CSC 742
Database Management Systems

Topic #16: Query Optimization

Agenda
- Typical steps of query processing
- Two main techniques for query optimization
  - Heuristics based query optimization
  - Cost based query optimization
- Translating SQL queries into relational algebra
- Heuristics based query optimization

Typical Steps when Processing A High-level Query

```
Query
  Scanning, Parsing, and Validating
  Intermediate form of query
    Query Optimizer
    Execution Plan
    Query Code Generator
    Code to execute the query
  Runtime database processor

Query result
```

Questions
- Difference between high-level query languages and high-level programming languages in terms of queries?
  - SQL v.s. C
- Can we find the best execution plan?
- Do we always use the best execution plan?

Two Main Techniques for Query Optimization
- Heuristic Rules
  - A heuristic is a rule that works well in most cases, but not always.
- Cost based query optimization
  - Estimate the cost for each execution plan, and choose the one with the lowest cost.
- Can we get the best execution plan?

Translating SQL queries into extended relational algebra
- SQL queries are decomposed into query blocks
  - Each query block contains a single SELECT – FROM – WHERE expression as well as GROUP BY and HAVING clauses.
  - Nested queries within a query are identified as separate query blocks.
- Query optimization
  - Choose an execution plan for each block
Basic Operations

- DBMS implements basic algorithms for certain operations
  - Select
  - Project
  - Join
  - Cartesian Product
  - Set operations
  - …
- These operations are combined to form the execution plans.

Query Tree

- A query tree is a data structure that corresponds to a relational algebra expression.
  - Input relations are leaf nodes of the tree
  - Relational operations are internal nodes.
  - An execution of the query tree consists of:
    - executing an internal node operation whenever its operands are available,
    - and then replacing the node by the relation that results from executing the operation.

Using Heuristics in Query Optimization

- An example of a heuristic
  - Apply SELECT before join.
  - Questions:
    - Does it work in most of cases?
    - Exception?
Using Heuristics in Query Optimization

- **General idea**
  - Many different relational algebra expressions (and thus query trees) are equivalent.
  - Transform the initial query tree of a query into an equivalent final query tree that is efficient to execute.

---

**Algorithm**

- Break up any SELECT operations with conjunctive conditions into a cascade of SELECT operations.
- Push SELECT operations as far down the query tree as possible.
- Rearrange binary operations so that:
  - the most restrictive SELECT operations are executed first.
  - Avoid CARTESIAN PRODUCT.
- Try to combine a CARTESIAN PRODUCT with a SELECT operation into a join operation.
- Break up PROJECT operation and move lists of projection attributes as down the tree as possible by creating new project operations.
- Identify sub-trees that represent groups of operations that can be executed by a single algorithm.

---

**Example Query**

```
SELECT LNAME
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE PNAME='Aquarius' AND PNUMBER=PNO
AND ISN=SSN AND BDATE>'1957-12-31'
```

**Step 1:** Break up SELECT operations

- \( \sigma_{PNAME='Aquarius'} \)
- \( \sigma_{PNUMBER=PNO} \)
- \( \sigma_{ISN=SSN} \)
- \( \sigma_{BDATE>'1957-12-31'} \)

**Step 2:** Push down SELECT operations

- \( \sigma_{BDATE>'1957-12-31'} \)
- \( \sigma_{PNAME='Aquarius'} \)
- \( \sigma_{PNUMBER=PNO} \)
- \( \sigma_{ISN=SSN} \)

**Step 3:** Apply restrictive SELECT first, Avoid cartesian product

- \( \sigma_{BDATE>'1957-12-31'} \)
- \( \sigma_{PNAME='Aquarius'} \)
- \( \sigma_{PNUMBER=PNO} \)
- \( \sigma_{ISN=SSN} \)

---
Step 4: Combine a Cartesian product with a subsequent `SELECT` into a job.

```
\[ \pi_{\text{LNAME}} \sigma_{\text{DATE} > '1957-12-31'}(\text{EMPLOYEE}) \times \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \]
```

Step 5: Push down `PROJECT` operations.

```
\[ \pi_{\text{LNAME}} \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \rightarrow \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \]
```

Step 6: Identify sub-trees.

```
\[ \pi_{\text{LNAME}} \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \rightarrow \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \]
```

```
\[ \pi_{\text{LNAME}} \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \rightarrow \pi_{\text{SSN}, \text{LNAME}}(\text{EMPLOYEE}) \]
```