Introducción a SQL

Introducción al lenguaje SQL

Contenidos

- Introducción al lenguaje SQL
- Sentencia SELECT: lista SELECT, FROM, WHERE
- CREATE TABLE
- INSERT, DELETE y UPDATE

Bibliografía

- Beaulieu, A.: Aprende SQL. Anaya-O'Reilly. 2006.
- Rivero, E. y otros: Introducción al SQL para usuarios y programadores. Thomson-Paraninfo. 2002.

Conceptos previos

Definición de Base de Datos (BD)

Colección de datos relacionados, entendiendo que un dato es un hecho conocido que se puede registrar y que tiene un significado explícito.

Sistema de Gestión de Bases de Datos (SGBD)

Sistema software de propósito general que facilita los procesos de definición, construcción y manipulación de bases de datos para distintas aplicaciones.

Conceptos previos

Los datos de una BD se almacenan en tablas.

Ejemplo: Tabla “Clientes”

DNI/NIF	Nombre	Dirección	Teléfono
11.222.333-A	Francisco Pérez	C/. Mayor, 1	600 200300
22.111.333-B	Antonio García	C/. Mayor, 2	600 400500
33.222.111-C	José González	C/. Paseo, 3	600 600700

Podría haber más tablas para almacenar los datos de los empleados, departamentos, proyectos, pedidos, facturas, etc.

Lenguaje SQL (I)

SQL (Structured Query Language)

- Lenguaje para acceder a los datos de un SGBD relacional
  - Crear, consultar, y modificar datos
- Lenguaje estándar (ISO/IEC)
  - Todos los SGBD relacionales utilizan este lenguaje
  - Normalmente con pequeñas variaciones
- Evolución
  - SEQUEL (*Structured English Query Language*). IBM System R
  - SQL-86 o SQL1. ANSI e ISO. Revisado en 1989
  - SQL-92, SQL:1999, SQL:2003, ...

Lenguaje SQL (II)

SQL (Structured Query Language)

- Data Manipulation Language (DML)
  - SELECT: consultas sobre los datos
  - INSERT: inserción de datos
  - DELETE: borrados
  - UPDATE: actualizaciones
- Data Definition Language (DDL)
  - CREATE TABLE: creación de tablas
  - DROP TABLE: eliminación de tablas
  - …

Lenguaje SQL (y III)

Otros aspectos

- Posibilidad de incluir SQL en lenguajes como C, Java, Pascal, etc.
- Lenguaje declarativo, no procedural
- Aspectos de control (seguridad, transacciones, concurrencia, etc.)
- Lenguaje intuitivo para todo tipo de usuarios

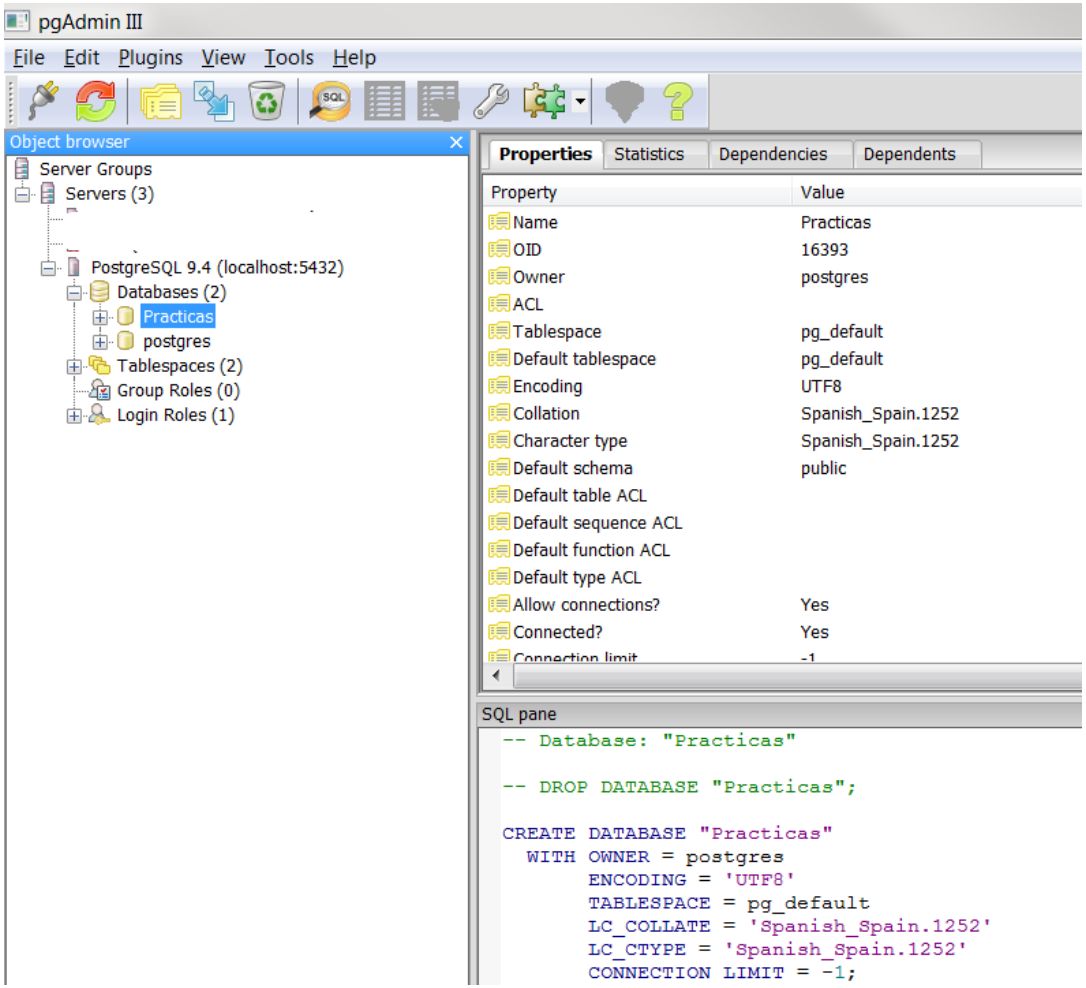
Entorno de prácticas

PostgreSQL

- SGBD free & open source (www.postgresql.com)
- Ya lo debéis tener instalado y con la BD de *Practicas*
- Existen muchos otros, pero nosotros usaremos este.

