Bases de Datos
Tema 5: Cálculo relacional

Dpto. de Ingeniería Informática

Contenidos

- Introducción
- Relaciones de ejemplo
- Cálculo relacional
- Cálculo relacional de tuplas
- Cálculo relacional de dominios
- Referencias

Introducción

- El modelo de datos relacional tiene asociado dos partes:
  - *Estática*: formada por las estructuras que almacenan a los datos y las restricciones que soportan estos datos
  - *Dinámica*: transformaciones entre estados de la Base de Datos
  - Si O es un operador, se puede pasar de un estado origen (BD_i) a un estado destino (BD_j) de la Base de Datos como sigue:

$$O(BD_i) = BD_j$$

  - Estos estados deben satisfacer las restricciones de integridad estática, y la transformación ha de cumplir las restricciones de integridad dinámica (entre estados)

Introducción

- El modelo relacional actúa sobre conjuntos de tuplas mediante lenguajes de manipulación relacionales, que asocian una sintaxis concreta a las operaciones
- Se dividen en dos tipos:
  - *Algebraicos*: los cambios de estado se especifican mediante operaciones, los operandos son relaciones y el resultado es una relación. Se conoce como álgebra relacional a «una colección de operaciones que sirven para manipular relaciones enteras»
  - *Predicativos*: los cambios de estado se especifican mediante predicados que definen el estado objetivo sin indicar las operaciones que hay que realizar para llegar al mismo. Basados en el cálculo de predicados

Relaciones

- En los ejemplos usamos la BD *Biblioteca*:
  - Autores:

Cód_Aut	Nombre	Apellido
1	Neal	Stephenson
2	Lawrence	Lessig
3	Noam	Chomsky

  - Editores:

Cód_Edit	Nombre	Apellido
1	Santiago	Ceri
2	Antón	De Miguel
3	Lawrence	Lessig

Relaciones

- En los ejemplos usamos la BD *Biblioteca*:
  - Socios:

Cód_Soc	Nombre	Apellido
1	Eric	Manrique
2	Mar	García

  - Libros:

Cód_Lib	Nombre	Cód_Edit	Cód_Aut
1	SSOO	1	1
2	BBDD	2	2
3	ABBDD	3	2

Relaciones

- En los ejemplos usamos la BD *Biblioteca*:
  - Préstamos:

Cód_Lib	Cód_Soc	Fecha
1	1	19/3/2010
2	1	9/10/2010
3	1	18/12/2010
1	2	5/1/2017

Cálculo relacional

- Lenguaje basado en el cálculo de predicados de primer orden
- Lenguaje *no procedimental*: se expresa lo que se quiere obtener como resultado, pero no cómo obtenerlo
  - Da mucha potencia y comodidad
  - Pero cuidado, a veces se puede caer en incongruencias (pedir algo que no puede existir, pedir algo que no es lo que se desea, ...) sin darse cuenta
    - “Se pierde” el seguimiento de cada operador que hay en álgebra (que “se parece” a la programación clásica)

Cálculo relacional

- Ha tenido una gran influencia en el diseño de lenguajes comerciales, especialmente SQL
- Se divide en:
  - Cálculo relacional de tuplas: se indican las condiciones que deben cumplir las tuplas del resultado
  - Cálculo relacional de dominios: se indican las condiciones que deben cumplir las *variables de dominio*

Cálculo relacional

- En cualquiera de los dos modelos de Cálculo relacional el resultado que quiero obtener es un …

Cálculo relacional

- En cualquiera de los dos modelos de Cálculo relacional el resultado que quiero obtener es un …
  - Conjunto de tuplas
  - Por lo que toda fórmula será *{ … }*

Cálculo relacional de tuplas

- Una *variable de tupla* es una variable que almacena tuplas de un esquema de relación en concreto
- Una consulta se expresa como:

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

siendo *t* una variable de tupla

- El resultado de esta consulta es el conjunto de todas las tuplas *t* para las que el predicado *P* es verdadero

