Optimization of Logical Queries

Integrated Exercise 8.3 p3

Translate the following SQL query to the relational algebra, remove redundant joins and use the algebraic laws to produce a better query plan.

```sql
SELECT F.budget, E.eid
FROM Emp E, Dept D, Finance F
WHERE E.did = D.did AND D.did = F.did
    AND E.hobby = 'yodeling'
    AND D.floor NOT IN
        ( SELECT D2.floor FROM Dept D2, Finance F2
            WHERE NOT D2.dname = 'CID'
            OR (F2.did = D2.did AND F2.expenses >= ALL
                (SELECT MAX(F3.expenses)
                    FROM Finance F3
                    WHERE F3.budget = F.budget)
            )
        )
```
Step 1. Normalize to Exists and Not Exists

```
SELECT F.budget, E.eid
FROM Emp E, Dept D, Finance F
WHERE E.did = D.did AND D.did = F.did AND E.hobby = 'yodeling'
  AND NOT EXISTS
  ( SELECT D2.floor FROM Dept D2, Finance F2
    WHERE D.floor = D2.floor AND
      ( NOT D2.dname = 'CID'
        OR (F2.did = D2.did AND NOT EXISTS
            (SELECT MAX(F3.expenses)
             FROM Finance F3
             WHERE F3.budget = F.budget
             HAVING MAX(F3.EXPENSES) > F2.EXPENSES)
            )
        )
    )
  )
```

Solution of the Exercises
Optimization of Logical Queries

Step 2. Normalize to the conjunctive normal form

```
SELECT F.budget, E.eid
FROM Emp E, Dept D, Finance F
WHERE E.did = D.did AND D.did = F.did AND E.hobby = 'yodeling'
    AND NOT EXISTS
( SELECT D2.floor FROM Dept D2, Finance F2
  WHERE (D.floor = D2.floor AND NOT D2.dname = 'CID')
    OR (D.floor = D2.floor AND F2.did = D2.did
        AND NOT EXISTS
            (SELECT MAX(F3.expenses)
             FROM Finance F3
             WHERE F3.budget = F.budget
             HAVING MAX(F3.EXPENSES) > F2.EXPENSES)
    )
)
```
Optimization of Logical Queries

Step 3. Normalize to UNION

SELECT F.budget, E.eid
FROM Emp E, Dept D, Finance F
WHERE E.did = D.did AND D.did = F.did AND E.hobby = 'yodeling'
AND NOT EXISTS
((SELECT D2.floor FROM Dept D2, Finance F2
WHERE D.floor = D2.floor AND NOT D2.dname = 'CID')
UNION
(SELECT D2.floor FROM Dept D2, Finance F2
WHERE D.floor = D2.floor AND NOT D2.dname = 'CID')
UNION
(SELECT D2.floor FROM Dept D2, Finance F2
WHERE D.floor = D2.floor AND F2.did = D2.did
AND NOT EXISTS
(SELECT MAX(F3.expenses)
FROM Finance F3
WHERE F3.budget = F.budget
HAVING MAX(F3.EXPENSES) > F2.EXPENSES)
)
);
Optimization of Logical Queries

Step 4. Translation of the innermost subquery

```sql
SELECT MAX(F3.expenses)
FROM Finance F3
WHERE F3.budget = F.budget
HAVING MAX(F3.EXPENSES) > F2.EXPENSES)
```

$$e_1 := \pi_{\text{MAX(F3.expenses), F.*}, F2.*} \sigma_{\text{MAX(F3.expenses)}>F2.expenses} \gamma_{\text{MAX(F3.expenses), F.*}, F2.*} \sigma_{F3.budget=F.budget} (\rho_F(\text{Finance}) \times \rho_{F2}(\text{Finance}) \times \rho_{F3}(\text{Finance}))$$
Optimization of Logical Queries

Step 5. Translation

\[(\text{SELECT } D2.\text{floor} \text{ FROM Dept } D2, \text{ Finance } F2 \text{ WHERE } D.\text{floor} = D2.\text{floor} \text{ AND } F2.\text{did} = D2.\text{did} \text{ AND NOT EXISTS} \]
\[(\text{SELECT } \text{MAX}(F3.\text{expenses}) \text{ FROM Finance } F3 \text{ WHERE } F3.\text{budget} = F.\text{budget} \text{ HAVING } \text{MAX}(F3.\text{EXPENSES}) > F2.\text{EXPENSES})\]

Translating the From clause yields:
\[e_2 := \rho_{D2}(\text{Dept}) \times \rho_{F2}(\text{Finance}) \times \rho_{D}(\text{Dept}) \times \rho_{F}(\text{Finance})\]

Since this is a NOT EXISTS subquery, we decorrelate by means of an antijoin:
\[f_2 := (e_2 \ \emptyset \ \pi_{F2.\ast,F.\ast}(e_1))\]

Adding the WHERE and SELECT clauses (with the necessary parameters) gives:
\[e_3 := \pi_{D2.\text{floor},D.\ast,F.\ast} \sigma_{D.\text{floor}=D2.\text{floor} \wedge F2.\text{did}=D2.\text{did}}(f_2)\]
Optimization of Logical Queries

Step 6. Translation

\[
\left( (\text{SELECT D2.floor FROM Dept D2, Finance F2}
\text{ WHERE D.floor = D2.floor AND NOT D2.dname = 'CID'}) \right)
\]

\[
e_4 := \pi_{\text{D2.floor}, D.\ast \sigma_{D.floor=D2.floor \land D2.name \neq 'CID'}(\rho_{D2}(\text{Dept}) \times \rho_{F2}(\text{Finance}) \times \rho_{D}(\text{Dept}))}
\]
Optimization of Logical Queries

