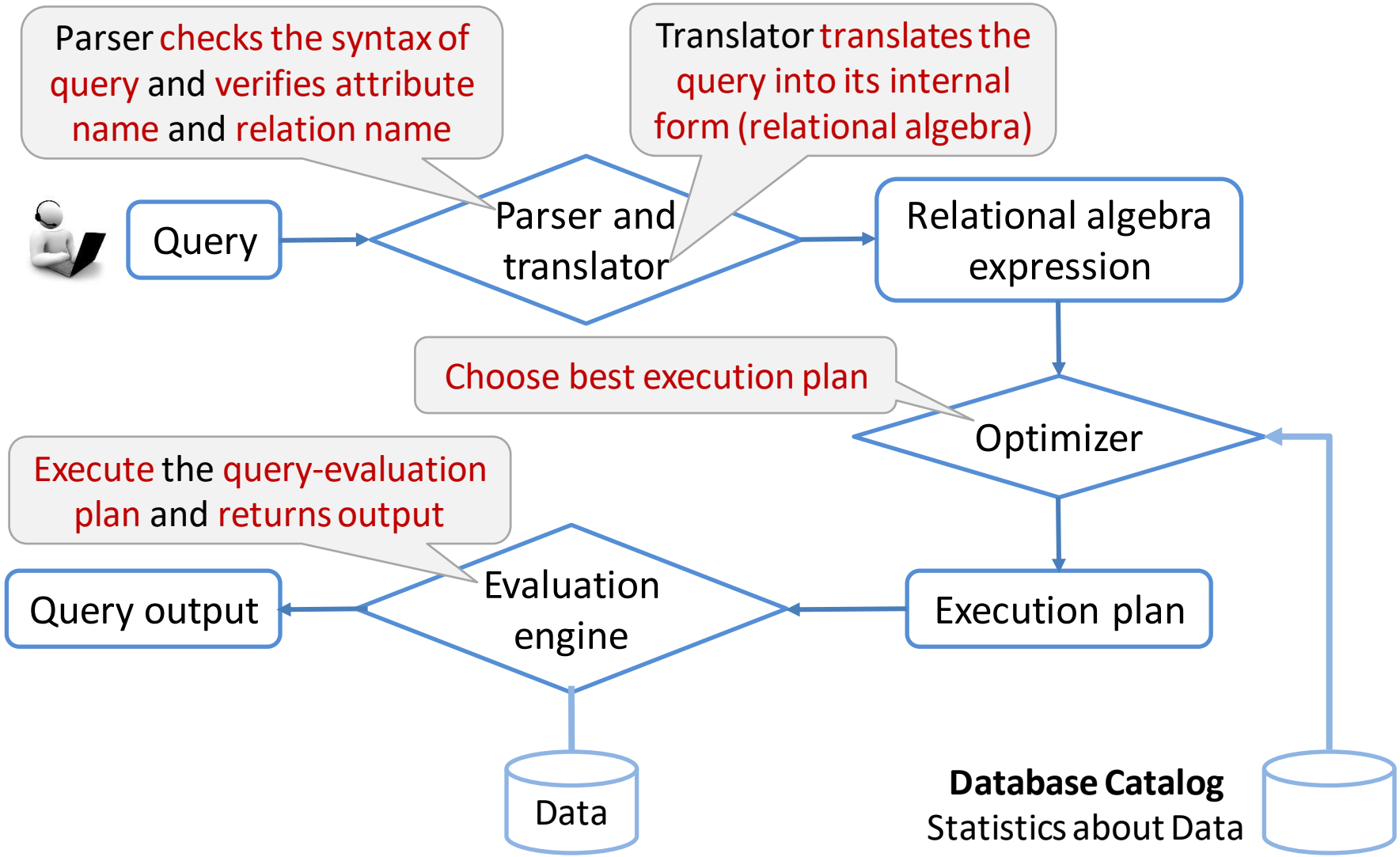
Query Processing & Optimization

BY: PROF. PRACHI SHAH

Query Processing & Optimization

- **Query Processing** refers to the range of activities involved in extracting data from database. It is a translation of high-level queries into low-level expression.
- It is a step wise process that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result.
- **Query Optimization** is a process in which multiple query-execution plans for a query are examined and most efficient plan is identified and executed.

