



# Bases de Datos - SQL

Santiago W. Fernández Lorenzo



# Introducción

Conceptos previos

# Introducción al lenguaje SQL

## Contenidos

- Descripción de entorno y tablas a usar
- ¿Cómo consultamos y modificamos datos?
- ¿Cómo definimos la base de datos?
- 9 clases de 1.5 horas => 80 diapositivas -> 10 por sesión

## Bibliografía

- MySQL + Oracle database
- W3 - Schools
- Susana Ladra / Oscar Pedreira / Luis Ares

# 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 propio.

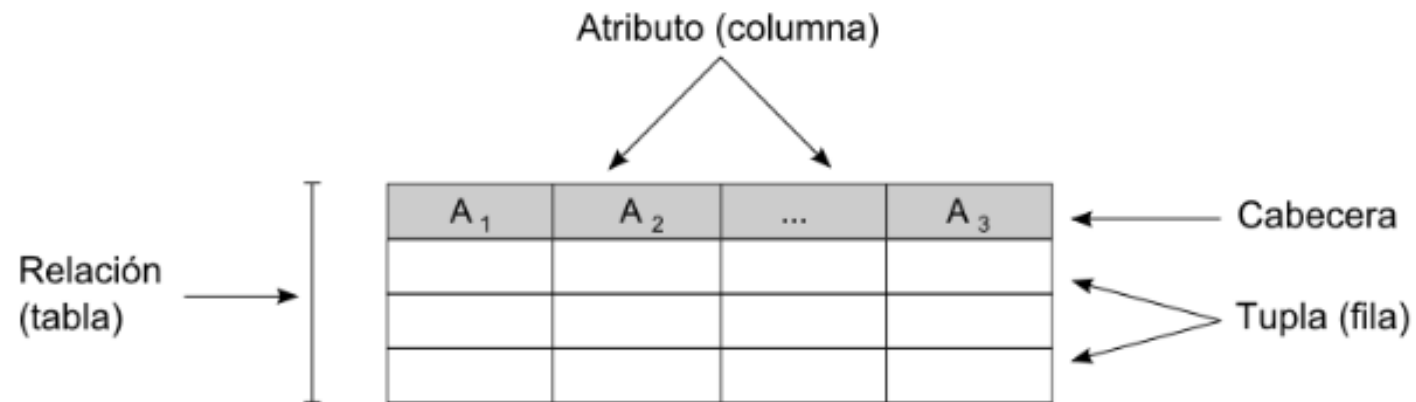
## **Sistema de Gestión de Bases de Datos (SGBD)**

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

# Tabla

Es la estructura básica en dónde se almacenarán los datos

- **Registro o tupla:** Es cada *fila* de la tabla.
- **Columna:** Representa un atributo. Debe tener un tipo.
- **Campo:** Es cada *celda* de la tabla.



# Tablas

Es la estructura básica en dónde se almacenarán los datos

- **Registro o tupla:** Es cada *fila* de la tabla.
- **Columna:** Representa un atributo. Debe tener un tipo.
- **Campo:** Es cada *celda* de la tabla.

Tabla de clientes

DNI	Nombre	Dirección	Teléfono
11.222.333-A	Juan Ares	Calle Mayor 1	981 123 456
44.555.666-B	Miguel Pérez	Calle Mayor 2	981 111 222
...	...	...	...

# SQL

- **SQL:** Structured Query Language
- Lenguaje estándar para realizar consultas en BD relacionales
  - La mayoría de los SGBD tienen *pequeñas diferencias* a la hora de construir las sentencias SQL.
- Permite recuperar los datos de una forma *intuitiva*
  - Con una *sentencia* podemos indicar:
    - ¿Qué datos queremos?
    - ¿De dónde?
    - ¿Con qué condiciones?
    - ¿En qué orden?