Step 7. Translation of the union

\[
((\text{SELECT } D2.\text{floor} \text{ FROM Dept } D2, \text{ Finance } F2 \\
\text{ WHERE } D.\text{floor} = D2.\text{floor} \text{ AND NOT } D2.\text{dname} = \text{ 'CID'})) \\
\text{UNION} \\
(\text{SELECT } D2.\text{floor} \text{ FROM Dept } D2, \text{ Finance } F2 \\
\text{ WHERE } D.\text{floor} = D2.\text{floor} \text{ AND } F2.\text{did} = D2.\text{did} \\
\text{ AND NOT EXISTS} \\
(\text{SELECT MAX}(F3.\text{expenses}) \\
\text{ FROM Finance } F3 \\
\text{ WHERE } F3.\text{budget} = F.\text{budget} \\
\text{ HAVING MAX}(F3.\text{EXPENSES}) > F2.\text{EXPENSES})
\)
\]

Notice that the schemas of \( e_3 \) and \( e_4 \) are not equivalent because they have other context relations. Therefore, to be able to take the union, we have to add the context relation \( F \) of \( e_3 \) to \( e_4 \).

\[
e_5 := \pi_{D2.\text{floor}, D.*, F.*}(e_3) \cup \pi_{D2.\text{floor}, D.*, F.*}(e_4 \times \rho_F(\text{Finance}))
\]
Optimization of Logical Queries

Step 8. Translation of the outermost query

Translation of the From clause yields:

\[ e_6 := \rho_E(Emp) \times \rho_D(Dept) \times \rho_F(Finance) \]

Since this is a NOT EXISTS subquery, we decorrelate by means of an antijoin:

\[ f_7 := e_6 \bowtie \pi_{D.*,F.*}(e_5) \]

And translate the remaining WHERE and SELECT clauses:

\[ e_7 := \pi_{F.budget,E.eid}(\sigma_{E.did=D.did \wedge D.did=F.did \wedge E.hobby='yodeling'}(f_7)) \]
Whole expression:

\[ \pi_{\text{F.budget,E.eid}}(\sigma_{\text{E.did=D.did}\land\text{D.did=F.did}\land\text{E.hobby='yodeling'}}(\rho_{\text{E.Emp}} \times \rho_{\text{D.Dept}} \times \rho_{\text{F.Finance}})) \]

\[ \setminus (\pi_{\text{D.}*F.} \ast \sigma_{\text{D.floor=D2.floor}\land\text{D2.name='CID'}}(\rho_{\text{D2.Dept}} \times \rho_{\text{F2.Finance}}) \]

\[ \times \rho_{\text{D.Dept}} \times \rho_{\text{F.Finance}}) \]

\[ \bigcup \pi_{\text{D.}*F.}((\sigma_{\text{D.floor=D2.floor}\land\text{F2.did=D2.did}}(\rho_{\text{D2.Dept}} \times \rho_{\text{F2.Finance}}) \]

\[ \times \rho_{\text{D.Dept}} \times \rho_{\text{F.Finance}})) \]

\[ \setminus (\pi_{\text{F2.}*F.} \ast \sigma_{\text{MAX(F3.expenses)>F2.expenses}} \gamma_{\text{MAX(F3.expenses),F.},F2.} \]

\[ \sigma_{\text{F3.budget=F.budget}}(\rho_{\text{F.Finance}} \times \rho_{\text{F2.Finance}} \times \rho_{\text{F3.Finance}}))) \]
Optimization of Logical Queries

Step 9. Removal of redundant joins

No atom can be removed from the following maximal subexpressions (why?)

- $\sigma_{E.did = D.did \land D.did = F.did \land E.hobby = 'yodeling'}(\rho_E(Emp) \times \rho_D(Dept) \times \rho_F(Finance))$
- $(\sigma_{D.floor = D2.floor \land F2.did = D2.did}(\rho_{D2}(Dept) \times \rho_{F2}(Finance) \times \rho_D(Dept) \times \rho_F(Finance)))$
- $\sigma_{F3.budget = F.budget}(\rho_F(Finance) \times \rho_{F2}(Finance) \times \rho_{F3}(Finance)))$
Optimization of Logical Queries

Step 9. Removal of redundant joins

Subexpression

\[ \pi_{D.*,F.*} \sigma_{D.floors=D2.floors \land D2.name \neq 'CID'}(\rho_{D2}(Dept) \times \rho_{F2}(Finance) \times \rho_{D}(Dept) \times \rho_{F}(Finance)) \]

Is not a select-project-join expression, because of the inequality. Therefore, no redundant join can be removed.

Therefore, in this example, no redundant join can be removed.
Optimization of Logical Queries

Step 10. Application of the algebraic laws

\[
\pi_{F.budget,E.eid}(\pi_{F.*,E.eid}(\pi_{E.eid,e.did}\sigma_{E.hobby='yodeling'}\rho_E(Emp)) \\
\times \pi_{D.did}\rho_D(Dept) \times \rho_F(Finance))) \\
\times \pi_{D2.floor}\rho_D(Dept) \times \pi\rho_{F2}(Finance) \\
\times \rho_F(Finance)) \\
\cup \pi_{F.*}((\pi_{F.*,F2.*}(\pi_{D.floor}\rho_{D2}(Dept)) \times \pi_{D2.floor,D2.did}\rho_D(Dept) \\
\times \rho_{F2}(Finance) \times \rho_F(Finance)) \\
\times F2.did=D2.did) \\
\times \pi_{F2.*,F.*}\sigma_{MAX(F3.expenses)>F2.expenses\gamma_{MAX(F3.expenses),F.*,F2.*}}(\rho_F(Finance)) \\
\times \pi_{F3.budget,F3.expenses}\rho_{F3}(Finance) \times \rho_{F2}(Finance)))) \\
\times F3.budget=F.budget
\]