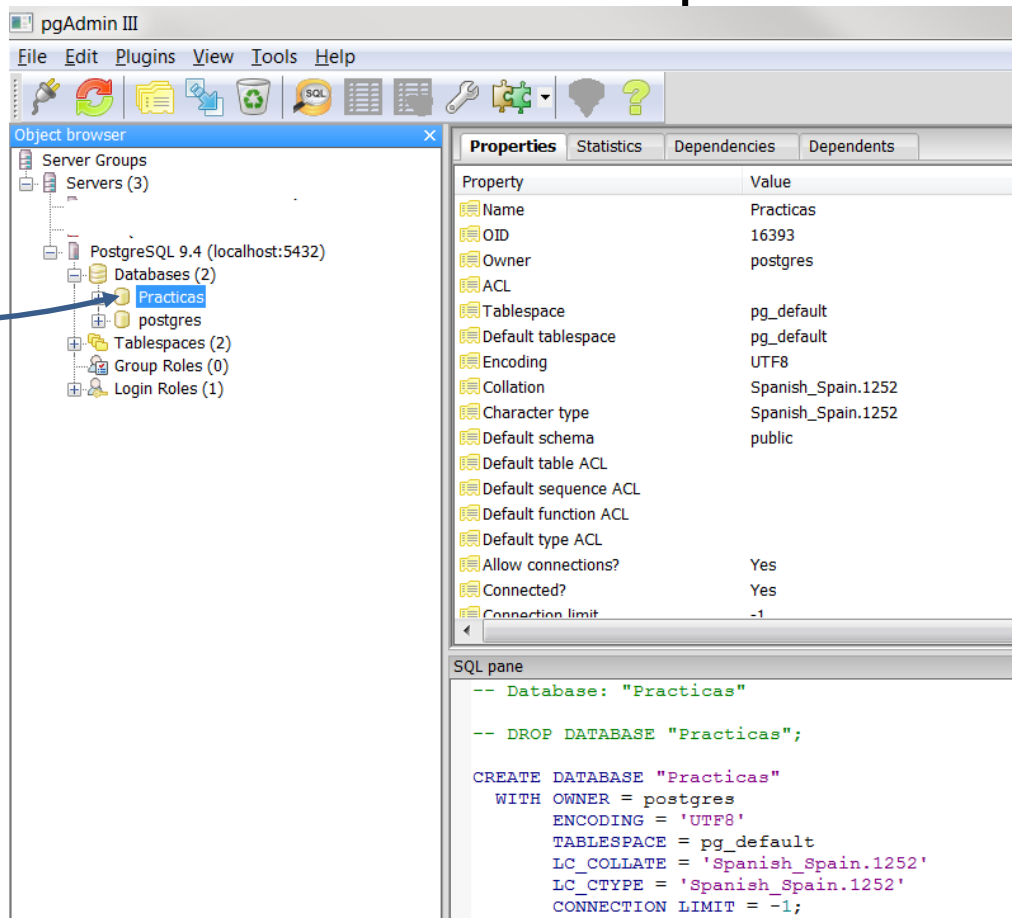
PgAdmin

Sencillo programa gráfico para manejar Postgesql



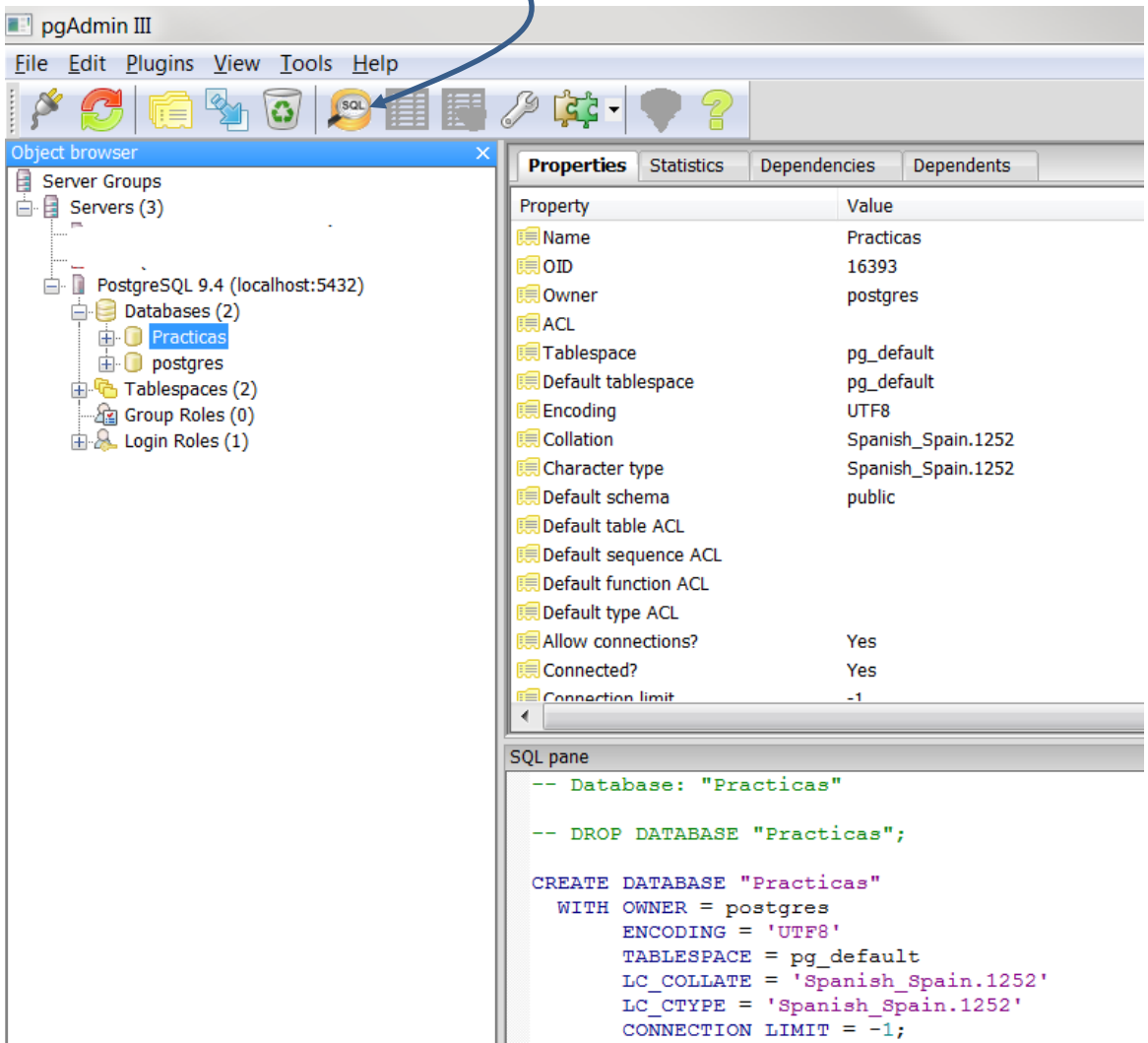
PgAdmin

- Para ejecutar consultas, primero selecciona la BD sobe la que quieres hacer consultas en la parte izquierda.



