

Temario

- ♦ Introducción y fundamentos
- ♦ Introducción a SQL
- ♦ Modelo Entidad / Relación
- ♦ Modelo relacional
- ♦ Diseño relacional: formas normales

- ♦ Consultas

Elmasri cap. 6

- Cálculo relacional
 - Álgebra relacional
- ♦ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B, B* y B+

Cálculo y álgebra relacional

- ♦ ¿Qué son?
 - Dos formalismos lógico-matemáticos para escribir consultas
 - Hasta cierto punto equivalentes a SQL pero permitiendo asegurar la consistencia matemática
 - Inicialmente pueden resultar más complejos que SQL, pero llegados a un punto de soltura, ayudan a despejar dudas mejor que en SQL (pues lógica & matemáticas son más universales que SQL, y además en el fondo se escriben más rápido)
 - Cálculo y álgebra preceden históricamente a SQL
- ♦ Nuestros objetivos en esta parte del curso
 - Escribir consultas (descritas en “castellano”) en cálculo y álgebra
 - Dar el resultado de una consulta en cálculo o álgebra
 - Traducir consultas entre cálculo, álgebra y SQL
 - Entender diferencias de optimización en diferentes formas de formular una misma consulta

Cálculo y álgebra relacional (cont)

- ♦ Formalismos para expresar operaciones de recuperación sobre una BD en modelo relacional
 - **Cálculo es declarativo.** En una expresión de CR no existe el orden de las operaciones para recuperar los resultados de la consulta: la expresión especifica la información que el resultado debería tener.
 - **Álgebra es procedural.** Especifica el conjunto de operaciones en el orden adecuado para recuperar los resultados de la consulta.
- ♦ Cálculo y álgebra son formalismos distintos pero lógicamente equivalentes
 - Toda expresión de cálculo se puede expresar en álgebra y viceversa.
 - Es decir, permiten expresar las mismas consultas.
- ♦ Un lenguaje de consulta es completo si permite expresar cualquier consulta del cálculo relacional.
 - Generalmente los SGBDs proporcionan un lenguaje completo con extensiones. 3

Cálculo y álgebra relacional (cont)

- ♦ Utilidad del cálculo relacional
 - Es más adecuado para establecer y verificar propiedades formales, la consistencia de los modelos relacionales y sus formalismos
 - Es útil para verificar detenidamente la corrección de aspectos complejos o delicados en ciertas consultas que lo precisen
 - La creación original del modelo relacional se fundamenta en el cálculo relacional. Interesa entenderlo para una comprensión más profunda del modelo relacional y el fundamento de la tecnología de bases de datos.
- ♦ Utilidad del álgebra relacional
 - Se utiliza con fines más prácticos; es más manejable que SQL para diseñar consultas complejas
 - Los motores de SQL basan su representación interna de las consultas en álgebra relacional (SQL se “parsea” a una estructura interna de álgebra)

Temario

- ♦ Introducción y fundamentos
- ♦ Introducción a SQL
- ♦ Modelo Entidad / Relación
- ♦ Modelo relacional
- ♦ Diseño relacional: formas normales
- ♦ Consultas
 - Cálculo relacional
 - Álgebra relacional
- ♦ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B
 - Hashing

Cálculo relacional

- ♦ Subconjunto del cálculo de predicados de primer orden (rama de la lógica matemática)
- ♦ Una consulta básica tiene la forma $\{ t \mid \textit{cond}(t) \}$, donde...
 - t representa una variable de tupla
 - \textit{cond} es una expresión condicional
 - La expresión representa (literalmente) un conjunto de tuplas que cumplen la condición
- ♦ El resultado es el conjunto de todas las tuplas que satisfacen la condición $\textit{cond}(t)$
- ♦ Vamos a ver la forma general...

