Query Processing

Lecture topics:
- query interpretation
- basic operations
- costs of basic operations

References:
- text 3rd edition: chapter 18, sections 2-4, 7
- text 4th edition: chapter 15, sections 1–4, 8

Steps in query interpretation

- Translation
  - check SQL syntax
  - check existence of relations and attributes
  - replace views by their definitions
  - transform query into an internal form (similar to relational algebra)
- Optimization
  - generate alternative access plans, i.e., procedure, for processing the query
  - select an efficient access plan
- Processing
  - execute the access plan
- Data Delivery
Example

\[
\text{select V.Vno, Vname, count(*)}, \text{sum(Amount)} \\
\text{from Vendor V, Transaction T} \\
\text{where V.Vno = T.Vno and V.Vno between 1000} \\
\text{and 2000} \\
\text{group by V.Vno, Vname} \\
\text{having sum(Amount) > 100}
\]

• The following is a possible access plan for this query:
  – Scan the Vendor table, select all tuples where Vno is between 1000 and 2000, eliminate attributes other than Vno and Vname, and place the result in a temporary relation \( R_1 \)
  – Join the tables \( R_1 \) and Transaction, eliminate attributes other than Vno, Vname, and Amount, and place the result in a temporary relation \( R_2 \). This may involve:
    • sorting \( R_1 \) on Vno
    • sorting Transaction on Vno
    • merging the two sorted relations to produce \( R_2 \)
  – Perform grouping on \( R_2 \), and place the result in a temporary relation \( R_3 \). This may involve:
    • sorting \( R_2 \) on Vno and Vname
    • grouping tuples with identical values of Vno and Vname
    • counting the number of tuples in each group, and adding their Amounts
  – Scan \( R_3 \), select all tuples with \( \text{sum(Amount)} > 100 \) to produce the result.
Pictorial Access Plan

Vendor

Scan (Vno between 1000 and 2000)

Transaction

Join (V.Vno = T.Vno)

Grouping (Vno, Vname)

Scan (Sum(Amount) > 100)

Result

Some basic query processing operations

• Tuple Selection
  – without an index
  – with a clustered index
  – with an unclustered index
  – with multiple indices

• Projection

• Joining
  – nested loop join
  – hash join
  – sort-merge join
  – and others...

• Grouping and Duplicate Elimination
  – by sorting
  – by hashing

• Sorting
Tuple Selection

• sequential scanning:
  – when no appropriate index is available, we must check all file blocks holding tuples for the relation.
  – even if an index is available, scanning the entire relation may be faster in certain circumstances:
    • the relation is very small
    • the relation is large, but we expect most of the tuples in the relation to satisfy the selection criteria
  – E.g.:

        select Anum
        from Accounts
        where Balance > 1.00

Joining relations

select C.Cnum, Balance
from Customer C, Accounts A
where C.Cnum = A.Cnum

• Two relations can be joined using the “nested loop” join method. Conceptually, that method works like this:

  for each tuple c in Customer do
    for each tuple a in Accounts do
      if c.Cnum = a.Cnum then
        output c.Cnum,a.Balance
      end
    end
  end
Other techniques for join

- There are many other ways to perform a join. For example, if there is an index on the Cnum attribute of the Accounts relation, an index join could be used. Conceptually, that would work as follows:

  for each tuple c in Customer do
    use the index to find Accounts tuples a
    where a.Cnum matches c.Cnum
    if there are any such tuples a then
      output c.Cnum, a.Balance
  end

...continued

- other join techniques:
  - **sort-merge join**: sort the tuples of Accounts and of Customer on their Cnum values, then merge the sorted relations.
  - **hash join**: assign each tuple of Accounts and of Customer to a “bucket” by applying a hash function to its Cnum value. Within each bucket, look for Account and Customer tuples that have matching Cnum values.
Costs of basic operations

- Alternative access plans may be compared according to cost.
- The cost of an access plan is the sum of the costs of its component operations.
- There are many possible cost metrics. However, most metrics reflect the amounts of system resources consumed by the access plan. System resources may include:
  - disk block I/O’s
  - processing time
  - network bandwidth

