Administrivia

- HW5 is due Thursday March 17th.
- You can pick up your mid-term from Marilyn Walgora’s office (GHC 8120).
- Christos is out of the country.
- No office hours next week.

Today’s Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration

Cost-based Query Sub-System

Queries

```
Select *
From Blah B
Where B.blah = blah
```
Query Optimization

- Remember that SQL is declarative.
  – User tells the DBMS *what* answer they want, not *how* to get the answer.
- There can be a big difference in performance based on plan is used:
  – See last week: 5.7 days vs. 45 seconds

Quick DB History Lesson

1960s – IBM IMS

- First database system.
- Hierarchical data model.
- Programmer-defined physical storage format.
- Tuple-at-a-time queries.

Hierarchical Data Model

**Schema**

- **SUPPLIER**
  - (sno, sname, scity, sstate)
- **PART**
  - (pno, pname, psize, qty, price)

**Instance**

- **SUPPLIER**
  - 3, “Dirty Rick’s Supplies”, New York, NY
- **PART**
  - 1001, “Battery Pack”, Large, 500, $100
Hierarchical Data Model

1970s – CODASYL

• COBOL people got together and proposed a standard based on a network data model.
• Tuple-at-a-time queries.
  – This forces the programmer to do manual query optimization.

Network Data Model

Complex Queries
1970s – Relational Model

- Ted Codd saw the maintenance overhead for IMS/Codasyl.
- Proposed database abstraction based on relations:
  - Store database in simple data structures.
  - Access it through high-level language.
  - Physical storage left up to implementation.

IBM System R

- Skunkworks project at IBM Research in San Jose to implement Codd’s ideas.
- Had to figure out all of the things that we are discussing in this course themselves.
- IBM never commercialized System R.

IBM System R

- First implementation of a query optimizer.
- People argued that the DBMS could never choose a query plan better than what a human could write.
- A lot of the concepts from System R’s optimizer are still used today.

Today’s Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration
- Nested Sub-queries
A query can be expressed in different ways.
The optimizer considers variations and choose the one with the lowest cost.
How do we know whether two queries are equivalent?
– Equivalence Rules (chapter 15.3)

Two relational algebra expressions are equivalent if they generate the same set of tuples.

\[
\pi_{\text{cname}, \text{amt}}(\sigma_{\text{amt}>1000}(\text{customer} \bowtie \text{account})) = \pi_{\text{cname}, \text{amt}}(\text{customer} \bowtie (\sigma_{\text{amt}>1000} \text{account}))
\]
Equivalence of Expressions

• Q: How to prove a transf. rule?
\[ \sigma_p(R_1 \bowtie R_2) = \sigma_p(R_1) \bowtie \sigma_p(R_2) \]

• Use relational tuple calculus to show that LHS = RHS:
\[ \sigma_p(R_1 \cup R_2) = \sigma_p(R_1) \cup \sigma_p(R_2) \]

LHS \quad RHS

Faloutsos/Pavlo
CMU SCS 15-415/615
21

QED

Equivalence of Expressions

- Q: How to disprove a rule?
\[ \pi_A(R_1 - R_2) = \sigma_p(R_1) - \sigma_p(R_2) \]

R1
\[ t \in LHS \iff t \in (R_1 \cup R_2) \land P(t) \iff (t \in R_1 \lor t \in R_2) \land P(t) \iff (t \in R_1 \land P(t)) \lor (t \in R_2 \land P(t)) \]

R2

Faloutsos/Pavlo
CMU SCS 15-415/615
24


Equivalence of Expressions

- **Selections:**
  - Perform them early
  - Break a complex predicate, and push
  
  \[ \sigma_{p_1 \land p_2 \land \ldots \land p_n}(R) = \sigma_{p_1}(\sigma_{p_2}(\ldots \sigma_{p_n}(R)) \ldots) \]

- **Simplify a complex predicate**
  - \((X=Y \land Y=3) \rightarrow X=3 \land Y=3\)

Faloutsos/Pavlo

Equivalence of Expressions

- **Projections:**
  - Perform them early
  - Smaller tuples
  - Fewer tuples (if duplicates are eliminated)
  - Project out all attributes except the ones requested or required (e.g., joining attr.)

