Query optimization

Tutorial

Exercise 1

Exercise 1. Optimize logically

- Consider the following relational schema:

Hotel (id, name, address)

Room (rid, hid, type, price)

Booking (hid, gid, date_ from, date_to, rid)

- Translate the following SQL queries into the relational algebra and use the algebraic laws to improve the query plans. Draw tree plans (or write RA expressions), and explain the logic behind each optimization step.

- Query A

```
SELECT R.rid, R.type, R.price
FROM Room R, Booking B, Hotel H
WHERE R.rid = B.rid AND B.hid = H.hid
AND H.name = 'Hilton' AND R.price > 100
```

- Query B

```
SELECT G.gid, G.name
FROM Room R, Hotel H, Booking B, Guest G
WHERE H.hid = B.hid AND G.gid = B.gid
AND H.hid = R.hid AND H.name = 'Hilton'
AND date_from >= '1-Oct-2003' AND date_to <= '31-Dec-2003'
```

```
SELECT R.rid, R.type, R.price
FROM Room R, Booking B, Hotel H
WHERE R.rid = B.rid AND B.hid = H.hid
AND H.name = 'Hilton' AND R.price > 100
```

- The translation gives us the following relational algebra expression:

$$\pi_{\text{R.rid,R.type,R.price}}\sigma_{\text{R.rid=B.rid}\wedge\text{B.hid=H.hid}\wedge\text{H.name='Hilton'}\wedge\text{R.price>100}}(\rho_{\text{R}}(\text{Room}) \times \rho_{\text{H}}(\text{Hotel}) \times \rho_{\text{B}}(\text{Booking}))$$

Optimization 1

- First, we split the selections:

$$\pi_{\text{R.rid,R.type,R.price}}\sigma_{\text{R.rid=B.rid}}\sigma_{\text{B.hid=H.hid}}\sigma_{\text{H.name='Hilton'}}\sigma_{\text{R.price>100}}(\rho_{\text{R}}(\text{Room}) \times \rho_{\text{H}}(\text{Hotel}) \times \rho_{\text{B}}(\text{Booking}))$$

And we push the selections:

$$\pi_{\text{R.rid,R.type,R.price}}\sigma_{\text{R.rid=B.rid}}(\sigma_{\text{R.price>100}}\rho_{\text{R}}(\text{Room}) \times \sigma_{\text{B.hid=H.hid}}(\sigma_{\text{H.name=0Hilton0}}\,\rho_{\text{H}}(\text{Hotel}) \times \rho_{\text{B}}(\text{Booking})))$$

Optimization 2

- Then, the joins are recognized:

$\pi_{R.rid,R.type,R.price}($

$\sigma_{R.price>100}\rho_R(Room) \bowtie$ (

$\sigma_{H.name='Hilton'}\ \rho_H(Hotel) \bowtie \rho_B(Booking)$

)

)

Optimization 3

- Finally, the projections are pushed:

$$\pi_{R.rid,R.type,R.price}\ ($$
$$\pi_{R.rid,R.type,R.price}\ \sigma_{R.price>100}\rho_{R}(Room) \bowtie\ \ ($$
$$\sigma_{H.name='Hilton'}\ \rho_{H}(Hotel) \bowtie \rho_{B}(Booking)$$
$$)$$
$$)$$

Exercise 2

Parameters to estimate the cost

- **R**: the name of the relation on disk
- **B(R)**: number of blocks of R
- **T(R)**: number of tuples of R
- **V(R, a)**: number of distinct values in column ***a*** of R

Estimating the output

Equality: $S = \sigma_{A=c}(R)$ **T(S) = T(R) / V(R,A)**

Range: $S = \sigma_{A<c}(R)$ **T(S) = T(R)/3**

Inequality: $S = \sigma_{A\neq c}(R)$ **T(S) = T(R)**

Conjunction: $S = \sigma_{C=c \text{ AND } D=d}(R)$ **T(S) = T(R)/(V(R,C)*V(R,D))**

Natural join: $P = R(X,Y) \bowtie S(Y,Z)$ **T(P) = T(R)*T(S)/max (V(R,Y),V(S,Y))**

Duplicate elimination: **$T(\delta(R)) = \min[V(R,a_1)*\ldots*V(R,a_n), 1/2*T(R)]$**

Exercise for size estimation

W (a,b)	X (b,c)	Y (c,d)	Z (d,e)
T (W) = 100	T (X) = 200	T (Y) = 300	T (Z) = 400
V (W, a) = 20	V (X, b) = 50	V (Y, c) = 50	V (Z, d) = 40
V (W, b) = 60	V (X, c) = 100	V (Y, d) = 50	V (Z, e) = 100

Estimate the sizes of relations that are the results of the following queries:

a) $\sigma_{a=10}$ (W)

b) $\sigma_{c=20}$ (W)

c) $\sigma_{d>10}$ (Z)

d) $\sigma_{a=1 \text{ AND } b=2}$ (W)

e) W x Y

f) W ⋈ X ⋈ Y ⋈ Z

g) $\sigma_{a=1 \text{ AND } b>2}$ (W) ⋈ X

Exercise 3

3. Cost of joins

- Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances (organized on a per department basis).
- Schema:

