Chapter 3 Query-Optimization

Chapter 3.1 Algebraic Optimization

if < transformation-condition > **then**

$A \rightarrow B$

Expression A can be replaced by B

$A \equiv B$

Properties and their Inheritance:

R^e: Rel. or result unique, i.e. contains no duplicates, for strict rel. Algebra always holds per definition.

$R^{[A, B, ... C]}$: Rel. or result is sorted by Attr. A, B, ..., C with sort criteria in this sequence. Theoretically secondary issue,since relations are sets.

R^d: Rel. may contain duplicates (in Bag-representation), not necessarily

Inheritance: in rel. Algebra

$$\pi(R^e) \rightarrow (\pi\ (R))^e$$
$$\pi_{A...B}\ R^{[A...B...C]} \rightarrow (\pi_{A...B}\ R)^{[A...B]}$$
$$\sigma_P(R^e) \rightarrow (\sigma_P\ (R))^e$$
$$\sigma_P R^{[A...B]} \rightarrow (\sigma_P R)^{[A...B]}$$
$$R^{[A...B]} \times S^{[C...D]} \rightarrow (R \times S)^e$$
$$R^e \times S^e \rightarrow (R \times S)^e$$
$$R^e \cup S^e \rightarrow (R \cup S)^e \quad ; \text{e expensive}$$
$$R^{[A...B]} \cup S^{[A...B]} \rightarrow (R \cup S)^{[A...B]} \quad ; \text{cheap}$$
$$R^{[A...B]} \cup S^{[C...D]} \rightarrow (R \cup S)^e \quad ; \text{e expensive}$$

Goal of Optimization: short execution time

Heuristics: small intermediate results

Usual tradeoff "storage for time" is not valid here, since:

small storage requirement

$\Rightarrow$ result fits completely in AS or less I/0

$\Rightarrow$ faster

Bottom Line: Algebraic optimization requires cost model about size of relations (is in Systemtable) and size of expected intermediate results.

Chapter 3.2 Non-algebraic Optimization

Chapter 3.2.1 Complexity of relational Algebra

naive Algorithms

Let |R| = n |S| = m

$\sigma_P R$: $\mathcal{O}$ (n)

$\pi_{A...B} R$: $\mathcal{O}$ (n^2) resp. $\mathcal{O}$ (n log n)

$\mathcal{O}$ (n) if key or key candidate $\subseteq$ A...B

R x S : $\mathcal{O}$ (n · m) and many special cases

Additional Operations:

δ : eliminate duplicates

$\delta(R) = R^e$; $\delta(R^e) = R^e$

$\omega^{[A...B]}$: sort by order [A...B]

$\omega^{[A...B]}(R) = R^{[A...B]}$

Problem: Ensuring uniqueness (duplicate-elimination), in general expensive, requires sorting!

$\mathcal{O}$ (|R| log |R|)

Goal: Reduction of time complexity for the entire operator tree

Auxiliary structures: Indexes

- on 1 Attribute
- on several Attrib. A...B
- universal Index: UB-Tree
  hB-Tree
- R-Trees
- Hash-Indexes

Primary Index: on key attributes

- clustered: records in Index-order
- unclustered: records not in Index-order

Secondary Index: Index-Attributes do not contain key candiate

$\Rightarrow$ i. e. several records per Index-key, always unclustered

Strategy for Optimization

1. Algebraic Optimization: small intermediate results
   $\Rightarrow$ Operator-Tree with relational Algebra

2. Cost model for evaluation? On level of rel. Alg. not reasonable
   - applicable algorithms depend on type of relation, auxiliary structures
   - cost depends on selected algorithms
   - problem coupled with 1

Consequence: Needed is extended algebra suitable for implementation:

50-100 different operators

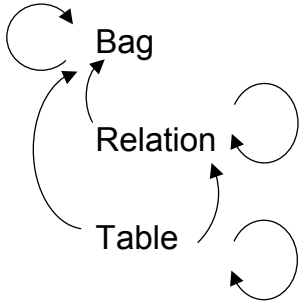
Optimization and cost model very difficult and uncertain!!