# SQL

## Data Manipulation Language

Se emplean para trabajar con los datos

- **SELECT** -> Consultar datos
- **INSERT** -> Insertar datos
- **UPDATE** -> Actualizar datos
- **DELETE** -> Eliminar datos



## Data Definition Language

Se emplean para modificar la base de datos

- **CREATE** -> Crear objetos
- **ALTER** -> Modificar objetos
- **DROP** -> Eliminar objetos





# Entorno de prácticas

MySQL Community Edition + MySQL Workbench

- SGDB relacional
- Probablemente el más popular
- Código abierto (GPL)
- En 2010 la compra Oracle y aparece MariaDB
- Existen muchos otros, pero usaremos éste

# MySQL Workbench

The screenshot displays the MySQL Workbench application window. The interface is divided into several panes:

- Left Sidebar (MANAGEMENT, INSTANCE, PERFORMANCE, SCHEMAS):** Contains navigation options for server management, instance control, performance monitoring, and a SCHEMAS pane. The SCHEMAS pane is expanded, showing a tree view of databases. The 'example' database is selected, and the 'dept' table is highlighted.
- SQL File 3\* (Query Window):** Contains the SQL query: `select * from dept;`. A toolbar above the query area includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown.
- Result Grid:** Displays the results of the query in a table format. The table has columns: #, deptno, dname, and loc. The data rows are: 1 | 10 | ACCOUNTING | NEW YORK, 2 | 20 | RESEARCH | DALLAS, 3 | 30 | SALES | CHICAGO, 4 | 40 | OPERATIONS | BOSTON, and a summary row with \* | NULL | NULL | NULL.
- Bottom Panel:** Shows the 'Object Info' tab for the 'dept' table, with a 'Session' tab also visible. The status bar at the bottom indicates 'Query Completed'.

