Course code : **CSE2007**
Course title : **Database Management System**
Module : **4**
Topic : **1**

Query Processing

Objectives

This session will give the knowledge about

- Translating SQL Queries into Relational Algebra
- Algorithms for External Sorting

Introduction to Query Processing

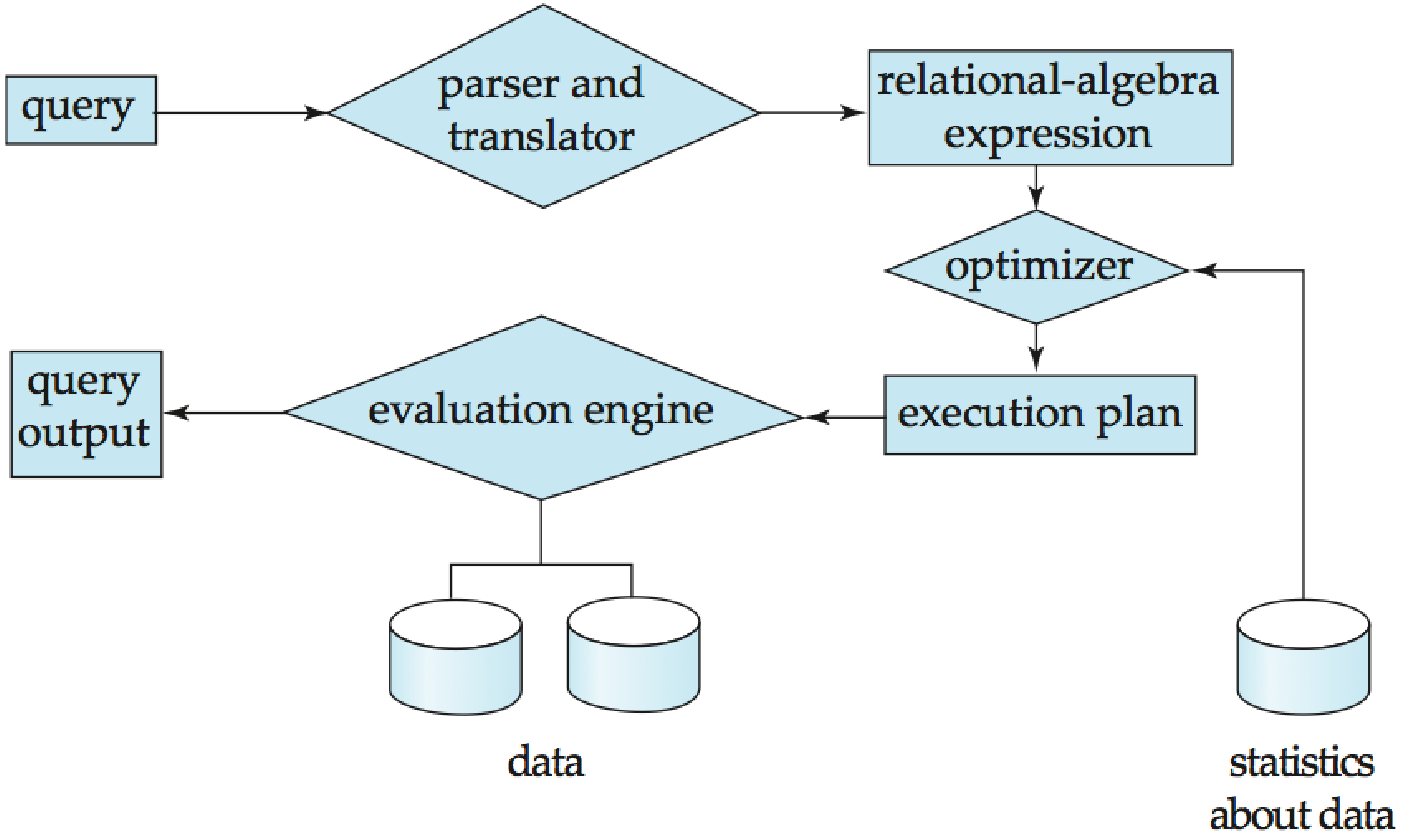
Query optimization:

The process of choosing a suitable execution strategy for processing a query.

