

# Fundamentos de Aplicaciones y Servicios Telemáticos

2º Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Ingeniería Telemática

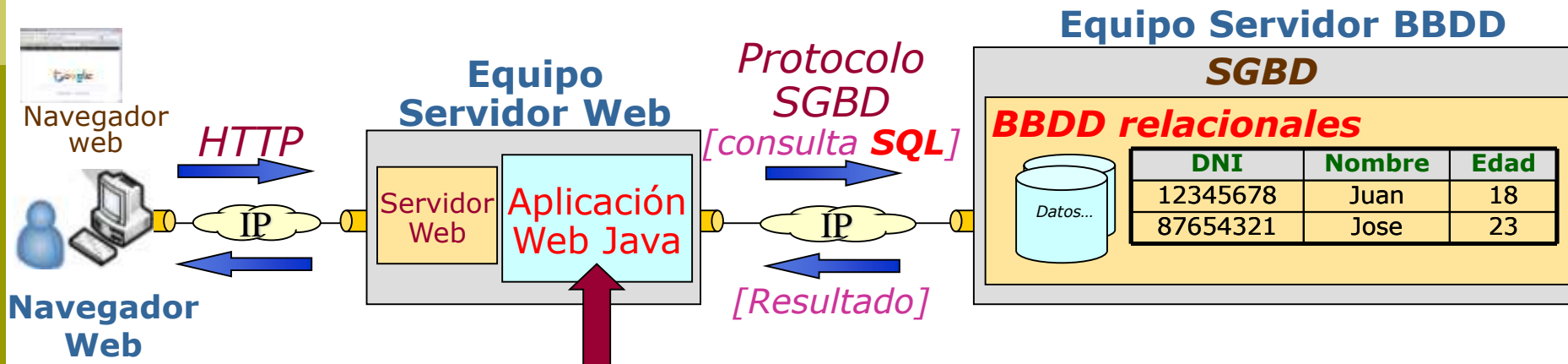


## Tema/Práctica 04

**Programación Web Dinámica  
(con interpretación en el Servidor)  
*con soporte de BBDD***

# Objetivo

- Aprender a **Diseñar**:
  - **Aplicaciones Web** “dinámicas en *Servidor*” (**Java**)
  - **con acceso a BBDD relacionales** (consultas en **SQL**).



# Contenido del Tema (Teoría + Práctica)

---

## 1. Introducción

- Conceptos BBDD y SGBD
- Tipos de BBDD
- Estructura BBDD relacionales

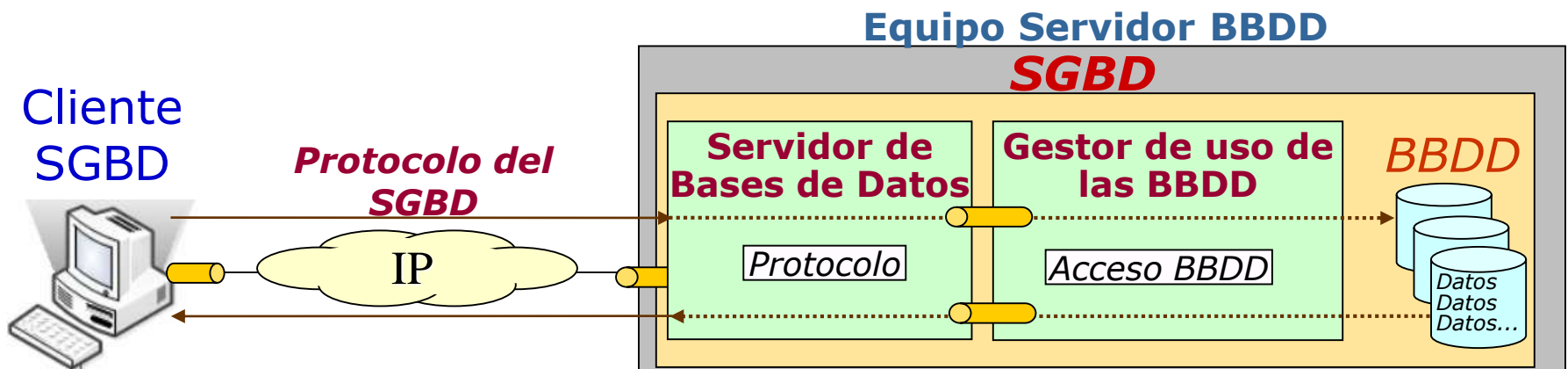
## 2. Lenguaje SQL

## 3. Acceso a BBDD desde Aplicaciones Web

- ▣ Aplicaciones Web **Java**: Conector JDBC (API)

# 1. Introducción: BBDD y SGBD

- ❑ **Objetivo BBDD**: almacenamiento/acceso eficiente de datos.
- ❑ **SGBD** (Sistema Gestor de Bases de Datos): programa que contiene los datos y controla su acceso.
  - ❑ Implementaciones SGBD: **PostgreSQL**, MariaDB, Oracle, ...
- ❑ Un SGBD consta de:
  - Bases de Datos (**BBDD**): colección datos (ficheros, memoria,...).
  - **Gestor BBDD**: accede y modifica los datos de las BBDD.
  - **Servidor BBDD**: permite acceso remoto a BBDD (protocolo red).
    - Cada SGBD tiene su propio protocolo (no normalizado): PostgreSQL Frontend/Backend Protocol, MariaDB Protocol, ...