Emp (eid, deptid, salary, hobby)

Dept (deptid, deptname, floor, phone)

Finance (deptid, budget, sales, expenses)

- Query:

SELECT D.name, F.budget

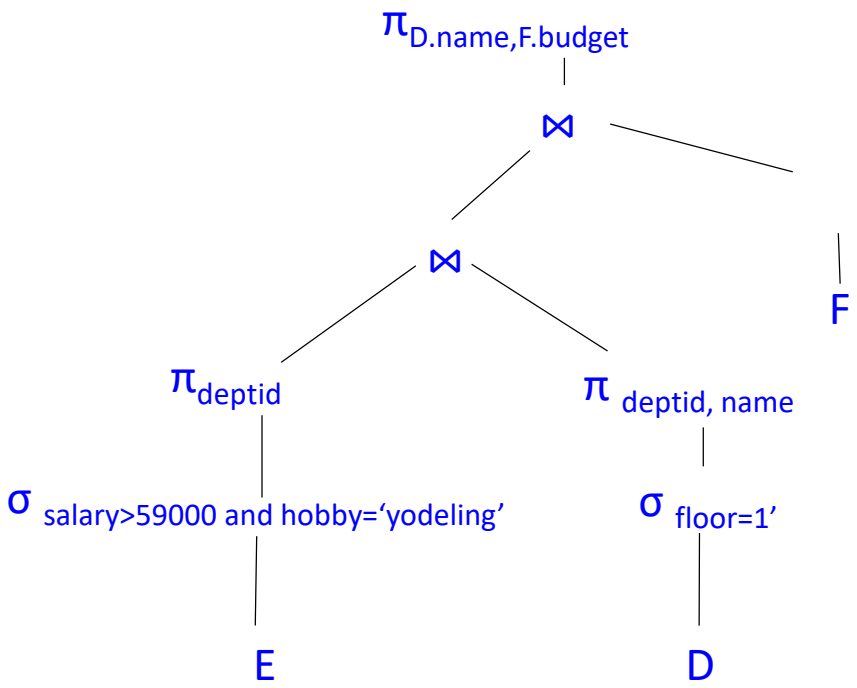
FROM Emp E, Dept D, Finance F

WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid

AND and D.floor = 1 AND E.salary > 59000 AND E.hobby = ‘yodeling’

```
SELECT D.name, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid
AND D.floor = 1 AND E.salary > 59000 AND E.hobby = ‘yodeling’
```

A. Identify a relational algebra tree that reflects the order of operations a decent query optimizer would choose.



```
SELECT D.name, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid
AND D.floor = 1 AND E.salary > 59000 AND E.hobby = 'yodeling'
```

- B. List the join orders (i.e., orders in which pairs of relations can be joined to compute the query. (Assume that the optimizer never considers plans that require the computation of cross-products). Briefly explain how you arrived at your list.

3 joins, can be executed in any order. We need to join 3 relations and for all of them the join attribute is deptid, so all join orders are valid - no Cartesian product

- $(E \bowtie D) \bowtie F$
- $(D \bowtie F) \bowtie E$
- $(E \bowtie F) \bowtie D$

```
SELECT D.name, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid
AND D.floor = 1 AND E.salary > 59000 AND E.hobby = ‘yodeling’
```

C. Suppose that the following statistics are available from the system catalog:

- Statistics:

1. Employee salaries range from 10,000 to 60,000.
2. Employees enjoy 200 different hobbies.
3. The company owns 2 floors of the building.
4. There are a total of 50,000 employees and 5,000 departments (each with the corresponding single entry in the Finance table)

- The DBMS used by the company has only one join method available: block nested loop join.

```
SELECT D.name, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid
AND D.floor = 1 AND E.salary > 59000 AND E.hobby = ‘yodeling’
```

- For each of the query’s relations (E, D, and F) estimate the number of tuples that would be initially selected from that relation if all non-join predicates were applied to them before any joining. Given your answer to the previous questions, which of the join orders produced in question B has the lowest estimated cost?

```
SELECT D.name, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.deptid = D.deptid AND Dept.deptid = F.deptid
AND D.floor = 1 AND E.salary > 59000 AND E.hobby = 'yodeling'
```

Given:

- T(D) = 5,000
- T(E) = 50,000
- T(F) = 5,000
- V(D, deptid) = 5,000
- V(E, hobby) = 200
- V (D, floor) = 2

A. (E ⋈ D) ⋈ F

Estimating intermediate size of E ⋈ D:

83*2500/5000 = 42

(Note that though cardinality V(D',deptID) =2,500 after selection, not all deptid in E would join. So the total cardinality V(D, deptID) should remain 5,000)

B. (D ⋈ F) ⋈ E

Estimating intermediate size (D ⋈ F) :

2500 * 5000 / 5000 = 2500

C. (E ⋈ F) ⋈ D

Estimating intermediate size (E ⋈ F):

83*5000 / 5000 = 83

- The preferred join order is A

After selections the estimated table sizes are:
For E:
50,000*1/3*1/200=83 tuples
For D:
5,000/2 = 2500 tuples
F unchanged at 5,000 tuples