CSC 430 – DATABASE MANAGEMENT SYSTEM

LECTURE 19: Query Processing and Optimization

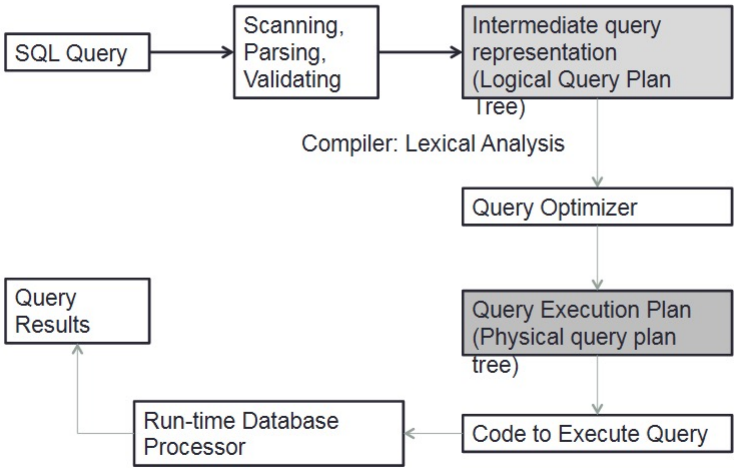
1

Query Processing

- Efficient Query Processing crucial for good or even effective operations of a database
- Query Processing depends on a variety of factors, not everything under the control of DBMS
- Insufficient or incorrect information can result in vastly ineffective plans
- Query Cataloging

2

Steps in Query Execution



3

Query Processing

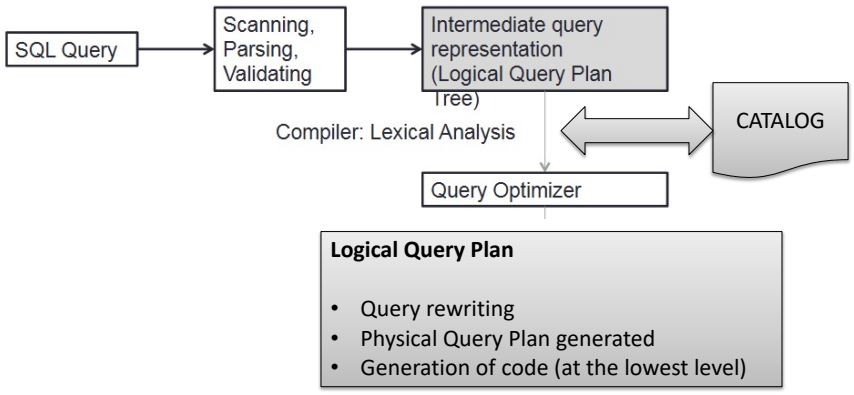
- Intermediate Form / Intermediate Query Plan
  - Usually a relational algebra form of the SQL query
  - Uses heuristics and cost-based measures for optimization
- Physical Query Plan
  - In a language that is interpreted and executed on the machine or compiled into machine code

4

INTERMEDIATE FORM OR LOGICAL QUERY PLAN

5

Steps in Query Processing



6

Query Optimization

- Based on rewriting Parse Tree representing a relational algebra expression of the query
- Heuristics-based optimization Vs Cost-based optimization

7

Parse Trees

- Syntactic structures of most programming languages can be expressed in the form of a “syntax tree” also called the “parse tree”
- Execution of a syntactic construct can be achieved by a “post-order traversal” of a parse tree

8

Example: Parse Tree

- For every project located in 'Stafford', retrieve the project number, the controlling department number and the department manager's last name, address and birthdate.

- SQL query:

```
Q2: SELECT P.NUMBER,P.DNUM,E.LNAME,
          E.ADDRESS, E.BDATE
    FROM  PROJECT AS P,DEPARTMENT AS D,
                EMPLOYEE AS E
    WHERE P.DNUM=D.DNUMBER AND
    D.MGRSSN=E.SSN AND P.PLOCATION='STAFFORD';
```

