

Unidad 3

09 - SQL Avanzado



Funciones Multi-fila (de grupo)





Funciones multi-fila (de grupo)

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	90	24000
2	90	17000
3	90	17000
4	60	9000
5	60	6000
6	60	4200
7	50	5800
8	50	3500
9	50	3100
10	50	2600
...		
18	20	6000
19	110	12000
20	110	8300

**Maximum salary in
EMPLOYEES table**

MAX(SALARY)
24000



Funciones multi-fila

```
SELECT      group_function(column), ...  
FROM        table  
[WHERE      condition]  
[ORDER BY   column];
```

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

	A 1	A 2	A 3	A 4
	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

```
SELECT MIN(hire_date), MAX(hire_date)  
FROM   employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	17-JUN-87	29-JAN-00



Funciones multi-fila

```
SELECT group_function(column), ...  
FROM   table  
[WHERE condition]  
[ORDER BY column];
```

```
SELECT COUNT(*)  
FROM   employees  
WHERE  department_id = 50;
```

	AZ	COUNT(*)
1		5

```
SELECT COUNT(commission_pct)  
FROM   employees  
WHERE  department_id = 80;
```

	AZ	COUNT(COMMISSION_PCT)
1		3

```
SELECT COUNT(DISTINCT department_id)  
FROM   employees;
```

	AZ	COUNT(DISTINCTDEPARTMENT_ID)
1		7



```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id ;
```

GROUP BY

```
SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	90	19333.333333333333...
3	20	9500
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	60	6400
8	10	4400

```
SELECT AVG(salary)
FROM employees
GROUP BY department_id ;
```

	AVG(SALARY)
1	7000
2	19333.333333333333...
3	9500
4	10150
5	3500
6	10033.333333333333...
7	6400
8	4400



```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY  department_id, job_id
ORDER BY  department_id;
```

	A Z	DEPARTMENT_ID	A Z	JOB_ID	A Z	SUM(SALARY)
1		10		AD_ASST		4400
2		20		MK_MAN		13000
3		20		MK_REP		6000
4		50		ST_CLERK		11700
5		50		ST_MAN		5800
6		60		IT_PROG		19200
7		80		SA_MAN		10500
8		80		SA_REP		19600
9		90		AD PRES		24000
10		90		AD_VP		34000
11		110		AC_ACCOUNT		8300
12		110		AC_MGR		12000
13		(null)		SA_REP		7000



```
SELECT department_id, COUNT(last_name)
FROM employees;
```

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id;
```

```
SELECT department_id, AVG(salary)
FROM employees
WHERE AVG(salary) > 8000
GROUP BY department_id;
```

Errores comunes



```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000 ;
```

GROUP BY HAVING

```
SELECT    column, group_function
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[HAVING   group_condition]
[ORDER BY column];
```

	DEPARTMENT_ID	MAX(SALARY)
1	90	24000
2	20	13000
3	110	12000
4	80	11000

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY  job_id
HAVING    SUM(salary) > 13000
ORDER BY  SUM(salary);
```

	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000



Vamos a ejercitarnos otro poquito

- Abrir en DBeaver la base de datos HR.db
- Realizar las consultas propuestas en el ejercicio





Ejercicio básicos

1. Encuentre el salario más alto, más bajo, suma y promedio de todos los empleados. Etiquete las columnas **Maximum**, **Minimum**, **Sum** y **Average**, respectivamente. Redondea tus resultados al número entero más cercano.
2. Modifique la consulta anterior para mostrar el salario mínimo, máximo, suma y promedio para cada tipo de trabajo.
3. Escriba una consulta para mostrar el número de personas con el mismo trabajo.
4. Determine la cantidad de gerentes sin nombrarlos. Etiquete la columna **Number of Managers**.

Sugerencia: use la columna `MANAGER_ID` para determinar el número de administradores.

5. Encuentra la diferencia entre el salario más alto y el más bajo. Etiquete la columna **DIFFERENCE**.



Ejercicio avanzados

6. Cree un informe para mostrar el número de gerente y el salario del empleado con el salario más bajo para ese gerente. Excluir a cualquiera cuyo gerente no sea conocido. Excluya cualquier grupo donde el salario mínimo sea de \$ 6,000 o menos. Ordene la salida en orden descendente de salario.
7. Cree una consulta para mostrar el número total de empleados y, de ese total, el número de empleados contratados en 2005, 2006, 2007 y 2008. Cree encabezados de columna apropiados.
8. Cree una consulta matricial para mostrar el trabajo, el salario de ese trabajo según el número de departamento y el salario total de ese trabajo, para los departamentos 20, 50, 80 y 90, dando a cada columna un encabezado apropiado.

