UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL MENDOZA

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

CÁTEDRA DE GESTIÓN DE DATOS
3º AÑO

TRABAJO ESPECIAL

"Secuencia Didáctica de Comandos del Lenguaje de Consultas Estructurado (SQL) - Ejemplos y Resultados"

Ing. Santiago C. Pérez
Laura Noussan Lettry
Carlos Campos

INDICE

SINTAXIS SQL

• LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

CREATE TABLE

```
CREATE TABLE [schema.]tabla
      (columna tipodato [DEFAULT expresión]
      [columna_constraint],
      ...
      [tabla_constraint] [, ... ]);
```

RENAME

```
RENAME tabla1 TO tabla2
```

ALTER TABLE

```
ALTER TABLE tabla
ADD         (columna tipodato [DEFAULT expr]
            [, columna tipodato] …);

ALTER TABLE tabla
MODIFY      (columna tipodato [DEFAULT expr]
            [, columna tipodato] …);

ALTER TABLE tabla
DROP        (columna);
```

DROP TABLE

```
DROP TABLE tabla ;
```

• LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Sentencia SELECT Básica

```
SELECT * | {[DISTINCT] columna | expresión [alias], ...}
FROM tabla;
```

Sentencia SELECT con Restricción

```
SELECT * | {[DISTINCT] columna | expresión [alias], ...}
FROM tabla
[WHERE condición(es)];
```

Cláusula ORDER BY

```
SELECT * | {[DISTINCT] columna | expresión [alias], ...}
FROM tabla
[WHERE condición(es)]
[ORDER BY {columna, expresión} [ASC | DESC]];
```

UNION de IGUALDAD

```
SELECT tabla1.columna, tabla2.columna
FROM tabla1, tabla2
WHERE tabla1.columna = tabla2.columna;
```

UNION de NO IGUALDAD

```
SELECT tabla1.columna, tabla2.columna
FROM tabla1, tabla2
WHERE tabla1.columna OPERADOR tabla2.columna;
```

UNION EXTERNA

```
SELECT tabla1.columna, tabla2.columna
FROM tabla1, tabla2
WHERE tabla1.columna(+) = tabla2.columna;

SELECT tabla1.columna, tabla2.columna
FROM tabla1, tabla2
WHERE tabla1.columna = tabla2.columna(+);
```

FUNCIONES DE GRUPO

```
SELECT [columna,] función de grupo(columna), ...
FROM tabla
[WHERE condición]
[GROUP BY columna]
[ORDER BY columna];
```

Restricción de RESULTADOS de GRUPO

```
SELECT [columna,] función de grupo(columna), ...
FROM tabla
[WHERE condición]
[GROUP BY columna]
[HAVING condición_grupo]
[ORDER BY columna];
```

FUNCIONES DE GRUPO

AVG → promedio
COUNT → número de filas
MAX → Valor máximo
MIN → Valor mínimo
STDDEV → Desviación estándar
SUM → Suma
VARIANCE → Varianza
Todas ignoran los valores nulos.

SINTAXIS SQL 1999 para UNIONES

```
SELECT tabla1.columna, tabla2.columna
FROM tabla1
[CROSS JOIN tabla2]
[NATURAL JOIN tabl2]
[JOIN tabla2 USING (nombre_columna)]
[JOIN tabla2
    ON (tabla1.nombre_columna = tabla2.nombre_columna)] |
[LEFT | RIGHT | FULL OUTER JOIN tabla2
    ON (tabla1.nombre_columna = tabla2.nombre_columna)]
```

INSERT

```
INSERT INTO tabla [(columna1 [, columna2, …])]
       VALUES (valor1 [, valor2, …]) ;
```

UPDATE

```
UPDATE tabla
     SET columna1 = valor1 [, columna2 = valor2,...]
     [WHERE condición(es)] ;
```

DELETE

```
DELETE [FROM] tabla
      [WHERE condición(es)];
```

PRÁCTICA SQL

- **ESTRUCTURA Y DATOS DE LAS TABLAS UTILIZADAS**

Tabla **JOBS**

```
DESCRIBE jobs
```

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

```
SELECT * FROM jobs;
```

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000

12 rows selected.

