

People matter, results count.

For internal use only

# Modulo SQL

Consultas a Multiples Tablas



# Relacionando múltiples tablas.

- Cada fila de datos en una tabla es identificada en forma única por la clave primaria (Primary Key).
- Podemos relacionar lógicamente datos de múltiples tablas usando las claves foráneas (Foreign Key).

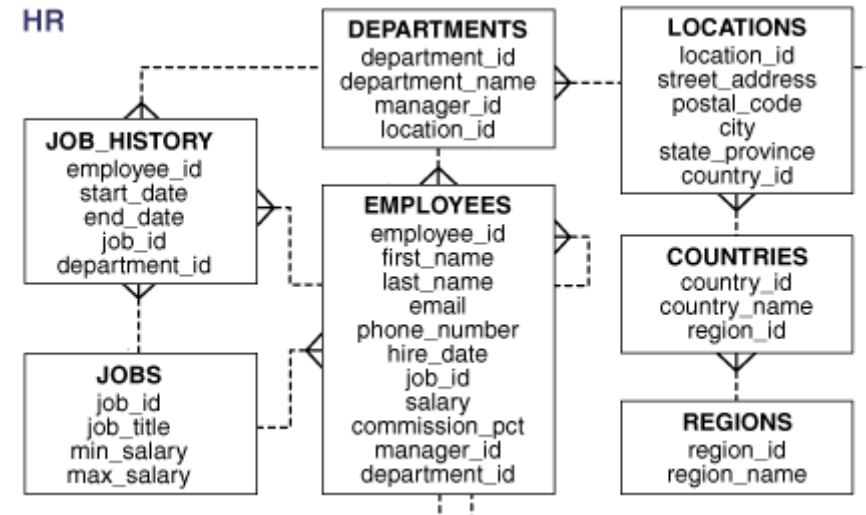


Table name: EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
174	Ellen	Abel	80
142	Curtis	Davies	50
102	Lex	De Haan	90
104	Bruce	Ernst	60
202	Pat	Fay	20
206	William	Gietz	110

Primary key

Foreign key

Table name: DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Primary key

# Seleccionando datos de múltiples tablas.

- **Obtener datos de empleados y el nombre del departamento al cual pertenecen.**
  - Qué datos nos están pidiendo ?
  - Dónde están los datos ?
  - Qué requisitos deben cumplir los registros ?

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing

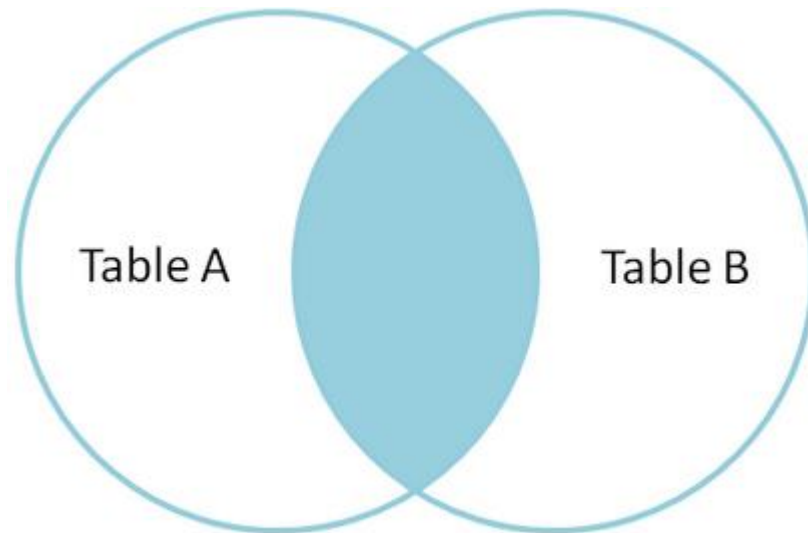
# JOIN.

