

# SQL

(Cap 8 - Elmasri 5<sup>a</sup> edición)

# Structured Query Language – SQL

- ♦ **Lenguaje de “programación”** para SGBDs (DBMSs)
  - **DDL**: Data definition language: creación del modelo de datos (diseño de tablas)
  - **DML**: Data manipulation language: inserción, modificación, eliminación de datos
  - **DQL**: Data query language: consultas
- ♦ El SQL se puede ejecutar sobre un SGBD (DBMS).
- ♦ El SQL facilita la migración entre SGBDs (DBMSs) y por eso su éxito comercial.
- ♦ Así proporciona un “interfaz” común entre los diferentes SGDBs (DBMSs).
- ♦ **Algebra relacional**: conjunto de operaciones que describen paso a paso como computar una respuesta sobre las relaciones en una BD (modelo relacional).
- ♦ **Cálculo relacional**: es un lenguaje de consulta, sobre relaciones, describiendo la respuesta deseada sobre una BD (no se especifica como obtenerla).

# Structured Query Language – SQL

- ♦ El **SQL** proporciona una interfaz de **lenguaje declarativo** (especifica lo que debe ser el resultado) de más alto nivel que puramente una consulta en álgebra relacional.
- ♦ Dejando así al **SGBD** (DBMS) las **decisiones de optimización** y de cómo se debe realizar la consulta.
- ♦ Aunque el SQL incluye algunas características de álgebra relacional, está muy basado en el cálculo relacional de tuplas.
- ♦ ¿Por qué no se utiliza cálculo?
- ♦ La ventaja del SQL es que la sintaxis es mucho más **amigable**.
- ♦ Leer el Capítulo 8 del libro:
  - Fundamentos de sistemas de bases de datos. Ramez Elmasri, Shamkant Navathe. Pearson Addison Wesley, 2007. INF/681.31.65/ELM.

# Structured Query Language – SQL

- ♦ El estándar más utilizado
  - Creado en 1974 (D. D. Chamberlin & R. F. Boyce, IBM)
  - ANSI en 1986, ISO en 1987
  - Core (todos los SGBD) + packages (modulos opcionales)
- ♦ Versiones
  - SQL1 – SQL 86
  - SQL2 – SQL 92, SQL 99
  - SQL 3 – no plenamente soportado por la industria
- ♦ Limitaciones
  - No es puramente relacional (p.e. las *vistas* son *multiconjuntos* de tuplas)
  - Importantes divergencias entre implementaciones (no es directamente portable en general, incompletitudes, extensiones) –uno termina aprendiendo variantes de SQL

# Structured Query Language – SQL

- ♦ Algunos SGBDs libres:

- PostgreSQL (<http://www.postgresql.org> Postgresql) Licencia BSD
- SQLite (<http://www.sqlite.org> SQLite) Licencia Dominio Público
- DB2 Express-C (<http://www.ibm.com/software/data/db2/express/>)
- Apache Derby (<http://db.apache.org/derby/>)
- MySQL (<http://dev.mysql.com/>)
- .....

- ♦ Algunos SGBDs no libres:

- MySQL: Licencia Dual, depende del uso
- dBase
- Fox Pro
- IBM DB2: Universal Database (DB2 UDB)
- Microsoft SQL Server
- Oracle
- .....

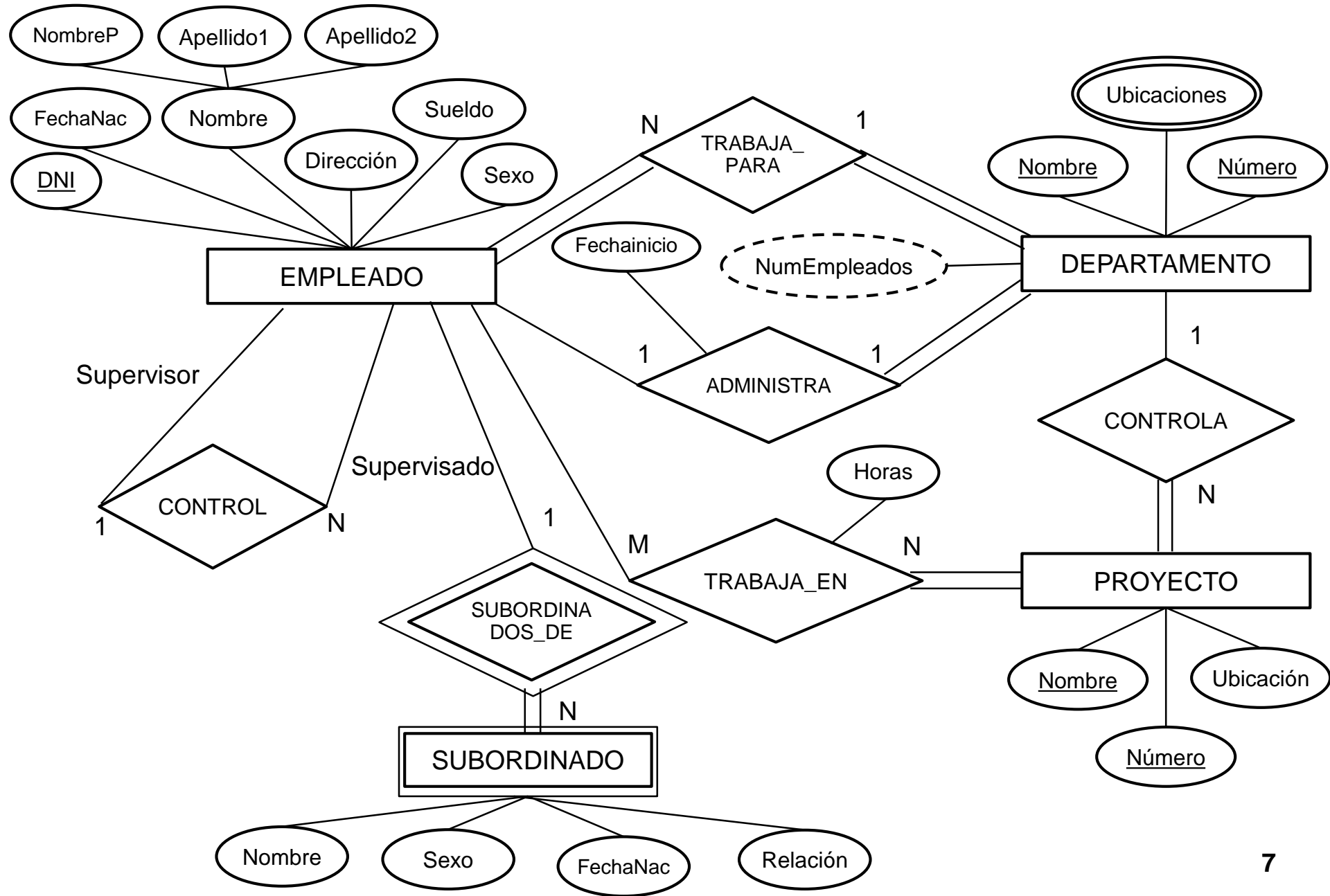
- ♦ Algunos SGBDs no libres y gratuitos:

- Microsoft SQL Server Compact Edition Basica
- Oracle Express Edition 10 (solo corre en un servidor, capacidad limitada)
- .....

# Elementos fundamentales de una base de datos SQL

- ♦ Base de datos = conjunto de tablas **RELACIONADAS**
- ♦ Tabla (relación, entidad, esquema...) =
  - Estructura fija de campos (esquema)
  - Conjunto de registros con valores de campos
- ♦ Campo (atributo, propiedad, “columna”), tiene un tipo de dato
- ♦ Registro (tupla, “fila”)
- ♦ Clave primaria
- ♦ Clave secundaria
- ♦ Clave externa

# Diagramas típicos para una BD (modelo E/R)



# Campos, Tuplas y Tablas en una BD

EMPLEADO

▲	Nombre text	Apellido1 text	Apellido2 text	<u>Dni</u> integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Jose	Perez	Perez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
2	Alberto	Campos	Sastre	333445555	1955-12-08	Avda Rios, 9	H	40000	888665555	5
3	Alicia	Jimenez	Celaya	999887777	1968-05-12	Gran Via, 38	M	25000	987654321	4
4	Juana	Sainz	Oreja	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
5	Eduardo	Ochoa	Paredes	888665555	1937-11-10	Las Peñas, 1	H	55000	[null]	1
6	Fernan...	Ojeda	Ordoñez	666884444	1962-09-15	Portillo, S/N	M	38000	333445555	5
7	Luis	Pajares	Morera	987987987	1969-03-29	Enebros, 90	H	25000	987654321	4
8	Aurora	Oliva	Avezuela	453453453	1972-07-31	Anton, 6	M	25000	333445555	5

TRABAJA\_EN

▲	<u>DniEmpleado</u> integer	<u>NumProy</u> integer	Horas numeric
1	123456789	1	32.5
2	123456789	2	7.5
3	666884444	3	40.0
4	453453453	1	20.0
5	453453453	2	20.0
6	333445555	2	10.0
7	333445555	10	10.0
8	333445555	3	10.0
9	333445555	20	10.0
10	999887777	30	30.0
11	999887777	10	10.0
12	987987987	10	35.0
13	987987987	30	5.0
14	987654321	30	20.0
15	987654321	20	15.0
16	888665555	20	[null]

PROYECTO

▲	NombreProyecto text	<u>NumProyecto</u> integer	UbicacionProyecto text	NumDptoProyecto integer
1	PruductoX	1	Valencia	5
2	ProductoY	2	Sevilla	5
3	ProductoZ	3	Madrid	5
4	Computacion	10	Gijon	4
5	Reorganizacion	20	Madrid	1
6	Comunicaciones	30	Gijon	4

LOCALIZACIONES\_DPTO

▲	<u>NumeroDpto</u> integer	<u>UbicacionDpto</u> text
1	1	Madrid
2	4	Gijon
3	5	Valencia
4	5	Sevilla
5	5	Madrid

SUBORDINADO

▲	<u>DniEmpleado</u> integer	<u>NombSubordinado</u> text	Sexo "char"	FechaNac date	Relacion text
1	333445555	Alicia	M	1986-04-05	Hija
2	333445555	Teodoro	H	1983-10-25	Hijo
3	333445555	Luisa	M	1958-05-03	Esposa
4	987654321	Alfonso	H	1942-01-28	Esposo
5	123456789	Miguel	H	1988-01-04	Hijo
6	123456789	Alicia	M	1988-12-30	Hija
7	123456789	Elisa	M	1967-05-05	Esposa

DEPARTAMENTO

▲	NombreDpto text	<u>NumeroDpto</u> integer	DniDirector integer	FechaIngresoDirector date
1	Investigacion	5	333445555	1988-05-22
2	Administracion	4	987654321	1995-01-01
3	Sede Central	1	888665555	1981-06-19



# Campos, Tuplas y Tablas en una BD

## EMPLEADO

Nombre	Apellido1	Apellido2	<u>Dni</u>	FechaNac	Direccion	Sexo	Sueldo	SuperDni	Dno
text	text	text	integer	date	text	"char"	numeric	integer	integer

## DEPARTAMENTO

NombreDpto	<u>NumeroDpto</u>	<u>DniDirector</u>	FechaIngresoDirector
text	integer	integer	date

## LOCALIZACIONES\_DPTO

<u>NumeroDpto</u>	<u>UbicacionDpto</u>
integer	text

## PROYECTO

NombreProyecto	<u>NumProyecto</u>	UbicacionProyecto	NumDptoProyecto
text	integer	text	integer

## TRABAJA\_EN

<u>DniEmpleado</u>	<u>NumProy</u>	Horas
integer	integer	numeric

## SUBORDINADO

<u>DniEmpleado</u>	<u>NombSubordinado</u>	Sexo	FechaNac	Relacion
integer	text	"char"	date	text

# Estructura léxica del lenguaje

## Operaciones SQL

- ♦ DDL – Creación, diseño, eliminación de tablas
- ♦ DML – Inserción, modificación, eliminación de registros
- ♦ DQL – Consulta

## Estructura léxica de SQL

- ♦ Case-insensitive, insignificant whitespace
- ♦ Sentencias, expresiones, valores, tipos de datos
- ♦ Referencias
  - Elmasri cap. 8
  - PostgreSQL SQL ref: <https://www.postgresql.org/docs/9.6/static/index.html>

# Esquemas y Catálogos en SQL

Versiones antiguas de SQL no incluían estos conceptos

**CREATE SCHEMA** *nombre* **AUTHORIZATION** *propietarios*

Definiciones de Dominios

Definiciones de Tablas

Definiciones de Vistas

.....

