INFO20003 Database Systems

Dr Renata Borovica-Gajic

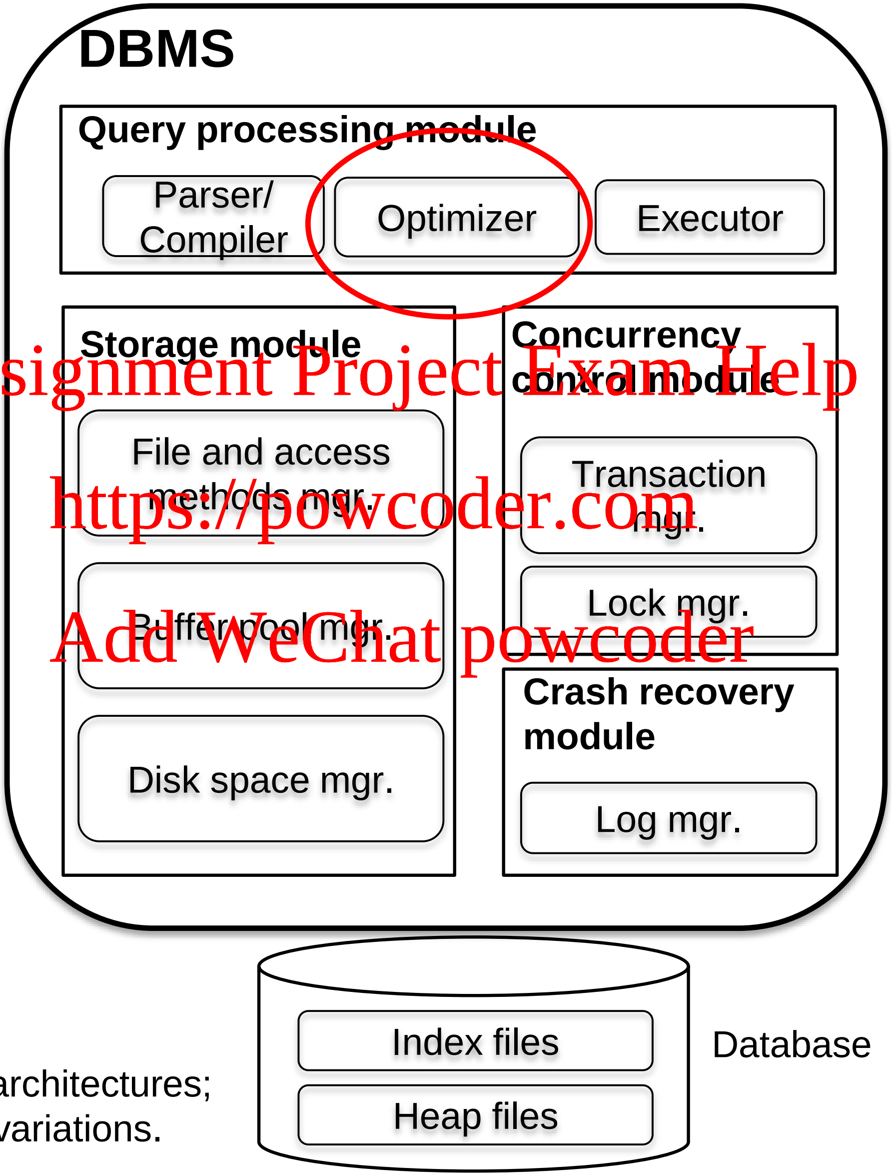
Lecture 13
Query Optimization Part I

Semester 2 2018, Week 7

A1 Collective Feedback

- Assignment 1 Feedback will be sent by the end of this week
- Best way to prepare for the MST is to look at your mistakes, compare with the provided solution, and try to mark yourself against the provided assessments criteria
  - Self-reflection is an important part of learning
  - Pay attention to business rules that your model does not support
  - Let’s have a look at one potential solution
- This assignment was a preview of a business analyst’s life