Steps in Query Processing



Measures of Query Cost

- Cost is generally measured as the **total time required to execute a statement/query**.
- Factors contribute to time cost
  1. Disk accesses (time to process a data request and retrieve the required data from the storage device)
  2. CPU time to execute a query
  3. Network communication cost
- Disk access is the predominant (major) cost, since disk access is slow as compared to in-memory operation.
- Cost to write a block is greater than cost to read a block because data is read back after being written to ensure that the write was successful.

Selection operation

- **Symbol**: σ (Sigma)
- **Notation**: $\sigma_{condition}(Relation)$
- **Operation**: Selects tuples from a relation that satisfy a given condition.

RollNo	Name	Branch	SPI
101	Raj	CE	8
102	Meet	ME	9
103	Harsh	EE	8
104	Punit	CE	9

$$\sigma_{Branch='CE'}(Student)$$

RollNo	Name	Branch	SPI
101	Raj	CE	8
104	Punit	CE	9

Search algorithm for selection operation

1. **Linear search (A1)**
2. **Binary search (A2)**

Linear search (A1)

- It **scans each blocks** and **tests all records** to see whether they **satisfy the selection condition**.
  - Cost of linear search (worst case) = **b_r**

    b_r denotes number of blocks containing records from relation r
- If the **selection condition is there on a (primary) key attribute**, then **system can stop searching if the required record is found**.
- Linear search can be applied regardless of
  - **selection condition** or
  - **ordering of records in the file (relation)**
- This algorithm is **slower** than binary search algorithm.

Binary search (A2)

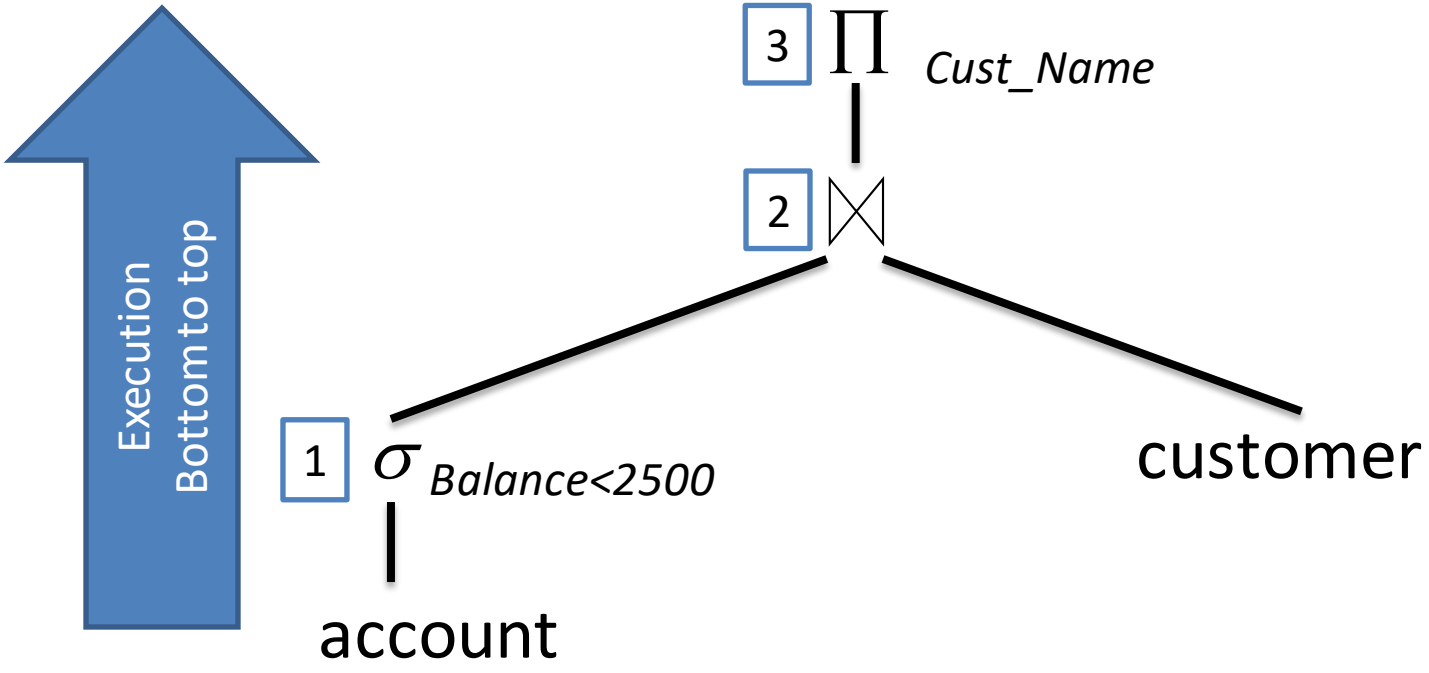
- Generally, this algorithm is used if **selection is an equality comparison on the (primary) key attribute** and **file (relation) is ordered (sorted) on (primary) key attribute**.
- cost of binary search = **$[\log_2(b_r)]$**
  - b_r denotes number of blocks containing records from relation r
- If the selection is on **non (primary) key attribute** then **multiple block may contains required records**, then the **cost of scanning such blocks need to be added to the cost estimate**.
- This algorithm is **faster** than linear search algorithm.