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
-----	---------	---------	---------	---------	-------

Mostrando datos de múltiples tablas





Uniendo tablas SQL

Natural joins

- NATURAL JOIN
- USING
- ON

Self-join

Nonequijoins

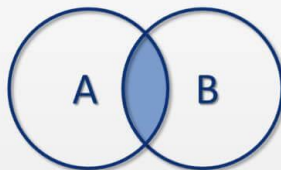
Outer join

- LEFT OUTER
- RIGHT OUTER
- FULL OUTER

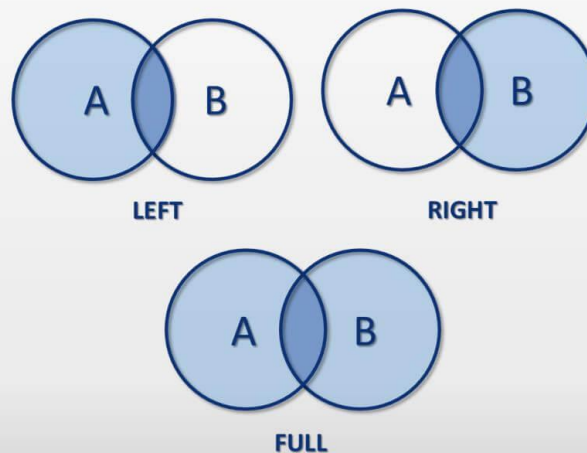
Cross join

```
SELECT  table1.column, table2.column
FROM    table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

INNER JOIN



OUTER JOIN





Natural Join

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford



JOIN - USING

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
       USING (department_id) ;
```

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	124	Mourgos	1500	50
5	144	Vargas	1500	50
6	143	Matos	1500	50
7	142	Davies	1500	50
8	141	Rajs	1500	50
9	107	Lorentz	1400	60
10	104	Ernst	1400	60
...				
19	205	Higgins	1700	110

```
SELECT l.city, d.department_name  
FROM   locations l JOIN departments d  
       USING (location_id)  
WHERE  d.location_id = 1400;
```





INNER JOIN / JOIN - ON

```
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);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	
10	104	Ernst	60	60	

...

```
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;
```



SELF JOIN

```
SELECT worker.last_name emp, manager.last_name mgr  
FROM   employees worker JOIN employees manager  
ON     (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King



NOEQUIJOIN

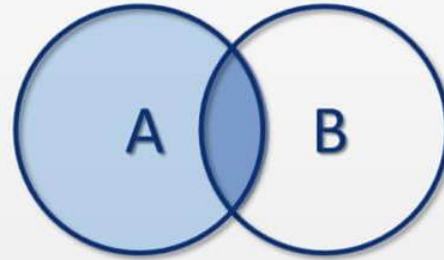
```
SELECT e.last_name, e.salary, j.grade_level  
FROM employees e JOIN job_grades j  
ON e.salary  
BETWEEN j.lowest_sal AND j.highest_sal;
```

	A2	LAST_NAME	A2	SALARY	A2	GRADE_LEVEL
1		Vargas		2500		A
2		Matos		2600		A
3		Davies		3100		B
4		Rajs		3500		B
5		Lorentz		4200		B
6		Whalen		4400		B
7		Mourgos		5800		B
8		Ernst		6000		C
9		Fay		6000		C
10		Grant		7000		C

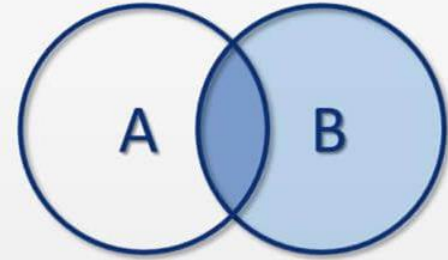


OUTER JOIN

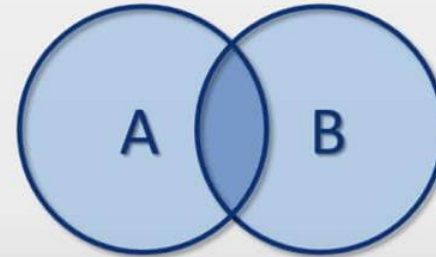
OUTER JOIN



LEFT



RIGHT



FULL



LEFT OUTER JOIN

```
1 SELECT e.last_name, e.department_id, d.department_name
2 FROM employees e
3 LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);
4
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Raphaely	30	Purchasing
Khoo	30	Purchasing
Baida	30	Purchasing
Tobias	30	Purchasing

...

Popp	100	Finance
Higgins	110	Accounting
Gietz	110	Accounting
Grant	-	-

Download CSV

107 rows selected.



RIGHT OUTER JOIN

```
1 SELECT e.last_name, e.department_id, d.department_name
2 FROM employees e
3 RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);
4
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Hunold	60	IT
Ernst	60	IT
Austin	60	IT
Pataballa	60	IT
Lorentz	60	IT
Greenberg	100	Finance

...

