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

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

- **Álgebra relacional** y **cálculo relacional** son formas equivalentes de manejo de datos del modelo relacional. Pero:
- Mientras que en un lenguaje algebraico hay que especificar los operadores que se tienen que aplicar a las relaciones para obtener el resultado, en el **cálculo relacional** sólo es preciso **indicar cuál es el resultado que se quiere obtener**.
- En el cálculo se definen las características de la relación, y el SGBD define las operaciones a aplicar para construir la relación. Este tipo de **lenguajes** se llaman **predicativos** porque utilizan el **cálculo de predicados** para construir expresiones.
- Se puede decir que el cálculo es **descriptivo** y que el álgebra es **prescriptivo**.

- Una característica del cálculo relacional es que se utilizan **variables** que toman valores de tuplas de una relación, y sus únicos valores permitidos son las tuplas de dicha relación.
- Las **variables** también pueden tomar valores en el dominio de un atributo.
- Los lenguajes de cálculo relacional pueden ser de dos tipos:
  - orientados a tuplas, en los que una variable se interpreta como si representase las tuplas de una relación.
  - orientados a dominios, en los que una variable se interpreta como si representase los valores de un dominio.

- Una Variable de Tupla se define mediante una proposición de la forma siguiente,

  RANGE OF T IS X1, X2, . . . , Xn

  donde
- T es la variable de tupla definida.
- X1, X2, . . . , Xn son nombres de relación, o bien una expresión del cálculo de tuplas entre paréntesis.
- Las relaciones Xi deben ser compatibles respecto de la unión, es decir, todas tienen el mismo esquema de tupla.
- La variable T toma valores en la unión de todas las relaciones Xi.

```
RANGE OF SX IS S
RANGE OF SY IS S
RANGE OF PX IS P
RANGE OF SPX IS SP
RANGE OF CIUDADX IS (SX.CIUDAD),(PX.CIUDAD)
RANGE OF SP4X IS (SX WHERE ∃SPX
(SPX.S#=SX.S# AND SPX.P#='P4'))
```

- **Variables Libres y Ligadas**
- Cada Ocurrencia de una Variable de Tupla en una fórmula bien formada puede ser Libre o Ligada.
  - Será **Ligada** si la variable se asocia a un cuantificador (∃ o ∀): $\forall\, x(x > 3)$
  - Será **Libre** si no es Ligada: $(x > 3)$

- Una expresión del cálculo relacional tiene la siguiente estructura:

  lista_de_objetivos [WHERE fbf]

lista_de_objetivos: objetivos separados por comas
objetivo: [X =] T.A

- T : variable tupla
- A : atributo de la relación asociada a T
- X : nuevo nombre del atributo

fbf: fórmula bien formada (predicado simple o combinación booleana de predicados)

Evaluación

- Sea T, U, ..., V el conjunto de variables tupla especificadas en la lista de objetivos y X1, X2, ... , Xn los nombres de los atributos a obtener en el resultado:
  (1) Se forma el producto cartesiano T x U x . . . x V (T, U, . . . , V toman todos los valores posibles de sus rangos).
  (2) Se eliminan (restricción) las tuplas del producto anterior que no satisfacen la fbf del WHERE, si lo hay.
  (3) Se proyecta el resultado anterior sobre X1, X2, ..., Xn.

- **Ejemplo:**

SX.SNOMBRE,SPX.pnum WHERE
(SX.snum=SPX.snum AND SPX.CANT>200)

Dadas E y G, dos fórmulas bien formadas:

E	G	NOT E	F AND G	E OR G	IF E THEN G (= NOT E OR G)
V	V	F	V	V	V
V	F	F	F	V	F
F	V	V	F	V	V
F	F	V	F	F	V

$\exists x(E)$ se evalúa a verdadero si al sustituir en E la variable tupla x por **alguna** tupla de su rango, E se evalúa a verdadero.

px.pnum,px.pnombre WHERE ∃px (spx.pnum=px.pnum AND px.cant>100)

$\forall x(E)$ se evalúa a verdadero si al sustituir en E la variable tupla x por **cada** tupla de su rango, E es verdadero para cada una de ellas (todas).

px.pnum,px.pnombre WHERE ∀sx (px.ciudad<>sx.ciudad)

px.pnum,px.pnombre WHERE ∀spx (IF spx.pnum=px.pnum THEN spx.cant>100)

- Existe la expresión IF - THEN, que no es primitiva, y cuya funcionalidad es la siguiente:

$$\text{IF } f \text{ THEN } g \equiv (\text{NOT } f) \text{ OR } g$$

Existe (∃ o EXISTS):

```
SELECT lista de objetivos
FROM tabla1 t1 JOIN tabla2 t2, ...
WHERE EXISTS (SUBCONSULTA)
```

Para todo (∀ o FORALL): Se puede transformar en not exists ($\nexists$ o NOT EXISTS):

$$\forall A(B) \Leftrightarrow \nexists A(\neg B)$$

«Para todo A se cumple B» **es equivalente** a «No existe un A que no cumpla B».