**CATÁLOGO** = conjunto de esquemas bajo un nombre

- Siempre incluye un esquema (INFORMATION\_SCHEMA) que contiene información sobre todos los elementos del catálogo
- Restricciones de Integridad: solo se pueden definir entre esquemas del mismo catálogo
- El estándar no proporciona mecanismos para definir y eliminar catálogos, depende de la implementación

# Literales, Expresiones y Operadores

## Valores literales

Cadenas de caracteres entre '...'

Valores numéricos similar p.e. al lenguaje C

## Expresiones

Se pueden utilizar en WHERE, SELECT, SET, DEFAULT, CHECK...

## Operadores

+ - \* / % ^

AND OR NOT

= < > <= >= LIKE ISNULL

operaciones con *strings*: concatenación, like, expresiones regulares ('%'  
'\_')

## Comentarios

--

/\* ... \*/

# Tipos de Datos y Dominios en SQL

## **Numéricos:**

- Entero de distintos tamaños (INTRGER o INT, SMALLINT)
- Reales de distinta precisión (FLOAT o REAL, DOUBLE PRECISION)
- Números con formato (NUMERIC(t,d), DECIMAL(t,d), d menor t, d=0 por defecto, t=n dig, d= n dig a la derecha del punto)

## **Cadenas de caracteres:**

- CHAR (n) – longitud fija con relleno a blancos
- CHAR VARYING (n) - longitud variable con límite
- TEXT - longitud variable sin límite

## **Cadenas de bits:**

- BIT(n)
- BIT VARYING (n)

# Tipos de Datos y Dominios en SQL

## Fecha y Hora:

- DATE (10 posiciones), YYYY-MM-DD
- TIME (8 posiciones), HH:MM:MM
- TIME (i), carácter separador más i posiciones para fracciones de segundo
- WITH TIME ZONE, 6 posiciones extra para el desplazamiento respecto al uso horario estándar universal, +-HH.MM
- TIMESTAMP YYYY-MM-DD HH.MM.SS.fracciones de segundo (6 posiciones), es opcional el calificador WITH TIME ZONE
- INTERVAL, periodo de tiempo (están cualificado para ser de dos tipos de intervalos generalmente)
  - AÑO/MES
  - DIA/HORA

# Tipos de Datos y Dominios en SQL

***Dominios***: Alternativamente a especificar los datos directamente se pueden crear dominios.

Ejemplo: CREATE DOMAIN Tipo\_Dni AS CHAR(9)

**CREATE DOMAIN** *nombre* [**AS**] *tipo\_datos*

**[Definición\_por\_defecto]**

**[Restricciones]**

*Definición\_por\_defecto*

**DEFAULT** {literal|función|NULL}

*Restricciones*

**[CONSTRAINT nombre ]**

CHECK (expresión\_condicional)

# Tipos de Datos y Dominios en SQL

*Ejemplos:*

```
CREATE DOMAIN CIUDADES CHAR(15)
```

```
    DEFAULT 'Madrid'
```

```
    CONSTRAINT MirestricciondeCiudades
```

```
        CHECK (VALUE IN('Atenas', 'Dublin',....., 'Madrid'))
```

```
CREATE DOMAIN NumEmp NUMERIC(4)
```

```
    DEFAULT 0
```

```
    CHECK (VALUE IN NOT NULL)
```



# TABLAS

```
CREATE TABLE nombre (  
    campo1 tipo1 [restricciones1],  
    campo2 tipo2 [restricciones2],  
    ...,  
    [restricciones ]  
);
```

```
ALTER TABLE nombre ADD COLUMN campo tipo [restricciones];  
ALTER TABLE nombre ADD restricción;  
ALTER TABLE nombre DROP COLUMN campo;  
  
DROP TABLE nombre;  
DROP CONSTRAINT nombre-restricción;
```

# TABLAS

♦ Las tablas pueden tener más opciones, como indicadores de comportamiento en ciertas acciones, o relaciones con otras tablas.

♦ Las restricciones de la tabla pueden ser de diferentes tipos:

- PRIMARY KEY (lista de columnas) (NO permite valores nulos, NULL)
- UNIQUE (lista de columnas) (permite valores nulos, NULL)
- FOREIGN KEY (lista de columnas) REFERENCES Tabla(columnas) (Indica que esa lista de columnas son clave primaria de otra tabla)

♦ Las restricciones referenciales de la tabla pueden ir con en el borrado o actualización

- ON DELETE
  - NO ACTION
  - SET DEFAULT
  - SET NULL
  - CASCADE
- ON UPDATE
  - NO ACTION
  - SET DEFAULT
  - SET NULL
  - CASCADE

# Crear Tablas: Ejemplos

## CREATE TABLE EMPLEADO

```
( Nombre          VARCHAR(15)  NOT NULL,
  Apellido1        CHAR,
  Apellido2        VARCHAR(15)  NOT NULL,
  Dni              CHAR(9)      NOT NULL,
  FechaNac        DATE,
  Dirección        VARCHAR(30),
  Sexo            CHAR,
  Sueldo          DECIMAL(10,2),
  SuperDni        CHAR(9),
  Dno             INT          NOT NULL,
  PRIMARY KEY (Dni),
  FOREIGN KEY(SuperDni) REFERENCES EMPLEADO(Dni),
  FOREIGN KEY(Dno) REFERENCES DEPARTAMENTO(NúmeroDpto) );
```

## CREATE TABLE DEPARTAMENTO

```
( NombreDpto      VARCHAR(15)  NOT NULL,
  NúmeroDpto      INT          NOT NULL,
  DniDirector     CHAR(9)      NOT NULL,
  FechaIngresoDirector DATE,
  PRIMARY KEY(NúmeroDpto),
  UNIQUE(NombreDpto),
  FOREIGN KEY(DniDirector) REFERENCES EMPLEADO(Dni) );
```

## CREATE TABLE LOCALIZACIONES\_DPTO

```
( NúmeroDpto      INT          NOT NULL,
  UbicaciónDpto   VARCHAR(15)  NOT NULL,
  PRIMARY KEY(NúmeroDpto, UbicaciónDpto),
  FOREIGN KEY(NúmeroDpto) REFERENCES DEPARTAMENTO(NúmeroDpto) );
```

## CREATE TABLE PROYECTO

```
( NombreProyecto  VARCHAR(15)  NOT NULL,
  NumProyecto     INT          NOT NULL,
  UbicaciónProyecto VARCHAR(15),
  NumDptoProyecto INT          NOT NULL,
  PRIMARY KEY(NumProyecto),
  UNIQUE(NombreProyecto),
  FOREIGN KEY(NumDptoProyecto) REFERENCES DEPARTAMENTO(NúmeroDpto)
```

## CREATE TABLE TRABAJA\_EN

```
( DniEmpleado    CHAR(9)      NOT NULL,
  NumProy        INT          NOT NULL,
  Horas          DECIMAL(3,1) NOT NULL,
  PRIMARY KEY(DniEmpleado, NumProy),
  FOREIGN KEY(DniEmpleado) REFERENCES EMPLEADO(Dni),
  FOREIGN KEY(NumProy) REFERENCES PROYECTO(NumProyecto) );
```

## CREATE TABLE SUBORDINADO

```
( NombSubordinado VARCHAR(15)  NOT NULL,
  Sexo            CHAR,
  FechaNac        DATE,
  Relación        VARCHAR(8),
  PRIMARY KEY(DniEmpleado, NombSubordinado),
  FOREIGN KEY(DniEmpleado) REFERENCES EMPLEADO(Dni) );
```

# Crear Tablas: Más Ejemplos

```
CREATE TABLE Artista (
```

```
    id            int            PRIMARY KEY,
```

```
    nombre        text          NOT NULL,
```

```
    nacionalidad text
```

```
);
```

```
CREATE TABLE Cancion (
```

```
    id            int            PRIMARY KEY,
```

```
    titulo        text          NOT NULL,
```

```
    genero        text,
```

```
    duracion      int,
```

```
    fecha         date,
```

```
    autor         int            NOT NULL REFERENCES Artista (id)
```

```
);
```

# Crear Tablas: Ejemplos

```
CREATE TABLE Usuario (  
    nick          varchar(30) PRIMARY KEY,  
    nombre        text        NOT NULL,  
    email         text        NOT NULL UNIQUE  
);
```

```
CREATE TABLE Contacto (  
    usuario1      varchar(30) REFERENCES Usuario (nick),  
    usuario2      varchar(30) REFERENCES Usuario (nick),  
    PRIMARY KEY (usuario1,usuario2)  
);
```

# Crear Tablas: Ejemplos

```
CREATE TABLE Escucha (  
    usuario      varchar(30) ,  
    cancion      int          REFERENCES Cancion (id),  
    PRIMARY KEY (usuario,cancion,instante)  
);
```

```
ALTER TABLE Escucha ADD instante timestamp; /* NULL's */  
ALTER TABLE Escucha DROP COLUMN instante;
```

```
ALTER TABLE Escucha ADD FOREIGN KEY (usuario)  
    REFERENCES Usuario (nick);
```

```
ALTER TABLE Usuario ADD PRIMARY KEY (nick);
```

# Especificación de Restricciones en SQL

## En un campo

NOT NULL

UNIQUE

PRIMARY KEY

REFERENCES *tabla* (*clave*) [(ON DELETE | ON UPDATE)  
(SET NULL | CASCADE | SET DEFAULT)]

DEFAULT *valor*

## Con nombre

CONSTRAINT *nombre* *restricción*

## En una tabla

PRIMARY KEY (*campo1*, *campo2*, ...)

FOREIGN KEY (*campo1*, *campo2*, ...)

REFERENCES *tabla* (*clave1*, *clave2*, ...)

UNIQUE (*campo1*, *campo2*, ...)

CHECK (*expresión*)

# Especificación de Restricciones de Atributo y Valores Predeterminados

- ♦ Como SQL permite atributos NULL, se puede especificar un atributo con NOT NULL.
- ♦ También se puede utilizar una clausula DEFAULT <valor> que se incluye en cada tupla si no se especifica ningún valor.
- ♦ También se puede restringir los valores de un atributo o dominio con la clausula CHECK
  - NumeroDpto **INT NOT NULL CHECK** (NumeroDpto >0 **AND** NumeroDpto < 21)
  - **CREATE DOMAIN NUM\_D AS INTEGER CHECK** (NUM\_D > 0 **AND** NUM\_D < 21)



# Especificación de Restricciones de Clave y Integridad Referencial

- ◆ Estas restricciones son muy importantes y cláusulas especiales:
  - PRIMARY KEY (lista de columnas) (**restricción de clave**)
  - UNIQUE (lista de columnas) (**restricción de clave**)
  - FOREIGN KEY (lista de columnas) REFERENCES Tabla(columnas) (**restricción de integridad referencial**)
- ◆ Una integridad referencial se puede violar por la inserción o eliminación de tuplas de atributos de FOREIGN KEY o la clave principal.
- ◆ La acción por defecto si se viola la restricción referencial es rechazar la operación (NO ACTION), pero se pueden poner más opciones:
  - SET DEFAULT
  - SET NULL
  - CASCADE
- ◆ Estos son los valores que se cambian en la tupla que se referencian
- ◆ Estas opciones deben calificarse:
  - ON DELETE
  - ON UPDATE

# Especificación de Restricciones de Clave y Integridad Referencial

## ♦Ejemplos

- ON DELETE CASCADE: elimina las tuplas referenciadas en cascada cuando eliminamos algo de la clave externa.
- ON UPDATE CASCADE: Actualiza la clave externa por la que se ha cambiado, en cascada.