Gietz	110	Accounting
-	-	IT Support
-	-	Operations
-	-	Payroll



FULL OUTER JOIN

```
1 SELECT e.last_name, e.department_id, d.department_name
2 FROM employees e
3 FULL OUTER JOIN departments d ON (e.department_id = d.department_id);
4
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Hunold	60	IT
Ernst	60	IT
Austin	60	IT
Pataballa	60	IT
Lorentz	60	IT

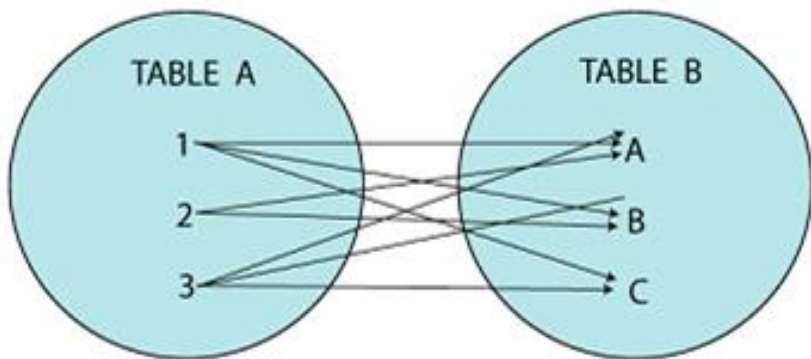
...

Higgins	110	Accounting
Grant	-	-
-	-	Treasury
-	-	Manufacturing
-	-	Corporate Tax



CROSS JOIN

```
SELECT last_name, department_name  
FROM employees  
CROSS JOIN departments ;
```



	R2	LAST_NAME	R2	DEPARTMENT_NAME
1		Abel		Administration
2		Davies		Administration
3		De Haan		Administration
4		Ernst		Administration
5		Fay		Administration

...

159		Whalen		Contracting
160		Zlotkey		Contracting



Vamos a ejercitarnos otro poquito

- Abrir en DBeaver la base de datos HR.db
- Realizar las consultas propuestas en el ejercicio





Ejercicio básicos

1. Escriba una consulta para el departamento de recursos humanos para producir las direcciones de todos los departamentos. Utilice las tablas de LOCATIONS y COUNTRIES. Muestre la identificación de ubicación, la dirección, la ciudad, el estado o la provincia y el país en la salida. Use una NATURAL JOIN para producir los resultados.
2. El departamento de recursos humanos necesita un informe de sólo aquellos empleados con los departamentos correspondientes. Escriba una consulta para mostrar el apellido, el número de departamento y el nombre de departamento para estos empleados.
3. El departamento de recursos humanos necesita un informe de los empleados en Toronto. Muestre el apellido, el trabajo, el número de departamento y el nombre del departamento para todos los empleados que trabajan en Toronto.



Ejercicio básicos

4. Cree un informe para mostrar el apellido y el número del empleado junto con el apellido y el número del gerente. Etiquete las **Employee**, **Emp#**, **Manager** y **Mgr#**, respectivamente.
5. Modifique el informe anterior para mostrar a todos los empleados, incluido King, que no tiene gerente. Ordene los resultados por el número de empleado.
6. Cree un informe para el departamento de recursos humanos que muestre los apellidos de los empleados, los números de departamento y todos los empleados que trabajan en el mismo departamento que un empleado determinado. Dé a cada columna una etiqueta apropiada.
7. El departamento de recursos humanos necesita un informe sobre las calificaciones laborales (tabla JOB_GRADES) y los salarios. Cree una consulta que muestre el nombre, el trabajo, el nombre del departamento, el salario y la calificación de todos los empleados.



Ejercicio avanzados

8. El departamento de recursos humanos quiere determinar los nombres de todos los empleados que fueron contratados después de Davies. Cree una consulta para mostrar el nombre y la fecha de contratación de cualquier empleado contratado después del empleado Davies.
9. El departamento de recursos humanos necesita encontrar los nombres y las fechas de contratación de todos los empleados que fueron contratados antes que sus gerentes, junto con los nombres y fechas de contratación de sus gerentes.

Subconsultas





Subconsultas

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

```
1 select salary
2 from employees
3 where last_name = 'Abel'
4
```

SALARY

11000

[Download CSV](#)

```
1 SELECT last_name, salary
2 FROM employees
3 WHERE salary > (
4     select salary
5     from employees
6     where last_name = 'Abel'
7 );
```

LAST_NAME

SALARY

King

24000



Subconsultas de fila única

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

```
1 SELECT last_name, job_id, salary
2 FROM employees
3 WHERE job_id = (
4     select job_id
5     from employees
6     where first_name = 'Jonathon'
7 )
8 AND salary > (
9     select salary
10    from employees
11    where first_name = 'Jonathon'
12 );
```

LAST_NAME	JOB_ID	SALARY
Tucker	SA_REP	10000
Bernstein	SA_REP	9500
Hall	SA_REP	9000
...	...	...

```
1 SELECT last_name, job_id, salary
2 FROM employees
3 WHERE salary = (
4     SELECT MIN(salary)
5     FROM employees
6 );
```

LAST_NAME	JOB_ID	SALARY
Olson	ST_CLERK	2100

[Download CSV](#)





Subconsultas de fila única

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

```
1 SELECT department_id, MIN(salary)
2 FROM employees
3 GROUP BY department_id
4 HAVING MIN(salary) > (
5     SELECT MIN(salary)
6     FROM employees
7     WHERE department_id = 50
8 );
```

DEPARTMENT_ID	MIN(SALARY)
40	6500
110	8300
90	17000

```
1 SELECT employee_id, last_name
2 FROM employees
3 WHERE salary = (
4     SELECT MIN(salary)
5     FROM employees
6     GROUP BY department_id
7 );
```

ORA-01427: single-row subquery returns more than one row



Subconsultas multi-fila

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. Compares a value to each value in a list or returned by a query. Evaluates to FALSE if the query returns no rows.
ALL	Must be preceded by =, !=, >, <, <=, >=. Compares a value to every value in a list or returned by a query. Evaluates to TRUE if the query returns no rows.

