TEMA 4

Introducción a SQL


```
CREATE TABLE ARTICULO (
codart varchar2(8) CONSTRAINT ARTICULO_PK PRIMARY KEY,
descrip varchar2(50),
precio number(7,2),
stock number(6),
stock_min number(6));

CREATE TABLE PROVINCIA (
codprov varchar2(2)
        CONSTRAINT PROVINCIA_PK PRIMARY KEY,
nombre varchar2(50));

CREATE TABLE LOCALIDAD (
codloc varchar2(5)
        CONSTRAINT LOCALIDAD_PK PRIMARY KEY,
nombre varchar2(50),
codprov varchar2(2)
        CONSTRAINT LOCALIDAD_PROVINCIA_FK REFERENCES PROVINCIA);

CREATE TABLE CPOSTAL (
codpostal varchar2(5)
        CONSTRAINT CPOSTAL_PK PRIMARY KEY,
codloc varchar2(5)
        CONSTRAINT CPOSTAL_LOCALIDAD_FK REFERENCES LOCALIDAD);

CREATE TABLE CLIENTE (
codcli number(5) CONSTRAINT CLIENTE_PK PRIMARY KEY,
nombre varchar2(50),
direccion varchar2(50),
```

CURSO 23-24

ÍNDICE

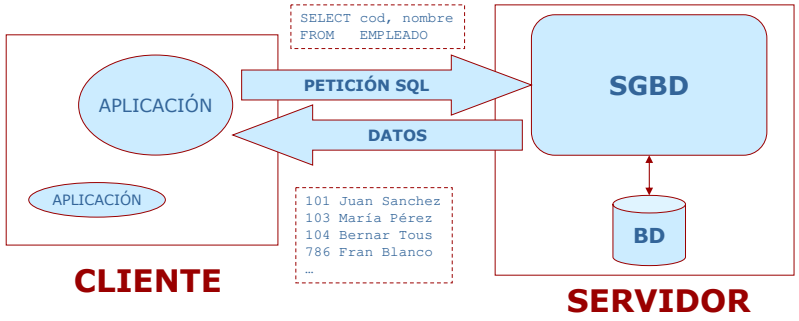
ÍNDICE

1.- INTRODUCCIÓN

- **SQL= Structured Query Language**
  - Lenguaje comercial para BD relacionales.
  - Se utiliza para la DEFINICIÓN, MANIPULACIÓN y GESTIÓN de BD relacionales:
    - Creación de la BD; consultas; altas, bajas y modificaciones; definición de restricciones de integridad; gestión de usuarios, gestión del almacenamiento, etc.
    - Esta amplia funcionalidad hace que sea un lenguaje utilizado por administradores, desarrolladores, e incluso usuarios avanzados.
  - Es un lenguaje **DECLARATIVO**: se especifica QUÉ se quiere obtener, no COMO obtenerlo.

1.- INTRODUCCIÓN

- SQL nos permite la comunicación con el SGBD:



1.- INTRODUCCIÓN

- SQL se puede utilizar de las siguientes formas:
  - **MODO INTERACTIVO**:
    - El usuario escribe las sentencias SQL y se muestran los resultados en pantalla.
    - Trabajamos así cuando utilizamos editores de comandos SQL: SQL* plus, SQL Developer, Toad, Antiguo SQL* plus Worksheet, etc.
  - **MODO EMBEBIDO**:
    - Las sentencias o comandos SQL se encuentran inmersos (o embebidos) entre las instrucciones de un lenguaje de programación anfitrión (Java, PHP, PL/SQL, etc.)
    - Los resultados los gestiona el código del programa.

ÍNDICE

2.- TIPOS DE SENTENCIAS SQL

- SQL proporciona un gran repertorio de sentencias que se utilizan para realizar distintas tareas.
- Dependiendo de las tareas, podemos clasificar las sentencias SQL en 2 grandes tipos. Cada uno de estos tipos de sentencias define un sublenguaje SQL.
  - **Sentencias de DEFINICIÓN de datos**:
    - **DDL** (Data Definition Language)
    - Se utilizan para definir y mantener la estructura de la BD.
  - **Sentencias de MANIPULACIÓN de datos**:
    - **DML** (Data Manipulation Language)
    - Se utilizan para manipular los datos contenidos en la BD.

2.- TIPOS DE SENTENCIAS SQL

DML (MANIPULACIÓN DE DATOS)	
SENTENCIA	**DESCRIPCION**
SELECT	CONSULTA de datos
INSERT	INSERCIÓN de datos
UPDATE	MODIFICACIÓN de datos
DELETE	BORRADO de datos

DDL (DEFINICIÓN DE OBJETOS)	
SENTENCIA	**DESCRIPCION**
CREATE	CREACIÓN de objetos de la BD
DROP	BORRADO de objetos de BD
ALTER	MODIFICACIÓN de objetos de la BD
GRANT/REVOKE	CONCEDER/REVOCAR privilegios
COMMIT/ROLLBACK	VALIDAR/DESHACER Transacción actual

2.- TIPOS DE SENTENCIAS SQL

- Actualmente se tiende a subdividir el DDL en 2 sublenguajes más:
  - **DCL** (Data Control Language)
  - **DTL** (Data Transaction Language)

DCL (CONTROL DE ACCESO)	
SENTENCIA	**DESCRIPCION**
GRANT/REVOKE	CONCEDER/REVOCAR privilegios

DTL (CONTROL TRANSACCIONES)	
SENTENCIA	**DESCRIPCION**
COMMIT/ROLLBACK	VALIDAR/DESHACER Transacción actual

2.- TIPOS DE SENTENCIAS SQL

- SQL= DML + DDL (Clasificación tradicional)
  - **DML**= Lenguaje de manipulación de datos:
    - Consulta
    - Inserción
    - Modificación
    - Borrado

    DE DATOS
  - **DDL**= Lenguaje de definición de datos
    - Creacion
    - Modificación
    - Borrado

    DE LOS OBJETOS DE LA BD
  - El DDL se encarga de la definición de todos los objetos y estructuras (básicas, derivadas, internas) de la BD:
    - Tablas, vistas, usuarios, privilegios, etc.

