Universidad de Granada

decsai.ugr.es

Fundamentos de Bases de Datos

Grado en Ingeniería Informática

Seminario 6: Cálculo relacional

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

Indice del tema

- Esquema general del tema:
  - Introducción: los modelos lógicos y las bases de datos relacionales. Nota histórica.
  - El cálculo de predicados como lenguaje para la representación de información.
  - Las bases de datos relacionales como un modelo de cálculo de predicados. Analogías intuitivas.
  - El cálculo relacional orientado a tuplas.

- Inicialmente:
  - Elementos lógicos en el modelo relacional (Codd 1971).
    - Cálculo relacional orientado a tuplas. Lenguaje Alpha.
    - Equivalencia entre enfoques de consulta.
  - Extensiones e implementaciones iniciales:
    - Lacroix y Pirrotte (1977). Primera versión del cálculo de dominios.
    - Implementaciones: QUEL(1980), QUERY_BY_EXAMPLE (1985).

- Conexión de los datos existentes en la base de datos con información “inteligente”:
  - Formalización de los modelos lógicos de Bases de Datos:
    - Desarrollos teóricos.
      - Lógica y Bases de Datos (Gallaire Minker y Nicolas 78,81,84)
      - Inteligencia Artificial y Bases de Datos (Brodie t. al. 81,84,86) (Reiter 1984)
    - Implementaciones:
      - Basadas en acoplamientos con PROLOG: Educe, PRO_SQL, QUINTUS_PROLOG etc.. (1986,87..)
      - Basadas en DATALOG: Nail

- Idea básicas:
  - El cálculo de predicados surge como sistema de representación del conocimiento en I.A.
  - Elementos de un sistema de representación del conocimiento:
    - Una Base de Conocimiento donde se almacena conocimiento a distintos niveles.
    - Un mecanismo de inferencia que permite derivar un conocimiento de otro.

El Cálculo de predicados como elemento de representación de información

- Para representar la información en la base de conocimiento los formalismos de la Lógica son muy adecuados ya que incluyen:
  - Mecanismos de representación
  - Mecanismos de derivación de conocimiento

- Entre los formalismos basados en la Lógica:
  - Cálculo de Proposiciones
  - Cálculo de Predicados
  - Formalismos Lógicos más avanzados:
    - Redes semánticas
    - Lógica multivaluada
    - Razonamiento por defecto, basado en casos etc..

El Cálculo de predicados como elemento de representación de información

- Definición formal: ideas básicas

  - Un lenguaje de Cálculo de Predicados se define para describir un "mundo". Debe tener símbolos y frases.

  - Este lenguaje debe incluir un alfabeto donde haya:
    - Símbolos para describir los objetos del mundo (constantes)
      - Juan, José, ...Seat,..... Rojo,....etc, GR-150-A....
    - Símbolos para describir funciones que nos dan unos objetos en función de otros:
      - Color, Padre, Madre, Propietario etc...
    - Símbolos para describir variables: x, y, z,
    - Símbolos de predicados que describan relaciones entre objetos:
      - Casados, Conduce, Prefiere.
    - Los predicados describen relaciones binarias, ternarias etc...

El Cálculo de predicados como elemento de representación de información

- El lenguaje además de símbolos tendrá que generar “frases” (fórmulas, expresiones...) por ello debe tener
  - Símbolos de puntuación: ( ) , . ; [ ]
  - Conectores: $\wedge$, $\vee$, $\neg$, $\rightarrow$
  - Cuantificadores: $\forall$, $\exists$

El Cálculo de predicados como elemento de representación de información

- Definición formal: Un lenguaje de Cálculo de Predicados se define como L=(S, W) donde
  - S es un conjunto de símbolos incluyendo:
    - Constantes
    - Funciones
    - Variables
    - Predicados
    - Símbolos adicionales
  - W un conjunto de frases “correctas” o fórmulas bien formadas (Well formed formulae) o “wff”
  - W se define de forma recursiva:
    - Un átomo a se define cómo:
      - Un símbolo de constante (Jose) ó
      - Un símbolo de variable (x) ó
      - f(a) donde f es un símbolo de función y a un átomo: Padre(José), Color(x)

El Cálculo de predicados como elemento de representación de información

- Una formula atómica se define como $p(a_1,a_2,..a_n)$ donde:
  - p es un símbolo de predicado n-ario
  - $a_1,a_2,....a_n$ son átomos
- Toda formula atómica es wff ($\in W$)
- Si $f_1,f_2 \in W$ entonces:
  - $f_1 \vee f_2 \in W$; $f_1 \wedge f_2 \in W$; $\neg f_1 \in W$; $f_1 \rightarrow f_2 \in W$
- Si $f_1(x) \in W$ entonces:
  - $\forall x\ f_1(x) \in W$; $\exists x\ f_1(x) \in W$