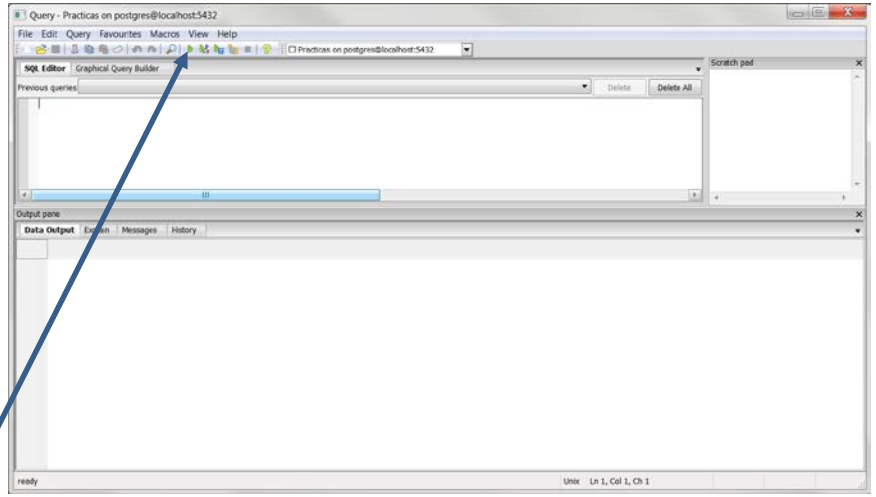
PgAdmin

- Luego, pincha en el botón SQL



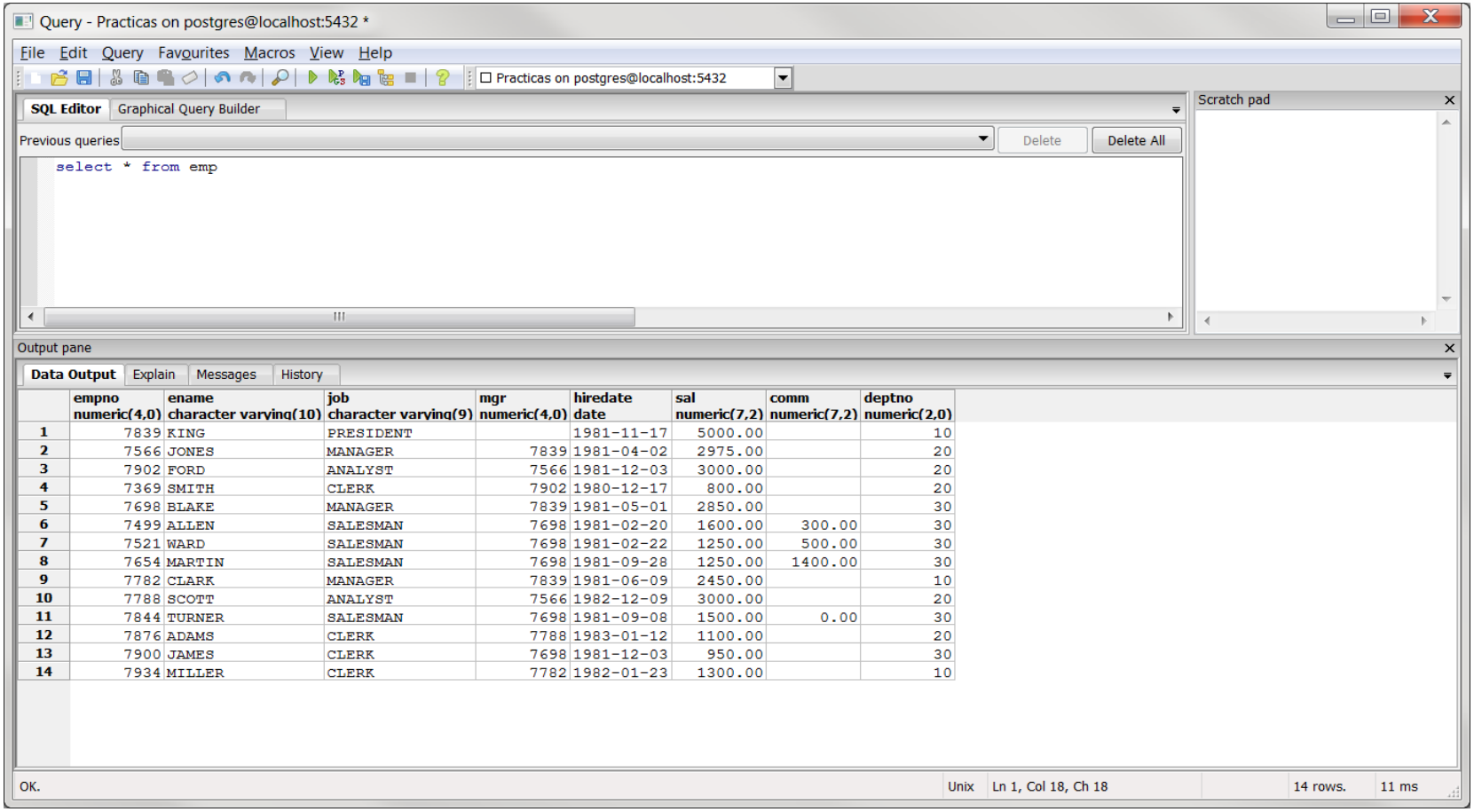
PgAdmin

- Aparece esta pantalla:



- En la parte superior puedes escribir la consulta.
- Pulsando en “excute query”, se ejecuta.
- El resultado aparece en la parte inferior.

PgAdmin


Query - Practicas on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
Practicas on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
select * from emp
Scratch pad
Output pane
Data Output
Explain
Messages
History
empno numeric(4,0)
ename character varying(10)
job character varying(9)
mgr numeric(4,0)
hiredate date
sal numeric(7,2)
comm numeric(7,2)
deptno numeric(2,0)
1 7839 KING PRESIDENT 1981-11-17 5000.00 10
2 7566 JONES MANAGER 7839 1981-04-02 2975.00 20
3 7902 FORD ANALYST 7566 1981-12-03 3000.00 20
4 7369 SMITH CLERK 7902 1980-12-17 800.00 20
5 7698 BLAKE MANAGER 7839 1981-05-01 2850.00 30
6 7499 ALLEN SALESMAN 7698 1981-02-20 1600.00 300.00 30
7 7521 WARD SALESMAN 7698 1981-02-22 1250.00 500.00 30
8 7654 MARTIN SALESMAN 7698 1981-09-28 1250.00 1400.00 30
9 7782 CLARK MANAGER 7839 1981-06-09 2450.00 10
10 7788 SCOTT ANALYST 7566 1982-12-09 3000.00 20
11 7844 TURNER SALESMAN 7698 1981-09-08 1500.00 0.00 30
12 7876 ADAMS CLERK 7788 1983-01-12 1100.00 20
13 7900 JAMES CLERK 7698 1981-12-03 950.00 30
14 7934 MILLER CLERK 7782 1982-01-23 1300.00 10
OK.
Unix
Ln 1, Col 18, Ch 18
14 rows.
11 ms

Tablas de los ejemplos

Trabajamos sobre dos tablas de ejemplo:

- Empleados (tabla “emp”)
  - Contiene datos de los empleados de una empresa
  - Código de empleado, nombre, puesto de trabajo, código del jefe, fecha de contratación, salario, comisión y código de departamento
- Departamentos (tabla “dept”)
  - Contiene datos de los departamentos de una empresa
  - Código del departamento, nombre y localización