♦NOTA: Borrar solo puede violar la integridad referencial siempre que la tupla a eliminar esté referenciada por claves externas de otras tuplas.

♦Ejemplo siguiente: si se elimina la tupla de un empleado supervisor, el valor de SuperDni queda a NULL automáticamente en todas la tuplas de empleado que hacían referencia a la tupla de empleado borrada.

♦Por el contrario si se actualiza el valor Dni de un empleado supervisor (se introdujo incorrectamente), entonces el valor nuevo se actualiza en cascada para el SuperDni de todas la tuplas de empleado que hacen referencia a la tupla actualizada.

♦Ver la siguiente transparencia.

# Crear Tablas con Restricciones

**CREATE TABLE EMPLEADO**

```
( ...  
    Dno          INT          NOT NULL      DEFAULT 1,  
    CONSTRAINT EMPPK  
        PRIMARY KEY(Dni),  
    CONSTRAINT SUPERFKEMP  
        FOREIGN KEY(SuperDni) REFERENCES EMPLEADO(Dni)  
            ON DELETE SET NULL    ON UPDATE CASCADE  
    CONSTRAINT EMPDEPTFK  
        FOREIGN KEY(Dno) REFERENCES DEPARTAMENTO(NumeroDpto)  
            ON DELETE SET DEFAULT ON UPDATE CASCADE );
```

**CREATE TABLE DEPARTAMENTO**

```
( ...,  
    DniDirector  CHAR(9)      NOT NULL      DEFAULT '888665555',  
    ...,  
    CONSTRAINT DEPTPK  
        PRIMARY KEY(NumeroDpto),  
    CONSTRAINT DEPTSK  
        UNIQUE(NombreDpto),  
    CONSTRAINT DEPTMGREFK  
        FOREIGN KEY(DniDirector) REFERENCES EMPLEADO(Dni)  
            ON DELETE SET DEFAULT  ON UPDATE CASCADE );
```

**CREATE TABLE LOCALIZACIONES\_DPTO**

```
( ...,  
    PRIMARY KEY(NumeroDpto, UbicaciónDpto),  
    FOREIGN KEY(NumeroDpto) REFERENCES DEPARTAMENTO(NúmeroDpto)  
        ON DELETE CASCADE    ON UPDATE CASCADE );
```

# Campos, Tuplas y Tablas en una BD

EMPLEADO

▲	Nombre text	Apellido1 text	Apellido2 text	<u>Dni</u> integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Jose	Perez	Perez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
2	Alberto	Campos	Sastre	333445555	1955-12-08	Avda Rios, 9	H	40000	888665555	5
3	Alicia	Jimenez	Celaya	999887777	1968-05-12	Gran Via, 38	M	25000	987654321	4
4	Juana	Sainz	Oreja	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
5	Eduardo	Ochoa	Paredes	888665555	1937-11-10	Las Peñas, 1	H	55000	[null]	1
6	Fernan...	Ojeda	Ordoñez	666884444	1962-09-15	Portillo, S/N	M	38000	333445555	5
7	Luis	Pajares	Morera	987987987	1969-03-29	Enebros, 90	H	25000	987654321	4
8	Aurora	Oliva	Avezuela	453453453	1972-07-31	Anton, 6	M	25000	333445555	5

TRABAJA\_EN

▲	<u>DniEmpleado</u> integer	<u>NumProy</u> integer	Horas numeric
1	123456789	1	32.5
2	123456789	2	7.5
3	666884444	3	40.0
4	453453453	1	20.0
5	453453453	2	20.0
6	333445555	2	10.0
7	333445555	10	10.0
8	333445555	3	10.0
9	333445555	20	10.0
10	999887777	30	30.0
11	999887777	10	10.0
12	987987987	10	35.0
13	987987987	30	5.0
14	987654321	30	20.0
15	987654321	20	15.0
16	888665555	20	[null]

PROYECTO

▲	NombreProyecto text	<u>NumProyecto</u> integer	UbicacionProyecto text	NumDptoProyecto integer
1	PruductoX	1	Valencia	5
2	ProductoY	2	Sevilla	5
3	ProductoZ	3	Madrid	5
4	Computacion	10	Gijon	4
5	Reorganizacion	20	Madrid	1
6	Comunicaciones	30	Gijon	4

LOCALIZACIONES\_DPTO

▲	<u>NumeroDpto</u> integer	<u>UbicacionDpto</u> text
1	1	Madrid
2	4	Gijon
3	5	Valencia
4	5	Sevilla
5	5	Madrid

SUBORDINADO

▲	<u>DniEmpleado</u> integer	<u>NombSubordinado</u> text	Sexo "char"	FechaNac date	Relacion text
1	333445555	Alicia	M	1986-04-05	Hija
2	333445555	Teodoro	H	1983-10-25	Hijo
3	333445555	Luisa	M	1958-05-03	Esposa
4	987654321	Alfonso	H	1942-01-28	Esposo
5	123456789	Miguel	H	1988-01-04	Hijo
6	123456789	Alicia	M	1988-12-30	Hija
7	123456789	Elisa	M	1967-05-05	Esposa

DEPARTAMENTO

▲	NombreDpto text	<u>NumeroDpto</u> integer	DniDirector integer	FechaIngresoDirector date
1	Investigacion	5	333445555	1988-05-22
2	Administracion	4	987654321	1995-01-01
3	Sede Central	1	888665555	1981-06-19

# Sentencias de SQL para cambiar el Esquema

- ♦ Hay comandos para evolucionar el esquema (alterar un esquema) de una BD en SQL
- ♦ Añadir o eliminar tablas, atributos, restricciones, etc....
- ♦ **DROP SCHEMA *empresa* CASCADE;** Se elimina todo el esquema con todos los elementos (... *empresa* **RESTRICT** solo se elimina si no contiene elementos)
- ♦ **DROP TABLE *subordinado* CASCADE;** (se elimina la relación y su definición, **RESTRICT** solo se elimina la tabla si no hace referencia a otra tabla)
- ♦ **DELETE FROM *tabla* [WHERE ...];** Para eliminar tuplas (**DELETE FROM *empleado* WHERE 'dni=123456789'; DELETE FROM EMPLEADO WHERE Apellido1='Cabrera';**)
- ♦ **ALTER TABLA *EMPRESA.EMPLEADO* ADD COLUMN Trabajo VARCHAR (12);** (no se introduce un valor para la nueva columna, se pone toda la nueva columna a NULL)
- ♦ **ALTER TABLA *EMPRESA.EMPLEADO* DROP COLUMN dirección CASCADE;** (**RESTRICT** solo se elimina la columna si no hay vistas o restricciones que hagan referencia a esta columna)

# Sentencias de SQL para cambiar el Esquema

## ♦ **ALTER TABLA *EMPRESA.DEPARTAMENTO* ALTER COLUMN**

DniDirector **DROP DEFAULT** (se elimina la clausula predeterminada para DniDirector)

## ♦ **ALTER TABLA *EMPRESA.DEPARTAMENTO* ALTER COLUMN**

DniDirector **SET DEFAULT** '333445555' (se define la clausula predeterminada para DniDirector)

## ♦ **ALTER TABLA *EMPRESA.EMPLEADO* DROP CONSTRAINT**

**SUPERFKEMP CASCADE**; (se elimina la restricción SUPERFKEMP de la relación *EMPLEADO*)

## ♦ **INSERT INTO *EMPLEADO* VALUES** ('Ricardo',

Roca', 'Flores', '653298653', '30-12-1962', 'Los Jarales, 47', 'H', '37000', '653298653', 4); (añade una nueva tupla, al menos hay que añadir los que están explícitamente en la definición de tabla a NOT NULL );

## ♦ **INSERT INTO *EMPLEADO* (*Nombre , Apellido1, Dno, Dni*) VALUES**

('Ricardo', 'Roca', '4', '653298653'); (añade parte de una nueva tupla, los no especificados se establecen a DEFAULT o a NULL);

# Sentencias de SQL para cambiar el Esquema

♦ **INSERT INTO EMPLEADO** (*Nombre* , *Apellido1*, *Dno*, *Dni*) **VALUES** ('Ricardo', 'Roca', '2', '653298653'); (Si el SGDB realiza la integridad referencial, se rechaza el comando); ¿Por qué?

♦ **INSERT INTO EMPLEADO** (*Nombre* , *Apellido1*, *Dno*) **VALUES** ('Ricardo', 'Roca', '4'); (Si el SGDB realiza la comprobación NOT NULL de 'Dni' no proporcionada y se rechaza el comando);

♦ **UPDATE PROYECTO SET** UbicaciónProyecto='Valencia', NumDptoProyecto = 5  
**WHERE** NumProyecto=10;

♦ **UPDATE Empleado SET** Sueldo=Sueldo\*1.1

**WHERE** Dno **IN** (**SELECT** NumeroDpto **FROM** DEPARTAMENTO  
**WHERE** NombreDpto='Investigación');

**IN** vamos a utilizarlo en consultas anidadas también.

♦ **TRUNCATE tabla**; (quita todas las filas de una tabla, pero permanecen la estructura y sus columnas, las restricciones, los índices, etc. Para quitar la definición de tabla además de los datos: **DROP TABLE**)

# Transacciones en SQL: BEGIN, COMMIT

- ♦ Todo SGDB maneja **transacciones**: Conjunto de acciones sobre una BD que altera los datos (**INSERT INTO**, **UPDATE**, **DELETE**, etc.) pero que deben ser realizados de manera atómica (para evitar concurrencia de usuarios, por ejemplo).
- ♦ **BEGIN, COMMIT, ROLLBACK, SAVEPOINT, ROLLBACK TO, RELEASE SAVEPOINT.**
- ♦ La documentación la podéis encontrar en el manual de POSTGRESQL:
  - <http://www.postgresql.org/docs/9.6/static/sql-begin.html>



# Transacciones en SQL: BEGIN, COMMIT

- ♦ Un ejemplo:

**BEGIN;**

Secuencia de comandos que alteran una BD determinada

**COMMIT;** (este comando finaliza las transacciones haciendo los cambios permanentes y visibles a todos los usuarios)

- ♦ **BEGIN** inicia un bloque de transacción, es decir, todas las declaraciones después de un comando **BEGIN** se ejecutarán en una sola transacción hasta que de manera explícita haya un **COMMIT** (plasma todos los cambios en la BD) o un **ROLLBACK** (deshace todos los cambios).

# Transacciones en SQL: ROLLBACK

- ♦ La transacción puede dar un error entre medias, para ello esta el comando **ROLLBACK**:

**BEGIN;**

Secuencia de comandos que alteran una BD determinada  
y en un punto determinado se genera un error

**ROLLBACK;** (este comando vuelve al **BEGIN** deshace todos  
los cambios)

Empiezas a hacer de nuevo los cambios, excepto el que te  
daba error.

**COMMIT;**

# Transacciones en SQL: SAVEPOINT

- ♦ Con el comando **SAVEPOINT**, también puedes grabar puntos interesantes intermedios, por si se produce un error inesperado y no tengas que volver al principio y así aproveches algunos cambios realizados:

**BEGIN;**

Secuencia de comandos que alteran una BD

**SAVEPOINT misavepoint**

Secuencia de comandos que alteran una BD

y en un punto determinado se genera un error

**ROLLBACK TO misavepoint;**

Secuencia nueva de comandos que alteran una BD

**COMMIT;**

# Transacciones en SQL: RELEASE SAVEPOINT

- ♦ Sirve para indicar que la aplicación ya no desea mantener el punto de salvaguarda especificado. Después de invocar esta sentencia, ya no es posible hacer una retrotracción hasta el punto de salvaguarda.