# 1. Introducción: Tipos de BBDD

Según cómo se organicen los datos (**Modelos de datos**):

➔ **BD Relacional: Tablas** ➔

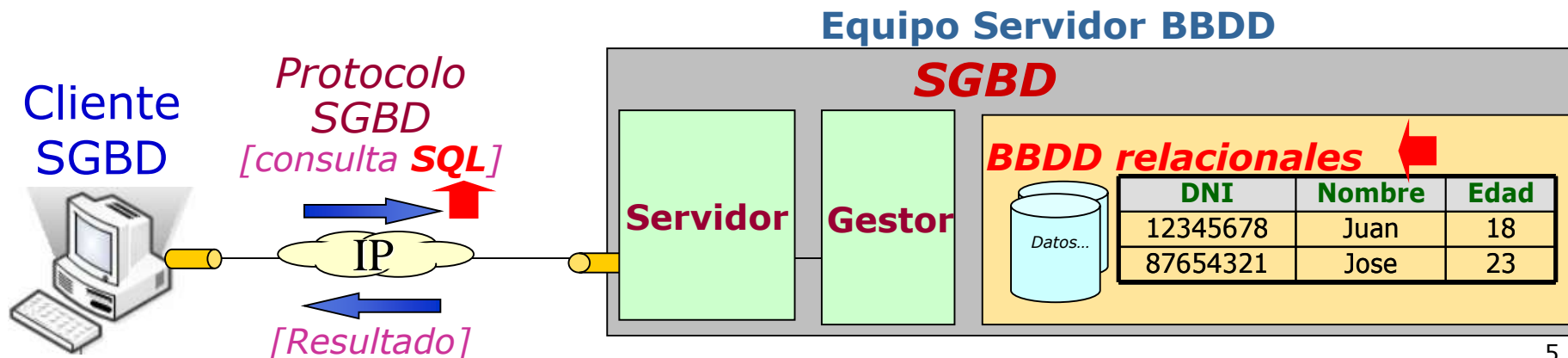
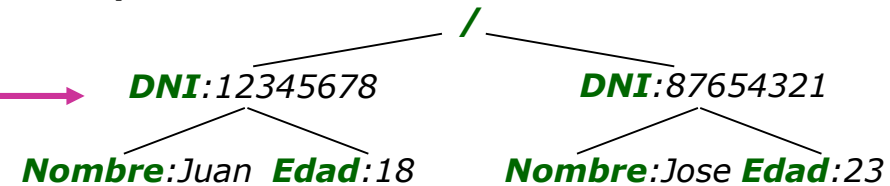
□ Tipo BBDD más simple y extendido:

➔ **lenguaje** habitual de consulta **SQL**

(e.g. "SELECT DNI FROM tabla;").

DNI	Nombre	Edad
12345678	Juan	18
87654321	Jose	23

■ **Otras BD: Semiestructurada** ➔  
(estructura arbórea e.g. XML), ...



# 1. Introducción: Estructura BBDD relacionales

- Estructura básica de las **BBDD relacionales**:
  - BBDD relacional: una/varias **tablas** (o relaciones), con su **nombre**.
  - Cada tabla:
    - **Filas**: **registros** (o tuplas)
    - **Columnas**: **campos** (o atributos)
      - Identificadas por un **nombre** (en la fila de cabecera).
      - En su **definición** incluyen: "tipo de datos" (entero, ...) de sus **celdas**
    - Contenido de una **Celda**: **instancia**.
  - **Clave o superclave**: **columna/s** en la que **todas las filas** tienen un **valor distinto o unívoco** (e.g. DNI). Se usa **para relacionar las tablas**.
    - Clave **primaria**: **clave** usada **para la indexación**. No admite valores NULL.
  - **Consulta** (o vista): **recuperación de información de las tablas** que cumpla ciertas condiciones (e.g.: **SELECT Nombre WHERE Edad=18**)

TABLA/RELACION "**tabla**"

Campos/Atributos		
integer	String	integer
DNI	Nombre	Edad
12345678	Juan	18
87654321	Jose	23

Cabecera: Campos/Atributos {



Registros {

Instancia

Clave primaria

## 2. Lenguaje SQL

- **SQL** (*Structured Query Language*): consultar BBDD relacionales
  - Estándar SQL:2011 (ISO/IEC 9075:2011).
  - Soportado por mayoría SGBD (PostgreSQL, MySQL, Oracle, ...).
- **Sentencias** (órdenes) SQL:

Tipo	<i>Sentencia</i> SQL	Funcionalidad
<b>Crear /Borrar tablas</b>	CREATE table	Crear <i>tabla</i>
	ALTER table	Modificar tabla
	DROP table	Eliminar tabla
 <b>Consultar/ Modificar datos</b>	INSERT	Insertar <i>filas</i>
	DELETE	Eliminar filas
	UPDATE	Cambiar <i>valor celdas</i>
	 SELECT	Obtener valor celdas
<b>Gestión</b>	GRANT	Crear <i>usuario</i> SGBD <b>y</b> asignarle <i>privilegios</i>
	REVOKE	Eliminar <i>usuario</i> SGBD o quitarle privilegios
	...	...

## 2. Lenguaje SQL: crear y borrar tabla

### ■ **Sentencias** (órdenes) **SQL** para crear y borrar una tabla

#### ■ CREATE TABLE

```
CREATE TABLE empleados(
  num_emp int4 NOT NULL,
  nombre varchar(50),
  fechanac date,
  salario integer,
  PRIMARY KEY (num_emp)
);
```


TABLA "**empleados**"

*Cabecera* {

*Registros* {

*Campos/Atributos*

<i>int4</i>	<i>varchar(50)</i>	<i>date</i>	<i>integer</i>
<b>num_emp</b>	<b>nombre</b>	<b>fechanac</b>	<b>salario</b>

 *Clave primaria*

#### ■ DROP TABLE

```
DROP TABLE usuarios;
```

*¡Ya no existe la tabla!*



## 2. Lenguaje SQL: insertar y borrar filas

### ■ Sentencias (órdenes) SQL para insertar y borrar filas

#### ■ INSERT INTO Tabla

Diagram illustrating the SQL INSERT INTO statement structure:

```

INSERT INTO empleados (num_emp, nombre, fechanac, salario)
VALUES
(1, 'J. Pérez', '1/1/1975', 500),
(2, 'A. García', '31/12/1980', 600);
    
```

Annotations in the diagram:

- Tabla**: Points to `empleados`.
- Campos**: Points to the column list `(num_emp, nombre, fechanac, salario)`.
- Valores**: Points to the data rows in the `VALUES` clause.