– Algunos ejemplos de wffs:
  - casados(Juan,Ana), casados(padre(x),madre(x)),prefiere(Ana,Honda,rojo)
  - conduce(Juan,GR-150-A)$\wedge\neg$conduce(propietario(GR-150-A),GR-150-A)
  - $\forall$x coche(x)$\rightarrow$prefiere(propietario(x),marca(x),color(x))
  - $\exists$y (persona(y) $\wedge\neg$casados(padre(y),madre(y)))

– Toda variable en una wff que no esté cuantificada se denomina variable libre
– En caso contrario se denomina variable ligada

El Cálculo de predicados como elemento de representación de información

Interpretación de un lenguaje

- Idea básica: un lenguaje L=(A,W) es una abstracción formal que puede describir muchas realidades. Para describir una concreta es necesario asociar
  - Símbolos de constantes con objetos del mundo
  - Predicados con relaciones concretas entre objetos

- Formalmente:

  - Sea L=(A,W) un lenguaje de CP ;$C \subset A$ es el conjunto de constantes
  - Llamaremos interpretación I de L al triple I=(D,K,E), donde
    - D un "universo de discurso": conjunto de objetos asociados a una realidad
    - $K: C \rightarrow D$ y permite asociar las constantes de A a objetos reales.
    - E se denomina "función extensión" y asocia a todo predicado n-ario $p \in A$ un conjunto $E(p) \subseteq D^n$. E(p) se denomina extensión de *p* en I.
    - A partir de ahora cuando hablemos de una interpretación identificaremos cada objeto con su nombre es decir, $\forall c \in C$ identificaremos c y K(c)

El Cálculo de predicados como elemento de representación de información

Interpretación de un lenguaje

- Valor de verdad: toda interpretación I=(D,K,E) de un lenguaje L=(A,W) permite asociar valores de verdad a ciertas wffs de W.
  - Toda wff que no incluya variables tiene un valor de verdad:
    - Toda fórmula atómica de la forma $P(c_1,..c_n)$ con $c_i \in C$ o $c_i=f(d_i)$ con $d_i \in C$ es cierta sii $(c_1,c_2...c_n) \in E(P)$, en caso contrario es falsa.
    - Sean $f_1, f_2 \in W$, $\forall$ el valor de verdad de:

      $$f_1 \vee f_2,\ f_1 \wedge f_2\ ,\ \neg f_1,\ \text{y}\ f_1 \rightarrow f_2$$

      se calculan de acuerdo con las reglas del "or" "and" y "not", y not(f_1 or f_2) supuestos conocidos los valores de verdad de f_1 y f_2
  - Toda wff que tenga todas sus variables ligadas tiene un valor de verdad de acuerdo con:
    - $\forall x\ f_1(x)$ es cierta si $f_1(c)$ es cierta $\forall c \in C$
    - $\exists x\ f_1(x)$ es cierta si $\exists c \in C$ para la que $f_1(c)$ es cierta
  - Una wff que tenga alguna variable libre genera un conjunto de constantes que son aquellas que hacen cierta la formula sustituyendo la variable por ellas.

- Modelo: dada una interpretación I=(D,K,E) de un lenguaje L=(A,W) y $M \subset W$, I es un modelo de M si toda $f \in M$ es cierta con respecto a I.