Tablas de los ejemplos

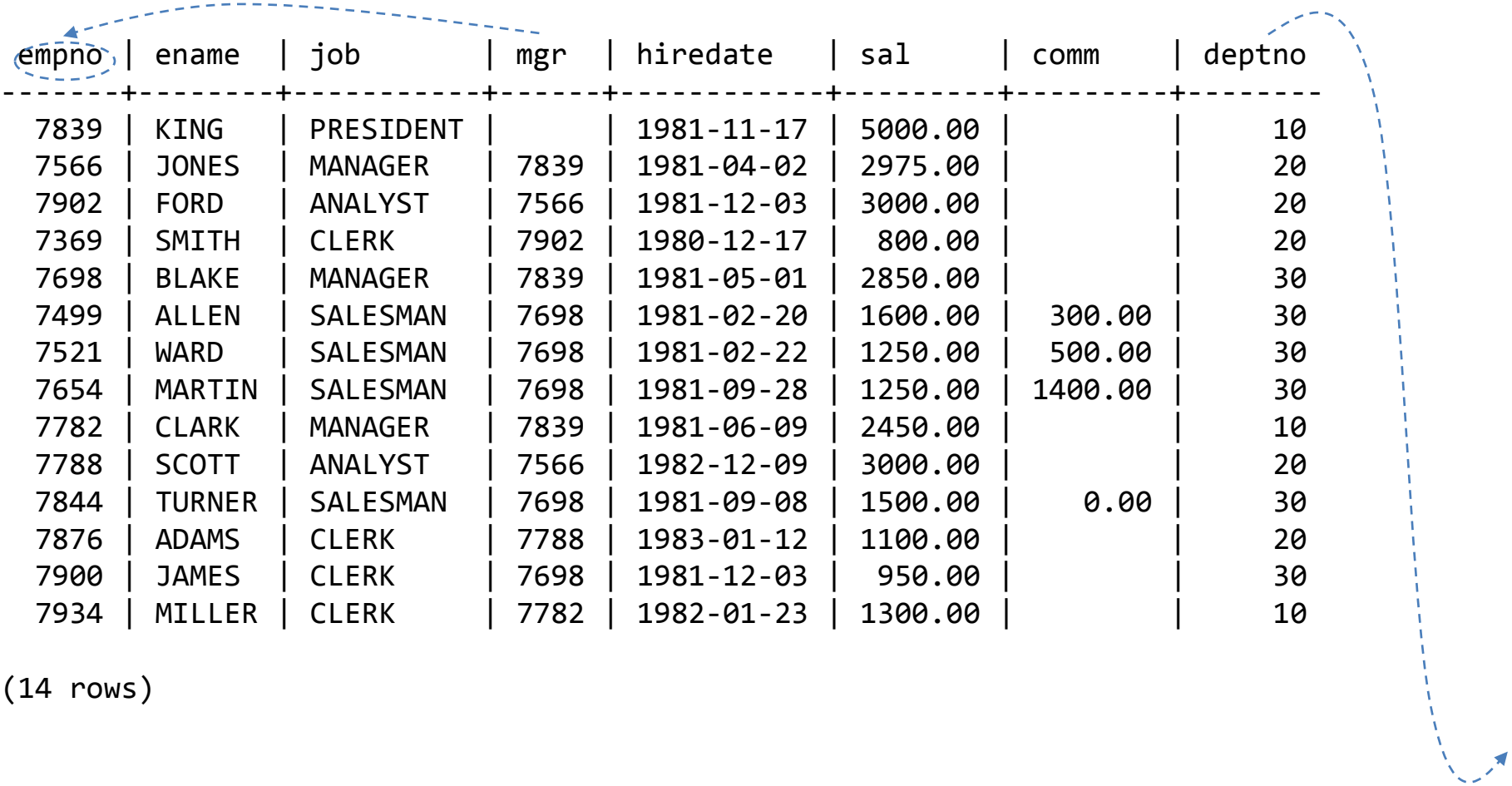
```
Table “public.emp”

   Column | Type                   | Modifiers
----------+------------------------+-----------
    empno | numeric(4,0)           | not null
    ename | character varying(10)  |
      job | character varying(9)   |
      mgr | numeric(4,0)           |
 hiredate | date                   |
      sal | numeric(7,2)           |
     comm | numeric(7,2)           |
   deptno | numeric(2,0)           |

Indexes:
    "emp_pkey" PRIMARY KEY, btree (empno)
Foreign-key constraints:
    "emp_deptno_fkey" FOREIGN KEY (deptno) REFERENCES dept(deptno) "emp_mgr_fkey"
    FOREIGN KEY (mgr) REFERENCES emp(empno)
Referenced by:
    TABLE "emp" CONSTRAINT "emp_mgr_fkey" FOREIGN KEY (mgr) REFERENCES emp(empno)
```

Tablas de los ejemplos

```
ib=> SELECT * FROM emp;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

(14 rows)



Tablas de los ejemplos

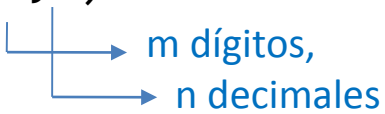
```
Table “public.dept”

 Column |                  Type | Modifiers
--------+-----------------------+-----------
 deptno | numeric(2,0)          | not null
  dname | character varying(14) |
    loc | character varying(13) |

Indexes:
    "dept_pkey" PRIMARY KEY, btree (deptno)
Referenced by: TABLE "emp" CONSTRAINT "emp_deptno_fkey" FOREIGN KEY (deptno)
    REFERENCES dept(deptno)

ib=> SELECT * FROM dept;

 deptno | dname      | loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON

(4 rows)
```

Tipos de datos

Cada columna de una tabla tiene un tipo de dato asignado, que determina los valores posibles y las operaciones permitidas sobre esos valores.

- Tipos de datos numéricos

```