Tabla **DEPARTMENTS**

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

```
SELECT * FROM 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

8 rows selected.

Tabla **EMPLOYEES**

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

```
SELECT * FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800		100	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500		124	50
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	3100		124	50
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600		124	50
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500		124	50
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	10500	.2	100	80
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	11000	.3	149	80
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	8600	.2	149	80
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	.15	149	
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400		101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000		100	20
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000		201	20
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000		101	110
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300		205	110

20 rows selected.

Tabla JOB_GRADES

```
DESCRIBE job_grades
```

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

```
SELECT * FROM job_grades;
```

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

6 rows selected.

Tabla LOCATIONS

```
DESCRIBE locations
```

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

```
SELECT * FROM locations;
```

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

Tabla COUNTRIES

```
DESCRIBE countries
```

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

```
SELECT * FROM countries;
```

CO	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

Tabla REGIONS

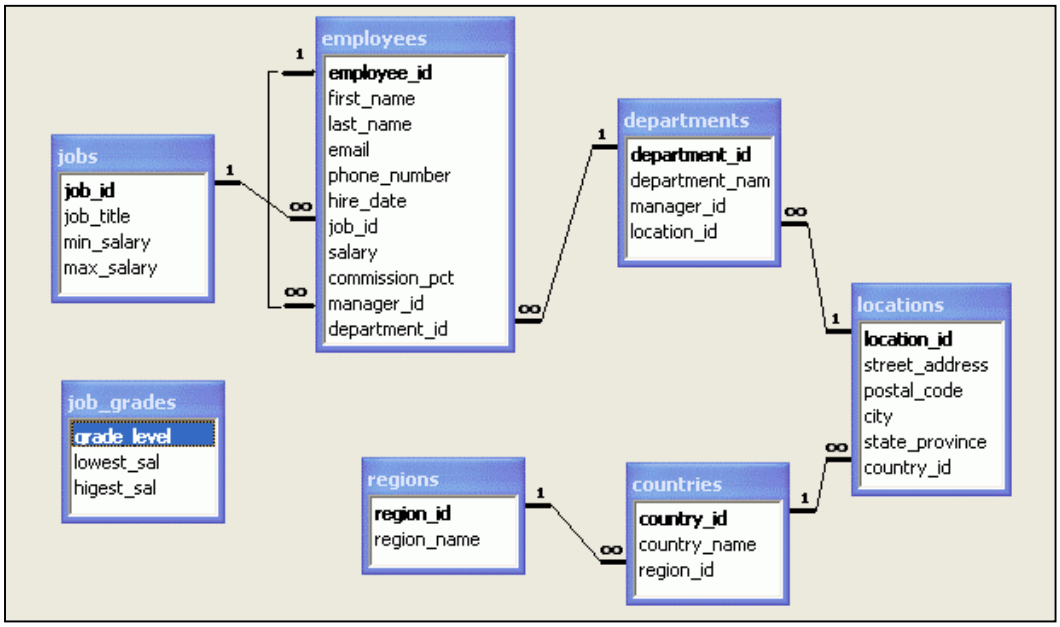
```
DESCRIBE regions
```

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

```
SELECT * FROM regions;
```

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

- **MODELO DE ENTIDAD RELACIÓN (MER)**



- **LENGUAJE DE DEFINICIÓN DE DATOS (DDL)**

CREATE TABLE→ Creación de la tabla manager.

CREATE TABLE

```
CREATE TABLE manager (
     manager_id NUMBER(6) NOT NULL PRIMARY KEY,
     descripcion CHAR(20));
```

La tabla tiene como clave primaria manager_id que es de tipo numérico y además al ser la clave primaria no puede ser nulo.

ALTER TABLE→ Modificar la tabla anterior agregando el atributo 'observaciones' y modificando el tipo de datos del atributo 'descripcion'

ALTER TABLE

```
ALTER TABLE manager ADD (observaciones VARCHAR2(100))
                    MODIFY (descripcion VARCHAR2(10));
```

La **cláusula ALTER** permite modificar la estructura de la tabla. En el ejemplo, la columna **descripcion** tenía como tipo de datos CHAR lo que se modifica mediante la **cláusula** MODIFY a VARCHAR2. Asimismo permite agregar nuevas columnas, como el caso de la columna **observaciones** mediante el uso de la **cláusula ADD.**

RENAME → Renombrar la tabla manager con el nombre gerentes

RENAME

```
RENAME manager TO gerentes;
```

DROP TABLE → Eliminar la tabla gerentes

DROP TABLE

```
DROP TABLE gerentes ;
```

- **LENGUAJE DE MANIPULACIÓN DE DATOS (DML)**

SELECCIÓN → Mostrar todos los departamentos existentes.

