Query Processing and Optimization

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

1. Introduction
2. Physical-Query-Plan Operators
3. Query Optimization

Reference:
Chapter 13-14, Database System Concepts, Sixth Edition
Chapter 15, Database System, the Complete Book.

Introduction

table course(course_id, title, dept_name);
table instructor(ID, name, dept_name, salary);
table teaches(ID, course_id, sec_id, semester, year);

- Use the knowledge we have learnt this week to discuss:
  - What are the possible ways to answer the above SQL query?
  - Which one is better?

Introduction

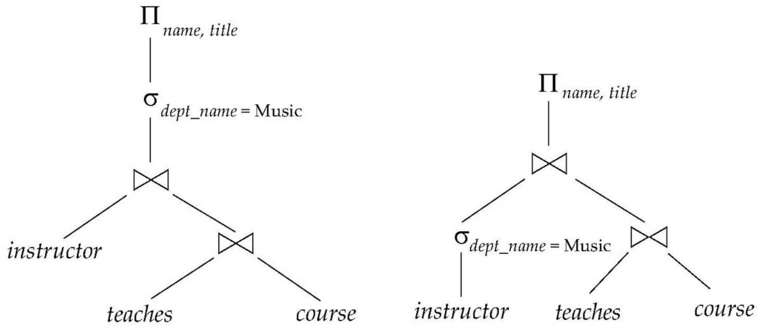
Relational Algebra

- $\Pi_{name,title}(\sigma_{dept\_name=Music}(instructor \bowtie (teaches \bowtie course))$
- $\Pi_{name,title}(\sigma_{dept\_name=Music}(instructor) \bowtie (teaches \bowtie course))$

Introduction

- $\Pi_{name,title}(\sigma_{dept\_name=Music}(instructor \bowtie (teaches \bowtie course))$
- $\Pi_{name,title}(\sigma_{dept\_name=Music}(instructor) \bowtie (teaches \bowtie course))$

Query plan (Logic plan):

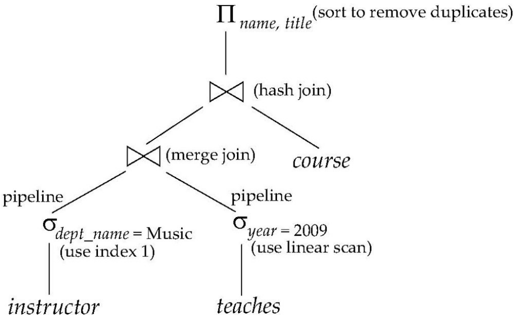


Introduction

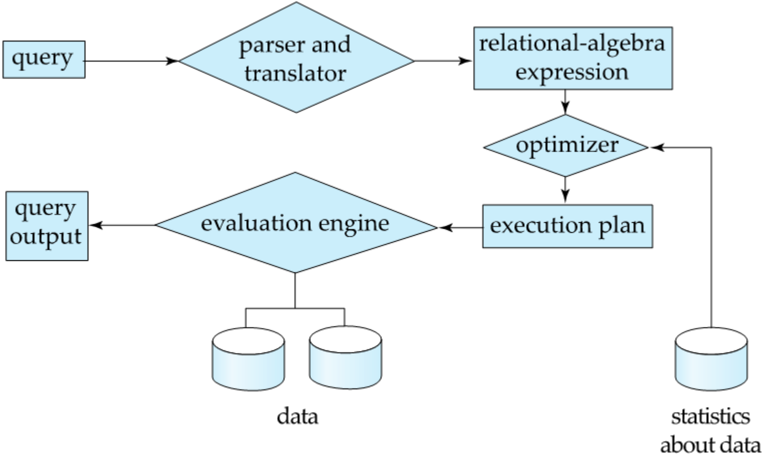
- $\Pi_{name,title}((\sigma_{dept\_name=Music}\, instructor \bowtie \sigma_{year=2009}\, teaches) \bowtie course)$

Physical/Evaluation plan

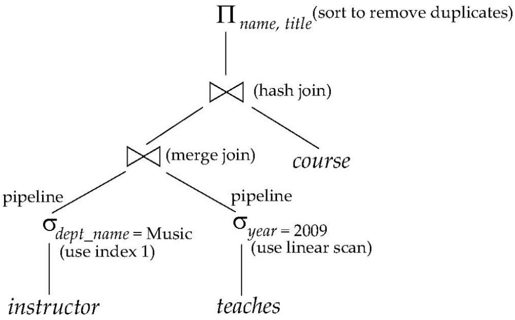
: an annotated expression on detailed evaluation strategy.