Cálculo relacional de tuplas

- *Condiciones de pertenencia*: sea t una variable de tupla y R una relación, la pertenencia se representan de la forma R(t) y será verdadera si t ∈ R
- Lo habitual es comenzar el predicado indicando que una variable de tupla pertenece a una relación concreta
  - Se puede indicar con R(t) o t ∈ R
  - Esa relación R debe estar definida previamente (si no está definida en el enunciado del problema, hacerlo explícitamente)

Cálculo relacional de tuplas

- Una vez indicada una relación de pertenencia, están especificados los atributos que tendrán las variables de tupla
- Y se permite acceder a ellos con el operador punto:

t.atributo1

t.atributo2

Sintaxis de las consultas

- *Condiciones de comparación*: se construyen combinando variables (o atributos de ellas) y operadores de comparación y aritméticos

t.atributo1 > 1

t.atributo2 + 10 = t.atributo3

…

  - OJO: igual que en álgebra relacional, no se pueden usar consultas que devuelvan un único valor como escalares. Las consultas devuelven conjuntos

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los libros cuyo código de autor sea mayor que 1»

$$\{t \mid t \in \text{Libros} \land t.\text{Cód\_Aut} > 1\}$$

ó

$$\{t \mid \text{Libros}(t) \land t.\text{Cód\_Aut} > 1\}$$

Tuplas t de la relación Libros y que tengan código de autor > 1

Cód_Lib	Nombre	Cód_Edit	Cód_Aut
2	BBDD	2	2
3	ABBDD	3	2

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los libros cuyo código de autor sea mayor que 1»

$$\{t \mid t \in \text{Libros} \wedge t.\text{Cód\_Aut} > 1\}$$

- Suponiendo que el sistema evaluara de izquierda a derecha:
  - En un primer paso *t* se instanciaría (almacenaría) todas las tuplas de Libros
  - Después se quedarían sólo aquellas que cumplieran la segunda parte del predicado

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los libros cuyo código de autor sea mayor que 1»

$$\{t \mid t.Cód\_Aut > 1\}$$

ó

$$\{t \mid t.Cód\_Aut > 1\}$$

- ¿Qué resultado daría?

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los libros cuyo código de autor sea mayor que 1»

$$\{t \mid t.Cód\_Aut > 1\}$$

ó

$$\{t \mid t.Cód\_Aut > 1\}$$

- ¿Qué resultado daría?
  - Error, no se ha indicado a qué relación pertenece t
  - Se desconocen sus atributos, dominios, operaciones que aceptan, etc

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los editores de libros cuyo código de autor sea mayor que 1»

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los editores de libros cuyo código de autor sea mayor que 1»

$$\{t.Cód\_Edit \mid t \in Libros \land t.Cód\_Aut > 1\}$$

ó

$$\{t.Cód\_Edit \mid Libros(t) \land t.Cód\_Aut > 1\}$$

Atributo Cód_Edit de las tuplas t de la relación Libros y que tengan código de autor > 1

Cód_Edit
2
3

Cálculo relacional de tuplas

- ¿Qué atributos se pueden usar en el predicado P?

$$\{t.atrib1, t.atrib2 | P(t) \}$$

- Aquellos que estén en la relación a la que pertenezca t

Cálculo relacional de tuplas

- Ejemplo: «encontrar todos los editores de libros cuyo código de autor sea mayor que 1»

$$\{t.Cód\_Edit \mid t \in Libros \land t.Cód\_Aut > 1\}$$

- Suponiendo que el sistema evaluara de izquierda a derecha:
  - En un primer paso principio t se instanciaría (almacenaría) todas las tuplas de Libros
  - Después se quedarían sólo aquellas que cumplieran la segunda parte del predicado
  - Finalmente sólo incluiría en el resultado los atributos indicados

De cálculo relacional de tuplas a SQL

- Ejemplo: «encontrar todos los editores de libros cuyo código de autor sea mayor que 1»

$$\{t.Cód\_Edit \mid t \in Libros \land t.Cód\_Aut > 1\}$$

- ¿Cómo sería en SQL?