```
1 SELECT employee_id, first_name, last_name
2 FROM employees
3 WHERE department_id IN (1, 3, 8, 10, 11)
4 ORDER BY first_name, last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
200	Jennifer	Whalen

[Download CSV](#)

```
1 SELECT employee_id, first_name, last_name
2 FROM employees
3 WHERE department_id IN (
4     SELECT department_id
5     FROM departments
6     WHERE location_id = 1700
7 )
8 ORDER BY first_name, last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
115	Alexander	Khoo
109	Daniel	Faviet



Subconsultas multi-fila

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. Compares a value to each value in a list or returned by a query. Evaluates to FALSE if the query returns no rows.
ALL	Must be preceded by =, !=, >, <, <=, >=. Compares a value to every value in a list or returned by a query. Evaluates to TRUE if the query returns no rows.

```
1 SELECT employee_id, first_name, last_name
2 FROM employees
3 WHERE department_id NOT IN (
4     SELECT department_id
5     FROM departments
6     WHERE location_id = 1700
7 )
8 ORDER BY first_name , last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
121	Adam	Fripp
106	Alexs	Walek

```
1 SELECT employee_id, first_name, last_name, salary
2 FROM employees
3 WHERE salary >= ALL (
4     SELECT MIN(salary)
5     FROM employees
6     GROUP BY department_id
7 )
8 ORDER BY first_name , last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
102	Lex	De Haan	17000



Subconsultas multi-fila

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. Compares a value to each value in a list or returned by a query. Evaluates to FALSE if the query returns no rows.
ALL	Must be preceded by =, !=, >, <, <=, >=. Compares a value to every value in a list or returned by a query. Evaluates to TRUE if the query returns no rows.

```
1 SELECT employee_id, first_name, last_name, salary
2 FROM employees
3 WHERE salary >= ANY (
4     SELECT MAX(salary)
5     FROM employees
6     GROUP BY department_id
7 );
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000

```
1 SELECT employee_id, first_name, last_name, salary
2 FROM employees
3 WHERE salary >= SOME (
4     SELECT MAX(salary)
5     FROM employees
6     GROUP BY department_id
7 );
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000





Subconsultas como tabla

```
1 SELECT ROUND(AVG(average_salary), 0)
2 FROM (
3     SELECT AVG(salary) average_salary
4     FROM employees
5     GROUP BY department_id
6 ) department_salary;
```

ROUND(AVG(AVERAGE_SALARY),0)
8153

[Download CSV](#)



Subconsultas como atributo

```
1 SELECT employee_id, first_name, last_name, salary,  
2    (SELECT ROUND(AVG(salary), 0) FROM employees) average_salary,  
3    salary - (SELECT ROUND(AVG(salary), 0) FROM employees) difference  
4 FROM employees  
5 ORDER BY first_name , last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	AVERAGE_SALARY	DIFFERENCE
121	Adam	Fripp	8200	6462	1738



Subconsulta con comparación multicolumna

```
1 SELECT employee_id, manager_id, department_id
2 FROM employees
3 WHERE (manager_id, department_id) IN (
4     SELECT manager_id, department_id
5     FROM employees
6     WHERE first_name = 'John')
7 AND first_name <> 'John';
8
```

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
109	108	100
111	108	100
112	108	100



Subconsultas correlacionadas

```
1 SELECT employee_id, first_name, last_name
2 FROM employees
3 WHERE department_id IN (
4     SELECT department_id
5     FROM departments
6     WHERE location_id = 1700
7 )
8 ORDER BY first_name , last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
115	Alexander	Khoo
109	Daniel	Faviet

```
1 SELECT department_name
2 FROM departments d
3 WHERE NOT EXISTS(
4     SELECT 1
5     FROM employees e
6     WHERE salary > 10000
7     AND e.department_id = d.department_id
8 )
9 ORDER BY department_name;
```

DEPARTMENT_NAME
Administration
Benefits



Subconsultas correlacionadas

```
1 SELECT employee_id, first_name, last_name, department_name, salary,  
2       (SELECT ROUND(AVG(salary),0)  
3       FROM employees  
4       WHERE department_id = e.department_id) avg_salary_in_department  
5 FROM employees e  
6 INNER JOIN departments d ON d.department_id = e.department_id  
7 ORDER BY department_name, first_name, last_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	SALARY	AVG_SALARY_IN_DEPARTMENT
205	Shelley	Higgins	Accounting	12008	10154
206	William	Gietz	Accounting	8300	10154

```
1 select last_name  
2 from employees  
3 where department_id in (  
4     select department_id  
5     from departments  
6     where location_id in (  
7         select location_id  
8         from locations  
9         where country_id = (  
10            select country_id  
11            from countries  
12            where country_name='United Kingdom'  
13        )  
14    )  
15 );
```

LAST_NAME



Cláusula WITH

Usando la cláusula WITH, puedes reutilizar un bloque SELECT más de una vez en la misma consulta.

