

Bases de datos relacionales y SQL

José R.R. Viqueira

Centro Singular de Investigación en Tecnoloxías Intelixentes (CITIUS)
Rúa de Jenaro de la Fuente Domínguez,
15782 - Santiago de Compostela.

Despacho: 209

Telf: 881816463

Mail: jrr.viqueira@usc.es

Skype: jrviqueira

URL: <https://citius.gal/team/jose-ramon-rios-viqueira>

Curso 2023/2024

- **Modelo relacional y creación del esquema**
- **Consulta de datos**
 - ▷ **Lenguajes formales: Álgebra y Cálculo Relacional**
 - ▷ **Sintaxis básica**
 - ▷ **Valores nulos**
 - ▷ **Agregación**
 - ▷ **Subconsultas**
 - ▷ **Operadores de conjunto**
 - ▷ **Joins**
- **Transacciones**
- **Otras funcionalidades**

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

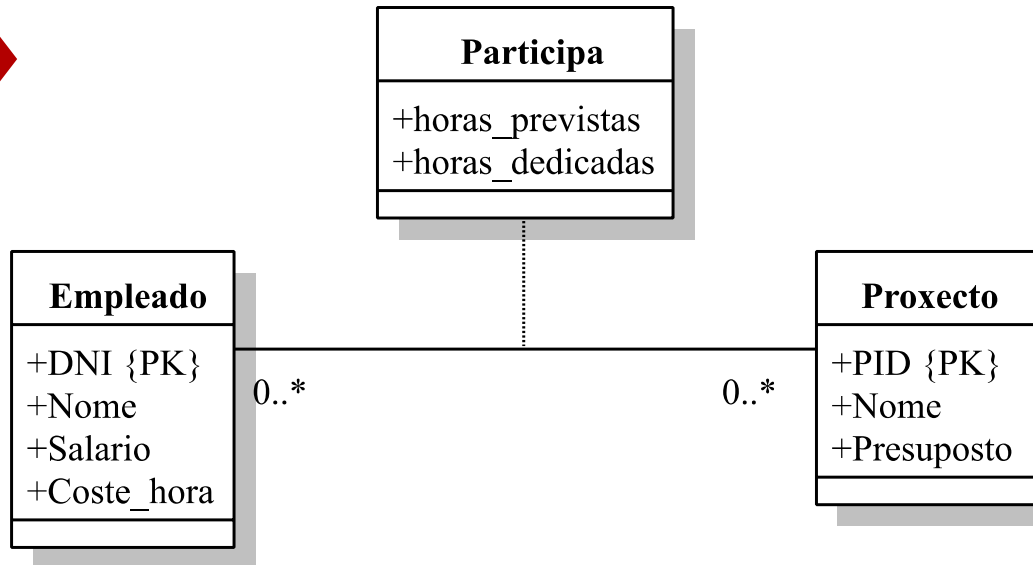
Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.



Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.



Tipos de Dato

- ▷ **char(n)**: tamaño fijo
- ▷ **varchar(n)**: tamaño variable
- ▷ **int, smallint**
- ▷ **numeric(p, d)**: punto fijo
- ▷ **real, double precision**: Punto flotante
- ▷ **float (n)**: mínima precisión de n dígitos
- ▷ **date, time, timestamp, interval**
- ▷ **Etc.**



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

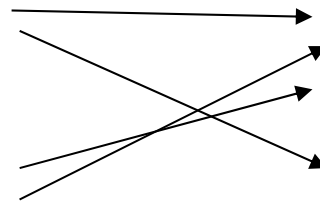
Transacciones

Etc.

TipoA

A1
A2
...

Relación



TipoB

B1
B2
...

Relación

A: TipoA	B: TipoB
A1	B1
A1	B12
...	...

Tipos no
necesariamente
distintos

NombreR

A1: T1	A2: T2	...	AN: TN
A1x	A2u	...	ANr
A1y	A2v	...	ANs
...

Esquema

Nombre de la relación + conjunto de pares (nombre de atributo, tipo de datos)

NombreR (A1:T1, A2:T2, ..., AN:TN)

Contenido

Subconjunto del producto cartesiano de los N tipos de datos

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

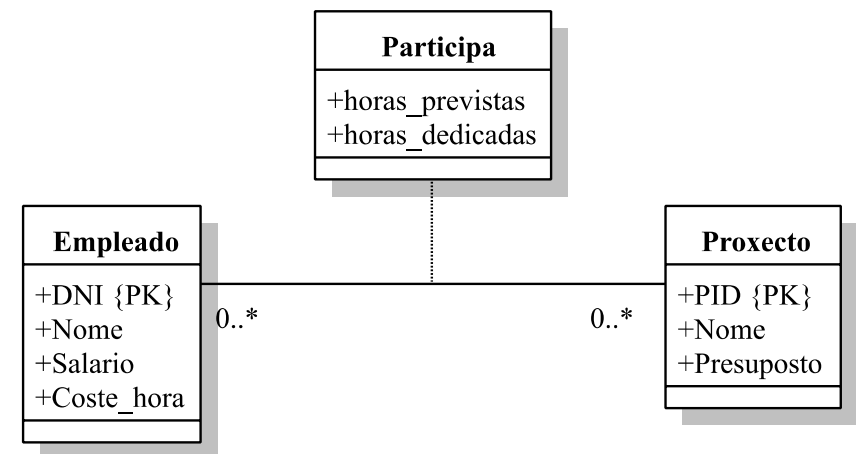
■ Lenguaje de definición de datos (DDL): Sintaxis básica

▷ **CREATE TABLE** Empleado (
dni char(8),
nome varchar(500) NOT NULL,
salario decimal(8,2),
coste_hora decimal(6,2) NOT NULL DEFAULT 40,
primary key (dni));

▷ **CREATE TABLE** Proxecto(
pid int primary key,
nome varchar(500) NOT NULL,
presuposto decimal(12, 2));

Esquema y
Restricciones

Se puede usar
UNIQUE (a1, a2, ...)
para definir otras
claves candidatas



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

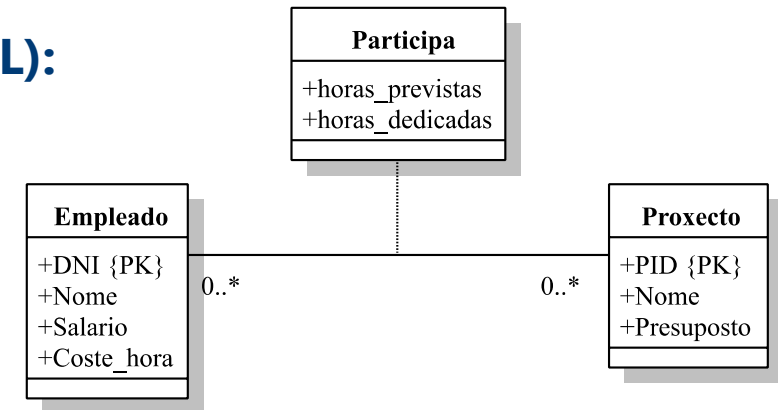
Transacciones

Etc.