SELECT GENERAL

```
SELECT * FROM departments;
```

El resultado de la consulta deberá mostrar todas las tuplas de la tabla.

```
SQL> SELECT * FROM departments ;
DEPARTMENT_ID DEPARTMENT_NAME              MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
           10 Administration                      200        1700
           50 Shipping                            124        1500
           60 IT                                  103        1400
           80 Sales                               149        2500
           90 Executive                           100        1700
          110 Accounting                          205        1700
          190 Contracting                                    1700
           20 Marketing                           201        1800
8 filas seleccionadas.
```

PROYECCIÓN → Mostrar sólo a los departamentos existentes con su correspondiente.

PROYECCIÓN

```
SELECT department_id, department_name
FROM departments;
```

El resultado de la consulta deberá mostrar todas las tuplas de la tabla pero sólo para los atributos especificados.

```
SQL> SELECT department_id, department_name
     FROM departments ;

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

8 filas seleccionadas.
```

RESTRICCIÓN → Mostrar todos los departamentos que tengan como jefe al empleado con código 100

RESTRICCIÓN

```
SELECT * FROM departments
WHERE manager_id = 100 ;
```

La cláusula WHERE hace que el resultado de la consulta muestre sólo las tuplas que coincidan con la condición.

```
SQL>  SELECT * FROM departments
      WHERE manager_id = 100;

DEPARTMENT_ID DEPARTMENT_NAME        MANAGER_ID LOCATION_ID
------------- ---------------------- ---------- -----------
           90 Executive                     100        1700

1 fila seleccionada.
```

PROYECCIÓN y RESTRICCIÓN → Mostrar el ID y nombre de dapartamento con su ID de localidad para aquellos ubicados en la localidad con ID = 1700

PROYECCIÓN y RESTRICCIÓN

```
SELECT department_id, department_name,
location_id
FROM departments
WHERE location_id=1700;
```

La consulta devolverá las columnas **(proyección)** indicadas en la cláusula SELECT sólo para aquellos departamentos **(restricción)** ubicados en la localidad con código =1700.

```
SQL>  SELECT department_id, department_name, location_id
      FROM departments
      WHERE location_id = 1700;

DEPARTMENT_ID DEPARTMENT_NAME     MANAGER_ID LOCATION_ID
------------- ------------------- ---------- -----------
           10 Administration                        1700
           90 Executive                             1700
          110 Accounting                            1700
          190 Centracting                           1700
```

USO DE OR y AND →

- **Primera Consulta:** Mostrar todos los departamentos que tengan como jefe al empleado con código 100 o cuyo código sea mayor a 200.
- **Segunda Consulta:** Mostrar todos los departamentos que tengan como jefe al empleado con código mayor a 200 y estén ubicados en una localidad con código menor a 1750.

USO DE OR y AND

```
SELECT * FROM departments
WHERE manager_id = 100 OR manager_id > 200;

SELECT * FROM departments
WHERE manager_ID >200 AND location_id < 1750 ;
```

```
SQL>  SELECT *
      FROM departments
      WHERE manager_id = 100
      OR manager_id > 200;

DEPARTMENT_ID DEPARTMENT_NAME        MANAGER_ID LOCATION_ID
------------- ---------------------- ---------- -----------
           90 Executive                     100        1700
          110 Accounting                    205        1700
           20 Marketing                     201        1800
```

```
SQL>  SELECT *
      FROM departments
      WHERE manager_ID >200
      AND location_id < 1750 ;

DEPARTMENT_ID DEPARTMENT_NAME           MANAGER_ID LOCATION_ID
------------- ------------------------- ---------- -----------
          110 Accounting                       205        1700
```

USO DE ORDER BY → Mostrar los empleados que hayan sido dados de alta con una fecha anterior al 01/01/94 ordenados por la fecha de ingreso.

USO DE ORDER BY

```
SELECT last_name, first_name,
department_id, hire_date
FROM employees
WHERE hire_date <'01/01/94'
ORDER BY hire_date;
```

La consulta devuelve el apellido, el nombre, el código de departamento y la fecha de ingreso de aquellos empleados cuya fecha de ingreso es < al 01 de Enero de 1994.