**BEGIN;**

Secuencia de comandos que alteran una BD

**SAVEPOINT misavepoint 1**

Secuencia de comandos que alteran una BD

**SAVEPOINT misavepoint 2**

Secuencia de comandos que alteran una BD

**RELEASE SAVEPOINT misavepoint2;**

**COMMIT;**

# Transacciones en SQL: ejemplos

- ♦ Para establecer un punto de salvaguarda y luego deshacer los efectos de todos los comandos ejecutados después de su creación, la transacción insertará los valores 1 y 3, pero no 2.:

**BEGIN;**

**INSERT INTO** table1 **VALUES** (1);

**SAVEPOINT** my\_savepoint;

**INSERT INTO** table1 **VALUES** (2);

**ROLLBACK TO SAVEPOINT** my\_savepoint;

**INSERT INTO** table1 **VALUES** (3);

**COMMIT;**

# Transacciones en SQL: ejemplos

- ♦ Establecer y posteriormente destruir un punto de salvaguarda, la transacción insertará tanto 3 y 4:

**BEGIN;**

**INSERT INTO table1 VALUES (3);**

**SAVEPOINT my\_savepoint;**

**INSERT INTO table1 VALUES (4);**

**RELEASE SAVEPOINT my\_savepoint;**

**COMMIT;**

# Más Ejemplos

```
INSERT INTO Artista VALUES (1, 'The Beatles', 'UK');
```

```
INSERT INTO Artista VALUES (2, 'The Rolling Stones', 'UK');
```

```
INSERT INTO Artista (id, nombre) VALUES (3, 'David Bowie');
```

```
INSERT INTO Cancion
```

```
VALUES (1, 'Norwegian wood', 'Pop', '125', '1965-03-12', 1);
```

```
INSERT INTO Cancion
```

```
VALUES (2, 'Here, there and everywhere', 'Pop', '145', '1966-08-05', 1);
```

```
INSERT INTO Cancion
```

```
VALUES (3, 'Jumping jack flash', 'Pop', '225', '1968-04-20', 2);
```

```
INSERT INTO Usuario VALUES ('lola', 'Dolores', 'lola@gmail.com');
```

```
INSERT INTO Usuario VALUES ('pepe', 'José', 'jose@gmail.com');
```

```
INSERT INTO Usuario VALUES ('chema', 'José María', 'chema@gmail.com');
```

```
INSERT INTO Usuario VALUES ('charo', 'Rosario', 'rosario@gmail.com');
```

# Más Ejemplos

```
INSERT INTO Contacto VALUES
```

```
    ('pepe', 'lola'),
```

```
    ('charo', 'pepe'),
```

```
    ('chema', 'charo');
```

```
INSERT INTO Escucha VALUES ('charo', 2, '2011-09-09 16:57:54');
```

```
INSERT INTO Escucha VALUES ('pepe', 3, '2011-09-12 21:15:30');
```

```
UPDATE Artista SET nacionalidad = 'UK' WHERE nombre = 'David Bowie';
```

```
UPDATE Album SET precio = precio * 1.2;
```

```
DELETE FROM Escucha WHERE instante < '2000-01-01 00:00:00';
```



# Consultas en SQL

**SELECT** [DISTINCT] *campos* **FROM** *tablas*  
[**WHERE** *condición*];

## Consulta A

**SELECT** FechaNac, Dirección **FROM** EMPLEADO  
**WHERE** Nombre='Jose' **AND** Apellido1='Pérez' **AND** Apellido2='Pérez';

## Otras consultas:

**SELECT** titulo, genero **FROM** Cancion  
**WHERE** fecha < '1967-01-01';

**SELECT DISTINCT** nacionalidad **FROM** Artista; (solo las tuplas diferentes permanecen en el resultado, en contraposición a **SELECT ALL**)

**SELECT \* FROM** Cancion, Artista

**WHERE** Cancion.autor = Artista.id **AND** Artista.nacionalidad = 'UK';

**SELECT** dni, teoria \* 0.6 + practicas \* 0.4 **FROM** Notas;

# Consultas en SQL sencillas

## *Ejemplo 1*

```
SELECT "FechaNac", "Direccion"  
FROM "EMPLEADO"  
WHERE "Nombre"='Jose' AND "Apellido1"='Perez' AND "Apellido2"='Perez';
```

## *Ejemplo 2*

```
SELECT *  
FROM "EMPLEADO"  
WHERE "Nombre"='Jose' AND "Apellido1"='Perez' AND "Apellido2"='Perez';
```

## *Ejemplo 3*

```
SELECT *  
FROM "EMPLEADO"  
WHERE "Sexo"='M';
```

# Consultas en SQL sencillas

## *Ejemplo 4*

```
SELECT "Sexo"  
FROM "EMPLEADO";
```

## *Ejemplo 5*

```
SELECT DISTINCT "Sexo"  
FROM "EMPLEADO";
```

## *Ejemplo 6*

```
SELECT "Nombre", "Apellido1", "Direccion"  
FROM "EMPLEADO", "DEPARTAMENTO"  
WHERE "NombreDpto"='Investigacion' AND "Dno"="NumeroDpto";
```

## *Ejemplo 7*

```
SELECT "EMPLEADO"."Nombre", "EMPLEADO"."Apellido1", "EMPLEADO"."Direccion"  
FROM "EMPLEADO", "DEPARTAMENTO"  
WHERE "EMPLEADO"."Dno"="DEPARTAMENTO"."NumeroDpto" AND  
"DEPARTAMENTO"."NombreDpto"='Investigacion';
```

# Consultas en SQL (INNER JOIN)

## *Consulta B*

```
SELECT Nombre, Apellido1, Dirección  
FROM EMPLEADO, DEPARTAMENTO (o de manera explícita FROM  
EMPLEADO INNER JOIN DEPARTAMENTO ON NumeroDpto =Dno )  
WHERE NombreDpto='Investigación' AND NumeroDpto =Dno;  
(selección-proyección-concatenación, Álgebra relacional)
```

## *Consulta C*

```
SELECT NumProyecto, NumDptoProyecto, Apellido1, Dirección,  
FechaNac  
FROM PROYECTO, DEPARTAMENTO, EMPLEADO  
WHERE NumDptoProyecto =NumeroDpto AND DniDirector=Dni AND  
UbicacionProyecto='Gijon';  
(selección-proyección-2 condiciones de concatenación, Álgebra relacional)
```

# Campos, Tuplas y Tablas en una BD

EMPLEADO

▲	Nombre text	Apellido1 text	Apellido2 text	<u>Dni</u> integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Jose	Perez	Perez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
2	Alberto	Campos	Sastre	333445555	1955-12-08	Avda Rios, 9	H	40000	888665555	5
3	Alicia	Jimenez	Celaya	999887777	1968-05-12	Gran Via, 38	M	25000	987654321	4
4	Juana	Sainz	Oreja	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
5	Eduardo	Ochoa	Paredes	888665555	1937-11-10	Las Peñas, 1	H	55000	[null]	1
6	Fernan...	Ojeda	Ordoñez	666884444	1962-09-15	Portillo, S/N	M	38000	333445555	5
7	Luis	Pajares	Morera	987987987	1969-03-29	Enebros, 90	H	25000	987654321	4
8	Aurora	Oliva	Avezuela	453453453	1972-07-31	Anton, 6	M	25000	333445555	5

TRABAJA\_EN

▲	<u>DniEmpleado</u> integer	<u>NumProy</u> integer	Horas numeric
1	123456789	1	32.5
2	123456789	2	7.5
3	666884444	3	40.0
4	453453453	1	20.0
5	453453453	2	20.0
6	333445555	2	10.0
7	333445555	10	10.0
8	333445555	3	10.0
9	333445555	20	10.0
10	999887777	30	30.0
11	999887777	10	10.0
12	987987987	10	35.0
13	987987987	30	5.0
14	987654321	30	20.0
15	987654321	20	15.0
16	888665555	20	[null]

PROYECTO

▲	NombreProyecto text	<u>NumProyecto</u> integer	UbicacionProyecto text	NumDptoProyecto integer
1	PruductoX	1	Valencia	5
2	ProductoY	2	Sevilla	5
3	ProductoZ	3	Madrid	5
4	Computacion	10	Gijon	4
5	Reorganizacion	20	Madrid	1
6	Comunicaciones	30	Gijon	4

LOCALIZACIONES\_DPTO

▲	<u>NumeroDpto</u> integer	<u>UbicacionDpto</u> text
1	1	Madrid
2	4	Gijon
3	5	Valencia
4	5	Sevilla
5	5	Madrid

SUBORDINADO

▲	<u>DniEmpleado</u> integer	<u>NombSubordinado</u> text	Sexo "char"	FechaNac date	Relacion text
1	333445555	Alicia	M	1986-04-05	Hija
2	333445555	Teodoro	H	1983-10-25	Hijo
3	333445555	Luisa	M	1958-05-03	Esposa
4	987654321	Alfonso	H	1942-01-28	Esposo
5	123456789	Miguel	H	1988-01-04	Hijo
6	123456789	Alicia	M	1988-12-30	Hija
7	123456789	Elisa	M	1967-05-05	Esposa

DEPARTAMENTO

▲	NombreDpto text	<u>NumeroDpto</u> integer	DniDirector integer	FechaIngresoDirector date
1	Investigacion	5	333445555	1988-05-22
2	Administracion	4	987654321	1995-01-01
3	Sede Central	1	888665555	1981-06-19

# Campos, Tuplas y Tablas en una BD

## EMPLEADO

Nombre	Apellido1	Apellido2	<u>Dni</u>	FechaNac	Direccion	Sexo	Sueldo	SuperDni	Dno
text	text	text	integer	date	text	"char"	numeric	integer	integer

## DEPARTAMENTO

NombreDpto	<u>NumeroDpto</u>	DniDirector	FechaIngresoDirector
text	integer	integer	date

## LOCALIZACIONES\_DPTO

<u>NumeroDpto</u>	<u>UbicacionDpto</u>
integer	text

## PROYECTO

NombreProyecto	<u>NumProyecto</u>	UbicacionProyecto	NumDptoProyecto
text	integer	text	integer

## TRABAJA\_EN

<u>DniEmpleado</u>	<u>NumProy</u>	Horas
integer	integer	numeric

## SUBORDINADO

<u>DniEmpleado</u>	<u>NombSubordinado</u>	Sexo	FechaNac	Relacion
integer	text	"char"	date	text

# Concatenación de Tablas, Join

**SELECT** *campos*

**FROM** *tabla1* **JOIN** *tabla2* **ON** *condición*

[**WHERE** *condición*];

(JOIN: Concepto de tabla concatenada o relación concatenada)

## *Consulta B*

**SELECT** Nombre, Apellido1, Dirección

**FROM** (EMPLEADO **JOIN** DEPARTAMENTO **ON** Dno= NumeroDpto)

**WHERE** NombreDpto='Investigación';

Uso típico (pero no sólo) con claves externas: ON externa = primaria

**SELECT** titulo **FROM** Cancion, Escucha **WHERE** usuario='lola';

**SELECT** titulo **FROM** (Cancion **JOIN** Escucha **ON** cancion = id);

**SELECT** \* **FROM** (Contacto **JOIN** Usuario **ON** (usuario1 = nick **OR** usuario2 = nick)) **WHERE** nombre = 'Rosario';

(ver tablas siguientes)

# Tablas

```
CREATE TABLE Usuario (  
    nick          varchar(30) PRIMARY KEY,  
    nombre        text        NOT NULL,  
    email         text        NOT NULL UNIQUE  
);
```

```
CREATE TABLE Contacto (  
    usuario1      varchar(30) REFERENCES Usuario (nick),  
    usuario2      varchar(30) REFERENCES Usuario (nick),  
    PRIMARY KEY (usuario1,usuario2)  
);
```



# Ejemplos de Consultas en SQL: Join

## *Ejemplo 6*

```
SELECT "Nombre", "Apellido1", "Direccion"  
FROM "EMPLEADO", "DEPARTAMENTO"  
WHERE "NombreDpto"='Investigacion' AND "Dno"="NumeroDpto";
```

## *Ejemplo 7*

```
SELECT "EMPLEADO"."Nombre", "EMPLEADO"."Apellido1", "EMPLEADO"."Direccion"  
FROM "EMPLEADO", "DEPARTAMENTO"  
WHERE "EMPLEADO"."Dno"="DEPARTAMENTO"."NumeroDpto" AND  
"DEPARTAMENTO"."NombreDpto"='Investigacion';
```

## *Ejemplo 8*

```
SELECT "Nombre", "Apellido1", "Direccion"  
FROM "EMPLEADO" INNER JOIN "DEPARTAMENTO" ON  
"Dno"="NumeroDpto"  
WHERE "NombreDpto"='Investigacion';
```

# Ejemplos de Consultas en SQL: Join

## *Ejemplo 9*

```
SELECT "Nombre", "Apellido1", "Direccion"  
FROM "EMPLEADO" JOIN "DEPARTAMENTO" ON "Dno"="NumeroDpto";
```

## *Ejemplo 10*

```
SELECT *  
FROM "EMPLEADO" JOIN "DEPARTAMENTO" ON "Dno"="NumeroDpto";
```

## *Ejemplo 11*

```
SELECT *  
FROM "EMPLEADO" INNER JOIN "DEPARTAMENTO" ON  
"Dno"="NumeroDpto";
```

# Concatenación de Tablas, Join

## *Consulta B*

```
SELECT Nombre, Apellido1, Dirección  
FROM (EMPLEADO NATURAL JOIN  
      (DEPARTAMENTO AS DEPT(NombreDpto, Dno,  
DniDirector, FechaIngresoDirector)))  
WHERE NombreDpto='Investigación';
```

(Antes de hacer el NATURAL JOIN si los nombres de los atributos de concatenación no coinciden en las relaciones base, se renombran para que coincidan, a través de AS para renombrar la relación. En este caso en la tabla DEPARTAMENTO NumeroDpto se renombra a Dno para el NJ)

Realmente AS se puede utilizar para definir un ALIAS, lo veremos más adelante.