A simple cost model

- The cost of an operation will be an estimate of the number of file block retrievals required to perform that operation.
- The cost of an operation will depend on the number of input tuples for that operation, and on the tuple blocking factor.
- The blocking factor is the number of tuples that are stored in each file block.
- For a relation $R$:
  - $|R|$ will represent the number of tuples in $R$
  - $b(R)$ will represent the blocking factor for $R$
  - $|R|/b(R)$ is the number of blocks used to store $R$
- We will assume that an index search has a cost of 2.
Cost of selection

• relation Mark = (Studnum, Course, Assignnum, Mark)

```sql
select Studnum, Mark
from Mark
where Course = PHYS
  and Studnum = 100
  and Mark > 90
```

• Indices:
  – clustering index CourseInd on Course
  – non-clustering index StudnumInd on Studnum

• Assume:
  – |Mark| = 10000
  – \( b(Mark) = 50 \)
  – 500 different students
  – 100 different courses
  – 10 different assignments

Selection Strategy: use CourseInd

• Assuming uniform distribution of tuples over the courses, there will be about \(|Mark|/100 = 100\) tuples with Course = PHYS.

• Searching the CourseInd index has a cost of 2. Retrieval of the 100 matching tuples adds a cost of \(100/b(Mark)\) data blocks, for a total cost of 4.

• Selection of \(N\) tuples from relation \(R\) using a clustered index has a cost of 2 + \(N/b(R)\)
Selection strategy: use StudnumInd

- Assuming *uniform distribution* of tuples over student numbers, there will be about |Mark|/500 = 20 tuples for each student.

- Searching the StudnumInd has a cost of 2. Since this is not a clustered index, we will make the pessimistic assumption that each matching record is on a separate data block, i.e., 20 blocks will need to be read. The total cost is 22.

- Selection of $N$ tuples from relation $R$ using a non-clustered index has a cost of $2 + N$

Selection strategy: scan the relation

- The relation occupies 10,000/50 = 200 blocks, so 200 block I/O operations will be required.

- Selection of $N$ tuples from relation $R$ by scanning the entire relation has a cost of $|R|/b(R)$
Cost of joining

- relation Mark = (Studnum, Course, Assignnum, Mark)

- relation Student = (Studnum, Surname, Initials, Address, Birthdate, Sex)

- select Surname, Course, Assignnum, Mark
  from Mark natural join Student

- Assume:
  - |Mark| = 10,000
  - b(Mark) = 50
  - |Student| = 1,000
  - b(Student) = 20

Cost of nested-loop join

for each Mark block M do
  for each Student block S do
    for each tuple m in M do
      for each tuple s in S do
        if m[Studnum] = s[Studnum] then
          join {m} and {s} and add to the result

- Each of the 200 blocks of Mark must be retrieved once.
- Each of the 50 blocks of Student must be retrieved once for each block of Mark, for a total of 200 x 50 = 10,000 student blocks retrieved.
- The total cost is 10,200 block retrievals.
The cost of a nested-loop join of \( R_1 \) and \( R_2 \) is either:

\[
\frac{|R_1|}{b(R_1)} + \frac{|R_1|}{b(R_1)} \cdot \frac{|R_2|}{b(R_2)}
\]

or

\[
\frac{|R_2|}{b(R_2)} + \frac{|R_1|}{b(R_1)} \cdot \frac{|R_2|}{b(R_2)}
\]

depending on which relation is the outer relation.

Cost of index join

Suppose there is a unique, non-clustered index on Student [Studnum]

for each Mark block Bm do

for each tuple m in Bm do

use the index to locate Student tuples s with s[Studnum] = m[Studnum]
join \{m\} and \{s\} and add to the result

Each of Mark's 200 blocks must be retrieved once. For each tuple of Mark, a single tuple of Student will be retrieved. This will require an index lookup and a data block retrieval, for a cost of 3.

The total cost is 10000 * 3 + 200 = 30,200 block retrievals.
• The cost of a unique index join, where $R_1$ is the outer relation, and $R_2$ is the inner, indexed relation, is:

$$\frac{|R_1|}{b(R_1)} + 3 \cdot |R_1|$$

Summary

• An access plan is a detailed plan for the execution of a query. It describes:
  – the sequence of basic operations (select, project, join, etc.) used to process the query
  – how each operation will be implemented, e.g., which join method will be used, which indices will be used to perform a selection.

• Each operation may be assigned a cost, and an access plan may be assigned a cost by summing the costs of its component operations.

• The cost assigned to an operation depends on assumptions about the data in the relations.