■ Lenguaje de definición de datos (DDL): Restricciones

▷ **CREATE TABLE** participa (
 empleado **char**(8),
 proxecto **int**,
 horas_previstas **int NOT NULL**,
 horas_dedicadas **int DEFAULT 0**,
primary key (empleado, proxecto),
foreign key (empleado) **references** Empleado(dni)
 on delete restrict
 on update cascade,
foreign key (proxecto) **references** Proxecto(pid)
 on delete cascade
 on update cascade,
check (horas_previstas >= horas_dedicadas));



- restrict
- cascade
- set null
- set default

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

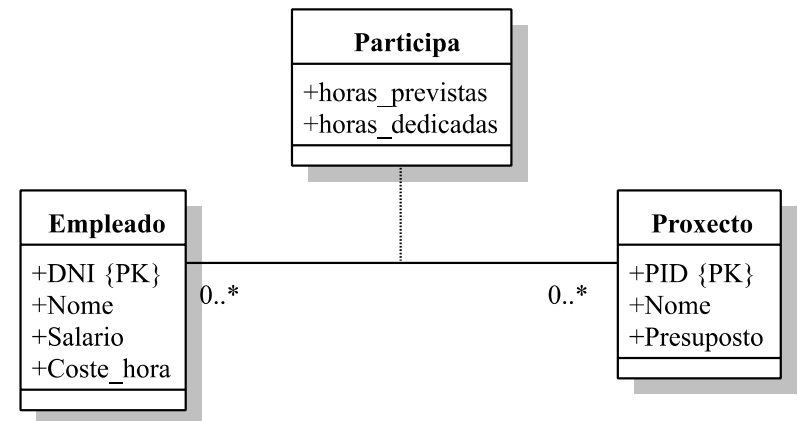
Transacciones

Etc.



■ Lenguaje de definición de datos (DDL): Eliminación y modificación del esquema

- ▷ DROP TABLE participa;
- ▷ ALTER TABLE Empleado
ADD email varchar(500);
- ▷ ALTER TABLE Empleado
DROP COLUMN email;
- ▷ ETC.



Modelo

Lenguajes
Formales



Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Álgebra relacional: Operaciones primitivas

- ▷ Selección: $\sigma_p(R)$
- ▷ Proyección (proyección generalizada): $\pi_{A_1, A_2, \dots, A_n}(R)$
- ▷ Producto cartesiano: $R_1 \times R_2$
- ▷ Unión: $R_1 \cup R_2$
- ▷ Diferencia: $R_1 - R_2$

■ Álgebra relacional: Operaciones derivadas

- ▷ Intersección: $R_1 \cap R_2$
- ▷ Join
 - Theta Join: $R_1 \bowtie_p R_2 \equiv \sigma_p(R_1 \times R_2)$
 - Natural Join: $R_1 \bowtie R_2$

Lenguaje
Procedimental

Modelo

Lenguajes
Formales



Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Cálculo de predicados (lógica de predicados)


▷ Constantes, variables, predicados, funciones

▷ Conectores: $\vee, \wedge, \rightarrow, \leftrightarrow, \neg$

▷ Cuantificadores: \forall, \exists

▷ Expresiones:

– $\text{numero}(x) \wedge \neg (\exists y(\text{numero}(y) \wedge \text{mayor}(x, y)))$



Lenguaje
Declarativo

Modelo

Lenguajes
Formales



Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Cálculo relacional (de tuplas)

- ▷ Expresiones: $\{t \mid P(t)\}$
- ▷ Variables referencian tuplas: t
- ▷ Función que accede al valor de un atributo: $t[A]$
- ▷ Constantes son valores de los tipos de datos
- ▷ Predicados
 - _ Relaciones: Empleado(t)
 - _ Otros predicados implementados en el sistema: $x > y$, $x = y$, etc.
- ▷ Conectores: \vee , \wedge , \rightarrow , \neg
- ▷ Cuantificadores: $\exists x(\text{Empleado}(x) \wedge x[\text{salario}] > y[\text{salario}])$

**Lenguaje
Declarativo**

Modelo

Lenguajes
Formales



Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Cálculo relacional (de tuplas)

▷ Expresiones Seguras

- _ Una expresión podría generar un resultado infinito
 - $\{ t \mid \neg \text{Empleado}(t) \}$
- _ El lenguaje debería de evitar que se puedan construir expresiones como esas
- _ Por ejemplo, forzar a que todas las variables tengan que estar referenciadas dentro de un predicado de relación sin negar.
 - $\{ t \mid \neg \text{Empleado}(t) \wedge \text{cliente}(t) \}$