INTEGER o INT, SMALLINT
FLOAT, REAL, DOUBLE PRECISION
DECIMAL(m,n) o DEC(m,n) o NUMERIC(m,n)
```



- Tipos de datos alfanuméricos

```
CHAR(n) o CHARACTER(n)
VARCHAR(n)
```

- Tipos de datos temporales

```
DATE (YEAR, MONTH, DAY)
TIME (HOUR, MINUTE, SECOND)
TIMESTAMP
```

Valores nulos

Valores nulos

El valor nulo **NULL** representa la ausencia de información, o bien por desconocimiento del dato, o bien porque no procede.

Debe diferenciarse de cualquier otro valor, entre ellos el valor 0 si se trata de un dato numérico, y de la cadena de caracteres vacía, si es un tipo de dato de tipo carácter.

Una columna de una tabla podrá admitir valores nulos (NULL) o no (NOT NULL). Por defecto admite nulos.

```
CREATE TABLE emp (
    empno    numeric(4,0) NOT NULL,
    ename    character varying(10) NOT NULL,
    job      character varying(9),
    mgr      numeric(4,0),
    hiredate date,
    sal      numeric(7,2),
    comm     numeric(7,2),
    deptno   numeric(2,0)
)
```

Expresiones

Una expresión es la formulación de una secuencia de operaciones, o sea, una combinación de operadores, operandos y paréntesis, que, cuando se ejecuta, devuelve un único valor como resultado.

Los operandos pueden ser constantes, nombres de columna, funciones, otras expresiones y otros elementos.

El tipo de dato de cada operando de una expresión debe ser el mismo. Si un operando es nulo, el resultado también es nulo.

Operadores numéricos: + - * /

Operadores alfanuméricos: ||

Ejemplos:

3

'Casa'

'A' || 'B'

ENAME

SAL * 1.5

SAL + COMM

SELECT

La sentencia SELECT permite seleccionar u obtener datos de una o varias tablas.

Sintaxis completa (no veremos GROUP BY y HAVING):