TABLA "**empleados**"

num_emp	nombre	fechanac	salario
1	J. Pérez	1/1/1975	500
2	A. García	31/12/1980	600

#### ■ DELETE FROM

```
DELETE FROM usuarios WHERE salario<550;
```

```
DELETE FROM usuarios;
```

*¡Sigue existiendo la tabla!*

## 2. Lenguaje SQL: modificar filas

### ■ Sentencias (órdenes) SQL para modificar filas

#### ■ UPDATE

num_emp	nombre	fechanac	salario
1	J. Pérez	1/1/1975	500
2	A. García	31/12/1980	600

```
UPDATE empleados SET salario=salario+50 WHERE salario>=600;
```



num_emp	nombre	fechanac	salario
1	J. Pérez	1/1/1975	500
2	A. García	31/12/1980	650

```
UPDATE empleados SET salario=900;
```



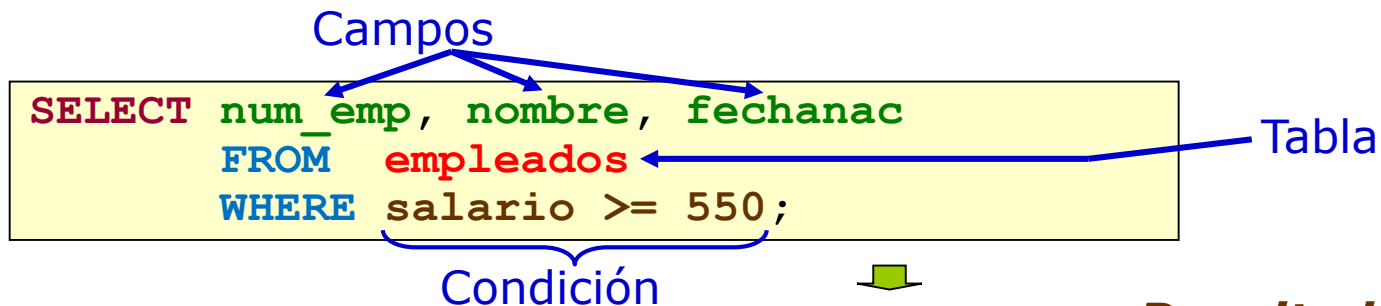
num_emp	nombre	fechanac	salario
1	J. Pérez	1/1/1975	900
2	A. García	31/12/1980	900

## 2. Lenguaje SQL: mostrar filas

### ■ Sentencias (órdenes) SQL para mostrar filas

#### ■ SELECT

num_emp	nombre	fechanac	salario
1	J. Pérez	1/1/1975	500
2	A. García	31/12/1980	600



**Resultado:**

num_emp	nombre	fechanac
2	A. García	31/12/1980

## 2. Lenguaje SQL: ejemplo spj 1

```
CREATE TABLE S (  
  ids INTEGER,  
  noms CHAR(20) NOT NULL,  
  estado CHAR(25),  
  ciudad CHAR(25) NOT NULL,  
  PRIMARY KEY (ids)  
);
```

*Suministradores (S)*

```
CREATE TABLE P (  
  idp INTEGER,  
  nomp CHAR(20) NOT NULL,  
  color CHAR(10) NOT NULL,  
  peso INTEGER NOT NULL,  
  ciudad CHAR(25) NOT NULL,  
  PRIMARY KEY (idp)  
);
```

*Piezas (P)*

```
CREATE TABLE J (  
  idj INTEGER,  
  nomj CHAR(20) NOT NULL,  
  ciudad CHAR(25) NOT NULL,  
  PRIMARY KEY (idj)  
);
```

*Proyectos (J = Jobs)*

```
CREATE TABLE SPJ (  
  ids INTEGER REFERENCES S (ids),  
  idp INTEGER REFERENCES P (idp),  
  idj INTEGER REFERENCES J (idj),  
  cantidad INTEGER,  
  PRIMARY KEY (ids, idp, idj)  
);
```

*Cantidades suministradas:  
cantidad de piezas idp del suministrador ids para el proyecto idj*

## 2. Lenguaje SQL: ejemplo spj 2

**select \* from s;**

ids	noms	estado	ciudad
1	Suministros 01		Madrid
2	Suministros 02		Londres
3	Suministros 03		Sevilla
4	Suministros 04		Londres
5	Suministros 05		París
6	Suministros 06		New York
7	Suministros 07		Atenas
8	Suministros 08		Sevilla
9	Suministros 09		Roma
10	Suministros 10		Múnich

**select \* from p;**

idp	nomp	color	peso	ciudad
1	Pieza 01	gris	100	Granada
2	Pieza 02	rojo	50	Madrid
3	Pieza 03	verde	20	Atenas
4	Pieza 04	azul	75	New York
5	Pieza 05	rojo	20	Londres
6	Pieza 06	celeste	10	Sevilla
7	Pieza 07	naranja	80	París
8	Pieza 08	rosa	40	Sevilla
9	Pieza 09	marrón	10	Londres
10	Pieza 10	rojo	20	Roma

**select \* from j;**

idj	nomj	ciudad
1	Edificio 01	Almería
2	Edificio 02	Sevilla
3	Edificio 03	Madrid
4	Edificio 04	Granada
5	Edificio 05	Londres
6	Edificio 06	Jaén
7	Edificio 07	Málaga
8	Edificio 08	Huelva
9	Edificio 09	Córdoba
10	Edificio 10	Cádiz
11	Edificio 11	Londres

**select \* from spj;**

ids	idp	idj	cantidad
1	8	3	300
1	5	3	350
1	2	4	400
1	7	1	500
2	7	10	400
2	5	5	500
3	2	4	350
3	2	1	200
4	2	7	550
4	9	5	700
5	1	5	400
5	2	8	300
5	3	7	350
5	4	3	500
5	5	9	400
5	6	1	500
5	7	2	400
5	8	8	600
5	9	7	500
5	10	5	400
6	4	8	200
7	5	11	300
7	9	11	800
8	7	4	900
9	1	8	800
10	1	8	700

