

COMP 3311

DATABASE MANAGEMENT

SYSTEMS

LECTURE 18

QUERY OPTIMIZATION

CATALOG INFORMATION FOR COST ESTIMATION

- The DBMS **system catalog** (**data dictionary**) stores, for each relation r , the following statistics:

n_r the number of tuples in r .

B_r the number of pages containing tuples of r .

l_r the size of a tuple of r in bytes.

bf_r the blocking factor of r (i.e., the number of tuples of r that fit into one disk page).

$V(A, r)$ the number of **distinct values** that appear in r for attribute A .

➤ Equivalent to $\pi_A(r)$.

👉 If tuples of r are stored together physically in a file, then

$$B_r = \lceil n_r / bf_r \rceil.$$

CATALOG INFORMATION ABOUT INDEXES

- For indexes, the system catalog stores the following information.
 - HT_i the number of levels in the index i (i.e., the height of i).
 - For a balanced tree index (such as a B⁺-tree) on attribute A of relation r ,
 $HT_i = \lceil \log_{f_i}(V(A, r)) \rceil$.
 - where f_i is the average fan-out of the internal nodes of the index.
 - $\min f_i = \lceil n/2 \rceil$ and $\max f_i = n$ where n is the maximum fan-out of an index node.
 - For a hash index, HT_i is 1 (or 1.2 if there are overflow buckets).
 - LB_i the number of lowest-level index pages in i (i.e., the number of pages at the leaf level of the index).

STATISTICS ESTIMATION

The output size of an operation determines (i) the cost of the operation and (ii) the cost of subsequent operations.

➡ Its accurate estimation is important for optimization.

- Therefore, the statistics should be updated whenever a relation is updated.

➡ Can incur substantial overhead.

- Statistics are updated only periodically.

➡ They are likely to not be completely accurate.

SELECTION CARDINALITY AND SELECTIVITY

$SC(\theta, r)$, the selection cardinality of predicate θ for relation r , is the average number of tuples that satisfy the predicate θ .

$\text{Selectivity}(\theta, r) = SC(\theta, r) / n_r$ (i.e., the fraction of tuples that satisfy θ)

✎ Selectivity is a number between 0 and 1.

✎ The smaller the selectivity, the fewer tuples are selected.

SIZE ESTIMATION: EQUALITY SELECTION

Equality selection: $\sigma_{A=v}(r)$

$SC(A=v, r) = n_r / V(A, r) \Rightarrow$ number of r tuples / number of distinct A values

Selectivity($A=v, r$) = $1 / V(A, r) \Rightarrow$ 1 / number of distinct A values

✎ Assumes uniform distribution of values of A over r .

- $\lceil SC(A=v, r) / bf_r \rceil \Rightarrow$ the number of pages that these tuples will occupy if they are ordered on attribute A .
- If the tuples are not ordered on A , then each tuple may reside in a different page.

✎ For equality selection on a key attribute:

$SC(A=v, r) = 1$ (only 1 tuple will qualify)

Selectivity($A=v, r$) = $1 / n_r$

SIZE ESTIMATION: EQUALITY SELECTION EXAMPLE

Given *Sailor* with $n_{\text{Sailor}} = 40,000$ tuples in $B_{\text{Sailor}} = 500$ pages

```
select sailorId
from Sailor
where rating=8;
```

- If $V(\text{rating}, \text{Sailor}) = 10$ distinct ratings, then the condition $\text{rating}=8$ is expected to retrieve
 $SC(\text{rating}=8, \text{Sailor}) = n_{\text{Sailor}} / V(\text{rating}, \text{Sailor}) = 40,000 / 10 = 4,000$ tuples
 (i.e., $1/10$ of the tuples *assuming uniform distribution of rating values*).
- The estimated cost (page I/Os) to execute the query using
 - a **clustering index** on rating is $1/10 * 500 = 50$ page I/Os.
 - a **non-clustering index** on rating is 4000 page I/Os since *each tuple* is assumed to require a page I/O.
 - In both cases we do not consider the cost of accessing the index because it is negligible. The index contains only 10 entries since there are only 10 rating values.
 - a **file scan** is 500 page I/Os.
 - a **binary search** (provided that the file is sorted on rating) is

$$\begin{aligned} \lceil \log_2(B_r) \rceil + \lceil SC(A, r) / bf_r \rceil - 1 &= \lceil \log_2(500) \rceil + \lceil 4000/80 \rceil - 1 = 9 + 49 \\ &= 58 \text{ page I/Os.} \end{aligned}$$