2.- TIPOS DE SENTENCIAS SQL

- Considerando la tendencia actual…
- SQL= DML + DDL + DCL+ DTL
  - DML= Lenguaje de manipulación de datos:
    - Consulta
    - Inserción
    - Modificación
    - Borrado

    DE DATOS
  - DDL= Lenguaje de definición de datos
    - Creacion
    - Modificación
    - Borrado

    DE LOS OBJETOS DE LA BD
  - DCL= Lenguaje de control de acceso
  - DTL= Lenguaje de control de transacciones

ÍNDICE

3.- TIPOS DE DATOS DE ORACLE

- **CADENAS DE CARACTERES.**
  - CHAR (tamaño)
  - VARCHAR2 (tamaño)
  - ~~LONG~~
  - CLOB
- **NUMÉRICOS.**
  - NUMBER [(precision[, escala])]
- **FECHA.**
  - DATE
- **DATOS BINARIOS.**
  - RAW (tamaño)
  - ~~LONG RAW~~
  - BLOB

Con más detalle…

3.1.- CADENAS DE CARACTERES

- **CHAR (tamaño)**
  - Cadenas de caracteres de **longitud fija**
    - ASCII o EBCDIC según la configuración en servidor.
    - Tamaño máximo[1]: 2000 caracteres.
  - Si se introduce en el campo una cadena:
    - De menor longitud a la definida: se rellena con blancos a la derecha hasta que alcanza la longitud.
    - De mayor longitud: Oracle devuelve un error.
- **VARCHAR2 (tamaño)**
  - Cadenas de caracteres de **longitud variable**.
    - Tamaño máximo: 4000 caracteres.
  - Si se introduce en el campo una cadena:
    - De menor longitud a la definida: se almacena con la longitud introducida (No se rellena con blancos).
    - De mayor longitud: Oracle devuelve un error.

1: Consultar siempre el tamaño maximo en ayudas en línea, ya que puede cambiar con las versiones.

3.1.- CADENAS DE CARACTERES

- **LONG (en desuso)**
  - Cadenas de caracteres de gran tamaño de longitud variable.
    - tamaño máximo: 2GB
  - Se utiliza para almacenar textos muy grandes.
  - Tiene MUCHAS restricciones:
    - Sólo se puede definir una columna LONG por tabla.
    - No pueden aparecer en CONSTRAINTS.
    - No sirve para indexar.
    - No es posible su uso en clausulas WHERE, GROUP BY, ORDER BY, DISTINCT, etc.
    - No se puede utilizar con UNION, INTERSECT, MINUS.
    - Etc, etc, etc.
  - Está en desuso. Se mantiene por compatibilidad.

3.2.- NUMÉRICOS

- **NUMBER [(precision[, escala])]**
  - Datos numéricos.
    - tanto enteros como reales, con o sin signo.
  - **Precision**: número total de dígitos (de 1 a 38)
  - **Escala**: número de dígitos decimales (de -38 a 127)
    - Reales en coma fija: NUMBER (precision, escala)
      - Se indica precisión y escala.
      - Escala Negativa: NO LA USAMOS.
    - Enteros de longitud predeterminada: NUMBER (precision)
      - Se indica la precisión. Escala es 0, no hay decimales.
    - Enteros sin longitud y Reales en coma flotante: NUMBER
      - Sin indicar precisión ni escala (precisión es 38)
      - El número se almacena tal y como se introduce, sea entero o real.
      - Se automatiza la escala (la precisión de las operaciones peligra)

3.2.- NUMÉRICOS

- Las siguientes **tablas de ejemplo** muestran como se almacenan los datos numéricos en Oracle con diferentes precisiones y escalas:

Input Data	Specified As	Stored As
7,456,123.89	`NUMBER`	7456123.89
7,456,123.89	`NUMBER(*,1)`	7456123.9
7,456,123.89	`NUMBER(9)`	7456124
7,456,123.89	`NUMBER(9,2)`	7456123.89
7,456,123.89	`NUMBER(9,1)`	7456123.9
7,456,123.89	`NUMBER(6)`	How Scale Factors Affect Numeric Data Storage exceeds precision)
7,456,123.89	`NUMBER(7,-2)`	7456100

3.2.- NUMÉRICOS

Actual Data	Specified As	Stored As
123.89	`NUMBER`	123.89
123.89	`NUMBER(3)`	124
123.89	`NUMBER(3,2)`	exceeds precision
123.89	`NUMBER(4,2)`	exceeds precision
123.89	`NUMBER(5,2)`	123.89
123.89	`NUMBER(6,1)`	123.9
123.89	`NUMBER(6,-2)`	100
.01234	`NUMBER(4,5)`	.01234