## 2. Lenguaje SQL: ejemplo spj 3

**Nombres de proyectos a los que ha suministrado el suministrador con ids='4'**

```
select * from spj where ids='4';
```

ids	idp	idj	cantidad
4	2	7	550
4	9	5	700

*en dos pasos*

```
select * from j where idj='5' or idj='7';
```

idj	nomj	ciudad
5	Edificio 05	Londres
7	Edificio 07	Málaga

*Producto cartesiano*

```
select * from j, spj where ids='4';
```

```
select * from j;
```

idj	nomj	ciudad
1	Edificio 01	Almería
2	Edificio 02	Sevilla
3	Edificio 03	Madrid
4	Edificio 04	Granada
5	Edificio 05	Londres
6	Edificio 06	Jaén
7	Edificio 07	Málaga
8	Edificio 08	Huelva
9	Edificio 09	Córdoba
10	Edificio 10	Cádiz
11	Edificio 11	Londres

```
select * from spj;
```

ids	idp	idj	cantidad
1	8	3	300
1	5	3	350
1	2	4	400
1	7	1	500
2	7	10	400
2	5	5	500
3	2	4	350
3	2	1	200
4	2	7	550
4	9	5	700
5	1	5	400
5	2	8	300
5	3	7	350
5	4	3	500
5	5	9	400
5	6	1	500
5	7	2	400
5	8	8	600
5	9	7	500
5	10	5	400
6	4	8	200
7	5	11	300
7	9	11	800
8	7	4	900
9	1	8	800
10	1	8	700

## 2. Lenguaje SQL: ejemplo spj 4

### Producto cartesiano

```
select * from j,spj where ids='4';
```

idj	nomj	ciudad	ids	idp	idj	cantidad
1	Edificio 01	Almería	4	2	7	550
1	Edificio 01	Almería	4	9	5	700
2	Edificio 02	Sevilla	4	2	7	550
2	Edificio 02	Sevilla	4	9	5	700
3	Edificio 03	Madrid	4	2	7	550
3	Edificio 03	Madrid	4	9	5	700
4	Edificio 04	Granada	4	2	7	550
4	Edificio 04	Granada	4	9	5	700
5	Edificio 05	Londres	4	2	7	550
5	Edificio 05	Londres	4	9	5	700
6	Edificio 06	Jaén	4	2	7	550
6	Edificio 06	Jaén	4	9	5	700
7	Edificio 07	Málaga	4	2	7	550
7	Edificio 07	Málaga	4	9	5	700
8	Edificio 08	Huelva	4	2	7	550
8	Edificio 08	Huelva	4	9	5	700
9	Edificio 09	Córdoba	4	2	7	550
9	Edificio 09	Córdoba	4	9	5	700
10	Edificio 10	Cádiz	4	2	7	550
10	Edificio 10	Cádiz	4	9	5	700
11	Edificio 11	Londres	4	2	7	550
11	Edificio 11	Londres	4	9	5	700

Solo interesan  
estas 2 líneas

```
select * from j;
```

idj	nomj	ciudad
1	Edificio 01	Almería
2	Edificio 02	Sevilla
3	Edificio 03	Madrid
4	Edificio 04	Granada
5	Edificio 05	Londres
6	Edificio 06	Jaén
7	Edificio 07	Málaga
8	Edificio 08	Huelva
9	Edificio 09	Córdoba
10	Edificio 10	Cádiz
11	Edificio 11	Londres

```
select * from spj;
```

ids	idp	idj	cantidad
1	8	3	300
1	5	3	350
1	2	4	400
1	7	1	500
2	7	10	400
2	5	5	500
3	2	4	350
3	2	1	200
4	2	7	550
4	9	5	700
5	1	5	400
5	2	8	300
5	3	7	350
5	4	3	500
5	5	9	400
5	6	1	500
5	7	2	400
5	8	8	600
5	9	7	500
5	10	5	400
6	4	8	200
7	5	11	300
7	9	11	800
8	7	4	900
9	1	8	800
10	1	8	700

## 2. Lenguaje SQL: ejemplo spj 5

Nombres de proyectos a los que ha suministrado el suministrador con ids='4'

Producto cartesiano

Referencia

```
select * from j,spj where ids='4' and j.idj=spj.idj;
```

idj	nomj	ciudad	ids	idp	idj	cantidad
5	Edificio 05	Londres	4	9	5	700
7	Edificio 07	Málaga	4	2	7	550

Nombres (y ciudad) de proyectos a los que ha suministrado el suministrador con ids='4'

```
select nomj,ciudad from j,spj where ids='4' and j.idj=spj.idj;
```

nomj	ciudad
Edificio 05	Londres
Edificio 07	Málaga

```
select * from j;
```

idj	nomj	ciudad
1	Edificio 01	Almería
2	Edificio 02	Sevilla
3	Edificio 03	Madrid
4	Edificio 04	Granada
5	Edificio 05	Londres
6	Edificio 06	Jaén
7	Edificio 07	Málaga
8	Edificio 08	Huelva
9	Edificio 09	Córdoba
10	Edificio 10	Cádiz
11	Edificio 11	Londres

```
select * from spj;
```

ids	idp	idj	cantidad
1	8	3	300
1	5	3	350
1	2	4	400
1	7	1	500
2	7	10	400
2	5	5	500
3	2	4	350
3	2	1	200
4	2	7	550
4	9	5	700
5	1	5	400
5	2	8	300
5	3	7	350
5	4	3	500
5	5	9	400
5	6	1	500
5	7	2	400
5	8	8	600
5	9	7	500
5	10	5	400
6	4	8	200
7	5	11	300
7	9	11	800
8	7	4	900
9	1	8	800
10	1	8	700



## 2. Lenguaje SQL: ejemplo spj 6

**Nombres de proyectos a los que ha suministrado el suministrador con ids='4'**

*consulta en spj*

```
select idj from spj where spj.ids='4';
idj
-----
7
5
```

*usando IN y una subconsulta en spj*

```
select nomj from j where idj in (select idj from spj where spj.ids='4');
nomj
-----
Edificio 05
Edificio 07
```

```
select * from j;
```

idj	nomj	ciudad
1	Edificio 01	Almería
2	Edificio 02	Sevilla
3	Edificio 03	Madrid
4	Edificio 04	Granada
5	Edificio 05	Londres
6	Edificio 06	Jaén
7	Edificio 07	Málaga
8	Edificio 08	Huelva
9	Edificio 09	Córdoba
10	Edificio 10	Cádiz
11	Edificio 11	Londres

```
select * from spj;
```

ids	idp	idj	cantidad
1	8	3	300
1	5	3	350
1	2	4	400
1	7	1	500
2	7	10	400
2	5	5	500
3	2	4	350
3	2	1	200
4	2	7	550
4	9	5	700
5	1	5	400
5	2	8	300
5	3	7	350
5	4	3	500
5	5	9	400
5	6	1	500
5	7	2	400
5	8	8	600
5	9	7	500
5	10	5	400
6	4	8	200
7	5	11	300
7	9	11	800
8	7	4	900
9	1	8	800
10	1	8	700