Forma general de una consulta

{ variables | condición }

Variables de una consulta

- ◆ Representan tuplas de esquemas relacionales
- ◆ Pueden ser una o más

$$\{ t_1, t_2, \dots, t_n \mid \text{condición} \}$$

- ◆ Pueden indicarse atributos específicos de las variables
(y mezclar variables con y sin atributos...)

$$\{ t_1.A, t_1.B, t_1.C, t_2, \dots, t_n \mid \text{condición} \} \quad /* A, B, C \text{ atributos de } t_1 */$$

Condición de una consulta

- ♦ Es una expresión (fórmula) de cálculo de predicados de primer orden
- ♦ Puede ser:
 1. Una fórmula atómica...
 - a) $R(t)$, donde R es un esquema relacional, y t es una variable de tupla
 - b) $t_1 . A \text{ op } t_2 . B$ o bien $t_1 . A \text{ op } c$ donde...
 - t_1 y t_2 son variables de tupla
 - A y B son atributos, c es una constante
 - op es un operador de comparación: $=, <, \leq, >, \geq, \neq$
 2. Operadores *and*, *or*, *not* aplicados a fórmulas
 3. $\forall t, \exists t$ aplicados a fórmulas

“Resultado” de una consulta

- ♦ El “resultado” de una consulta en cálculo relacional es un conjunto de tuplas
- ♦ Cuyos atributos son la unión de los atributos de todas las variables de tupla, más los atributos indicados directamente

Por ejemplo, dados los esquemas:

Vuelo (número, origen, destino, hora)

Aeropuerto (código, ciudad)

Y la consulta:

$\{ v.\text{número}, a \mid \text{Vuelo}(v) \text{ and } \text{Aeropuerto}(a) \text{ and } v.\text{origen} = a.\text{código} \}$

Los atributos de las tuplas de la consulta son:

$(\underbrace{\text{número}}_{v . \text{número}}, \underbrace{\text{código, ciudad}}_a)$

- ♦ La condición filtra qué tuplas exactamente se incluyen en ese conjunto

Correspondencia con SQL

- ◆ Variables de consulta Los términos que siguen a SELECT (con DISTINCT)
 - Salvo que en SELECT no hay variables de tuplas, sólo campos
- ◆ Condiciones de tipo $R(t)$ Equivale a 'R as t' en la cláusula FROM
 - Pero no se concreta si es JOIN, producto cartesiano, etc.
- ◆ Condiciones con \exists y \forall Se pueden expresar con EXISTS, SOME y ALL
 - La mayoría de las veces \exists se traduce simplemente en un esquema en FROM, que no aparece en SELECT
 - Si es difícil expresar un \forall , se puede jugar con \exists y negación
- ◆ Otras condiciones Aparecen tras WHERE, ON, etc.
 - Pueden volverse implícitas en un NATURAL JOIN
- ◆ En principio no es común en cálculo relacional (pero se puede):
 - Operaciones de conjuntos: unión, intersección, diferencia, pertenencia
 - Operaciones de agregación: COUNT, AVG, MAX, etc.
 - Vistas y consultas anidadas
- ◆ ORDER BY no tiene sentido ya que el orden de tuplas no existe en el modelo relacional

Ejemplos...

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos entre Charles de Gaulle y Heathrow

- **SELECT** * **FROM** vuelo v **WHERE** v.origen='CDG' **AND** v.destino='LHR'
- {v | VUELO(v) and v.origen='CDG' and v.destino='LHR'}

Hora de salida de los vuelos entre Charles de Gaulle y Heathrow

- **SELECT** v.salida **FROM** vuelo v **WHERE** v.origen='CDG' **AND** v.destino='LHR'
- {v.salida | VUELO(v) and v.origen='CDG' and v.destino='LHR'}

Ejemplos...

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos que cubren el trayecto Charles de Gaulle – Heathrow en cualquier sentido

```
SELECT
    *
FROM
    vuelo v
WHERE
    (v.origen='CDG' AND v.destino='LHR') OR (v.origen='LHR' AND v.destino='CDG'))
```

```
{v | VUELO(v) and ((v.origen='CDG' and v.destino='LHR') OR (v.origen='LHR' and
v.destino='CDG'))}
```

Ejemplos...

