SNS COLLEGE OF TECHNOLOGY

COIMBATORE

AN AUTONOMOUS INSTITUTION

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE New Delhi & affiliated to the Anna University, Chennai

DEPARTMENT OF MCA

Course Name : 19CAT609 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / II Semester

Unit IV - QUERY EVALUATION AND DATABASE DESIGN

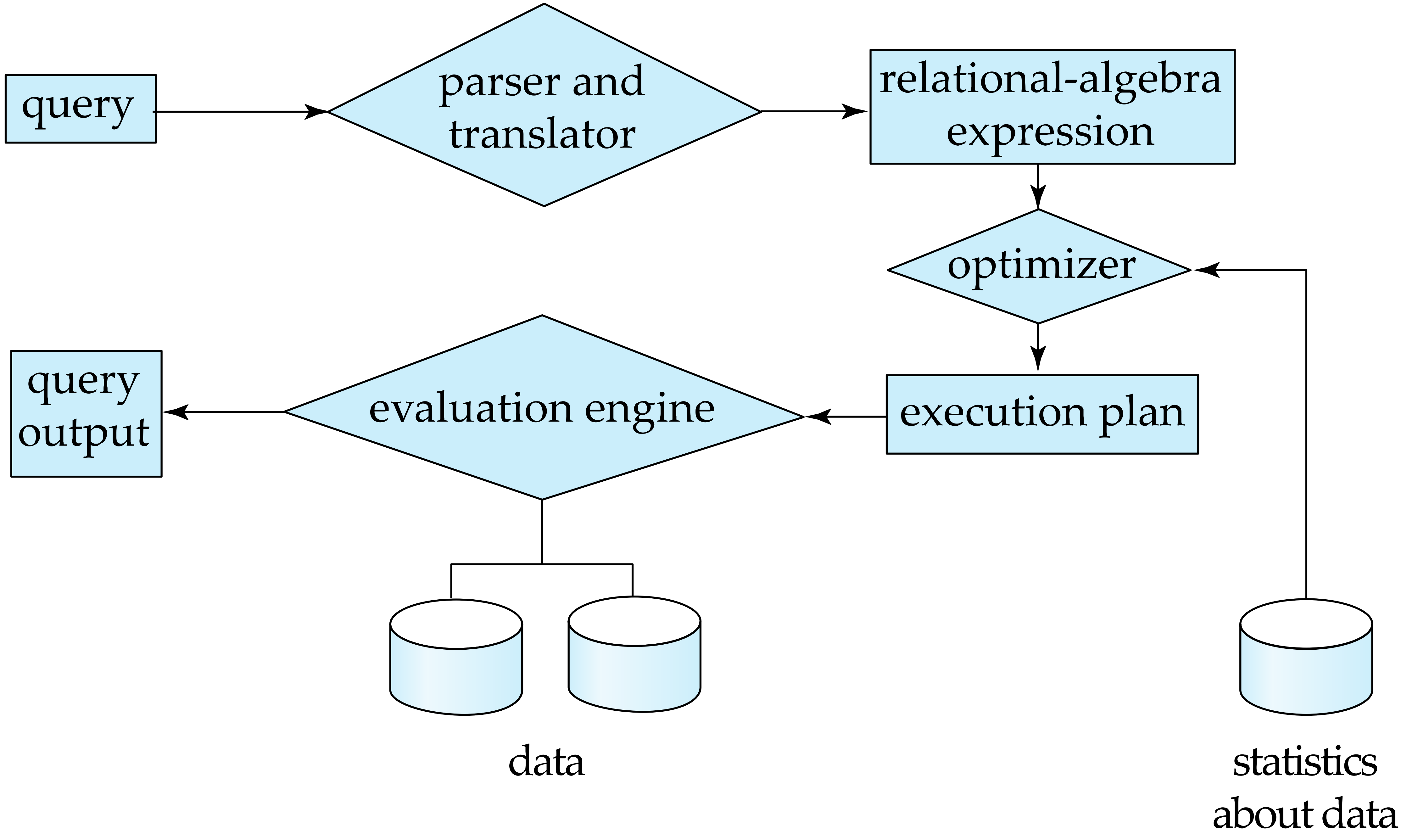
Topic I – Query Processing

Query Processing

- Overview
- Measures of Query Cost
- Selection Operation
- Sorting
- Join Operation
- Other Operations
- Evaluation of Expressions

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



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

Correspondingly, a relational-algebra expression can be evaluated in many ways.

Annotated expression specifying detailed evaluation strategy is called an **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

In Chapter 14

- We study how to optimize queries, that is, how to find an evaluation plan with lowest estimated cost

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*
Typically disk access is the predominant cost, and is also relatively easy to estimate. 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
data is read back after being written to ensure that the write was successful

Measures of Query Cost

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

Measures of Query Cost

Several algorithms can reduce disk IO by using extra buffer space
- Amount of real memory available to buffer depends on other concurrent queries and OS processes, known only during execution
  - We often use worst case estimates, assuming only the minimum amount of memory needed for the operation is available

Required data may be buffer resident already, avoiding disk I/O
- But hard to take into account for cost estimation

Selection Operation

File scan

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 seek
  - b_r denotes number of blocks containing records from relation r
- If selection is on a key attribute, can stop on finding record
  - cost = $(b_r/2)$ block transfers + 1 seek
- Linear search can be applied regardless of
  - selection condition or
  - ordering of records in the file, or
  - availability of indices

Note: binary search generally does not make sense since data is not stored consecutively

- except when there is an index available,
- and binary search requires more seeks than index search

Selections Using Indices

Index scan – search algorithms that use an index

selection condition must be on search-key of index.

A2 (**primary index, equality on key**). Retrieve a single record that satisfies the corresponding equality condition

$Cost = (h_i + 1) * (t_T + t_S)$

A3 (**primary index, equality on nonkey**) Retrieve multiple records.

Records will be on consecutive blocks

Let b = number of blocks containing matching records

$Cost = h_i * (t_T + t_S) + t_S + t_T * b$

Selections Using Indices

A4 (**secondary index, equality on nonkey**).

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 = $(h_i + n) * (t_T + t_S)$

Can be very expensive!

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 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

For $\sigma_{A \leq V}(r)$ just scan relation sequentially till first tuple $> v$; do not use index

A6 (**secondary 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 combination of θ_i 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**).

- Applicable if *all* conditions have available indices.
  - Otherwise use linear scan.
- Use corresponding index for each condition, and take union of all the obtained sets of record pointers.
- Then fetch records from file

Negation: $\sigma_{\neg\theta}(r)$

- Use linear scan on file
- If very few records satisfy $\neg\theta$, and an index is applicable to θ
  - Find satisfying records using index and fetch from file

Reference

1. https://www.tutorialspoint.com/dbms/dbms_file_structure.htm#:~:text=Relative%20data%20and%20information%20is,blocks%20that%20can%20store%20records.
2. https://www.javatpoint.com/dbms-file-organization
3. https://www.tutorialspoint.com/dbms/dbms_storage_system.htm

THANK YOU