Base de datos distribuidas

Prof. Alejandro Reyes Ortiz

“DESCOMPOSICIÓN DE CONSULTAS: Simplificar y Re-escritura”

Simplificación de la consulta

- El objetivo es eliminar la redundancia. La consulta en forma normal puede contener predicados redundantes.
- Una evaluación directa de la consulta con redundancia puede llevarnos a realizar trabajo duplicado (no necesario).
- La redundancia puede ser eliminada apoyandose de reglas de equivalencia y aplicando sucesivamente reglas de idempotencia

Simplificación de la consulta

- Reglas de equivalencias para operaciones lógicas

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $p_1 \wedge (p_2 \wedge p_3) \Leftrightarrow (p_1 \wedge p_2) \wedge p_3$
4. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
5. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
6. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$
7. $\neg(p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$
8. $\neg(p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$
9. $\neg(\neg p) \Leftrightarrow p$

- Reglas de idempotencia (eliminación de redundancia).

1. $p \wedge p \Leftrightarrow p$
2. $p \vee p \Leftrightarrow p$
3. $p \wedge true \Leftrightarrow p$
4. $p \vee false \Leftrightarrow p$
5. $p \wedge false \Leftrightarrow false$
6. $p \vee true \Leftrightarrow true$
7. $p \wedge \neg p \Leftrightarrow false$
8. $p \vee \neg p \Leftrightarrow true$
9. $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
10. $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

Simplificación de la consulta

- Ejemplo de una consulta en SQL

```
SELECT TITLE
FROM   EMP
WHERE  (NOT (TITLE = "Programmer")
AND    (TITLE = "Programmer"
OR     TITLE = "Elect. Eng.")
AND    NOT (TITLE = "Elect. Eng."))
OR     ENAME = "J. Doe"
```

Consulta simplificada

```
SELECT TITLE
FROM   EMP
WHERE  ENAME = "J. Doe"
```

Simplificación de la consulta

- Proceso de simplificación
  - P1 es TITLE="Programmer"
  - P2 es TITLE ="Elect. Eng"
  - P3 es ENAME= "J. Doe"

La clausula WHERE se puede representar como

$$(\neg p_1 \wedge (p_1 \vee p_2) \wedge \neg p_2) \vee p_3$$

Simplificación de la consulta

$$(\neg p_1 \wedge (p_1 \vee p_2) \wedge \neg p_2) \vee p_3$$

Paso 1: Aplicamos la regla 5 (equivalencias) para obtener la forma normal disyuntiva.

$$(\neg p_1 \wedge ((p_1 \wedge \neg p_2) \vee (p_2 \wedge \neg p_2))) \vee p_3$$

Paso 2: Aplicamos la regla 5 (equivalencias) para obtener la forma normal disyuntiva.

$$(\neg p_1 \wedge p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2 \wedge \neg p_2) \vee p_3$$

Paso 3: Aplicamos la regla 7 (idempotencia) para obtener la forma normal disyuntiva.

$$(false \wedge \neg p_2) \vee (\neg p_1 \wedge false) \vee p_3$$

Simplificación de la consulta

$$(false \wedge \neg p_2) \vee (\neg p_1 \wedge false) \vee p_3$$

Paso 4: Aplicamos la regla 5 (idempotencia) para obtener la forma normal disyuntiva.

$$false \vee false \vee p_3$$

Paso 5: Aplicamos la regla 4 (idempotencia) para obtener la forma normal disyuntiva.

$$p_3$$

```
SELECT TITLE
FROM   EMP
WHERE  ENAME = "J. Doe"
```

Re-escritura

- Su meta es generar un árbol del algebra relacional, denominado "Árbol de expresiones (operadores)"

- Árbol de expresiones: Es un árbol en el cual cada hoja es una relación almacenada en la base de datos y los nodos no-hojas son relaciones intermedias producidas por un operador de álgebra relacional.

Re-escritura

- Construcción del árbol
  - Se crea una hoja diferente para cada variable (correspondiente a una relación). En SQL, las hojas serán obtenidas de la cláusula FROM.
  - Se crea el nodo raíz como una operación de proyección involucrando los atributos resultados. Esto es encontrado en la clausula SELECT.
  - La cláusula WHERE es traducida como la secuencia apropiada de operaciones relacionales (selecciones, join, unión, etc.) partiendo de la raíz a las hojas.