Remember this? Components of a DBMS



TODAY & Next time

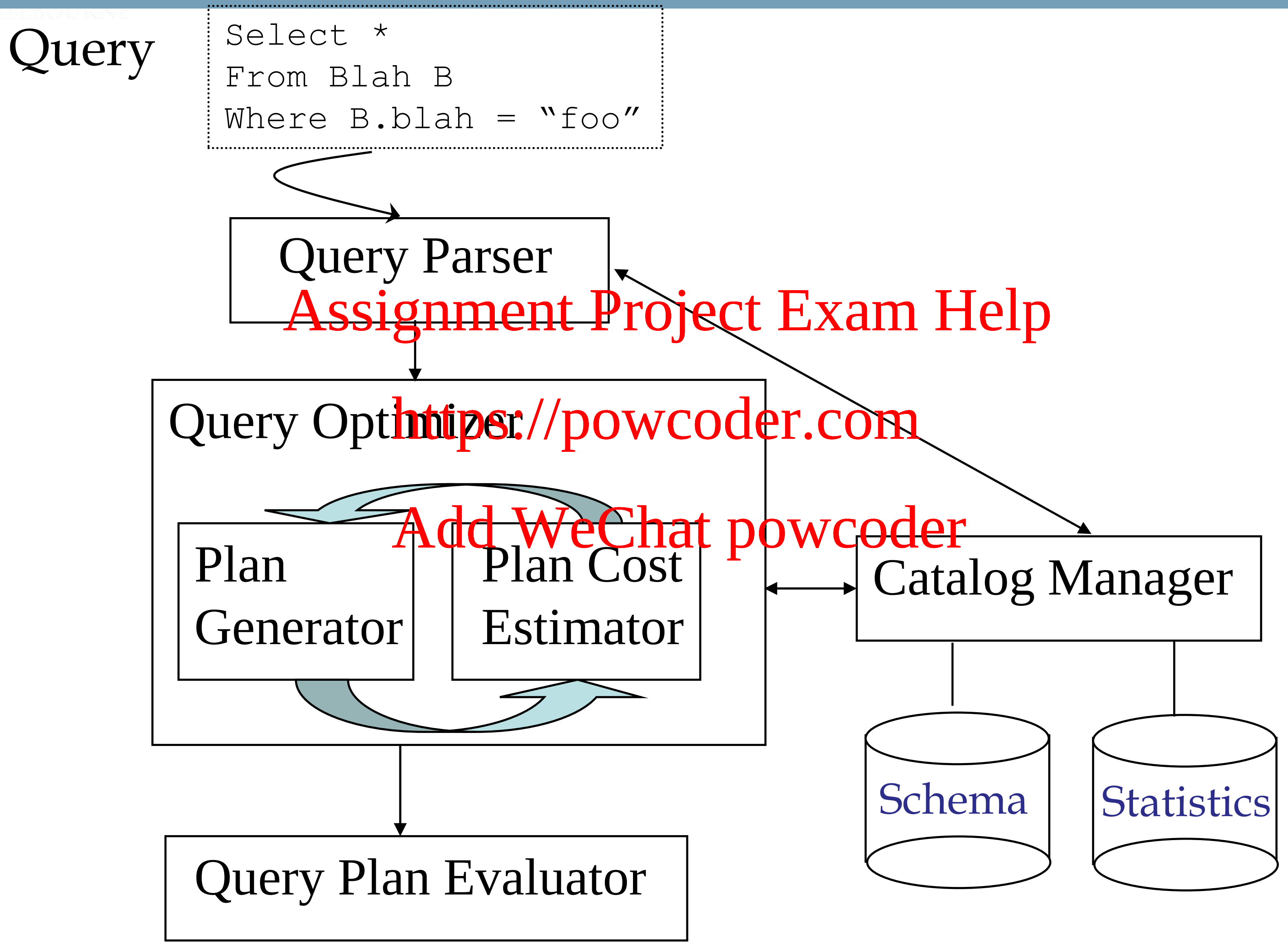
This is one of several possible architectures; each system has its own slight variations.