El resultado se presenta ordenado por fecha de ingreso. Como no se especifica, la ordenación es **ascendente**, que es el modo por defecto.

```
SQL>  SELECT last_name, first_name, department_id, hire_date
      FROM employees
      WHERE hire_date <'01/01/94'
      ORDER BY hire_date;

LAST_NAME            FIRST_NAME      DEPARTMENT_ID HIRE_DATE
-------------------- --------------- ------------- ---------
King                 Steven                     90 17/06/87
Whalen               Jennifer                   10 17/09/87
Kochhar              Neena                      90 21/09/89
Hunold               Alexander                  90 03/01/90
Ernst                Bruce                      90 21/05/91
De Haan              Lex                        90 13/01/93

6 filas seleccionadas.
```

.

USO DE OPERADORES ARITMÉTICOS → Mostrar el sueldo de todos los empleados y registrar un aumento de $300. Ordenarlos según el sueldo que ganan.

USO DE OPERADORES ARITMÉTICOS

```
SELECT last_name, salary,
salary + 300 as NUEVO_SALARIO
FROM employees ORDER BY salary ;
```

La consulta devuelve una nueva columna (que en realidad no existe en ninguna tabla) que muestra el cálculo del nuevo sueldo.

Se utiliza **alias de columna** para brindar mayor claridad. El resultado está ordenado por el salario original en forma ascendente.

```
SQL>  SELECT last_name, salary,
      salary + 300 AS NUEVO_SALARIO
      FROM employees
      ORDER BY salary;
```

LAST_NAME	SALARY	NUEVO_SALARIO
Vargas	2500	2800
Matos	2600	2900
Davies	3100	3400
Rajs	3500	3800
Lorentz	4200	4500
Whalen	4400	4700
Mourgos	5800	6100
Ernst	6000	6300
Fay	6000	6300
Grant	7000	7300
Gietz	8300	8600
Taylor	8600	8900
Hunold	9000	9300
Zlotkey	10500	10800
Abel	11000	11300
Higgins	12000	12300
Hartstein	13000	13300
Kochhar	17000	17300
De Haan	17000	17300
King	24000	24300

20 filas seleccionadas.

USO DE OPERADORES ARITMÉTICOS (prioridad y paréntesis) →

- **Primera Consulta:** Calcular el aguinaldo de todos los empleados teniendo en cuenta el aumento de $300. Ordenarlos según el sueldo que ganan.
- **Segunda Consulta:** Lo mismo pero usando paréntesis.

USO DE OPERADORES ARITMÉTICOS: prioridad y paréntesis

```
SELECT last_name, salary,salary + 300 / 2 as AGUINALDO
FROM employees
ORDER BY salary ;

SELECT last_name, salary + 300 NUEVO_SALARIO,
(salary + 300) / 2 as AGUINALDO
FROM employees
ORDER BY salary;
```

- **Primera Consulta:** Devuelve una nueva columna renombrada como AGUINALDO. Como la división y la multiplicación tienen **precedencia** sobre la suma y la resta, al efectuar el cálculo primero dividió 300/2 y luego lo sumó al salario. Para evitar estos problemas se deben utilizar paréntesis.

```
SQL>SELECT last_name, salary + 300 NUEVO_SALARI
          (salary + 300) / 2 AS AGUINALDO
          FROM employees ORDER BY salary;
```

LAST_NAME	NUEVO_SALARIO	AGUINALDO
Vargas	2800	1400
Matos	2900	1450
Davies	3400	1700
Rajs	3800	1900
Lorentz	4500	2250
Whalen	4700	2350
Mourgos	6100	3050
Ernst	6300	3150
Fay	6300	3150
Grant	7300	3650
Gietz	8600	4300
Taylor	8900	4450
Hunold	9300	4650
Zlotkey	10800	5400
Abel	11300	5650
Higgins	12300	6150
Hartstein	13300	6650
Kochhar	17300	8650
De Haan	17300	8650
King	24300	12150

20 filas seleccionadas.