Chapter 3.2.2 Relations, Bags, Tables

Isa-Hierarchy:

Bag
|
Relations
|
Table

Mixed Algebra: **1. Projektion**



$\overline{\pi}_{A...B}$

A...B contains key candidate.

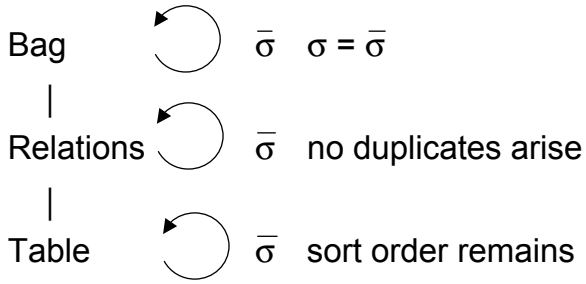
A...B contains sort attribute of table

$\overline{\pi}_{A...B}$: tuplewise projection by removal of Attributes without duplicate-check

$$\pi = \delta\overline{\pi}$$

$\Rightarrow$ in algebraically optimized operator tree split π into $\delta\overline{\pi}$ and delay δ.

2. Selection

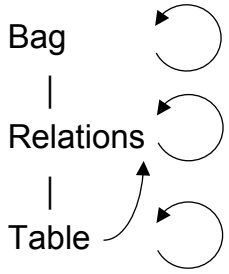


$\sigma = \bar{\sigma}$

no duplicates arise

sort order remains

3. Joins



no duplicates arise

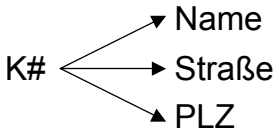
frequent special case:

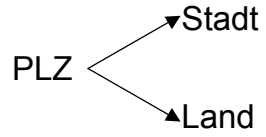
$R[A...B] \quad \iota_{R.F = S.F} \quad S$

where F is key candidate in S
and F is foreign key in R,
i. e. for all n : 1 relationships with standard rel. modelling.

Example: from DBS I:

KNADR = (K#, Name, Straße, Stadt, Land, PLZ)





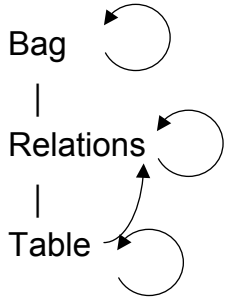
Kunden: sorted

Postleitzahlen:

Fremdschlüssel PLZ

Join preserves sortorder of Kunden, i.e. result is a table.

4. Division $\overline{\div}$



in general, some attributes disappear

Special condition?

Example for: Rel. with $\overline{\div}$

Preise = (**K#, T#,** Preis)

Teile = (T#, ...)

Query: Kunden #, which provide all parts independent of Preis.

let $Preise^{[T\#, P, K\#]}$

$\pi_{T\#, K\#}$ (Preise) $\overline{\div}$ $\pi_{T\#}$ (Teile) destroys sort order

Example for: Tab 

let $Preise^{[K\#, T\#, P]}$

$\pi_{T\#, K\#}$ (Preise) $\overline{\div}$ $\pi_{T\#}$ (Teile) preserves sorting according to K# of Preise

Condition for Tab 

R ÷ S

R

S

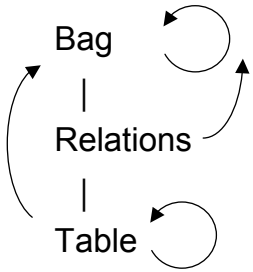
let $R^{[A, F]}$, F foreign key

÷ very simple, $\boldsymbol{O}$ (|R|)

in general: Let R = (A ... B F C ... D)
S = (F C ... D) and
F is key candidate of S

sort R according to A ... B F;
S according to F;
modified "nested loop" algorithm
preserves order of R and has complexity $\boldsymbol{O}$ (|R|)