De cálculo relacional de tuplas a SQL

- Ejemplo: «encontrar todos los editores de libros cuyo código de autor sea mayor que 1»

$$\{t.Cód\_Edit \mid t \in Libros \land t.Cód\_Aut > 1\}$$

- ¿Cómo sería en SQL?

```
SELECT t.Cód_Edit
FROM Libros AS t
WHERE t.Cód_Aut > 1;
```

Correspondencia directa de cada elemento

Sintaxis de los predicados

- Formalmente una expresión tiene la forma

  $\{ t_1.A_1, t_2.A_2, \ldots , t_n.A_n \mid COND(t_1, t_2, \ldots , t_n, \ldots , t_m)\}$

- Donde:
  - Los t_i son variables de tupla
  - Los A_i son atributos de las relaciones de los t_i
  - COND es una condición o una fórmula de cálculo relacional de tuplas

Sintaxis de los predicados

- Si COND es una fórmula F, F representa el conjunto de todas las tuplas tal que la fórmula *F(t)* sea verdadera
- La fórmula se construye combinando átomos de cálculo de predicados, que pueden ser:
  - $t_i \in R$ ó $R(t_i)$
  - t_i.A *op* t_j.B, donde *op* está en {=, >, ≥, <, ≤, ≠}
  - t_i.A *op const* ó *const op* t_i.A (siendo const una constante)

Sintaxis de los predicados

- Si COND es una fórmula F, F representa el conjunto de todas las tuplas tal que la fórmula *F(t)* sea verdadera
- La fórmula se construye combinando átomos de cálculo de predicados, que pueden ser:
  - $t_i \in R$ ó $R(t_i)$
  - $t_i.A$ *op* $t_j.B$, donde *op* está en {=, >, ≥, <, ≤, ≠}
  - $t_i.A$ *op const* ó *const op* $t_i.A$ (siendo *const* una constante)
- NO se aceptan otros constructores

Sintaxis de los predicados

- Respetando la formulación anterior todo átomo siempre se evaluará a VERDADERO o FALSO para un conjunto de tuplas concreto
  - $t_i \in R$ ó $R(t_i)$ es VERDADERO si a t_i puede asignarse una tupla de R (si R no es vacía, …)
  - Para los casos de operadores, se sigue el comportamiento habitual
- Las fórmulas atómicas F_i pueden formar otras fórmulas combinándose con operadores lógicos:
  - $\neg F_i$ (not), $F_i \wedge F_j$ (and), $F_i \vee F_j$ *(or)*

Sintaxis de los predicados

- El cálculo ofrece dos tipos de cuantificadores:
  - *Cuantificador existencial*: la fórmula (∃t)(F) será verdadera si la fórmula F es verdadera para al menos una tupla asignada a las ocurrencias libres de t en F. En caso contrario será falsa
  - *Cuantificador universal*: la fórmula (∀t)(F) será verdadera si la fórmula F es verdadera para toda tupla asignada a las ocurrencias libres de t en F. En caso contrario será falsa

Importante respetar la sintaxis (paréntesis)

Sintaxis de los predicados

- Variables de tuplas libres o ligadas
  - Una ocurrencia de una variable de tupla en una fórmula F que es un átomo está libre en F
  - Una ocurrencia de una variable de tupla en una fórmulas combinada F estará libre o ligada dependiendo de cómo esté en sus componentes
    - En F = ($F_i \wedge F_j$) puede ser que una variable esté libre en F_i pero no en F_j, viceversa, no aparecer en ambas, …
  - Toda ocurrencia libre de una variable de tupla en una fórmula cuantificada F' estará ligada

Sintaxis de los predicados

- Ejemplos de variables de tuplas libres o ligadas
  - Aunque que no se hayan presentado los cuantificadores, ¿cómo estarían *d* y *t* en cada fórmula?

    F_1=(d.Nombre=”Alejandro”)

    •

    F_2=(∃t)(d.DNI=t.DNI)

    •

    F_3=(∀d)(d.tlf=”954954954”)

    •

Sintaxis de los predicados

- Ejemplos de variables de tuplas libres o ligadas
  - Aunque que no se hayan presentado los cuantificadores, ¿cómo estarían *d* y *t* en cada fórmula?

