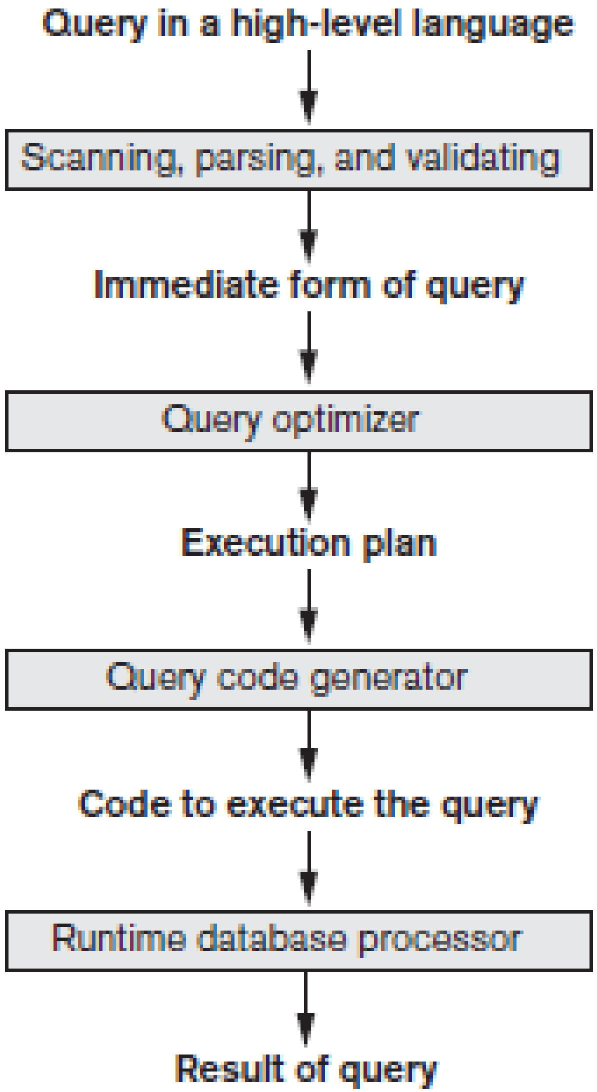
Database Management Systems II

Query processing and Optimization

Steps of processing a high level query



Scanner

Identifies the query tokens—such as SQL keywords, attribute names, and relation names—that appear in the text of the query,

Parser

Checks the query syntax to determine whether it is formulated according to the syntax rules (rules of grammar) of the query language.

Validated

Checking that all attribute and relation names are valid and semantically meaningful names in the schema of the particular database being queried.

Validated

Validated by checking that all attribute and relation names are valid and semantically meaningful names in the schema of the particular database being queried.

Query Optimization

- 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 strategy* for executing the query
- For lower-level navigational database languages in legacy systems (network DML or the hierarchical DL/1)the programmer must choose the query execution strategy while writing a database program.

Query Optimization Cont..

- A high-level query language (SQL or OQL) for object DBMSs (ODBMSs)—is more declarative in nature because it specifies what the intended results of the query are, rather than identifying the details of *how* the result should be obtained.

Techniques to Query Optimization

- The first technique is based on **heuristic rules** for ordering the operations in a query execution strategy. A heuristic is a rule that works well in most cases but is not guaranteed to work well in every case.
- The second technique involves **systematically estimating** the cost of different execution strategies and choosing the execution plan with the lowest cost estimate.

Steps for Query Optimization

- SQL Query → Relation Algebra → Query Tree → Optimized

Query → Relation Algebra

Operation	My HTML	Symbol
Projection	**PROJECT**	π
Selection	**SELECT**	σ
Renaming	**RENAME**	ρ
Union	**UNION**	$\cup$
Intersection	**INTERSECTION**	$\cap$
Assignment	<-	$\leftarrow$

Operation	My HTML	Symbol
Cartesian product	**X**	$\times$
Join	**JOIN**	$\bowtie$
Left outer join	**LEFT OUTER JOIN**	⟕
Right outer join	**RIGHT OUTER JOIN**	⟖
Full outer join	**FULL OUTER JOIN**	⟗
Semijoin	**SEMIJOIN**	$\ltimes$

Query → Relation Algebra Cont…

- SELECT clause attributes are mapped to the root as a Project operation.
- WHERE clause condition is the next level as a Select operation.
- Relations of the FROM clause are joined as Cartesian product.

Example

Employee Table

no	name	salary
1	John	100
5	Sarah	300
7	Tom	100

- View name of employees
- View name, salary of employees
- View name of employees who has salary more than 200

Example 02

SELECT p.pno, d.dno, e.ename FROM Project as p, Department as d, Employee as e WHERE d.dno=p.dept and d.mgr=e.empno and p.location=‘Colombo’;

- T1 $\leftarrow$ Project $\infty_{dno=dept}$ Department
- T2 $\leftarrow$ T1 $\infty_{mgr=empno}$ Employee
- T3 $\leftarrow$ $\sigma_{location='Colombo'(}$T2)
- Result $\leftarrow$ $\pi_{pno,\ dno,\ ename}$(T3)

Day 2

```
SELECT Orders.OrderID,
Customers.CustomerName
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID = Customers.CustomerID
WHERE Customer.Age>30 ;
```

Transformation rules for relational algebra operations

- There are many rules for transforming relational algebra operations into equivalent ones.
- These are in addition to those discussed under relational algebra.
- These rules are used in heuristic optimization.
- Algorithms that utilize these rules are used to transform an initial query tree into an optimized tree that is more efficient to execute.

Rule 1 (cascade of σ)

- Break up any SELECT operations (σ) with conjunctive conditions (AND) into a cascade (sequence) of individual SELECT operations.

$$\sigma_{c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$$

- $\sigma_{\text{location = 'Colombo' and age > 50}}$ (Employee)
- $\sigma_{\text{location='Colombo'}}$ (σ $_{\text{age> 50}}$ (Employee))

Rule 2 (Commutative of σ)

- The SELECT operation is commutative
- $\sigma c1(\sigma c2(R)) = \sigma c2(\sigma c1(R))$

Rule 3 (Commutative of σ with π)

- If the SELECT condition c involves only attributes a1, a2, …, an in the PROJECTION list, the two operations can be commuted.

- $\pi_{a1, a2, \ldots, an} (\sigma_c (R)) = \sigma_c (\pi_{a1, a2, \ldots, an} (R))$

Rule 4 (commutative of σ with X or ∞)

If all the attributes in the selection condition c involve only the attributes of one of the relations being joined (say R) the two operations can be commuted as

c (R□S) □ (□c (R)) □ S