3.2.- NUMÉRICOS

- **Conviene observar que**:
  - Precision=* permite indicar solo escala (*=38)
  - Si no se indica la escala se asume 0.
  - Si no se indica precisión se almacena la del dato introducido (la precisión máxima sería 38).
  - Si se excede la precisión se produce un error.
  - Si se excede la escala se produce un redondeo.
  - La escala puede ser negativa. El numero se redondea a la izquierda del punto decimal los dígitos indicados.
    - **NO LO USAMOS**
  - La escala puede ser mayor que la precisión. Sólo tiene sentido para números decimales menores que 1.
    - **NO LO USAMOS**

3.3.- FECHA

- **DATE**
  - Información de fecha/hora.
  - Almacena siglo/año/mes/día/hora/minutos/segundos
  - El formato se especifica mediante el parámetro de configuración de sesión: **NLS_DATE_FORMAT**
    - Normalmente suele ser: 'dd/mm/yy'
  - El formato se puede cambiar modificando el parámetro **NLS_DATE_FORMAT** con la orden ALTER SESSION
    - Los posibles formatos se recuerdan en prácticas.
- Existen otros tipos de datos para el manejo de fechas con mayor precisión.
  - Permiten fracciones de segundo. No los vamos a usar.
  - Consultadlos en las ayudas.

3.4.- DATOS BINARIOS

- **RAW (tamaño)**
  - Datos binarios.
  - Similar a VARCHAR2 pero para datos binarios (bytes) en lugar de caracteres
    - Tamaño máximo: 2000 bytes.
  - Se utiliza para almacenar gráficos, archivos de sonido, documentos, etc.
- **LONG RAW (en desuso)**
  - Similar a LONG pero para datos binarios de gran tamaño.
    - Tamaño máximo: 2GB
  - Está en desuso. Se mantiene por compatibilidad.

3.5.- TIPOS LOB (LARGE OBJECTS)

- **CLOB (Character Large Objects)**
  - Cadenas de caracteres de tamaño SUPER (hasta 4GB)
- **BLOB (Binary Large Objects)**
  - Objeto binario de tamaño SUPER (hasta 4GB)
- Oracle recomienda estos tipos frente a los LONG:
  - LONG, LONG RAW.
  - En desuso. Se mantienen por compatibilidad.
- Su utilización es un poco peculiar.

3.- TIPOS DE DATOS DE ORACLE

- **RESUMEN:**

	TIPO DE DATOS	TAMAÑO	OBSERVACIONES
CADENAS	CHAR (N)	2000B	Longitud fija
	VARCHAR2 (N)	4000B	Longitud variable
	~~LONG~~	~~2GB~~	~~Sólo Compatibilidad~~
	CLOB	4GB	
NUMERICOS	NUMBER [(P[,E])]	P=1..38	Enteros y reales
FECHAS	DATE		
BINARIOS	RAW (N)	2000B	
	~~LONG RAW~~	~~2GB~~	~~Sólo Compatibilidad~~
	BLOB	4GB	

3.- TIPOS DE DATOS DE ORACLE

- Es conveniente consultar la documentación sobre los tipos de datos en cada versión, ya que estos sufren revisiones (tamaño, etc.) :
  - Directamente en la documentación Web de Oracle:
    - http://docs.oracle.com
  - A través de buscador WEB (es a veces más rápido)
    - Buscad en Google: Oracle 11g Datatypes .
    - Consultad detenidamente el resultado.
- A partir de ahora indicaremos los tipos de datos Oracle en los esquemas, ya que vamos a trabajar con este SGBD.

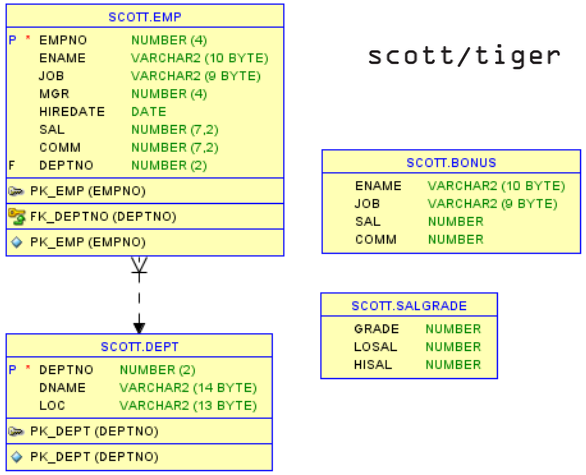
ÍNDICE

4.- ESQUEMAS LÓGICOS DE EJEMPLO

- **1.- TABLAS EMPLE Y DEPART:**
  (TRADUCCIÓN ESQUEMA EJEMPLO SCOTT)
  - Este mini esquema lógico de ejemplo lo utilizaremos en algunos ejemplos y prácticas del tema.
  - Son tablas obtenidas y traducidas del famoso esquema de usuario de Oracle SCOTT (scott/tiger)
  - Esquema de ejemplo de Oracle en desuso sustituido por los esquemas de ejemplo vistos en la práctica 1 del tema 3:
    - HR, OE, OC, PM, IX, SH
    - Estos esquemas los utilizaremos tal cual

4.- ESQUEMAS LÓGICOS DE EJEMPLO

ESQUEMA USUARIO SCOTT (visión grafica DM)



4.- ESQUEMAS LÓGICOS DE EJEMPLO

Who is Scott/Tiger?

