Overview of a DBMS

- Naive user
- Casual user
- DBA
- DML parser
- DML parser
- DDL parser
- Data files
- Catalog
- Buffer manager
- Transaction manager

General Overview - rel. model
- Relational model - SQL
- Functional Dependencies & Normalization
- Physical Design
- Indexing
- Query optimization
- Transaction processing

Overview - detailed
- Why q-opt?
- Equivalence of expressions
- Cost estimation
- Cost of indices
- Join strategies

Why Q-opt?
- SQL: ~declarative
- good q-opt -> big difference
  - eg., sort, scan vs
  - B-tree index, on P=1,000 pages

Q-opt steps
- bring query in internal form (eg., parse tree)
- … into ‘canonical form’ (syntactic q-opt)
- generate alt. plans
- estimate cost: pick best
Q-opt - example

select name from STUDENT, TAKES where oldID = 415 and STUDENT. name = TAKES.name

\( \pi \sigma \)

STUDENT TAKES

Q-opt - example

Canonical form

\( \pi \sigma \)

STUDENT TAKES

STUDENT TAKES

Q-opt - example

Hash join
merge join
nested loops

\( \pi \sigma \rightarrow \text{Index seq scan} \)

STUDENT TAKES

Overview - detailed

- Why q-opt?
- Equivalence of expressions
- Cost estimation
- Cost of indices
- Join strategies

Equivalence of expressions

- Aka.: syntactic q-opt
- in short: perform selections and projections early
- More details: see transf. rules in text

Equivalence of expressions

- Q: How to prove a transf. rule?
  \( \sigma_p(R_1 \bowtie R_2) \rightarrow \sigma_p(R_1) \bowtie \sigma_p(R_2) \)
- A: use RTC, to show that LHS = RHS, e.g.
  \( \sigma_p(R_1 \cup R_2) = \sigma_p(R_1) \cup \sigma_p(R_2) \)
Equivalence of expressions

\[ \sigma_p(R_1 \cup R_2) \equiv \sigma_p(R_1) \cup \sigma_p(R_2) \]
\( t \in \text{LHS} \iff \)
\( t \in (R_1 \cup R_2) \land P(t) \iff \)
\( (t \in R_1 \lor t \in R_2) \land P(t) \iff \)
\( (t \in R_1 \land P(t)) \lor (t \in R_2 \land P(t)) \iff \)

QED

Equivalence of expressions

\[ \sigma_p(R_1 \cup R_2) = \sigma_p(R_1) \cup \sigma_p(R_2) \]
\[ \text{...} \]
\[ (t \in R_1 \land P(t)) \lor (t \in R_2 \land P(t)) \iff \]
\[ (t \in \sigma_p(R_1)) \lor (t \in \sigma_p(R_2)) \iff \]
\[ t \in \sigma_p(R_1 \cup \sigma_p(R_2) \iff \]
\[ t \in \text{RHS} \]

Equivalence of expressions

- Selections
  - Perform them early
  - Break a complex predicate, and push
    \[ \sigma_{p_1 \land p_2 \land ... \land p_n}(R) = \sigma_{p_1}(\sigma_{p_2}(\ldots \sigma_{p_n}(R) \ldots)) \]
  - Simplify a complex predicate
    - \((X = Y \land Y = Z) \iff X = Z \land Y = Z\)

Equivalence of expressions

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

Equivalence of expressions

- Joins
  - Commutative, associative
    \[ R \bowtie S = S \bowtie R \]
    \[ (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T) \]
  - Q: n-way join - how many diff. orderings?
Equivalence of expressions

- Joins - Q: n-way join - how many diff. orderings?
- A: Catalan number \( \sim 4^n \)
  - Exhaustive enumeration: too slow.

Q-opt steps

- bring query in internal form (eg., parse tree)
- … into ‘canonical form’ (syntactic q-opt)
- generate alt. plans
- estimate cost: pick best

Cost estimation

- Eg., find ssn’s of students with an ‘A’ in 415 (using seq. scanning)
- How long will a query take?
  - CPU (but: small cost; decreasing; tough to estimate)
  - Disk (mainly, # block transfers)
- How many tuples will qualify?
- (what statistics do we need to keep?)