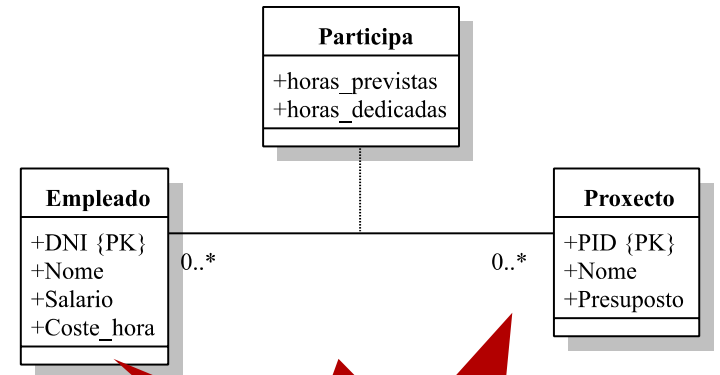
Modelo

Lenguajes
Formales



■ Procesamiento de consultas

- ▷ Nombre de los empleados con un salario mayor de 25000



$\{t[\text{nome}] \mid \text{empleado}(t) \wedge t[\text{salario}] > 25000\}$



$\pi_{\text{nombre}}(\sigma_{\text{salario} > 25000}(\text{Empleado}))$



π_{nombre}

$\sigma_{\text{salario} > 25000}$

empleado

Lenguaje
Declarativo



Lenguaje
Procedimental

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

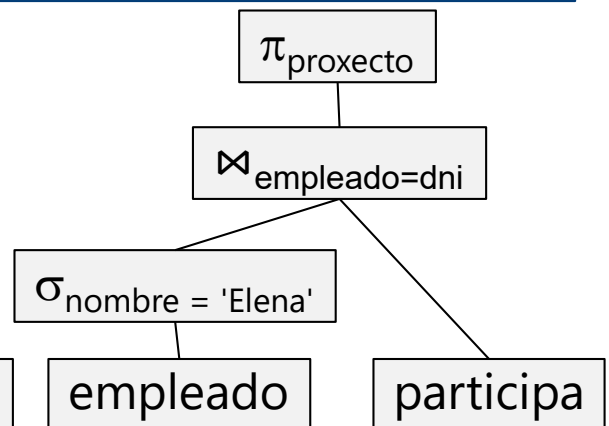
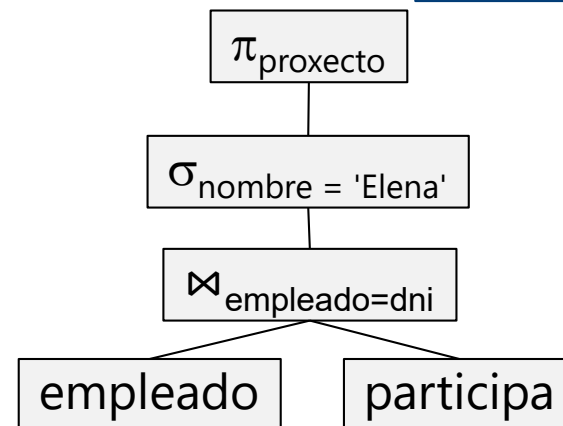
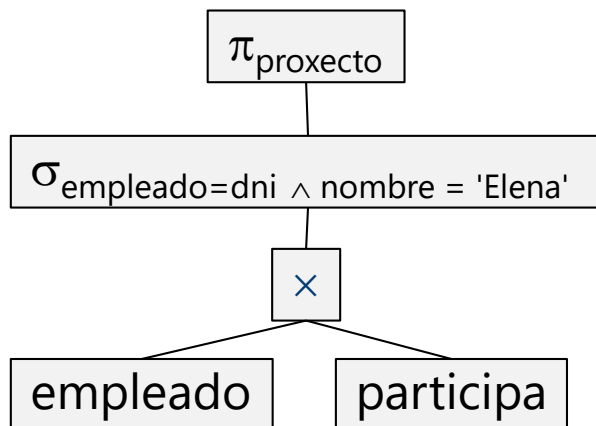
Transacciones

Etc.

■ Procesamiento de consultas

- ▷ Identificadores de proyecto de los proyectos en los que ha participado 'Elena'

```
{
  p[proxecto] |
  participa(p) ∧ empleado(e) ∧
  p[empleado]=e[dni] ∧
  e[nombre]='Elena'
}
```



Ejercicio

Empleado
+DNI {PK}
+Nome
+Salario
+Coste_hora

Participa
+horas_previstas
+horas_dedicadas

Proxecto
+PID {PK}
+Nome
+Presuposto

Modelo

Lenguajes
Formales

Sintaxis básica



- Sintaxis amigable (parecido a expresarlo en Inglés)
- Expresiones seguras del cálculo relacional de tuplas
- Generación de árboles de ejecución basados en álgebra relacional
- Uso de multi conjuntos en lugar de conjuntos

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

③ **SELECT** e_1, e_2, \dots, e_n

① **FROM** R_1 as r_1, R_2 as r_2, \dots, R_m as r_m

② **WHERE** p

$\pi_{e_1, e_2, \dots, e_n}$

$R_1 \times R_2 \times \dots \times R_m$

σ_p

$\{ e_1, e_2, \dots, e_n \mid$
 $R_1(r_1) \wedge R_2(r_2) \wedge \dots \wedge R_m(r_m)$
 $\wedge p(r_1, r_2, \dots, r_m) \}$

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

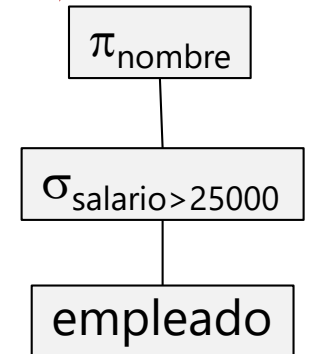
Etc.

- Nombre de los empleados con un salario mayor de 25000

```
SELECT nome
FROM empleado as e
WHERE e.salario > 25000
```

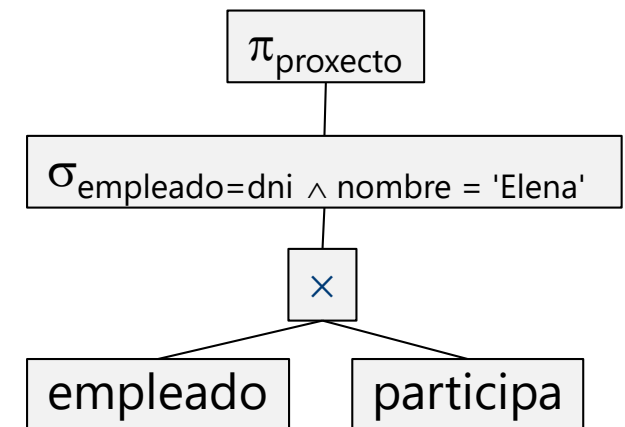
```
SELECT nome
FROM empleado
WHERE salario > 25000
```

Ejercicio



- Identificadores de los proyectos en los que ha participado 'Elena'

```
SELECT p.proyecto
FROM empleado as e, participa as p
WHERE e.dni=p.empleado and
      e.nombre='Elena'
```



Modelo

Lenguajes
Formales

Sintaxis básica



Nulos

Agregación

Subconsultas

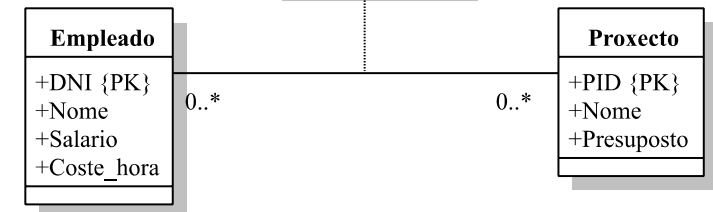
Oper. conjunto

Joins

Transacciones

Etc.