Introduction



Introduction



Cost estimation The cost of an operator is dependent on

- Metadata: The size of the underlying relation.
- Implementation: E.g., table scan, or using indexes (B+tree/hashing/$\cdots$).
- reporting: Whether the result needs to be stored on disk.

In this course, we assume that we don't need to store the query results on the disk.

Overview

1. Introduction
2. Physical-Query-Plan Operators
   - One-pass algorithms
   - Multi-pass algorithms
   - Index-based algorithms

Physical-Query-Plan Operators

External Memory Model

Denote by B the block size of the system. Let M be the size of the main memory. The cost of an algorithm is decided by the number of blocks (# of I/Os) transferred between the main memory and the disk.

Sometimes we use $m = M/B$ pages as the size of the main memory to simplify the analysis. In this scenario, the allocation of the memory is in pages.

Physical-Query-Plan Operators

Iterators:

```
Open() {
    b := the first block of R;
    t := the first tuple of block b;
}

GetNext() {
    IF (t is past the last tuple on block b) {
        increment b to the next block;
        IF (there is no next block)
            RETURN NotFound;
        ELSE /* b is a new block */
            t := first tuple on block b;
    } /* now we are ready to return t and increment */
    oldt := t;
    increment t to the next tuple of b;
    RETURN oldt;
}

Close() {
}
```

Physical-Query-Plan Operators

Iterators:

- Combines several operations into one
- Avoids writing temporary files
- Many iterators may be active at one time

Overview

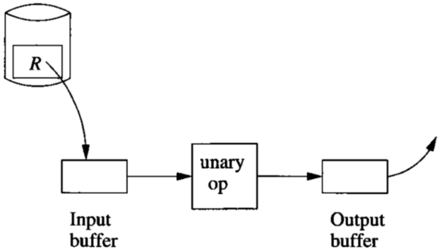
1. Introduction
2. Physical-Query-Plan Operators
   - One-pass algorithms
   - Multi-pass algorithms
   - Index-based algorithms

One-pass algorithms

- Tuple-at-a-time, unary operations: selection and projection
- Full-relation, unary operations: the grouping operator and duplication deduction operator, conditioned that the relation can fit in to the main memory.
- Full-relation, binary operations: unions, intersection, difference, join, product, conditioned that one of the relation fit into the main memory.

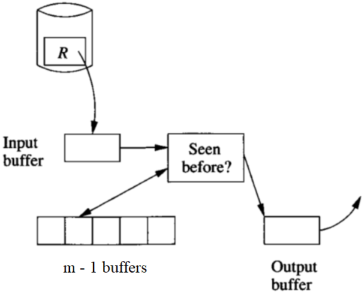
One-pass algorithms

- Tuple-at-a-time, unary operations: selection and projection



One-pass algorithms

- Tuple-at-a-time, unary operations: selection and projection
- Full-relation, unary operations: the grouping operator and duplication deduction operator, conditioned that the relation can fit in to the main memory.



One-pass algorithms

- Tuple-at-a-time, unary operations: selection and projection
- Full-relation, unary operations: the grouping operator and duplication deduction operator, conditioned that the relation can fit in to the main memory.
- Full-relation, binary operations: unions, intersection, difference, join, product, conditioned that one of the relation fit into the main memory.

Arrangement of the memory:

- $M - B - 1$: Hold one relation
- B: buffer one page of the other relation
- 1: buffer for the output

Overview

1. Introduction
2. Physical-Query-Plan Operators
   - One-pass algorithms
   - Multi-pass algorithms
     - Multi-way merge sort
     - Join with both relations larger than the memory size
   - Index-based algorithms

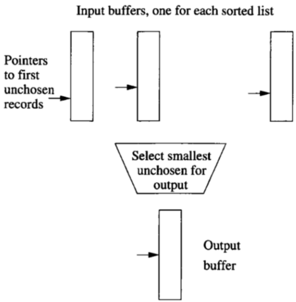
Multi-way Merge Sort

Sort-based algorithms:

- Duplication elimination
- Grouping and aggregation
- Union
- Intersection and difference

Multi-way Merge Sort

Atom operation: merge $m-1$ sorted list into one sorted list by allocating $m-1$ input buffer and 1 output buffer. The cost is linear to the size of the data.



