 VALUES LESS THAN(MAXVALUE)
);
```

* No habilitado en Oracle XE

Vistas

Definición

Se define una vista (tabla lógica) en base a una o más tablas o vistas ya existentes. No contiene datos. Las tablas en las que se basa una vista se denominan tablas base.

Las vistas son usadas para :

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

Tipos

Existen diferentes tipos de vistas [en Oracle] y cada una tiene sus características:

- Vistas simples
  - basada en una sola tabla, generalmente modificable
- Vistas complejas
  - Involucran múltiples tablas o agregaciones, generalmente no modificable
- Vistas complejas con control de actualización
  - Permiten modificaciones usando *triggers* INSTEAD OF
- Vistas materializadas
  - Almacena datos físicamente para mejorar el rendimiento

Sintaxis

```
CREATE [OR REPLACE] VIEW view_name [ ( column_name [, column_name] * ) ]
AS subquery [ subquery_restriction_clause ]
[WITH READ ONLY]
[WITH CHECK OPTION]
```

Para eliminar una vista

```
DROP VIEW view_name
[CASCADE CONSTRAINT];
```

Característica

Además, una vista [en Oracle]:

- Con la cláusula WITH READ ONLY no permite ninguna operación de modificación de datos como INSERT, UPDATE o DELETE. Significa que los usuarios solo pueden consultar los datos a través de la vista, sin riesgo de modificar el contenido de las tablas subyacentes.

- Con la cláusula WITH CHECK OPTION se asegura de que cualquier dato insertado o actualizado a través de la vista cumpla con la condición definida en la consulta definitoria.

Sintaxis

Ejemplos:

```
CREATE VIEW VCOMISION (COD_EMPLEADO, COMISION, BONO) AS
 SELECT EMPLOYEE_ID, COMM, BONUS
 FROM EMPLOYEES;
```

VISTA SIMPLE

```
CREATE VIEW VLOCALIDADES AS
SELECT LOCATION_ID, CITY, L.COUNTRY_ID, C.COUNTRY_NAME, R.REGION_NAME
FROM LOCATIONS L, COUNTRIES C, REGIONS R
WHERE
L.COUNTRY_ID = C.COUNTRY_ID AND C.REGION_ID = R.REGION_ID;
```

VISTA COMPLEJA
Modificable*

Sintaxis

Ejemplos:

```
CREATE VIEW VHONORARIOS AS
SELECT D.DEPARTMENT_NAME , SUM(E.SALARY) AS TOTAL_SALARIO
FROM DEPARTMENTS D, EMPLOYEES E
WHERE
D.DEPARTMENT_ID = E.DEPARTMENT_ID
GROUP BY D.DEPARTMENT_NAME;
```

VISTA COMPLEJA
No modificable*

```
CREATE MATERIALIZED VIEW VPLANILLA AS
SELECT JOB_ID, LAST_NAME , FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE
SALARY >= 10000;
```

VISTA MATERIALIZADA

Secuencias

Definición

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 NOMBRE_SEQUENCE
  start with VALORENTERO
  increment by VALORENTERO
  maxvalue VALORENTERO
  minvalue VALORENTERO
  cycle  | nocycle;
```

Para eliminar una secuencia

```
DROP SEQUENCE nombre_sequence;
```

Definición

- **START WITH** indica el valor inicial de los números secuenciales. Si no se especifica, se inicia con el valor que indique "minvalue".
- **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 999999999999999999999999.
- **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.