Two internal representations of a query:

- Query Tree
- Query Graph

Introduction to Query Processing

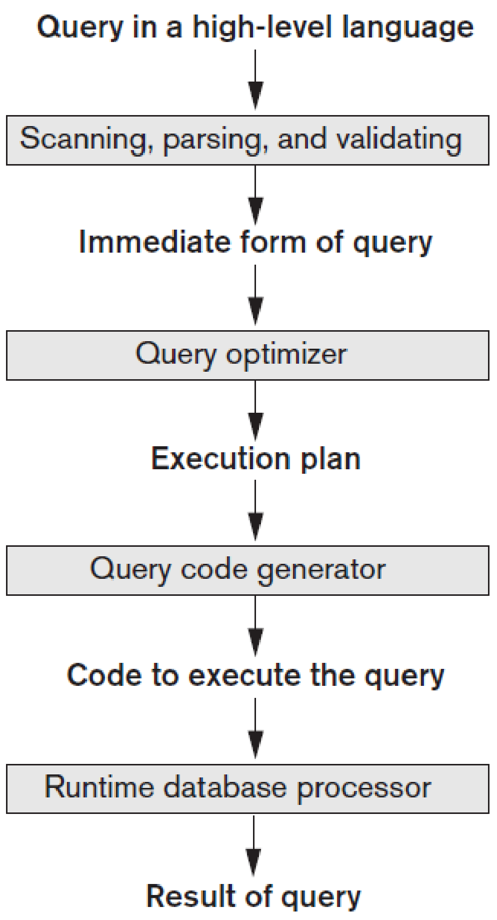


Introduction to Query Processing

Query processing is a set of all activities starting from query placement to displaying the results of the query.

Typical steps when processing a high-level query.

Executed directly (interpreted mode) Stored and executed later whenever needed (compiled mode)



Introduction to Query Processing

- A query expressed in a high-level query language such as SQL must first be scanned, parsed, and validated.
- The scanner identifies the query tokens - such as SQL keywords, attribute names, and relation names that appear in the text of the query
- The parser checks the query syntax to determine whether it is formulated according to the syntax rules (rules of grammar) of the query language.
- The query must also be validated by checking that all attribute and relation names are valid and semantically meaningful names in the schema of the particular database being queried.

Introduction to Query Processing

- An internal representation of the query is then created, usually as a tree data structure called a query tree.
- It is also possible to represent the query using a graph data structure called a query graph, which is generally a directed acyclic graph (DAG).
- The DBMS must then devise an execution strategy or query plan for retrieving the results of the query from the database files.
- A query has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as query optimization.

Introduction to Query Processing

- The term optimization is actually a misnomer because in some cases the chosen execution plan is not the optimal (or absolute best) strategy - it is just a reasonably efficient or the best available strategy for executing the query.

- Finding the optimal strategy is usually too time-consuming - except for the simplest of queries.

- Hence, planning of a good execution strategy may be a more accurate description than query optimization.

Introduction to Query Processing

There are two main techniques of query optimization.

- The first technique is based on heuristic rules for ordering the operations in a query execution strategy that works well in most cases but is not guaranteed to work well in every case.
- The rules typically reorder the operations in a query tree.
- The second technique involves cost estimation of different execution strategies and choosing the execution plan that minimizes estimated cost.

Translating SQL Queries into Relational Algebra

- Query block:

  The basic unit that can be translated into the algebraic operators and optimization.

- A query block contains a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clause if these are part of the block.

- Nested Queries within a query are identified as separate query blocks.

- Aggregate operators in SQL must be included in the extended algebra.