    F_1=(d.Nombre=”Alejandro”)
    - *d está libre*

    F_2=(∃t)(d.DNI=t.DNI)
    - *d está libre y t ligada*

    F_3=(∀d)(d.tlf=”954954954”)
    - d está ligada

Sintaxis de los predicados

- Las únicas variables de tuplas no ligadas deben ser las de la izquierda de la |
  - Son las que tomarán valores que “hagan verdadera” la condición de la derecha de la |
  - Las demás deben “acotarse” para desambiguar
  - Ejemplo: ¿qué significaría?

$$\{ t.Nombre \mid t \in Editor \wedge$$

$$\wedge r \in Libros \wedge (r.Cód\_Edit = t.Cód\_Edit) \}$$

Sintaxis de los predicados

- Las únicas variables de tuplas no ligadas deben ser las de la izquierda de la |
  - Son las que tomarán valores que “hagan verdadera” la condición de la derecha de la |
  - Las demás deben “acotarse” para desambiguar
  - Ejemplo: ¿qué significaría?

$$\{ t.Nombre \mid t \in Editor \wedge$$

$$\wedge r \in Libros \wedge (r.Cód\_Edit = t.Cód\_Edit) \}$$

¿los nombres de los editores con <u>algún</u> libro editado? O ¿los nombres del editor que ha editado <u>todos</u> los libros?

Ejemplos

- Listar los códigos de los socios que tienen al menos un libro en préstamo

Ejemplos

- Listar los códigos de los socios que tienen al menos un libro en préstamo

$$\{ t.Cód\_Soc \mid Prestamos(t) \}$$

Ejemplos

- Listar los códigos de los socios que tienen al menos un libro en préstamo

$$\{ t.Cód\_Soc \mid Prestamos(t) \}$$

- Listar los códigos de los socios que tienen al menos un libro en préstamo y el código del libro prestado

Ejemplos

- Listar los códigos de los socios que tienen al menos un libro en préstamo

$$\{ t.Cód\_Soc \mid Prestamos(t) \}$$

- Listar los códigos de los socios que tienen al menos un libro en préstamo y el código del libro prestado

$$\{ t.Cód\_Soc, t.Cód\_Lib \mid Prestamos(t) \}$$

- ¿Grado y cardinalidad de las dos consultas?
  - Grado: la primera VERSUS la segunda
  - Cardinalidad: la primera VERSUS la segunda

Ejemplos

- Listar los códigos de los socios que tienen al menos un libro en préstamo

$$\{ t.Cód\_Soc \mid Prestamos(t) \}$$

- Listar los códigos de los socios que tienen al menos un libro en préstamo y el código del libro prestado

$$\{ t.Cód\_Soc, t.Cód\_Lib \mid Prestamos(t) \}$$

- ¿Grado y cardinalidad de las dos consultas?
  - Grado: la primera < la segunda
  - Cardinalidad: la primera ≤ la segunda

Ejemplos

- Listar los nombres de los socios que tienen al menos un libro en préstamo

Ejemplos

- Listar los nombres de los socios que tienen al menos un libro en préstamo

$$\{\ t.Nombre \mid t \in Socios \wedge$$

$$\wedge\ (\exists r)\ ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc))\ \}$$

Ejemplos

- Listar los nombres de los socios que tienen al menos un libro en préstamo

$$\{ t.Nombre \mid t \in Socios \wedge$$

$$\wedge (\exists r) ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc)) \}$$

- Listar los nombres de los socios que tienen al menos un libro en préstamo y el nombre del libro

Ejemplos

- Listar los nombres de los socios que tienen al menos un libro en préstamo

$$\{ t.Nombre \mid t \in Socios \wedge$$

$$\wedge (\exists r) ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc)) \}$$

- Listar los nombres de los socios que tienen al menos un libro en préstamo y el nombre del libro

$$\{ t.Nombre, d.Nombre \mid t \in Socios \wedge d \in Libros \wedge$$

$$\wedge (\exists r) ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc) \wedge \wedge (r.Cód\_Lib = d.Cód\_Lib)) \}$$

¿Y cómo distingo los dos nombres?