- **Segunda Consulta:** Se utiliza paréntesis para indicar la precedencia. El cálculo por lo tanto es correcto: al (salario + 300)/2 que es el aguinaldo.

```
SQL>  SELECT last_name, salary,
      salary + 300 / 2 AS AGUINALDO
      FROM employees
      ORDER BY salary;
```

LAST_NAME	SALARY	AGUINALDO
Vargas	2500	2650
Matos	2600	2750
Davies	3100	3250
Rajs	3500	3650
Lorentz	4200	4350
Whalen	4400	4550
Mourgos	5800	5950
Ernst	6000	6150
Fay	6000	6150
Grant	7000	7150
Gietz	8300	8450
Taylor	8600	8750
Hunold	9000	9150
Zlotkey	10500	10650
Abel	11000	11150
Higgins	12000	12150
Hartstein	13000	13150
Kochhar	17000	17150
De Haan	17000	17150
King	24000	24150

20 filas seleccionadas.

USO DE OPERADORES de CONCATENACIÓN y CARACTERES LITERALES→
Mostrar en una sola oración a cada empleado con su salario.

USO DE OPERADORES DE CONCATENACIÓN Y CARACTERES LITERALES

```
SELECT last_name ||', '|| first_name
|| 'tiene un salario de: ' || salary
FROM employees ;
```

Se utiliza el **operador de concatenación ||** lo que da como resultado de la consulta una única columna.

Nota: en Oracle, para la concatenación, no es necesario utilizar ninguna función de conversión con el campo 'salary' que es numérico, con respecto a los otros campos; en DB2 sí debe utilizarse la función **cast**

```
SQL> SELECT last_name ||', '||first_name ||
          ' tiene un salario de: ' || salary
          FROM employees;

LAST_NAME||','||FIRST_NAME||'TIENEUNSALARIODE:'||SALARY
--------------------------------------------------------
King, Steven tiene un salario de: 24000
Kochhar, Neena tiene un salario de: 17000
De Haan, Lex tiene un salario de: 17000
Hunold, Alexander tiene un salario de: 9000
Ernst, Bruce tiene un salario de: 6000
Lorentz, Diana tiene un salario de: 4200
Mourgos, Kevin tiene un salario de: 5800
Rajs, Trenna tiene un salario de: 3500
Davies, Curtis tiene un salario de: 3100
Matos, Randall tiene un salario de: 2600
Vargas, Peter tiene un salario de: 2500
Zlotkey, Eleni tiene un salario de: 10500
Abel, Ellen tiene un salario de: 11000
Taylor, Jonathon tiene un salario de: 8600
Grant, Kimberely tiene un salario de: 7000
Whalen, Jennifer tiene un salario de: 4400
Hartstein, Michael tiene un salario de: 13000
Fay, Pat tiene un salario de: 6000
Higgins, Shelley tiene un salario de: 12000
Gietz, William tiene un salario de: 8300
20 filas seleccionadas.
```

USO DE OPERADORES de CONCATENACIÓN y CARACTERES LITERALES utilizando ALIAS de COLUMNA→ Mostrar en una sola oración a cada empleado con su salario, renombrando a la columna como "Datos de los Empleados"

USO DE OP. DE CONCATENACIÓN Y CARACTERES LITERALES UTILIZANDO ALIAS de COLUMNA

```
SELECT last_name ||', '|| first_name
|| 'tiene un salario de: ' || salary  "Datos de los Empleados"
FROM employees;
```

```
SQL>  SELECT last_name ||', '|| first_name || ' tiene un salario de:
            salary  "Datos de los Empleados"
            FROM employees;

Datos de los Empleados
----------------------------------------
King, Steven tiene un salario de: 24000
Kochhar, Neena tiene un salario de: 17000
De Haan, Lex tiene un salario de: 17000
Hunold, Alexander tiene un salario de: 9000
Ernst, Bruce tiene un salario de: 6000
Lorentz, Diana tiene un salario de: 4200
Mourgos, Kevin tiene un salario de: 5800
Rajs, Trenna tiene un salario de: 3500
Davies, Curtis tiene un salario de: 3100
Matos, Randall tiene un salario de: 2600
Vargas, Peter tiene un salario de: 2500
Zlotkey, Eleni tiene un salario de: 10500
Abel, Ellen tiene un salario de: 11000
Taylor, Jonathon tiene un salario de: 8600
Grant, Kimberely tiene un salario de: 7000
Whalen, Jennifer tiene un salario de: 4400
Hartstein, Michael tiene un salario de: 13000
Fay, Pat tiene un salario de: 6000
Higgins, Shelley tiene un salario de: 12000
Gietz, William tiene un salario de: 8300

20 filas seleccionadas.
```

UNIÓN DE IGUALDAD ➔ Mostrar a todos los empleados ordenados alfabéticamente, determinando el departamento en donde trabajan y la ciudad en que se ubica.