SIZE ESTIMATION: NONEQUALITY SELECTIONS

Range selection: $\sigma_{A \leq v}(r)$
 (The case for $\sigma_{A \geq v}(r)$ is symmetric.)

- Assume that $\min(A, r)$ and $\max(A, r)$ are available in the catalog.

$SC(A \leq v, r) = 0$ if $v < \min(A, r)$ (since no tuples will qualify)

$SC(A \leq v, r) = n_r$ if $v \geq \max(A, r)$ (since all tuples will qualify)

$SC(A \leq v, r) = n_r * \frac{v - \min(A, r)}{\max(A, r) - \min(A, r)}$ (assumes uniform distribution of tuples over the values)

Example:

Suppose $\min(\text{rating})=1$ and $\max(\text{rating})=10$.

For each rating there are $40,000/10=4,000$ tuples.

Then $SC(\text{rating} \leq 2, \text{Sailor}) = n_{\text{Sailor}} * (2 - 1) / (10 - 1)$
 $= 40,000 * 1/9 = 4445$ tuples (actual value 8000).

```
select sailorId
from Sailor
where rating <= 2;
```

👉 Can get more accurate estimates using histograms.

HISTOGRAM

- The previous estimates assume that **each value of A has the same probability** (i.e., that the values of A are **uniformly distributed** over the tuples).

 **The uniformity assumption rarely holds in practice.**

- A **histogram assumes local uniformity** within each partition (but not global uniformity).

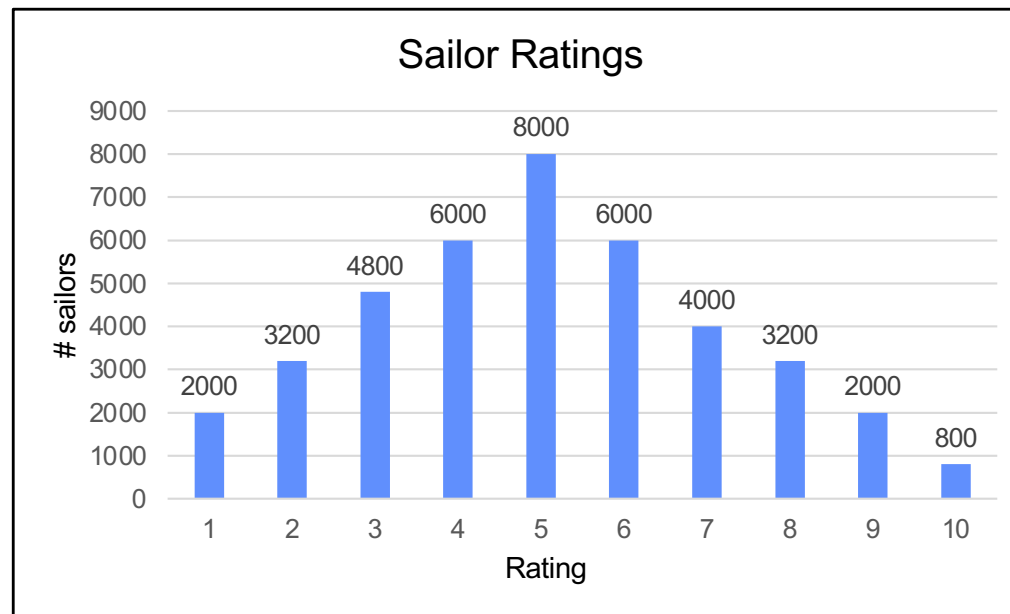
Equiwidth divide A values into subranges of **equal size** (e.g., equal rating ranges).

Equidepth divide A values into subranges such that the **number of tuples within each subrange is equal**.

