BASES DE DATOS (IG18 Semipresencial)
El Modelo Relacional
Algebra Relacional y SQL

Lledó Museros / Ismael Sanz
museros@icc.uji.es / isanz@icc.uji.es

- **Álgebra relacional** y **cálculo relacional** : la base de los lenguajes relacionales (Codd).
  - Álgebra relacional → lenguaje procedural
  - Cálculo relacional → lenguaje no procedural

  } ¡¡pero ambos lenguajes son equivalentes!!
- **AVISO:** son lenguajes formales no muy "amigables", pero se deben estudiar porque sirven para comprender las operaciones básicas de los lenguajes de manejo de datos.
- Un **lenguaje es relacionalmente completo** si permite obtener cualquier relación que se pueda derivar mediante el álgebra relacional (o el cálculo relacional).
- **SQL** es relacionalmente completo, de hecho, se diseño basándose en el álgebra y cálculo relacional.

- El Álgebra relacional (en adelante Álgebra) está constituida por una colección de operadores de alto nivel que, aplicados a las relaciones, dan como resultado nuevas relaciones.

- Si R1, R2, . . . , Rj y R' son relaciones y Op un operador del Álgebra, una operación consiste en aplicar Op a la relación o relaciones R1, R2, . . . , Rj que da como resultado la relación R':

$$Op(R1, R2, ..., Rj) = R'$$

- El Álgebra cumple la propiedad de cierre (clausura), ya que el resultado de una operación es otra relación. Si Op1, Op2, . . . , Opn son operadores del Álgebra, se cumple:

$$Opn(...(Op2(Op1(R)))) = R'$$

- **Asignación**: Operación auxiliar que se utiliza para almacenar resultados en una nueva relación o para resultados intermedios cuando se desea dividir una operación compleja en una secuencia de operaciones más simples.

$$NR := Op(R)$$

- Permite copiar una relación en otra nueva (siendo el operador Op la identidad):

$$NR := R$$

- O renombrar los atributos de la relación original:

$$NR(A1,A2, ..., An) := R$$

- **Renombramiento**: Operación auxiliar que se utiliza para renombrar los atributos y así evitar los problemas causados al operar con varias relaciones en las que coincide el nombre de algún atributo.

En este caso, si no renombramos, la relación resultante cuya cabecera es un subconjunto de los atributos de las relaciones implicadas en la operación, no estaría bien definida porque los nombres de atributos deben ser únicos.

R RENAME ai AS bj

R RENAME a1 AS bj, a2 AS b2, ...

- **Restricción (selección)**: *R WHERE predicado*
- Obtiene las tuplas de *R* que cumplen el *predicado* especificado . Este *predicado* es una comparación en la que aparece al menos un atributo de *R*; puede ser una combinación booleana de varios predicados.

Piezas rojas con peso mayor de 15:

P WHERE COLOR = 'rojo' AND PESO >15

Pnum	PNOMBRE	COLOR	PESO	CIUDAD
P2	perno	rojo	17	Londrés
P6	engrane	rojo	19	París

- **SQL:**

Restricción

```
SELECT *
FROM tabla
WHERE predicado
```

- Piezas rojas con peso mayor de 15:

```
SELECT *
FROM p
WHERE color = 'rojo'
AND peso > 15;
```

- **Proyección**: *R [ai, ..., ak]*

- Obtiene una relación que contiene un subconjunto vertical de *R*, extrayendo los valores de los atributos especificados y **eliminando tuplas duplicadas**.

Proveedores y piezas que envían:

SP[snum,pnum]

snum	pnum
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
....	