Scott

Bruce Scott
(Oracle employee no. 4)

Tiger

Bruce's daughter's cat

4.- ESQUEMAS LÓGICOS DE EJEMPLO

1.- TABLAS EMPLE Y DEPART:

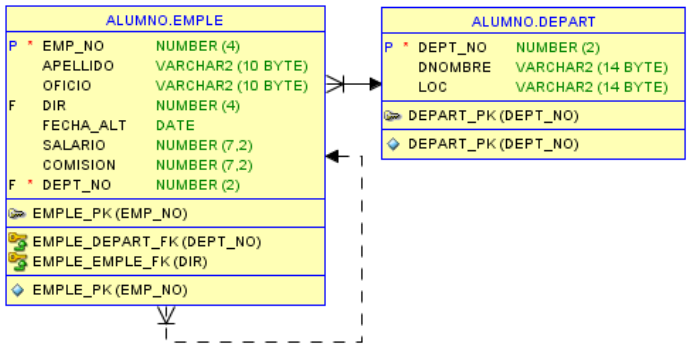
- **Esquema lógico (notación utilizada en clase)**

ESQUEMA LÓGICO
DEPART (dept_no: number (2), dnombre: varchar2 (14), loc: varchar2 (14)) CP= {dept_no}
EMPLE (emp_no: number (4), apellido: varchar2 (10), oficio: varchar2 (10), dir: number (4), fecha_alt: date, salario: number (7,2), comision: number (7,2), dept_no: number (2)) CP= {emp_no} CA= {dir} → EMPLE CA= {dept_no} → DEPART

4.- ESQUEMAS LÓGICOS DE EJEMPLO

1.- TABLAS EMPLE Y DEPART:

- **Visión gráfica DM**



4.- ESQUEMAS LÓGICOS DE EJEMPLO

Datos de ejemplo (ocurrencia esquema)

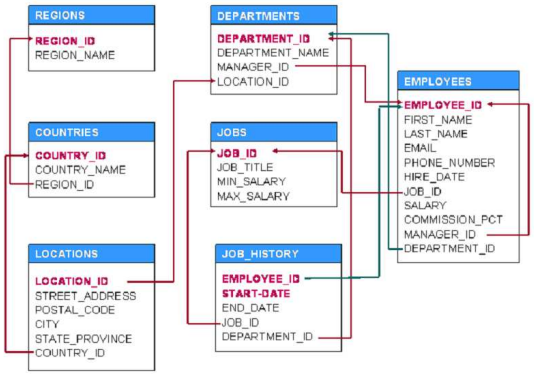
DEPT_NO	DNOMBRE	LOC
10	CONTABILIDAD	SEVILLA
20	INVESTIGACIÓN	MADRID
30	VENTAS	BARCELONA
40	PRODUCCIÓN	BILBAO

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

4.- ESQUEMAS LÓGICOS DE EJEMPLO

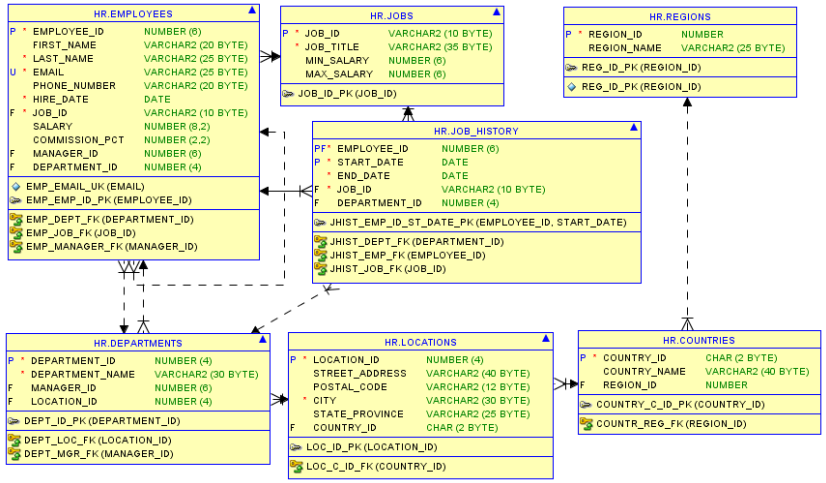
◎ 2.- ESQUEMA HR

- **HR (Human resources)**: esquema simple útil para trabajar los conceptos básicos de BD



4.- ESQUEMAS LÓGICOS DE EJEMPLO

◎ 2.- ESQUEMA HR (Visión grafica DM)



4.- ESQUEMAS LÓGICOS DE EJEMPLO

ESQUEMA LÓGICO NOTACIÓN CLASE (ESQUEMA USUARIO HR)
REGIONS (region_id, region_name) CP= {region_id}
COUNTRIES (country_id, country_name, region_id) CP= {country_id} CA= {region_id}→REGIONS
LOCATIONS (location_id, street_address, postal_code, city, state_province, country_id) CP= {location_id} VNN= {city} CA= {country_id}→COUNTRIES
JOBS (job_id, job_title, min_salary, max_salary) CP= {job_id} VNN= {job_title}
EMPLOYEES (employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary, commission_pct, manager_id, department_id) CP= {employee_id} VNN= {last_name, email, hire_date, job_id} CAlt= {email} CA= {department_id} →DEPARTMENTS CA= {job_id} →JOBS CA= {manager_id} →EMPLOYEES
DEPARTMENTS (department_id, department_name, manager_id, location_id) CP={department_id} VNN= {department_name} CA={manager_id} →EMPLOYEES CA={location_id} →LOCATIONS
JOB_HISTORY (employee_id, start_date, end_date, job_id, department_id) CP={employee_id, start_date} VNN={end_date, job_id} CA={employee_id} →EMPLOYEES CA={job_id} →JOBS CA={department_id} →DEPARTMENTS