5. Union: $\overline{\cup} : \cup = \delta \overline{\cup}$



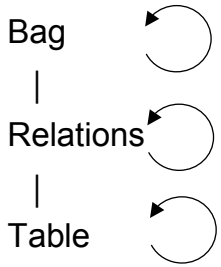
$\overline{\cup}$ is concatenation

$\mathcal{O}$

if both tables are sorted in the same way

Cost: $\mathcal{O}$ (|R| + |S|) or $\mathcal{O}$ (1) for general list in all cases

6. Intersection: $\overline{\cap}$



for all r ∈ R **do**
  if r ∈ B **then**
    r ∈ R $\overline{\cap}$ B

$\overline{\cap}$	B	R	T
B	B	B	B
R	R	R	R
T	T	T	T

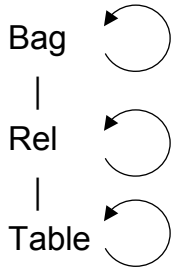
⇒ it depends only on 1st operand

Complexity of $\overline{\cap}$: depends on test r ∈ B

$\overline{\cap}$	B_2	R_2	T_2
B_1	O ($\|B_1\| * \|B_2\|$)	O ($\|B_1\| * \|R_2\|$)	O ($\|B_1\| * \log \|T_2\|$)
R_1	O ($\|R_1\| * \|B_2\|$)	O ($\|R_1\| * \|R_2\|$)	O ($\|R_1\| * \log \|T_2\|$)
T_1	O ($\|T_1\| * \|B_2\|$)	O ($\|T_1\| * \|R_2\|$)	O ($\|T_1\| * \log \|T_2\|$)

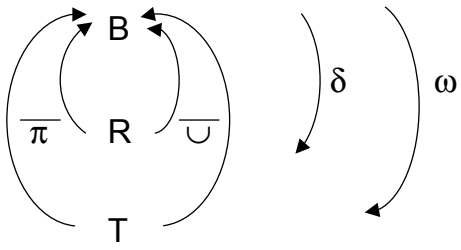
Question: When is $|T_1|*\log|T_2| < |T_2|*\log|T_1|$?
Special case: T_1, T_2 sorted equally?
Does sorting pay?

7. Differenz: $\overline{\setminus}$



algorithm and complexity like $\overline{\cap}$

Problem Cases and Type Conversion:



cost of δ: $\boldsymbol{O}$ (n log n)

cost of ω: $\boldsymbol{O}$ (n log n)

⇒ it is better, to evaluate ω directly
perform ω instead of δ

Comparison of Complexity: see also chapter 3.2.1

π : $\boldsymbol{O}$ (n log n) $\overline{\pi}$: $\boldsymbol{O}$ (n)

∪ : $\boldsymbol{O}$ (n log n) $\overline{\cup}$: $\boldsymbol{O}$ (n)

σ_P : $\boldsymbol{O}$ (n) or with Index: $\boldsymbol{O}$ (log n)

⋈ : $\boldsymbol{O}$ (n · m) nested loop
$\boldsymbol{O}$ (n + m) if 1 Rel fits in AS
$\boldsymbol{O}$ (n · log n) + $\boldsymbol{O}$ (m log m) + $\boldsymbol{O}$ (r) with sort-merge and r is size of result

Goal conflict: tables lower costs of other operators, but are more expensive to generate

⇒ not a local problem, optimization of the entire operator tree

Limits of Optimization

1. Many Operator Trees:

for j operators, exponential number of trees, e.g.:
for j Joins > 2^j Join sequences in addition to exchanges with π, σ?

⇒ for 30 operators 2^{30} **>** 1 Mrd. semantically equivalent trees.

2. Cost Model

Estimatoin of size of intermediate results necessary, but uncertain i. e. expected values?

$\Rightarrow$ - strict, complete optimization unrealistic
- good general algorithms
- heuristics for optimization

Selectivity of Predicates:

$|\sigma_P R| \approx |R| \cdot p\%$

Number of Join Partners:

$|R \rhd\lhd S| \approx |R| \cdot j$

if every $r \in R$ has j Join Partners from S in the average.