Ciudades en las que se almacena alguna pieza:

P[ciudad]

CIUDAD
París
Roma
Londres

- **SQL:**

Proyección

```
SELECT [DISTINCT] ai, ..., ak
FROM tabla;
```

Proveedores y piezas que envían:

```
SELECT
sp.snum,sp.pnum
FROM sp;
```

Ciudades en las que se almacena alguna pieza:

```
SELECT p.ciudad
FROM p;
```

- **¿Son equivalentes estas sentencias a las del Álgebra?**

- **Unión**: *R UNION S*
- Obtiene una relación cuyas tuplas son las que se encuentran en *R*, o en *S*, o en ambas relaciones a la vez. Para poder realizar esta operación, *R* y *S* deben ser compatibles para la unión.
- Dos relaciones son **compatibles para la unión** si ambas tienen el **mismo esquema de tupla (cabecera)**, es decir, si tienen el mismo número de atributos, se llaman igual y se encuentran definidos sobre los mismos dominios.
- Ciudades en las que hay proveedores o se almacenan piezas:

S[CIUDAD]

CIUDAD
Londres
París
Atenas

P[CIUDAD]

CIUDAD
París
Roma
Londres

S[CIUDAD] UNION P[CIUDAD]

CIUDAD
París
Londres
Roma
Atenas

- **SQL:** En SQL todas las operaciones de conjuntos eliminan duplicados.

Unión

```
SELECT ai, ..., ak
FROM tabla1
UNION
SELECT ai, ....., ak
FROM tabla2
```

Ciudades en las que hay proveedores o se almacenan piezas:

```
SELECT s.ciudad
FROM s
UNION
SELECT p.ciudad
FROM p;
```

- **Diferencia**: *R MINUS S*
- Obtiene una relación que tiene las tuplas que se encuentran en R y no se encuentran en S. Para poder realizar esta operación, R y S deben ser compatibles para la unión.
- Ciudades en las que hay proveedores y no se almacenan piezas:

S[CIUDAD]

CIUDAD
Londres
París
Atenas

P[CIUDAD]

CIUDAD
París
Roma
Londres

S[CIUDAD] MINUS P[CIUDAD]

→

CIUDAD
Atenas

- Ciudades en las que se almacenan piezas y no hay proveedores:

S[CIUDAD]

CIUDAD
Londres
París
Atenas

P[CIUDAD]

CIUDAD
París
Roma
Londres

P[CIUDAD] MINUS S[CIUDAD] =

CIUDAD
Roma

- **SQL:**

```
SELECT ai, ..., ak
FROM tabla1
MINUS
SELECT ai, ..., ak
FROM tabla2
```

- Ciudades en las que hay proveedores y no se almacenan piezas:

```
SELECT s.ciudad
FROM s
MINUS
SELECT p.ciudad
FROM p;
```

- **Producto cartesiano**: *R TIMES S*

- El producto cartesiano de dos relaciones *R* y *S*, de cardinalidades $m1$ y $m2$ respectivamente, es una relación definida sobre la unión de los atributos de ambas relaciones y cuyo cuerpo está constituido por las $m1$ x $m2$ tuplas formadas concatenando cada tupla de *R* con cada una de las tuplas de *S*. Es decir, obtiene una relación cuyas tuplas están formadas por la concatenación de todas las tuplas de R con todas las tuplas de S.

- Datos de los proveedores que envían la pieza P1 y cantidad que envían:

SP where pnum='P1'

snum	pnum	CANT
S1	P1	300
S2	P1	300

- Datos de los proveedores que envían la pieza P1 y cantidad que envían: **S TIMES (SP where pnum='P1')**

S.snum	SNOMBRE	ESTADO	CIUDAD	SP.snum	pnum	CANT
S1	Salazar	20	Londres	S1	P1	300
S2	Jaimes	10	París	S1	P1	300
S3	Bernal	30	París	S1	P1	300
S4	Corona	20	Londres	S1	P1	300
S5	Aldana	20	Atenas	S1	P1	300
S1	Salazar	20	Londres	S2	P1	300
S2	Jaimes	10	París	S2	P1	300
S3	Bernal	30	París	S2	P1	300
S4	Corona	20	Londres	S2	P1	300
S5	Aldana	20	Atenas	S2	P1	300

¿Era esto lo que queríamos?