Por ejemplo: “Todos los clientes son de Castellón” es equivalente a “No hay ningún cliente que no es de Castellón”

Así, en SQL podemos escribir el ∀ del siguiente modo:

∀ como NOT EXISTS:

```
SELECT lista de objetivos
FROM tabla1 t1 JOIN tabla2 t2, ...
WHERE NOT EXISTS (SUBCONSULTA CONTRARIA)
```

En **algunos casos** se puede expresar utilizando el operador ALL:

∀:

```
SELECT lista de objetivos
FROM tabla1 t1 JOIN tabla2 t2, ...
WHERE valor opLog ALL (SUBCONSULTA)
```

Piezas que se envian en cantidades superior a 100:

px.pnum,px.pnombre WHERE ∃spx
(spx.pnum=px.pnum AND px.cant>100)

```
SELECT px.pnum, px.pnombre
FROM p AS px
WHERE EXISTS (SELECT *
                    FROM sp AS spx
                    WHERE spx.pnum=px.pnum
                    AND spx.cant>100)
```

Piezas que en todos los envíos la cantidad es mayor a 100:

px.pnum,px.pnombre WHERE ∀spx (IF spx.pnum=px.pnum
THEN spx.cant>100)

Aplicando la equivalencia $\forall\ A(B) \Leftrightarrow \nexists\ A(\neg B)$:

px.pnum,px.pnombre WHERE ∃spx NOT (IF spx.pnum=px.pnum
THEN spx.cant>100)

Sustituyendo el IF por un OR:
px.pnum,px.pnombre WHERE ∃ spx NOT (spx.pnum<>px.pnum
OR spx.cant>100)

Aplicando $\neg(A\ OR\ B) \Leftrightarrow \neg\ A\ AND \neg\ B$

px.pnum,px.pnombre WHERE ∃ spx (spx.pnum=px.pnum
AND spx.cant<=100)

```
SELECT px.pnum, px.pnombre
FROM p AS px
WHERE NOT EXISTS (SELECT *
                  FROM sp AS spx
                   WHERE
      NOT(spx.pnum<>px.pnum
                  OR spx.cant>100))

SELECT px.pnum, px.pnombre
FROM p AS px
WHERE NOT EXISTS (SELECT *
                  FROM sp AS spx
                  WHERE spx.pnum=px.pnum
                  AND spx.cant<=100))
```

- El álgebra relacional y el cálculo de tuplas son dos formalismos equivalentes.
- Codd demostró esta equivalencia con su Algoritmo de Reducción, que genera una expresión del álgebra semánticamente equivalente a la expresión del cálculo de la que se parte.
- De modo similar a éste algoritmo se podría definir un algoritmo de conversión de una expresión del cálculo de tuplas a una expresión del álgebra.

Algoritmo de Codd. Pasos

1. Obtener el rango de cada variable tupla, aplicando las restricciones del WHERE que sea posible (rangos restringidos).
2. Construir el producto cartesiano de los rangos obtenidos en el paso (1).
3. Restringir el producto cartesiano del paso (2) usando las restricciones de join del WHERE.

4. Aplicar los cuantificadores de derecha a izquierda del siguiente modo:
   1. ∃RX : proyectar el resultado actual para eliminar todos los atributos de la relación asociada a RX.
   2. ∀RX : dividir el resultado actual entre la relación que contiene el rango restringido asociado a RX.
5. Proyectar el resultado del paso (4) de acuerdo con las especificaciones de la lista de objetivos.

Algoritmo de Codd. Ejemplo

S

S#	SNOMBRE	SITUACION	CIUDAD
S1	Salazar	20	Londres
S2	Jaimes	10	París
S3	Bernal	30	París
S4	Corona	20	Londres
S5	Aldana	30	Atenas

P

P#	PNOMBRE	COLOR	PESO	CIUDAD
P1	Tuerca	Rojo	12	Londres
P2	Perno	Verde	17	París
P3	Birlo	Azul	17	Roma
P4	Birlo	Rojo	14	Londres
P5	Leva	Azul	12	París
P6	Engrane	Rojo	19	Londres

J

J#	JNOMBRE	CIUDAD
J1	Clasificador	París
J2	Perforadora	Roma
J3	Lectora	Atenas
J4	Consola	Atenas
J5	Compaginador	Londres
J6	Terminal	Oslo
J7	Cinta	Londres

SPJ

S#	P#	J#	CANT
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	200
S2	P3	J5	500
S2	P3	J6	600
S2	P3	J7	400
S2	P5	J2	800
S3	P3	J1	100
S3	P4	J2	200
S4	P6	J3	500
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

- "Obtener Nombres y Ciudades de los Proveedores que suministran por lo menos 50 Unidades de cada parte a por lo menos un proyecto de Atenas"
- Expresión del Cálculo de Tuplas

SX.SNOMBRE, SX.CIUDAD

WHERE ∃ JX FORALL PX ∃ SPJX

(JX.CIUDAD = 'Atenas' AND JX.J# = SPJX.J#
AND PX.P# = SPJX.P# AND SX.S# = SPJX.S#
AND SPJX.CANT >= 50)

Resultado del Algoritmo de Codd

T1 := (S RENAME S# AS SNUM, CIUDAD AS SCIUDAD)

T2 := (P RENAME P# AS PNUM, CIUDAD AS PCIUDAD)

T3 := (J RENAME J# AS JNUM, CIUDAD AS JCIUDAD)

T4 := (SPJ WHERE CANT >= 50)

T5 := (T3 WHERE JCIUDAD = 'Atenas')

T6 := (T1 TIMES T2 TIMES T4 TIMES T5)

T7 := (T6 WHERE J# = JNUM AND P# = PNUM)

T8 := (T7 WHERE S# = SNUM)

T9 := T8[S , P , J]

T10 := (T9 DIVIDEBY T2)

T11 := T10[SNUM , SNOMBRE, SITUACION , SCIUDAD]

T12 := T11 [SNOMBRE , SCIUDAD]

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

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