```
SELECT [DISTINCT | ALL] {*|<expr1>,<expr2>],…}
FROM <tabla1>, <tabla2>
WHERE <condición_where>
GROUP BY <group_by_expr1>[,<group_by_expr2>,…]
HAVING <condición_having>
ORDER BY <expr_orderby1>[,<expr_orderby2>,…]
```

SELECT

El **orden de ejecución** de las cláusulas y la **función** de cada una es:

1. **FROM** (obligatoria)

   Determina la tabla o tablas de las que se seleccionarán los datos.

2. **WHERE** (opcional)

   Indica un predicado que expresa la condición que debe cumplir cada fila que interviene en la consulta. Así la consulta se restringe a las filas que cumplen la condición.

3. **SELECT** (obligatoria)

   Incluye los datos que se solicitan en la consulta, normalmente una o varias expresiones. Alternativamente un ***** indica todas las columnas de las tablas involucradas. Si hubiese filas repetidas, por defecto aparecen, pero no lo hacen si se incluye **DISTINCT**.

4. **ORDER BY** (opcional)

   Permite determinar el criterio de ordenación de las filas de la tabla resultado. Sin ella obtendremos las mismas filas, pero puede que en órdenes distintos, según la estrategia seguida por el SGBD para extraer los datos.

   ASC : orden ascendente (por defecto)
   DESC : orden descendente

SELECT

Primeros ejemplos

1. Obtener todos los datos de la tabla de empleados:

```
SELECT *
FROM emp;
```

2. Obtener los códigos y nombres de los empleados:

```
SELECT empno, ename
FROM emp;
```

3. Obtener todos los datos de la tabla de empleados, ordenando el resultado por el nombre de los empleados:

```
SELECT *
FROM emp
ORDER BY ename;
```

```
SELECT *
FROM emp
ORDER BY ename ASC;
```

Filas repetidas

Las filas repetidas se pueden eliminar del resultado con **DISTINCT**:

1. Obtener todos los puestos de trabajo.

```
SELECT job
FROM emp;
```

2. Obtener todos los puestos de trabajo, sin filas repetidas.

```
SELECT DISTINCT job
FROM emp;
```

   En este caso las filas repetidas se eliminan del resultado final.

3. Obtener los puestos de trabajo y comisiones, sin filas repetidas.

```
SELECT DISTINCT job, comm
FROM emp;
```

   DISTINCT **se aplica al conjunto** de todas las expresiones de SELECT (en este caso a `job` y `comm`, no sólo a `job`)

Predicados

Predicados elementales

Un predicado expresa una condición y como en BD relacionales existe una **lógica de tres valores**, verdadero, falso y nulo, la evaluación de una condición puede ser **TRUE**, **FALSE** o **NULL**.

Un predicado puede aparecer en una cláusula WHERE evaluando la condición de cada fila de las tablas involucradas, de forma que sólo las filas que cumplen la condición permanecen involucradas en la consulta, ignorando las restantes.

Los predicados más elementales son los de comparación, que comparan dos expresiones según un operador de comparación, que puede ser:

`<` `<=` `=` `!=` `<>` `>=` `>`

Algunos ejemplos…

Predicados

1. Obtener los códigos y nombres de los empleados que trabajan en el departamento 10:

```
SELECT empno, ename
FROM   emp
WHERE  deptno = 10;
```

2. Seleccionar todos los datos del departamento 40:

```
SELECT *
FROM   dept
WHERE  deptno = 40;
```

3. Seleccionar los datos de los empleados cuyo sueldo es mayor que 1000:

```
SELECT *
FROM emp
WHERE sal > 1000;
```

Predicados

4. Obtener los datos de los departamentos ordenados por el nombre del departamento:

```
SELECT *
FROM   dept
ORDER BY dname;
```

5. Obtener la comisión, departamento y nombre de los empleados cuyo salario sea inferior a 1900, ordenándolos por departamento en orden creciente, y por comisión en orden decreciente dentro de cada departamento:

```
SELECT comm, deptno, ename
FROM   emp
WHERE  sal < 1900
ORDER BY deptno, comm DESC;
```

Predicado de valor nulo

Predicado de valor nulo (IS NULL)

- Los predicados de comparación no sirven para determinar los valores nulos. Por ejemplo, no es válido `COMM = NULL` porque sería discernir si un valor que puede ser desconocido es igual a otro también desconocido.
- Formato: <expr1> IS [NOT] NULL

Ejemplo incorrecto:

```
SELECT *
FROM emp
WHERE comm = NULL;
```

Ejemplo correcto:

```
SELECT *
FROM emp
WHERE comm IS NULL;
```

Predicado BETWEEN

Predicado de rango o predicado **BETWEEN**

- Compara si los valores de una expresión están o no entre los valores de otras dos.
- Formato: <expr1> [NOT] **BETWEEN** <expr2> **AND** <expr3>

```
SELECT *
FROM emp
WHERE sal BETWEEN 1500 AND 3000;
```

```
SELECT *
FROM emp
WHERE sal >=1500 AND sal <=3000 ;
```

(predicado compuesto "AND" lo veremos más adelante)

Predicado de pertenencia a conjunto

Predicado de pertenencia a conjunto (**IN**)

- Comprueba si el valor de una expresión coincide con alguno de los valores incluidos en una lista de expresiones.
- Formato: <expr1> [NOT] **IN** (<expr2>[, <expr3>, …])

```
SELECT *
FROM emp
WHERE deptno IN (10,30,40);
```

```
SELECT *
FROM emp
WHERE deptno = 10 OR deptno = 30
                  OR deptno =40;
```

(predicado compuesto “OR" lo veremos más adelante)

```
SELECT ename
FROM emp
WHERE job IN (‘CLERK’, ’SALESMAN’);
```

Predicado LIKE

Predicado de correspondencia con un patrón o modelo

Comprueba si el valor de una expresión alfanumérica se corresponde con un modelo. El modelo puede incluir dos caracteres que actúan como comodines:

_	Indica un único carácter, incluido el blanco.
%	Indica una cadena de caracteres de cualquier longitud, incluida la cadena vacía.

Formato: <expr1> [NOT] **LIKE** <modelo>

```
SELECT *
FROM emp
WHERE ename LIKE '%NE%'
```

```
SELECT *
FROM emp
WHERE ename LIKE '_____'
```

Predicados compuestos

Son la unión de dos o más predicados simples mediante los operadores lógicos AND, OR y NOT. Al existir una lógica de tres valores, debemos considerar el efecto del valor **NULL**.

