Cálculo Relacional

Bibliografía: Fundamentos de bases de datos

Korth , Silberschatz

Cálculo Relacional de Tuplas

- Es un **lenguaje de consulta no procedimental**
- **Describe la información** deseada sin dar un procedimiento específico para obtenerla.
- Una **consulta** en el CRT se expresa como

$$\{t \;/\; P(t)\}$$

  – es decir, el conjunto de todas las tuplas t, tal que el predicado P, es verdadero para t.

Ejemplos

Dadas las relaciones r y s:

- la **unión** se expresa

$$\{ t \,/\, r(t) \vee s(t) \}$$

  – es decir, el conjunto de tuplas t tales que t está en r **ó** en s

Ejemplos

Dadas las relaciones r y s:

- la **diferencia** se expresa

$$\mathbf{r - s = \{ t \,/\, r(t) \land \lnot \; s(t) \}}$$

  - es decir, el conj. de tuplas t tales que t está en r y **no** en s

Ejemplos

Dadas las relaciones r y s:

- la **proyección** $\pi_{\mathbf{i1,\ldots ik}}$ **(r)** se expresa

  { t (k) / $\exists$ u (r(u)$\wedge$ t[1]=u[i1] $\wedge$...$\wedge$ t[k]=u[ik]) }

  – donde **t (k)** significa tuplas de grado k

Consultas de ejemplo

- Encontrar el nombre-sucursal, número-préstamo, nombre-cliente y cantidad para préstamos mayores de 1200 dólares.

$$\{t \,/\, t \in \textbf{préstamo} \wedge t[\textbf{cantidad}] > 1200\}$$

- Encontrar los **clientes** que tienen un préstamo mayor de 1200 dólares.

  – Si se desea **únicamente** el atributo **nombre-cliente** (**y no todos)**, se necesita una expresión para una relación sobre el

  esquema (nombre-cliente)

  – Necesitamos aquellas tuplas en (nombre-cliente) tales que exista una tupla en préstamo correspondiente a ese nombre-cliente con el atributo cantidad > 1200

- Para expresar esto necesitamos
  - la **construcción «existe»** de la lógica matemática.

- La notación

$$\exists\ \mathbf{t} \in \mathbf{r(Q\ (t))}$$

  - significa «**existe una tupla t en la relación r tal que el predicado Q(t) es verdadero**».

- Usando esta notación podemos escribir

$$\{ t \ / \ \exists \ s \in \text{préstamo}(t[\text{nombre-cliente}]=s[\text{nombre-cliente}] \wedge s[\text{cantidad}]>1200)\}$$

  - Esto se lee: «el conjunto de todas las tuplas t tal que existe una tupla s en préstamo para la cual los valores de t y s para el atributo nombre-cliente son iguales y el valor de s para el atributo cantidad es mayor de 1200».
  - t se define sólo en el atributo nombre-cliente => el resultado es una relación sobre (nombre-cliente).

- Encontrar los clientes que tienen un préstamo en Perryridge y las ciudades en las que viven.

  – Esta consulta involucra dos relaciones: **cliente** y **préstamo**
  – Se requiere tener **dos** cláusulas «**existe**» en la expresión conectadas por y ($\wedge$)

$$\{ t \,/\, \exists\, s \in \text{préstamo}(t[\text{nombre-cliente}] = s[\text{nombre-cliente}] \wedge s[\text{nombre-sucursal}]=\text{"Perryridge"} \wedge \exists\, u \in \text{cliente} (u[\text{nombre-cliente}] = s[\text{nombre-cliente}] \wedge t[\text{ciudad-cliente}] = u[\text{ciudad-cliente}]))\}$$

- Encontrar los clientes que tienen un préstamo, una cuenta **o** las dos, en la sucursal Perryridge.

  - En **álgebra relacional** se usa la operación **unión**.