Coverage

- Overview
- Query optimization
- Cost estimation

Readings: Chapter 12 and 15, Ramakrishnan & Gehrke, Database Systems

Query Processing Workflow: Review



Query Optimization

- Typically there are many ways of executing a given query, all giving the same answer
- Cost of alternative methods often **varies enormously**
- Query optimization aims to find the execution strategy with the lowest cost

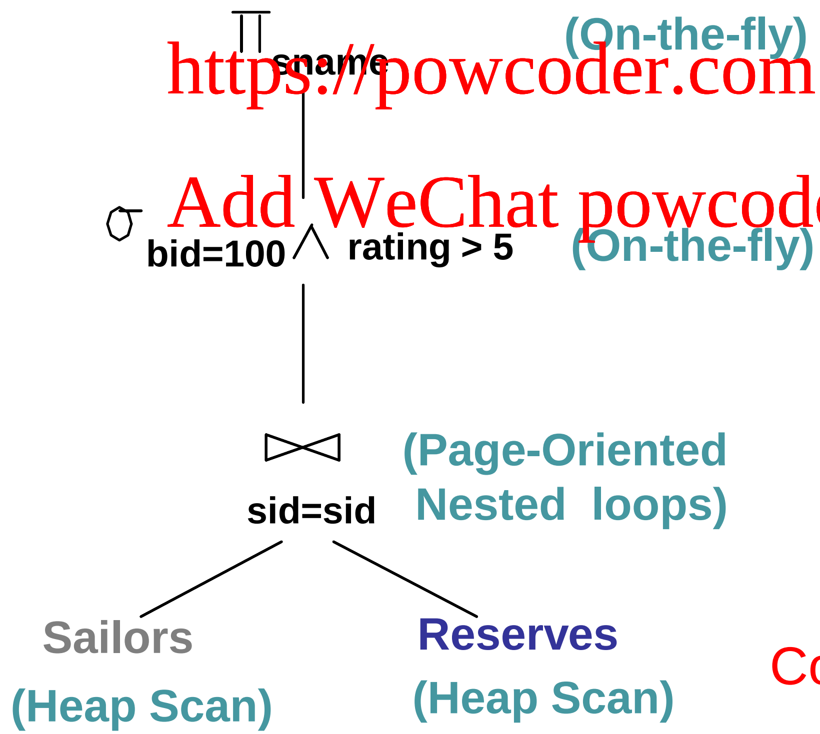
- We will cover:
  - Relational algebra equivalences
  - Cost estimation
    - Result size estimation and reduction factors
  - Enumeration of alternative plans

Query Plan

- A tree, with relational algebra operators as nodes
- Each operator labeled with a choice of algorithm

SELECT sname from Sailors NATURAL JOIN Reserves
WHERE bid = 100 and rating > 5

Plan:



Cost: 500+500*1000 I/O

- Overview
- Query optimization
- Cost estimation

Readings: Chapter 15, Ramakrishnan & Gehrke, Database Systems

A Familiar Schema for Examples

Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)

Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

Boats (*bid*: integer, *bname*: string, *color*: string)

Query Optimization Overview

Example:

```
SELECT  S.sname
  FROM  Reserves R, Sailors S
WHERE  R.sid=S.sid AND
   R.bid=100 AND S.rating>5
```


Query optimization steps:

1. Query first broken into “blocks”
2. Each block converted to relational algebra
3. Then, for each block, several alternative query plans are considered
4. Plan with the lowest estimated cost is selected

Step 1: Break query into query blocks

- Query block is any statement starting with select
- Query block = unit of optimization
- Typically inner most block is optimized first, then moving towards outers

Outer block

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.age IN
    (SELECT  MAX (S2.age)
     FROM  Sailors S2
     GROUP BY  S2.rating)
```

Nested block

Step 2: Convert query block into relational algebra expression

Query:

