ĐẠI HỌC
BÁCH KHOA

25 YEARS ANNIVERSARY
SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Lesson 7
Query Processing

Outline

- Overview
  - What is query processing
  - Phrases of query processing
  - Parser
  - Optimizer
- Understanding query optimizer
  - Step 1: Equivalence transformation
  - Step 2: Annotation for the algorithm of the RA expression
  - Step 3: Cost estimation for different query execution plans

Objectives

- Upon completion of this lesson, students will be able to:
  - Understanding different phases of query processing
  - Understanding the implementation of query optimizer

Keywords

Query processing	Activities involved in retrieving/storing data from/to the database
Query optimization	Selection of an efficient query execution plan
Relational algebra	An algebra whose operands are relations or variables that represent Relations

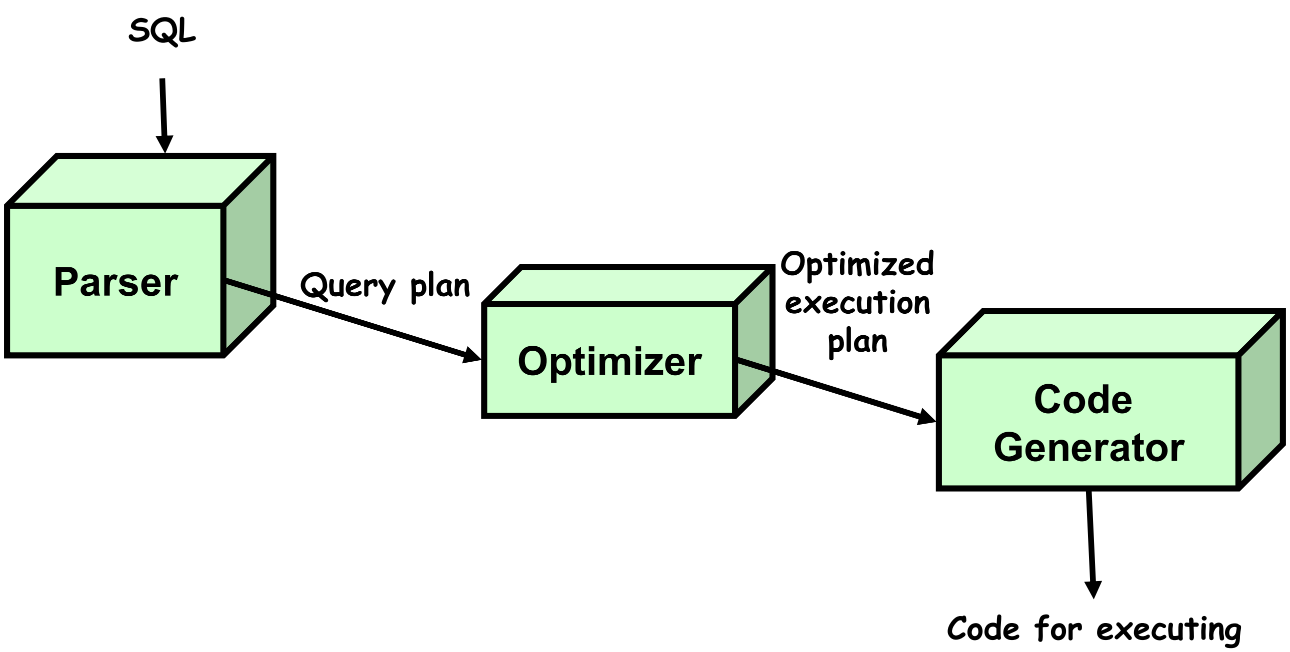
1. Overview

1. What is query processing
2. Phrases of query processing
3. Parser
4. Optimizer

1.1. What is query processing

- The entire process or activities involved in retrieving data from the database
  - SQL query translation into low level instructions (usually relational algebra)
  - Query optimization to save resources, cost estimation or evaluation of query
  - Query execution for the extraction of data from the database.