HISTOGRAM (cont'd)

Example: Given the histogram below on sailor ratings, then
 $SC(\text{rating} \leq 2, \text{Sailor}) = 2000 + 3200 = 5200$.
(Compared to 4445 under the uniform distribution assumption.)

```
select sailorId  
from Sailor  
where rating <= 2;
```



ATTRIBUTE INDEPENDENCE

- The previous estimates assume that **values of attributes are independent**.

☞ **The attribute independence assumption rarely holds in practice.**

Example: Consider the **Sailor** relation and assume that the rating of a sailor increases with her experience. We have histograms on both **rating** and **age**. Furthermore, the number of sailors with age 20 and 50 are equal.

- The two queries below are estimated to have the **same SC**.

```
select *  
from Sailor  
where rating=10 and age=20;
```

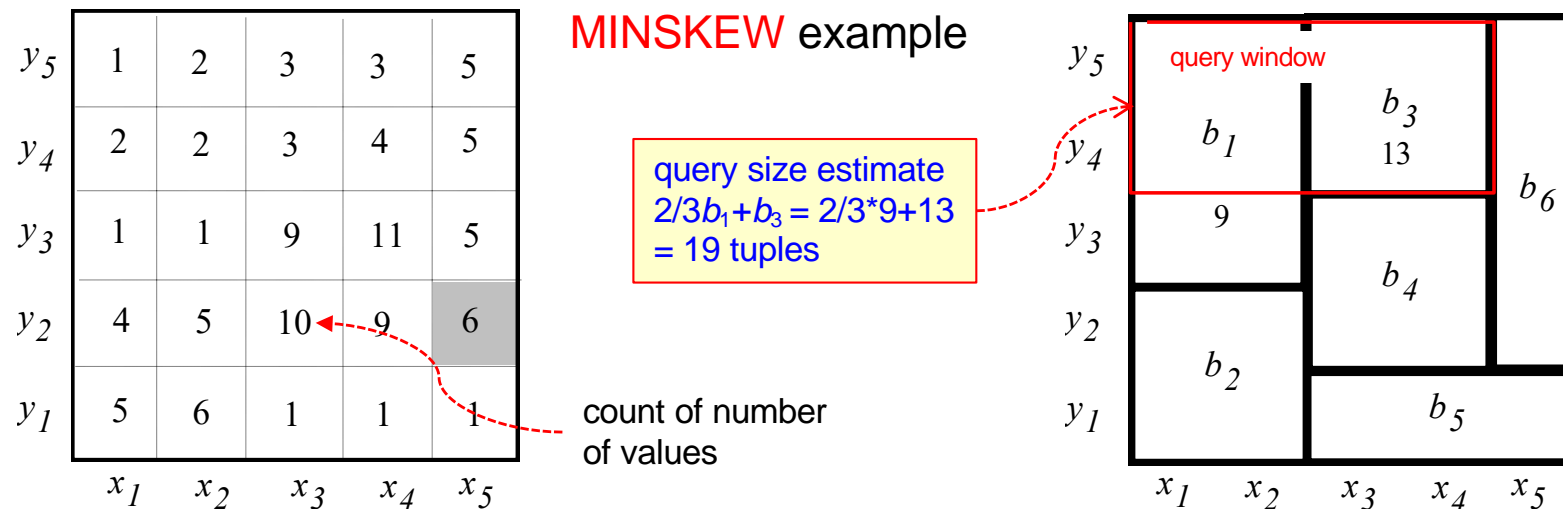
```
select *  
from Sailor  
where rating=10 and age=50;
```

- Which query is expected to retrieve more tuples?

Solution: Multidimensional histograms

MULTI-DIMENSIONAL HISTOGRAM

- Divide the space (e.g., *age-rating*) into partitions, so that **data values in each partition** have an *almost uniform distribution of values*.
- Keep in memory the **partition extents** (i.e., the ranges of values included in the partition) and the **number of tuples per partition**.
- Use this information to estimate the number of tuples in the **query window**.