El Cálculo de predicados como elemento de representación de información

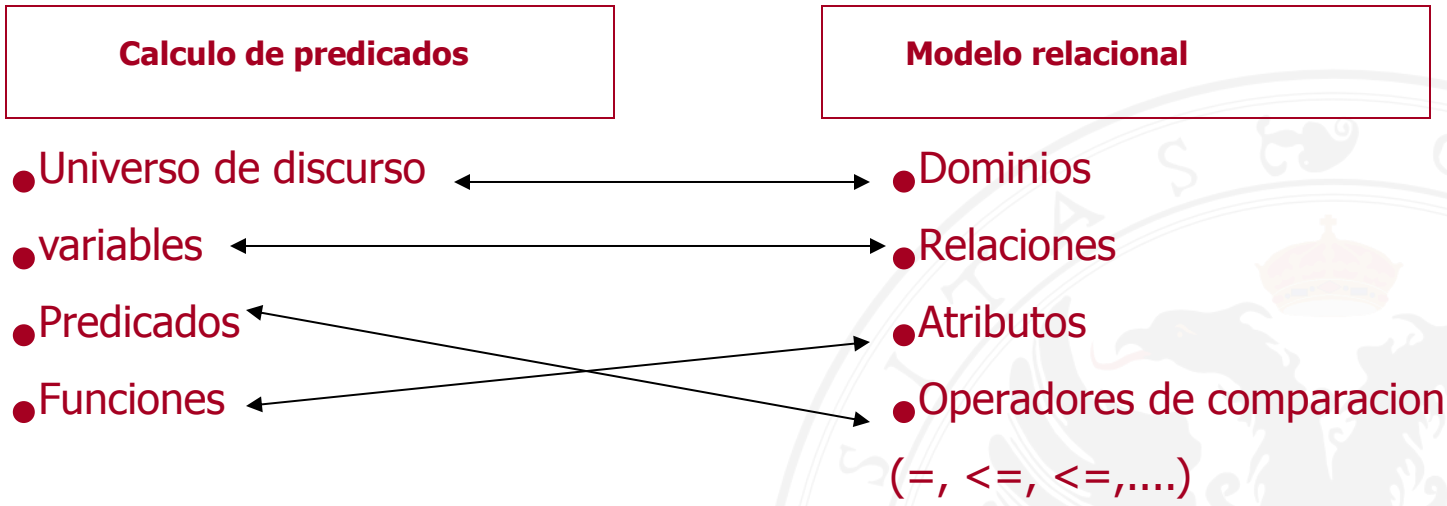
- Ejemplos:
  - casados(Juan,Ana) será cierta si (Juan,Ana)$\in$E(casados)
  - prefiere(Ana,Honda,rojo) es cierta si (Ana,Honda,rojo)$\in$E(prefiere)
  - conduce(Juan,GR-150-A)$\wedge\neg$conduce(propietario(GR-150-A),GR-150-A) sera cierta si conduce(Juan,GR-150-A) es cierta y conduce(propietario(GR-150-A),GR-150-A) es falsa
  - $\exists$y (persona(y) $\wedge\neg$casados(padre(y),madre(y))) será cierta si podemos encontrar una constante c tal que
    - (persona(c) $\wedge\neg$casados(padre(c),madre(c)))
  - x | casados(padre(x),madre(x)) define un conjunto de constantes

Analogías intuitivas entre el calculo de predicados y el modelo relacional

- Ideas básicas:
  - Un lenguaje de calculo de predicados y un modelo relacional son estructuras formales para describir la realidad: ambas pueden identificarse.
  - Una instancia de una base de datos se identificaría entonces con una interpretación de su lenguaje asociado.
  - Las reglas de integridad serían wffs y la interpretación debería ser un modelo para ellas.
  - Las consultas se generarían mediante wffs con variables libres. Los conjuntos de constantes que las hacen ciertas serán la solución de la consulta.

Analogías intuitivas entre el calculo de predicados y el modelo relacional

- Identificación en el caso del calculo relacional orientado a tuplas



- Cambio de notación:
  - Operadores de comparación: notación de operador
    - =(a,b) se sustituye por a=b, etc...
  - Funciones: notación de atributo
    - f(x) se sustituye por x.f

El calculo relacional orientado a tuplas

- Definición de una consulta:
  - Consideremos una base de datos con relaciones $R(A_1,...A_n)$, $S(B_1,...B_m)$ etc... y le asociamos un lenguaje de Cálculo de Predicados.
  - Supongamos que R_x, R_y ....S_x, S_y.. etc.. son variables que toman valores en R, S etc.. Son variables tupla.
  - Una consulta en C.R. orientado a tuplas (lenguaje QUEL) tiene la forma:

$$\text{Select } R_x.A_i, R_x.A_j ..., R_y.A_h, S_x.B_l,... \text{ Where } wff(R_x,R_y,S_x...)$$

- $R_x.A_i, R_x.A_j ..., R_y.A_h,S_x.B_l,...$ se denomina “lista objetivo” .
- $wff(R_x,R_y,S_x...)$ es una fórmula cuyas variables libres aparecen en la lista objetivo.
- La particularización de la lista objetivo para las tuplas que hacen cierta esta fórmula nos da la solución a la consulta.

El calculo relacional orientado a tuplas

- Ejemplo:
  - Modelo relacional
    - S(S#,nombres,ciudad,status)
    - P(P#,tipop,peso,color,ciudad)
    - J(J#,nombre,ciudad,director,presupuesto)
    - E(S#,P#,J#,cantidad,fecha)
  - Lenguaje
    - Constantes: s_1,s_2,..., p_1,p_2..., Madrid,...., 24, 25 ,..., etc.....
    - Variables:
      - Range S_x, S_y ... in S, Range P_x, P_y, ...in P
    - Funciones: S#, nombres,....P#,.....
  - Consultas
    - Select S_x.S#,S_x.nombres,S_x.ciudad where S_x.status=25
    - Select S_x.S#,S_x.nombres where $\exists E_y(E_y.s\#=S_x.S\# \wedge E_y.p\#=`p_1' \wedge E_y.cantidad>=200)$