♦OJO: En SQL se pueden utilizar el mismo nombre para dos o más atributos siempre que se encuentren en relaciones diferentes. Para eso están los **alias** o **variables de tupla**, (ejemplos a continuación).

# Campos, Tuplas y Tablas en una BD

## EMPLEADO

	Nombre text	Apellido1 text	Apellido2 text	Dni integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Jose	Perez	Perez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
2	Alberto	Campos	Sastre	333445555	1955-12-08	Avda Rios, 9	H	40000	888665555	5
3	Alicia	Jimenez	Celaya	999887777	1968-05-12	Gran Via, 38	M	25000	987654321	4
4	Juana	Sainz	Oreja	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
5	Eduardo	Ochoa	Paredes	888665555	1937-11-10	Las Peñas, 1	H	55000	[null]	1
6	Fernan...	Ojeda	Ordoñez	666884444	1962-09-15	Portillo, S/N	M	38000	333445555	5
7	Luis	Pajares	Morera	987987987	1969-03-29	Enebro, 90	H	25000	987654321	4
8	Aurora	Oliva	Avezuela	453453453	1972-07-31	Anton, 6	M	25000	333445555	5

## DEPARTAMENTO

*Con AS pasa a Dno*

	NombreDpto text	NumeroDpto integer	DniDirector integer	FechaIngresoDirector date
1	Investigacion	5	333445555	1988-05-22
2	Administracion	4	987654321	1995-01-01
3	Sede Central	1	888665555	1981-06-19

# Nombres de Atributos, Alias y Variables de Tupla.

## *Consulta D*

```
SELECT E.Nombre, E.Apellido1, S.Nombre, S.Apellido1  
FROM EMPLEADO AS E, EMPLEADO AS S  
WHERE E.SuperDni=S.Dni;
```

- ♦ Por cada empleado se recupera el nombre y primer apellido del mismo y el nombre y primer apellido de su supervisor inmediato.
- ♦ La tabla E se va a utilizar como tabla para extraer la información de los empleados que son supervisados. La tabla S se utiliza para extraer la información de los empleados supervisores.
- ♦ Darse cuenta que las tablas E y S son copias de la tabla empleado.

## *Consulta D (notación reducida)*

```
SELECT E.Nombre, E.Apellido1, S.Nombre, S.Apellido1  
FROM EMPLEADO E, EMPLEADO S  
WHERE E.SuperDni=S.Dni;
```

# Tipos de join

- ♦ INNER Por defecto (no hace falta ponerlo)
- ♦ NATURAL La condición consiste en igualdad entre la combinación de los campos que se llamen igual entre ambas tablas (EQUIJOIN) (no se repiten los campos)
- ♦ LEFT | RIGHT | FULL Se añaden también filas que no cumplen la condición (incompatible con INNER)(OUTER JOIN)
  - $R (->< \text{Dni}=\text{DniDirector}) S$  (LEFT OUTER JOIN, mantiene cada tupla de la relación izquierda aunque no se encuentre ninguna tupla en S que cumple la conexión, esos atributos se rellenan a NULL)
  - $R (><- \text{Dni}=\text{DniDirector}) S$  (RIGHT OUTER JOIN, mantiene cada tupla de la relación derecha aunque no se encuentre ninguna tupla en R que cumple la conexión, esos atributos se rellenan a NULL)

# Ejemplos de Consultas en SQL: Join

## *Ejemplo 12*

```
SELECT *  
  
FROM "EMPLEADO" LEFT JOIN "DEPARTAMENTO" ON  
"Dno"="NumeroDpto";
```

## *Ejemplo 13*

```
SELECT *  
  
FROM "EMPLEADO" LEFT JOIN "DEPARTAMENTO" ON  
"Dni"="DniDirector";
```

## *Ejemplo 14*

```
SELECT *  
  
FROM "EMPLEADO" RIGHT JOIN "DEPARTAMENTO" ON  
"Dni"="DniDirector";
```

# Ejemplos de Consultas en SQL: Join

## *Ejemplo 15*

```
SELECT "NumProyecto", "NumDptoProyecto", "Apellido1", "Direccion",  
"FechaNac"  
  
FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"  
  
WHERE "NumDptoProyecto"="NumeroDpto" AND "DniDirector"="Dni" AND  
"UbicacionProyecto"='Gijon';
```

## *Ejemplo 16*

```
SELECT *  
  
FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"  
  
WHERE "NumDptoProyecto"="NumeroDpto" AND "DniDirector"="Dni";
```



# Ejemplos de Consultas en SQL: AS

## *Ejemplo 18*

```
SELECT "Nombre", "Apellido1", "Direccion"  
FROM "EMPLEADO" NATURAL JOIN "DEPARTAMENTO" AS  
"DEPT"("NombreDpto","Dno","DniDirector","FechaIngresoDirector")  
WHERE "NombreDpto"='Investigacion';
```

## *Ejemplo 19*

```
SELECT *  
FROM "EMPLEADO" NATURAL JOIN "DEPARTAMENTO" AS  
"DEPT"("NombreDpto","Dno","DniDirector","FechaIngresoDirector");
```

## *Ejemplo 20*

```
SELECT "E"."Nombre", "E"."Apellido1", "S"."Nombre", "S"."Apellido1"  
FROM "EMPLEADO" AS "E", "EMPLEADO" AS "S"  
WHERE "E"."SuperDni"="S"."Dni";
```

# Más ejemplos de join

```
CREATE TABLE Alumno (  
    dni VARCHAR(12) PRIMARY KEY, nombre text);  
CREATE TABLE Asignatura (  
    codigo NUMERIC PRIMARY KEY, nombre text);  
CREATE TABLE Notas (  
    dni VARCHAR(12) REFERENCES Alumno(dni),  
    codigo NUMERIC REFERENCES Asignatura(codigo),  
    teoria NUMERIC (4,2), practicas NUMERIC (4,2),  
    PRIMARY KEY (dni, codigo));
```

```
SELECT nombre, teoria FROM Notas NATURAL JOIN Asignatura;
```

```
SELECT nombre, teoria FROM Notas JOIN Asignatura  
ON Notas.codigo = Asignatura.codigo;
```

*Recordar la notación: ON externa = primaria*

# Mas Alias

**SELECT** *campos* **FROM** *tabla* **AS** *alias* [(*alias-campo1*, *alias-campo2*, ...)]  
[**WHERE** *condición*];

**SELECT** *campo* **AS** *alias* **FROM** ...

Ejemplos:

**SELECT** dni, teoria \* 0.6 + practicas \* 0.4 as media  
**FROM** Notas;

**SELECT** u1.nombre  
**FROM** Usuario **AS** u1, Usuario **AS** u2  
**WHERE** u1.nombre = u2.nombre **AND** u1.nick <> u2.nick;  
(ver tablas anteriores y siguientes)

# Tablas

```
CREATE TABLE Usuario (  
    nick          varchar(30) PRIMARY KEY,  
    nombre        text        NOT NULL,  
    email         text        NOT NULL UNIQUE  
);
```

```
CREATE TABLE Contacto (  
    usuario1      varchar(30) REFERENCES Usuario (nick),  
    usuario2      varchar(30) REFERENCES Usuario (nick),  
    PRIMARY KEY (usuario1,usuario2)  
);
```

# Consultas anidadas

Son conexiones entre consultas a través del operador de comparación normalmente IN.

**SELECT** *campos* FROM *tabla*

WHERE *campo1, campo2, ...* **IN** (SELECT *campo1, campo2, ...*);

Ejemplo: (selecciona los números de proyectos que tienen Pérez como director)

**SELECT DISTINCT** NumProyecto

**FROM** PROYECTO

**WHERE** NumProyecto **IN**

(SELECT NumProyecto

**FROM** PROYECTO, DEPARTAMENTO, EMPLEADO

**WHERE** NumDptoProyecto =NumeroDpto **AND** DniDirector=Dni **AND**  
Apellido1='Pérez';)

(Con la palabra clave **DISTINCT** eliminamos las tuplas iguales, solo permanecen en el resultado las tuplas distintas)

# Consultas anidadas (más opciones)

**SELECT** *campos* FROM *tabla*

WHERE *campo comparación* (SOME | ALL) (SELECT ...);

**SELECT** *campos* FROM *tabla*

WHERE EXISTS (SELECT ...);

**SELECT** *campos* FROM *tabla*

WHERE (SELECT ...) **CONTAINS** (SOME | ALL) (SELECT ...);

# Consultas anidadas

## Ejemplo

```
SELECT u2.nombre FROM Usuario AS u1, Usuario as u2
WHERE
    (u1.nick, u2.nick) IN
        ((SELECT usuario1, usuario2 FROM Contacto)
         UNION
         (SELECT usuario2, usuario1 FROM Contacto))
AND u1.nombre = 'Rosario';
```

# Ejemplos de Consultas anidadas

*Ejemplo 26* (Características de proyectos que tienen a Campos como director)

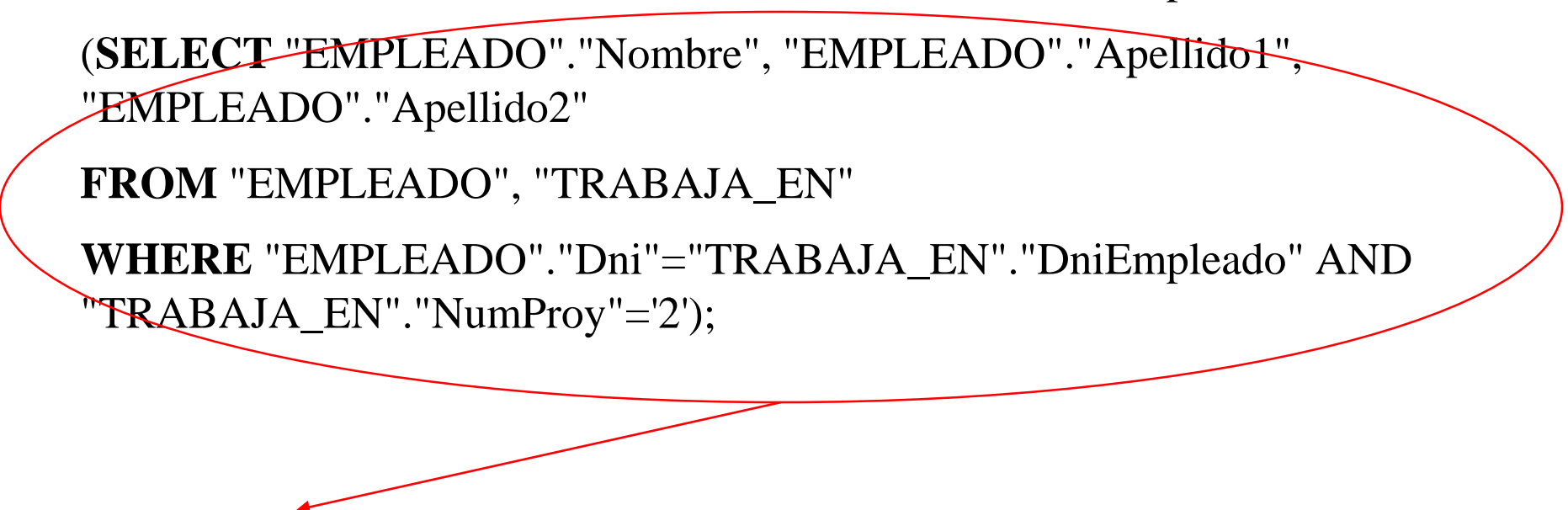
```
SELECT *  
FROM "PROYECTO"  
WHERE "NumProyecto" IN  
(SELECT "NumProyecto"  
FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"  
WHERE "NumDptoProyecto"="NumeroDpto" AND "DniDirector"="Dni" AND  
"Apellido1"='Campos');
```