#	deptno	dname	loc
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON
*	NULL	NULL	NULL



# Introducción

Base de datos del curso

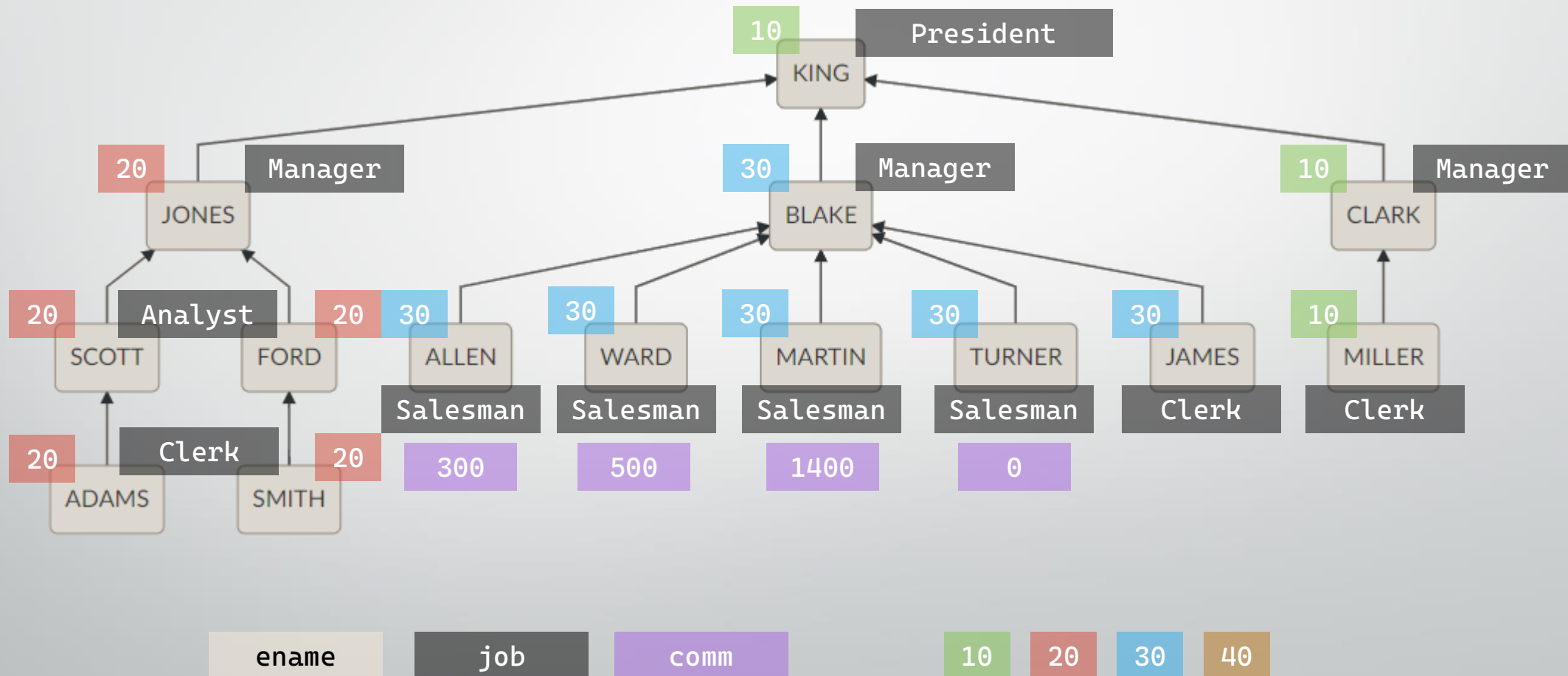
# Modelar empleados de empresa

En nuestra empresa tenemos **empleados** con diferentes **puestos** (*Clerk, Salesman, Analyst, Manager*) cada empleado tiene un **superior** asociado, excepto el presidente de la empresa.

Cada empleado tiene un **sueldo** diferente, en función de cómo haya negociado su contrato. Los *Salesman* tiene un **plus** en su sueldo en función de los objetivos que haya cumplido el año anterior. También queremos almacenar la **fecha de contratación.**

Cada empleado trabaja en un único departamento, que (por ahora) lo identificaremos por su **código** (*10, 20, 30, 40*).

# Empleados: Jerarquía



# Tabla de Empleados

- Empleados -> emp
  - Número de empleado -> empno
  - Nombre -> ename
  - Puesto -> job
  - Responsable -> mgr
  - Fecha de contratación -> hiredate
  - Salario -> sal
  - Comisión -> comm
  - Departamento -> deptno

# Tabla de Empleados

- Empleados: Tabla "emp"

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

# Tabla Empleados

- Empleados: Tabla "emp"
  - Número de empleado -> empno
  - Nombre -> ename
  - Puesto -> job
  - Responsable -> mgr
  - Fecha de contratación -> hiredate
  - Salario -> sal
  - Comisión -> comm
  - Departamento -> deptno

emp	
empno	DECIMAL(4,0)
ename	VARCHAR(10)
job	VARCHAR(9)
mgr	DECIMAL(4,0)
hiredate	DATE
sal	DECIMAL(7,2)
comm	DECIMAL(7,2)
deptno	DECIMAL(2,0)



# DESC

- DESC nos permite ver la descripción de la tabla
  - DESC emp;

#	Field	Type	Null	Key	Default
1	empno	decimal(4,0)	NO	PRI	NULL
2	ename	varchar(10)	YES		NULL
3	job	varchar(9)	YES		NULL
4	mgr	decimal(4,0)	YES		NULL
5	hiredate	date	YES		NULL
6	sal	decimal(7,2)	YES		NULL
7	comm	decimal(7,2)	YES		NULL
8	deptno	decimal(2,0)	YES	MUL	NULL

# Tipos de datos

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

- Tipos de datos numéricos
  - INTEGER o INT, SMALLINT, BIGINT, ...    Enteros (+ -)
  - DECIMAL(e,d) o DEC(e,d), ...    Enteros con decimales
  - FLOAT, DOUBLE PRECISION, ...    Notación científica
- Tipos de datos alfanuméricos
  - CHAR(n), VARCHAR(n)
- Tipos de datos temporales
  - DATE, TIME, DATETIME

# Valores nulos

- ¿Cómo guardamos un valor que no conocemos?
  - ¿0? No, esto indica que lo conocemos y es 0
  - "" Cadena de texto vacía. Tampoco, eso indica que es un texto y no tiene nada
- Un valor **NULL** indica que el valor del campo es desconocido.
- Un valor **NULL** es diferente a un valor 0 o a una cadena de caracteres vacía
- Se puede indicar que una columna de una tabla admita valores **NULL** o no
- ¿Qué pasará con las operaciones que involucren a **NULL**?
- **Lo veremos más adelante...**