SQL Queries into Relational Algebra

```
SELECT    LNAME, FNAME
FROM  EMPLOYEE
WHERE     SALARY > (    SELECT    MAX (SALARY)
          FROM  EMPLOYEE
          WHERE     DNO = 5);
```

```
SELECT    LNAME, FNAME
FROM      EMPLOYEE
WHERE     SALARY > C
```

```
SELECT    MAX (SALARY)
FROM      EMPLOYEE
WHERE     DNO = 5
```

$$\pi_{\text{LNAME, FNAME}}(\sigma_{\text{SALARY>C}}(\text{EMPLOYEE}))$$

$$\mathcal{F}_{\text{MAX SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$$

Additional Operators: Semi-Join and Anti-Join

Semi-join is generally used for unnesting EXISTS, IN, and ANY subqueries

EMPLOYEE (Ssn, Bdate, Address, Sex, Salary, Dno)
DEPARTMENT (Dnumber, Dname, Dmgrssn, Zipcode)

```
Q (SJ) :  SELECT COUNT(*)
FROM      DEPARTMENT D
WHERE     D.Dnumber IN ( SELECT E.Dno
                         FROM   EMPLOYEE E
                         WHERE  E.Salary > 200000)
```

```
( SELECT  E.Dno
  FROM    EMPLOYEE E    WHERE E.Salary > 200000)
```

```
SELECT COUNT(*)
FROM    EMPLOYEE E, DEPARTMENT D
WHERE   D.Dnumber S= E.Dno and E.Salary > 200000;
```

Semi-join

$\ltimes$

Additional Operators Semi-Join and Anti-Join

Anti-join is generally used for unnesting NOT IN subqueries

Anti-join

$\rhd$

```
Q (AJ) :  SELECT COUNT(*)
FROM      EMPLOYEE
WHERE     EMPLOYEE.Dno NOT IN (SELECT   DEPARTMENT.Dnumber
                               FROM     DEPARTMENT
                               WHERE    Zipcode =30332)
```

↓

```
SELECT COUNT(*)
FROM    EMPLOYEE, DEPARTMENT
WHERE   EMPLOYEE.Dno A= DEPARTMENT AND Zipcode =30332
```

Algorithms for External Sorting

External sorting:

Refers to sorting algorithms that are suitable for large files of records stored on disk that do not fit entirely in main memory, such as most database files.

Sort-Merge strategy:

- Starts by sorting small subfiles (runs) of the main file and then merges the sorted runs, creating larger sorted subfiles that are merged in turn.

- Sorting phase: $n_R = \left\lceil \left(\frac{b}{n_B} \right) \right\rceil$

Algorithms for External Sorting

- Merging phase:
  - $d_m = Min(n_B - 1, n_R\ )$
  - $n_P = \lceil(\log_{dM}(n_R))\rceil$

  Where

    - n_R: number of initial runs;
    - b: number of file blocks;
    - n_B: available buffer space;
    - d_M: degree of merging;
    - n_P: number of passes.

Outline of the sort-merge algorithm

set $i \leftarrow 1;$

$j \leftarrow b;$ {size of the file in blocks}

$k \leftarrow n_B;$ {size of buffer in blocks}

$m \leftarrow \lceil (j/k) \rceil;$ {number of subfiles- each fits in buffer}

{Sorting Phase}

while ($i \leq m$)

do {

read next k blocks of the file into the buffer or if there are less than k blocks remaining, then read in the remaining blocks;

sort the records in the buffer and write as a temporary subfile;

$i \leftarrow i + 1;$

}

Outline of the sort-merge algorithm

{Merging Phase: merge subfiles until only 1 remains}

set $i \leftarrow 1$;
$p \leftarrow \lceil \log_{k-1} m \rceil$ {p is the number of passes for the merging phase}
$j \leftarrow m$;
while ($i \leq p$)
do {
$n \leftarrow 1$;
$q \leftarrow (j/(k-1) \rceil$; {number of subfiles to write in this pass}
while ($n \leq q$)
do {
read next $k-1$ subfiles or remaining subfiles (from previous pass)
one block at a time;
merge and write as new subfile one block at a time;
$n \leftarrow n + 1$;
}
$j \leftarrow q$;
$i \leftarrow i + 1$;
}

Summary

This session will give the knowledge about

- Translating SQL Queries into Relational Algebra
- Algorithms for External Sorting