Sintaxis

En este caso se crea una secuencia llamada seq_codigo_cliente.

```
CREATE SEQUENCE seq_codigo_cliente
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.

Sintaxis

Para recuperar el valor de la 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 la secuencia usamos:

  NOMBRESECUENCIA.CURRVAL;

Los valores retornados por "CURRVAL" y "NEXTVAL" pueden usarse en sentencias INSERT y UPDATE.

Consultas

Inicializamos la secuencia

```
SELECT  seq_codigo_cliente.NEXTVAL FROM DUAL;
```

La primera vez que se referencie la secuencia debe emplearse "NEXTVAL" para inicializarla.

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

```
INSERT INTO CLIENTE VALUES
 (seq_codigo_cliente.currval, ‘Tienda Azul S.A', ‘Av. La Floresta 1802');
```

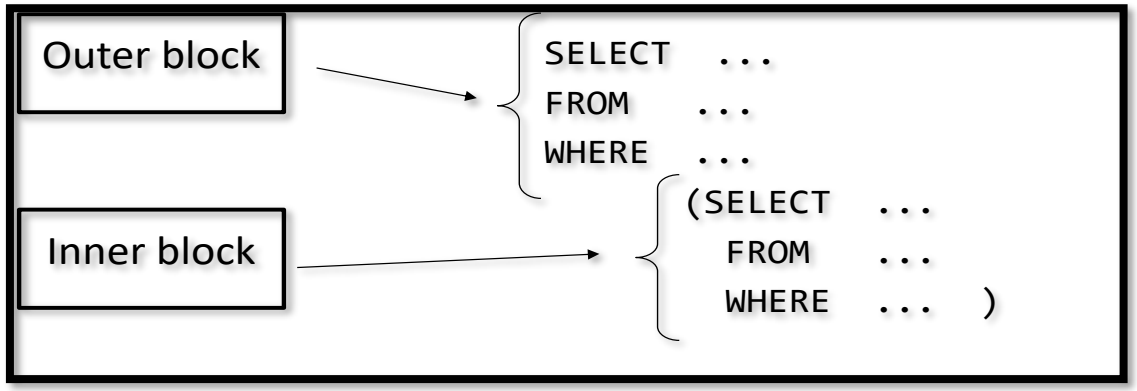
Luego ingresamos otro registro en “cliente":

```
INSERT INTO CLIENTE VALUES
  (seq_codigo_cliente.NEXTVAL, ‘Tienda Blanca S.A', ‘Jr. Morro Solar 257');
```

Subconsultas

Anidamiento

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 evaluada sólo una vez para el comando padre.

Puede ser:

- Subconsulta escalar
- Subconsulta correlacionada

Anidamiento

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

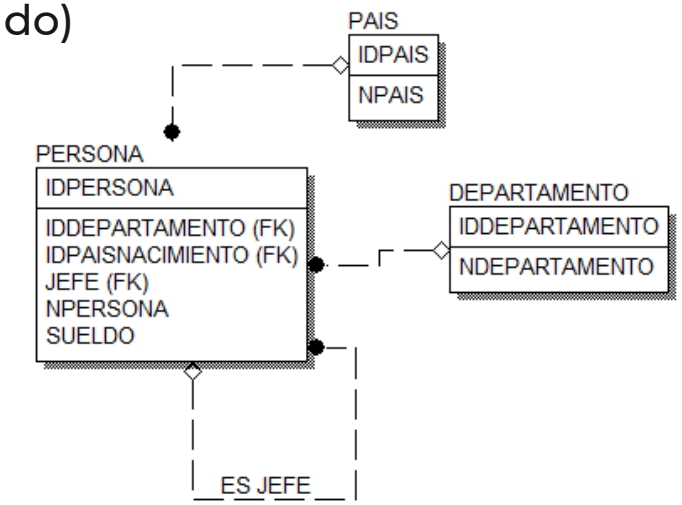
- La subconsulta (consulta interna) se ejecuta una vez antes de la consulta principal
- El resultado de la subconsulta es usado por la consulta principal

Consulta

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



Consulta

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)
```

Pertenencia

Cláusula IN

Sintaxis

Pertenencia (ε)