- Para quedarse sólo con los datos de los proveedores que envían la pieza P1, hay que hacer una restricción:

(S TIMES (SP where pnum='P1')) WHERE S.snum=SP.snum

S.snum	SNOMBRE	ESTADO	CIUDAD	SP.snum	Pnum	CANT
S1	Salazar	20	Londres	S1	P1	300
S2	Jaimes	10	París	S2	P1	300

Ahora nos quedamos sólo con los atributos buscados con una proyección:

((S TIMES (SP where pnum='P1')) WHERE S.snum=SP.snum) [S.snum, SNOMBRE, ESTADO, CIUDAD, CANT]

S.snum	SNOMBRE	ESTADO	CIUDAD	CANT
S1	Salazar	20	Londres	300
S2	Jaimes	10	París	300

- **SQL:**

Producto Cartesiano

```
SELECT t.ai, ..., t.ak,
                    u.bj, ...,u.b1
FROM tabla1 t, tabla2 u
```

- Datos de los proveedores que envían la pieza P1 y cantidad que envían:

```
SELECT s.snum, s.snombre, s.estado,
                    s.ciudad, sp.cant
FROM s, sp
WHERE s.snum=sp.snum
AND sp.pnum='P1';
```

- **Combinación natural (concatenación o join)**: *R JOIN S*
- Obtiene una relación cuyas tuplas son todas las tuplas de *R* concatenadas con todas las tuplas de *S* que en los atributos comunes (**los que se llaman igual**) tienen los mismos valores. Estos atributos comunes aparecen una sola vez en el resultado.
- Datos de los proveedores que envían la pieza P1 y cantidad que envían: **SP where pnum='P1'**

snum	pnum	CANT
S1	P1	300
S2	P1	300

S JOIN (SP where pnum='P1')

S.snum	SNOMBRE	ESTADO	CIUDAD	pnum	CANT
S1	Salazar	20	Londres	P1	300
S2	Jaimes	10	París	P1	300

- Mediante una proyección nos quedamos sólo con los atributos que nos interesan:

(S JOIN (SP where pnum='P1')) [snum, SNOMBRE, STADO, CIUDAD, CANT]

S.snum	SNOMBRE	ESTADO	CIUDAD	CANT
S1	Salazar	20	Londres	300
S2	Jaimes	10	París	300

- Nótese que esta expresión obtiene el mismo resultado que la expresión:

((S TIMES (SP where pnum='P1')) WHERE S.snum=SP.snum) [S.snum, SNOMBRE, ESTADO, CIUDAD, CANT]

- Porque la **Combinación Natural** es **un producto cartesiano**

  R1= (S TIMES (SP where pnum='P1'))

una restricción de igualdad sobre los atributos comunes.

R2= R1 WHERE S.snum=SP.snum

y **una proyección** eliminando (uno de) los atributos repetidos: **RDO= R2 [S.snum, SNOMBRE, ESTADO, CIUDAD, CANT]**

- **SQL:**

Concatenación Natural

```
SELECT *
FROM tabla1 NATURAL JOIN tabla2
```

- Datos de los proveedores que envían la pieza P1 y cantidad que envían:

```
SELECT s.snum, s.snombre, s.estado,
                          s.ciudad, sp.cant
FROM sp NATURAL JOIN s
WHERE sp.pnum='P1';
```

- **SQL:**

Concatenación

```
SELECT *
FROM tabla1 JOIN tabla2
USING (ai,ak,...)
```

- Datos de los proveedores que envían la pieza P1 y cantidad que envían:

```
SELECT s.snum, s.snombre, s.estado, s.ciudad, sp.cant
FROM sp JOIN s USING (snum)
WHERE sp.pnum='P1';
```

- **SQL:**

Concatenación

```
SELECT *
FROM tabla1 JOIN tabla2
ON tabla1.ai = tabla2.bj,
        tabla1.ak = tabla2.bl, ...
```

- Datos de los proveedores que envían la pieza P1 y cantidad que envían:

```
SELECT s.snum, s.snombre, s.estado, s.ciudad, sp.cant
FROM sp JOIN s ON s.snum=sp.snum
WHERE sp.pnum='P1';
```

- **Atención:** El **JOIN** del álgebra se corresponde con el **NATURAL JOIN** de SQL. El **JOIN** de SQL no existe en el álgebra como un único operador, pero se puede simular (TIMES+ proyección)

