Introduction to database systems

Exercises: query evaluation and optimization

Task

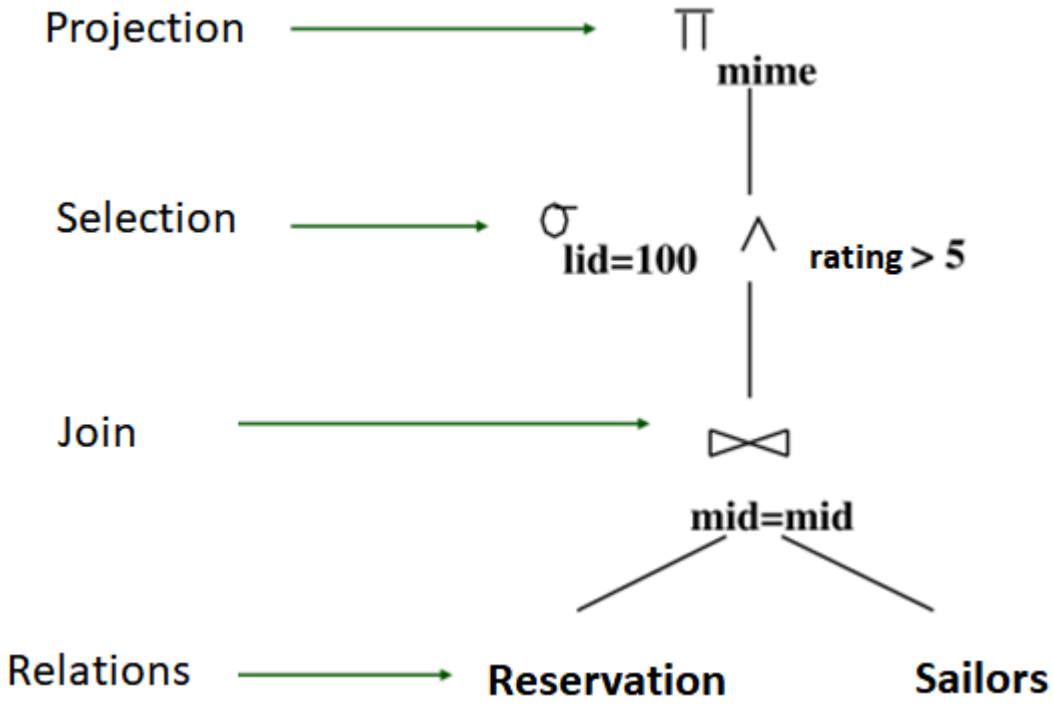
- estimate execution plan of SQL queries:

  we are counting the number of transfered pages from/to the disk (from a dynamic memory)

- we can have several different execution plans for each query
- If we understand the plan, we can choose the optimal solution

What is query execution plan

```
SELECT    M.ime
FROM      Reservation R, Sailors M
WHERE     R.mid = M.mid
          AND R.lid = 100 AND M.grade > 5
```



In relational algebra:

$\pi_{mime}(\sigma_{lid\ =\ 100\ \wedge\ grade\ >\ 5}(Reservation \bowtie_{mid\ =\ mid} Sailors))$

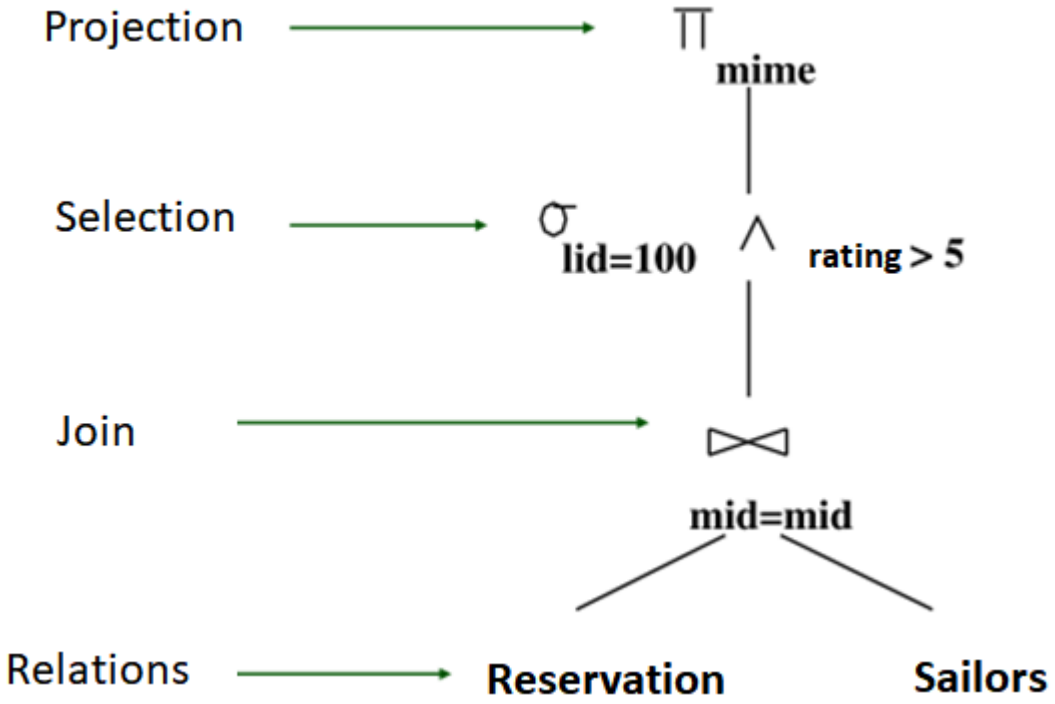
Cost: M + M * N = 1000 + 1000 * 500 = 501 000 pages