Multi-way Merge Sort

Sort the data of n blocks on a machine with a memory of m blocks.

- Create runs (pass 0):
  - Each time load m blocks into memory, sort them and then output to create a run
- Merge pass i, $i = 1, 2, \cdots$: for every $m - 1$ runs created by pass $i - 1$
  - synchronize scan these runs to create a single run
  - one block used for output buffer

  until a pass creates only one run.

Complexity.

$O(n \log_{m-1} \frac{n}{m} + n) = O(n \log_m n)$ I/Os

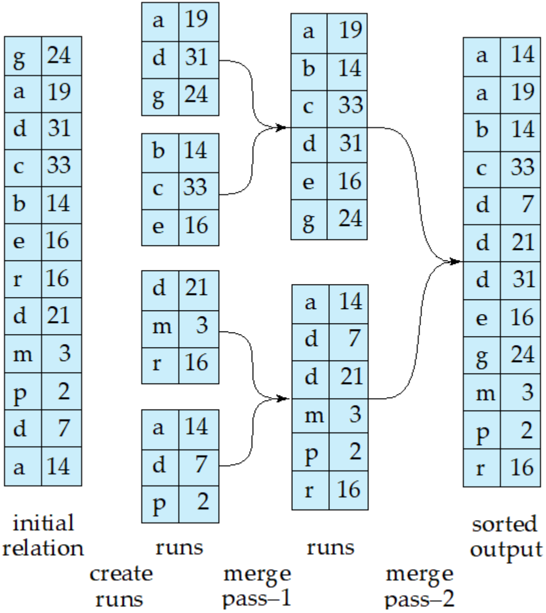


Figure: Example: $m = 3$, blocking factor $= 1$

Join

- Block nested-loop join
- Merge join
- Hash join

Block nested-loop join

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND | SCIENCE DEPARTMENT OF COMPUTER SCIENCE

$r \bowtie s$: r is stored in n_r blocks. s is stored in n_s blocks
Complexity: $\lceil n_r/(m-1) \rceil n_s + n_r$ I/Os.

Merge Join

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

$r \bowtie s$: r is stored in n_r blocks. s is stored in n_s blocks.
Join attribute: A.
Assume that r and s are both ordered A.

pr

a1	*a2*
a	3
b	1
d	8
d	13
f	7
m	5
q	6

r

ps

a1	*a3*
a	A
b	G
c	L
d	N
m	B

s

Complexity.

- Best case: $n_r + n_s$
- Worst case: block-nested-loop-join
- Consider the cost when
  - A is a key of r, or
  - all the attribute values are having the same frequency

Hashing: Partition a relation r to k buckets.

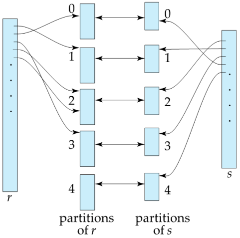
Consider a relation r with an attribute A and an integer $k \in [1, m)$;
Hash function $h_A()$ maps a tuple t to an integer in range $[0, k)$ based on its value on A.
Denote by r_i the i-th bucket of r.

Hash Join

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

$r \bowtie s$: r is stored in n_r blocks. s is stored in n_s blocks.
Join attribute: A.
Hash function $h_A()$ maps from a tuple to an integer range $[0, k)$ based on A.

1. Partition r to k buckets using hash function h_A
2. Partition s to k buckets using hash function h_A
3. For each integer $i \in [0, k)$
   - Join on r_i and s_i (one pass/multi pass)



Examples: Cost Estimation

Goal is to count disk I/Os. But we First have to estimate sizes of intermediate results?

- V(A,r) number of distinct values of attribute A in relation r.
- |r| number of tuples in relation r.

Consider relation $r(A, B)$ with $n_r = 1000$ blocks and relation $s(A, C)$ with $n_s = 500$ blocks. The memory had $m = 101$ pages.

- Nested-loop-join
- Multi-way merge join
- Hash join

Examples: Cost Estimation, Nested-Loop-Join

Consider relation $r(A, B)$ with $n_r = 1000$ blocks and relation $s(A, C)$ with $n_s = 500$ blocks. The memory had $m = 101$ pages.