UNIÓN DE IGUALDAD

```
SELECT employees.last_name, employees.first_name,
departments.department_name, locations.city
FROM employees, departments, locations
WHERE employees.department_id = departments.department_id
AND departments.location_id = locations.location_id
ORDER BY employees.last_name ;
```

La consulta consiste en la **unión de tres tablas** (unión de igualdad). La unión de las tablas se efectúa **igualando las claves primarias y las foráreas** de las tablas que están relacionadas. Así **department_id** es la clave primaria de la tabla departamentos y una clave foránea de la tabla empleados; **location_id** es la clave primaria en la tabla localidades y una clave foránea de la tabla departamentos.

El mismo ejemplo anterior pero utilizando **alias de tabla.** El alias de tabla se especifica en la **cláusula FROM.** De esta forma se da mayor claridad, se reducen los errores de tipeo y se logra una mayor performance.

UNIÓN DE IGUALDAD UTILIZANDO ALIAS DE TABLA

```
SELECT e.last_name, e.first_name,
d.department_name, l.city
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
AND d.location_id = l.location_id
ORDER BY e.last_name;
```

```
SQL> SELECT employees.last_name, employees.first_name,
     departments.department_name, locations.city
     FROM employees, departments, locations
     WHERE employees.department_id = departments.department_id
     AND departments.location_id = locations.location_id
     ORDER BY employees.last_name ;

LAST_NAME       FIRST_NAME      DEPARTMENT_NAME     CITY
--------------- --------------- ------------------- ------------------
Abel            Ellen           Sales               Oxford
Davies          Curtis          Shipping            South San Francisco
De Haan         Lex             Executive           Seatle
Ernst           Bruce           Executive           Seatle
Fay             Pat             Marketing           Toronto
Gietz           William         Accounting          Seatle
Hartstein       Michael         Marketing           Toronto
Higgins         Shelley         Accounting          Seatle
Hunold          Alexander       Executive           Seatle
King            Steven          Executive           Seatle
Kochhar         Neena           Executive           Seatle
Lorentz         Diana           IT                  Southlake
Matos           Randall         Shipping            South San Francisco
Mourgos         Kevin           Shipping            South San Francisco
Rajs            Trenna          Shipping            South San Francisco
Taylor          Jonathon        Sales               Oxford
Vargas          Peter           Shipping            South San Francisco
Whalen          Jennifer        Administration      Seatle
Zlotkey         Eleni           Sales               Oxford
19 filas seleccionadas.
```

UNIÓN DE NO IGUALDAD → Mostrar el nombre de departamento y la ciudad en la que está ubicado para todos aquellas ciudades que no estén ubicadas en Estados Unidos.

UNIÓN DE NO IGUALDAD

```
SELECT l.city, d.department_name
FROM locations l, deparments d
WHERE l.location_id = d.location_id
      AND l.country_id != 'US' ;
```

En el caso de las uniones de no igualdad el operador debe ser distinto al de igualdad.

Las tablas tienen la columna **location_id** en común. El resultado sólo muestra aquellas tuplas de ambas tablas que coinciden y que además el código de país es distinto a Estados Unidos. Actúa restringiendo la UNIÓN.

```
SQL> SELECT l.city, d.department_name
     FROM locations l, departments d
     WHERE l.location_id = d.location_id
     AND l.country_id != 'US' ;

CITY                           DEPARTMENT_NAME
------------------------------ ------------------------------
Oxford                         Sales
Toronto                        Marketing
```

AUTOUNIÓN → Determinar el jefe de cada empleado.

Como se trata de una **autounión o autorelación**, se ha utilizado **alias de tabla.** Además se ha utilizado el operador de concatenación visto anteriormente y un **alias de columna (Ejemplo de Autounión)** para la única cabecera de columna que mostrará el resultado. Cada tupla mostrará el apellido y nombre de cada empleado y el apellido y nombre del jefe inmediato. El orden ha sido establecido por apellido y nombre de los empleados.