- **La sentencia JOIN en SQL permite combinar registros de dos o más tablas en una base de datos relacional.**
  - ❖ **[Inner] Join**
  - ❖ **Left [outer] Join**
  - ❖ **Right [outer] Join**
  - ❖ **Full [outer] Join**

## Inner Join.

- Con esta operación se calcula el producto cruzado de todos los registros; así cada registro en la tabla A es combinado con cada registro de la tabla B; pero sólo permanecen aquellos registros en la tabla combinada que satisfacen las condiciones que se especifiquen.

```
SELECT * FROM TableA  
INNER JOIN TableB  
ON TableA.name = TableB.name
```



## Ejemplos.

- Listamos los empleados y sus departamentos.

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

- Utilizando la cláusula **WHERE**

The screenshot shows a SQL IDE with a query editor and a data grid. The query in the editor is:

```
--Empleados y sus departamentos  
select e.first_name, e.last_name, e.department_id, d.department_name  
from employees e, departments d  
where e.department_id=d.department_id
```

The data grid displays the results of the query:

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Steven	King	90	Executive
Neena	Kochhar	90	Executive
Lex	De Haan	90	Executive
Alexander	Hunold	60	IT

## Aplicando condiciones adicionales al JOIN.

- Agregamos una condición de filtro utilizando el **WHERE**.

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
175	Taylor	80	80	2500

# Otra forma de aplicar condiciones adicionales al JOIN.

- Utilizando la cláusula **WHERE**

```
45
46      --Empleados para el departamento Executive
47
48      select e.first_name, e.last_name, e.department_id, d.department_name
49      from employees e, departments d
50      where e.department_id=d.department_id
51      and e.manager_id=149
52
53
54
```

Data Grid

Data Grid | DBMS Output (disabled) | Script Output

Cancel

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Ellen	Abel	80	Sales
Alyssa	Hutton	80	Sales
Jonathon	Taylor	80	Sales
Jack	Livingston	80	Sales
Charles	Johnson	80	Sales



# Usando tres tablas para el JOIN.

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN    departments d
ON      d.department_id = e.department_id
JOIN    locations l
ON      d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

...

19 rows selected.

## Otra forma de usar tres tablas.

- Utilizando la cláusula **WHERE**

```
61  
62 select e.employee_id, l.city, d.department_name  
63 from employees e, departments d, locations l  
64 where e.department_id=d.department_id  
65 and d.location_id = l.location_id  
66
```

Data Grid			
Data Grid   DBMS Output (disabled)   Script Output			
[Icons] [Cancel]			
EMPLOYEE_ID	CITY	DEPARTMENT_NAME	
100	Seattle	Executive	
101	Seattle	Executive	
102	Seattle	Executive	
103	Southlake	IT	
104	Southlake	IT	

## INNER versus OUTER Join.

- El join entre dos tablas que devuelve como resultado sólo las filas coincidentes se denomina INNER JOIN.
  - El join entre dos tablas que retorna como los resultados del INNER JOIN como así también las filas que no son coincidentes se denomina OUTER JOIN.
  - El join entre dos tablas que retorna como los resultados del INNER JOIN como así también las filas que no son coincidentes de la tabla de la izquierda se denomina LEFT OUTER JOIN.
  - El join entre dos tablas que retorna como los resultados del INNER JOIN como así también las filas que no son coincidentes de la tabla de la derecha se denomina RIGHT OUTER JOIN.
-

# OUTER JOIN

- Mediante esta operación no se requiere que cada registro en las tablas a tratar tenga un registro equivalente en la otra tabla.

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

**There are no employees in  
department 190.**



EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...

20 rows selected.

# OUTER JOIN

- Usaremos como ejemplo las tablas **EMPLOYEES** y **DEPARTMENTS**.
  - Empleados sin departamentos asignados (LEFT OUTER JOIN).
  - Departamentos sin empleados asignados (RIGHT OUTER JOIN).

**DEPARTMENTS**

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

**There are no employees in  
department 190.**



**EMPLOYEES**

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...

20 rows selected.