# Ejemplos de Consultas anidadas

**Ejemplo 27** (Todos los empleados que NO trabajen en el proyecto 2, con la operación resta algebraica, EXCEPT)

```
SELECT "EMPLEADO"."Nombre", "EMPLEADO"."Apellido1",  
"EMPLEADO"."Apellido2"  
FROM "EMPLEADO", "TRABAJA_EN"  
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" EXCEPT  
(SELECT "EMPLEADO"."Nombre", "EMPLEADO"."Apellido1",  
"EMPLEADO"."Apellido2"  
FROM "EMPLEADO", "TRABAJA_EN"  
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" AND  
"TRABAJA_EN"."NumProy"='2');
```



*Son todos los que trabajan en el proyecto 2*

# Campos, Tuplas y Tablas en una BD

## EMPLEADO

▲	Nombre text	Apellido1 text	Apellido2 text	<u>Dni</u> integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Jose	Perez	Perez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
2	Alberto	Campos	Sastre	333445555	1955-12-08	Avda Rios, 9	H	40000	888665555	5
3	Alicia	Jimenez	Celaya	999887777	1968-05-12	Gran Via, 38	M	25000	987654321	4
4	Juana	Sainz	Oreja	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
5	Eduardo	Ochoa	Paredes	888665555	1937-11-10	Las Peñas, 1	H	55000	[null]	1
6	Fernan...	Ojeda	Ordoñez	666884444	1962-09-15	Portillo, S/N	M	38000	333445555	5
7	Luis	Pajares	Morera	987987987	1969-03-29	Enebros, 90	H	25000	987654321	4
8	Aurora	Oliva	Avezuela	453453453	1972-07-31	Anton, 6	M	25000	333445555	5

## PROYECTO

▲	NombreProyecto text	<u>NumProyecto</u> integer	UbicacionProyecto text	NumDptoProyecto integer
1	PruductoX	1	Valencia	5
2	ProductoY	2	Sevilla	5
3	ProductoZ	3	Madrid	5
4	Computacion	10	Gijon	4
5	Reorganizacion	20	Madrid	1
6	Comunicaciones	30	Gijon	4

## LOCALIZACIONES\_DPTO

▲	<u>NumeroDpto</u> integer	<u>UbicacionDpto</u> text
1	1	Madrid
2	4	Gijon
3	5	Valencia
4	5	Sevilla
5	5	Madrid

## TRABAJA\_EN

▲	<u>DniEmpleado</u> integer	<u>NumProy</u> integer	Horas numeric
1	123456789	1	32.5
2	123456789	2	7.5
3	666884444	3	40.0
4	453453453	1	20.0
5	453453453	2	20.0
6	333445555	2	10.0
7	333445555	10	10.0
8	333445555	3	10.0
9	333445555	20	10.0
10	999887777	30	30.0
11	999887777	10	10.0
12	987987987	10	35.0
13	987987987	30	5.0
14	987654321	30	20.0
15	987654321	20	15.0
16	888665555	20	[null]

## SUBORDINADO

▲	<u>DniEmpleado</u> integer	<u>NombSubordinado</u> text	Sexo "char"	FechaNac date	Relacion text
1	333445555	Alicia	M	1986-04-05	Hija
2	333445555	Teodoro	H	1983-10-25	Hijo
3	333445555	Luisa	M	1958-05-03	Esposa
4	987654321	Alfonso	H	1942-01-28	Esposo
5	123456789	Miguel	H	1988-01-04	Hijo
6	123456789	Alicia	M	1988-12-30	Hija
7	123456789	Elisa	M	1967-05-05	Esposa

## DEPARTAMENTO

▲	NombreDpto text	<u>NumeroDpto</u> integer	DniDirector integer	FechaIngresoDirector date
1	Investigacion	5	333445555	1988-05-22
2	Administracion	4	987654321	1995-01-01
3	Sede Central	1	888665555	1981-06-19

# Campos, Tuplas y Tablas en una BD

## EMPLEADO

Nombre	Apellido1	Apellido2	<u>Dni</u>	FechaNac	Direccion	Sexo	Sueldo	SuperDni	Dno
text	text	text	integer	date	text	"char"	numeric	integer	integer

## DEPARTAMENTO

NombreDpto	<u>NumeroDpto</u>	DniDirector	FechaIngresoDirector
text	integer	integer	date

## LOCALIZACIONES\_DPTO

<u>NumeroDpto</u>	<u>UbicacionDpto</u>
integer	text

## PROYECTO

NombreProyecto	<u>NumProyecto</u>	UbicacionProyecto	NumDptoProyecto
text	integer	text	integer

## TRABAJA\_EN

<u>DniEmpleado</u>	<u>NumProy</u>	Horas
integer	integer	numeric

## SUBORDINADO

<u>DniEmpleado</u>	<u>NombSubordinado</u>	Sexo	FechaNac	Relacion
integer	text	"char"	date	text

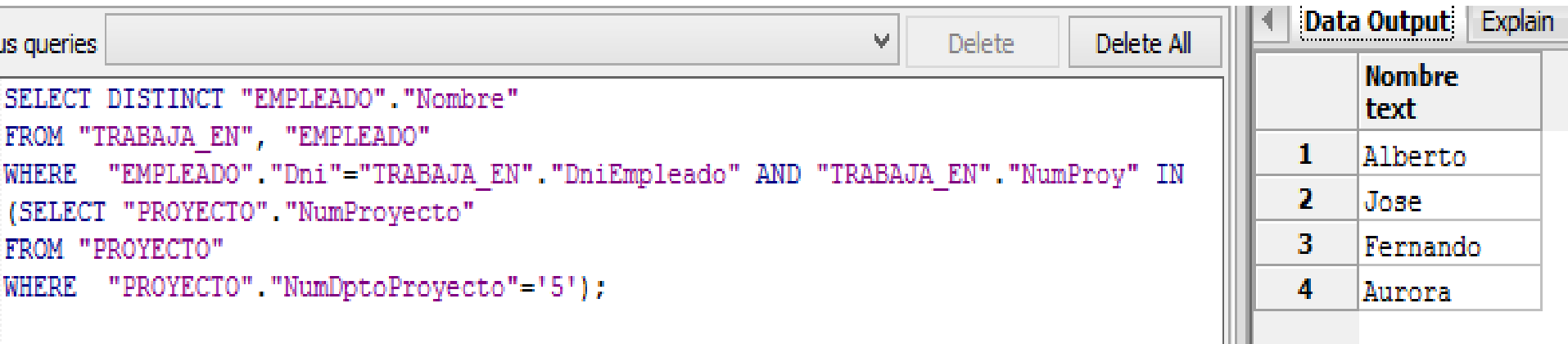
# Ejemplos de Consultas anidadas

***Ejemplo 29*** (Enumere el nombre de todos los empleados que trabajan en algún proyecto controlado por el departamento 5. El segundo SELECT me proporciona los números de proyecto que controla el departamento 5)

```
SELECT DISTINCT "EMPLEADO"."Nombre"  
FROM "TRABAJA_EN", "EMPLEADO"  
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" AND  
"TRABAJA_EN"."NumProy" IN  
(SELECT "PROYECTO"."NumProyecto"  
FROM "PROYECTO"  
WHERE "PROYECTO"."NumDptoProyecto"='5');
```

# Consultas anidadas (detalle IN)

Enumere el nombre de todos los empleados que trabajan en algún proyecto controlado por el departamento 5.



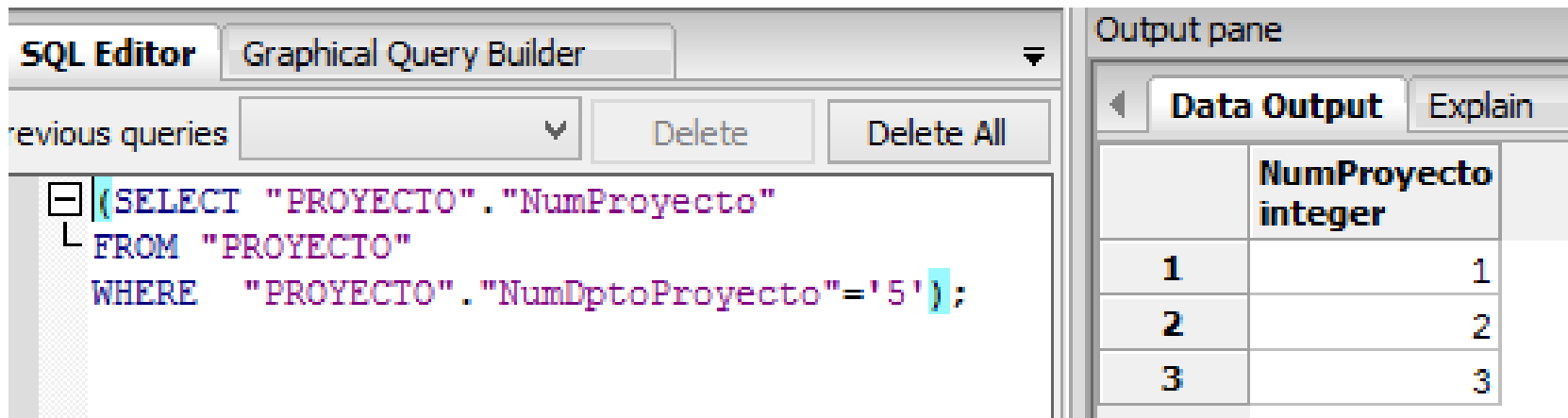
The screenshot shows a SQL IDE interface. On the left, the 'Previous queries' list is empty. The main editor contains the following SQL query:

```
SELECT DISTINCT "EMPLEADO"."Nombre"
FROM "TRABAJA_EN", "EMPLEADO"
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" AND "TRABAJA_EN"."NumProy" IN
(SELECT "PROYECTO"."NumProyecto"
FROM "PROYECTO"
WHERE "PROYECTO"."NumDptoProyecto"='5');
```

On the right, the 'Data Output' pane shows the results of the query:

	Nombre text
1	Alberto
2	Jose
3	Fernando
4	Aurora

El segundo SELECT me proporciona los números de proyecto que controla el departamento 5.



The screenshot shows a SQL IDE interface. The 'SQL Editor' tab is active, and the 'Graphical Query Builder' is also visible. The main editor contains the following SQL query:

```
(SELECT "PROYECTO"."NumProyecto"
FROM "PROYECTO"
WHERE "PROYECTO"."NumDptoProyecto"='5');
```

On the right, the 'Output pane' shows the results of the query:

	NumProyecto integer
1	1
2	2
3	3

# Consultas anidadas (detalle IN)

La palabra clave **IN** equivale a establecer condiciones sobre un mismo campo conectadas por el operador OR.