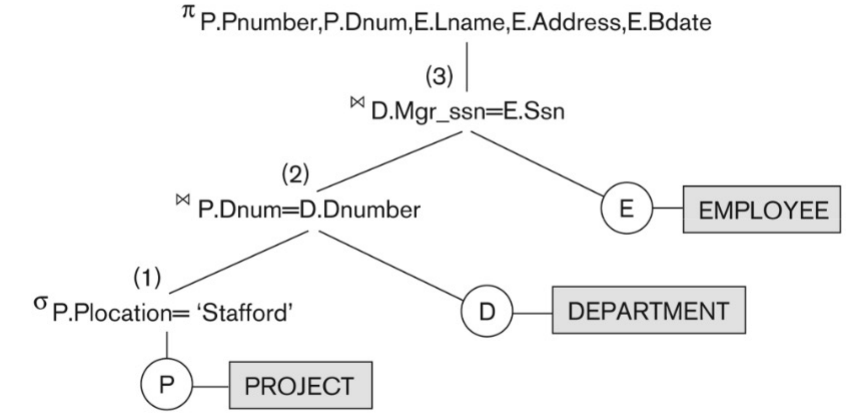
- Relation algebra:

$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}$ (($\sigma_{\text{PLOCATION='STAFFORD'}}$(PROJECT))

⋈ ►◄ DNUM=DNUMBER (DEPARTMENT)) ►◄MGRSSN=SSN (EMPLOYEE))

9

Corresponding parse tree



$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}$ (($\sigma_{\text{PLOCATION='STAFFORD'}}$(PROJECT))

⋈ ►◄ DNUM=DNUMBER (DEPARTMENT)) ►◄MGRSSN=SSN (EMPLOYEE))

10

Checks on Parse Tree

- Syntactic Checks: Is the syntax of every operator correct?
- Entity checks: Does every relation name refer to a valid relation?
- View Expansion: If a relation name refers to a view, replace the relation node with the parse tree of the view
- Attribute checks: Does every attribute name refer to valid attributes?
- Type checks: Does each attribute participating in an expression have the proper type?

11

Rewriting Parse Trees

- Queries are optimized by rewriting parse trees
- Rewriting parse trees is guided by a set of **rewrite rules**
- Parse tree should be **expanded to its maximum** extent before rewriting
- Some rewrite rules are situation specific: they work if certain conditions hold on the data set.

12

Rewrite Thumb Rules

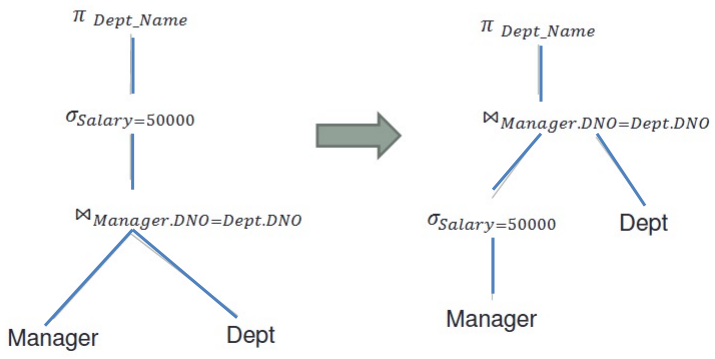
a) Pushing Selects (σ) and Exceptions
   b. Cascading / Conjunctive Selects
   c. Expanding Views
   d. Exceptions with Joins

b) Inserting Projects (Π)

13

Rewrite Rule: (a) Pushing Selects

- Since a select statement reduces the size of a relation they can be pushed as far down a parse tree as possible.



14

Rewrite Rule: (a. b) Cascading Selects

- Conjunctive selects can be split and pushed to form cascading selects that progressively reduce relation size:

$$\sigma_{C \wedge D}(\mathrm{R}) \Leftrightarrow \sigma_C(\sigma_D(R))$$

15

(a.c) Exception to Pushing Selects (VIEWS)

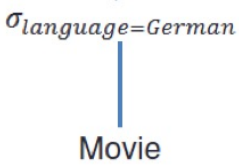
- When a query contains a view, Selects may have to be first moved up before they are moved down:

Consider relations:

- Movie (title, year, director, language)
- Starsin (title, year, StarName, language)

and the view:

```
CREATE VIEW GermanMovies AS
    SELECT *
    FROM Movie
    Where language = 'German';
```

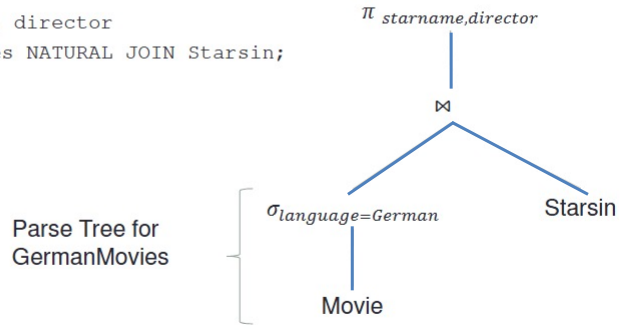


16

Exception to Pushing Selects

- Consider the query: *List all the stars and their corresponding directors in all German movies?*