Los resultados obtenidos, son guardados temporalmente en memoria.

Se puede utilizar para mejorar el rendimiento.

```
1 WITH
2 dept_costs AS (
3     SELECT d.department_name, SUM(e.salary) AS dept_total
4     FROM employees e
5     JOIN departments d ON e.department_id = d.department_id
6     GROUP BY d.department_name
7 ),
8 avg_cost AS (
9     SELECT SUM(dept_total)/COUNT(*) AS dept_avg
10    FROM dept_costs
11 )
12 SELECT *
13 FROM dept_costs
14 WHERE dept_total > (
15     SELECT dept_avg
16     FROM avg_cost
17 )
18 ORDER BY department_name;
```

DEPARTMENT_NAME	DEPT_TOTAL
Sales	304500
Shipping	156400

Download CSV

2 rows selected.



Vamos a ejercitarnos otro poquito

- Abrir en DBeaver la base de datos HR.db
- Realizar las consultas propuestas en el ejercicio





Ejercicio básicos

1. El departamento de recursos humanos necesita una consulta que solicite al usuario el apellido de un empleado. La consulta luego muestra el apellido y la fecha de contratación de cualquier empleado en el mismo departamento que el empleado cuyo nombre proporcionan (excluyendo a ese empleado). Por ejemplo, si el usuario ingresa a Zlotkey, busque todos los empleados que trabajan con Zlotkey (excluyendo Zlotkey).
2. Cree un informe que muestre el número de empleado, el apellido y el salario de todos los empleados que ganan más que el salario promedio. Ordene los resultados en orden de salario ascendente.
3. Escriba una consulta que muestre el número de empleado y el apellido de todos los empleados que trabajan en un departamento con cualquier empleado cuyo apellido contenga la letra "u".



Ejercicio básicos

4. El departamento de recursos humanos necesita un informe que muestre el apellido, el número de departamento y la identificación de trabajo de todos los empleados cuya identificación de ubicación de departamento es 1700.
5. Cree un informe para Recursos Humanos que muestre el apellido y el salario de cada empleado que se reporta a King.
6. Cree un informe para RR. HH. Que muestre el número de departamento, el apellido y la identificación del trabajo para cada empleado del departamento Ejecutivo.
7. Cree un informe que muestre una lista de todos los empleados cuyo salario es mayor que el de cualquier empleado del departamento 60.