Ejemplos

- Listar los nombres de los socios que tienen al menos un libro en préstamo y el nombre del libro

$$\{ t.Nombre, d.Nombre \mid t \in Socios \wedge d \in Libros \wedge$$

$$\wedge (\exists r) ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc) \wedge$$
$$\wedge (r.Cód\_Lib = d.Cód\_Lib)) \}$$

¿Y cómo distingo los dos nombres?

  - Se puede considerar que no hay problema con nombres, como en SQL con el AS
  - Se puede definir explícitamente la relación Resultado:
    - r ∈ Resultado(Nom_Soc, Nom_Lib), r = { … }

Ejemplos

- Listar los datos de los socios que tienen al menos un libro en préstamo así como el código del libro y la fecha del préstamo

Ejemplos

- Listar los datos de los socios que tienen al menos un libro en préstamo así como el código del libro y la fecha del préstamo

Resultado(Cód_Soc, Nombre, Apellido, Cód_Lib, Fecha)

$\{t \mid t \in Resultado\}$

Ejemplos

- Listar los datos de los socios que tienen al menos un libro en préstamo así como el código del libro y la fecha del préstamo

$$Resultado(Cód\_Soc, Nombre, Apellido, Cód\_Lib, Fecha)$$

$$\{t \mid t \in Resultado\}$$

$$\{t \mid (\exists r)\ ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc))$$

$$\wedge (\exists s)\ ((s \in Socios) \wedge (s.Cód\_Soc = t.Cód\_Soc))\}$$

Ejemplos

- Listar los datos de los socios que tienen al menos un libro en préstamo así como el código del libro y la fecha del préstamo

  Resultado(Cód_Soc, Nombre, Apellido, Cód_Lib, Fecha)

  $\{t \mid t \in Resultado\}$

  $\{t \mid (\exists r)\ ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc))$

  $\wedge (\exists s)\ ((s \in Socios) \wedge (s.Cód\_Soc = t.Cód\_Soc))\}$

  - ¿Estaría correcto así? Sólo se afirman cosas de *t.Cód_Soc ...*

Ejemplos

- Listar los datos de los socios que tienen al menos un libro en préstamo así como el código del libro y la fecha del préstamo

  Resultado(Cód_Soc, Nombre, Apellido, Cód_Lib, Fecha)

  $$\{t \mid t \in Resultado\}$$

  $$\{t \mid (\exists r)\ ((r \in Prestamos) \wedge (r.Cód\_Soc = t.Cód\_Soc)) \wedge (\exists s)\ ((s \in Socios) \wedge (s.Cód\_Soc = t.Cód\_Soc))\}$$

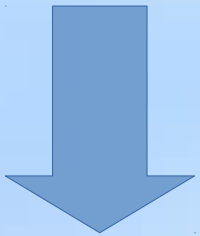
  - ¿Estaría correcto así? Sólo se afirman cosas de *t.Cód_Soc …*
    - Sí, porque obliga a instanciar correctamente los demás atributos (no hay nombre de atributos repetidos)
    - Pero no está de más igualar cada atributo ...

Ejemplos

- Resultado donde toma valores la variable *t*

Cód_Soc	Nombre	Apellido
1	Eric	Manrique
2	Mar	García

Cód_Lib	Cód_Soc	Fecha
1	1	19/3/2010
2	1	9/10/2010
3	1	18/12/2010
1	2	5/1/2017



Cód_Soc	Nombre	Apellido	Cód_Lib	Fecha
1	Eric	Manrique	1	19/3/2010
1	Eric	Manrique	2	9/10/2010
1	Eric	Manrique	3	18/12/2010
2	Mar	García	1	5/1/2017

Ejemplos

- Consultando tuplas de Libros y Socios sí tendría *Libros.Nombre* y *Socios.Nombre*
  - Habría que igualar cada atributo del resultado
- Ej: Nombre de los autores que sólo han publicado con el editor con código 8