SIZE ESTIMATION: COMPLEX SELECTIONS

Conjunction: $\sigma_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n}(r)$

The probability that a tuple satisfies the conjunctive condition is the **product of the selectivities** of the predicates. If the **selectivity** of θ_1 is $\frac{s_1}{n_r}$, θ_2 is $\frac{s_2}{n_r}$... θ_i is $\frac{s_i}{n_r}$, then

$$SC(\sigma_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n}, r) = n_r * \frac{s_1}{n_r} * \frac{s_2}{n_r} * \dots * \frac{s_n}{n_r} = n_r * \frac{s_1 * s_2 * \dots * s_n}{n_r^n}$$

Disjunction: $\sigma_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n}(r)$

The probability that a tuple satisfies the disjunctive condition is 1 minus the probability that it will satisfy *none* of the conditions. If the **selectivity** of θ_i is $\frac{s_i}{n_r}$, then $\bar{\theta}_i$ is $(1 - \frac{s_i}{n_r})$ and

$$SC(\sigma_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n}, r) = n_r * (1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) * \dots * (1 - \frac{s_n}{n_r}))$$

based on the logical equivalence: $\theta_1 \vee \theta_2 \vee \dots \vee \theta_n = \overline{\overline{\theta_1} \wedge \overline{\theta_2} \wedge \dots \wedge \overline{\theta_n}}$

SIZE ESTIMATION: JOIN

The Cartesian product $r \times s$ contains $n_r * n_s$ tuples.
Each tuple occupies $s_r + s_s$ bytes.

- If $r \cap s = \emptyset$
 - $r \bowtie s$ is the same as $r \times s$ (no common attributes).
 - If $r \cap s = A$ is a key for r
 - Each tuple of s joins with *at most one tuple* from r (since the join attributes are a key of r).
- ☞ The number of tuples in $r \bowtie s$ is no greater than number of tuples in s .

Example: In $Sailor \bowtie Reserves$, $sailorId$ in $Sailor$ is a key of $Sailor$.

☞ The result has exactly $n_{Reserves}$ tuples.

☞ If $sailorId$ in $Reserves$ can be null, then the number of tuples in the result will be *less than* the number in $Reserves$.

SIZE ESTIMATION: JOIN (cont'd)

The Cartesian product $r \times s$ contains $n_r * n_s$ tuples.
Each tuple occupies $s_r + s_s$ bytes.

- If $r \cap s = A$ is a (not null) foreign key in s referencing r
 - The number of tuples in $r \bowtie s$ is the same as the number of tuples in s (since the join attribute is a foreign key in s of r).
 - 👉 The case where $r \cap s$ is a foreign key in r referencing s is symmetric.

Example: Estimate the size result for $\text{Sailor} \bowtie \text{Reserves}$ where the join attribute, sailorId , in Reserves is a foreign key of Sailor .

Catalog information: $n_{\text{Sailor}} = 10,000$ $n_{\text{Reserves}} = 5,000$

- Since sailorId is a foreign key in Reserves , the size of the result is the same as $n_{\text{Reserves}} = \underline{5,000}$.

SIZE ESTIMATION: JOIN (cont'd)

The Cartesian product $r \times s$ contains $n_r * n_s$ tuples.
Each tuple occupies $s_r + s_s$ bytes.

- If $r \cap s = A$ is *not a key* for r or s
 - If we assume that every tuple in r produces $n_s / V(A, s)$ tuples in $r \bowtie s$, then the number of tuples in $r \bowtie s$ is estimated to be:

$$\frac{n_r * n_s}{V(A, s)}$$

- If instead we assume that every tuple in s produces $n_r / V(A, r)$ tuples in $r \bowtie s$, then the number of tuples in $r \bowtie s$ is estimated to be:

$$\frac{n_r * n_s}{V(A, r)}$$

- The **lower** of these two estimates is probably the more accurate one.

SIZE ESTIMATION: JOIN (cont'd)

Example: Estimate the result size of $\text{Sailor} \bowtie \text{Reserves}$ *without using information about foreign keys*.