```
SELECT *
FROM emp
WHERE sal > 1000 AND job = ‘CLERK’;

SELECT ename
FROM emp
WHERE sal > 1000 OR comm > 500;
```

Funciones

Pueden formar parte de una expresión. Constan de un nombre y pueden uno o más argumentos entre paréntesis. El **resultado** es **un valor**.

Algunos ejemplos:

- Funciones numéricas
  - ABS(x)
  - POWER(x,y)
  - SQRT(x)
  - MAX(x,y)
  - MIN(x,y)
- Funciones alfanuméricas
  - UPPER(s)
  - LOWER(s)
  - SUBSTR(s,x,y)

Ejemplo con SUBSTR

SELECT SUBSTR('prueba',2,4); ➔ rueb
SELECT UPPER('prueba'); ➔ PRUEBA

COALESCE

Función útil cuando hay valores nulos en expresiones.

Formato**: COALESCE(<expr1>, <expr2>)**

Funcionamiento:

Evalúa la expresión 1. Si su valor es distinto de NULL, devuelve dicho valor. En caso contrario, evalúa la expresión 2 y devuelve el resultado.

```
SELECT COALESCE(sal + comm, sal)
FROM emp;
```

En este ejemplo se evalúa la suma del salario y la comisión de cada empleado. Si el resultado es distinto de NULL, se devuelve dicho resultado. Si el resultado es NULL (debido a que la comisión del empleado es NULL), se evalúa el salario de cada empleado y se devuelve ese valor.

JOIN (unir varias tablas)

Permite obtener como resultado de una consulta datos de más de una tabla, vinculando las columnas de varias tablas mediante operadores de comparación.

Sintaxis:

```
SELECT *
FROM <tabla1> JOIN <tabla2>
ON <condición de join>
```

La vinculación de las columnas se determina en la condición de join y usando el INNER JOIN aparecen las que cumplen esa condición.

Ejemplos:

```
SELECT *
FROM emp JOIN dept
ON emp.deptno=dept.deptno
```

```
SELECT *
FROM emp e JOIN emp emp j
ON e.mgr=j.empno
```

JOIN (unir varias tablas)

Departamento

Nombre_Dept	Num_Dept	NSS-Jefe	Fech_Ini_Jefe
Dirección	1	1122	12-8-95
Desarrollo	2	5544	25-5-97

Localizaciones_Dept

Número_Dept	Localización_Dept
1	A Coruña
2	Ferrol
2	Lugo
2	Vigo

```
SELECT *
FROM Departamento, Localizaciones_Dept
```

Nombre_Dept	Num_Dept	NSS_Jefe	Fech_Ini_Jefe	Número_Dept	Localización_Dept
Dirección	1	1122	12-8-95	1	A Coruña
Dirección	1	1122	12-8-95	2	Ferrol
Dirección	1	1122	12-8-95	2	Lugo
Dirección	1	1122	12-8-95	2	Vigo
Desarrollo	2	5544	25-5-97	1	A Coruña
Desarrollo	2	5544	25-5-97	2	Ferrol
Desarrollo	2	5544	25-5-97	2	Lugo
Desarrollo	2	5544	25-5-97	2	Vigo

Resultado Deseado

Nombre_Dept	Localización_Dept
Dirección	A Coruña
Desarrollo	Ferrol
Desarrollo	Lugo
Desarrollo	Vigo

JOIN (unir varias tablas)

Departamento

Nombre_Dept	Num_Dept	NSS-Jefe	Fech_Ini_Jefe
Dirección	1	1122	12-8-95
Desarrollo	2	5544	25-5-97

Localizaciones_Dept

Número_Dept	Localización_Dept
1	A Coruña
2	Ferrol
2	Lugo
2	Vigo

```
SELECT *
FROM Departamento, Localizaciones_Dept
```

Nombre_Dept	Num_Dept	NSS_Jefe	Fech_Ini_Jefe	Número_Dept	Localización_Dept
Dirección	1	1122	12-8-95	1	A Coruña
Dirección	1	1122	12-8-95	2	Ferrol
Dirección	1	1122	12-8-95	2	Lugo
Dirección	1	1122	12-8-95	2	Vigo
Desarrollo	2	5544	25-5-97	1	A Coruña
Desarrollo	2	5544	25-5-97	2	Ferrol
Desarrollo	2	5544	25-5-97	2	Lugo
Desarrollo	2	5544	25-5-97	2	Vigo

Resultado Deseado

Nombre_Dept	Localización_Dept
Dirección	A Coruña
Desarrollo	Ferrol
Desarrollo	Lugo
Desarrollo	Vigo

JOIN (unir varias tablas)

Departamento

Nombre_Dept	Num_Dept	NSS-Jefe	Fech_Ini_Jefe
Dirección	1	1122	12-8-95
Desarrollo	2	5544	25-5-97

Localizaciones_Dept

Número_Dept	Localización_Dept
1	A Coruña
2	Ferrol
2	Lugo
2	Vigo

```
Select *
FROM Departamento JOIN Localizaciones_Dept ON
Departamento.Num_Dept=Localizaciones_Dept.Numero_Dept
```

Nombre_Dept	Num_Dept	NSS_Jefe	Fech_Ini_Jefe	Número_Dept	Localización_Dept
Dirección	1	1122	12-8-95	1	A Coruña
Desarrollo	2	5544	25-5-97	2	Ferrol
Desarrollo	2	5544	25-5-97	2	Lugo
Desarrollo	2	5544	25-5-97	2	Vigo

Inserción (INSERT)

Sentencia INSERT

Permite introducir filas en una tabla.

Formato:

```
INSERT INTO <tabla> [(col1, col2, …)]
{ VALUES (<valor col1>, <valor col2>, …) / Sentencia SELECT }
```

Ejemplos:

1. Añade una fila a la tabla emp con datos en todas sus columnas, incluido un valor nulo.