Faloutsos/Pavlo

Equivalence of Expressions

- **Joins:**
  - Commutative, associative
    
    \[ R \bowtie S = S \bowtie R \]
    
    \[ (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T) \]

  - Q: How many different orderings are there for an \(n\)-way join?
    
    A: Catalan number \(~ 4^n\)
    
    - Exhaustive enumeration: too slow.
    
    - We’ll see in a second how an optimizer limits the search space...

Faloutsos/Pavlo
Today’s Class

• History & Background
• Relational Algebra Equivalences
• Plan Cost Estimation
• Plan Enumeration

Query Optimization

• Bring query in internal form (e.g., parse tree)
• … into “canonical form” (syntactic q-opt)
• Generate alternative plans.
• Estimate cost for each plan.
• Pick the best one.

Cost Estimation

• How long will a query take?
  – CPU: Small cost; tough to estimate.
  – Disk: # of block transfers.
  – Network: # of messages
• How many tuples will qualify?
• What statistics do we need to keep?

Cost Estimation – Statistics

• For each relation $R$ we keep:
  – $N_R$ → # tuples
  – $S_R$ → size of tuple in bytes
  – $V(A,R)$ → # of distinct values of attribute ‘A’
Derivable Statistics

- $F_R \rightarrow$ max # records/block
- $B_R \rightarrow$ # blocks
- $SC(A,R) \rightarrow$ selection cardinality
  \hspace{1cm} \text{avg # of records with } A = \text{given}

Note that this assumes data uniformity
- 10,000 students, 10 colleges – how many students in SCS?

Additional Statistics

- For index $i$:
  - $F_i \rightarrow$ average fanout (≈50-100)
  - $HT_i \rightarrow$ # levels of index $i$ (≈2-3)
    \hspace{1cm} \sim \log(#\text{entries})/\log(F_i)$
  - $LB_i \# \rightarrow$ blocks at leaf level

Statistics

- Where do we store them?
- How often do we update them?
Selection Statistics

- We saw simple predicates (name="Christos")
- How about more complex predicates, like
  - salary > 10000
  - age=30 AND jobTitle="Costermonger"
- What is their selectivity?

Selections – Complex Predicates

- Selectivity $\text{sel}(P)$ of predicate $P$:
  \[ \text{sel}(P) = \frac{\text{SC}(P)}{N_R} \]

Selection Cardinality

# of tuples

Selections – Complex Predicates

- Assume that $V(\text{rating}, \text{SAILORS})$ has 5 distinct values (i.e., 0 to 4).
- simple predicate $P$: $A=\text{constant}$
  - $\text{sel}(A=\text{constant}) = \frac{1}{V(A,R)}$
  - eg., $\text{sel}(\text{rating}=\text{‘2’}) = \frac{1}{5}$
- What if $V(A,R)$ is unknown??

Range Query: $\text{sel}(\text{rating} \geq \text{‘2’})$

\[ \text{sel}(A>a) = \frac{(A_{\text{max}} - a)}{(A_{\text{max}} - A_{\text{min}})} \]
Selections – Complex Predicates

• Negation: \( \text{sel}(\text{rating} \neq '2') \)
  – \( \text{sel}(\text{not } P) = 1 - \text{sel}(P) \)

• Observation: selectivity \( \approx \) probability

![Graph showing selectivity vs. probability](image)

Selections – Complex Predicates

• Conjunction:
  – \( \text{sel}(\text{rating} = '2' \text{ and } \text{name LIKE} 'C\%') \)
  – \( \text{sel}(P1 \land P2) = \text{sel}(P1) \cdot \text{sel}(P2) \)
  – INDEPENDENCE ASSUMPTION

![Venn diagram showing independence](image)