σ

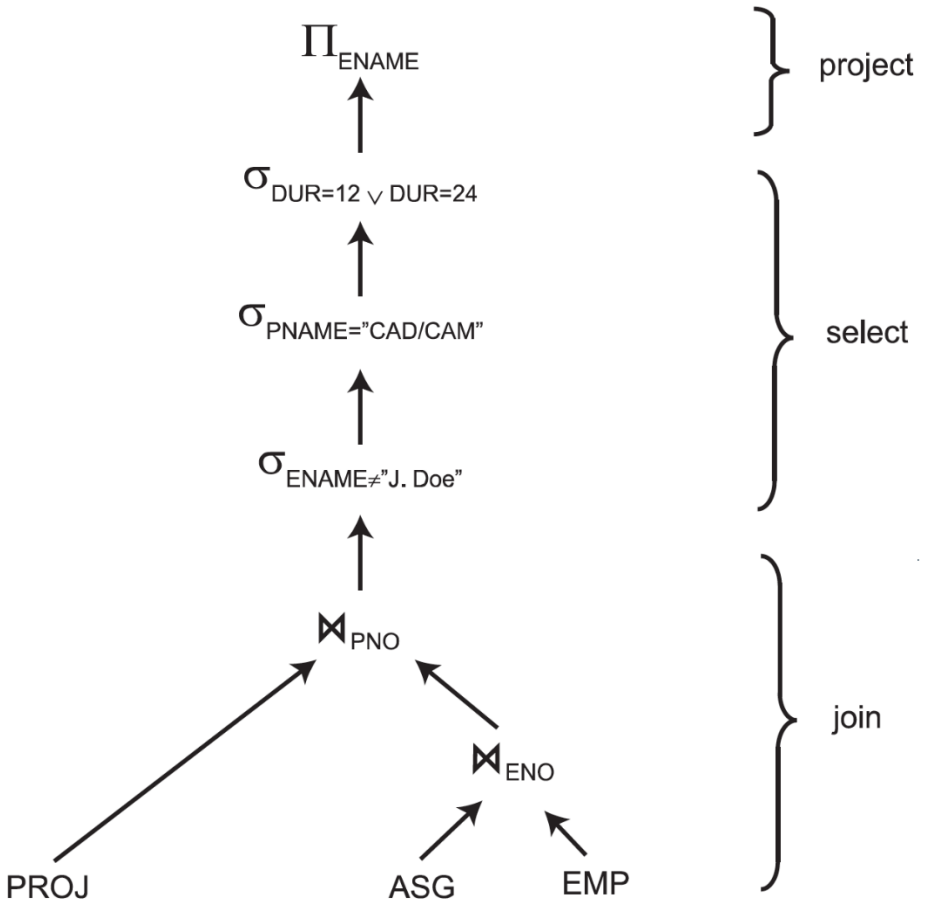
Re-escritura

"Encontrar los nombres de los empleados que no sean J. Doe y que trabajaron con el proyecto CAD/CAM por uno o dos años"

```
SELECT ENAME
FROM   PROJ, ASG, EMP
WHERE  ASG.ENO = EMP.ENO
AND    ASG.PNO = PROJ.PNO
AND    ENAME != "J. Doe"
AND    PROJ.PNAME = "CAD/CAM"
AND    (DUR = 12 OR DUR = 24)
```