Reservation [mid, bid, dan, rime]
entry size 40 bytes
prm = page size 100 entries
M = 1000 pages

Sailors [mid, mime, ocena, starost]
entry size 50 bytes
prn = page size 80 entries
N = 500 pages

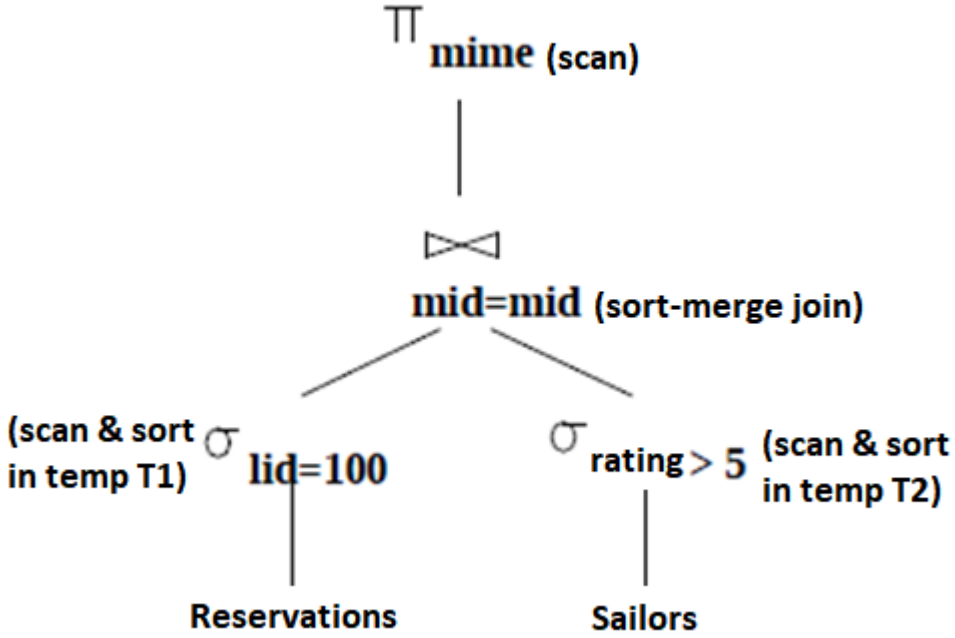
Optimization

1st plan:



Cost: 1000 + 1000 * 500 = 501 000 pages

2nd plan:



Cost: 1010 + 750 + 40 + 1500 + 260 = 3560 pages

Formulas

1. Simple Nested Loops Join: M + M*prm*N (M – outer relation, N- inner relation)
N + N*prn*M (N – outer relation, M- inner relation)

- For each tuple in the outer relation R1, we scan the entire inner relation R2.