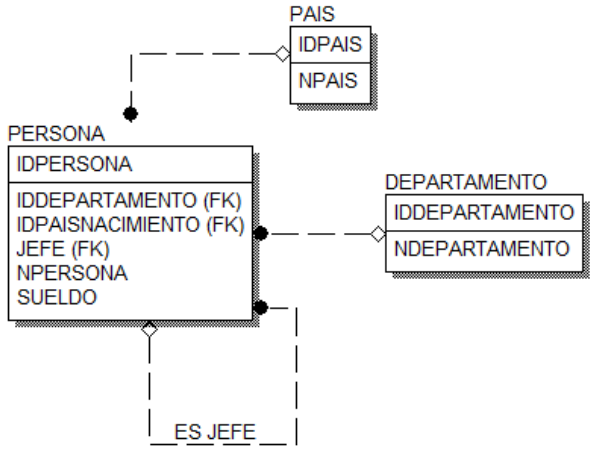
SELECT a_1
FROM R_x
WHERE a_1 **IN** (SELECT a_2
FROM R_y ...)

Consulta

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

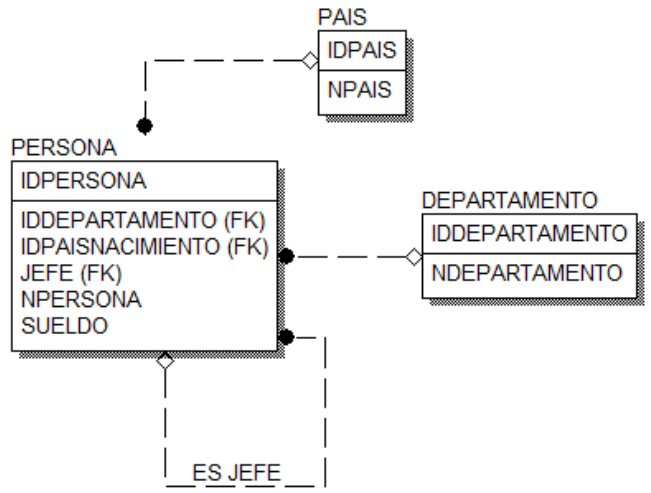


Consulta

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



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 =
             (SELECT department_id FROM departments
              WHERE location_id = 1800)
        THEN 'Canada' ELSE 'USA' END) location
FROM   employees;
```

20

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

Subconsulta Correlacionada

Una subconsulta correlacionada (*correlated subqueries*) 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 el comando padre, es necesario el alias. En otros casos siempre es recomendable el uso de alias para calificar a las columnas.

Consulta

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

Existencia

Cláusula EXISTS

Definición

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.

Sintaxis

Existencia (∃)

SELECT a_1, ... a_n
FROM R_1 , ... R_n
WHERE **EXISTS** (SELECT a_1 ...
FROM R_x ...)

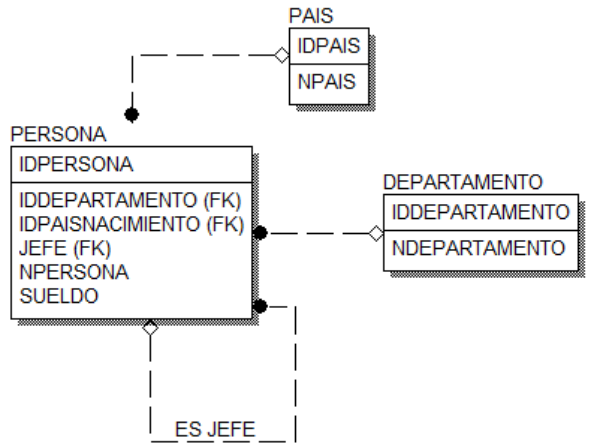
SELECT a1, ... an
FROM R_1 , ... R_n