- Outer-loop: load chunk of 100 blocks of s to the main memory

  5 chunks, 100 blocks each
- Inner-loop: scan r in 1000 blocks
- Total cost 5500 I/Os.

Switch the inner and outer loop:

- Outer-loop: load chunk of 100 blocks of r to the main memory 10 chunks, 100 blocks each
- Inner-loop: scan r in 500 blocks
- Total cost 6000 I/Os.

Conclusion: slight advantage in having the smaller relation on the outer loop.

Examples: Cost Estimation, Merge Join

Consider relation $r(A, B)$ with $n_r = 1000$ blocks and relation $s(A, C)$ with $n_s = 500$ blocks. The memory had $m = 101$ pages.

- The sorting cost of r: 4000 (two reads and two writes per block)
- The sorting cost of s: 2000 (two reads and two writes per block)
- Merge join: If A is the key of one relation, then the cost is 1500 blocks.
- Total cost 7500 I/Os.

Why linear time (when A is a key) merge join is not as good as quadratic time nested loop join in this case?

Examples: Cost Estimation, Hash Join

Consider relation $r(A, B)$ with $n_r = 1000$ blocks and relation $s(A, C)$ with $n_s = 500$ blocks. The memory had $m = 101$ pages. Let $k = 100$.

- The average size for each bucket is 10 blocks for relation r and 5 for s.
- Partition n and s (linear time): $1500 \times 2 = 3000$ I/Os
- Since for each $i \in [0, k)$, the buckets of r and s can altogether fit in main memory, one-pass join: 1500 I/Os for loading.
- Total cost: 4500 I/Os.

Overview

1. Introduction
2. Physical-Query-Plan Operators
   - One-pass algorithms
   - Multi-pass algorithms
   - Index-based algorithms

Index-based Algorithms

Application of Index in

- Selection
- Join

To avoid/reduce the number of table scan.

Index-based Algorithms: Cost Estimation Assumptions

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND | SCIENCE DEPARTMENT OF COMPUTER SCIENCE

We first have to estimate sizes of the intermediate results.

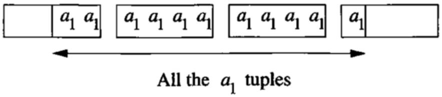
- $V(A, r)$ number of distinct values of attribute A in relation r.
- $|r|$ number of tuples in relation r.
- n_r number of blocks used to store relation r.

Important assumptions:

- Simple selection: All $V(A, r)$ values are equally likely for attribute A; in other words, $|\sigma_{A=c} r| = |r|/V(A, r)$.
- Selection involving inequality $|\sigma_{A<c} r|$: Common assumption that $1/3$ will meet the condition.
- Complex conditions AND: use decompositions.
  - $|\sigma_{A=c \text{ and } B<d} r| = |r|/3V(A, r)$.

Index-based Algorithms: Selection

Sequential file:



The actual cost of $\sigma_{A=v}r$ can be slightly larger than $n_r/V(A,r)$.

- The index is not kept entirely in main memory, some disk I/Os are need to support the index lookup.
- Even when all the tuples with $A = v$ might fit in b blocks, they could be spread over $b+1$ blocks because they don't start at the beginning of a block.
- The blocks can be not full, e.g., the B^+tree's leaf nodes can be not full.

Unordered file

- We assume that we need to visit $|r|/V(A,r)$ blocks for answering $\sigma_{A=v}r$.

Index-based Algorithms: Selection

Assume that for relation $r(A, B)$, $n_r = 1000$ and $|r| = 20,000$. Consider $\sigma_{A=0} r$. We ignore the cost accessing the index blocks in all cases.

- r is sequential on A but we do not use the index — 1000 I/Os.
- r is unordered on A but we use an index on A — 20,000 I/Os.
- r is sequential on A, $V(A, r) = 100$, use an index — 10 I/Os.
- r is unordered on A, $V(A, r) = 10$, use an index — 2000 I/Os.
- A is the key of r, use an index — 1 I/O.

Index-based Algorithms: Nested-Loop Join

$r(A, B) \bowtie s(A, C)$:

- r has $|r|$ tuples on n_r blocks, s is stored in n_s blocks
- s has an index on A;

Complexity: For each tuple t of r, an average of $|s|/V(A, s)$ tuples should be retrieved, the cost is dependent on s

- If s is sequential, for each t the cost is $n_s/V(A, s)$ I/Os;
- Otherwise, the cost is $|s|/V(A, s)$ I/Os.