Ejemplos

Trabajadores (id_trabajador, nombre, trf_hr, tipo_de_oficio, id_supv)
Edificios (id_edificio, dir_edificio, tipo, nivel_calidad, categoria)
Asignaciones (id_trabajador, id_edificio, fecha_inicio, num_dias)
Oficios (tipo_de_oficio, prima, horas_por_sem)

Ejemplos

- Encontrar los datos de aquellos trabajadores que son electricistas:

```
RANGE Tx IN Trabajadores
SELECT Tx.*
WHERE Tx.tipo_de_oficio=' Electricista'
```

$\sigma_{\text{tipo\_de\_oficio=' Electricista'}}$ TRABAJADORES

- Encontrar el nombre de aquellos trabajadores que son electricistas:

```
RANGE Tx IN Trabajadores
SELECT Tx.nombre
WHERE Tx.tipo_de_oficio=' Electricista'
```

$\Pi_{\text{nombre}}(\sigma_{\text{tipo\_de\_oficio=' Electricista'}}$ TRABAJADORES)

Ejemplos

- Encontrar el número de horas semanales que trabaja cada trabajador:

```
RANGE Tx IN Trabajadores
RANGE Ox IN Oficios
SELECT Tx.nombre, Ox.horas_por_sem
WHERE Tx.tipo_de_oficio=Ox.tipo_de_oficio
```

$\Pi_{nombre,horas\_por\_sem}$ (TRABAJADORES|X|OFICIOS)

Ejemplos

- Encontrar los nombres de trabajadores que han trabajado tanto en el edificio 312 como en el edificio 460:

RANGE Tx IN Trabajadores

RANGE Ax, Ay IN Asignaciones

SELECT Tx.nombre

WHERE ($\exists$Ax,Ay ((Ax.id_trabajador=Tx.id_trabajador) $\wedge$
(Ay.id_trabajador=Tx.id_trabajador) $\wedge$ (Ax.id_edificio=312) $\wedge$
(Ay.id_edificio=460))

Π_{nombre} (TRABAJADORES|X| (($\Pi_{id\_trabajador}$ $\sigma_{id\_edificio=312}$ ASIGNACIONES) $\cap$
($\Pi_{id\_trabajador}$ $\sigma_{id\_edificio=469}$ ASIGNACIONES))

Ejemplos

- Encontrar los nombres de trabajadores que han trabajado o en el edificio 312 o en el edificio 460:

RANGE Tx IN Trabajadores

RANGE Ax IN Asignaciones

SELECT Tx.nombre

WHERE $\exists$Ax ((Ax.id_trabajador=Tx.id_trabajador) $\wedge$
((Ax.id_edificio=312) $\vee$ (Ax.id_edificio=460)))

Π_{nombre}(TRABAJADORES|X|$\sigma_{id\_edificio=312 \vee id\_edificio=460}$ ASIGNACIONES)

Ejemplos

- Encontrar el nombre de aquellos trabajadores que no han trabajado en el edificio 312:

RANGE Tx IN Trabajadores

RANGE Ax IN Asignaciones

SELECT Tx.nombre

WHERE $\neg(\exists$Ax ((Ax.id_trabajador=Tx.id_trabajador) $\wedge$
(Ax.id_edificio=312)))

Π_{nombre} (TRABAJADORES |X| ($\Pi_{id_trabajador}$TRABAJADORES -
$\Pi_{id\_trabajador}$($\sigma_{id\_edificio=312}$ ASIGNACIONES)))

Ejemplos

- Encontrar parejas de trabajadores que tengan el mismo oficio:

RANGE Tx, Ty IN Trabajadores

SELECT Tx.nombre, Ty.nombre

WHERE (Tx.tipo_de_oficio=Ty.tipo_de_oficio) $\wedge$
Tx.nombre<Ty.nombre)}

A:= $\Pi_{\text{nombre, tipo\_de\_oficio}}$ TRABAJADORES

B:= $\Pi_{\text{nombre, tipo\_de\_oficio}}$ TRABAJADORES

$\Pi_{\text{A.nombre, B.nombre}}$ ($\sigma_{\text{A.tipo\_de\_oficio=B.tipo\_de\_oficio} \wedge \text{A.nombre<B.nombre}}$ (AXB))

Ejemplos

- Encontrar aquellos edificios en los que han trabajado todos los trabajadores de la empresa:

RANGE Tx IN Trabajadores

RANGE Ex IN Edificios

RANGE Ax IN Asignaciones

SELECT Ex.*

WHERE $\forall$Tx ($\exists$Ax ((Ax.id_trabajador=Tx.id_trabajador) $\wedge$ (Ax.id_edificio=Ex.id_edificio)))

($\Pi_{\text{id\_edificio, id\_trabajador}}$ ASIGNACIONES)

$\div$

($\Pi_{\text{id\_trabajador}}$ TRABAJADORES)