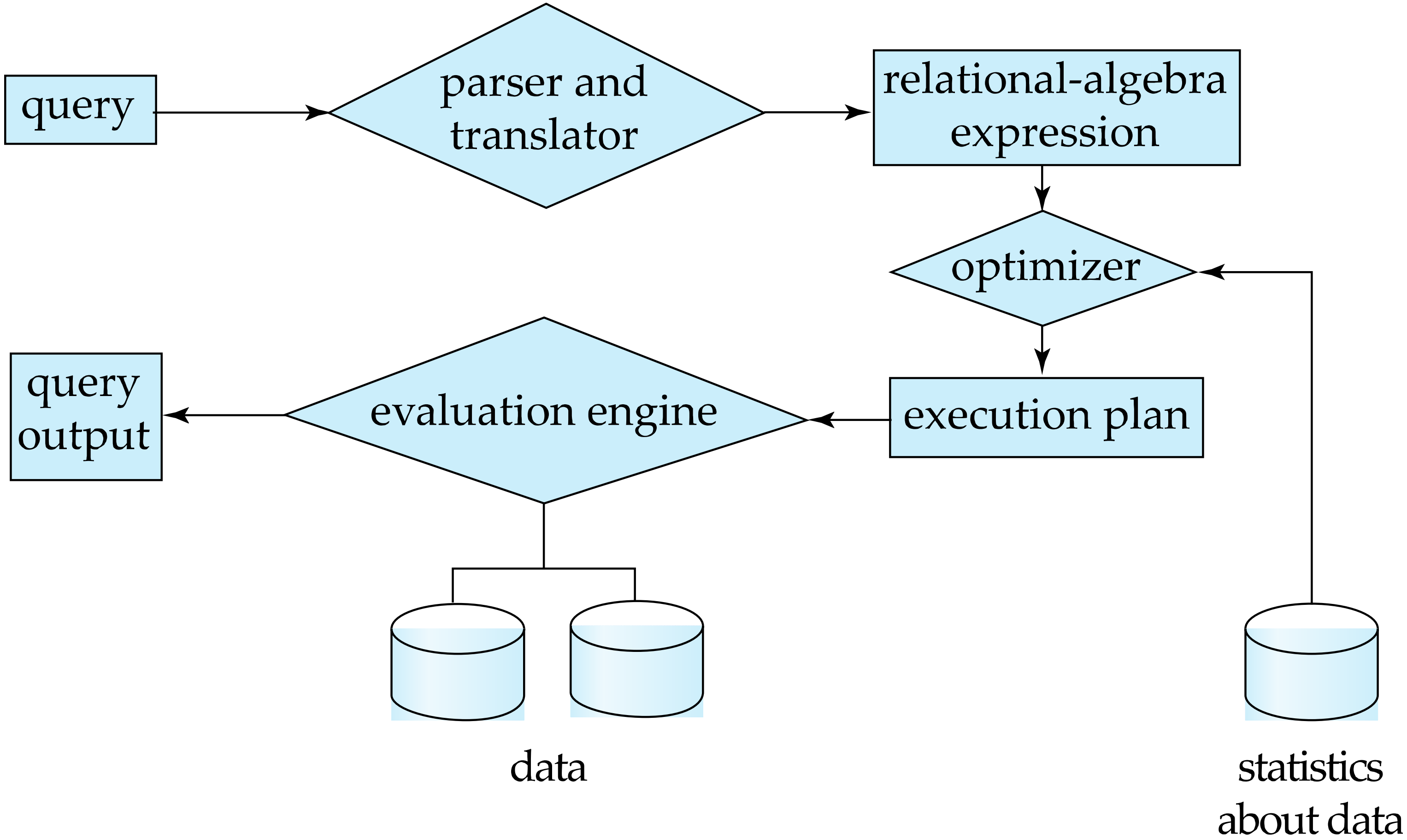
Database Systems
Lecture14 – Chapter 15: Query Processing

SUNG KYUN KWAN UNIVERSITY

Beomseok Nam (남범석)
bnam@skku.edu

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Basic Steps in Query Processing (Cont.)

- Parsing and translation
  - Parser checks syntax, verifies relations
  - translate the query into relational algebra.

- Evaluation
  - The query-execution engine takes a query-evaluation plan, executes that plan, and returns answers to the query.

Review: Relational Algebra

- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

Example Query written in Relational Algebra

- Find the names of all instructors in the Physics department, along with the *course_id* of all courses they have taught

  - Query 1

    $$\Pi_{instructor.ID,course\_id}\,(\sigma_{dept\_name=\text{"Physics"}}\,(\sigma_{instructor.ID=teaches.ID}\,(instructor \times teaches)))$$

  - Query 2

    $$\Pi_{instructor.ID,course\_id}\,(\sigma_{instructor.ID=teaches.ID}\,(\sigma_{dept\_name=\text{"Physics"}}\,(instructor) \times teaches))$$

Basic Steps in Query Processing : Optimization

- A relational algebra expression may have many equivalent expressions
  - E.g., $\sigma_{salary<75000}(\Pi_{salary}(instructor))$ is equivalent to $\Pi_{salary}(\sigma_{salary<75000}(instructor))$

- Each relational algebra operation can be evaluated using one of several different algorithms
- A sequence of primitive operation to evaluate a query is a **query-evaluation-plan**.
  - E.g., can use an index on *salary* to find instructors with salary < 75000,
  - or can perform complete relation scan and discard instructors with salary $\geq$ 75000

Basic Steps: Optimization (Cont.)