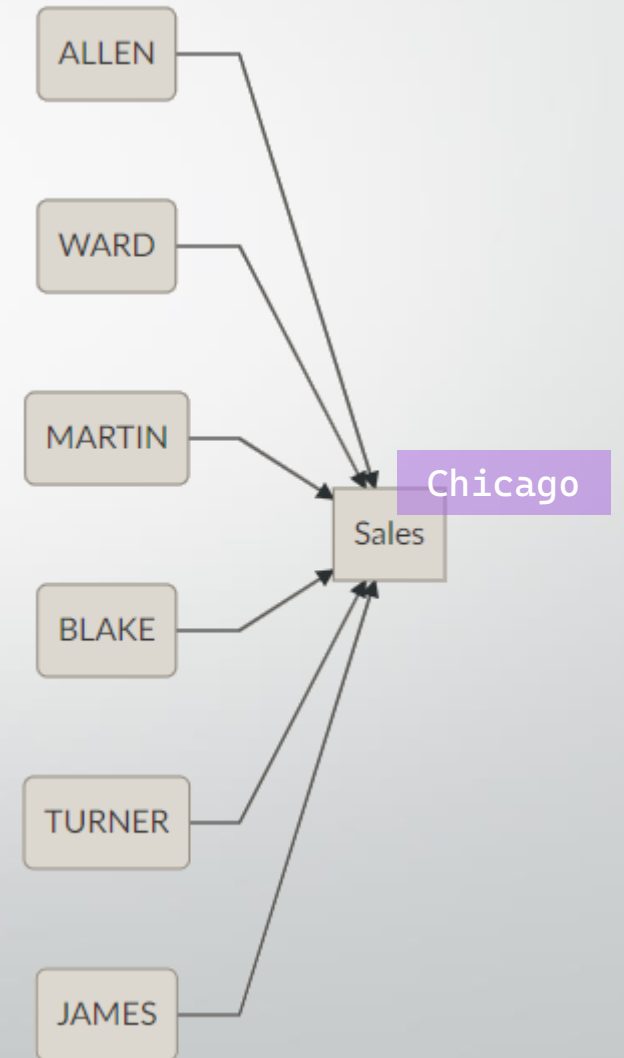
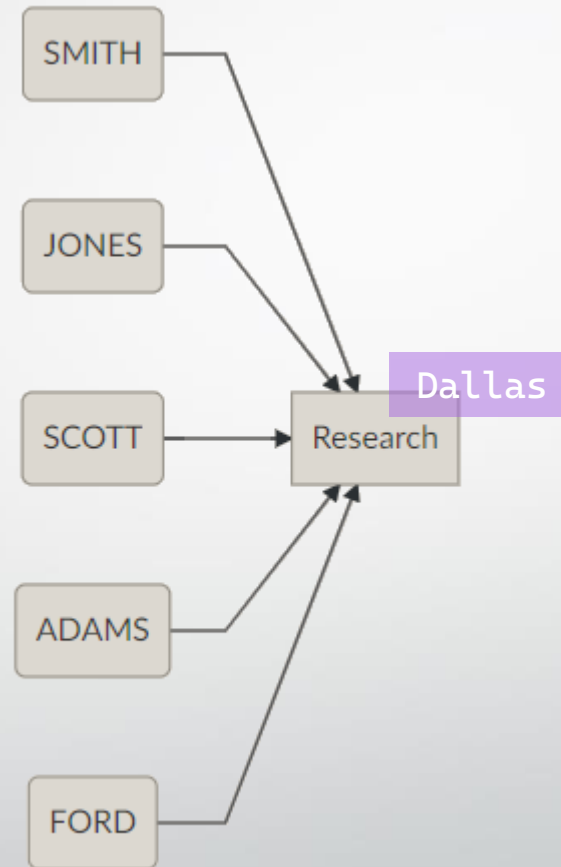
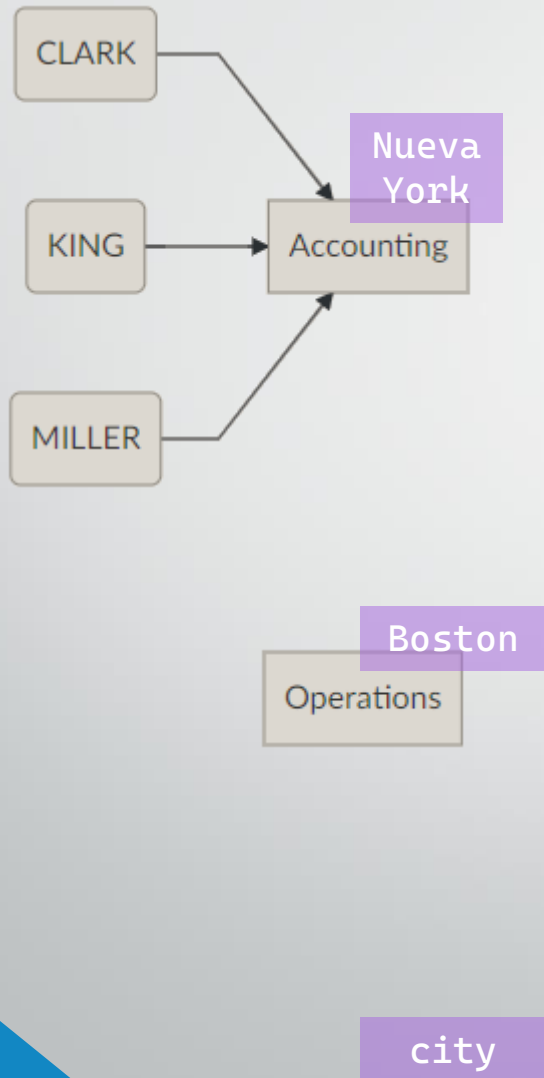
# Modelar empleados / departamentos