Operaciones de conjuntos





El futuro digital
es de todos

MinTIC

Operaciones de conjunto



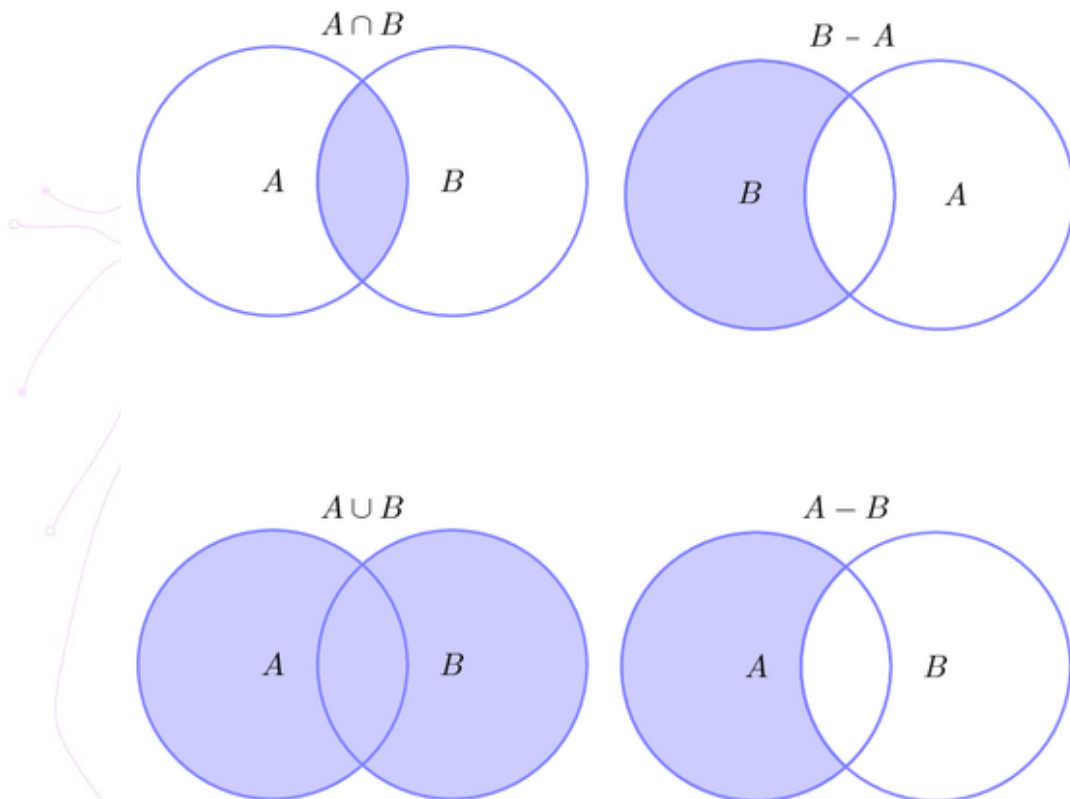
Misión
TIC2022

UNION

UNION ALL

INTERSECT

MINUS





UNION

```
SELECT employee_id, job_id  
FROM employees  
UNION  
SELECT employee_id, job_id  
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
1	100 AD_PRES
2	101 AC_ACCOUNT
...	
22	200 AC_ACCOUNT
23	200 AD_ASST
24	201 MK_MAN



UNION ALL

```
SELECT employee_id, job_id, department_id  
FROM employees  
UNION ALL  
SELECT employee_id, job_id, department_id  
FROM job_history  
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
...		
16	144 ST_CLERK	50
17	149 SA_MAN	80
18	174 SA_REP	80
19	176 SA_REP	80
20	176 SA_MAN	80
21	176 SA_REP	80
22	178 SA_REP	(null)
...		
30	206 AC_ACCOUNT	110



INTERSECT

```
SELECT employee_id, job_id  
FROM employees  
INTERSECT
```

```
SELECT employee_id, job_id  
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST



MINUS

```
SELECT employee_id  
FROM employees  
MINUS  
SELECT employee_id  
FROM job_history;
```

	AZ	EMPLOYEE_ID
1		100
2		103
3		104
4		107
5		124

...

14		205
15		206



Características a tener en cuenta con operaciones de conjuntos

```
SELECT employee_id, job_id, salary
FROM employees
UNION
SELECT employee_id, job_id, 0
FROM job_history;
```

Columnas debe coincidir
en sus tipos

Si existe ordenamiento,
sólo debe existir una vez
y al final de la sentencia

```
SELECT location_id, department_name "Department",
       TO_CHAR(NULL) "Warehouse location"
FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department",
       state_province
FROM locations
ORDER BY location_id, "Warehouse location";
```



Vamos a ejercitarnos otro poquito

- Abrir en DBeaver la base de datos HR.db
- Realizar las consultas propuestas en el ejercicio





Ejercicio básicos

1. El departamento de recursos humanos necesita una lista de ID de departamento para los departamentos que no contienen la ID de trabajo **'Stock Clerk'**.
2. El departamento de recursos humanos necesita una lista de países que no tienen departamentos ubicados en ellos. Muestra la identificación del país y el nombre de los países.
3. Produzca una lista de trabajos para los departamentos 10, 50 y 20, en ese orden. Mostrar la ID del trabajo y la ID del departamento.



Ejercicio básicos

4. Cree un informe que enumere las identificaciones de los empleados y las identificaciones de los empleados que actualmente tienen un título de trabajo que es el mismo que el de su trabajo cuando fueron contratados inicialmente por la empresa (es decir, cambiaron de trabajo, pero ahora han vuelto a haciendo su trabajo original).
5. El departamento de recursos humanos necesita un informe con las siguientes especificaciones:
 - a. Apellido e identificación del departamento de todos los empleados de la tabla EMPLOYEES, independientemente de si pertenecen o no a un departamento
 - b. ID de departamento y nombre de departamento de todos los departamentos de la tabla DEPARTMENTS, independientemente de si tienen empleados trabajando en ellos

Escriba una consulta compuesta para lograr esto.

Manipulando datos DML

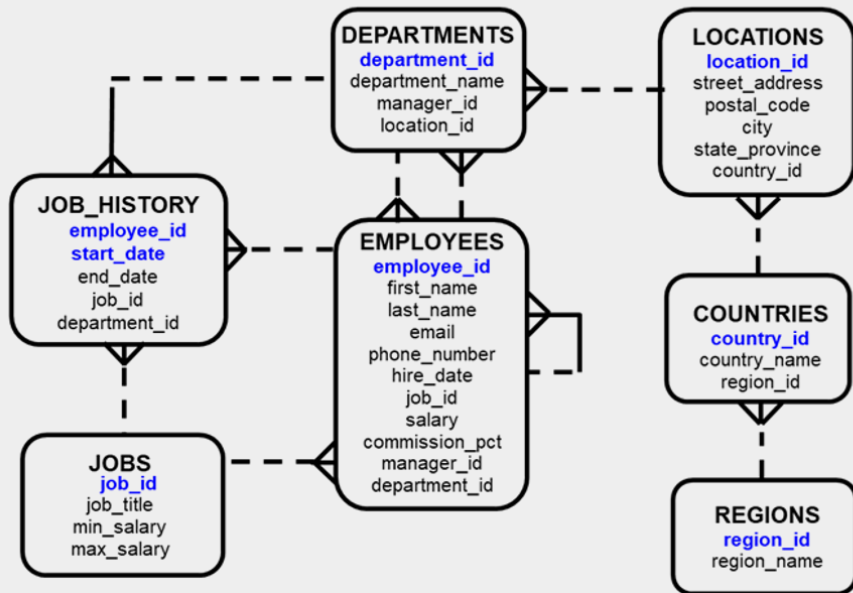




Modelo de Datos

Estructura base de datos de
ejemplo

The Human Resources (HR) Schema





Insertando nuevas filas

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);
```