```
SELECT  S.sid
FROM  Sailors S, Reserves R, Boats B
WHERE  S.sid = R.sid AND R.bid = B.bid AND B.color = "red"
```


Relational algebra:

$$\pi_{S.sid}(\sigma_{B.color = \text{"red"}}(\text{Sailors} \bowtie \text{Reserves} \bowtie \text{Boats}))$$

Step 3: Relational Algebra Equivalences

- *Selections*: $\sigma_{c_1 \wedge \cdots \wedge c_n}(R) \equiv \sigma_{c_1}\left(\ldots\left(\sigma_{c_n}(R)\right)\right)$ $(Cascade)$

  $\sigma_{c_1}\left(\sigma_{c_2}(R)\right) \equiv \sigma_{c_2}\left(\sigma_{c_1}(R)\right)$ $(Commute)$

- *Projections:* $\pi_{a_1}(R) \equiv \pi_{a_1}\left(\ldots\left(\pi_{a_n}(R)\right)\right)$ *(Cascade)*

  a_i is a set of attributes of R and $a_i \subseteq a_{i+1}$ for $i = 1 \ldots n-1$

- These equivalences allow us to 'push' selections and projections ahead of joins.

Examples

Selection:

$$\sigma_{age<18 \wedge rating>5}\ (\text{Sailors})$$

$$\leftrightarrow \sigma_{age<18}\ (\sigma_{rating>5}\ (\text{Sailors}))$$

$$\leftrightarrow \sigma_{rating>5}\ (\sigma_{age<18}\ (\text{Sailors}))$$

Projection:

$$\cancel{\pi_{age,rating}\ (\text{Sailors}) \leftrightarrow \pi_{age}\ (\pi_{rating}\ (\text{Sailors}))}\ ?$$

$$\pi_{age,rating}\ (\text{Sailors}) \leftrightarrow \pi_{age,rating}\ (\pi_{age,rating,sid}\ (\text{Sailors}))$$

Another Equivalence

- A projection commutes with a selection that only uses attributes retained by the projection

$$\pi_{\text{age, rating, sid}}\ (\sigma_{\text{age<18} \wedge \text{rating>5}}\ (\text{Sailors}))$$

$$\leftrightarrow\ \sigma_{\text{age<18} \wedge \text{rating>5}}\ (\pi_{\text{age, rating, sid}}\ (\text{Sailors}))$$

$$\pi_{\text{age, sid}}\ (\sigma_{\text{age<18} \wedge \text{rating>5}}\ (\text{Sailors}))$$

?

$$\cancel{\leftrightarrow\ \sigma_{\text{age<18} \wedge \text{rating>5}}\ (\pi_{\text{age, sid}}\ (\text{Sailors}))}$$

Equivalences Involving Joins

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \qquad \textit{(Associative)}$$

$$(R \bowtie S) \equiv (S \bowtie R) \qquad \textit{(Commutative)}$$

- These equivalences allow us to choose **different join orders**

Mixing Joins with Selections & Projections

- Converting selection + cross-product to join

$$\sigma_{S.sid = R.sid}\ (\text{Sailors x Reserves})$$

$$\leftrightarrow \text{Sailors} \bowtie_{S.sid = R.sid} \text{Reserves}$$

- Selection on just attributes of S commutes with R $\bowtie$ S

$$\sigma_{S.age<18}\ (\text{Sailors} \bowtie_{S.sid = R.sid} \text{Reserves})$$

$$\leftrightarrow (\sigma_{S.age<18}\ (\text{Sailors})) \bowtie_{S.sid = R.sid} \text{Reserves}$$

- We can also "push down" projection (*but be careful…*)

$$\pi_{S.sname}\ (\text{Sailors} \bowtie_{S.sid = R.sid} \text{Reserves})$$

$$\leftrightarrow \pi_{S.sname} (\pi_{sname,sid}(\text{Sailors}) \bowtie_{S.sid = R.sid} \pi_{sid}(\text{Reserves}))$$

Query Optimization

- Overview
- Query optimization
- Cost estimation

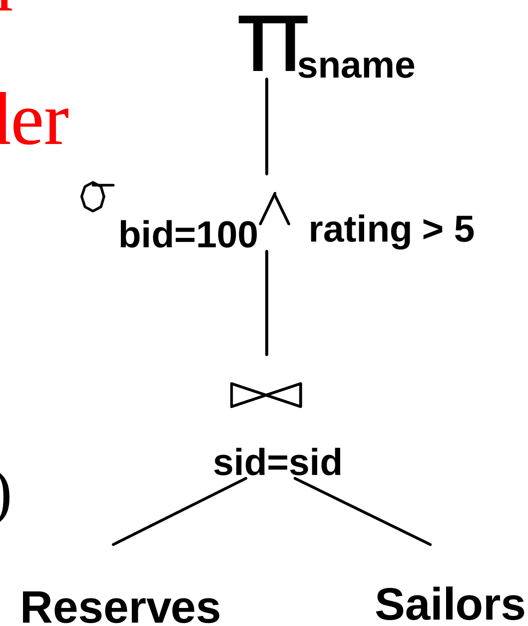
Readings: Chapter 15, Ramakrishnan & Gehrke, Database Systems

THE UNIVERSITY OF MELBOURNE

Recall: Query Optimization Overview

1. Query first broken into “blocks”
2. Each block converted to relational algebra
3. Then, for each block, several alternative query plans are considered
4. Plan with lowest estimated cost is selected