## 2. Lenguaje SQL: ejemplo spj 7

```
select nomj from j,spj where j.idj=spj.idj and spj.ids='1';
      nomj
-----
Edificio 01
Edificio 03
Edificio 03
Edificio 04
```

**Nombres de proyectos a los que ha suministrado el suministrador con ids='1'**

*usando DISTINCT*

```
select distinct nomj from j,spj where j.idj=spj.idj and spj.ids='1';
      nomj
-----
Edificio 04
Edificio 03
Edificio 01
```

*usando IN y una subconsulta en spj*

```
select nomj from j where idj in (select idj from spj where spj.ids='1');
      nomj
-----
Edificio 03
Edificio 04
Edificio 01
```

```
select * from j;
idj |      nomj      |      ciudad
-----+-----+-----
1 | Edificio 01    | Almeria
2 | Edificio 02    | Sevilla
3 | Edificio 03    | Madrid
4 | Edificio 04    | Granada
5 | Edificio 05    | Londres
6 | Edificio 06    | Jaén
7 | Edificio 07    | Málaga
8 | Edificio 08    | Huelva
9 | Edificio 09    | Córdoba
10 | Edificio 10    | Cádiz
11 | Edificio 11    | Londres
```

```
select * from spj;
ids | idp | idj | cantidad
-----+-----+-----+-----
1 | 8 | 3 | 300
1 | 5 | 3 | 350
1 | 2 | 4 | 400
1 | 7 | 1 | 500
2 | 7 | 10 | 400
2 | 5 | 5 | 500
3 | 2 | 4 | 350
3 | 2 | 1 | 200
4 | 2 | 7 | 550
4 | 9 | 5 | 700
5 | 1 | 5 | 400
5 | 2 | 8 | 300
5 | 3 | 7 | 350
5 | 4 | 3 | 500
5 | 5 | 9 | 400
5 | 6 | 1 | 500
5 | 7 | 2 | 400
5 | 8 | 8 | 600
5 | 9 | 7 | 500
5 | 10 | 5 | 400
6 | 4 | 8 | 200
7 | 5 | 11 | 300
7 | 9 | 11 | 800
8 | 7 | 4 | 900
9 | 1 | 8 | 800
10 | 1 | 8 | 700
```

## 2. Lenguaje SQL: ejemplo spj 8

```
select * from j,spj where j.idj=spj.idj order by j.idj;
```

idj	nomj	ciudad	ids	idp	idj	cantidad
1	Edificio 01	Almería	5	6	1	500
1	Edificio 01	Almería	3	2	1	200
1	Edificio 01	Almería	1	7	1	500
2	Edificio 02	Sevilla	5	7	2	400
3	Edificio 03	Madrid	1	8	3	300
3	Edificio 03	Madrid	1	5	3	350
3	Edificio 03	Madrid	5	4	3	500
4	Edificio 04	Granada	3	2	4	350
4	Edificio 04	Granada	1	2	4	400
4	Edificio 04	Granada	8	7	4	900
5	Edificio 05	Londres	5	1	5	400
5	Edificio 05	Londres	5	10	5	400
5	Edificio 05	Londres	2	5	5	500
5	Edificio 05	Londres	4	9	5	700
7	Edificio 07	Málaga	5	9	7	500
7	Edificio 07	Málaga	4	2	7	550
7	Edificio 07	Málaga	5	3	7	350
8	Edificio 08	Huelva	6	4	8	200
8	Edificio 08	Huelva	10	1	8	700
8	Edificio 08	Huelva	9	1	8	800
8	Edificio 08	Huelva	5	8	8	600
8	Edificio 08	Huelva	5	2	8	300
9	Edificio 09	Córdoba	5	5	9	400
10	Edificio 10	Cádiz	2	7	10	400
11	Edificio 11	Londres	7	9	11	800
11	Edificio 11	Londres	7	5	11	300

(26 rows)

*Las tablas j y spj tienen una columna en común, por lo tanto se puede utilizar:*

**natural join**

*La columna en común sólo aparece una vez*

```
select * from j natural join spj order by idj;
```

idj	nomj	ciudad	ids	idp	cantidad
1	Edificio 01	Almería	5	6	500
1	Edificio 01	Almería	3	2	200
1	Edificio 01	Almería	1	7	500
2	Edificio 02	Sevilla	5	7	400
3	Edificio 03	Madrid	1	8	300
3	Edificio 03	Madrid	1	5	350
3	Edificio 03	Madrid	5	4	500
4	Edificio 04	Granada	3	2	350
4	Edificio 04	Granada	1	2	400
4	Edificio 04	Granada	8	7	900
5	Edificio 05	Londres	5	1	400
5	Edificio 05	Londres	5	10	400
5	Edificio 05	Londres	2	5	500
5	Edificio 05	Londres	4	9	700
7	Edificio 07	Málaga	5	9	500
7	Edificio 07	Málaga	4	2	550
7	Edificio 07	Málaga	5	3	350
8	Edificio 08	Huelva	6	4	200
8	Edificio 08	Huelva	10	1	700
8	Edificio 08	Huelva	9	1	800
8	Edificio 08	Huelva	5	8	600
8	Edificio 08	Huelva	5	2	300
9	Edificio 09	Córdoba	5	5	400
10	Edificio 10	Cádiz	2	7	400
11	Edificio 11	Londres	7	9	800
11	Edificio 11	Londres	7	5	300

(26 rows)