The screenshot shows a SQL Editor window with two tabs: "SQL Editor" and "Graphical Query Builder". The "SQL Editor" tab is active, displaying a SQL query. The query is as follows:

```
SELECT DISTINCT "EMPLEADO"."Nombre"  
FROM "TRABAJA_EN", "EMPLEADO"  
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" AND  
("TRABAJA_EN"."NumProy"=1 OR "TRABAJA_EN"."NumProy"=2 OR "TRABAJA_EN"."NumProy"=3);
```

The WHERE clause condition is circled in red. Below the query editor is a scroll bar. At the bottom of the window is the "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing a table with the results of the query.

	Nombre text
1	Alberto
2	Aurora
3	Fernando
4	Jose

# Ejemplos de Consultas anidadas

**Ejemplo 30-A** (Obtener una lista de los números de los proyectos que impliquen a cualquier empleado cuyo primer apellido sea 'Campos', independientemente de que sean trabajadores o directores del departamento que gestiona dicho proyecto)

```
SELECT "PROYECTO"."NumProyecto"  
FROM "PROYECTO"  
WHERE "PROYECTO"."NumProyecto" IN  
(SELECT "PROYECTO"."NumProyecto"  
FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"  
WHERE "PROYECTO"."NumDptoProyecto"="DEPARTAMENTO"."NumeroDpto" AND  
"DEPARTAMENTO"."DniDirector"="EMPLEADO"."Dni" AND "Apellido1"='Campos')  
OR  
"PROYECTO"."NumProyecto" IN  
(SELECT "TRABAJA_EN"."NumProy"  
FROM "TRABAJA_EN", "EMPLEADO"  
WHERE "TRABAJA_EN"."DniEmpleado"="EMPLEADO"."Dni" AND  
"Apellido1"='Campos');
```

# Consultas anidadas

Obtener una lista de los números de los proyectos que impliquen a cualquier empleado cuyo primer apellido sea 'Campos', independientemente de que sean trabajadores o directores del departamento que gestiona dicho proyecto:

The screenshot shows a SQL Editor window with a query that uses nested subqueries to find project numbers. The query is as follows:

```
SELECT "PROYECTO"."NumProyecto"
FROM "PROYECTO"
WHERE
  "PROYECTO"."NumProyecto" IN
  (SELECT "PROYECTO"."NumProyecto"
   FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"
   WHERE "PROYECTO"."NumDptoProyecto"="DEPARTAMENTO"."NumeroDpto"
   AND "DEPARTAMENTO"."DniDirector"="EMPLEADO"."Dni"
   AND "Apellido1"='Campos')
OR
  "PROYECTO"."NumProyecto" IN
  (SELECT "TRABAJA_EN"."NumProy"
   FROM "TRABAJA_EN", "EMPLEADO"
   WHERE "TRABAJA_EN"."DniEmpleado"="EMPLEADO"."Dni" AND "Apellido1"='Campos');
```

Annotations in the image include:

- A red oval around the first subquery, with an arrow pointing to a table labeled *Directores*.
- A red oval around the second subquery, with an arrow pointing to a table labeled *Empleados*.

The *Directores* table has the following data:

	NumProyecto integer
1	1
2	2
3	3

The *Empleados* table has the following data:

	NumProy integer
1	2
2	10
3	3
4	20

The Output pane on the right shows the final result set:

	NumProyecto integer
1	1
2	2
3	3
4	10
5	20



# Álgebra de conjuntos

*consulta1* UNION *consulta2*

*consulta1* INTERSECT *consulta2*

*consulta1* EXCEPT *consulta2*

Tuplas homogéneas: los conjuntos de tuplas tienen que tener los mismos campos  
Aplica un DISTINCT implícito (a menos que indiquemos ALL)

Ejemplo:

```
(SELECT usuario2 FROM Contacto WHERE usuario1 = 'charo'  
UNION
```

```
SELECT usuario1 FROM Contacto WHERE usuario2 = 'charo')  
INTERSECT
```

```
(SELECT usuario2 FROM Contacto WHERE usuario1 = 'lola'  
UNION
```

```
SELECT usuario1 FROM Contacto WHERE usuario2 = 'lola')
```

# Ejemplos de Consultas anidadas

*Ejemplo 30-B* (Obtener una lista de los números de los proyectos que impliquen a cualquier empleado cuyo primer apellido sea ‘Campos’, independientemente de que sean trabajadores o directores del departamento que gestiona dicho proyecto)

También se puede hacer con **UNION**:

```
SELECT "PROYECTO"."NumProyecto"
FROM "PROYECTO"
WHERE "PROYECTO"."NumProyecto" IN
((SELECT "PROYECTO"."NumProyecto"
FROM "PROYECTO", "DEPARTAMENTO", "EMPLEADO"
WHERE "PROYECTO"."NumDptoProyecto"="DEPARTAMENTO"."NumeroDpto" AND
"DEPARTAMENTO"."DniDirector"="EMPLEADO"."Dni" AND "Apellido1"='Campos')
UNION
(SELECT "TRABAJA_EN"."NumProy"
FROM "TRABAJA_EN", "EMPLEADO"
WHERE "TRABAJA_EN"."DniEmpleado"="EMPLEADO"."Dni" AND
"Apellido1"='Campos'));
```

# Orden, agregación

**SELECT COUNT** (*campos*) FROM *tabla* ...

[**GROUP BY** *campo1, campo2, ...*];

**SELECT SUM | MAX | MIN | AVG** (*campo*) FROM *tabla* ...

[**GROUP BY** *campo1, campo2, ...*];

**SELECT** ...

[**ORDER BY** *campo1, campo2, ...*];

# Orden, agregación

Ejemplos:

```
SELECT COUNT (*) FROM Escucha JOIN Cancion ON cancion = id  
WHERE titulo = 'Norwegian Wood';
```

```
SELECT autor, COUNT (*) FROM Escucha JOIN Cancion ON cancion = id  
GROUP BY autor;
```

```
SELECT * FROM Usuario  
WHERE (SELECT COUNT (*) FROM Contacto  
       WHERE usuario1 = nick OR usuario2 = nick) > 2;
```

```
SELECT * FROM Usuario  
ORDER BY (SELECT COUNT (*) FROM Contacto  
          WHERE usuario1 = nick OR usuario2 = nick);
```

# Orden, agregación

ID_EMPLEADO	NOMBRE	APELLIDOS	F_NACIMIENTO	SEXO	CARGO	SALARIO
1	Carlos	Jiménez Clarín	1985-05-03	H	Mozo	1500
2	Elena	Rubio Cuestas	1978-09-25	M	Secretaria	1300
3	José	Calvo Sisman	1990-11-12	H	Mozo	1400
4	Margarita	Rodríguez Garcés	1992-05-16	M	Secretaria	1325.5

```
select SEXO , count(*) as EMPLEADOS  
  from EMPLEADOS  
group by SEXO
```

SEXO	EMPLEADOS
H	2
M	2

```
select count(*) as EMPLEADOS  
  from EMPLEADOS  
group by SEXO
```

EMPLEADOS
2
2

```
select distinct SEXO  
  from EMPLEADOS
```

SEXO
H
M

# Orden y agregación

Ejemplos de funciones agregadas, **COUNT**, **SUM**, **MAX**, **MIN** y **AVG**:

- ♦ La función **COUNT** devuelve el número de tuplas o valores especificados en una consulta.
- ♦ Las funciones **SUM**, **MAX**, **MIN** y **AVG** se aplican a un conjunto o multiconjunto de valores numéricos.

```
SELECT SUM("EMPLEADO"."Sueldo"), MAX("EMPLEADO"."Sueldo"),  
MIN("EMPLEADO"."Sueldo"), AVG("EMPLEADO"."Sueldo")  
FROM "EMPLEADO"
```

	<b>sum</b> numeric	<b>max</b> numeric	<b>min</b> numeric	<b>avg</b> numeric
1	281000	55000	25000	35125.0000000000000000

# Orden y agregación

## Consulta 20

Visualizar la suma de los salarios de todos los empleados del departamento 'Investigación', así como el salario más alto, el salario más bajo, y el salario medio de este departamento;

```
SELECT SUM("EMPLEADO"."Sueldo"), MAX("EMPLEADO"."Sueldo"),  
MIN("EMPLEADO"."Sueldo"), AVG("EMPLEADO"."Sueldo")  
FROM "EMPLEADO" JOIN "DEPARTAMENTO" ON "Dno" =  
"NumeroDpto"  
WHERE "NombreDpto"='Investigacion';
```

	sum numeric	max numeric	min numeric	avg numeric
1	133000	40000	25000	33250.0000000000000000

# Orden y agregación

## *Consulta 21.*

Recuperar el número total de empleados de la empresa:

```
SELECT COUNT (*)  
FROM “EMPLEADO”;
```

	count bigint
1	8



# Orden y agregación

## *Consulta 22.*

Recuperar el número de empleados del departamento 'investigación':

**SELECT COUNT (\*)**

**FROM "EMPLEADO" , "DEPARTAMENTO"**

**WHERE "Dno" = "NumeroDpto" AND "NombreDpto"='Investigacion';**

	count bigint
1	4

# Orden y agregación

## *Consulta 23-a.*

Contar el número de sueldos diferentes almacenados en la base de datos:

**SELECT COUNT** (distinct 'sueldo')

**FROM** "EMPLEADO";

	count bigint
1	1

# Orden y agregación

## *Consulta 23-b.*

Consulta anidada correlacionada con la función agregada: La siguiente consulta anidada recupera los nombres de todos los empleados que tienen dos o más subordinados:

```
SELECT "Apellido1", "Nombre"  
FROM "EMPLEADO"  
WHERE (SELECT COUNT (*)  
        FROM "SUBORDINADO"  
        WHERE "EMPLEADO"."Dni"=  
              "SUBORDINADO"."DniEmpleado") >= 2;
```

	Apellido1 text	Nombre text
1	Perez	Jose
2	Campos	Alberto

# Orden y agregación

## Consulta 24.

Consulta anidada correlacionada con la función agregada: La siguiente consulta anidada recupera los nombres de todos los empleados que tienen dos o más subordinados:

```
SELECT "Dno", COUNT(*), AVG("Sueldo")  
FROM "EMPLEADO"  
GROUP BY "Dno";
```

	Dno integer	count bigint	avg numeric
1	4	3	31000.0000000000000000
2	5	4	33250.0000000000000000
3	1	1	55000.0000000000000000

	Nombre text	Apellido1 text	Apellido2 text	Dni integer	FechaNac date	Direccion text	Sexo "char"	Sueldo numeric	SuperDni integer	Dno integer
1	Eduardo	Ochoa	Paredes	38665555	1937-11-10	Las Peña...	H	55000	[null]	1
2	Alicia	Jimenez	Celaya	99887777	1968-05-12	Gran Via, ...	M	25000	987654321	4
3	Luis	Pajares	Morera	37987987	1969-03-29	Enebras, ...	H	25000	987654321	4
4	Juana	Sainz	Oreja	37654321	1941-06-20	Cerquilla...	M	43000	888665555	4
5	Aurora	Oliva	Avezuela	53453453	1972-07-31	Anton, 6	M	25000	333445555	5
6	Alberto	Campos	Sastre	33445555	1955-12-08	Avda Rios...	H	40000	888665555	5
7	Fernando	Ojeda	Ordoñez	56884444	1962-09-15	Portillo, S...	M	38000	333445555	5
8	Jose	Perez	Perez	23456789	1965-09-01	Eloy I, 98	H	30000	333445555	5

# Orden y agregación: GROUP BY (dos atributos)

## Consulta 25.

Por cada proyecto, recuperar el número de proyecto, el nombre de proyecto y el número de empleados que trabajan en ese proyecto:

```
SELECT "NumProyecto", "NombreProyecto", COUNT(*)  
FROM "PROYECTO", "TRABAJA_EN"  
WHERE "NumProyecto" = "NumProy"  
GROUP BY "NumProyecto", "NombreProyecto";
```

	NumProyecto integer	NombreProyecto text	count bigint
1	20	Reorganizacion	3
2	1	ProductoX	2
3	2	ProductoY	3
4	10	Computacion	3
5	30	Comunicaciones	3
6	3	ProductoZ	2