Cost estimation

- Statistics: for each relation ‘r’ we keep
  - \( nr \): # tuples;
  - \( Sr \): size of tuple in bytes

Cost estimation

- Statistics: for each relation ‘r’ we keep
  - \( nr \): # tuples;
  - \( Sr \): size of tuple in bytes

Derivable statistics

- fr: blocking factor = max # records/block (=??)
- br: # blocks (=??)
- \( SC(A,r) = \) selection cardinality = avg # of records with \( A = \) given (=??)
Derivable statistics

- fr: blocking factor = max # records/block (= B/Sr ; B: block size in bytes)
- br: # blocks (= nr / fr )

Derivable statistics

- SC(A,r) = selection cardinality = avg # of records with A=given (= nr / V(A,r) )
  (assumes uniformity ...) – eg: 10,000 students, 10 colleges – how many students in SCS?

Additional quantities we need:

- For index 'i';
  - fi: average fanout (~50-100)
  - HTi: # levels of index 'i' (~2-3)
    - ~ log(#entries)/log(fi)
  - LBi: # blocks at leaf level

Statistics

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

Q-opt steps

- bring query in internal form (eg., parse tree)
- … into “canonical form” (syntactic q-opt)
- generate alt. plans
  - selections; sorting; projections
  - joins
- estimate cost; pick best

Cost estimation + plan generation

- Selections – eg,
  select *
  from TAKES
  where grade = ‘A’
- Plans?
Cost estimation + plan generation

- Plans?
  - seq. scan
  - binary search
    - if sorted & consecutive
    - index search
      if an index exists

Cost estimation + plan generation

seq. scan = cost?
- br (worst case)
- br/2 (average, if we search for primary key)

Cost estimation + plan generation

Binary search = cost?
if sorted and consecutive:
- \( \sim \log(br) \)
- \( SC(A_1,r)/fr = \text{blocks spanned by qual. tuples} \)

Cost estimation + plan generation

estimation of selection cardinalities \( SC(A_1,r) \):
non-trivial - details later

Cost estimation + plan generation

method#3: index = cost?
- levels of index +
  - blocks w/ qual. tuples

Cost estimation + plan generation

method#3: index = cost?
- levels of index +
  - blocks w/ qual. tuples

Case#1: primary key
Case#2: sec. key - clustering index
Case#3: sec. key - non-clustered index

HT1 + 1
Cost estimation + plan generation

method#3: index = cost?
- levels of index +
- blocks w/ qual. tuples

Case #2: sec. key - clustering index
HTI + SC(Arz)/fr

Case #3: sec. key - non-clustering index
HTI + SC(Arz)
(actually, pessimistic)

Cost estimation – arithm. examples
- find accounts with branch-name = 'Perryridge'
- account(branch-name, balance, ...)

Arithm. examples – cont’d
- n=account = 10,000 tuples
- f=account = 20 tuples/block
- V(balance, account) = 500 distinct values
- V(branch-name, account) = 50 distinct values
- for branch-index: fanout fi = 20

Arithm. examples
- Q1: cost of seq. scan?
- A1: 500 disk accesses
- Q2: assume a clustering index on branch-name = cost?
Arith. examples

- A2:
  \[ HT1 + SC(\text{branch-name, account})/\text{account} \]
  \[ HT1: 50 \text{ values, with index fanout } 20 \rightarrow \]
  \[ HT1=2 \text{ levels } (\log(50)/\log(20)) = 1+ \]
- \( SC(\omega) = \# \text{ qual records} = \]
  \[ nN(\Lambda, \omega) = 10,000/50 = 200 \text{ tuples} \]
  \[ \text{spanning } 200/20 = 10 \text{ blocks} \]

Q-opt steps

- bring query in internal form (eg., parse tree)
- … into “canonical form” (syntactic q-opt)
- generate all plans
  - selections, sorting, projections
  - joins
- estimate cost: pick best

Examples

- (selection) find student record with \( \text{ssn} = 123 \)
  \[ \{ t | t \in \text{STUDENT} \land t[\text{ssn}] = 123 \} \]

Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
<td>Cid</td>
</tr>
<tr>
<td>123</td>
<td>51413</td>
</tr>
<tr>
<td>234</td>
<td>51412</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
</tr>
<tr>
<td>123</td>
</tr>
<tr>
<td>234</td>
</tr>
</tbody>
</table>
DML - nested subqueries

Drill: find the ssn of the student with the highest GPA

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>c-id</td>
</tr>
<tr>
<td>123 smith</td>
<td>15-413</td>
</tr>
<tr>
<td>234 jones</td>
<td>15-412</td>
</tr>
</tbody>
</table>

TAKES

<table>
<thead>
<tr>
<th>ssn</th>
<th>c-id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

File organization

Eg., 'Student' records - how would you store them on disk?

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 smith</td>
<td>main str</td>
</tr>
<tr>
<td>234 jones</td>
<td>forbes ave</td>
</tr>
</tbody>
</table>

Data files - catalog