```
SELECT  S.sname
FROM  Reserves R, Sailors S
WHERE  R.sid=S.sid AND
   R.bid=100 AND S.rating>5
```



$$\pi_{(sname)}\sigma_{(bid=100 \wedge rating > 5)} (Reserves \bowtie Sailors)$$

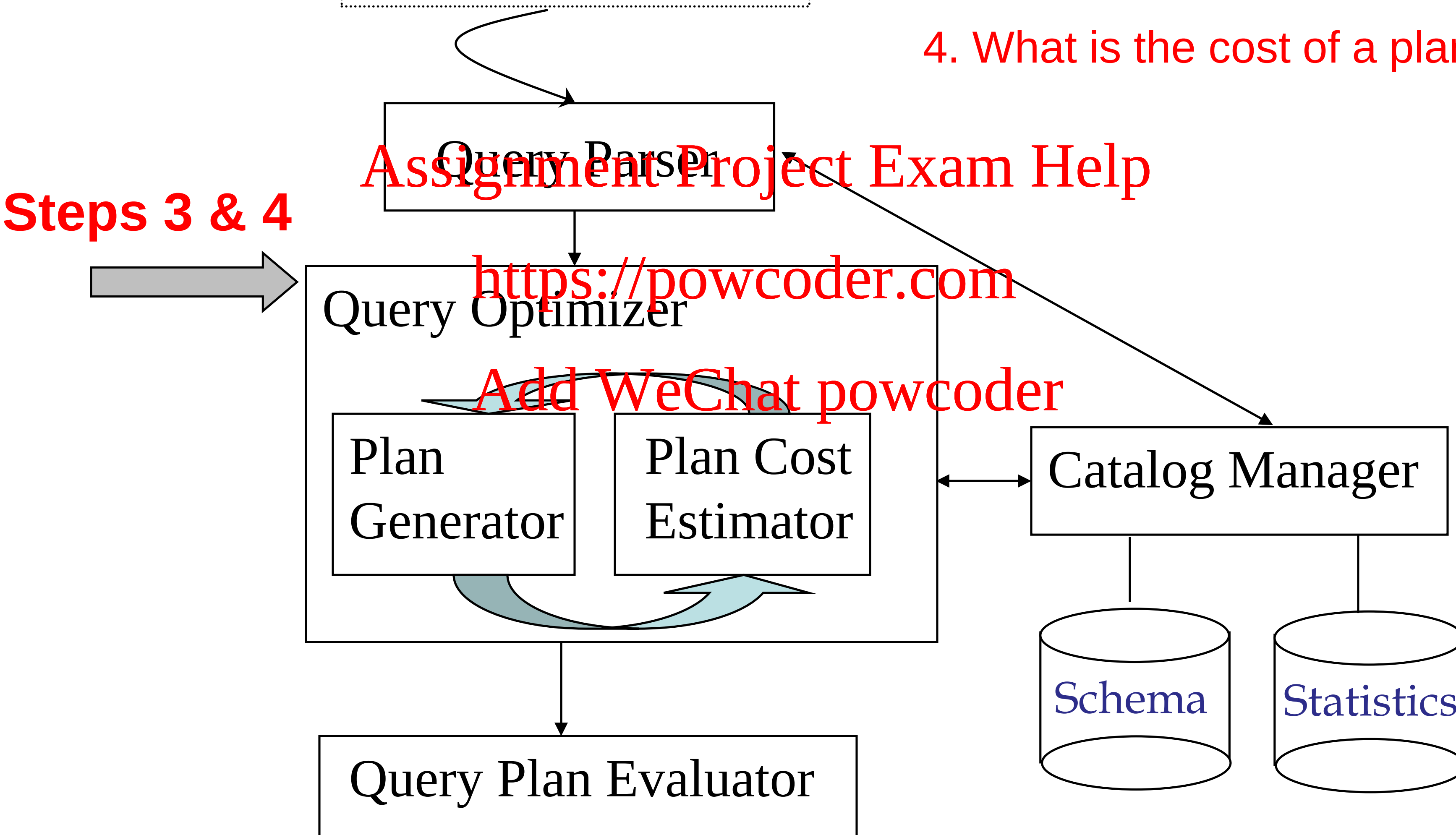
Cost-based Query Sub-System

Queries