  - En **cálculo relacional de tuplas** necesitaremos **dos cláusulas «existe» conectadas por o ($\vee$)**

- Encontrar los clientes que tienen un préstamo, una cuenta **o** las dos, en la sucursal Perryridge.

$\{ t \,/\, \exists\, \mathbf{s} \in \textbf{préstamo}(t[\text{nombre-cliente}] = s[\text{nombre-cliente}]$

$\quad\quad \wedge\ s[\text{nombre-sucursal}] = \text{"Perryridge"})$

$\vee\ \ \exists\, \mathbf{u} \in \textbf{depósito}\ (t\ [\text{nombre-cliente}] = u[\text{nombre-cliente}]$

$\quad\quad \wedge\ u[\text{nombre-sucursal}] = \text{"Perryridge"})\}$

  - Devuelve tuplas **nombre-cliente** que cumplen al menos que:
    - **nombre-cliente** aparece en alguna tupla de **préstamo** con un préstamo en Perryridge
    - **nombre-cliente** aparece en alguna tupla de **depósito** como depositante en Perryridge

- Encontrar los clientes que tienen un préstamo, una cuenta **o** las dos, en la sucursal Perryridge.

$$\{ t \,/\, \exists\, \mathbf{s} \in \textbf{préstamo}(t[\text{nombre-cliente}] = s[\text{nombre-cliente}] \wedge s[\text{nombre-sucursal}] = \text{"Perryridge"}) \vee \exists\, \mathbf{u} \in \textbf{depósito}\ (t\ [\text{nombre-cliente}] = u[\text{nombre-cliente}] \wedge u[\text{nombre-sucursal}] = \text{"Perryridge"}) \}$$

- El resultado de esta consulta es:

- Encontrar únicamente aquellos clientes que tienen una cuenta **y** un préstamo en Perryridge.

{ t / ∃ **s** ∈ **préstamo**(t [nombre-cliente] = s [nombre-cliente]
∧ s [nombre-sucursal] = "Perryridge")
∧ ∃ **u** ∈ **depósito** (t [nombre-cliente] = u [nombre-cliente]
∧ u[nombre-sucursall = "Perryridge")}

- El resultado de esta consulta es:

- Encontrar los clientes que tienen una cuenta en Perryridge pero no un préstamo en ella.

$$\{ t \,/\, \exists\, u \in \text{depósito} \, (t[\text{nombre-cliente}]=u[\text{nombre-cliente}]$$
$$\wedge\; u\,[\text{nombre-sucursal}] = \text{"Perryridge"}$$
$$\wedge\; \neg\, \exists\, s \in \text{préstamo} \, (t[\text{nombre-cliente}]=s[\text{nombre-cliente}]$$
$$\wedge\; s\,[\text{nombre-sucursal}] = \text{"Perryridge"} \,))\}$$

- Encontrar los clientes que tienen una cuenta en Perryridge pero no un préstamo en ella.

$$\{ t \,/\, \exists\, u \in \text{depósito} \,(\, t[\text{nombre-cliente}]=u[\text{nombre-cliente}]$$
$$\wedge\; u\,[\text{nombre-sucursal}] = \text{"Perryridge"}$$
$$\wedge\; \neg\, \exists\, s \in \text{préstamo} \,(\, t[\text{nombre-cliente}]=s[\text{nombre-cliente}]$$
$$\wedge\; s\,[\text{nombre-sucursal}] = \text{"Perryridge"} \,))\}$$

  - $\exists\, u \in$ depósito(...): exige que el cliente tenga una cuenta en Perryridge, y
  - $\neg\, \exists\, s \in$ préstamo(...): elimina los clientes que aparezcan en alguna tupla de préstamo por tener un préstamo de Perryridge.