**Datos de proyectos (j) a los que se ha suministrado algo (su idj está en spj), y lo que se ha suministrado (spj)**

## 2. Lenguaje SQL: ejemplo spj 9

```
select * from j natural join spj order by idj;
```

idj	nomj	ciudad	ids	idp	cantidad
1	Edificio 01	Almeria	5	6	500
1	Edificio 01	Almeria	3	2	200
1	Edificio 01	Almeria	1	7	500
2	Edificio 02	Sevilla	5	7	400
3	Edificio 03	Madrid	1	8	300
3	Edificio 03	Madrid	1	5	350
3	Edificio 03	Madrid	5	4	500
4	Edificio 04	Granada	3	2	350
4	Edificio 04	Granada	1	2	400
4	Edificio 04	Granada	8	7	900
5	Edificio 05	Londres	5	1	400
5	Edificio 05	Londres	5	10	400
5	Edificio 05	Londres	2	5	500
5	Edificio 05	Londres	4	9	700
7	Edificio 07	Málaga	5	9	500
7	Edificio 07	Málaga	4	2	550
7	Edificio 07	Málaga	5	3	350
8	Edificio 08	Huelva	6	4	200
8	Edificio 08	Huelva	10	1	700
8	Edificio 08	Huelva	9	1	800
8	Edificio 08	Huelva	5	8	600
8	Edificio 08	Huelva	5	2	300
9	Edificio 09	Córdoba	5	5	400
10	Edificio 10	Cádiz	2	7	400
11	Edificio 11	Londres	7	9	800
11	Edificio 11	Londres	7	5	300

(26 rows)

Las tabla j tiene filas cuya columna en común no aparece en spj. Para obtener también esas filas, se usa:

natural **left** join

```
select * from j natural left join spj order by idj;
```

idj	nomj	ciudad	ids	idp	cantidad
1	Edificio 01	Almeria	5	6	500
1	Edificio 01	Almeria	3	2	200
1	Edificio 01	Almeria	1	7	500
2	Edificio 02	Sevilla	5	7	400
3	Edificio 03	Madrid	1	8	300
3	Edificio 03	Madrid	1	5	350
3	Edificio 03	Madrid	5	4	500
4	Edificio 04	Granada	3	2	350
4	Edificio 04	Granada	1	2	400
4	Edificio 04	Granada	8	7	900
5	Edificio 05	Londres	5	1	400
5	Edificio 05	Londres	5	10	400
5	Edificio 05	Londres	2	5	500
5	Edificio 05	Londres	4	9	700
6	Edificio 06	Jaén			
7	Edificio 07	Málaga	5	9	500
7	Edificio 07	Málaga	4	2	550
7	Edificio 07	Málaga	5	3	350
8	Edificio 08	Huelva	6	4	200
8	Edificio 08	Huelva	10	1	700
8	Edificio 08	Huelva	9	1	800
8	Edificio 08	Huelva	5	8	600
8	Edificio 08	Huelva	5	2	300
9	Edificio 09	Córdoba	5	5	400
10	Edificio 10	Cádiz	2	7	400
11	Edificio 11	Londres	7	9	800
11	Edificio 11	Londres	7	5	300

(27 rows)

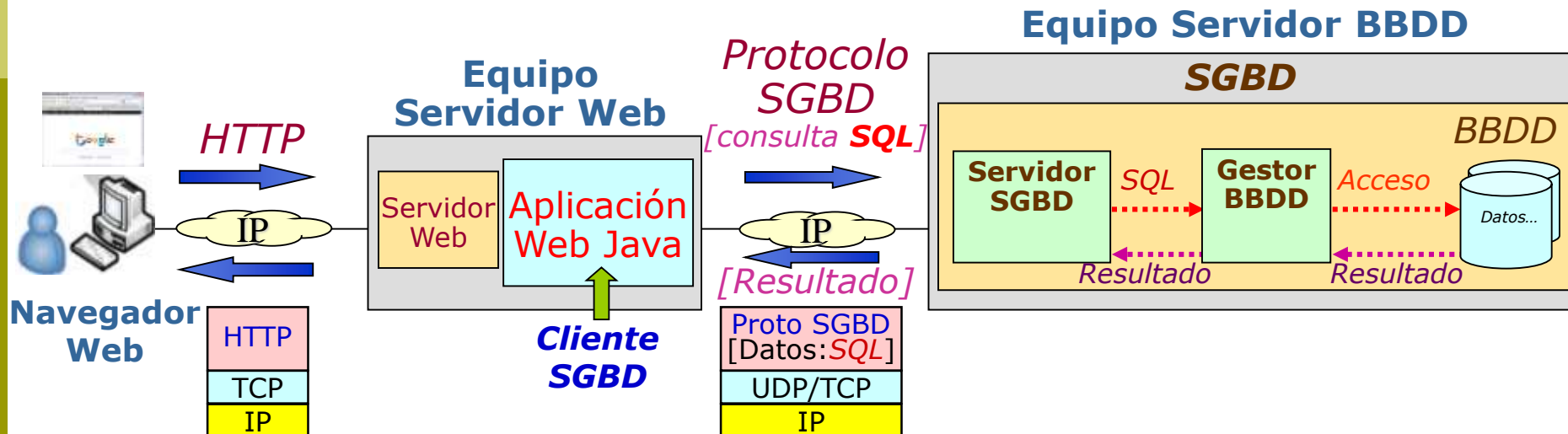
Datos de proyectos a los que se ha suministrado algo, y lo que se ha suministrado

Datos de proyectos a los que se ha suministrado algo, y lo que se ha suministrado, y **también** los demás proyectos

## 3. Acceso a BBDD desde Aplicaciones Web

### □ Aplicación Web: *cliente SGBD*

- Control de Acceso al SGBD: **usuario/clave** (cada usuario tiene sus privilegios de gestión y acceso a BBDD).
- Cliente SGBD debe conocer el protocolo de red para comunicarse con el **SGBD** (PostgreSQL, MariaDB, Oracle, ...)