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Nombre, fecha y destino de viaje de todos los pasajeros que vuelan desde Madrid

select

p.nombre, r.fecha, a2.ciudad

from

pasajero p **join** reserva r on p.dni=r.dni

join vuelo v on r.numero=v.numero

join aeropuerto a1 on a1.codigo=v.origen

join aeropuerto a2 on a2.codigo=v.destino

where

a1.ciudad='Madrid'

{p.nombre, r.fecha, v.destino | Vuelo(v) and Aeropuerto(a1) and Aeropuerto(a2) and Pasajero(p) and Reserva(r) and v.origen=a1.código and v.destino=a2.código and p.dni=r.dni and r.numero=v.numero and a1.ciudad='Madrid'}

Ejemplos...

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos que no tienen ninguna reserva

SELECT

v.numero

FROM

vuelo v

WHERE

NOT EXISTS (SELECT * FROM reserva r WHERE r.numero=v.numero)

$\{v.numero \mid \text{Vuelo}(v) \text{ and } (\neg \exists r)(\text{Reserva}(r) \text{ and } r.numero=v.numero)\}$

Temario

- ♦ Introducción y fundamentos
- ♦ Introducción a SQL
- ♦ Modelo Entidad / Relación
- ♦ Modelo relacional
- ♦ Diseño relacional: formas normales
- ♦ Consultas
 - Cálculo relacional
 - Álgebra relacional
- ♦ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B
 - Hashing

Álgebra relacional

- ♦ Expresa consultas en forma de operaciones a realizar para obtener las tuplas deseadas
 - A diferencia del cálculo relacional, en el que se expresan las condiciones que deben cumplir las tuplas que se desean obtener
 - Por este motivo el álgebra se considera procedural, y el cálculo declarativo
- ♦ El resultado de una consulta en álgebra relacional es un conjunto de tuplas
 - Igual que en cálculo
- ♦ Tres tipos de operación
 - Unarias*
 - Específicas de BD: selección, proyección, renombrado, join y sus variantes
 - De conjuntos: unión, intersección, diferencia, producto cartesiano
 - Adicionales: proyección generalizada, join externo, agregación...
 - Binarias*

Operaciones propias de BDs

- ◆ Selección
- ◆ Proyección
- ◆ Renombrado
- ◆ Joins
- ◆ División (la omitiremos, se puede derivar de otras operaciones)

Select: $\sigma_{condición}(R)$

- ♦ Operación “horizontal”: filtra tuplas de una relación
 - Las que cumplen una condición
- ♦ El operando R puede ser un esquema relacional, o una expresión de álgebra
 - Por tanto un conjunto de tuplas en cualquier caso
 - Esto es así para los operándos de todas las operaciones (lo sobreentendemos en adelante)
- ♦ La condición puede ser:
 - Una comparación simple $A \text{ op } B$ o bien $A \text{ op } c$ donde:
 - A y B son atributos, c es una constante
 - op es un operador de comparación: $=, <, \leq, >, \geq, \neq$ (se pueden ampliar)
 - Operadores *and*, *or*, *not*, aplicados a otras condiciones
- ♦ Es decir, las condiciones son como las del cálculo relacional, salvo que...

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos entre Charles de Gaulle y Heathrow

$\sigma_{\text{Origen} = \text{'CDG'} \text{ and } \text{Destino} = \text{'LHR'}} \text{ (VUELO)}$

$\rightarrow \{ (903, \text{'CDG'}, \text{'LHR'}, \text{'14:40'}), (447, \text{'CDG'}, \text{'LHR'}, \text{'17:00'}) \}$

Reservas por menos de 200€

$\sigma_{\text{Precio} < 200} \text{ (RESERVA)}$

$\rightarrow \{ (123, 345, \text{'20-12-10'}, 170), (456, 345, \text{'03-11-10'}, 190) \}$