- Encontrar los clientes que tienen una cuenta en Perryridge pero no un préstamo en ella.

$$\{ t \,/\, \exists\, u \in \text{depósito} \; (t[\text{nombre-cliente}]=u[\text{nombre-cliente}]$$
$$\wedge \; u\,[\text{nombre-sucursal}] = \text{"Perryridge"}$$
$$\wedge \; \neg \exists\, s \in \text{préstamo} \; (t[\text{nombre-cliente}]=s[\text{nombre-cliente}]$$
$$\wedge \; s\,[\text{nombre-sucursal}] = \text{"Perryridge"} \;))\}$$

- El resultado de esta consulta es:

- Encontrar los clientes que tienen una cuenta en **todas** las sucursales situadas en Brooklyn.

  - En **AR** se resuelve con la operación **división**.
  - En **CRT** se introducen: **para todos (∀)** e **implicación (=>)**.

  - **P => Q** significa
    - «si P es verdadera, entonces Q debe ser verdadera».

  - ∀ **t** ∈ **r(Q (t))** significa
    - « Q es verdadera para todas las tuplas t en la relación r ».

- Encontrar los clientes que tienen una cuenta en **todas** las sucursales situadas en Brooklyn.

$$\{t \;/\; \forall\; \mathbf{u} \in \mathbf{sucursal}\ (u\,[\text{ciudad-sucursal}] = \text{"Brooklyn"}$$
$$\mathbf{=>}\ \exists\ \mathbf{s} \in \mathbf{depósito}\ (t\,[\text{nombre-cliente}] = s\,[\text{nombre-cliente}]$$
$$\wedge\quad u\,[\text{nombre-sucursal}] = s[\text{nombre-sucursal}]))\}$$

Es decir:

- el conjunto de todos los clientes (tuplas t (nombre-cliente))
- tal que para **todas** las tuplas u en la relación sucursal,
- si el valor de u en el atributo ciudad-sucursal es Brooklyn
- entonces el cliente tiene una cuenta en la sucursal cuyo nombre aparece en el atributo nombre-sucursal de u.

Definición formal de CRT

Una expresión del **cálculo relacional de tuplas** es de la forma:

$$\{t \ / \ P(t)\}$$

donde

- **t** es una variable de tupla.
- **P** es una **fórmula** construida a partir de **átomos** y **operadores**.
- En una fórmula pueden aparecer varias variables de tuplas.

Definición formal de CRT

Definiciones:

- Una variable de tupla es una **variable libre** si **no** está **cuantificada** por un $\exists$ o por un $\forall$.
- Una variable de tupla **cuantificada** por un $\exists$ o por un $\forall$, es una **variable límite** ó **acotada**.

Por **ejemplo**, en:

$t \in$ préstamo $\wedge \exists s \in$ cliente (t[nombre-cliente]=s[nombre-cliente])

- **t** es una variable **libre**.
- **s** es una variable **límite** ó **acotada.**

Definición formal de CRT

Una **fórmula** en el CRT se compone de **átomos.**

Un **átomo** tiene una de las siguientes formas:

- $s \in r$
- $s[x] \ \alpha \ u \ [y]$
- $s[x] \ \alpha \ c$

Definición formal de CRT

Un **átomo** tiene una de las siguientes formas:

- **s** $\in$ **r** donde **s** es una **variable de tupla** y **r** es una **relación**
- **s[x]** α **u [y]** donde:
  - **s** y **u** son variables de tuplas,
  - **x** es un atributo sobre el que **s** está definida,
  - **y** es un atributo sobre el que **u** está definida, y
  - α es un **operador de comparación**. (<, <=, =, >, >=).
  - **x** e **y** deben tener dominios cuyos miembros puedan compararse.
- **s[x]** α **c** donde:
  - **s** es una variable de tupla,
  - **x** es un atributo sobre el que **s** está definida,
  - α es un operador de comparación, y
  - **c** es una constante en el dominio del atributo **x**.