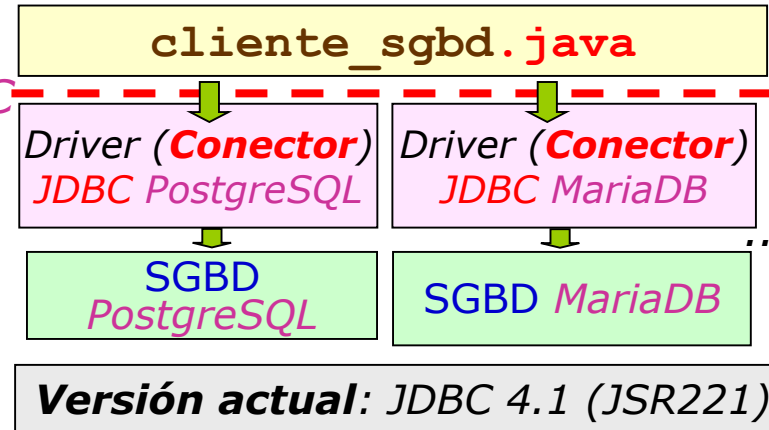


### 3. Acceso a BBDD desde Aplicaciones Web Java, JDBC

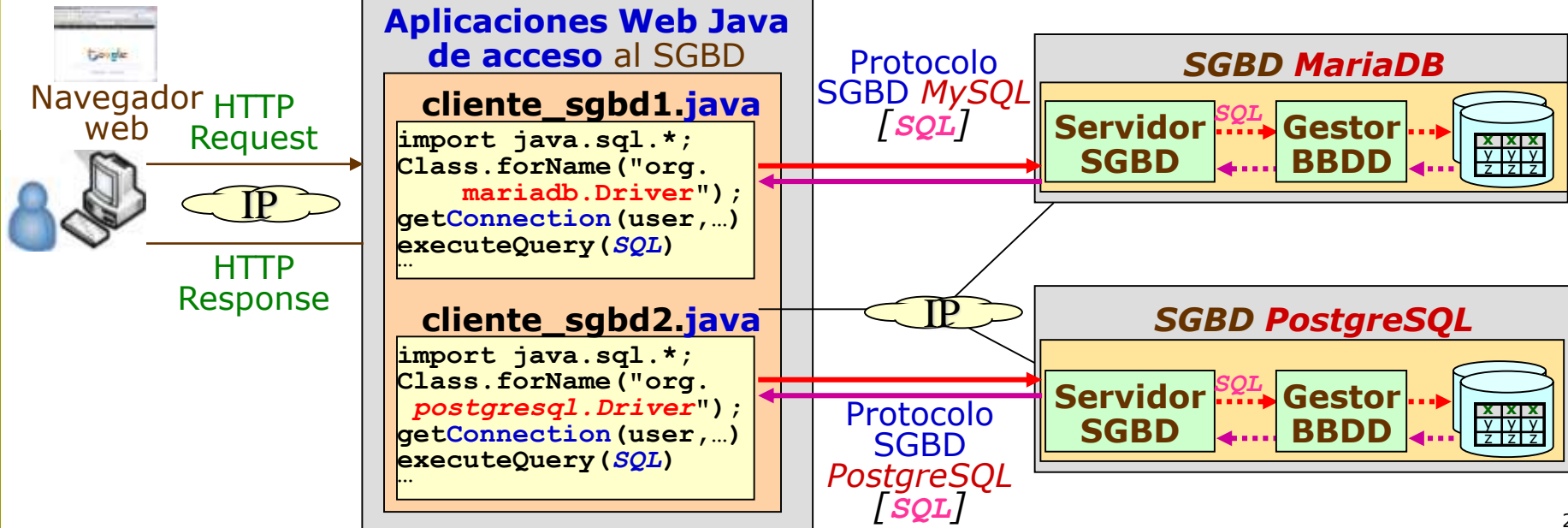
□ **JDBC** (**Java** DataBase **Connectivity**):  
permite acceder a cualquier SGBD: *API JDBC*

- **API**: funciones para Aplicación Java.
  - Abstracta: Común para cualquier SGBD (basta indicar cuál usar).
- Implementa protocolo de cada SGBD

□ **Aplicación Web Java con JDBC:**



#### Equipo Servidor Web



## 3. Acceso desde Aplicaciones Web Java (2): API JDBC

**Pasos del código web** para acceder a BBDD

[Falta la gestión de errores]

**A**  
Abrir conexión

Nº	Descripción	JDBC
1º	Cargar el driver (la 1ª vez, no necesario desde JDBC 4.0)	<code>Class.forName("sgbd.Driver")</code> <i>SGBD a usar</i>
2º	Conexión con el SGBD	<code>DriverManager.getConnection (</code>
3º	Selección de BBDD	<code>url,user,pass)</code> <i>Driver JDBC</i>

**B**  
Sentencia

**C**  
Consultar

4º	Envío de una consulta SQL al SGBD.	<div> <code>Connection</code>  <code>.createStatement()</code>  <code>Statement</code>  <code>.executeQuery(sql)</code>  <code>.executeUpdate(sql)</code> </div> <div> <code>Connection</code>  <code>.prepareStatement(sql)</code>  <code>PreparedStatement</code>  <code>.executeQuery()</code>  <code>.executeUpdate()</code> </div>
----	------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**D**  
Obtener/ Procesar Resultados

5º	Obtención metadatos del resultado (nº columnas, tipo de datos, ...)	<code>ResultSet.getMetaData()</code> <code>ResultSetMetaData.getColumnCount()</code> <i>Resultados</i> → <i>Metadatos</i> ...
6º	Obtención de los datos	<code>ResultSet.next()</code> <code>ResultSet.getString("...")</code> ...

**E**  
Cerrar conexión

7º	Liberación de los recursos	<code>ResultSet.close();</code> <code>Statement.close()</code>
8º	Cierre de la conexión	<code>Connection.close()</code>

## 3. Acceso desde Aplicaciones Web Java (3): API JDBC

### ❑ *Ejemplo del código* web JSP para acceder a BBDD:

```
<%@ page import="java.sql.*" %>
<% try {
    Class.forName("org.postgresql.Driver");
    String url = "jdbc:postgresql://localhost:5432/dit";
    String user = "dit";
    String pass = "dit";
    Connection conn = DriverManager.getConnection(url, user, pass);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM usuarios");
%>
```

```
<% while(rs.next())
{
    String name = rs.getString("name");
    String password = rs.getString("password");
%>
```

```
<%
    }
    rs.close();
    st.close();
    conn.close();
} catch (SQLException e) {
    out.println("Excepción SQL Exception: " + e.getMessage());
    e.printStackTrace();
}
%>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>JDBC a PostgreSQL</title>
    <meta charset="utf-8" />
</head>
<body>
    <table>
    <tr>
        <th>name</th><th>password</th>
</tr>
```

```
<tr>
    <td><%=name%></td>
    <td><%=password%></td>
</tr>
```

```
</table>
</body>
</html>
```