1 rows inserted

```
INSERT INTO departments (department_id,  
                        department_name)  
VALUES (30, 'Purchasing');
```

1 rows inserted

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);
```

1 rows inserted



Insertando nuevas filas

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

```
INSERT INTO sales_reps(id, name, salary, commission_pct)  
SELECT employee_id, last_name, salary, commission_pct  
FROM employees  
WHERE job_id LIKE '%REP%';
```

4 rows inserted



Actualizando filas

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

1 rows updated

```
UPDATE    copy_emp
SET       department_id = 110;
```

22 rows updated



Actualizando filas

```
UPDATE      table
SET          column = value [, column = value, ...]
[WHERE      condition];
```

```
UPDATE      employees
SET          job_id = (SELECT  job_id
                        FROM      employees
                        WHERE     employee_id = 205),
            salary = (SELECT  salary
                      FROM      employees
                      WHERE     employee_id = 205)
WHERE        employee_id = 113;
```

1 rows updated



Actualizando filas

```
UPDATE      table
SET          column = value [, column = value, ...]
[WHERE      condition];
```

```
UPDATE copy_emp
SET    department_id = (SELECT department_id
                        FROM employees
                        WHERE employee_id = 100)
WHERE  job_id = (SELECT job_id
                 FROM employees
                 WHERE employee_id = 200);
```

```
1 rows updated
```



Eliminando filas

```
DELETE [FROM] table  
[WHERE condition];
```

```
DELETE FROM departments  
WHERE department_name = 'Finance';
```

1 rows deleted

```
DELETE FROM copy_emp;
```

22 rows deleted

```
DELETE FROM employees  
WHERE department_id =
```

```
(SELECT department_id  
FROM departments  
WHERE department_name  
LIKE '%Public%');
```

1 rows deleted



Vamos a ejercitarnos otro poquito

- Abrir en DBeaver la base de datos HR.db
- Realizar las consultas propuestas en el ejercicio





Ejercicio básicos

1. Cree una instrucción INSERT para agregar la primera fila de datos a la tabla MY_EMPLOYEE a partir de los siguientes datos de muestra. No nombre las columnas en la cláusula INSERT. No ingrese todas las filas todavía.
2. Rellene la tabla MY_EMPLOYEE con la segunda fila de los datos de muestra de la lista anterior. Esta vez, enumere las columnas explícitamente en la cláusula INSERT.
3. Verifique su inserción a la tabla.

```
CREATE TABLE my_employee (  
    id NUMBER(4) NOT NULL,  
    last_name VARCHAR2(25),  
    first_name VARCHAR2(25),  
    userid VARCHAR2(8),  
    salary NUMBER(9,2)  
);
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550



Ejercicio básicos

4. Rellene la tabla MY_EMPLOYEE con las siguientes dos filas de los datos de muestra de la lista anterior.
5. Verifique su inserción a la tabla.
6. Hacer que las adiciones de datos sean permanentes
7. Cambie el apellido del empleado 3 a Drexler.
8. Cambie el salario a \$ 1.000 para todos los empleados que tengan un salario inferior a \$ 900.
9. Verifique sus cambios en la tabla.

```
CREATE TABLE my_employee (  
    id NUMBER(4) NOT NULL,  
    last_name VARCHAR2(25),  
    first_name VARCHAR2(25),  
    userid VARCHAR2(8),  
    salary NUMBER(9,2)  
);
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550



Ejercicio básicos

10. Eliminar Betty Dancs de la tabla MY_EMPLOYEE.
11. Verifique sus cambios en la tabla.
12. Rellene la tabla con la última fila de los datos de muestra.
13. Verifique su inserción a la tabla.
14. Elimine todas las filas de la tabla MY_EMPLOYEE.
15. Verifique que la tabla está vacía.

```
CREATE TABLE my_employee (  
    id NUMBER(4) NOT NULL,  
    last_name VARCHAR2(25),  
    first_name VARCHAR2(25),  
    userid VARCHAR2(8),  
    salary NUMBER(9,2)  
);
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550