Selections – Complex Predicates

• Disjunction:
  – \( \text{sel}(\text{rating} = '2' \text{ or } \text{name LIKE} 'C\%') \)
  – \( \text{sel}(P1 \lor P2) = \text{sel}(P1) + \text{sel}(P2) - \text{sel}(P1 \lor P2) \)
  – \( \text{INDEPENDENCE ASSUMPTION, again} \)

![Venn diagram showing disjunction](image)

Selections – Complex Predicates

• Disjunction, in general:
  – \( \text{sel}(P1 \text{ or } P2 \text{ or } \ldots \text{ Pn}) = \)
  – \( 1 - (1 - \text{sel}(P1)) \cdot (1 - \text{sel}(P2)) \cdot \ldots \cdot (1 - \text{sel}(Pn)) \)

![General disjunction Venn diagram](image)
Selections – Summary

- \(sel(A=\text{constant}) \rightarrow 1/V(A,r)\)
- \(sel(A>a) \rightarrow (A_{\text{max}} - a) / (A_{\text{max}} - A_{\text{min}})\)
- \(sel(\text{not } P) \rightarrow 1 - sel(P)\)
- \(sel(P_1 \text{ and } P_2) \rightarrow sel(P_1) \cdot sel(P_2)\)
- \(sel(P_1 \text{ or } P_2) \rightarrow sel(P_1) + sel(P_2) - sel(P_1) \cdot sel(P_2)\)
- \(sel(P_1 \text{ or } \ldots \text{ or } P_n) = 1 - (1-sel(P_1)) \cdot \ldots \cdot (1-sel(P_n))\)

Joins

- Q: Given a join of \(R\) and \(S\), what is the range of possible result sizes in #of tuples?
  - Hint: what if \(R_{\text{cols}} \cap S_{\text{cols}} = \emptyset\)?
  - \(R_{\text{cols}} \cap S_{\text{cols}}\) is a key for \(R\) and a foreign key in \(S\)?

Result Size Estimation for Joins

- General case: \(R_{\text{cols}} \cap S_{\text{cols}} = \{A\}\) where \(A\) is not a key for either table.
  - \(Hint: for a given tuple of \(R\), how many tuples of \(S\) will it match?\)
  - \(N_R \cdot N_S \leq N_S\)
Result Size Estimation for Joins

- General case: \( R_{\text{cols}} \cap S_{\text{cols}} = \{ A \} \) where \( A \) is not a key for either table.
  - Match each \( R \)-tuple with \( S \)-tuples:
    \[
    \text{estSize} \approx \frac{N_R \cdot N_S}{V(A,S)}
    \]
  - Symmetrically, for \( S \):
    \[
    \text{estSize} \approx \frac{N_R \cdot N_S}{V(A,R)}
    \]
- Overall:
  \[
  \text{estSize} \approx \frac{N_R \cdot N_S}{\max(\{ V(A,S), V(A,R) \})}
  \]
Today’s Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration

Query Optimization

- Bring query in internal form (e.g., parse tree)
- ... into “canonical form” (syntactic q-opt)
- Generate alternative plans.
  - Single relation.
  - Multiple relations.
- Estimate cost for each plan.
- Pick the best one.

Plan Generation

- What are our plan options?

- Sequential Scan
- Binary Search
  - if sorted & consecutive
- Index Search
  - if an index exists
### Sequential Scan

- $B_R$ (worst case)
- $B_R/2$ (on average, if we search for primary key)

### Binary Search

- $\sim \log(B_R) + \frac{SC(A,R)}{F_R}$
- Extra blocks are ones that contain qualifying tuples

### Index Search

- Index Search:
  - levels of index + blocks w/ qual. tuples

Case#1: Primary Key
Case#2: Secondary key – clustering index
Case#3: Secondary key – non-clust. index

---

We showed that estimating this is non-trivial.
Index Search: Case #1

• Primary Key
  – cost: \( HT_i + 1 \)

Index Search: Case #2

• Secondary key with clustering index:
  – cost: \( HT_i + \frac{SC(A,R)}{F_R} \)

Index Search: Case #3

• Secondary key with non-clustering index:
  – cost: \( HT_i + SC(A,R) \)

Query Optimization

• Bring query in internal form (e.g., parse tree)
• … into “canonical form” (syntactic q-opt)
• Generate alternative plans.
  – Single relation.
  – Multiple relations.
• Estimate cost for each plan.
• Pick the best one.
Queries over Multiple Relations

- As number of joins increases, number of alternative plans grows rapidly
  - We need to restrict search space.