En nuestra empresa tenemos empleados con diferentes puestos (*Clerk, Salesman, Analyst, Manager*) cada empleado tiene un superior asociado, excepto el presidente de la empresa.

Cada empleado tiene un sueldo diferente, en función de cómo haya negociado su contrato. Los *Salesman* tiene un plus en su sueldo en función de los objetivos que haya cumplido el año anterior. También queremos almacenar la fecha de contratación.

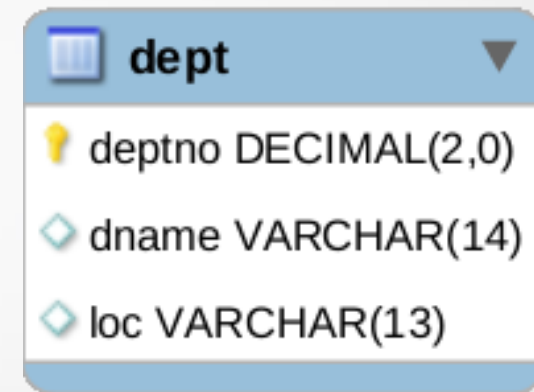
Nuestra empresa tiene la operativa dividida en 4 departamentos, (*Accounting, Research, Sales, Operations*), cada departamento está en una ciudad. Nos gustaría poder consultar con facilidad qué empleados trabajan en cada departamento.

# Departamentos



# Tabla Departamento

- Empleados: Tabla "dept"
  - Número de departamento
  - Nombre del departamento
  - Ubicación



A screenshot of a database interface showing the definition of a table named 'dept'. The table has three columns: 'deptno' of type DECIMAL(2,0) marked as a primary key with a yellow key icon, 'dname' of type VARCHAR(14) marked with a blue diamond icon, and 'loc' of type VARCHAR(13) also marked with a blue diamond icon.

dept	
deptno	DECIMAL(2,0)
dname	VARCHAR(14)
loc	VARCHAR(13)

deptno	dname	loc
10	ACCOUNTING	NEWYORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



# Data Modeling Language I

Consultar y manipular datos de la base de datos  
Sentencia SELECT

# SELECT

La sentencia SELECT permite recuperar los datos

```
SELECT * FROM emp;
```

Sintaxis completa

- SELECT ----> ¿Qué queremos?
- FROM -----> ¿De dónde?
- WHERE ----> ¿Bajo qué condiciones?
- GROUP BY -> ¿Hacemos grupos?
- HAVING ----> ¿Condiciones sobre los grupos?
- ORDER BY -> ¿En qué orden?



# SELECT

El orden de ejecución de las cláusulas es

- FROM (obligatoria)
- WHERE (opcional)
- SELECT (obligatoria)
- ORDER BY (opcional)
  - ASC.: orden ascendente (por defecto)
  - DESC: orden descendente

# SELECT

Ejemplos:

Obtener todos los datos de la tabla de empleados:

```
SELECT *  
FROM emp;
```

Obtener los nombres y los códigos de los empleados:

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

# LIMIT

- Permite limitar el número de resultados a mostrar
- ¡Tomará los X primeros resultados que aparezcan!
- ¡Si no se especifica orden esos resultados pueden diferir!

```
SELECT *  
FROM emp  
LIMIT 5;
```

```
SELECT *  
FROM emp  
ORDER BY sal  
LIMIT 5;
```