ÍNDICE

5.- SINTAXIS DEL LENGUAJE

- La notación que se va a utilizar para presentar el lenguaje SQL es una versión extendida de BNF*:
  - PALABRAS EN MAYÚSCULA: indican palabras reservadas de SQL.
  - *Cadenas en minúscula cursiva*: indican un elemento sintáctico que se define posteriormente.
  - Cadenas en minúscula (sin cursiva): indican un elemento sintáctico terminal (que no se define posteriormente).
  - La barra vertical | separa distintas alternativas de entre las que debe elegirse solo una .
  - Los corchetes [] indican elementos sintácticos opcionales.
  - Las llaves {} indican elementos sintácticos obligatorios.
  - Si xyz es un elemento sintáctico, comalista_xyz indica que el elemento xyz puede repetirse las veces que se necesite mediando una coma entre cada par de elementos.
  - Si xyz es un elemento sintáctico, lista_xyz indica que el elemento xyz puede repetirse las veces que se necesite mediando al menos un espacio entre cada par de elementos.

*Backus-Naur Form

5.- SINTAXIS DEL LENGUAJE

- **Por ejemplo: (Sintaxis sentencia SELECT)**

```
SELECT [ALL|DISTINCT]
        {comalista_ref_columna | *}
FROM   comalista_ref_tabla
[WHERE condicion]
[ORDER BY comalista_ref_columna [DESC|ASC]]
```

```
SELECT [ALL|DISTINCT]
        {ref_columna1, ref_columna2, ref_columna3, … | *}
FROM   ref_tabla1, ref_tabla2, ref_tabla3…
[WHERE condicion]
[ORDER BY ref_columna1, ref_columna2, … [DESC|ASC]]
```

```
ref_columna::= atributo|
               constante|
               expresion_aritmetica|
               ref_funcion
```

```
ref_tabla::= tabla [alias]
```


5.- SINTAXIS DEL LENGUAJE

- **Por ejemplo: (Sintaxis sentencia SELECT)**

Basándonos en el esquema lógico de EMPLE y DEPART:

- Sentencias SELECT BIEN construidas sintácticamente:

SELECT * FROM DEPART	SELECT dept_no FROM DEPART
SELECT dept_no, dnombre FROM DEPART	SELECT * FROM DEPARTAMENTO, EMPLE
SELECT DISTINCT dnombre FROM DEPART	SELECT * FROM DEPART ORDER BY dnombre

- Sentencias SELECT MAL construidas sintácticamente:

SELECT *	SELECT cod_dep, FROM DEPART
SELECT FROM DEPART	FROM DEPART, EMPLE
SELECT * FROM DEPART WHERE	SELECT * FROM ORDER BY dnombre

ÍNDICE

6.- DML: CONSULTA DE DATOS

Introducción a la sentencia SELECT

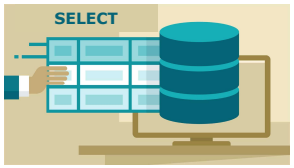
6.1.- Sintaxis básica para consultas simples.

6.2.- Comparación de cadenas.

6.3.- Comparación de valores nulos.

6.4.- Comparación con conjuntos de valores.

6.5.- Comparación con rangos de valores.



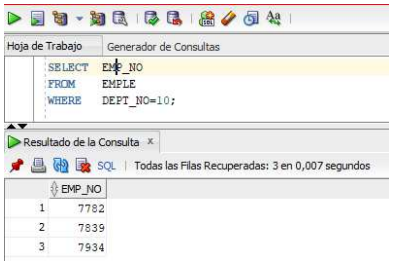
6.1.- Sintaxis básica consultas simples

- La sentencia SELECT se utiliza para obtener datos de una BD relacional (consultar datos)
- **Por ejemplo**: *consultando la tabla EMPLE*
  - *"Obtener el Nº de los empleados del departamento 10"*

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

6.1.- Sintaxis básica consultas simples

- Una sentencia SELECT siempre dará como resultado una **tabla temporal** con los datos consultados.
  - El resultado de la consulta anterior será una tabla temporal que contiene:
    - Una sola columna (EMP_NO)
    - Las filas de la tabla EMPLE con nº de departamento (DEPT_NO) 10.