Index-based Algorithms: Nested-Loop Join

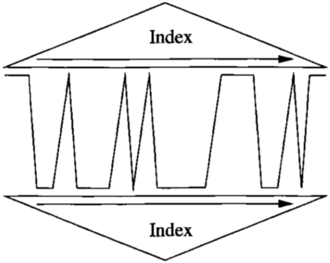
Example: Consider $r(A, B)$ with $n_r = 1000$ and $s(A, C)$ with $n_s = 500$ while ten tuples of either relation fit in one block. Therefore, $|r| = 10,000$ and $|s| = 5000$. Assume that $V(A, s) = 100$. Suppose that s is ordered on A and there is a clustering index of s on A. Compute the cost of $r \bowtie s$.

- The number of I/Os in accessing s is $10000 \times 500/100 = 50000$ I/Os.

If r is considerably smaller than s then the index-based nested-loop join would be better than nested-loop join.

Index-based Algorithms: Merge Join

Both r and s are sequentially stored, each having a clustered index on A.



Overview

1. Introduction
2. Physical-Query-Plan Operators
   - One-pass algorithms
   - Multi-pass algorithms
   - Index-based algorithms
3. Query Optimization

Query Optimization

To determine:

- What operators do we run, and in what order?
- For operators with several implementations(e.g. join), which one to use?
- How to read each table? — Index scan, table scan $\cdots$.

Query Optimization

Three steps of query optimization:

- Equivalent expressions generation
  - generate expressions based on equivalent rules
- Cost estimation with statistics
  - statistical information about relations:
    # of tuples, # of distinct values for an attribute
  - statistical estimation for intermediate results
  - cost computed for algorithms of join, selection, etc..
- Cost-based optimization
  - dynamic programming

Query Optimization

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND
SCIENCE DEPARTMENT OF COMPUTER SCIENCE

1. Equivalent expressions generation
2. Cost estimation with statistics
3. Cost-based optimization

Equivalent Expression Generation

- Let E_1 and E_2 be two relational algebra expressions. E_1 and E_2 are equivalent if they generate the same set of tuples on every *legal* database instance D.
- An equivalent rule states that expressions of two forms are equivalent.

We introduce 12 equivalent rules, respectively.

Equivalent Expression Generation

Equivalent rules.

1. Conjunctive selection operations can be decomposed into a sequence of selections: $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$
2. Selection operations are commutative: $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$
3. Only the out projection counts. $\Pi_{L_1}(\Pi_{L_2}(\cdots(\Pi_{L_n}(E))\cdots)) = \Pi_{L_1}(E)$
4. Selection can be combined with join/Cartesian products:
   - $\sigma_\theta E_1 \times E_2 = E_1 \times_\theta E_2$,
   - $\sigma_{\theta_2}(E_1 \bowtie_{\theta_1} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$
5. Join operations are commutative: $E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$
6. Join operations are associative:
   6.1 Natural join: $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$
   6.2 Theta join: $(E_1 \bowtie_{\theta_3} E_2) \bowtie_{\theta_1 \wedge \theta_2} E_3 = E_1 \bowtie_{\theta_2 \wedge \theta_3} (E_2 \bowtie_{\theta_1} E_3)$ where θ_1 Involves attributes from only E_2 and E_3. When any of them becomes empty, the Cartesian product is also associative.

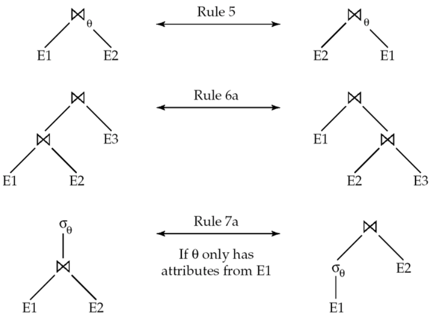
Equivalent Expression Generation

Equivalent rules.