Ejemplos

- Consultando tuplas de Libros y Socios sí tendría *Libros.Nombre* y *Socios.Nombre*
  - Habría que igualar cada atributo del resultado
- Ej: Nombre de los autores que sólo han publicado con el editor con código 8

$$t \in Resultado(Nom\_Aut)$$

$$\{t.Nom\_Aut \mid (\exists v)\ ((v \in Autores) \wedge (v.Nombre = t.Nom\_Aut)$$

$$\wedge (\forall r)\ ((r \in Libro) \wedge (r.Cód\_Edit=8) \wedge (r.Cód\_Aut = t.Cód\_Aut))$$

$$\wedge (\exists s)\ ((s \in Autores) \wedge (s.Cód\_Aut = t.Cód\_Aut) \wedge (s.Nombre = t.nombre))\}$$

Transformación de cuantificadores

- Toda fórmula que tenga un cuantificador (universal o existencial) se puede convertir en otra equivalente usando el otro cuantificador:
  - $(\forall t)(F) \equiv \text{not } (\exists t)(\text{not } F)$
  - $(\exists t)(F) \equiv \text{not } (\forall t)(\text{not } F)$
  - $(\forall t)(F_i \wedge F_j) \equiv \text{not } (\exists t)((\text{not } F_i) \vee (\text{not } F_j))$
  - $(\forall t)(F_i \vee F_j) \equiv \text{not } (\exists t)((\text{not } F_i) \wedge (\text{not } F_j))$
  - $(\exists t)(F_i \wedge F_j) \equiv \text{not } (\forall t)((\text{not } F_i) \vee (\text{not } F_j))$
  - $(\exists t)(F_i \vee F_j) \equiv \text{not } (\forall t)((\text{not } F_i) \wedge (\text{not } F_j))$

Relación entre cuantificadores

- También se cumplen las implicaciones:
  - $(\forall t)(F) \rightarrow (\exists t)(F)$
  - not $(\exists t)(F) \rightarrow$ not $(\forall t)(F)$
- ¿Para qué pueden servir estas relaciones y transformaciones?

Relación entre cuantificadores

- También se cumplen las implicaciones:
  - $(\forall t)(F) \rightarrow (\exists t)(F)$
  - not $(\exists t)(F) \rightarrow$ not $(\forall t)(F)$
- ¿Para qué pueden servir estas relaciones y transformaciones?
  - Porque no todos los lenguajes de los SGBD tienen que ofrecer los dos cuantificadores necesariamente
    - ¿SQL los ofrece?

Relación entre cuantificadores

- También se cumplen las implicaciones:
  - $(\forall t)(F) \rightarrow (\exists t)(F)$
  - not $(\exists t)(F) \rightarrow$ not $(\forall t)(F)$
- ¿Para qué pueden servir estas relaciones y transformaciones?
  - Porque no todos los lenguajes de los SGBD tienen que ofrecer los dos cuantificadores necesariamente
    - ¿SQL los ofrece? No, sólo el EXISTS
  - Además, puede ser que a veces uno sea más eficiente que otro, …

Fórmulas sanas

- Al escribir fórmulas en cálculo hay que ser cuidadoso de usar fórmulas con sentido
- Las fórmulas se clasifican en:
  - Fórmulas sanas (*safe*) aquellas que garantizan que su resultado es un conjunto finito de elementos
  - Fórmulas insanas (*unsafe*) son aquellas que no

Fórmulas sanas

- Ejemplo de fórmula insana:

$$\{t \mid \neg\, Libro(t)\}$$

  - Su resultado son toda tupla *t* que no esté en la relación Libros (conjunto infinito y “sale de rango”)
- Concepto de *dominio de una expresión*: el conjunto de valores constantes que:
  - Aparecen en la expresión
  - Existen en cualquier atributo de cualquier tupla que referencie la expresión

Fórmulas sanas

- Tenemos garantía de que una expresión es sana si todo valor que da su resultado está en el dominio de una expresión
- Puede ser necesario aplicar las las transformaciones entre cuantificadores para evitar fórmulas insanas

Fórmulas sanas

- El álgebra relacional no presenta estos problemas pues los operadores algebraicos son finitos y los resultados de estas operaciones también
  - Las expresiones algebraicas son siempre sanas
- Cualquier expresión de cálculo equivalente a otra de álgebra será una fórmula sana
- Se denomina *cálculo restringido de tupla* o *dominio* a todas las expresiones de cálculo que tienen alguna expresión algebraica equivalente
- Cualquier expresión del álgebra se puede expresar en cálculo restringido por lo que ambos lenguajes son equipotentes

Cálculo relacional de dominios

- El cálculo relacional de dominios tiene igual potencia que el de tuplas
- Se desarrolló a raíz del lenguaje QBE (Query-By-Example)
  - Primer lenguaje visual para hacer consultas
  - Desarrollado por IBM a la vez que SQL en los 70s (menos éxito)
  - Retomado por … ¿?