6.1.- Sintaxis básica consultas simples

- Al realizar una sentencia SELECT se debe decidir:
  1. TABLAS que contienen la información a obtener.
  2. Características o CONDICIONES que debe cumplir la información a obtener de las tablas especificadas.
  3. Información de las tablas especificadas que se desea seleccionar o MOSTRAR en la tabla resultado.
  4. ORDENACIÓN deseada en el resultado.

```
3 SELECT [ALL|DISTINCT]
          {comalista_ref_columna | *}
1 FROM   comalista_ref_tabla
2 [WHERE condicion]
4 [ORDER BY comalista_ref_columna [DESC|ASC]]
```

6.1.- Sintaxis básica consultas simples

SINTAXIS BÁSICA:

```
3 SELECT [ALL|DISTINCT]
         {comalista_ref_columna | *}
1 FROM   comalista_ref_tabla
2 [WHERE condicion]
4 [ORDER BY comalista_ref_columna [DESC|ASC]]
```

- FROM...: especifica de que tabla (o lista de tablas) se recuperará la información

```
ref_tabla::= tabla [alias]
```

- SELECT...: especifica la información a obtener de la BD (información final a mostrar en la consulta resultado)

```
ref_columna::= atributo|
               constante|
               expresion_aritmetica|
               ref_funcion
```


6.1.- Sintaxis básica consultas simples

Ejemplos SELECT/FROM:

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT *
FROM   EMPLE
```

```
SELECT OFICIO
FROM   EMPLE
```

```
SELECT EMP_NO, SALARIO
FROM   EMPLE E
```

- * indica que se desean seleccionar todas las columnas de la tabla (o lista de tablas) especificadas en la clausula FROM.

6.1.- Sintaxis básica consultas simples

- WHERE...: expresa una condición que deben cumplir las filas a obtener (se comprueba por cada fila)

```
condicion::= comparacion|
             condicion AND condicion|
             condicion OR condicion|
             NOT condicion
```

```
comparacion::= comparacion_operador |
               comparacion_like |
               comparacion_null |
               comparacion_in |
               comparacion_between |
               comparacion_all_any |
               comparacion_exists
```

```
comparacion_operador::= operando {=|>|>=|<|<=|!=|<>} operando
```

```
operando::= ref_columna | subconsulta
```


6.1.- Sintaxis básica consultas simples

Ejemplos WHERE:

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT *
FROM   EMPLE
WHERE  SALARIO>=1000
```

```
SELECT *
FROM   EMPLE
WHERE  SALARIO>=1000 AND
       DEPT_NO=20
```

```
SELECT SALARIO
FROM   EMPLE E
WHERE  OFICIO != 'DIRECTOR'
```

6.1.- Sintaxis básica consultas simples

◎ COMPARACIÓN DE VALORES NULOS:

comparacion_operador::= *operando* {=|>|>=|<|<=|!=|<>} *operando*

- Las comparaciones entre cualquier valor y el valor NULO (NULL) dan como resultado INDEFINIDO (lógica trivaluada).
- Las filas que evalúan la condición a indefinido NO se incluyen en la selección (**Indefinido no es verdadero**).
- **Ejemplo**:

```
SELECT      *
FROM        tabla
WHERE       atributo1< atributo2
```

atributo1	atributo2
1	10
13	NULL
NULL	1
NULL	NULL

INDEFINIDO

6.1.- Sintaxis básica consultas simples

◎ OPERADORES DE COMPARACIÓN Y LÓGICOS:

COMPARACIÓN

=	Igual
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
!= <>	Distinto

LÓGICOS

AND
OR
NOT

- Se utilizan para formar condiciones (diapositiva 48).
- Conviene NO olvidar la lógica TRIVALUADA.

G	H= NOT G
Falso	Cierto
Indefinido	Indefinido
Cierto	Falso

G	H	F= G AND H	F=G OR H
Falso	Falso	Falso	Falso
Indefinido	Falso	Falso	Indefinido
Cierto	Falso	Falso	Cierto
Falso	Indefinido	Falso	Indefinido
Indefinido	Indefinido	Indefinido	Indefinido
Cierto	Indefinido	Indefinido	Cierto
Falso	Cierto	Falso	Cierto
Indefinido	Cierto	Indefinido	Cierto
Cierto	Cierto	Cierto	Cierto

Falso AND X = Falso
Cierto AND X = X
Cierto OR X = Cierto
Falso OR X = X
Indefinido AND/OR Indefinido= Indefinido

6.1.- Sintaxis básica consultas simples

◎ OPERADORES DE COMPARACIÓN Y LÓGICOS:

- **Combinación de operadores lógicos:**
  - ◆Conviene tener en cuenta el orden de precedencia o prioridad de los operadores lógicos.

Orden de prioridad

NOT
AND
OR

  - ◆Los paréntesis rompen el orden de precedencia.
  - ◆Son necesarios en determinadas ocasiones, sino los resultados pueden ser indeseados.

6.1.- Sintaxis básica consultas simples

◎ OPERADORES DE COMPARACIÓN Y LÓGICOS:

- Ejemplo de uso OBLIGATORIO de paréntesis:
  - ◆*Obtener el apellido, salario y nº de departamento de los empleados cuyo salario sea mayor de 2000 Euros pertenecientes a los departamentos 10 o 20.*

```
SELECT      APELLIDO, SALARIO, DEPT_NO
FROM        EMPLE
WHERE       SALARIO>2000 AND
            DEPT_NO=10 OR
            DEPT_NO=20
```

```
SELECT      APELLIDO, SALARIO, DEPT_NO
FROM        EMPLE
WHERE       SALARIO>2000 AND
            (DEPT_NO=10 OR
            DEPT_NO=20)
```

6.1.- Sintaxis básica consultas simples

- ORDER BY: especifica un criterio de ordenación para el resultado de la consulta.
  - **ASC**: ordenación ascendente (POR DEFECTO)
  - **DESC**: ordenación descendente.
- [ALL | DISTINCT]
  - **ALL**: recupera todas las filas, aunque estén repetidas
    - Opción por defecto.
  - **DISTINCT**: elimina las filas repetidas.