Algunas propiedades de σ

- ♦ Los atributos de $\sigma_c(R)$ y los de R son los mismos
- ♦ $|\sigma_c(R)| \leq |R|$
- ♦ σ es conmutativa: $\sigma_c(\sigma_d(R)) = \sigma_d(\sigma_c(R)) = \sigma_{c \text{ and } d}(R)$

Proyección: $\pi_{\text{atributos}}(R)$

- ♦ Operación “vertical”: se filtran atributos de una relación
- ♦ Los atributos deben ser un subconjunto de los atributos de R
- ♦ Si el conjunto de atributos de la proyección no contiene ninguna clave, pueden repetirse tuplas
 - Se eliminan las duplicaciones en tal caso (puesto que se trata de un conjunto)
 - Como ya sabemos, los SGBD no necesariamente aplican esto de forma estricta

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Ciudades con aeropuerto

$\pi_{\text{Ciudad}}(\text{AEROPUERTO})$

$\rightarrow \{ ('Madrid'), ('Londres'), ('París') \}$

Aeropuertos con conexión entre sí

$\pi_{\text{Origen, Destino}}(\text{VUELO})$

$\rightarrow \{ ('MAD', 'CDG'), ('MAD', 'ORY'), ('LHR', 'CDG'), ('CDG', 'LHR') \}$

Número de los vuelos con salida desde Madrid

$\pi_{\text{Numero}}(\sigma_{\text{Origen} = 'MAD'}(\text{VUELO}))$

$\rightarrow \{ (345), (321) \}$

Algunas propiedades de π

- ♦ $\pi_X(\pi_Y(R)) = \pi_X(R)$
 - Siempre y cuando $X \subset Y$, pues en otro caso la expresión es incorrecta
- ♦ $|\pi_X(R)| \leq |R|$ (las duplicadas se eliminan a diferencia de en un SGBD)
- ♦ Si X es una superclave de $R \Rightarrow |\pi_X(R)| = |R|$
- ♦ π no es conmutativa
 - $\pi_X(\pi_Y(R))$ y $\pi_Y(\pi_X(R))$ sólo serían ambas correctas si $X = Y$
 - Lo cual no tendría mucho sentido pues π_X y π_Y serían redundantes

Renombrado: ρ y \leftarrow

1. Renombrado

De atributos: $\rho_{B_1, B_2, \dots, B_n}(R)$ renombra los atributos de R a B_1, \dots, B_n

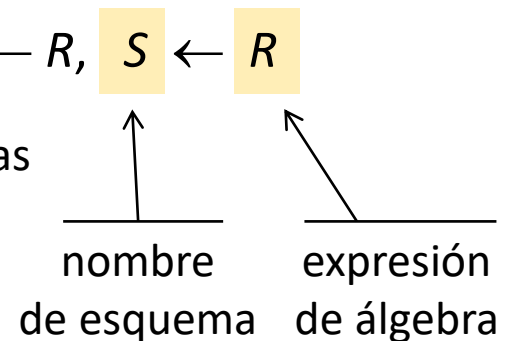
De algunos atributos: $\rho_{A_1/B_1, \dots, A_k/B_k}(R)$

De relación $\rho_S(R)$, de relación y atributos: $\rho_{S(B_1, \dots, B_n)}(R)$

- Útil para distinguir atributos que tienen el mismo nombre en las condiciones de joins y σ

2. Renombrado como asignación: $S(B_1, \dots, B_n) \leftarrow R$,

- Útil para descomponer expresiones complejas



Ejemplos

Cuando veamos join y otros operadores...

Join \bowtie

Pero veamos antes el producto cartesiano...