- Nombres y salarios de los empleados que ya han dedicado entre 20 y 40 horas al proyecto 'Tráfico'
- Ordena el resultado por salario de forma descendente y después por nombre de forma ascendente



- ③ **SELECT** e.nome, e.salario
- ① **FROM** empleado as e, participa pa, proxecto pr
- ② **WHERE** e.dni=pa.empleado
and pr.pid=pa.proyecto
and pa.horas_dedicadas **between** 20 **and** 40
and pr.nombre = 'Tráfico'
- ④ **ORDER BY** e.salario **DESC**, e.nombre **ASC**

Ya no es
una
relación,
es un
listado

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Nombres de los empleados que han participado en algún proyecto con un presupuesto mayor de 90000

Ejercicio

Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

```
SELECT e.nome
FROM empleado as e, participa pa,
      proxecto pr
WHERE e.dni=pa.empleado
      and pr.pid=pa.proxecto
      and pr.presuposto > 90000
```

Nome

Alfredo

Sofía

Alfredo

Ricardo

Elena

Ernesto

```
SELECT DISTINCT e.nome
FROM empleado as e, participa pa,
      proxecto pr
WHERE e.dni=pa.empleado
      and pr.pid=pa.proxecto
      and pr.presuposto > 90000
```

Nome

Alfredo

Sofía

Ricardo

Elena

Ernesto

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

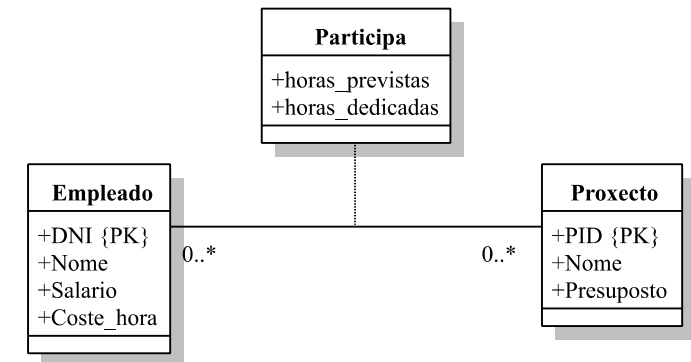
Etc.

■ Cada tipo de dato tiene un valor especial **NULL**

- ▷ Valor no conocido, todavía no insertado, etc.
- ▷ También el tipo de datos **Boolean** tiene el valor **NULL**. Las siguientes expresiones booleanas evalúan a **NULL**
 - _ **NULL = NULL**
 - _ **A = NULL**
- ▷ Clausula WHERE obtiene solo las tuplas para las que el predicado se evalúa a **TRUE**

Todos los atributos

```
SELECT *
FROM empleado
WHERE salario = NULL
```



```
SELECT *
FROM empleado
WHERE salario IS NULL
```

```
SELECT *
FROM empleado
WHERE salario IS NOT NULL
```

DDL

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Obtener la media del salario de todos los empleados

Función
de
Agregado

```
SELECT AVG(salario)
FROM empleado
```

Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

AVG = 37800

```
SELECT AVG(salario), MIN(salario), MAX(salario), COUNT(*), COUNT(DISTINCT coste_hora)
FROM empleado
```

```
SELECT AVG(salario), nome
FROM empleado
```

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Para cada proyecto, obtener su nombre y el número de empleados que participan

④ **SELECT** pr.nome, COUNT(*)
① **FROM** proxecto pr, participa pa
② **WHERE** pr.pid = pa.proxecto
③ **GROUP BY** pr.pid, pr.nome

Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
58325647	2	600	10
78878965	2	800	5
78532564	2	300	100

Empleado	Proxecto	horas_previsas	horas_dedicadas	PID	Nome	Presuposto
23456238	1	1800	95	1	Tráfico	198000
25368964	1	800	30	1	Tráfico	198000
23456238	2	600	10	2	Xardinería	97000
78878965	2	800	5	2	Xardinería	97000
78532564	2	300	100	2	Xardinería	97000

Nome	count
Tráfico	2
Xardinería	3

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Para cada empleado, obtener la suma de horas que ha dedicado ya a sus proyectos



Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

```
SELECT e.dni, e.nombre, SUM(pa.horas_dedicadas)
FROM empleado e, participa p
WHERE e.dni = p.empleado
GROUP BY e.dni
```

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Para cada empleado que participe en dos o más proyectos, obtener el número de horas que tiene previsto dedicar en media a cada proyecto

Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

⑤ **SELECT** e.dni, e.nombre,
AVG(horas_previsas)

① **FROM** empleado e, participa p

② **WHERE** e.dni = p.empleado

③ **GROUP BY** e.dni

④ **HAVING** count(*) > 2

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

dni	Nome	Salario	coste_hora	Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	Alfredo	35000	45	23456238	1	1800	95
23456238	Alfredo	29500	30	23456238	2	600	10
25368964	Sofía	43000	60	25368964	1	800	30
78878965	Elena	40500	55	78878965	2	800	5
78532564	Ernesto	41000	56	78532564	2	300	100

dni	Nome	horas_previsas
23456238	Alfredo	1200

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

- Para cada proyecto que tenga más de dos empleados, obtener su identificador, nombre, presupuesto total, presupuesto previsto para personal y presupuesto previsto para otros gastos



Proxecto

PID	Nome	Presuposto
1	Tráfico	198000
2	Xardinería	97000
3	Festas	86000

Empleado

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

Participa

Empleado	Proxecto	horas_previsas	horas_dedicadas
23456238	1	1800	95
25368964	1	800	30
23456238	2	600	10
78878965	2	800	5
78532564	2	300	100

```
SELECT pr.pid, pr.nome, pr.presupuesto,
       SUM(pa.horas_previsas*e.coste_hora) AS personal,
       pr.presupuesto - SUM(pa.horas_previsas*e.coste_hora) AS otros_gastos
FROM proxecto pr, participa pa, empleado e
WHERE pr.pid=pa.proxecto and e.dni=pa.empleado
GROUP BY pr.pid
HAVING count(*) > 2
```


Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Subconsultas en la cláusula FROM

```
SELECT e1, e2, ..., en
FROM R1 as r1, (SELECT ...) as r2, ..., Rm as rm
WHERE p
```

Devuelve
una tabla

Ejercicio

- ▶ Permiten implementar estrategias de tipo **divide y vencerás**
- ▶ Para cada proyecto, obtén el número de empleados con un coste por hora mayor de 40 y el número de empleados con un coste por hora menor o igual que 40

Empleado
+DNI {PK}
+Nome
+Salario
+Coste_hora

Participa
+horas_previstas
+horas_dedicadas

Proyecto
+PID {PK}
+Nome
+Presupuesto

```
SELECT p.pid, p.nome, Tmas40.mas40, Tmenos40.menos40
FROM proyectos p,
  (SELECT p.proyecto, count(*) as mas40
   FROM participa p, empleado e
   WHERE p.empleado=e.dni and coste_hora > 40
   GROUP BY p.proyecto) as Tmas40,
  (SELECT p.proyecto, count(*) as menos40
   FROM participa p, empleado e
   WHERE p.empleado=e.dni and coste_hora <= 40
   GROUP BY p.proyecto) as Tmenos40
WHERE p.pid = Tmas40.proyecto and p.pid = Tmenos40.proyecto
```

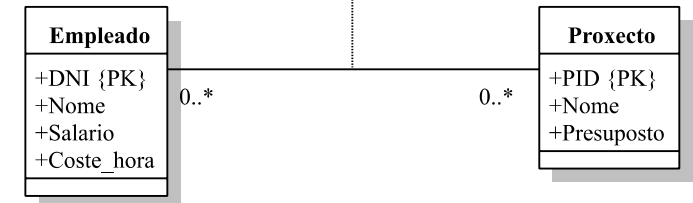
Devuelve
un único
valor

■ Subconsultas en otras subclausulas

```
SELECT e1, (SELECT ...) as e2, ..., en
FROM R1 as r1, R2 as r2, ..., Rm as rm
WHERE ... (SELECT ...) ...
GROUP BY ...
HAVING ... (SELECT ...) ...
```

Ejercicio

- ▷ Obtén todos los datos del empleado con el máximo salario



```
SELECT *
FROM empleado
WHERE salario = (SELECT MAX(salario) FROM empleado)
```

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Predicados de conjuntos y cuantificadores

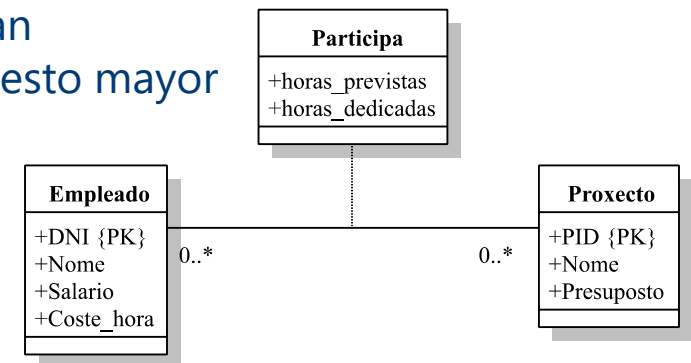
- ▷ Se usan en la clausulas **WHERE** y **HAVING**
 - **IN**
 - Modificadores de comparaciones (**ANY**, **SOME**, **ALL**)
 - **EXISTS**
- ▷ Obtener los datos de los empleados que han participado en algún proyecto con presupuesto mayor de 90000

Ejercicio

```
SELECT DISTINCT e.*
FROM empleado e, participa pa, proxecto pr
WHERE e.dni = pa.empleado
and pr.pid = pa.proyecto
and pr.presupuesto > 90000
```

```
SELECT *
FROM empleado
WHERE dni IN
(SELECT empleado
FROM participa pa, proxecto pr
WHERE pa.proyecto=pr.pid
and pr.presupuesto > 90000)
```

```
SELECT *
FROM empleado
WHERE dni = ANY
(SELECT empleado
FROM participa pa, proxecto pr
WHERE pa.proyecto=pr.pid
and pr.presupuesto > 90000)
```



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Predicados de conjuntos y cuantificadores

- ▷ Se usan en la cláusulas **WHERE** y **HAVING**
 - **IN**
 - Modificadores de comparaciones (**ANY**, **SOME**, **ALL**)
 - **EXISTS**
- ▷ Obtener los datos del empleado con el mayor salario

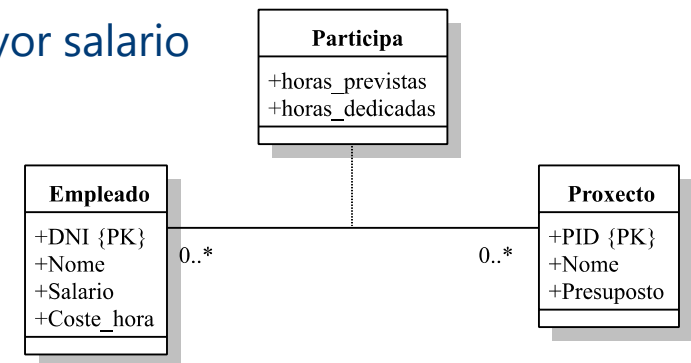
```
SELECT *
FROM empleado
WHERE salario = (SELECT MAX(salario)
                 FROM empleado)
```

```
SELECT *
FROM empleado
WHERE salario >= ALL (SELECT salario
                     FROM empleado)
```

```
SELECT *
FROM empleado e1
WHERE NOT EXISTS (SELECT * FROM empleado e2
                  WHERE e1.salario < e2.salario)
```

$$\{e1 \mid \text{empleado}(e1) \wedge \neg \exists e2(\text{empleado}(e2) \wedge e1[\text{salario}] < e2[\text{salario}])\}$$

Ejercicio



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Union, intersección y diferencia de conjuntos

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

```
...  
UNION  
SELECT ...  
FROM ...
```

```
...  
ORDER BY ...
```

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

```
...  
UNION ALL  
SELECT ...  
FROM ...
```

```
...  
ORDER BY ...
```

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

```
...  
EXCEPT  
SELECT ...  
FROM ...
```

```
...  
ORDER BY ...
```

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

```
...  
INTERSECT  
SELECT ...  
FROM ...
```

```
...  
ORDER BY ...
```