6.1.- Sintaxis básica consultas simples

- **Ejemplos ORDER BY/DISTINCT:**

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT    *
FROM      EMPLE E
ORDER BY  SALARIO DESC
```

```
SELECT    DISTINCT OFICIO
FROM      EMPLE
```

```
SELECT    DISTINCT DEPT_NO
FROM      EMPLE
ORDER BY  DEPT_NO
```

```
SELECT    EMP_NO, SALARIO
FROM      EMPLE
WHERE     SALARIO<1000
ORDER BY  SALARIO ASC
```


6.1.- Sintaxis básica consultas simples

- **ALIAS DE COLUMNAS**:
  - Cuando se realizan consultas a la BD, los nombres de las columnas seleccionadas se muestran como cabeceras de presentación de la tabla resultado.
  - Existe la posibilidad de crear un nombre alternativo (alias) para una columna en una consulta SQL
    - Sólo para visualizarlo, el nombre de columna no cambia.
    - Se indica a la derecha de columna.
    - Puede ir o no entre comillas dobles (si hay espacios).

```
SELECT    EMP_NO "Nº DE EMPLEADO", DIR JEFE
FROM      EMPLE
ORDER BY  SALARIO
```


6.1.- Sintaxis básica consultas simples

- **EJEMPLOS:**
  - Analiza el resultado de las siguientes sentencias

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT    EMP_NO "Nº", APELLIDO
FROM      EMPLE
WHERE     DIR=7698
ORDER BY  APELLIDO
```

```
SELECT    APELLIDO, OFICIO
FROM      EMPLE
WHERE     SALARIO<2000
ORDER BY  DEPT_NO,  SALARIO
```

6.1.- Sintaxis básica consultas simples

- **OPERADORES ARITMÉTICOS**:
  - Se utilizan para formar expresiones aritméticas

+
-
*
/

```
ref_columna::=   atributo|
                 constante|
                 expresion_aritmetica|
                 ref_funcion
```

```
expresion_aritmetica::= [+| -] ref_columna |
                        ref_columna [+| -| *| /] ref_columna
```


6.1.- Sintaxis básica consultas simples

- **Ejemplos OPERADORES ARITMETICOS:**

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT      EMP_NO, SALARIO, SALARIO+COMISION "SALARIO TOTAL"
FROM        EMPLE
WHERE       FECHA_ALT >='01/01/1980' AND
            FECHA_ALT<='31/12/1980'
ORDER BY    FECHA_ALT
```


6.1.- Sintaxis básica consultas simples

- **Resumen final (funcionamiento SELECT).**

```
3 SELECT [ALL|DISTINCT]
         {comalista_ref_columna | *}
1 FROM   comalista_ref_tabla
2 [WHERE condicion]
4 [ORDER BY comalista_ref_columna [DESC|ASC]]
```

- **Selección de columnas**
  - Se realiza en la parte SELECT.

```
SELECT [ALL|DISTINCT]
       [comalista_ref_columna | *]
```

- **Selección de filas:**
  - Se realiza en la parte WHERE.

```
[WHERE condicion]
```


6.- DML: CONSULTA DE DATOS

Introducción a la sentencia SELECT

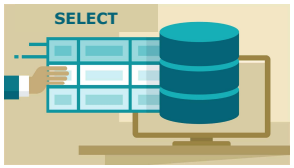
6.1.- Sintaxis básica para consultas simples.

6.2.- Comparación de cadenas.

6.3.- Comparación de valores nulos.

6.4.- Comparación con conjuntos de valores.

6.5.- Comparación con rangos de valores.



6.2.- Comparación de cadenas

- *comparacion_like*

```
comparacion::=      comparacion_operador |
                    comparacion_like |
                    comparacion_null |
                    comparacion_in |
                    comparacion_between |
                    comparacion_all_any |
                    comparacion_exists
```

comparacion_like::= atributo [NOT] LIKE 'cadena_comparacion'

- Permite comprobar si un atributo de tipo cadena sigue un patrón determinado
  - cadena_comparacion puede contener los siguientes caracteres especiales (comodín):
    - '%': representa cualquier secuencia de caracteres
    - '_': representa cualquier carácter individual

6.2.- Comparación de cadenas

- **EJEMPLOS:**

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO LIKE 'A%'
```

```
SELECT  *
FROM    EMPLE
WHERE   OFICIO LIKE 'A_A'
```

```
SELECT  *
FROM    EMPLE
WHERE   OFICIO LIKE '%R'
```

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO LIKE 'GIL'
```


6.2.- Comparación de cadenas

- **CUIDADO!!**
- Si buscamos coincidencia exacta de cadenas os recomiendo utilizar operador de comparación en lugar de like

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO = 'GIL'
```

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO LIKE 'GIL'
```

- LIKE se utiliza principalmente con caracteres comodín:

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO LIKE 'A%'
```

- Los caracteres comodín **NO** pueden utilizarse con los operadores de comparación

```
SELECT  *
FROM    EMPLE
WHERE   APELLIDO = 'A%'
```


6.2.- Comparación de cadenas

- *comparacion_like*
- Si deseamos "escapar" el significado de los caracteres comodín y buscarlos como caracteres literales en cadenas debemos utilizar un **carácter de escape**
- En SQL estándar se establece un carácter de escape para una *comparación_like* añadiendo:

comparacion_like::= atributo [NOT] LIKE 'cadena_comparacion' [**ESCAPE 'caracter']**

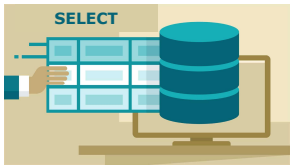
- Indica que 'caracter' se utiliza como carácter de escape en 'cadena_comparacion'
- Es frecuente utilizar la contrabarra '\' como carácter de escape.
  - La mayoría de los SGBD lo utilizan por defecto.

```
SELECT APELLIDO
FROM   EMPLE
WHERE  APELLIDO LIKE '%\%%' ESCAPE '\'
```

→ QUE OBTIENE?