7. Selection operations are distributive:
   - $\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta (E_2)$ if θ_0 involves only the attributes of E_1.
   - $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta \sigma_{\theta_2}(E_2)$ if θ_1 involves only the attributes of E_1 and θ_2 involve only the attributes of E_2.
8. Projection operations are distributive: Let L_1 and L_2 be subset of attributes of E_1 and E_2, respectively.
   - Suppose that join condition θ involves only attributes in $L_1 \cup L_2$: $\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1}(E_1) \bowtie_\theta \Pi_{L_2}(E_2)$.
   - Let L_3 (or L_4, resp.) be attributes of E_1 (or E_2, resp.) that are involved in join condition θ but not in $L_1 \cup L_2$: $\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}(\Pi_{L_1 \cup L_3}(E_1) \bowtie_\theta \Pi_{L_2 \cup L_4}(E_2))$

Equivalent Expression Generation

Equivalent rules.



Equivalent Expression Generation

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND | SCIENCE DEPARTMENT OF COMPUTER SCIENCE

Equivalent rules (set operations).

9. The set operations, union and intersection, are commutative.
   - union: $E_1 \cup E_2 = E_2 \cup E_1$
   - intersection: $E_1 \cap E_2 = E_2 \cap E_1$
10. The set operations, union and intersection, are associative.
    - union: $(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$
    - intersection: $(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$
11. The selection operations are distributive over union, intersection and set minus.
    - set minus: $\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$, and similarly for union and intersection.
12. Projection operations are distributive over union: $\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$

Equivalent Expression Generation

Heuristics: push selections.

Query: find the names of all instructors in the Music department, along with the titles of the courses that they teach.

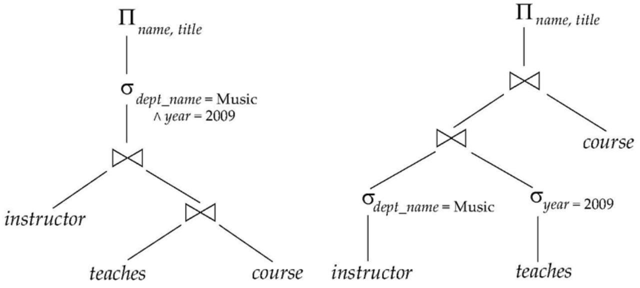
$$\Pi_{name,title}\sigma_{dept\_name=Music}(instructor \bowtie (teaches \bowtie \Pi_{course\_id,title}course))$$
$$\Rightarrow \Pi_{name,title}(\sigma_{dept\_name=Music}instructor) \bowtie (teaches \bowtie \Pi_{course\_id,title}course)$$

[rule 7.a]

Equivalent Expression Generation

Multiple transformations.

Use rule 6*a* (join associative) and then 7*a* (selection pushed):



(a) Initial expression tree (b) Tree after multiple transformations

Equivalent Expression Generation

Pushing projections.

$$
\begin{aligned}
&\Pi_{name,title}\sigma_{dept\_name=Music}(instructor \bowtie (teaches \bowtie \Pi_{course\_id,title}course)) \\
\Rightarrow &\Pi_{name,title}(\Pi_{name,course\_id}(\sigma_{dept\_name=Music}instructor) \bowtie teaches) \\
&\bowtie \Pi_{course\_id,title}course
\end{aligned}
$$

[rules 8a and 8b]

Equivalent Expression Generation

Generation. Let E be the given relational algebra expression.

Initialize set $S = \{E\}$.
The process that **systematically** generates the equivalent expressions:

- Let S' be the set of expressions that is equivalent to one expression $e \in S$ under one or more equivalence rules.
- If $S' \not\subseteq S$
  - $S \leftarrow S' \cup S$,
  - Repeat the process.

Now, for an expression E, we have all equivalent expressions under rules 1-12 included in set S. The next question is, which expression in S shall we choose?

Query Optimization

1. Equivalent expressions generation
2. Cost estimation with statistics
3. Cost-based optimization

Cost Estimation with Statistics

Use statistics to estimate the cost of each expression.

- statistical information about relations:

  # of tuples, # of distinct values for an attribute $\cdots$
- statistical estimation for intermediate results
- cost computed for algorithms of join, selection, etc..

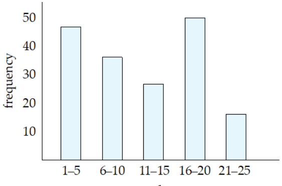
Cost Estimation with Statistics

We will use the following notations.

- n_r: the $\#$ of tuples in relation r.
- b_r: the $\#$ of blocks that contain tuples of r.
- l_r: size of a tuple of r.
- f_r: the blocking factor of r — $\#$ of tuples of r fit into a block.
- $V(A, r)$: the $\#$ of distinct values that appear in r for attribute A, that is, $|\Pi_A(r)|$.