Ejercicio básicos

```
CREATE TABLE my_employee (  
  id NUMBER(4) NOT NULL,  
  last_name VARCHAR2(25),  
  first_name VARCHAR2(25),  
  userid VARCHAR2(8),  
  salary NUMBER(9,2)  
);
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

Transacciones TCL





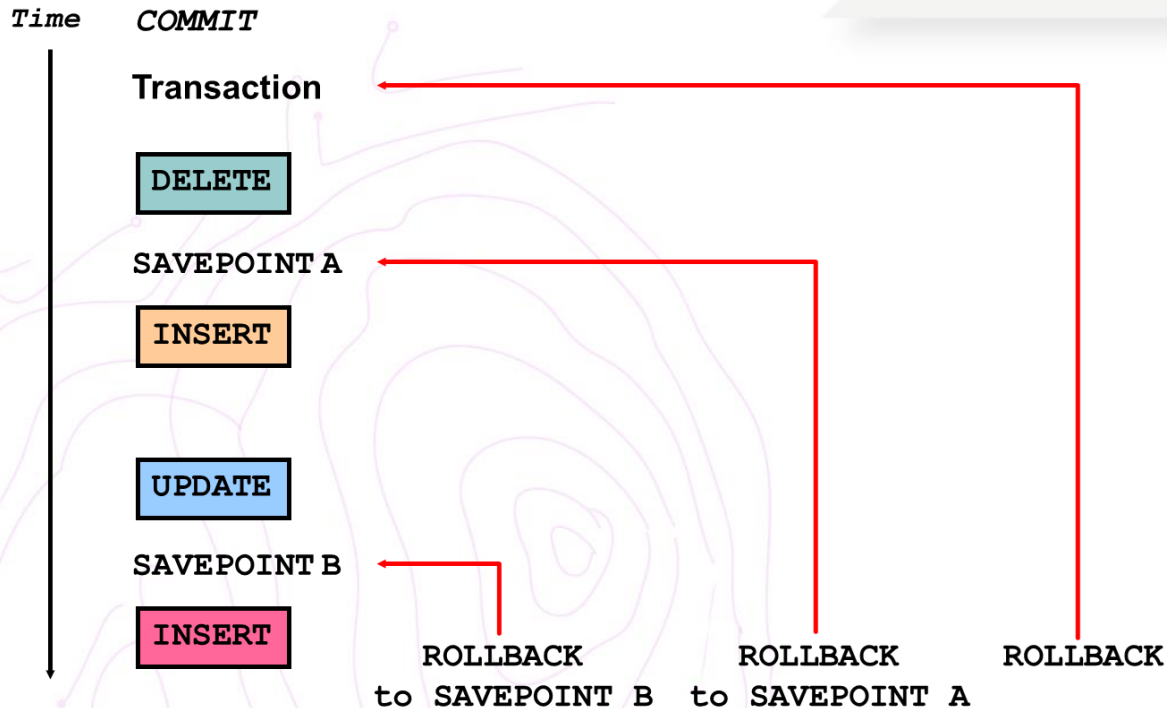
Control de Transacciones

Transacciones cumplen criterios ACID:

- Atomicity (Atomicidad)
- Consistency (Consistencia)
- Isolation (Aislamiento)
- Durability (Durabilidad)

Eventos Transacciones

- DML se ejecuta en ambiente de usuario (Aislado)
- SAVEPOINT es un punto intermedio de recuperación.
- COMMIT o ROLLBACK
- DDL o DCL realiza COMMIT automático





Devolviendo Cambios

```
UPDATE...
```

```
SAVEPOINT update_done;
```

```
SAVEPOINT update_done succeeded.
```

```
INSERT...
```

```
ROLLBACK TO update_done;
```

```
ROLLBACK TO succeeded.
```

```
DELETE FROM copy_emp;
```

```
ROLLBACK ;
```



Devolviendo o confirmando cambios

```
DELETE FROM test;  
25,000 rows deleted.
```

```
ROLLBACK;  
Rollback complete.
```

```
DELETE FROM test WHERE id = 100;  
1 row deleted.
```

```
SELECT * FROM test WHERE id = 100;  
No rows selected.
```

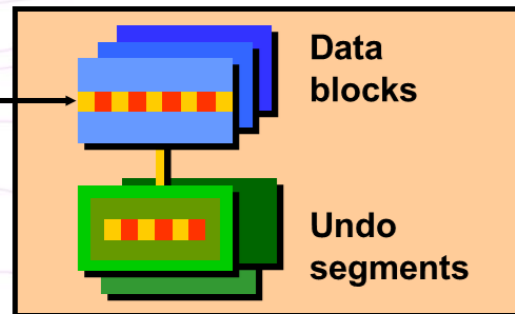
```
COMMIT;  
Commit complete.
```



User A

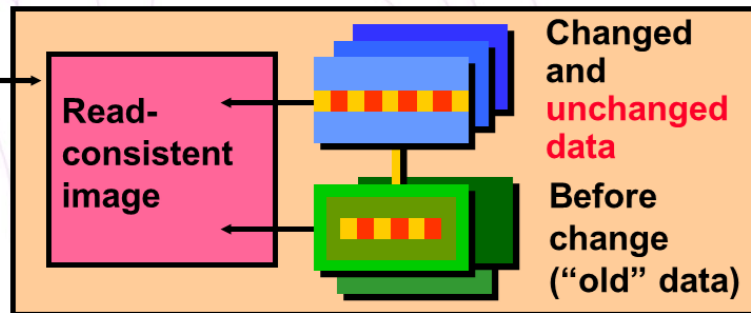


```
UPDATE employees  
SET    salary = 7000  
WHERE  last_name = 'Grant';
```



User B

```
SELECT *  
FROM userA.employees;
```



Implementando consistencia de lectura



Para la próxima sesión...

- Terminar los ejercicios que no se terminaron... (si aplica)