**Recordar que las
tablas deben ser
compatibles**

**Por defecto se
eliminan
duplicados**

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ INNER Theta Join (cualquier predicado)

- Mostrar los datos de los empleados junto con los datos de su departamento

```
SELECT *  
FROM empleado e, departamento d  
WHERE e.depld = d.depld
```

Empleado

DNI	Nome	Salario	Coste_hora	depld
23456238	Alfredo	35000	45	1
25368964	Sofía	43000	60	1
58325647	Ricardo	29500	30	2
78878965	Elena	40500	55	2
78532564	Ernesto	41000	56	

Departamento

depld	nom_dept
1	Marketing
2	Ventas
3	Compras

DNI	Nome	Salario	Coste_hora	depld	depld	nom_dept
23456238	Alfredo	35000	45	1	1	Marketing
25368964	Sofía	43000	60	1	1	Marketing
58325647	Ricardo	29500	30	2	2	Ventas
78878965	Elena	40500	55	2	2	Ventas

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ INNER Theta Join (cualquier predicado)

- Mostrar los datos de los empleados junto con los datos de su departamento

Tipo de Join
por defecto

Se puede
eliminar

```
SELECT *
FROM empleado e INNER JOIN departamento d
ON (e.depld = d.depld)
```

DNI	Nome	Salario	Coste_hora	depld	depld	nom_dept
23456238	Alfredo	35000	45	1	1	Marketing
25368964	Sofía	43000	60	1	1	Marketing
58325647	Ricardo	29500	30	2	2	Ventas
78878965	Elena	40500	55	2	2	Ventas

Empleado

DNI	Nome	Salario	Coste_hora	depld
23456238	Alfredo	35000	45	1
25368964	Sofía	43000	60	1
58325647	Ricardo	29500	30	2
78878965	Elena	40500	55	2
78532564	Ernesto	41000	56	

Departamento

depld	nom_dept
1	Marketing
2	Ventas
3	Compras

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ INNER Equi Join (predicado =)

- Mostrar los datos de los empleados junto con los datos de su departamento

Mismo nombre
de columna

Se elimina una
copia del
resultado

```
SELECT *  
FROM empleado e JOIN departamento d  
USING (depld)
```

DNI	Nome	Salario	Coste_hora	depld	nom_dept
23456238	Alfredo	35000	45	1	Marketing
25368964	Sofía	43000	60	1	Marketing
58325647	Ricardo	29500	30	2	Ventas
78878965	Elena	40500	55	2	Ventas

Empleado

DNI	Nome	Salario	Coste_hora	depld
23456238	Alfredo	35000	45	1
25368964	Sofía	43000	60	1
58325647	Ricardo	29500	30	2
78878965	Elena	40500	55	2
78532564	Ernesto	41000	56	

Departamento

depld	nom_dept
1	Marketing
2	Ventas
3	Compras

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ INNER NATURAL JOIN

- Mostrar los datos de los empleados junto con los datos de su departamento

Equi Join por
todos los
atributos con el
mismo nombre

```
SELECT *
FROM empleado e NATURAL JOIN departamento d
```

DNI	Nome	Salario	Coste_hora	depld	nom_dept
23456238	Alfredo	35000	45	1	Marketing
25368964	Sofía	43000	60	1	Marketing
58325647	Ricardo	29500	30	2	Ventas
78878965	Elena	40500	55	2	Ventas

Empleado

DNI	Nome	Salario	Coste_hora	depld
23456238	Alfredo	35000	45	1
25368964	Sofía	43000	60	1
58325647	Ricardo	29500	30	2
78878965	Elena	40500	55	2
78532564	Ernesto	41000	56	

Departamento

depld	nom_dept
1	Marketing
2	Ventas
3	Compras

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ OUTER JOINS

▷ Se pierden los datos de Ernesto y del departamento de Compras

```
SELECT *
FROM empleado e
      NATURAL LEFT OUTER JOIN
departamento d
```

```
SELECT *
FROM empleado e NATURAL RIGHT JOIN departamento d
```

```
SELECT *
FROM empleado e NATURAL FULL JOIN departamento d
```

Empleado

DNI	Nome	Salario	Coste_hora	depld
23456238	Alfredo	35000	45	1
25368964	Sofía	43000	60	1
58325647	Ricardo	29500	30	2
78878965	Elena	40500	55	2
78532564	Ernesto	41000	56	

Departamento

depld	nom_dept
1	Marketing
2	Ventas
3	Compras

DNI	Nome	Salario	Coste_hora	depld	nom_dept
23456238	Alfredo	35000	45	1	Marketing
25368964	Sofía	43000	60	1	Marketing
58325647	Ricardo	29500	30	2	Ventas
78878965	Elena	40500	55	2	Ventas
78532564	Ernesto	41000	56		
				3	Compras

↑ FULL ↓

INNER
RIGHT LEFT

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Lenguaje de Manipulación de datos: Inserción de datos

```
INSERT INTO nombre_tabla (A1, A2, ..., An)
VALUES (v11, v12, ..., v1n),
       (v21, v22, ..., v2n),
       ...
       (vm1, vm2, ..., vmn);
```

```
INSERT INTO nombre_tabla (A1, A2, ..., An)
SELECT e1, e2, ..., en
FROM ...
```

Si no se especifican
se consideran todos
los atributos de la
tabla

Los no especificados
se rellenan con valor
por defecto o con
nulos

Si no tienen valor
por defecto y no
admiten nulos se
genera una
excepción

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones 

Etc.