If tuples of r are stored in a file, then $b_r = O(\frac{n_r}{f_r})$.

Histogram: a representation of the distribution of numerical data.



Cost Estimation with Statistics

The input of an operation can be the result of a sub-expression, therefore, we need to estimate the size of expression results.

To estimate the size of the resulting set of a selection.

- $$estimate(|\sigma_{A=v}(r)|) = \begin{cases} 1 & \text{if A is a key} \\ \frac{n_r}{V(A,r)} & \text{otherwise (assume a uniform distribution.)} \end{cases}$$
- $estimate(|\sigma_{A \leq v}(r)|)$
  $$= \begin{cases} 0 & \text{if } v < min(A,r) \\ n_r \frac{v - min(A,r)}{max(A,r) - min(A,r)} & \text{otherwise (assume a uniform distribution)} \end{cases}$$

  Refine this if histograms are available.

  In absence of statistical information, report $n_r/3$ [1].

[1] One may think that on average this number should be $n_r/2$, but there is an intuition that queries involving an inequality tend to retrieve a small fraction of the possible tuples.

Cost Estimation with Statistics

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND | SCIENCE DEPARTMENT OF COMPUTER SCIENCE

To estimate the size of the resulting set of a selection.

Denote by s_i the # of tuples in r that satisfy θ_i.

- Conjunctive selection $\sigma_{\theta_1 \wedge \theta_2 \wedge \cdots \wedge \theta_n}(r)$, assume independence.

$$estimate(|\sigma_{\theta_1 \wedge \theta_2 \wedge \cdots \wedge \theta_n}(r)|) = n_r \frac{s_1 \times s_2 \times \cdots \times s_n}{n_r^n}$$

- Negation $\sigma_{\neg\theta}(r)$: $estimate(|\sigma_{\neg\theta}(r)|) = n_r - size(\sigma_\theta(r))$
- Disjunctive selection $\sigma_{\theta_1 \vee \theta_2 \vee \cdots \vee \theta_n}(r)$, assume independence.

$$estimate(|\sigma_{\theta_1 \vee \theta_2 \vee \cdots \vee \theta_n}(r)|) = n_r - n_r(1 - \frac{s_1}{n_r})(1 - \frac{s_2}{n_r}) \cdots (1 - \frac{s_n}{n_r})$$

This is because:

$$\sigma_{\theta_1 \vee \theta_2 \vee \cdots \vee \theta_n}(r) = r - \sigma_{\neg\theta_1 \wedge \neg\theta_2 \wedge \cdots \wedge \neg\theta_n}(r)$$

Cost Estimation with Statistics

To estimate the size of the resulting set of join: consider a binary join on two relations r and s with schema R and S, respectively.

1. Cartesian product $r \times s$ or ($r \bowtie s$ with $R \cap S = \emptyset$).
   The resulting set has a size of $n_r \times n_s$.
2. If $R \cap S$ contains a key of R, then $|r \bowtie s| \leq n_s$: each tuple t in s will join with at most one tuple in r. More precisely, if $R \cap S$ contains a foreign key of S referencing R, then $|r \bowtie s| = n_s$.

Cost Estimation with Statistics

To estimate the size of the resulting set of join: consider a binary join on two relations r and s with schema R and S, respectively.

3. If $R \cap S$ does not contain any form of keys,
   - assume that a value in $\Pi_A(s)$ appears in the same total number of tuples in r: $estimate(|r \bowtie s|) = n_s \times \frac{n_r}{V(A,r)}$
   - assume that a value in $\Pi_A(r)$ appears in the same total number of tuples in s: $estimate(|r \bowtie s|) = n_r \times \frac{n_s}{V(A,s)}$

Cost Estimation with Statistics

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

To estimate the size of the resulting set of other operators.

Rule of thumb: use an upper bound.

4. Projection: $|\Pi_A(r)| = V(A, r)$.
5. Set operations:
   - $\mathit{estimate}(|r \cup s|) = n_r + n_s$
   - $\mathit{estimate}(|r \cap s|) = \min\{n_r, n_s\}$
   - $\mathit{estimate}(|r - s|) = n_r$

Cost Estimation with Statistics

To estimate the # of distinct values $V(A, \sigma_\theta(r))$ in selection $\sigma_\theta(r)$.
Rule of thumb: use an upper bound.