WHERE **NOT EXISTS** (SELECT a_2
FROM R_y ...)

Consulta

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

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

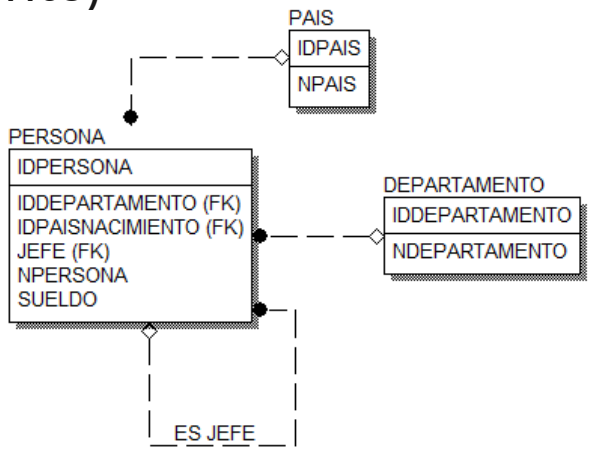
No necesita especificar columnas



Consulta

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

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



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

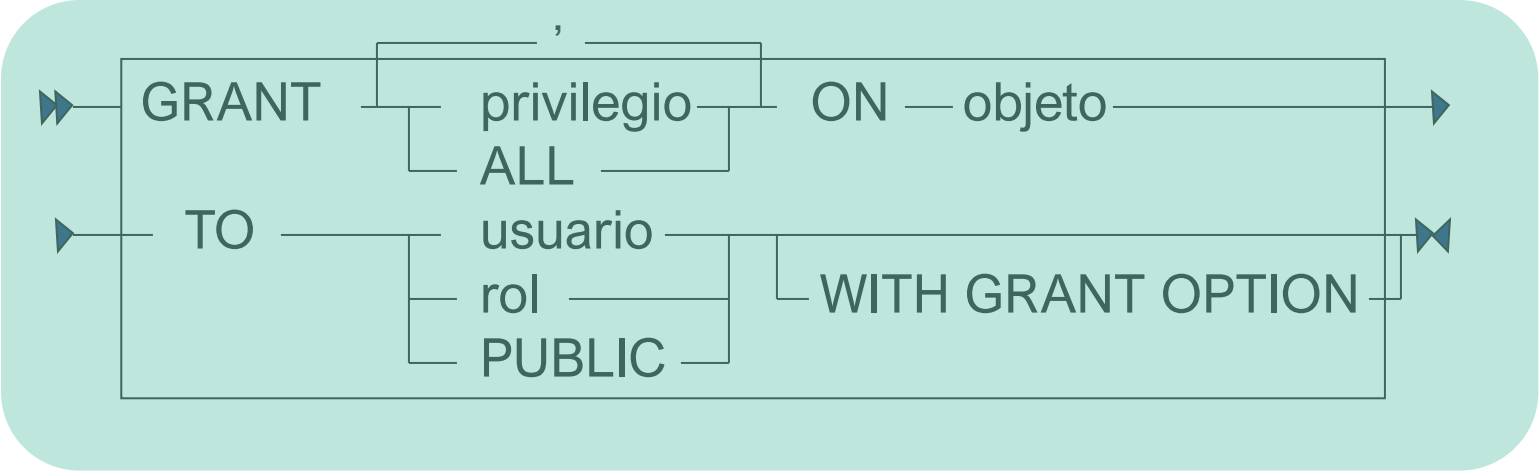
Define un rol, que es un conjunto de privilegios a otorgarse a usuarios u otros roles.

Data Control Language

- GRANT
  - Otorga privilegios de sistema a usuarios y roles.
  - 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 Sintaxis



Priviliegio	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;
```

Conclusiones

En esta sesión, debe haber comprendido lo siguiente:

- Uso de View, Index y Sequence en una base de datos
- Uso de comandos CREATE USER y CREATE ROL

Referencias

- AR. Elmasri y S.B. Navathe. (2007). Fundamentos de Sistema de Base de Datos, 5ta edición
- Oracle Help Center. (29 de septiembre de 2024). *Database SQL Reference*. https://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm

¡Gracias!

PUCP