■ Borrado de datos

```
DELETE FROM nombre_tabla  
WHERE p;
```

■ Modificación de datos

```
UPDATE nombre_tabla  
SET  $a_1 = e_1, a_2 = e_2, \dots, a_n = e_n$   
WHERE p;
```

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

TRANSACCIÓN

Unidad de ejecución de un programa que accede y posiblemente modifica varios elementos de datos

- Unidad simple e indivisible para el usuario (**Atomicidad**)
 - ▷ Complejo: Datos en memoria y en disco
- Operan sin interferencia de las operaciones de otras transacciones (**Aislamiento**)
- Sus efectos perduran a pesar de caídas del sistema (**Durabilidad**)
- Su ejecución aislada y atómica deja la base de datos en un estado consistente (**Consistencia**)
 - ▷ Dependiente de la aplicación

Begin Transaction

Operación
Operación
Operación
...
Operación

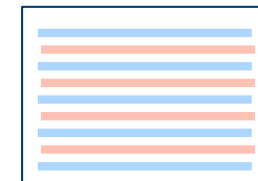
End Transaction

Deshacer

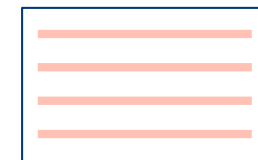
T1



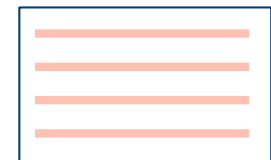
T2



T3



T3



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Propiedades **ACID**

▷ **Atomicidad** (**Atomicity**)

- Se ejecutan todas las operaciones o no se ejecuta ninguna

▷ **Consistencia** (**Consistency**)

- La ejecución aislada de una transacción preserva la consistencia de la base de datos

▷ **Aislamiento** (**Isolation**)

- Aunque varias transacciones se ejecuten de forma concurrente, el sistema garantiza que cada par de transacciones T_i y T_j , para T_i da la impresión de que o bien T_j terminó su ejecución antes de iniciarla T_i o T_j inició su ejecución después de terminarla T_i .

▷ **Durabilidad** (**Durability**)

- Después de que una transacción termine con éxito, los cambios que ha hecho en la base de datos persisten, incluso si hay fallos en el sistema.

Asegurar el aislamiento
puede tener un impacto
grande en el
rendimiento del
sistema

Necesario?

Depende la
aplicación

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones 

Etc.

■ Consistencia

- ▷ Suma de A y B debe permanecer inalterada en la transacción
 - _ T1 no debe crear ni destruir dinero
- ▷ **Responsabilidad del programador**
- ▷ Comprobación automática de restricciones de integridad puede ayudar a cumplir esta propiedad
 - _ PRIMARY KEY, FOREIGN KEY, CHECK, etc.

■ Atomicidad

- ▷ Estado inconsistente no provocado por la mala praxis del programador
- ▷ Estado inconsistente por el que pasa la transacción es temporal
 - _ No se dará si se ejecuta de forma atómica
- ▷ Se mantiene un registro (log) de valores antiguos
 - _ Restaurar valores antiguos en caso de fallo
- ▷ **Responsabilidad de SGBDs**
 - _ Sistema de recuperación

T1

```
leer(A);
A:= A - 50;
escribir(A);
leer(B);
B:=B+50;
escribir(B);
```

```
leer(A);
A:= A - 50;
escribir(A);
```

Fallo

Registro
Histórico

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones 

Etc.

■ Durabilidad

- ▷ Una vez ha terminada la transacción, los cambios perduran en la base de datos, incluso si hay un fallo en el sistema después de completarse la transacción.
- ▷ Para garantizar la durabilidad
 - _ La información sobre las modificaciones de la transacción se guarda en el disco
 - _ Dicha información permite al SGBD reconstruir las modificaciones cuando el sistema se reinicia después de un fallo.
- ▷ **Responsabilidad del SGBDs**
 - _ **Sistema de recuperación**
 - También responsable de la atomicidad.

T1

```
leer(A);
A:= A - 50;
escribir(A);
leer(B);
B:= B + 50;
escribir(B);
```


Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Aislamiento

- ▷ Incluso si la ejecución aislada de una transacción es Consistente, la ejecución concurrente con otra puede llevar a la base de datos a un estado inconsistente
 - El estado inconsistente puede mantenerse después del final de ambas transacciones
- ▷ Solución:
 - Ejecución secuencial de las transacciones
 - Solución poco eficiente en cuanto a rendimiento del sistema
- ▷ Esta propiedad garantiza que la ejecución concurrente de varias transacciones produce un resultado equivalente a la ejecución de las transacciones una detrás de la otra (secuencial) en algún orden
- ▷ Responsabilidad del SGBDs
 - Sistema de control de concurrencia