Cálculo relacional de dominios

- El cálculo relacional de dominios tiene igual potencia que el de tuplas
- Se desarrolló a raíz del lenguaje QBE (Query-By-Example)
  - Primer lenguaje visual para hacer consultas
  - Desarrollado por IBM a la vez que SQL en los 70s (menos éxito)
  - Retomado por Microsoft para Access … y Excel :_(

Cálculo relacional de dominios

- La principal diferencia respecto al cálculo de tuplas es que cada variable no almacena/instancia tuplas, sino valores de los dominios de los atributos
- Las expresiones son de la forma:

$$< x_1 , x_2 , \ldots , x_n >| \text{COND}(< x_1 , \ldots, x_n , \ldots, x_m>)$$

donde $x_1 , \ldots , x_n$ representan las variables de dominios y COND representa una fórmula compuesta de átomos

Cálculo relacional de dominios

- La COND se construye un poco distinta al cálculo de tuplas:
  - Es necesario indicar una variable por cada atributo de toda relación $R(x_1 , x_2 , ... , x_n)$
    - Se suelen eliminar las comas: $R(x_1\ x_2\ ...\ x_n)$
    - No se acepta: $t_i \in R$
  - x_i *op* x_j, donde *op* está en $\{=, >, \geq, <, \leq, \neq\}$
  - x_i *op const* ó *const op* x_i (siendo *const* una constante)

Cálculo relacional de dominios

- Listar el nombre de los socios que se apellidan Torvalds
  - Recordamos que Socio(Cód_Soc, Nombre, Apellido)

$$\{n \mid (\exists c)\,(\exists a)\,(Socio(cna) \wedge a = \text{“Torvalds”})\}$$

Cálculo relacional de dominios

- Listar el nombre de los socios que se apellidan Torvalds
  - Recordamos que Socio(Cód_Soc, Nombre, Apellido)

$$\{n \mid (\exists c)\ (\exists a)\ (Socio(cna) \wedge a = \text{“Torvalds”})\}$$

- Ó en una notación más libre (Access)