(del grupo de las operaciones de conjuntos)

Producto cartesiano $R \times S$

- ♦ También llamado “cross join”
- ♦ Es la misma operación que en álgebra de conjuntos
- ♦ Pero en vez de formar pares de tuplas $((a_1, \dots, a_n), (b_1, \dots, b_m))$,
se concatenan los atributos de las tuplas $(a_1, \dots, a_n, b_1, \dots, b_m)$

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

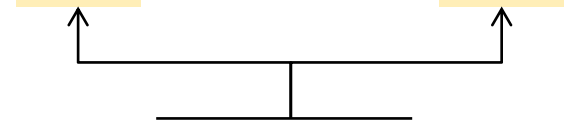
RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos que salen de París

$\sigma_{\text{Origen} = \text{Codigo and Ciudad} = \text{'París'}} (\text{VUELO} \times \text{AEROPUERTO})$

→ { (903, 'CDG', 'LHR', '14:40', 'CDG', 'París'),
(447, 'CDG', 'LHR', '17:00', 'CDG', 'París') }


Redundancia:
desaparecerá en natural join

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

PASAJERO × RESERVA

→ { (123, María, 789, 165, 07-01-11, 210),
(123, María, 123, 345, 20-12-10, 170),
(123, María, 789, 321, 15-12-10, 250),
(123, María, 456, 345, 03-11-10, 190),
(456, Pedro, 789, 165, 07-01-11, 210),
(456, Pedro, 123, 345, 20-12-10, 170),
(456, Pedro, 789, 321, 15-12-10, 250),
(456, Pedro, 456, 345, 03-11-10, 190),
(789, Isabel, 789, 165, 07-01-11, 210),
(789, Isabel, 123, 345, 20-12-10, 170),
(789, Isabel, 789, 321, 15-12-10, 250),
(789, Isabel, 456, 345, 03-11-10, 190) }

← En general no tiene mucho sentido (y el coste es alto!): lo lógico sería conectar las dos tablas con alguna condición → select, join...
SELECT * FROM pasajero CROSS JOIN reserva

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

$\sigma_{Dni} = DniPasajero (PASAJERO \times \rho_{Dni / DniPasajero} (RESERVA))$

→ { (123, María, 123, 345, 20-12-10, 170),
 (456, Pedro, 456, 345, 03-11-10, 190),
 (789, Isabel, 789, 165, 07-01-11, 210),
 (789, Isabel, 789, 321, 15-12-10, 250) }

Algunas propiedades de \times

Dados $R (A_1, \dots, A_n)$ y $S (B_1, \dots, B_m)$

- ♦ $|R \times S| = |R| |S|$
- ♦ $R \times S$ tiene $n + m$ atributos: $(R \times S) (A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$
- ♦ \times es conmutativo y asociativo

Join: $R \bowtie_{condición} S$

- ♦ Ahora sí... $R \bowtie_{condición} S$ es equivalente a $\sigma_{condición}(R \times S)$
- ♦ Tipos particulares de join: equijoin y natural join
- ♦ Equijoin: la condición es un *and* de comparaciones de igualdad entre atributos de R y S
- ♦ Natural join: equijoin donde sólo se incluye un atributo por cada par emparejado
 - Notación $R \bowtie S$ sin indicar condición: la condición es de igualdad entre todos los atributos comunes a R y S

- O bien se puede indicar la lista de atributos a emparejar:

$$R \bowtie_{(A_1, \dots, A_n), (B_1, \dots, B_n)} S = \pi_{X, A_1, \dots, A_n, Y} (R \bowtie_{A_1 = B_1, \dots, A_n = B_n} S)$$

Donde X son los atributos de R menos A_i , e Y son los de S menos B_i

- Típicamente los atributos emparejados son clave externa / clave primaria

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

PASAJERO ⋈ RESERVA

→ { (123, María, 345, 20-12-10, 170),
(456, Pedro, 345, 03-11-10, 190),
(789, Isabel, 165, 07-01-11, 210),
(789, Isabel, 321, 15-12-10, 250) }

SELECT * FROM pasajero NATURAL JOIN reserva

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Vuelos entre Madrid y París

$R \leftarrow \text{VUELO} \bowtie_{\text{Origen, Codigo}} \rho_{\text{Ciudad} / \text{CiudadOrigen}} (\text{AEROPUERTO})$

$S \leftarrow R \bowtie_{\text{Destino, Codigo}} \rho_{\text{Ciudad} / \text{CiudadDestino}} (\text{AEROPUERTO})$