- **Query Optimization**: Amongst all equivalent evaluation plans choose the one with lowest cost.
  - Cost is estimated using statistical information from the database catalog
    - e.g. number of tuples in each relation, size of tuples, etc.
- In this chapter we study
  - How to measure query costs
  - Algorithms for evaluating relational algebra operations
  - How to combine algorithms for individual operations in order to evaluate a complete expression

Measures of Query Cost

- Cost is generally measured as total elapsed time for answering query
  - Many factors contribute to time cost
    - *disk accesses, CPU*, or even network *communication*
- Disk access is the predominant cost, and is easy to estimate.
- Disk access is measured by taking into account
  - Number of seeks * average-seek-cost
  - Number of blocks read * average-block-read-cost
  - Number of blocks written * average-block-write-cost
    - Cost to write a block is greater than cost to read a block

Measures of Query Cost (Cont.)

- For simplicity we just use the **number of block transfers** *from disk and the* **number of seeks** as the cost measures
  - ***t_T* – time to transfer one block**
  - ***t_S* – time for one seek**
  - Cost for b block transfers plus S seeks

    $b * t_T + S * t_S$
- We ignore CPU costs for simplicity
  - Real systems do take CPU cost into account
- We do not include cost to writing output to disk in our cost formulae

- **File scan**
  - The lowest level operator to access data
- Algorithm **A1** (**linear search**). Scan each file block and test all records to see whether they satisfy the selection condition.
  - Cost estimate = b_r block transfers + 1 initial seek
    - b_r : number of blocks containing records from relation r
    - We assume file blocks are stored contiguously.
  - If selection is on a unique key attribute, can stop on finding record
    - cost = (0.5 b_r) block transfers + 1 seek
  - Linear search can be applied regardless of
    - selection condition or
    - ordering of records in the file, or
    - availability of indices

Selections Using Indices

- **Index scan** – search algorithms that use an index
  - selection condition must be on search-key of index.
- **A2** (**primary B+-tree index, equality on key**).
  - Retrieve a single record that satisfies the corresponding equality condition
  - $Cost = h_i * (t_T + t_S)$
  - h_i : *height of the B+-tree*
- **A3** (**primary B+-tree index, equality on non unique key**)
  - Retrieve multiple records.
  - Records will be on consecutive blocks
    - Let b = number of blocks containing matching records
  - $Cost = h_i * (t_T + t_S) + t_T * \mathrm{b}$

Selections Using Indices

- **primary index vs secondary index**
  - **primary index is an index whose search key ordering is used to order records on physical disk.**
  - **secondary index provides pointers to records instead**

- **A4** (**secondary index, equality on non unique key**).
  - Retrieve a single record if the search-key is a candidate key
    - *Cost* = $(h_i + 1) * (t_T + t_S)$
  - Retrieve multiple records if search-key is not a candidate key
    - each of *n* matching records may be on a different block
    - Cost = $(h_i + n) * (t_T + t_S)$

Selections Involving Comparisons

- Can implement selections of the form $\sigma_{A \leq V}(r)$ or $\sigma_{A \geq V}(r)$ by using
  - a linear file scan,
  - or by using indices in the following ways:

- **A5** (**primary B+-tree index, comparison**). (Relation is sorted on A)
  - For $\sigma_{A \geq V}(r)$ use index to find first tuple $\geq v$ and scan relation sequentially from there
    - $Cost = h_i * (t_T + t_S) + b * t_T$
  - For $\sigma_{A \leq V}(r)$ just scan relation sequentially till first tuple > v
    - do not use index

Selections Involving Comparisons

- **A6** (**secondary B+-tree index, comparison**).
  - For $\sigma_{A \geq V}(r)$ use index to find first index entry $\geq v$ and scan index sequentially from there, to find pointers to records.
  - For $\sigma_{A \leq V}(r)$ just scan leaf pages of index finding pointers to records, till first entry > v
  - In either case, retrieve records that are pointed to
    - requires an I/O for each record
    - Linear file scan may be cheaper

Implementation of Complex Selections

- **Conjunction:** $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \theta n}(r)$
- **A7** (**conjunctive selection using one index**).
  - Select a θ_i (or multiple θ_i if possible) and algorithms A1 through A7 that results in the least cost for $\sigma_{\theta i}(r)$.
  - Test other conditions on tuple after fetching it into memory buffer.
- **A8** (**conjunctive selection using composite index**).
  - Use appropriate composite (multiple-key) index if available.
- **A9** (**conjunctive selection by intersection of identifiers**).
  - Requires indices with record pointers.
  - Use corresponding index for each condition, and take intersection of all the obtained sets of record pointers.
  - Then fetch records from file
  - If some conditions do not have appropriate indices, apply test in memory.

Algorithms for Complex Selections

- **Disjunction:** $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \theta n}(r)$.
- **A10** (**disjunctive selection by union of identifiers**).
  - Take the union of retrieved pointers only if *all* conditions have available indices.
    - Otherwise use linear scan.

- **Negation:** $\sigma_{\neg\theta}(r)$
  - Use linear scan on file
  - Use index if very few records satisfy $\neg\theta$ and an index is applicable to θ