### 3. Ejemplo 1: Statement

```

<% try {
    Class.forName("org.postgresql.Driver");    //No es necesario hacerlo siempre.
    Connection conn=DriverManager.getConnection("jdbc:postgresql://localhost:5432/dit","dit","dit");
    String sql = "SELECT * FROM p WHERE ciudad='"+request.getParameter("ciudad")+"'";
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(sql); %>
<table>
  <tr>
    <th>Idp</th><th>Nombre</th><th>Color</th><th>Peso</th><th>Ciudad</th>
  </tr>
  <% while(rs.next()) { %>
    <tr>
      <td><%=rs.getInt("idp")%></td><td><%=rs.getString("nomp")%></td>
      <td><%=rs.getString("color")%></td>
      <td><%=rs.getInt("peso")%></td><td><%=rs.getString("ciudad")%></td>
    </tr>
  <%
    }
    rs.close();
    st.close();
    conn.close();
  } catch (SQLException e) {
    out.println("Excepción SQL Exception: " + e.getMessage());
    e.printStackTrace();
  } %>
</table>

```

*¡Cuidado!  
Inyección  
SQL*

*Poco eficiente  
crear la  
conexión en  
cada petición.  
Es mejor  
guardarla y  
reutilizarla.*

## 3. Ejemplo 2: PreparedStatement

```

<% try {
    Class.forName("org.postgresql.Driver");
    Connection conn=DriverManager.getConnection("jdbc:postgresql://localhost:5432/dit","dit","dit");
    String sql = "SELECT * FROM p WHERE ciudad=?";
    PreparedStatement st = conn.prepareStatement(sql);
    st.setString(1, request.getParameter("ciudad"));
    ResultSet rs = st.executeQuery(); %>
<table>
  <tr>
    <th>Idp</th><th>Nombre</th><th>Color</th><th>Peso</th><th>Ciudad</th>
  </tr>
  <% while(rs.next()) { %>
    <tr>
      <td><%=rs.getInt("idp")%></td><td><%=rs.getString("nomp")%></td>
      <td><%=rs.getString("color")%></td>
      <td><%=rs.getInt("peso")%></td><td><%=rs.getString("ciudad")%></td>
    </tr>
  <%
    }
    rs.close();
    st.close();
    conn.close();
  } catch (SQLException e) {
    out.println("Excepción SQL Exception: " + e.getMessage());
    e.printStackTrace();
  } %>
</table>

```

*Líneas modificadas.  
No permite la  
inyección SQL*

*Poco eficiente  
crear la  
conexión en  
cada petición.  
Es mejor  
guardarla y  
reutilizarla.*

### 3. Ejemplo 3: Referencia a DataSource

```

<% try {           //JNDI – Java Naming and Directory Interface
    InitialContext ctx = new InitialContext();
    DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/dit");
    Connection conn = ds.getConnection();
    String sql = "SELECT * FROM p WHERE ciudad=?";
    PreparedStatement st = conn.prepareStatement(sql);
    st.setString(1, request.getParameter("ciudad"));
    ResultSet rs = st.executeQuery(); %>
<table>
<tr> <th>Idp</th><th>Nombre</th><th>Color</th><th>Peso</th><th>Ciudad</th> </tr>
<% while(rs.next()) { %>
<tr>
    <td><%=rs.getInt("idp")%></td>
    <td><%=rs.getString("nomp")%></td>
    <td><%=rs.getString("color")%></td>
    <td><%=rs.getInt("peso")%></td>
    <td><%=rs.getString("ciudad")%></td>
</tr>
<%
}
rs.close();
st.close();
conn.close();
} catch (SQLException e) {
    out.println("Excepción SQL Exception: " + e.getMessage());
    e.printStackTrace();
} %>
</table>

```

Se reutiliza la conexión: "jdbc/dit" está definido en el servidor y en el web.xml

```

...../tomcat/conf/server.xml
<Context docBase="AppWeb" path="/AppWeb"
    reloadable="true" privileged="true"
    source="org.eclipse.jst.jee.server.AppWeb">
    <Resource name="jdbc/dit" auth="Container"
        type="javax.sql.DataSource"
        driverClassName="org.postgresql.Driver"
        url="jdbc:postgresql://127.0.0.1:5432/dit"
        username="dit" password="dit" maxTotal="20"
        maxIdle="10" maxWaitMillis="-1" />
</Context>

```

```

...../AppWeb/WEB-INF/web.xml
<resource-ref>
    <description>EJEMPLO</description>
    <res-ref-name>jdbc/dit</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>

```

## 3. Otras opciones

---

- ❑ Patrón de diseño **DAO** (Objeto de Acceso a Datos).
  - Objeto que sirve de interfaz entre la aplicación y uno o más dispositivos de almacenamiento de datos:
    - ❑ + flexibilidad: los datos pueden almacenarse en sitios distintos sin modificar la aplicación (no sabe dónde están)
    - ❑ pero + complejidad: hay que crear más clases y código.
  - Ejemplo: las consultas SQL se hacen en un bean de Java.
    - ❑ Hay que convertir el ResultSet obtenido en una lista de objetos.
  
- ❑ **JPA** (Java Persistence API) [[JSR 338 Java Persistence 2.1](#)]
  - Interactuar con la base de datos usando orientación a objetos.
  - Usa JPQL (Java Persistence Query Language) similar a SQL.
  - Se asocian clases Java a tablas SQL.
    - ❑ Cada fila de una tabla se asocia a un objeto de manera automática.
      - Al cambiar un objeto se actualiza la base de datos (y viceversa).
    - ❑ Las consultas devuelven ya listas de objetos, no ResultSet.

# Referencias

---

## □ SQL:

- *Estándar SQL:2011:* ISO/IEC 9075:2011
- *Uso práctico:* <http://www.w3schools.com/sql/>
  - ➔ □ *Sintaxis:* [http://www.w3schools.com/sql/sql\\_quickref.asp](http://www.w3schools.com/sql/sql_quickref.asp)
  - *Tipos datos:* [http://www.w3schools.com/sql/sql\\_datatypes.asp](http://www.w3schools.com/sql/sql_datatypes.asp)
- *Implementación SGBDs:* <http://troels.arvin.dk/db/rdbms/>

## □ JDBC:

- *Especific. (JSRs):* <http://jcp.org/>  
<http://www.oracle.com/technetwork/java/javaee/tech/>
- *API (JavaEE):* <http://docs.oracle.com/javaee/>
- *Document.:* <http://www.oracle.com/technetwork/java/javaee/documentation/>