```
Select *
From Blah B
Where B.blah = "foo"
```

3. What plans are considered?

4. What is the cost of a plan?



Cost Estimation

- For each plan considered, must estimate cost:
  - Must *estimate size of result* for each operation in tree
    - Use information about input relations (from the system catalogs), and apply rules (discussed next)
  - Must *estimate cost* of each operation in plan tree
    - Depends on input cardinalities
    - We've already discussed how to estimate the cost of operations (sequential scan, index scan, joins)
    - Next time we will calculate the cost of entire plans…

Statistics and Catalogs

- To decide on the cost, the optimizer needs information about the relations and indexes involved. This information is stored in the system **catalogs**.

- **Catalogs** typically contain at least:
  - # tuples (**NTuples**) and # pages (**NPages**) per relation
  - # distinct key values (**NKeys**) for each index (or relation attribute)
  - low/high key values (**Low/High**) for each index (or relation attribute)
  - Index height (**Height(I)**) for each tree index
  - # index pages (**NPages(I)**) for each index

- Statistics in catalogs are updated periodically

Result size estimation

- Consider a query block:

```
SELECT  attribute list
  FROM  relation list
 WHERE  predicate1 AND ... AND predicate_k
```


- Maximum number of tuples in the result is the product of the cardinalities of relations in the FROM clause

- Reduction factor (RF) associated with each predicate reflects the impact of the predicate in reducing the result size. RF is also called selectivity.

Result size estimation calculations

- Single table selection:

$$\textbf{ResultSize} = \mathbf{N}\boldsymbol{Tuples}(\boldsymbol{R})\prod_{\boldsymbol{i=1..n}}\boldsymbol{RF_i}$$

- Joins (over k tables):

$$\textbf{ResultSize} = \prod_{\boldsymbol{j=1..k}}\boldsymbol{NTuples}(\boldsymbol{R_j})\prod_{\boldsymbol{i=1..n}}\boldsymbol{RF_i}$$

- If there are no selections (no predicates), reduction factors are simply ignored, i.e. they are ==1

Calculating Reduction Factors(RF)

- Depend on the type of the predicate:

1. Col = value
   RF = 1/NKeys(Col)

2. Col > value
   RF = (High(Col) – value) / (High(Col) – Low(Col))

3. Col < value
   RF = (val – Low(Col)) / (High(Col) – Low(Col))

4. Col_A = Col_B (for joins)
   RF = 1/ (Max (NKeys(Col_A), NKeys(Col_B)))

5. In no information about Nkeys or interval, use a “magic number” 1/10
   RF = 1/10

Let's try it..

Sailors (S): NTuples(S) =1000, Nkeys(rating) = 10 interval [1-10], age interval [0-100], Nkeys(sid)=1000

*SELECT * FROM Sailors WHERE rating = 3 AND age > 50;*

Calculate result size:

NTuples(S) = 1000

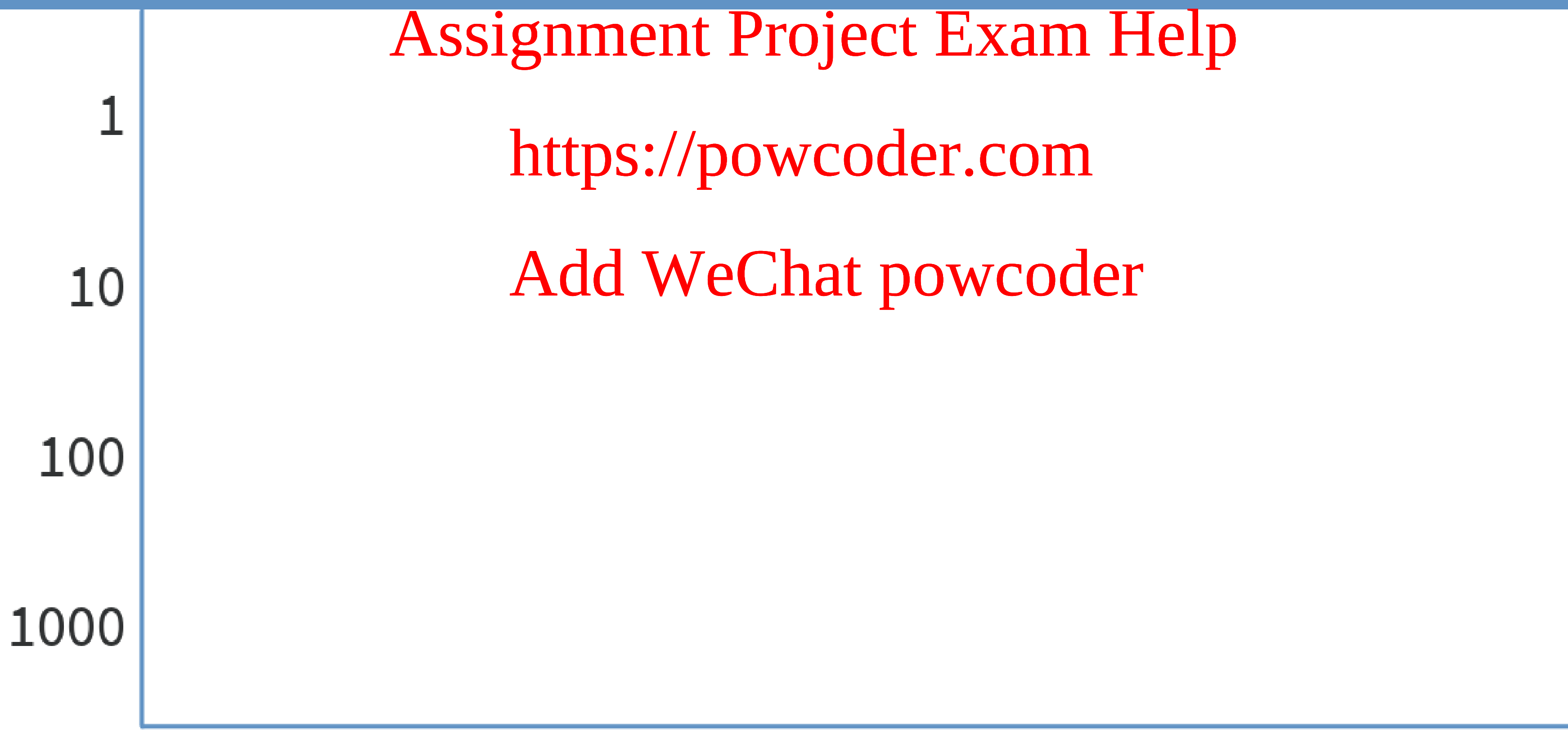
RF(rating) = 1/10 = 0.1

RF(age) = (100-50)/(100-0) = 0.5

ResultSize = NTuples(S)*RF(rating)*RF(age)

= 1000*0.1*0.5= 50 tuples

Given Reserves (R): NTuples(R) = 100, Nkeys(sid) =100 and Sailors (S): NTuples(S) =1000, Nkeys(rating) = 10 interval [1-10], age interval [0-100], Nkeys(sid)=1000 and the query: SELECT * FROM Sailors as S INNER JOIN Reserves as R ON S.SID = R.SID WHERE rating =8 and 20< age < 30; Calculate Result Size



What's examinable

- What is query optimization/describe steps?
- Equivalence classes
- Result size estimation

- Important for Assignment 3 as well

Next Lecture

- Query optimization Part II
  - Plan enumeration