1.2. Phases of query processing



1.3. Parser

- **Scans and parses the query into individual tokens and examines for the correctness of query**
  - Does it containt the right keywords?
  - Does it conform to the syntax?
  - Does it containt the valid tables, attributes?
- **Output: Query plan**
  - E.g.
    - Input: SELECT balance FROM account WHERE balance < 2500
    - Output: Relational algebra expression
    - But it's not unique

$$\Pi_{balance}(\sigma_{balance<2500}(account))$$
$$\sigma_{balance<2500}(\Pi_{balance}(account))$$

1.4. Optimizer

- **Input: RA expression** $\Pi_{balance}(\sigma_{balance<2500}(account))$
- **Output: Query execution plan**

- Query execution plan = query plan + the algorithms for the executions of RA operations
- Aims to choose the cheapest execution plan out of the possible ones
  - Step 1: Equivalence transformation
  - Step 2: Annotation for the algorithm of the RA expression
  - Step 3: Cost estimation for different query execution plans

$$\Pi_{balance}$$

$$|$$

$$\sigma_{balance<2500}$$

use index 1

$$|$$

$$account$$

2. Understanding optimizer

Choose the cheapest execution plan out of the possible ones

- Step 1: Equivalence transformation
- Step 2: Annotation for the algorithmic execution of the RA expression
- Step 3: Cost estimation for different query execution plans

2.1. Step 1: Equivalence transformation

- RA expressions are equivalent if they generate the same set of tuples on every database instance
- **Equivalence rules:**
  - Transform one relational algebra expression into equivalent one
  - Similar to numeric algebra: a + b = b + a, a(b + c) = ab + ac, etc
- **Why producing equivalent expressions?**
  - equivalent algebraic expressions give the same result
  - but usually the execution time varies significantly

2.1. Step 1: Equivalence transformation

- Equivalance tranformation rules
- (1) Conjunctive selection operations can be deconstructed into a sequence of individual seections; cascade of σ
  - $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}\left(\sigma_{\theta_2}(E)\right)$
- (2) Selection operations are commutative
  - $\sigma_{\theta_1}\left(\sigma_{\theta_2}(E)\right) = \sigma_{\theta_2}\left(\sigma_{\theta_1}(E)\right)$
- (3) Only the final operations in a sequence of projection operations is needed; cascade of Π
  - $\Pi_{L_1}\left(\Pi_{L_2}\left(\ldots \Pi_{L_n}(E) \ldots\right)\right) = \Pi_{L_1}(E)$
- (4) Selections can be combined with Cartesian products and theta joins
  - $\sigma_{\theta_1}(E_1 \times E_2) = E_1 \bowtie_{\theta_1} E_2$
  - $\sigma_{\theta_1}\left(E_1 \bowtie_{\theta_2} E_2\right) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

2.1. Step 1: Equivalence transformation

- Equivalance tranformation rules
- (5) Theta Join operations are commutative
  - $E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$
- (6) Natural join operations are associative
  - $E_1 \bowtie (E_2 \bowtie E_3) = (E_1 \bowtie E_2) \bowtie E_3$
  - Theta join are associative in the following manner where θ_2 involves attributes from E2 and E3 only
  - $(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$

2.1. Step 1: Equivalence transformation

- Equivalance tranformation rules
- (7) Selection distributes over joins in the following ways
  - If predicate involves attributes of E1 only
  - $\sigma_{\theta_1}\left(E_1 \bowtie_{\theta_2} E_2\right) = \sigma_{\theta_1}(E_1) \bowtie_{\theta_2} E_2$
  - If predicate θ_1 involves only attributes of E1 and θ_2 involves only attributes of E2 (a consequence of rule 7 and 1)
  - $\sigma_{\theta_1 \wedge \theta_2}\left(E_1 \bowtie_{\theta_3} E_2\right) = \sigma_{\theta_1}(E_1) \bowtie_{\theta_3} \sigma_{\theta_2}(E_2)$

2.1. Step 1: Equivalence transformation

- Equivalance tranformation rules
- (8) Projection distributes over join as follows
  - $\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1}(E_1) \bowtie_\theta \Pi_{L_2}(E_2)$
  - If θ involves attributes in $L_1 \cup L_2$ only and L_i contains attributes of E_i
- (9) The set operations union and intersection are commutative
  - $E_1 \cup E_2 = E_2 \cup E_1$
  - $E_1 \cap E_2 = E_2 \cap E_1$
- (10) The union and intersection are associative
  - $(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$

2.1. Step 1: Equivalence transformation

- Equivalance tranformation rules
- (11) The selection operation distributes over union, intersection, and set-difference
  - $\sigma_\theta(E_1 \cup E_2) = \sigma_\theta(E_1) \cup \sigma_\theta(E_2)$
  - $\sigma_\theta(E_1 \cap E_2) = \sigma_\theta(E_1) \cap \sigma_\theta(E_2)$
  - $\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$
- (12) The project operation distributes over the union
  - $\Pi_L(E_1 \cup E_2) = \Pi_L(E_1) \cup \Pi_L(E_2)$

2.2. Step 2: Execution algorithms of RA operations

- **Algebra expression is not a query execution plan.**
- **Additional decisions required:**
    - which indexes to use, for example, for joins and selects?
    - which algorithms to use, for example, sort-merge vs. hash join?
    - materialize intermediate results or pipeline them?