```
INSERT INTO emp
VALUES (7777, 'ANA', 'ANALISTA', 7369, '15/01/05', 2500, NULL, 10);
```

2. Añade una fila a la tabla con datos en las columnas referenciadas en la lista.

```
INSERT INTO emp (empno, ename, sal)
VALUES (8888, 'Juan', 2000);
```

Inserción (INSERT)

3. Añade filas con la sentencia SELECT.

```
INSERT INTO emp_new
SELECT empno, ename, sal, comm
FROM emp_old
WHERE deptno = 10 AND sal > 1500;
```

Actualización (UPDATE)

Sentencia UPDATE

Permite modificar datos en columnas existentes.

Formato:

```
UPDATE <tabla>
SET <columna1> = <expresión1> [, <columna2> = <expresión2>, …]
[WHERE <predicado>]
```

Ejemplos*:*

1. Modifica el valor de la columna sal en todas las filas de la tabla emp.

```
UPDATE emp
SET sal = sal + 100;
```

2. Modifica el valor de varias columnas para las filas que cumplen la condición.

```
UPDATE emp
SET comm = 110, deptno = 10
WHERE comm IS NULL;
```

Borrado (DELETE)

Sentencia DELETE

Permite borrar filas de una tabla.

Formato:

```
DELETE FROM <tabla>
[WHERE <predicado>]
```

Ejemplos*:*

1. Borra las filas que cumplan la condición indicada.

```
DELETE FROM emp
WHERE sal > 2000;
```

2. Si no se pone ninguna condición, o sea, si no existe la subcláusula WHERE, el borrado afecta a todas las filas de la tabla.

```
DELETE FROM emp;
```

Creación de tablas (CREATE TABLE)

Sentencia CREATE TABLE

Crea la estructura de una tabla.

Formato (simplificado):

```
CREATE TABLE <tabla> (
      <definición de columna 1>,
      ...,
      <definición de columna n>,
      <definición de restricción 1>,
      ...,
      <definición de restricción n>
);
```

En la definición del atributo o columna puede aparecer:

```
<nombre de columna>  <tipo de dato> [ <restricción de columna> ]
```

Creación de tablas (CREATE TABLE)

Restricciones

Las restricciones se usan para declarar en el modelo relacional las condiciones que los datos cumplen en el mundo real.

Pueden ser de diverso tipo (valor defectivo, valor nulo, clave primaria, clave foránea, de unicidad, de comprobación de una condición, etc.)

- Valor defectivo de una columna. Se expresa con `DEFAULT <valor>`
- Valor (defectivo) nulo o no nulo de una columna. Se expresa con `[NOT] NULL`.

```
CREATE TABLE emp1 (
      empno NUMERIC(4) NOT NULL,
      ename VARCHAR(15),
      mgr NUMERIC(4) DEFAULT 7500,
      deptno NUMERIC(2)
);
```

Creación de tablas (CREATE TABLE)

Restricción de clave primaria

Indica la condición de que un conjunto de columnas toma valores diferentes para cada fila y ninguno de ellos es nulo. Se expresa mediante PRIMARY KEY.

Toda restricción puede nombrarse incluyendo CONSTRAINT <nombre restricción>

```
CREATE TABLE emp2 (
   empno NUMERIC(4) PRIMARY KEY,
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2));
```

```
CREATE TABLE emp2 (
   empno NUMERIC(4),
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2),
   PRIMARY KEY (empno));
```

```
CREATE TABLE emp2 (
   empno NUMERIC(4),
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2),
   CONSTRAINT cp_emp2 PRIMARY KEY (empno));
```

Creación de tablas (CREATE TABLE)

Restricción de unicidad

Indica que un conjunto de atributos no puede tener valores iguales en distintas filas.

```
CREATE TABLE emp2 (
   empno NUMERIC(4) PRIMARY KEY,
   ename VARCHAR(15) UNIQUE,
   mgr NUMBER(4),
   deptno NUMERIC(2));
```

```
CREATE TABLE emp2 (
   empno NUMERIC(4) PRIMARY KEY,
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2),
   CONSTRAINT unq_emp2 UNIQUE (ename));
```

Restricción de comprobación o CHECK

Permiten declarar una condición que deben cumplir uno o más atributos.

```
CREATE TABLE emp2 (
   empno NUMERIC(4) PRIMARY KEY,
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2),
   CONSTRAINT sal_emp2 CHECK (sal > 1000));
```

Creación de tablas (CREATE TABLE)

Restricción de clave foránea

Un conjunto de atributos es una clave foránea si sus valores se corresponden con los de otro conjunto de atributos que es clave candidata en otra tabla.

```
CREATE TABLE emp2 (
   empno NUMERIC(4) PRIMARY KEY,
   ename VARCHAR(15),
   mgr NUMBER(4),
   deptno NUMERIC(2),
   CONSTRAINT fk_deptno FOREIGN KEY (deptno) REFERENCES dept2(deptno));
```