# DISTINCT

Permite eliminar filas repetidas

```
SELECT DISTINCT ...  
FROM ...
```

Obtener los puestos de trabajo (Sin repetir)

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

Obtener los puestos de trabajo y comisiones.  
Sin filas repetidas

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

# [AS]

Nos permite poner un alias a una de la columna que se muestra en el resultado o a las tablas que usamos en la consulta

```
SELECT DISTINCT e.job AS 'Trabajo'  
FROM emp AS e;
```

En MySQL escribir AS es opcional:

```
SELECT DISTINCT e.job 'Trabajo'  
FROM emp e;
```

# WHERE

- Permite aplicar condiciones sobre los datos a recuperar

## Matemáticos

- = != (En algunos SGBD <>)

- SELECT \* FROM emp WHERE sal = 1000

- > >= < <=

- SELECT \* FROM emp WHERE sal <= 1000

- + - \* /

- SELECT \* FROM emp WHERE sal/2 > 1000

# WHERE

## Grupos y Strings

- [NOT] BETWEEN (Rango)
  - `SELECT * FROM emp WHERE sal BETWEEN 1000 AND 2000`
- [NOT] IN (Conjunto)
  - `SELECT * FROM emp WHERE empno IN (7782, 7934)`
- [NOT] LIKE (Patrón)
  - `SELECT * FROM emp WHERE ename LIKE 'ALLEN'`
  - `%` -> 0..N caracteres
  - `SELECT * FROM emp WHERE ename LIKE 'A%N'`
  - `_` -> 1 carácter
  - `SELECT * FROM emp WHERE ename LIKE 'A_L_N'`

# WHERE

## Condiciones lógicas

- AND (Y)
  - `SELECT * FROM emp WHERE sal > 2000 AND job LIKE 'MANAGER'`
- OR (O)
  - `SELECT * FROM emp WHERE job LIKE 'CLERK' OR deptno = 30`
- NOT (Patrón)
  - `SELECT * FROM emp WHERE job NOT LIKE 'SALESMAN'`
- Combinados
  - `SELECT * FROM emp  
WHERE (sal > 2500 OR deptno = 10) AND ename NOT LIKE '%A%'`



# ORDER BY

Nos permite especificar el orden en el que queremos que se recuperen los datos.

- **ASC** en orden ascendente (Predeterminado)
- **DESC** en orden descendente

```
SELECT *  
FROM emp e  
ORDER BY e.ename
```

```
SELECT *  
FROM emp e  
ORDER BY e.ename DESC
```

# Predicados

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

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

Seleccionar todos los datos del departamento 20

```
SELECT *  
FROM dept  
WHERE deptno = 20;
```

# Predicados

Obtener todos los datos de los empleados cuyo sueldo es mayor que 1000

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

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

# Leyes de DeMorgan

***(no A) o (no B) es lo mismo que no (A y B)***

- *El superior no sea BLAKE (7698) o el salario no sea menor de 1500*
- NOT (mgr = 7698) OR NOT (sal < 1500)
- *Ni el superior sea BLAKE ni el salario sea menor de 1500*
- NOT (mgr = 7698 AND sal < 1500)

## ***Cambiando operadores matemáticos***

- *El superior no sea BLAKE (7698) o el salario no sea menor de 1500*
- mgr <> 7698 OR sal >= 1500
- *Ni el superior sea BLAKE ni el salario sea menor de 1500*
- NOT (mgr = 7698 AND sal < 1500)

# Leyes de DeMorgan

**(no A) y (no B) es lo mismo que no (A o B)**

- El superior *no* sea BLAKE (7698) *y* el salario *no* sea menor de 1500
- NOT (mgr = 7698) AND NOT (sal < 1500)
- El superior *no* sea BLAKE ni el salario sea menor de 1500
- NOT (mgr = 7698 OR sal < 1500)

## ***Cambiando operadores matemáticos***

- El superior *no* sea BLAKE (7698) *y* el salario *no* sea menor de 1500
- NOT (mgr = 7698) AND NOT (sal < 1500)
- El superior *no* sea BLAKE ni el salario sea menor de 1500
- NOT (mgr = 7698 OR sal < 1500)