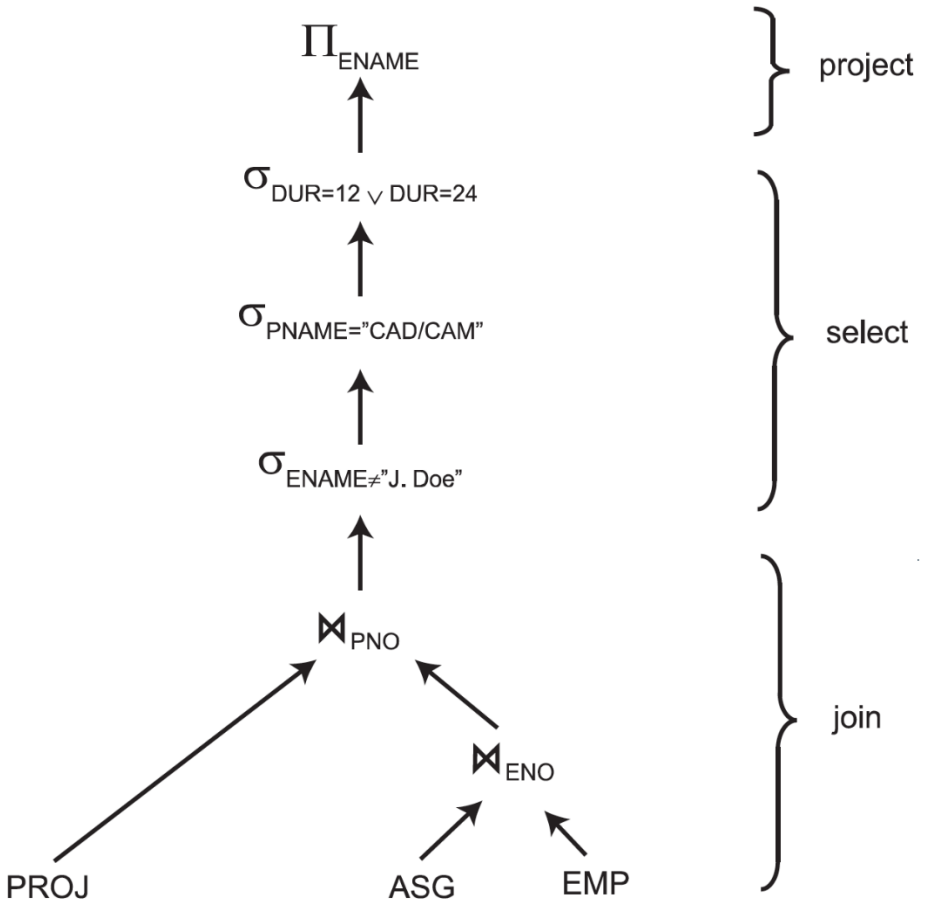
Re-escritura

```
SELECT ENAME
FROM   PROJ, ASG, EMP
WHERE  ASG.ENO = EMP.ENO
AND    ASG.PNO = PROJ.PNO
AND    ENAME != "J. Doe"
AND    PROJ.PNAME = "CAD/CAM"
AND    (DUR = 12 OR DUR = 24)
```



Re-escritura

```
SELECT ENAME
FROM   PROJ, ASG, EMP
WHERE  ASG.ENO = EMP.ENO
AND    ASG.PNO = PROJ.PNO
AND    ENAME != "J. Doe"
AND    PROJ.PNAME = "CAD/CAM"
AND    (DUR = 12 OR DUR = 24)
```



Ejercicio

- Considere el siguiente esquema de base de datos
  - Alumno: matrícula, nombreA, edad, carrera
  - Grupo: claveGrupo, claveUEA, numEcon
  - Lista: idLista, claveGrupo, matrícula
  - Profesor: numEcon, nombreP, correo
  - UEA: claveUEA, nombreUEA, créditos

Ejercicio (1)

SELECT matrícula
FROM Alumno
WHERE (carrera=”Ing. en Computación" **AND** edad>20)
OR ((nombreA= ”Juan López" **OR** nombreA= ”María Rodríguez")
AND (**NOT** nombreA= ”Juan López")
AND (**NOT** nombreA= ”María Rodríguez”))

*Obtenga la forma normal (normalización) y realizar la simplificación, es decir, e*limine la redundancia, en caso de existir, mediante la aplicación de las reglas lógicas de equivalencias y reglas de idempotencia. Finalmente, escriba la consulta simplificada en SQL. Escriba el árbol de expresiones.

Ejercicio (1)

Seleccionar los nombres de los alumnos y su matrícula de la carrera de Ing. en Computación y que han cursado la UEA base de datos distribuidas con el profesor Alejandro Reyes.

- Obtenga la consulta en SQL y la forma normal.
- Obtenga el grafo de la consulta o el grafo de conexión explicar si es o no una consulta válida.
- Realice la simplificación de la consulta en caso de ser necesaria.
- Obtenga el árbol operador basado en algebra relacional para la consulta.