Definición formal de CRT

Las fórmulas **se construyen** a partir de átomos **usando las siguientes reglas**:

- Un **átomo es** una **fórmula**.
- Si **P1** es una fórmula, entonces también lo son

$$\neg \mathbf{P1} \quad \text{y} \quad \mathbf{(P1)}$$

- Si **P1** y **P2** son fórmulas, entonces también lo son

$$\mathbf{P1} \vee \mathbf{P2}, \quad \mathbf{P1} \wedge \mathbf{P2}, \quad \text{y} \quad \mathbf{P1} \Rightarrow \mathbf{P2}$$

- Si **P1(s)** es una fórmula que contiene una variable de tupla **libre s**, entonces también son

$$\exists\ \mathbf{s} \in \mathbf{r}\ (\mathbf{P1}\ (\mathbf{s})) \quad \mathbf{y} \quad \forall\ \mathbf{s} \in \mathbf{r}\ (\mathbf{P1}(\mathbf{s}))$$

Definición formal de CRT

Como en el caso de AR es posible escribir expresiones equivalentes:

En CRT estas **equivalencias** incluyen tres reglas:

- $\mathbf{P1 \wedge P2}$ es equivalente a $\neg\ \mathbf{(\neg P1 \vee \neg P2)}$
- $\forall\ \mathbf{t} \in \mathbf{r(P1(t))}$ es equivalente a $\neg \exists\ \mathbf{t} \in \mathbf{r(\neg P1(t))}$
- $\mathbf{P1 \Rightarrow P2}$ es equivalente a $\neg\ \mathbf{P1} \vee \mathbf{P2}$

Poder expresivo de los lenguajes

El CRT restringido a expresiones seguras **es equivalente en poder expresivo al AR**.

Es decir:

para cada expresión en el AR

existe una expresión **equivalente** en el CRT

y viceversa.

Definición formal de Cálculo Relacional de Dominios

- Usa **variables** de **dominio** que toman valores del dominio de un atributo.

- Una expresión en el CRD es de la forma

$$\{< x1,x2 , ... , n > \ | \ P(x1, x2 , ... , xn)\}$$

  - donde
    - x1 , x2 , . . . , xn representan **variables de dominio**
    - **P** es una fórmula compuesta por átomos

Definición formal de Cálculo Relacional de Dominios

Un **átomo** en el CRD tiene una de las formas siguientes:

- **< x1 , x2 , . . . , xn >** $\in$ **r** ó (**r(x1 , x2 , . . . , xn)**) donde
  - r es una relación en n atributos y
  - x1 , x2 , . . . , xn son variables de dominio o ctes de dominio.

- **x** α **y** donde
  - **x** e **y** son **variables de dominio**
  - α es un **operador de comparación** (< , <=, = , <>, >, >=).
  - x e y tienen **dominios que puedan compararse** por medio de α

- **x** α **c** donde
  - **x** es una **variable de dominio**,
  - α es un **operador de comparación**
  - **c** es una **constante** en el dominio del atributo correspondiente

Definición formal de Cálculo Relacional de Dominios

Las **fórmulas** se construyen a partir de **átomos** usando las **reglas** siguientes:

- Un **átomo** es una **fórmula**.
- Si **P1** es una fórmula, entonces también lo son

$$\neg \mathbf{P1} \quad \text{y} \quad \mathbf{(P1)}$$

- Si **P1** y **P2** son fórmulas, entonces también lo son

$$\mathbf{P1 \ v \ P2}\ , \quad \mathbf{P1} \wedge \mathbf{P2}\ , \ \text{y} \quad \mathbf{P1 \Rightarrow P2}$$

- Si **P1(x)** es una fórmula en x, donde x es una variable de **dominio**, entonces también son **fórmulas**

$$\exists \ \mathbf{x\ (P1\ (x)\ )} \quad \text{y} \quad \forall \ \mathbf{x\ (P1(x))}$$

Consultas de ejemplo

- Encontrar **nombre** de sucursal, **número** de préstamo, **nombre** de cliente y **cantidad** de préstamos mayores de 1200 dólares.