$$\{n \mid Socio(c,n,\text{“Torvalds”})\}$$

Cálculo relacional de dominios

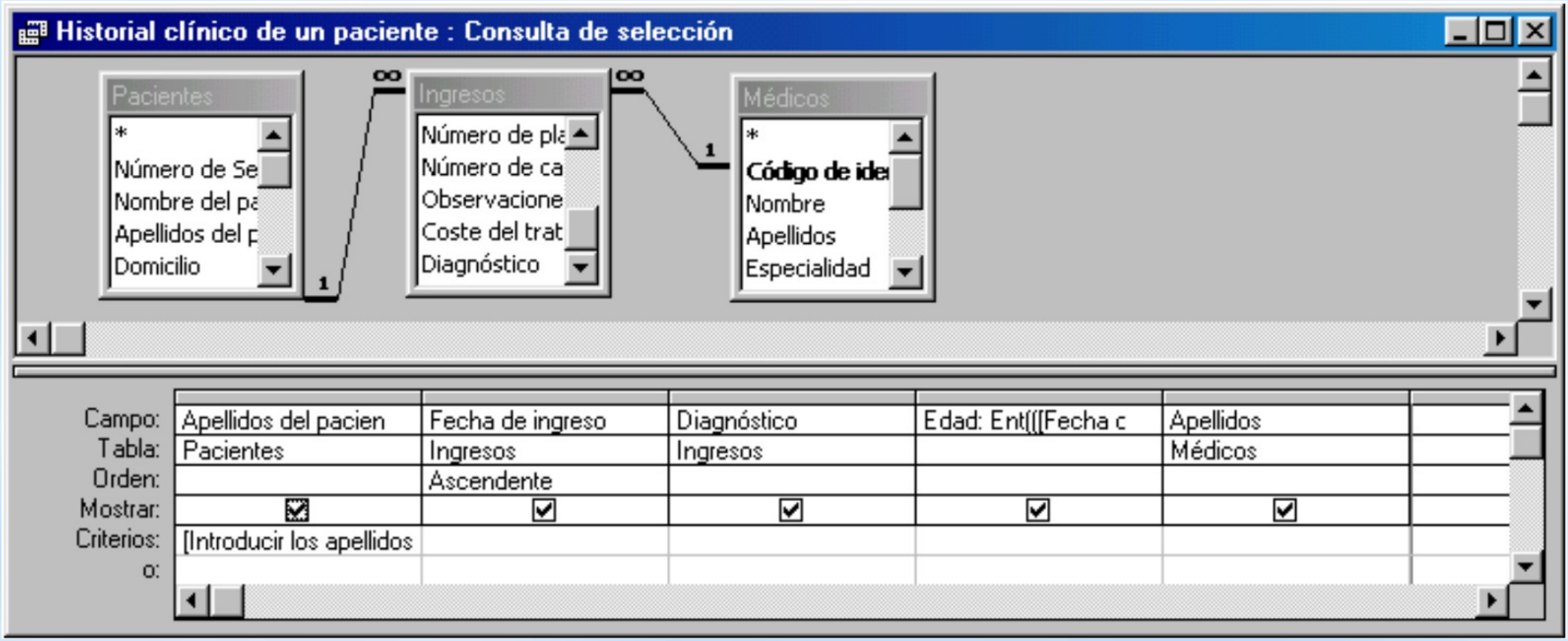
- Listar el nombre y apellidos de los socios que tienen un préstamos desde antes de enero
  - Recordamos que Socio(Cód_Soc, Nombre, Apellido)
  - Recordamos que Préstamo(Cód_Lib, Cód_Soc, Fecha)

$$\{na \mid (\exists c)(\exists l)(\exists s)(\exists f)\ (Socio(cna) \wedge Prestamo(lsf) \wedge f < (1/1/2017) \wedge c = s)\}$$

- Ó en una notación más libre (Access)

$$\{na \mid Socio(c,n,a) \wedge Prestamo(l,c,f) \wedge f < (1/1/2017)\}$$

QBE en Access



Cálculo relacional de dominios

- Hacer los ejemplos de cálculo de tuplas con el cálculo de dominios

Cálculo relacional

- ¿Cómo se implementarían los operadores del álgebra con cálculo?
  - Proyección
  - Selección
  - Producto cartesiano
  - Producto natural
  - Unión
  - ...

Poder expresivo del álgebra y del cálculo

- El álgebra y el cálculo (tanto de tuplas como de dominios) tiene el mismo poder de expresión
  - Se denominan lenguajes “relacionalmente completos”
  - Existen otros lenguajes relacionalmente completos
- Los lenguajes de los SGBD comerciales suelen ser relacionalmente completos, pero contienen operadores adicionales:
  - Funciones de agregación, agrupación y ordenación
  - Repeticiones en resultados, uso de resultados como escalares, …

Referencias

- Apuntes Esther Gadeschi
- Libro Elmasri, 3ª ed.
- https://en.wikipedia.org/wiki/Relational_algebra#Outer_joins
- https://en.wikipedia.org/wiki/Mathematical_operators_and_symbols_in_Unicode
- SQL as a Data Manipulation Language http://www-inf.it-sudparis.eu/COURS/BD/PRIVATE/EM2ENG/SLIDES/4SQL-DMLEM2ENG.pdf
- Fundamentos de bases de datos. Korth , Silberschatz https://www.dsi.fceia.unr.edu.ar/downloads/base_de_datos/CalculoRelacional.pdf
- Tema 3. Lenguajes QBE y SQL

Gracias por la atención

¿Preguntas?