2. Block Nested Loops Join: M+M*N

- For each page of R1, get each page of R2, and write out matching pairs of tuples
- If we have enough space in the memory for N+2 pages, where N is smaller relation: M+N
- If there is not enough space (lets say B=102):
  M+N*(M/(B-2)) = 1000+500*1000/100=6000

3. Index Nested Loops Join (if we have hash index on one of the join attributes, we put relation with index on the inner part of the join)

Example for hash index: N+M*prn*1,2 M+N*prm*1,2

Formulas

4. Sort-Merge Join: M and N (example: B=100)

$2*|M|* (1+\log_{B-1}(|M|/B)) + 2*|N|* (1+\log_{B-1}(|N|/B)) + |M| + |N|$

Example: 2*1000*(1+1) + 2*500*(1+1) + 1000 + 500 = 7500

Also useful:

$\log_{B-1}(|M|/B) = \log_{10}(M/B) / \log_{10}(B-1)$

Formulas

5. Hash Join (hash index on both join attributes)
Both relations with hash index we split to B-1 partitions

- 3*(M+N), if B > √N (N is smaller relation)
- If not, we have the example of excesses and each partition of the smaller relationships, that doesn‘t fit into the memory is recursively divided.

Rough estimations for index

B+ tree index:

1,5 * (#pages_of_table/#pages_for_data_entry_of_index + 1)

- B+ index is 67% full on average, hence => 100/67 = 1,492 = rounded to 1,5

Hash index:

1,25 * (#pages_of_table/#pages_for_data_entry_of_index + 1)

- Hash index is 80% full on average, hence => 100/80 = 1,25

#pages_of_table => table on which we have the index

Calculation for #pages_for_data_entry_of_index:

- |Data entry| = 16B (except if task says otherwise)
- #Data_entries / page = Size_of_disk_page / 16B
- #Pages_of_table_where_index_is_on / result of previous step ^

Exercises

Schema:

PILOTS (IdP int(11), Surname varchar(50), Name varchar(50), Emso varchar(13), Address varchar(250), EmploymentDate date, YearsOfExperience varchar(20), NoOfFlights int(11), NoOfHoursPiloting int(11), IdLD int(11))

FLIGHTS (IdLT int(11), IdP int(11), IdL int(11), IdLEFlight int(11), FlightDate date, FlightTime time, IdLELand int(11), LandDate date, LandTime time)

RESERVATIONS (IdR int(11), Number int(11), Price float, Surname varchar(50), Name varchar(50), Address varchar(250), Telephone varchar(20), SeatNo int(11), Class int(11), IdLT int(11))

Take the following information into account:

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

FLIGHTS: 20 records per page, 150 pages

RESERVATIONS: 50 records per page, 30000 pages, there are 150 different reservation prices and 70 of those are higher than 80.000.

For each query, we have 10,000 pages in the buffer (DBMS).

Exercise 1

```
SELECT Surname, Name
FROM Pilots
WHERE NoOfHoursPiloting>500
```

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

Build a query execution plan for each query and give a cost estimation of the execution plan:

a)
```
SELECT Surname, Name
FROM PILOTS
```

b)
```
SELECT Surname, Name
FROM Pilots
WHERE NoOfHoursPiloting>500
```

Exercise 2

```
SELECT Surname, Name
FROM Pilots
WHERE NoOfHoursPiloting>500
```

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

What is the cost of the execution plan for the second query in the previous case if it uses a clustered tree index (height 2) on the attribute NoOfHoursPiloting?

Exercise 3

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

FLIGHTS: 20 records per page, 150 pages

RESERVATIONS: 50 records per page, 30000 pages, there are 150 different reservation prices and 70 of those are higher than 80.000.

The following algorithms for the imlementation of joins are available: Sort-Merge Join and Simple Nested Loops Join.

a) Evaluate the implementation plan: Find all IdL (plane ids), where the price for reservation of their flights is higher than 80,000.

Exercise 3

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

FLIGHTS: 20 records per page, 150 pages

RESERVATIONS: 50 records per page, 30000 pages, there are 150 different reservation prices and 70 of those are higher than 80.000.

The following algorithms for the imlementation of joins are available: Sort-Merge Join and Simple Nested Loops Join.

b) Evaluate the implementation plan: Find all IdL (plane ids), where the price for reservation of its flights pri is higher than 80,000 and was piloted by a pilot with id IdP=50.

Exercise 4

PILOTS: 10 records per page, 40 pages, each record represents a single pilot, relation stores 100 different values for NoOfHoursPiloting and the pilots are evenly distributed among these values, at the same time there are 70 different values over 500.

FLIGHTS: 20 records per page, 150 pages

RESERVATIONS: 50 records per page, 30000 pages, there are 150 different reservation prices and 70 of those are higher than 80.000.

For how much is the cost of the execution plan for the second query in the previous exercise different if it uses in its execution:

- 2-level tree index on IdP in Flight relation and
- Hash index on Price and hash index on IdLT in Reservations relation