2.2. Step 2: Execution algorithms of RA operations

- Basic Operators
- One-pass operators:
  - Scan
  - Select
  - Project
- Multi-pass operators:
  - Join
    - Various implementations
    - Handling of larger-than-memory sources
  - Aggregation, union, etc.

2.2. Step 2: Execution algorithms of RA operations

- 1-Pass Operators: Scanning a Table
  - Sequential scan: read through blocks of table
  - Index scan: retrieve tuples in index order

2.2. Step 2: Execution algorithms of RA operations

- Nested-loop JOIN

```
For each tuple tr in r {
    for each tuple ts in s {
        if (tr and ts satisfy the join condition) {
            add tuple tr x ts to the result set
        }
    }
}
```

  - No index needed
  - Any join condition types
  - Expensive: $O(n^2)$

2.2. Step 2: Execution algorithms of RA operations

- Single-loop JOIN (Index-based)

```
for each tube tr in R {
    seach for ts in s thought index {
        if ts.exist() {
         add tr x ts to the result set
        }
    }
}
```

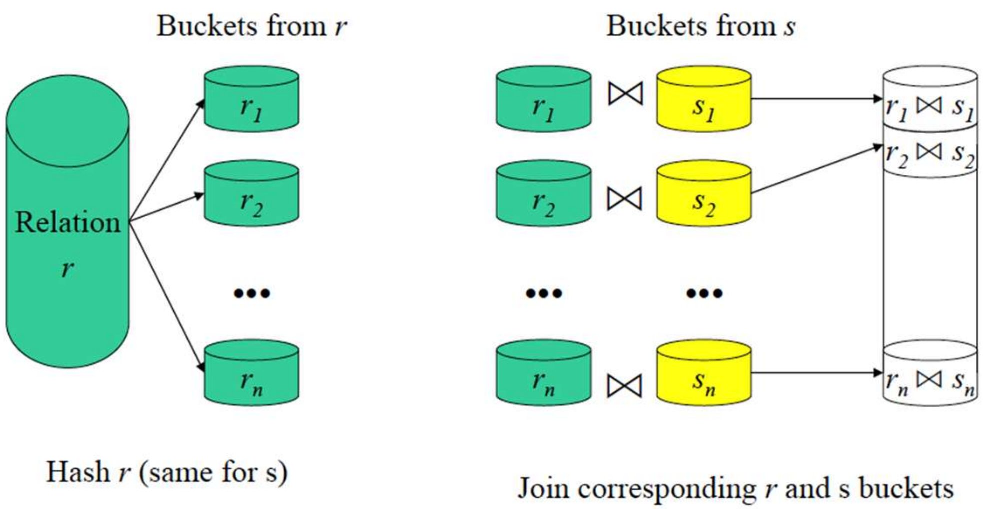
  - Index needed
  - Cheaper: O(nlogm)

2.2. Step 2: Execution algorithms of RA operations

- Sort-merge JOIN
  - Requires data physically sorted by join attributes: Merge and join sorted files, reading sequentially a block at a time
  - Maintain two file pointers
    - While tuple at R < tuple at S, advance R (and vice versa)
    - While tuples match, output all possible pairings
  - Very efficient for presorted data. Otherwise, may require a sort (adds cost + delay)

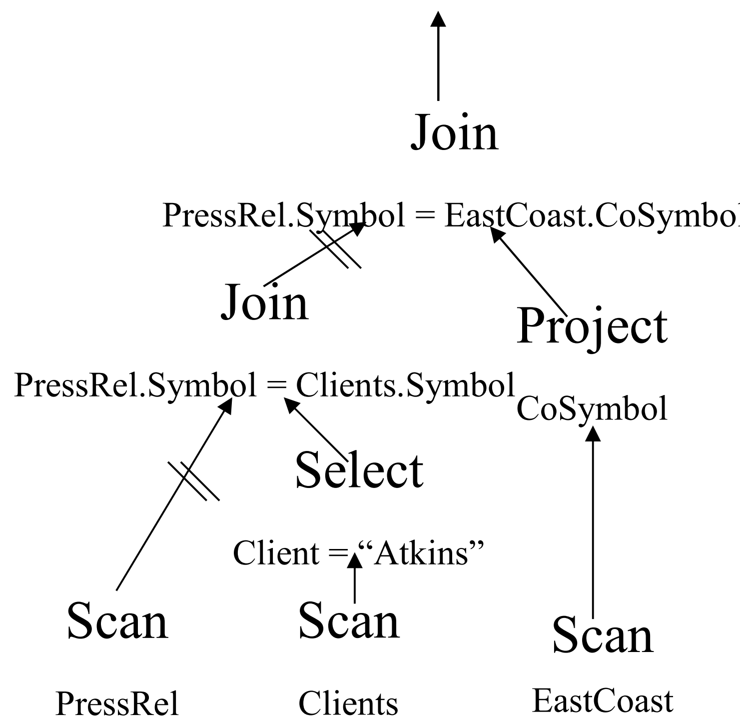
2.2. Step 2: Execution algorithms of RA operations