6. If θ forces A to take a specific set of k values, e.g., $k = 3, \theta : A = 3 \vee A = 5 \vee A = 7$
   $V(A, \sigma_\theta(r)) = k$.
7. If we know that $\sigma_\theta(r)$ has a selectivity of s', then

$$estimate(V(A, \sigma_\theta(r))) = V(A, r) \times s'.$$

8. In general cases, use the upper bound to estimate

$$estimate(V(A, \sigma_\theta(r))) = \min\{V(A, r), n_{\sigma_\theta(r)}\}.$$

Cost Estimation with Statistics

To estimate the # of distinct values $V(A, r \bowtie s)$ in join $r \bowtie s$.
Rule of thumb: use an upper bound.

9. If all attributes in A are from r, use the upper bound to estimate

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

10. If $A = A_1 \cup A_2$ consists of A_1 from r and A_2 from s, $estimate(V(A, r \bowtie s)) = \min\{V(A_1, r) \times V(A_2 - A_1, s), V(A_1 - A_2, r) \times V(A_2, s), n_{r \bowtie s}\}$.

All the costs can also be estimated using sampling.

Query Optimization

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

1. Equivalent expressions generation
2. Cost estimation with statistics
3. Cost-based optimization

Cost-Based Optimization

Practical query optimizers incorporate elements of the following two approaches.

- Exhaustive search: search all the plans and choose the best plan in a cost-based fashion.
- Heuristics: use heuristics to choose a plan.

Cost-Based Optimization

Exhaustive search (not examinable).
Consider: find the best join order for $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$. There are

$$\frac{(2(n-1))!}{(n-1)!}$$

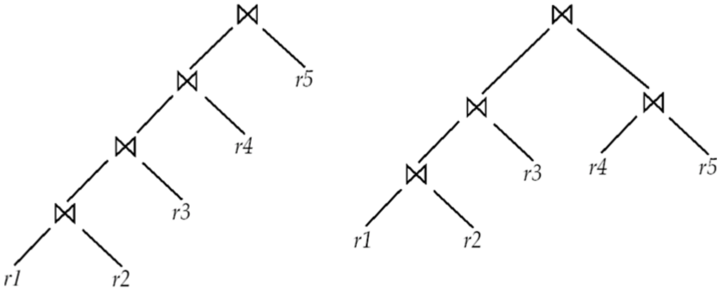
different join orders for the join expression. (Hint: based on Catalan number, the total number of binary trees with n nodes is $\frac{1}{n+1}\binom{2n}{n}$, that with n leaf node, would be $\frac{1}{n}\binom{2(n-1)}{(n-1)}$).

Example: when $n = 7$, the number is 665280, with $n = 10$, the number is ≥ 176 billion.
—This is prohibitive!
Luckily, the technique of dynamic programing can bring down the complexity to $O(3^n)$.

Cost-Based Optimization

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Pairwise join plan (bushy join tree) $\Rightarrow$ left deep join plan.



(a) Left-deep join tree (b) Non-left-deep join tree

Cost-Based Optimization

Other factors to be considered in the execution plan computation: pipeline and interesting sort orders.
Interesting sort order: a particular sort order of tuples that could be useful for a later operation.

One operation generates tuples in certain orders (interesting sort order) such that the tuples can be directed fed to another operator without an export (pipeline).
Example. Consider the join of $r_1 \bowtie r_2 \bowtie r_3$, denote by A the join attribute of r_1 and $(r_2 \bowtie r_3)$. A join process that is pipelined with two steps:

1. Perform a block nested loop join of r_2 and r_3 such that the joined tuple will be generated in non-decreasing order of their values on A;
2. Use the result of step 1 directly to perform a merge join with r_1.

Since the intermediate result $r_2 \bowtie r_3$ is not exported, we've saved the cost.

Cost-Based Optimization

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

To find the best execution plan of a relational algebra expression E, we

- Focus on the set of equivalent expressions of E.
- Estimate the cost of a plan with statistical informations of each relation/attribute.
- Optimize the search for the best execution plan.

Cost-Based Optimization

Heuristics: to reduce the size of the intermediate results, we

- Perform selections early.
- Perform projections early.
- Perform most restrictive selection and join operations before other similar operations.

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE

Thank you for your attention!

Any questions?