UNIÓN DE UNA TABLA CONSIGO MISMA (AUTOUNIÓN)

```
SELECT worker.last_name || ', ' || worker.first_name
|| ' trabaja para ' || manager.last_name ||
', ' || manager.first_name AS "Ejemplo de AUTOUNIÓN"
FROM employees worker, employees manager
WHERE worker.manager_id = manager.employee_id
ORDER BY worker.last_name, worker.first_name;
```

```
SQL> SELECT worker.last_name || ', ' || worker.first_name
     || ' trabaja para ' || manager.last_name ||
     ', ' || manager.first_name AS "Ejemplo de AUTOUNIÓN"
     FROM employees worker, employees manager
     WHERE worker.manager_id=manager.employee_id
     ORDER BY worker.last_name, worker.first_name;

Ejemplo de AUTOUNIÓN
--------------------------------------------------------------------
Abel, Ellen trabaja para Zlotkey, Eleni
Davies, Curtis trabaja para Mourgos, Kevin
De Haan, Lex trabaja para King, Steven
Ernst, Bruce trabaja para Hunold, Alexander
Fay, Pat trabaja para Hartstein, Michael
Gietz, William trabaja para Higgins, Shelley
Grant, Kimberely trabaja para Zlotkey, Eleni
Hartstein, Michael trabaja para King, Steven
Higgins, Shelley trabaja para Kochhar, Neena
Hunold, Alexander trabaja para De Haan, Lex
Kochhar, Neena trabaja para King, Steven
Lorentz, Diana trabaja para Hunold, Alexander
Matos, Randall trabaja para Mourgos, Kevin
Mourgos, Kevin trabaja para King, Steven
Rajs, Trenna trabaja para Mourgos, Kevin
Taylor, Jonathon trabaja para Zlotkey, Eleni
Vargas, Peter trabaja para Mourgos, Kevin
Whalen, Jennifer trabaja para Kochhar, Neena
Zlotkey, Eleni trabaja para King, Steven

19 filas seleccionadas.
```

UNIÓN EXTERNA → Mostrar todos los países y ciudades.

En el caso de las **uniones externas**, una de las tablas contiene información insuficiente para realizar la unión. El o perador que se utiliza es (+) y se coloca al lado de la tabla o relación que tiene información insuficiente.

UNIÓN EXTERNA

```
SELECT c.country_name, l.city
FROM countries c, locations l
WHERE c.country_id = l.country_id (+);
```

```
SQL> SELECT c.country_name, l.city
     FROM countries c, locations l
     WHERE c.country_id = l.country_id (+) ;

COUNTRY_NAME                     CITY
-------------------------------- ------------------------
Canada                           Toronto
Germany
United Kingdom                   Oxford
United States of America         Southlake
United States of America         South San Francisco
United States of America         Seatle

6 filas seleccionadas.
```

Nota: en la ejecución de la sentencia se ha utilizado el DBMS Oracle. Puede conseguirse el mismo resultado utilizando SQL 1999 con RIGHT/LEFT OUTER JOIN , denpendiendo de la estructura de la sentencia. En DB2 debe utilizarse SQL 1999.

FUNCIONES DE GRUPO → Calcular el promedio de los sueldos de todos los empleados, sueldo mas elevado y el menor y la suma de todos.

FUNCIONES DE GRUPO

```
SELECT AVG(salary), MAX(salary),
MIN(salary), SUM(salary)
FROM employees ;
```

```
SQL> SELECT AVG(salary), MAX(salary),
     MIN(salary), SUM(salary)
     FROM employees;

AVG(SALARY) MAX(SALARY) MIN(SALARY) SUM(SALARY)
----------- ----------- ----------- -----------
       8775       24000        2500      175500
```

FUNCIONES DE GRUPO – Cláusula GROUP BY → Calcular el promedio de los sueldos de todos los empleados, sueldo mas elevado y el menor y la suma de todos.