$$\{<b,l,c,a> \;/\; <b,l,c,a> \in \text{ préstamo} \wedge a>1200\}$$

- Encontrar los clientes que tienen un préstamo por una cantidad mayor de 1200 dólares.

$$\{<c> \; / \; \exists \; b,l,a \; (<b,l,c,a> \in \text{préstamo} \wedge \; a>1200)\}$$

- Encontrar **clientes** que tienen un préstamo de sucursal Perryridge y **ciudad** en que viven.

$$\{ < c,x > \ / \ \exists\ b,l,a\ (<b,l,c,a> \in \text{préstamo}$$
$$\wedge\ b=\text{"Perryridge"}$$
$$\wedge\ \ \exists\ y\ (<c,y,x> \in \text{cliente}\ ))\}$$

- Encontrar clientes que tienen un **préstamo**, una **cuenta**, **o ambos** en sucursal Perryridge.

$$\{ <c> \ / \ \exists \ b,l,a \ (< b,l,c,a > \in \text{préstamo} \wedge b = \text{"Perryridge"} \vee \exists \ b,a,n \ (<b,a,c,n> \in \text{depósito} \wedge b = \text{"Perryridge"})\}$$

- Encontrar clientes que tienen una cuenta en **todas** las sucursales situadas en Brooklyn:

$$\{\langle c\rangle \ / \ \forall\, x,y,z\ ((\ \langle x,y,z\rangle \in \text{sucursal})$$

$$\wedge\ z = \text{"Brooklyn"}\ \mathbf{=>}\ (\exists\ a,n\ (\ \langle x,a,c,n\rangle \in\ \text{depósito}\ )))\}$$

Poder expresivo de los lenguajes

Son **equivalentes**:

- El álgebra relacional.
- El cálculo relacional de tuplas.
- El cálculo relacional de dominios.

Completitud relacional

- Un **lenguaje es relacionalmente completo** si es al menos **tan expresivo** como el **álgebra**,
  - es decir si sus expresiones permiten la definición de cualquier relación que pueda definirse mediante expresiones del álgebra.

Como el álgebra es relacionalmente completa

para **demostrar** que **cualquier lenguaje** L es **completo**

basta demostrar que L incluye análogos de cada una de las **cinco operaciones algebraicas primitivas: selección, proyección, producto cartesiano,**

unión y resta.

- SQL, QUEL, QBE son completos.

Comparación de lenguajes algebraicos y de cálculo

- Los lenguajes de **cálculo** son de **más alto nivel** que los **algebraicos** porque:
  - lenguajes algebraicos especifican el orden de las operaciones
  - lenguajes de cálculo dejan que el compilador determine la manera (el orden) más eficiente

Ejemplo

Dadas las relaciones R(A,B) y S(B,C)

1 - la expresión algebraica:

$$\pi_{C}\, \sigma_{A=a1}(\, R \,|X|\, S\,) \text{ significa}$$

- "listar los valores C asociados con el valor A=a1 en la relación JOIN de columnas ABC"

- esta expresión **da un orden particular de operaciones**
  - 1º) join natural de r y s => ordena los valores B en ambas relaciones
  - 2º) selección con A=a1
  - 3º) muestra los valores C asociados

2 - La expresión algebraica **1 -** es equivalente a

$$\pi_C\ (\pi_B\ (\ \sigma_{A=a1}(R))\ |X|\ S)$$

- para evaluar esta realiza:
  - 1º) la selección en R de las tuplas con A= a1
  - 2º) encuentra los B asociados
  - 3º) asocia las tuplas de S solamente para esos B
  - 4º) muestra los valores C asociados

3 - En el cálculo de dominios:

$$\{c/\ \exists\, b\ (R(a_1 b) \wedge S(bc)\}$$

expresa lo que se quiere

Observaciones:

- 1, 2 y 3 son equivalentes
- Dependiendo de la organización de R y S la opción **1** puede llevar más tiempo que **2**.
- **Optimización** => convertir una expresión a una equivalente de menor costo