# LEFT OUTER JOIN.

- Empleados sin departamentos asignados:
  - Aparece null en las columnas department\_id y department\_name.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

# RIGHT OUTER JOIN.

- Departamentos sin empleados asignados:
  - Aparece null en las columnas last\_name que corresponde a la tabla de empleados.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

# FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
	190	Contracting

21 rows selected.

- Esta operación presenta los resultados de tabla izquierda y tabla derecha aunque no tengan correspondencia en la otra tabla. La tabla combinada contendrá, entonces, todos los registros de ambas tablas y presentará valores nulos para registros sin pareja.



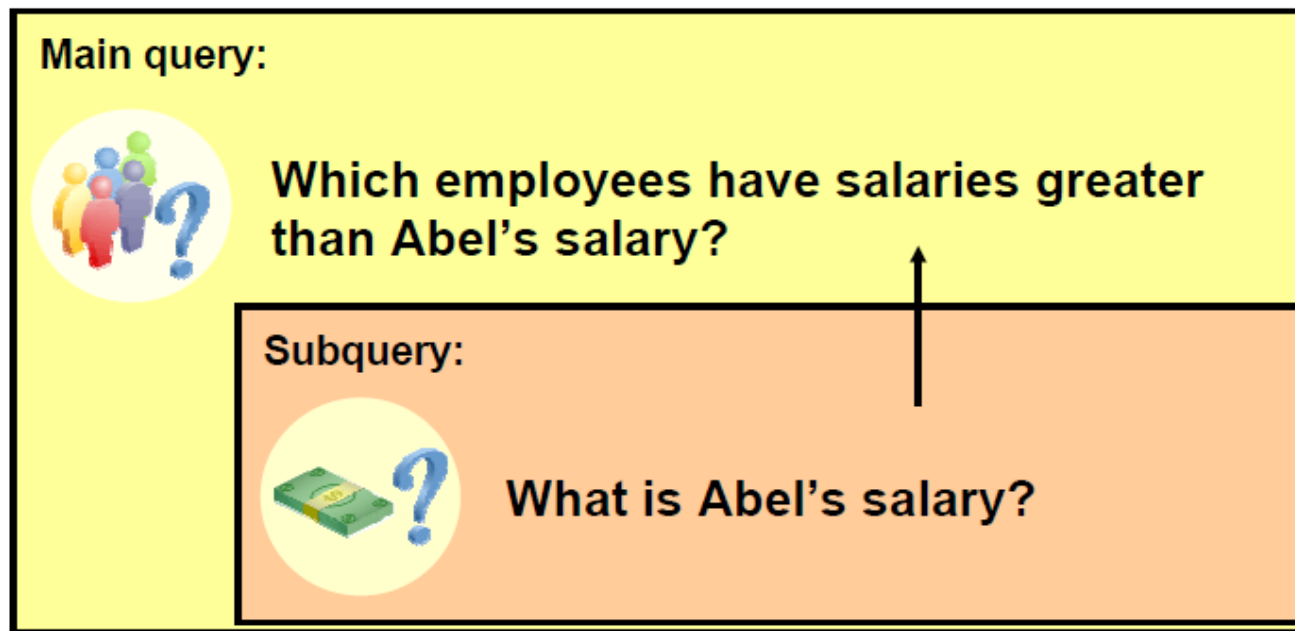


**Usando Subqueries para resolver queries.**

---

# Usando una subconsulta para resolver un problema.

- Cuáles son los empleados que tienen el salario mayor que el salario de Abel ?



# Sintaxis

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

- La subconsulta (inner query) ejecuta una vez antes de la consulta principal (outer query).
- El resultado de la subconsulta es utilizado por la consulta principal.