- Partition-hash JOIN
  - Hash two relations on join attributes
  - Join buckets accordingly



2.2. Step 2: Execution algorithms of RA operations

- Execution Strategy: Materialization vs. Pipelining
  - Execution strategy defines how to walk the query execution plan
    - Materialization
    - Pipelining



2.2. Step 2: Execution algorithms of RA operations

- **Materialization**
  - Performs the innermost or leaf-level operations first of the query execution plan
  - The intermediate result of each operation is materialized into temporary relation and becomes input for subsequent operations.
  - The cost of materialization is the sum of the individual operations plus the cost of writing the intermediate results to disk
    - lots of temporary files, lots of I/O.

2.2. Step 2: Execution algorithms of RA operations

- **Pipelining**
  - Operations form a queue, and results are passed from one operation to another as they are calculated
  - Pipelining restructures the individual operation algorithms so that they take streams of tuples as both input and output.
  - **Limitation**
    - algorithms that require sorting can only use pipelining if the input is already sorted beforehand
    - since sorting by nature cannot be performed until all tuples to be sorted are known.

2.3. Step 3: Cost estimation

- Each relational algebra expression can result in many query execution plans
- Some query execution plans may be better than others
- Finding the fastest one
  - Just an estimation under certain assumptions
  - Huge number of query plans may exist

2.3. Step 3: Cost estimation

- Cost estimation factors
  - Catalog information: database maintains statistics about relations
  - Ex.
    - number of tuples per relation
    - number of blocks on disk per relation
    - number of distinct values per attribute
    - histogram of values per attribute
  - Problems
    - cost can only be estimated
    - updating statistics is expensive, thus they are often out of date

2.3. Step 3: Cost estimation

- Choosing the cheapest query plan
  - **Problem:**
    - Estimating cost for all possible plans too expensive.
  - **Solutions:**
    - pruning: stop early to evaluate a plan
    - heuristics: do not evaluate all plans
  - **Real databases use a combination of**
    - Apply heuristics to choose promising query plans.
    - Choose cheapest plan among the promising plans using pruning.
  - **Examples of heuristics:**
    - perform selections as early as possible
    - perform projections early avoid Cartesian products

2.3. Step 3: Cost estimation

- Heuristic rules
  - Break apart conjunctive selections into a sequence of simple selections
  - Move σ down the query tree as soon as possible
  - Replace σ-x pairs by $\bowtie$
  - Break apart and move Π down the tree as soon as possible
  - Perform the joins with the smallest expected result first

Remark

- Query processing is the entire process or activities involved in retrieving data from the database
  - Parser
  - Optimizer
  - Code generator
- Query optimizer
  - Step 1: Equivalence transformation
  - Step 2: Annotation for the algorithm of the RA expression
  - Step 3: Cost estimation for different query execution plans

Quiz 1.

Quiz Number	1	Quiz Type	OX	Example Select
Question	What is the output of the parser in query processing?			
Example	A. Query execution plan B. Query plan C. Relational algebra expression D. Code for executing			
Answer	B and C			
Feedback	The output of the parser is a query plan, usually in term of a relational algebra expression			

Quiz 2.

Quiz Number	2	Quiz Type	OX	Example Select
Question	What can the query optimizer do?			
Example	A. Equivalence transformation B. Annotation for the algorithmic execution of the RA expression C. Cost estimation D. Executing the query plan and return results			
Answer	A, B and C			
Feedback	A, B and C are 3 common steps in query optimization			

Summary

- Overview of query processing
  - Parser
  - Optimizer
- Understanding query optimizer
  - Step 1: Equivalence transformation
  - Step 2: Annotation for the algorithmic execution of the RA expression
  - Step 3: Cost estimation for different query execution plans

Next lesson: Transaction management

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book. Pearson Prentice Hall. the 2nd edition. 2008: Chapter 7
- Nguyen Kim Anh, Nguyên lý các hệ cơ sở dữ liệu, NXB Giáo dục. 2004: Chương 7