FUNCIONES DE GRUPO CON GROUP BY

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

La **cláusula GROUP BY** agrupa la consulta por código de departamento. Por lo tanto las funciones de grupo AVG, SUM, MAX y MIN **operan sobre cada grupo.**

```
SQL> SELECT department_id, AVG(salary),
            MAX(salary), MIN(salary), SUM(salary)
     FROM employees
     GROUP BY department_id;

DEPARTMENT_ID AVG(SALARY) MAX(SALARY) MIN(SALARY) SUM(SALARY)
------------- ----------- ----------- ----------- -----------
           10        4400        4400        4400        4400
           20        9500       13000        6000       19000
           50        3500        5800        2500       17500
           60        4200        4200        4200        4200
           80  10033,3333       11000        8600       30100
           90       14600       24000        6000       73000
          110       10150       12000        8300       20300
                     7000        7000        7000        7000
8 filas seleccionadas.
```

FUNCIONES DE GRUPO – Función COUNT→ Determinar la cantidad de empleados por departamento y el salario total por departamento.

FUNCION DE GRUPO COUNT

```
SELECT department_id, COUNT(*) CTOSEMP,
SUM(salary) TOTSAL
FROM employees
GROUP BY department_id;
```

```
SQL>  SELECT department_id, COUNT(*) CTOSEMP,
      SUM(salary) TOTSAL
      FROM employees
      GROUP BY department_id;

DEPARTMENT_ID    CTOSEMP     TOTSAL
------------- ---------- ----------
           10          1       4400
           20          2      19000
           50          5      17500
           60          1       4200
           80          3      30100
           90          5      73000
          110          2      20300
                       1       7000
8 filas seleccionadas.
```

FUNCIONES DE GRUPO – Cláusula HAVING→ Calcular el promedio de los sueldos según el departamento, mostrando al sueldo más alto y el menor junto con la suma de todos los sueldos de cada departamento. Teniendo en cuenta solo a los departamentos con código menor a 60.

FUNCIONES DE GRUPO CON HAVING
(restricción de grupo)

```
SELECT department_id, AVG(salary),
MAX(salary), MIN(salary), SUM(salary)
FROM employees
GROUP BY department_id
HAVING department_id <60 ;
```

La **cláusula HAVING** sirve para **restringir** los resultados de grupo.
En el ejemplo, el resultado mostrará los cálculos para aquellos departamentos cuyo código es < 60.

```
SQL> SELECT department_id, AVG(salary),
           MAX(salary), MIN(salary), SUM(salary)
     FROM employees
     GROUP BY department_id
     HAVING department_id < 60;

DEPARTMENT_ID AVG(SALARY) MAX(SALARY) MIN(SALARY) SUM(SALARY)
------------- ----------- ----------- ----------- -----------
           10        4400        4400        4400        4400
           20        9500       13000        6000       19000
           50        3500        5800        2500       17500
```

FUNCIONES DE GRUPO – Cláusulas WHERE y HAVING→ Calcular la suma de sueldos mayores a 5000 según el departamento, y sólo para los departamentos con código menor a 60.

FUNCIONES DE GRUPO CON WHERE y HAVING
(restricción consulta y de grupo)

```
SELECT department_id, SUM(salary)
FROM employees
WHERE salary > 5000
GROUP BY department_id
HAVING department_id < 60 ;
```

La **cláusula WHERE** en este caso actúa sobre la consulta; es decir sólo se considerarán los salarios > 5000.
El resultado mostrará la suma de los salarios >5000 agrupados por departamento y sólo se mostrarán aquellos cuyo código de departamento es < 60.

```
SQL>  SELECT department_id, SUM(salary)
      FROM employees
      WHERE salary > 5000
      GROUP BY department_id
      HAVING department_id < 60 ;

DEPARTMENT_ID SUM(SALARY)
------------- -----------
           20       19000
           50        5800
```

INSERT → Insertar el departamento SECURITY en la tabla departments, con los siguientes datos: department_id=180, manager_id=100 y location_id=1700

INSERT

```
INSERT INTO departments (department_id, department_name,
            manager_id, location_id)
            VALUES (180,'SECURITY',100,1700) ;
```

UPDATE → Modificar el nombre del departamento SECURITY a Security

UPDATE

```
UPDATE departments
     SET department_name = 'Security'
WHERE department_id = 180;
```

En este caso debe especificarse mediante la **cláusula WHERE** la fila o tupla a modificar, para lo cual se utiliza la clave primaria de la tabla

DELETE → Eliminar el departamento Security de la tabla departments

DELETE

```
DELETE FROM departments
WHERE department_id = 180 ;
```

Como en el caso anterior, debe especificarse en la **cláusula WHERE** el departamento que va a eliminarse.