$\sigma_{\text{CiudadOrigen} = \text{'Madrid'} \text{ and } \text{CiudadDestino} = \text{'París'}} (S)$

Nombre, fecha y destino de viaje de todos los pasajeros que vuelan desde Madrid

$R \leftarrow \text{VUELO} \bowtie_{\text{Origen} = \text{Codigo and Ciudad} = \text{'Madrid'}} \text{AEROPUERTO}$

$S \leftarrow R \bowtie_{\text{Destino, Codigo}} \rho_{\text{Ciudad} / \text{CiudadDestino}} (\text{AEROPUERTO})$

$\pi_{\text{Nombre, Fecha, CiudadDestino}} (S \bowtie \text{RESERVA} \bowtie \text{PASAJERO})$

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Nombre, fecha y destino de viaje de todos los pasajeros que vuelan desde Madrid

$$S_1 \leftarrow \text{PASAJERO} \bowtie \text{RESERVA}$$

$$S_2 \leftarrow \text{VUELO} \bowtie_{\text{Destino, Codigo}} \rho_{\text{Ciudad} / \text{CiudadDestino}} (\text{AEROPUERTO})$$

$$S_3 \leftarrow S_2 \bowtie_{\text{Origen, Codigo}} \rho_{\text{Ciudad} / \text{CiudadOrigen}} (\text{AEROPUERTO})$$

$$\pi_{\text{Nombre, Fecha, CiudadDestino}} (\sigma_{\text{CiudadOrigen} = \text{'Madrid'}} (S_1 \bowtie S_3))$$

Ejemplos

VUELO

Numero	Origen	Destino	Salida
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

AEROPUERTO

Codigo	Ciudad
MAD	Madrid
LGW	Londres
LHR	Londres
ORY	París
CDG	París

PASAJERO

Dni	Nombre
123	María
456	Pedro
789	Isabel

RESERVA

Dni	Numero	Fecha	Precio
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Nombre, fecha y destino de viaje de todos los pasajeros que vuelan desde Madrid

$S_1 := \text{SELECT } * \text{ FROM pasajero NATURAL JOIN reserva}$

$S_2 := \text{SELECT v.numero, a.ciudad c_origen, v.destino FROM vuelo v JOIN aeropuerto a ON v.origen=a.codigo}$

$S_3 := \text{SELECT numero, c_origen, a.ciudad as c_destino FROM (SELECT v.numero, a.ciudad c_origen, v.destino FROM vuelo v JOIN aeropuerto a ON v.origen=a.codigo) as s2 JOIN aeropuerto a ON s2.destino=a.codigo}$

$\text{SELECT s1.nombre, s1.fecha, s3.c_destino FROM (SELECT } * \text{ FROM pasajero NATURAL JOIN reserva) as s1 NATURAL JOIN (SELECT numero, c_origen, a.ciudad as c_destino FROM (SELECT v.numero, a.ciudad c_origen, v.destino FROM vuelo v JOIN aeropuerto a ON v.origen=a.codigo) as s2 JOIN aeropuerto a ON s2.destino=a.codigo) as s3 WHERE c_origen='Madrid'}$

Algunas propiedades de \bowtie

Dados $R (A_1, \dots, A_n)$ y $S (B_1, \dots, B_m)$

- ♦ $R \bowtie_c S$ tiene $n + m$ atributos : $(R \bowtie_c S) (A_1, \dots, A_n, B_1, \dots, B_m)$
- ♦ $|R \bowtie_c S| \leq |R| |S|$
 - La “selectividad” del join es la tasa $|R \bowtie_c S| / |R| |S|$
- ♦ \bowtie es asociativo y conmutativo (conmutando/asociando adecuadamente las condiciones del join)

Operaciones de conjuntos

- ◆ Unión
- ◆ Intersección
- ◆ Diferencia
- ◆ Producto cartesiano (ya visto)