# Orden, agregación

SQL Editor Graphical Query Builder

Previous queries

```
SELECT "PROYECTO"."NumProyecto", "PROYECTO"."NombreProyecto", COUNT(*) as NumEmpleados
FROM "PROYECTO", "TRABAJA_EN"
WHERE "TRABAJA_EN"."NumProy"="PROYECTO"."NumProyecto"
GROUP BY "PROYECTO"."NumProyecto", "PROYECTO"."NombreProyecto"
```

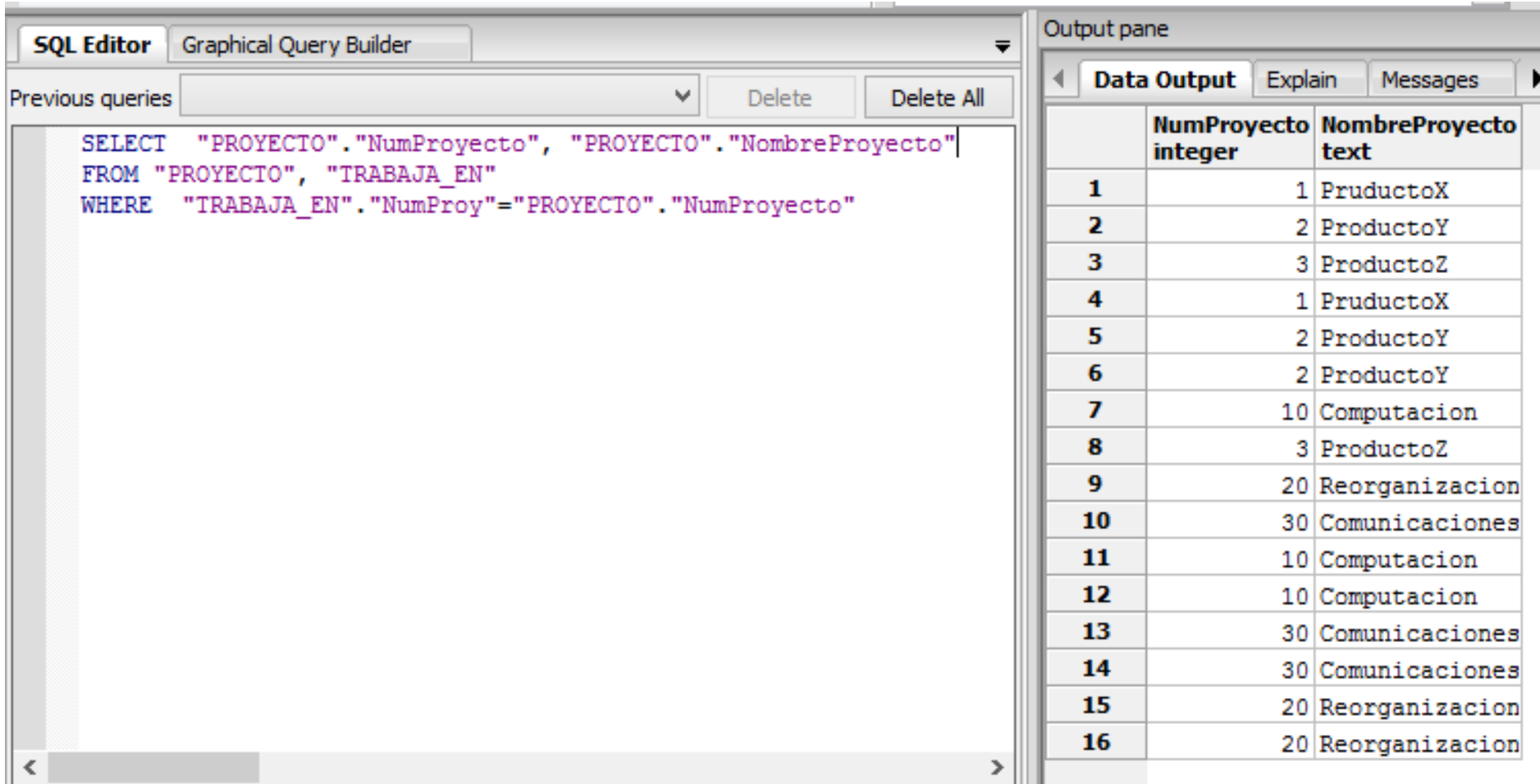
Output pane

Data Output Explain Messages History

	NumProyecto integer	NombreProyecto text	numempleados bigint
1	20	Reorganizacion	3
2	1	PruductoX	2
3	2	ProductoY	3
4	10	Computacion	3
5	30	Comunicaciones	3
6	3	ProductoZ	2

# Orden, agregación

Que sucede si no agrupo!!!!



The screenshot shows a SQL IDE interface. On the left, the 'SQL Editor' tab is active, displaying a query that joins 'PROYECTO' and 'TRABAJA\_EN' tables. The query selects 'NumProyecto' and 'NombreProyecto' from 'PROYECTO' where they match the 'NumProy' in 'TRABAJA\_EN'. On the right, the 'Output pane' shows the 'Data Output' tab with a table of 16 rows. The table has two columns: 'NumProyecto' (integer) and 'NombreProyecto' (text). The results show multiple rows for each project number, indicating a lack of grouping.

```
SELECT "PROYECTO"."NumProyecto", "PROYECTO"."NombreProyecto"
FROM "PROYECTO", "TRABAJA_EN"
WHERE "TRABAJA_EN"."NumProy"="PROYECTO"."NumProyecto"
```

	NumProyecto integer	NombreProyecto text
1	1	ProductoX
2	2	ProductoY
3	3	ProductoZ
4	1	ProductoX
5	2	ProductoY
6	2	ProductoY
7	10	Computacion
8	3	ProductoZ
9	20	Reorganizacion
10	30	Comunicaciones
11	10	Computacion
12	10	Computacion
13	30	Comunicaciones
14	30	Comunicaciones
15	20	Reorganizacion
16	20	Reorganizacion

# Orden y agregación: GROUP BY (dos atributos)

## Consulta 26.

Por cada proyecto en el que trabajan más de dos empleados, recuperar el número el nombre y número de empleados que trabajan para el:

```
SELECT "NumProyecto", "NombreProyecto", COUNT(*)  
FROM "PROYECTO", "TRABAJA_EN"  
WHERE "NumProyecto" = "NumProy"  
GROUP BY "NumProyecto", "NombreProyecto"  
HAVING COUNT(*) > 2;
```

*Estos grupos no se seccionan con HAVING, por lo tanto el GROUP BY solo actuaría sobre el resto de los grupos.*

	NombreProyecto text	NumProyecto integer	UbicacionProyecto text	NumDptoProyecto integer	DniEmpleado integer	NumProy integer	Horas numeric
1	ProductoX	1	Valencia	5	453453453	1	20.0
2	ProductoX	1	Valencia	5	123456789	1	32.5
3	ProductoY	2	Sevilla	5	333445555	2	10.0
4	ProductoY	2	Sevilla	5	453453453	2	20.0
5	ProductoY	2	Sevilla	5	123456789	2	7.5
6	ProductoZ	3	Madrid	5	333445555	3	10.0
7	ProductoZ	3	Madrid	5	666884444	3	40.0
8	Computacion	10	Gijon	4	999887777	10	10.0
9	Computacion	10	Gijon	4	333445555	10	10.0
10	Computacion	10	Gijon	4	987987987	10	35.0
11	Reorganizacion	20	Madrid	1	888665555	20	[null]
12	Reorganizacion	20	Madrid	1	333445555	20	10.0
13	Reorganizacion	20	Madrid	1	987654321	20	15.0
14	Comunicaciones	30	Gijon	4	987654321	30	20.0
15	Comunicaciones	30	Gijon	4	999887777	30	30.0
16	Comunicaciones	30	Gijon	4	987987987	30	5.0

	NumProyecto integer	NombreProyecto text	count bigint
1	20	Reorganizacion	3
2	30	Comunicaciones	3
3	2	ProductoY	3
4	10	Computacion	3



# Orden y agregación

## Consulta 27.

Por cada proyecto, recuperar el número, el nombre y la cantidad de empleados del departamento 5 que trabajan en dicho proyecto:

```
SELECT "NumProyecto", "NombreProyecto", COUNT(*)  
FROM "PROYECTO", "TRABAJA_EN" , "EMPLEADO"  
WHERE "NumProyecto" = "NumProy" AND  
      "Dni" = "DniEmpleado" AND "Dno" = 5  
GROUP BY "NumProyecto", "NombreProyecto"
```

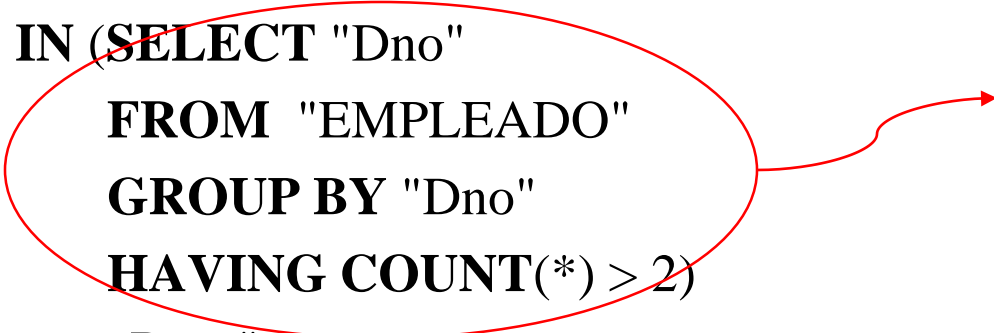
	<b>NumProyecto</b> integer	<b>NombreProyecto</b> text	<b>count</b> bigint
1	1	PruductoX	2
2	2	ProductoY	3
3	3	ProductoZ	2
4	10	Computacion	1
5	20	Reorganizacion	1

# Orden y agregación

## Consulta 28.

Por cada departamento que tiene más de 2 empleados, recuperar el número de departamento y el número de empleados que ganan mas de 40000:

```
SELECT "NumeroDpto", COUNT(*)  
FROM "DEPARTAMENTO", "EMPLEADO"  
WHERE "NumeroDpto" = "Dno" AND "Sueldo" > 40000 AND  
      "Dno" IN (SELECT "Dno"  
                  FROM "EMPLEADO"  
                  GROUP BY "Dno"  
                  HAVING COUNT(*) > 2)  
GROUP BY "NumeroDpto";
```



	Dno integer
1	4
2	5

	NumeroDpto integer	count bigint
1	4	1

# Ejemplos de orden y agregación

**Ejemplo 29** (El empleado que más horas trabaja en un proyecto)

```
SELECT "EMPLEADO"."Nombre", "TRABAJA_EN"."Horas"  
FROM "EMPLEADO", "TRABAJA_EN"  
WHERE "EMPLEADO"."Dni"="TRABAJA_EN"."DniEmpleado" AND  
"TRABAJA_EN"."Horas" IS NOT NULL  
ORDER BY "TRABAJA_EN"."Horas" DESC Limit 1;
```

**Ejemplo 32** (Por cada proyecto recuperar el número de proyecto, el nombre del proyecto y el numero de empleados que trabajan en el proyecto)

```
SELECT "PROYECTO"."NumProyecto", "PROYECTO"."NombreProyecto",  
COUNT(*)  
FROM "PROYECTO", "TRABAJA_EN"  
WHERE "TRABAJA_EN"."NumProy"="PROYECTO"."NumProyecto"  
GROUP BY "PROYECTO"."NumProyecto", "PROYECTO"."NombreProyecto"
```

# Vistas

**CREATE VIEW** *nombre* **AS** SELECT...;

Dan un nombre a una consulta, permiten usarla como tabla

Útil para reutilizar consultas y evitar ejecutarlas varias veces

Pueden configurarse para que se almacenen en disco

Ejemplos:

```
CREATE VIEW Contactos_Usuario AS
```

```
SELECT u1.nick, u2.nombre FROM Usuario AS u1, Usuario as u2
```

```
WHERE (u1.nick, u2.nick) IN
```

```
((SELECT usuario1, usuario2 FROM Contacto)
```

```
UNION (SELECT usuario2, usuario1 FROM Contacto));
```

```
SELECT nombre FROM Contactos_Usuario WHERE nick = 'pepe';
```