Catalog information:

$$\begin{aligned} n_{\text{Sailor}} &= 10,000 & V(\text{sailorId}, \text{Sailor}) &= 10,000 & V(\text{sailorId}, \text{Reserves}) &= 2,500 \\ n_{\text{Reserves}} &= 5,000 & \Rightarrow & \text{only 2,500 sailors have reservations} \end{aligned}$$

- The two estimates are:

$$\frac{n_{\text{Sailor}} * n_{\text{Reserves}}}{V(\text{sailorId}, \text{Sailor})} = 10,000 * 5,000 / 10,000 = \underline{5,000}$$

$$\frac{n_{\text{Sailor}} * n_{\text{Reserves}}}{V(\text{sailorId}, \text{Reserves})} = 10,000 * 5,000 / 2,500 = \underline{20,000}$$

- We **choose the lower estimate**, which, in this case, is the same as our earlier computation using foreign keys.

SIZE ESTIMATION: OTHER OPERATIONS

Projection: The estimated size of $\pi_A(r) = V(A, r)$

Aggregation: The estimated size of group-by $A = V(A, r)$

Set operations

- For union / intersection of selections on the same relation, rewrite the query and use size estimate for selections.
 - E.g. $\sigma_{\theta_1}(r) \cup \sigma_{\theta_2}(r)$ can be rewritten as $\sigma_{\theta_1 \vee \theta_2}(r)$.
- For operations on different relations:
 - Estimated size of $r \cup s = \text{size of } r + \text{size of } s$.
 - Estimated size of $r \cap s = \text{minimum (size of } r, \text{ size of } s)$.
 - Estimated size of $r - s = r$.

 **All three estimates may be quite inaccurate, but provide upper bounds on the sizes.**

SIZE ESTIMATION: NUMBER OF DISTINCT VALUES

Selection: $\sigma_{\theta}(r)$

- If the selection condition θ is on attribute A , estimate as $V(A, \sigma_{\theta}(r)) = V(A, r) * s$ where s is the selection *selectivity*.

Example: $V(\text{rating}, \sigma_{\text{rating}=2}(\text{Sailor})) = 10/10 = 1$ (where s is $1/10$).

- In all other cases (e.g., selection on another attribute) use the rough estimate $\min(V(A, r), n_{\sigma_{\theta}(r)})$.

Example: $V(\text{age}, \sigma_{\text{rating}=8}(\text{Sailor})) =$
 $\min(\# \text{ different age values}, \# \text{ tuples in } \sigma_{\text{rating}=8}(\text{Sailor})).$

- Join $r \bowtie s$ estimate as $V(A, r \bowtie s) = \min(V(A, r), n_{r \bowtie s})$.

Example: $V(\text{sName}, \text{Sailor} \bowtie \text{Reserves}) =$
 $\min(V(\text{sName}, \text{Sailor}), \text{output size of join}).$

SIZE ESTIMATION: NUMBER OF DISTINCT VALUES (control)

Projection: $\pi_A(r)$

- The number of distinct values is the same as $V(A, r)$. The same holds for the grouping attributes of an aggregation.

Aggregated values

- For $\min(A)$, $\max(A)$, the number of distinct values can be estimated as $\min(V(A, r), V(G, r))$ where G denotes the grouping attributes.

Example:

```
select max(age)
from Sailor
group by rating;
```

$V(\max(\text{age}), \text{group by rating}) =$
 $\min(\# \text{different age values}, \# \text{different rating values}) =$
 $\# \text{different rating values}$ (i.e., for each rating we will
get a different $\max(\text{age})$ value).

- For other aggregates, assume all values are distinct and use $V(G, r)$.

Example:

```
select avg(age)
from Sailor
group by rating;
```

$V(\text{avg}(\text{age}), \text{group by rating}) =$
 $\# \text{different rating values}.$

QUERY OPTIMIZATION: SUMMARY

- Since **disk accesses slow query processing**, it is worthwhile to allocate a considerable amount of processing to **choose an execution strategy** that will **minimize disk accesses**.
- The **first step** is to **transform the user query into equivalent ones** using relational algebra equivalence rules.
- Next, **use statistics** about relations to **estimate the cost of each execution plan**.
 - 👉 **Histograms provide more accurate estimates.**
- Finally, use **cost estimates** and **heuristics** to **choose an execution plan**.