Operaciones de conjuntos

$$R \cup S, R \cap S, R - S$$

- ♦ R y S deben tener el mismo nº y dominio de los atributos (“unión-compatible”)
 - Esto no es así con el producto cartesiano, que no lo precisa
- ♦ La definición es la misma que en álgebra de conjuntos:
 - **UNION:** $R \cup S$ es una relación que incluye todas las tuplas que están en R o están en S o en ambas R y S (No se incluyen tuplas duplicadas)
 - **INTERSECCIÓN:** $R \cap S$ es una relación que incluye todas las tuplas que están en R y S
 - **DIFERENCIA:** $R - S$ es una relación que incluye todas las tuplas que están en R pero no están en S

Operaciones de conjuntos: \cup , \cap

<u>dni</u>	Nombre
123	Susana
456	Pedro
789	Laura
246	Juan

$$|E \cup P| = 4$$



<u>dni</u>	Nombre
789	Laura

$$|E \cap P| = 1$$

Estudiante

<u>dni</u>	Nombre
123	Susana
456	Pedro
789	Laura

$$|E| = 3$$



Profesor

<u>dni</u>	Nombre
246	Juan
789	Laura

$$|P| = 2$$

Estudiante

<u>dni</u>	Nombre
123	Susana
456	Pedro
789	Laura

$$|E| = 3$$



Profesor

<u>dni</u>	Nombre
246	Juan
789	Laura

$$|P| = 2$$

Operaciones de conjuntos: MINUS (-)

<u>dni</u>	Nombre
123	Susana
456	Pedro

$$|E-P|=2$$

≠

<u>dni</u>	Nombre
246	Juan

$$|P-E|=1$$

Estudiante

<u>dni</u>	Nombre
123	Susana
456	Pedro
789	Laura

$$|E|=3$$

Profesor

<u>dni</u>	Nombre
246	Juan
789	Laura

$$|P|=2$$

Profesor

<u>dni</u>	Nombre
246	Juan
789	Laura

$$|P|=2$$

Estudiante

<u>dni</u>	Nombre
123	Susana
456	Pedro
789	Laura

$$|E|=3$$

Algunas propiedades de \cup y \cap

Dados $R(A_1, \dots, A_n)$ y $S(B_1, \dots, B_n)$

- ♦ $R \cup S$ y $R \cap S$ tienen n atributos
- ♦ $\max(|R|, |S|) \leq |R \cup S| \leq |R| + |S|$
- ♦ $|R \cap S| \leq \min(|R|, |S|)$
- ♦ \cup y \cap son conmutativos y asociativos
 - $R \cup S = S \cup R$ y $R \cap S = S \cap R$
 - $R \cup (S \cup T) = (R \cup S) \cup T$ y $R \cap (S \cap T) = (R \cap S) \cap T$
- ♦ La operación MINUS no es conmutativa
 - $R - S \neq S - R$
- ♦ La intersección se puede poner en términos de unión y diferencia
 - $R \cap S = R \cup S - (R - S) - (S - R)$

Algunas propiedades de \cup y \cap (cont)

Dados $R (A_1, \dots, A_n)$ y $S (B_1, \dots, B_n)$

- ♦ $\sigma_c(R) \cap \sigma_d(S) = \sigma_{c \text{ and } d}(R \cap S)$
- ♦ $\sigma_c(R) \cup \sigma_d(R) = \sigma_{c \text{ or } d}(R)$

Algunas propiedades globales más

- ♦ Las operaciones binarias (excepto la diferencia de conjuntos) se pueden generalizar a operaciones n-arias
 - De forma obvia por asociatividad de las operaciones binarias
- ♦ Las operaciones $\sigma, \pi, \cup, -, \times$ forman un conjunto completo de operaciones
 - Las demás se pueden derivar de ellas:

$$\cup - \rightarrow \cap$$

$$\sigma \times \rightarrow \bowtie_c$$

$$\pi \sigma \times \rightarrow \bowtie$$

Operaciones adicionales

- ♦ Son extensiones externas al álgebra relacional propiamente dicha
 - Se utilizan por motivos prácticos
- ♦ Proyección generalizada
 - Admite operaciones sobre los atributos: $\pi_{f_1(X_1), \dots, f_n(X_n)}(R)$
donde X_i son conjuntos de atributos de R , y f_i son Count, Sum, Avg, Max, ó Min
- ♦ Agrupación y agregación
 - $A_1, \dots, A_n \text{ G }_{f_1(X_1), \dots, f_n(X_n)}(R)$
donde A_i y B_i son atributos de R , y f_i son Count, Sum, Avg, Max, ó Min
- ♦ Join externo
 - Incluyen tuplas de uno u otro operando o ambos (left / right / full), las que no tienen tupla asociada en el otro conjunto
 - Se ponen NULLs en los atributos que corresponderían al otro esquema
- ♦ Limitación: el álgebra relacional no tiene iteración/recursión (tampoco SQL)
 - P.e. no es posible calcular la raíz de un árbol, distancias en una red social, etc.

Correspondencia con SQL

- ♦ $\pi_{\text{atributos}}(\sigma_{\text{condición}}(R))$ SELECT *atributos* FROM *R* WHERE *condición*
- ♦ $\rho_{A/C}(\pi_{A,B}(\sigma_{\text{condición}}(R)))$ SELECT A AS C, B FROM *R* WHERE *condición*
- ♦ $S \leftarrow (\pi_{\text{atributos}}(\sigma_{\text{condición}}(R)))$ CREATE VIEW S AS SELECT *atributos* FROM *R* WHERE *condición*
- ♦ $\pi_{\text{atributos}}(\sigma_{\text{condición}}(R \bowtie S))$ SELECT *atributos* FROM *R* NATURAL JOIN *S* WHERE *condición*
- ♦ $\pi_{\text{atributos}}(R \bowtie_{\text{condición}} S)$ SELECT *atributos* FROM *R* JOIN *S* ON *condición* // O bien: WHERE *condición*
- ♦ $\pi_{\text{atributos}}(\sigma_{\text{condición}}(R \times S))$ SELECT *atributos* FROM *R*, *S* WHERE *condición*
- ♦ $R \cup S, R \cap S, R - S$ *R* UNION *S*, *R* INTERSECT *S*, *R* EXCEPT *S*
- ♦ $\text{atributos G Count(A), Sum(B)... (R)}$ SELECT Count(A), Sum(B)... FROM *R* GROUP BY *atributos*

Para no hacerlo repetitivo omitimos aquí DISTINCT
(pero debe sobreentenderse!)

Optimización de consultas

- ♦ El coste de una consulta puede variar mucho según cómo se exprese

– Ejemplo: $\left. \begin{array}{l} \sigma_{\text{Origen} = \text{'LHR'}} (\sigma_{\text{Destino} = \text{'CDG'}} (\text{VUELO})) \\ \sigma_{\text{Destino} = \text{'CDG'}} (\sigma_{\text{Origen} = \text{'LHR'}} (\text{VUELO})) \end{array} \right\} \leftarrow \text{Cuál es más eficiente?}$

- ♦ Objetivos generales

- **Reducir el tamaño** promedio del resultado de las expresiones
- **Formar subexpresiones comunes** dentro de o entre consultas para ejecutarlas una sola vez

- ♦ Estrategias generales

- **Introducción de select** hacia subexpresiones más internas

El tamaño de un select es menor que el del conjunto al que se aplica;
cuanto más internamente se sitúe el select, antes tiene lugar esta reducción

Situar los select más restrictivos más al interior que otros menos selectivos

- **Evitar productos cartesianos** en las operaciones más internas;
es la operación que genera conjuntos más grandes

Es preferible un join $R \bowtie_c S$ que un producto cartesiano $\sigma_c(R \times S)$

- **Introducir proyecciones** hacia el interior para operar sólo con los atributos realmente necesarios; la proyección es poco costosa y puede eliminar tuplas duplicadas