T1

```
leer(A);
A:= A - 50;
escribir(A);

leer(B);
B:= B + 50;
escribir(B);
```

T2

```
leer(A);
Leer (B);
S:=A+B
escribir(S);
```

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

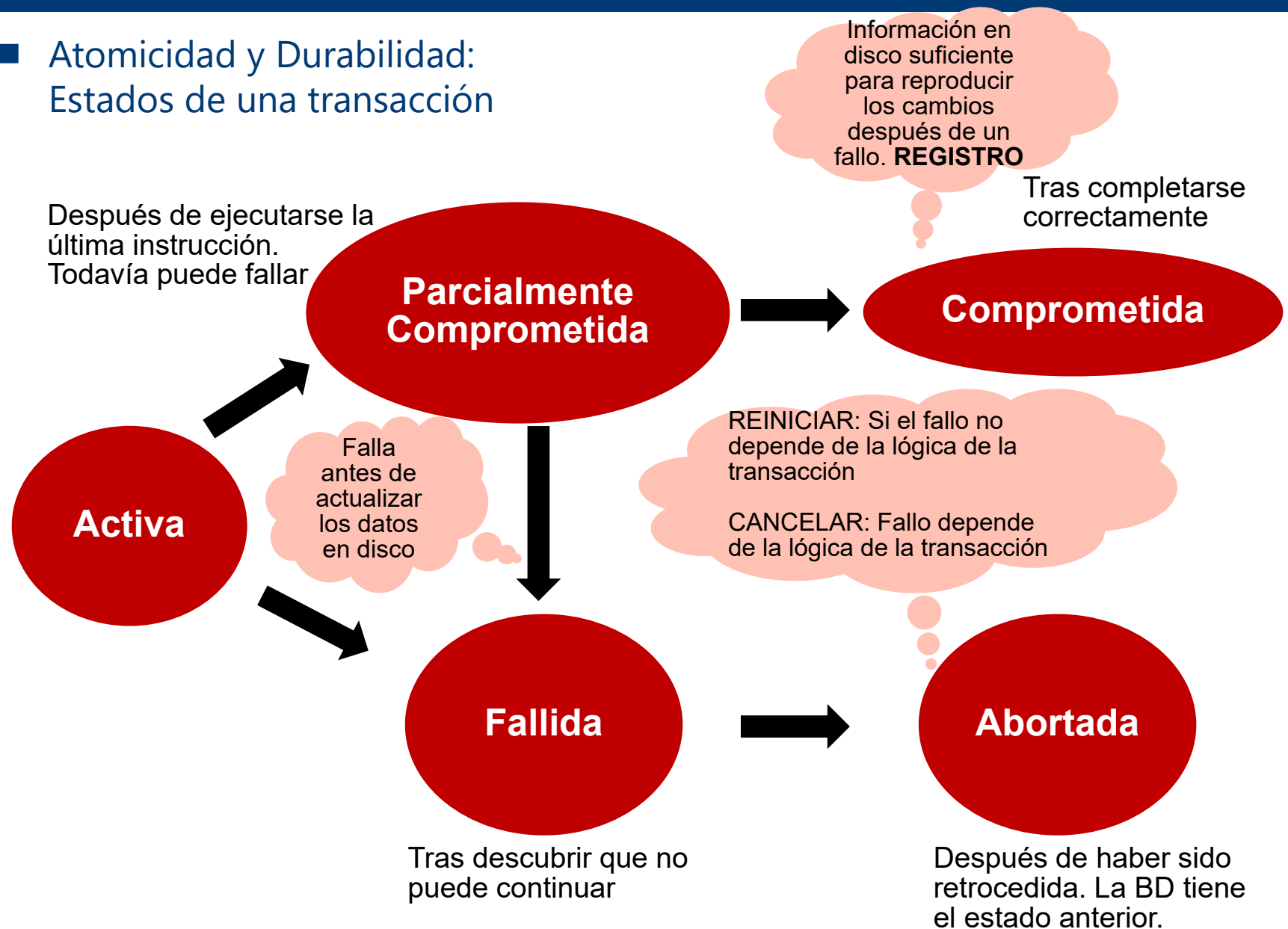
Oper. conjunto

Joins

Transacciones

Etc.

■ Atomicidad y Durabilidad: Estados de una transacción



Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

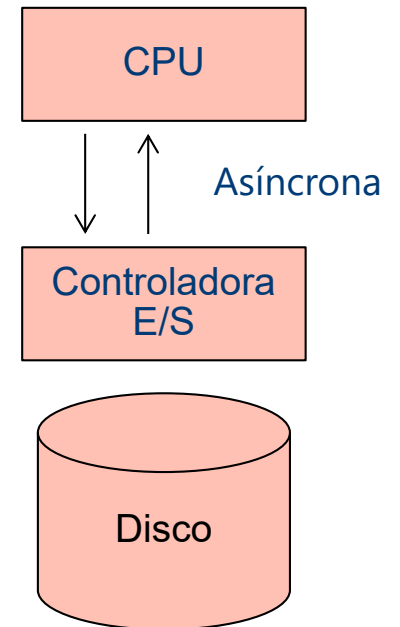
Joins

Transacciones

Etc.

■ Aislamiento

- ▷ Ejecución concurrente de varias transacciones puede generar inconsistencias
- ▷ Razones para no limitarse a la ejecución secuencial
 - _ Mejora del rendimiento y uso de los recursos
 - Ejecución paralela de instrucciones E/S y CPU
 - Disminuye **tiempo total de ejecución**, aumenta uso CPU y E/S
 - _ Reducción **tiempo medio de respuesta**
 - Transacciones cortas no tienen que esperar a que termine una larga
- ▷ Necesarios planificadores que garanticen aislamiento



■ Aislamiento y Atomicidad

- ▷ ¿Qué ocurre si falla una transacción durante la ejecución concurrente de varias?
- ▷ Si T falla, es necesario deshacer sus efectos (**atomicidad**)
 - _ Cada T_i , que ha leído datos escritos por T debe deshacerse también!
 - **Abortar transacciones que no han fallado**
 - **Si se han comprometido ya, nos e pueden abortar!**

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Niveles de aislamiento en el estándar SQL

▷ **Secuenciable**

- Normalmente asegura la **ejecución secuenciable** de la transacción
 - Algunos SGBDs utilizan protocolos que **no lo garantizan en todos los casos**

▷ **Lectura repetible**

- Sólo se pueden leer datos **comprometidos** (escritos por transacciones comprometidas)
- Entre dos lecturas del mismo dato en una transacción, otra **no puede modificarlo**
- No garantiza la secuencialidad

▷ **Lectura comprometida**

- Sólo permite leer datos comprometidos
- **NO** garantiza lecturas repetibles

▷ **Lectura no comprometida**

- Permite leer datos no comprometidos
- Nunca **escrituras sucias**
 - Escribir sobre un dato escrito por otra transacción no comprometida

T I	T J
SELECT count(*) FROM empleado WHERE dept = 'Ventas'	INSERT INTO empleado (1, 'Juán', 23k, 'Ventas')

Las instrucciones de TI y TJ no deberían de tener conflicto ya que no actúan sobre los mismos elementos de datos

Sin embargo el orden de ejecución importa

HAI CONFLICTO

Fenómeno Fantasma

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.

■ Protocolos de implementación del aislamiento

▷ Estrategias Pesimistas

- _ Toman medidas que fuerzan esperas o retrocesos de transacciones como prevención
- _ Basados en bloqueos
 - Protocolo de bloqueo en dos fases
- _ Basados en marcas de tiempo

▷ Estrategias optimistas

- _ Permiten la ejecución normal y solo al final comprueban si hay problemas
- _ Basados en validación
- _ Basados en versiones múltiples
 - **Aislamiento de instantáneas**

No garantiza la
secuencialidad

Pueden ser necesarias
medidas adicionales

Cláusula **FOR UPDATE**
(trata las lecturas como
escrituras)

Modelo

Lenguajes
Formales

Sintaxis básica

Nulos

Agregación

Subconsultas

Oper. conjunto

Joins

Transacciones

Etc.



■ Vistas

```
CREATE VIEW nombre_vista AS  
SELECT ...  
FROM ...;
```

■ Creación de índices

■ Tipos, funciones, procedimientos definidos por el usuario

■ Disparadores

■ Seguridad

■ Acceso a la Base de datos desde un lenguaje de programación

■ Consultas Recursivas

■ Agregación avanzada y OLAP (... Inteligencia de Negocio)

Bases de datos relacionales y SQL

José R.R. Viqueira

Centro Singular de Investigación en Tecnoloxías Intelixentes (CITIUS)
Rúa de Jenaro de la Fuente Domínguez,
15782 - Santiago de Compostela.

Despacho: 209

Telf: 881816463

Mail: jrr.viqueira@usc.es

Skype: jrviqueira

URL: <https://citius.gal/team/jose-ramon-rios-viqueira>

Curso 2023/2024