Evaluation of expressions

- Expression may contain more than one operations, solving expression will be difficult if it contains more than one expression.

$$\Pi_{Cust\_Name}(\sigma_{Balance<2500}(account) \bowtie customer)$$

- To evaluate such expression we need to evaluate each operation one by one in appropriate order.



Evaluation of expressions

- Methods for evaluating an entire expression tree are:
  1. Materialization
  2. Pipelining

Materialization

- Materialization **evaluates the expression tree of the relational algebra operation from the bottom** and **performs the innermost or leaf-level operations first**.
- The **intermediate result of each operation is materialized** (store in temporary relation) and **becomes input for subsequent (next) operations**.
- The **cost of materialization is the sum of the individual operations plus the cost of writing the intermediate results to disk**.
- The problem with materialization is that
  - it **creates lots of temporary relations**
  - it **performs lots of I/O operations**

Pipelining

- In pipelining, **operations form a queue**, and **results are passed from one operation to another as they are calculated**.
- To reduce number of intermediate temporary relations, we pass results of one operation to the next operation in the pipelines.
- **Combining operations into a pipeline eliminates the cost of reading and writing temporary relations**.

Query optimization

- It is a **process of selecting the most efficient query evaluation plan** from the available possible plans.

$$\Pi_{Cust\_Name}(\sigma_{Balance<2500}(Account) \bowtie Customer)$$

Efficient plan

2 records

4 records

$$\Pi_{Cust\_Name}(\sigma_{Balance<2500}(Account \bowtie Customer))$$

4 records

4 records

Customer

Cid	Ano	Cust_name
C01	A01	Raj
C02	A02	Meet
C03	A03	Harsh
C04	A04	Punit

Account

Ano	Balance
A01	3000
A02	1000
A03	2000
A04	4000

Approaches to Query Optimization

1. Exhaustive Search Optimization
   - **Generates all possible query plans** and then the **best plan is selected**.
   - It **provides best solution**.
2. Heuristic Based Optimization
   - Heuristic based optimization uses rule-based optimization approaches for query optimization.
   - **Performs select and project operations before join operations**. This is done by moving the select and project operations down the query tree. This **reduces the number of tuples available for join**.
   - Avoid **cross-product operation** because they result in very large-sized intermediate tables.
   - This **algorithms do not necessarily produce the best query plan**.