6.- DML: CONSULTA DE DATOS

Introducción a la sentencia SELECT



6.3.- Comparación de valores nulos

◎ *comparacion_null*

comparacion::= *comparacion_operador* |
comparacion_like |
comparacion_null | ←
comparacion_in |
comparacion_between |
comparacion_all_any |
comparacion_exists

comparacion_null::= atributo IS [NOT] NULL

- Permite comprobar si un atributo es nulo
  - El valor nulo en SQL es NULL.
  - No se pueden utilizar las comparaciones normales para comprobar este valor

~~Atributo = NULL~~ → ¡¡¡INDEFINIDO!!!

6.3.- Comparación de valores nulos

◎ **EJEMPLOS:**

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

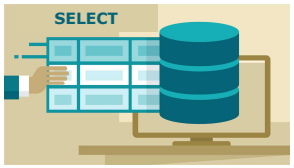
```
SELECT    *
FROM      EMPLE
WHERE     COMISION IS NOT NULL
```

```
SELECT    *
FROM      EMPLE
WHERE     DIR IS NULL
```


6.- DML: CONSULTA DE DATOS

Introducción a la sentencia SELECT

6.1.- Sintaxis básica para consultas simples.
6.2.- Comparación de cadenas.
6.3.- Comparación de valores nulos.
6.4.- Comparación con conjuntos de valores.
6.5.- Comparación con rangos de valores.



6.4.- Comparación con conjuntos de valores

◎ *comparacion_in*

comparacion::= *comparacion_operador* |
comparacion_like |
comparacion_null |
comparacion_in | ←
comparacion_between |
comparacion_all_any |
comparacion_exists

comparacion_in::= *operando* [NOT] IN {(comalista_valores) | *subconsulta*}

- Permite comprobar si un *operando* se encuentra dentro de un conjunto de valores.
  - ◆X IN (a, b, ...,z) ≡ (X=a) OR (X=b) OR...OR (X=z)
  - ◆X NOT IN (a, b, ...,z) ≡ NOT (X IN (a, b, ...,z))
- El tipo de los valores debe coincidir con el de *operando*

6.4.- Comparación con conjuntos de valores

◎ **EJEMPLOS:**

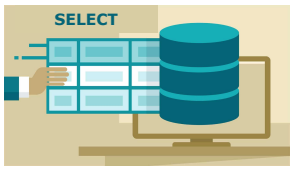
EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT    *
FROM      EMPLE
WHERE     DEPT_NO IN (10,20)
```

```
SELECT    *
FROM      EMPLE
WHERE     OFICIO NOT IN ('PRESIDENTE', 'DIRECTOR')
```


6.- DML: CONSULTA DE DATOS

Introducción a la sentencia SELECT



6.5.- Comparación con rangos de valores

◎ *comparacion_between*

comparacion::= *comparacion_operador* |
comparacion_like |
comparacion_null |
comparacion_in |
comparacion_between | ←
comparacion_all_any |
comparacion_exists

comparacion_between::= *operando* [NOT] BETWEEN *operando* AND *operando*

Valor inicial — Valor final

- Permite comprobar si un *operando* está comprendido dentro de un rango.
  - ◆Y BETWEEN X AND Z ≡ (X<=Y) AND (Y<=Z)
  - ◆Y NOT BETWEEN X AND Z ≡ NOT (Y BETWEEN X AND Z)

6.5.- Comparación con rangos de valores

EJEMPLOS:

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/1980	1040	(null)	20
7499	ARROYO	VENDEDOR	7698	20/02/1980	2080	390	30
7521	SALA	VENDEDOR	7698	22/02/1981	1625	650	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/1981	3867	(null)	20
7654	MARTÍN	VENDEDOR	7698	29/09/1981	1625	1820	30
7698	NEGRO	DIRECTOR	7839	01/05/1981	3705	(null)	30
7782	CEREZO	DIRECTOR	7839	09/06/1981	3185	(null)	10
7788	GIL	ANALISTA	7566	09/11/1981	3900	(null)	20
7839	REY	PRESIDENTE	(null)	17/11/1981	6500	(null)	10
7844	TOVAR	VENDEDOR	7698	08/09/1981	1950	0	30
7876	ALONSO	EMPLEADO	7788	23/09/1981	1430	(null)	20
7900	JIMENO	EMPLEADO	7698	03/12/1981	1235	(null)	30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/1981	3900	(null)	20
7934	MUÑOZ	EMPLEADO	7782	23/01/1982	1690	(null)	10

```
SELECT    *
FROM      EMPLE
WHERE     SALARIO BETWEEN 1500 AND 2000
```

```
SELECT    *
FROM      EMPLE
WHERE     SALARIO NOT BETWEEN 1500 AND 2000
```