- **Combinación natural externa izquierda**:

R LEFT OUTER JOIN S

- La Combinación natural externa izquierda es una Combinación natural en la que las tuplas de R que no tienen valores en común con ninguna tupla de S, también aparecen en el resultado.
- Se puede definir mediante un JOIN y sobre el resultado aplicar la operación UNION para agregar las tuplas de la relación de la izquierda que no estén ya añadidas

- **Combinación natural externa izquierda**:

R LEFT OUTER JOIN S

- **SQL:**

Concatenación natural externa izquierda

```
SELECT *
FROM tabla1 NATURAL LEFT OUTER JOIN tabla2
```

Concatenación externa izquierda

```
SELECT *
FROM tabla1 LEFT OUTER JOIN tabla2
USING (ai) / ON tabla1.ai = tabla2.bj
```

- **Combinación natural externa derecha**:

R RIGHT OUTER JOIN S

- La combinación natural externa derecha es una Combinación natural en la que las tuplas de S que no tienen valores en común con ninguna tupla de R, también aparecen en el resultado.
- Se puede definir mediante un JOIN y sobre el resultado aplicar la operación UNION para agregar las tuplas de la relación de la derecha que no estén ya añadidas

- **Combinación natural externa derecha**:

R RIGHT OUTER JOIN S

- **SQL:**

Concatenación natural externa derecha

```
SELECT *
FROM tabla1 NATURAL RIGHT OUTER JOIN tabla2
```

Concatenación externa izquierda

```
SELECT *
FROM tabla1 RIGHT OUTER JOIN tabla2
USING (ai) / ON tabla1.ai = tabla2.bj
```

- **Combinación natural externa completa:**

R FULL OUTER JOIN S

- La Combinación natural externa completa es una Combinación natural en la que las tuplas de S que no tienen valores en común con ninguna tupla de R y las tuplas de R que no tienen valores en común con ninguna tupla de S , también aparecen en el resultado.

- Se puede definir mediante un JOIN y sobre el resultado aplicar la operación UNION para agregar las tuplas de la relación de la izquierda y de la derecha que no estén ya añadidas.

- **Combinación natural externa completa:**

R FULL OUTER JOIN S

- **SQL:**

Concatenación natural externa completa

```
SELECT *
FROM tabla1 NATURAL FULL OUTER JOIN tabla2
```

Concatenación externa completa

```
SELECT *
FROM tabla1 FULL OUTER JOIN tabla2
USING (ai) / ON tabla1.ai = tabla2.bj
```

- Datos de todos los proveedores con las piezas que envían:

S LEFT OUTER JOIN SP

snum	SNOMBRE	ESTADO	CIUDAD	pnum	CANT
S1	Salazar	20	Londres	P1	300
S1	Salazar	20	Londres	P2	200
S1	Salazar	20	Londres	P3	400
S1	Salazar	20	Londres	P4	200
S1	Salazar	20	Londres	P5	100
S1	Salazar	20	Londres	P6	100
S2	Jaimes	10	París	P1	300
S2	Jaimes	10	París	P2	400
S3	Bernal	30	París	P2	200
S4	Corona	20	Londres	P2	200
S4	Corona	20	Londres	P4	300
S4	Corona	20	Londres	P5	400
S5	Aldana	30	Atenas	NULL	NULL

- Parejas de piezas azules y proveedores de la misma ciudad. Las piezas y proveedores que queden desparejados también se deben obtener:

PAZ := (P WHERE COLOR='azul')

PAZ FULL OUTER JOIN S

pnum	PNOMBRE	COLOR	PESO	CIUDAD	snum	SNOMBRE	ESTADO
P3	birlo	Azul	17	Roma	NULL	NULL	NULL
P5	leva	Azul	12	París	S2	Jaimes	10
P5	leva	Azul	12	París	S3	Bernal	30
NULL	NULL	NULL	NULL	Londres	S1	Salazar	20
NULL	NULL	NULL	NULL	Londres	S4	Corona	20
NULL	NULL	NULL	NULL	Atenas	S5	Aldana	30