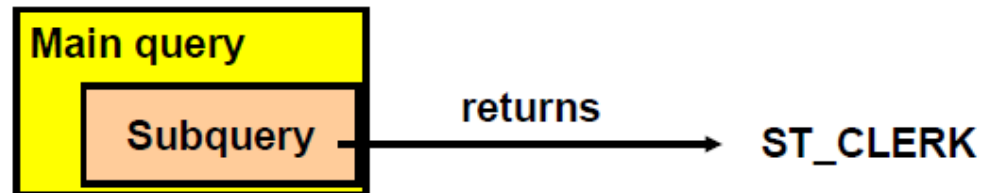
```
SELECT last_name, salary
FROM   employees 11000
WHERE  salary >
      (SELECT salary
       FROM   employees
       WHERE  last_name = 'Abel');
```

## Tener en cuenta al escribir subconsultas

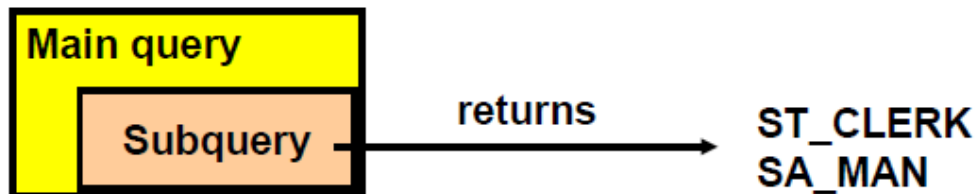
- Encerrar la subconsulta entre paréntesis.
  - Colocar la subconsulta del lado derecho de la condición de comparación.
  - Tener en cuenta usar los operadores que corresponda para realizar la comparación con la subconsulta: operadores de una sola fila con operadores de una sola fila y el uso de los operadores de varias filas con subconsultas de varias filas.
-

# Tipos de Subconsultas.

## ■ Single-Row



## ■ Multiple-Row

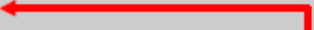



# Single-Row Subqueries.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

- Retornan una sola fila.
- Hay que usar los operadores de comparación de una sola fila.

## Ejemplo single-row subqueries.

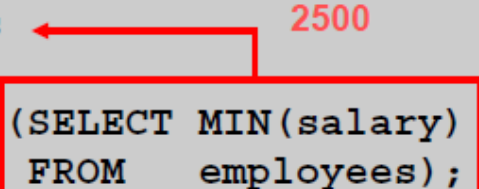
```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =  ST_CLERK
      (SELECT job_id
       FROM   employees
       WHERE  employee_id = 141)
AND    salary >  2600
      (SELECT salary
       FROM   employees
       WHERE  employee id = 143);
```

- La subconsulta devuelve un solo valor a la consulta principal por lo que utilizamos los operadores que corresponden para realizar las comparaciones.

## Otros ejemplos ...

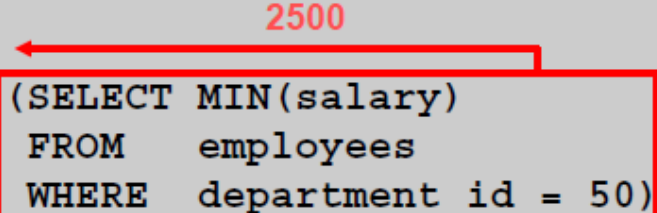
- Usando funciones de grupo

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
      (SELECT MIN(salary)
       FROM   employees);
```



- Usando la cláusula HAVING

```
SELECT  department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING  MIN(salary) >
      (SELECT MIN(salary)
       FROM    employees
       WHERE   department_id = 50);
```





# Qué está incorrecto en la siguiente consulta ?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
      (SELECT  MIN(salary)
       FROM    employees
       GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

**Single-row operator with multiple-row subquery**

## La siguiente sentencia retorna filas ?

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  last_name = 'Haas');
```

```
no rows selected
```

**Subquery returns no values.**

## Multiple-Row Subqueries.

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

- Retornan más de una fila.
- Hay que usar los operadores de comparación para más de una fila.

# Uso de los operadores

## ■ Operador ANY

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

## ■ Operador ALL

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

# Revisión Cumplimiento Objetivos

- ✓ Revisar conceptos generales de una Base de Datos
  - ✓ Brindar los conocimientos básicos del lenguaje de gestión de base de datos relacionales (SQL).
  - ✓ Aplicar dichos conocimientos a un motor de base de datos.
-