- **Fundamental decision in System R:** only left-deep join trees are considered.

- **Fundamental decision in System R:** only left-deep join trees are considered.

  - Allows for fully pipelined plans where intermediate results not written to temp files.
  - Not all left-deep trees are fully pipelined (e.g., SM join).
Queries over Multiple Relations

- Enumerate the orderings (= left deep tree)
- Enumerate the plans for each operator
- Enumerate the access paths for each table
- Use **dynamic programming** to save cost estimations.

Dynamic Programming Example

Assumption: NO package deals: cost CDG->PVG is always $800, no matter how reached CDG

Solution: compute partial optimal, left-to-right:
Solution: compute partial optimal, left-to-right:

So, best price is $1,500 – which legs?
So, best price is $1,500 – which legs?
A: follow the winning edges, backwards

Q: what are the states, costs and arrows, in q-opt?
Q: what are the states, costs and arrows, in q-opt?
A: set of intermediate result tables

Q-Opt + Dynamic Programming

- Details: how to record the fact that, say R is sorted on R.a? or that the user requires sorted output?
- Consider the following query:

```
SELECT * 
FROM R, S, T 
WHERE R.a = S.a AND S.b = T.b 
ORDER BY R.a
```
Q-Opt + Dynamic Programming

• E.g., compute \( R \) join \( S \) join \( T \) order by \( R.a \)

Any other changes?

Candidate Plans

1. Enumerate relation orderings:

Prune plans with cross-products immediately!
1. Enumerate relation orderings:

- **Prune plans with cross-products immediately!**

2. Enumerate join algorithm choices:

- **Do this for the other plans.**

3. Enumerate access method choices:

- **Do this for the other plans.**

4. Now we can estimate the cost of each plan.
Query Optimization

- Bring query in internal form (e.g., parse tree)
- … into “canonical form” (syntactic q-opt)
- Generate alternative plans.
  - Single relation.
  - Multiple relations.
  - Nested sub-queries.
- Estimate cost for each plan.
- Pick the best one.

Nested Sub-Queries

- Re-write nested queries
- to: de-correlate and/or flatten them

```
SELECT S.sid, MIN(R.day)
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid
  AND R.bid = B.bid
  AND B.color = 'red'
  AND S.rating = (SELECT MAX(S2.rating)
                  FROM Sailors S2)
GROUP BY S.sid
HAVING COUNT(*) > 1
```

For each sailor with the highest rating (over all sailors) and at least two reservations for red boats, find the sailor id and the earliest date on which the sailor has a reservation for a red boat.

Decomposing Queries into Blocks

- The optimizer breaks up queries into blocks and then concentrates on one block at a time.
Decomposing Queries into Blocks

The optimizer breaks up queries into blocks and then concentrates on one block at a time.

Split $n$-way joins into 2-way joins, then individually optimize.

Query Optimizer Overview

- **System R:**
  - Break query in query blocks
  - Simple queries (ie., no joins): look at stats
  - $n$-way joins: left-deep join trees; ie., only one intermediate result at a time
    - **Pros:** smaller search space; pipelining
    - **Cons:** may miss optimal
  - 2-way joins: NL and sort-merge

Conclusions

- Ideas to remember:
  - Syntactic q-opt – do selections early
  - Selectivity estimations (uniformity, indep.; histograms; join selectivity)
  - Hash join (nested loops; sort-merge)
  - Left-deep joins
  - Dynamic programming