- ***SQL:***

Datos de todos los proveedores con las piezas que envían:

```
SELECT *
FROM s NATURAL LEFT OUTER JOIN sp;
```

Parejas de piezas azules y proveedores de la misma ciudad. Las piezas y los proveedores que queden desparejados también se deben obtener:

```
SELECT *
FROM s NATURAL FULL OUTER JOIN p
WHERE p.color='azul';
```

- **Intersección**: *R INTERSECT S*
- Obtiene una relación que contiene las tuplas de *R* que también se encuentran en *S*. Para poder realizar esta operación, *R* y *S* deben ser compatibles para la unión.

R INTERSECT S

obtiene el mismo resultado que

R MINUS (R MINUS S)

- Ciudades en las que hay proveedores y se almacenan piezas:

S[CIUDAD]

CIUDAD
Londres
París
Atenas

P[CIUDAD]

CIUDAD
París
Roma
Londres

P[CIUDAD] INTERSECT S[CIUDAD] =

CIUDAD
Londres
París

- **SQL:**

Ciudades en las que hay proveedores y se almacenan piezas:

```
SELECT ciudad
FROM p
INTERSECT
SELECT ciudad
FROM s;
```

- **División**: *R DIVIDEBY S*
- Suponiendo que la cabecera de *R* es el conjunto de atributos *A* y la cabecera de *S* es el conjunto de atributos *B*, tales que *B* es un subconjunto de *A*. Si consideramos *C* como el subconjunto de los atributos de *A* que no están en *B* (*A* - *B*), la división obtiene una relación cuya cabecera es el conjunto de atributos *C* y que contiene las tuplas de *R* que están acompañadas de todas las tuplas de *S*.
- La división de una relación *R* (dividendo) por otra relación *S* (divisor) es una relación *RES* (cociente) tal que, al realizarse su combinación natural con el divisor, todas las tuplas resultantes se encuentran en el dividendo.
- No existe en **SQL** una orden equivalente. Se puede «simular» contando tuplas o mediante equivalencias del álgebra

- Proveedores que suministran todas las piezas que se almacenan en Londres:

Piezas que se almacenan en Londres:

PL:= (P WHERE

CIUDAD = ‘Londres’) [pnum]

pnum
P2
P4

Proveedores que almacenan **TODAS** las piezas de PL:

SP[snum, pnum] DIVIDEBY PL

snum
S1
S4

- **Agrupación (resumen)**:

SUMMARIZE R GROUPBY(ai,...,ak) ADD cálculo AS atributo

- Esta operación agrupa las tuplas de *R* que tienen los mismos valores en los atributos especificados y realiza un cálculo sobre los grupos obtenidos (*SUM*, *AVG*, *MAX*, *MIN*, *COUNT*). La relación resultado tiene como cabecera los atributos por los que se ha agrupado y el cálculo realizado con el nombre especificado en atributo.

- Número total de unidades de piezas suministradas por cada proveedor:

SUMMARIZE SP GROUPBY(snum) ADD SUM(CANT) AS CANT_TOTAL

snum	CANT_TOTAL
S1	1300
S2	700
S3	200
S4	900

- Número medio de unidades suministradas por envío:

SUMMARIZE SP GROUPBY () ADD AVG(CANT) AS MEDIA

MEDIA
258

- **SQL:**

Agrupación

```
SELECT cálculo(atributo)
FROM tabla
 GROUPBY ai, ..., ak
HAVING condición
```

- Número total de unidades de piezas suministradas por cada proveedor:

```
SELECT SUM(CANT) AS CANT_TOTAL
FROM sp
GROUPBY snum;
```

- Número medio de unidades suministradas por envío:

```
SELECT AVGCANT) AS MEDIA
FROM sp;
```

BASES DE DATOS (IG18 Semipresencial)
El Modelo Relacional. Algebra Relacional
¿DUDAS?

Lledó Museros / Ismael Sanz
museros@icc.uji.es / isanz@icc.uji.es