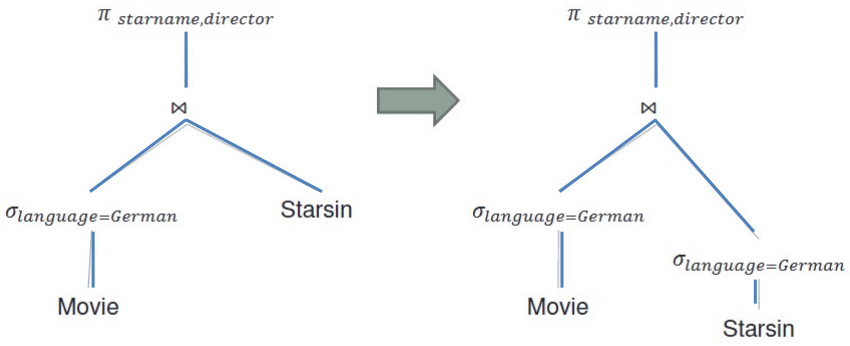
```
SELECT starname, director
FROM GermanMovies NATURAL JOIN Starsin;
```



Parse Tree for GermanMovies

17

Exception to Pushing Selects



18

(a.d) Pushing Selects (in JOINS)

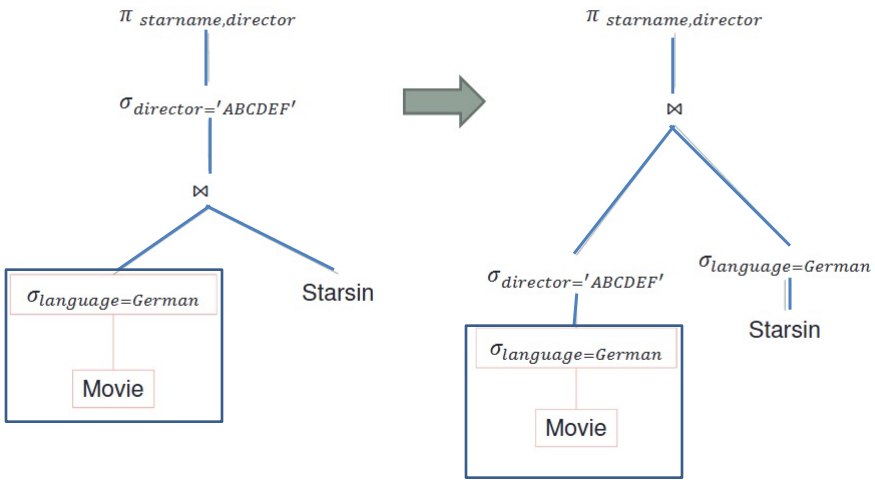
- If select over a join involves attributes of only one of the relations, move select below the join.
- Consider the following query over the Movies database comprising of Movie , Starsin and GermanMovies relations.
- Which stars acted under the direction of ABCDEF in GermanMovies,
- In SQL:

```
SELECT starname FROM
GermanMovies NATURAL JOIN Starsin WHERE
director ='ABCDEF'
```

19

Pushing Selects

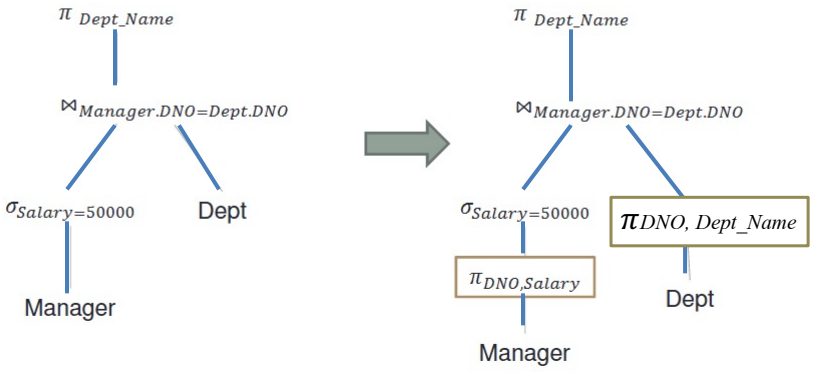
```
SELECT starname FROM
GermanMovies NATURAL JOIN Starsin WHERE
director ='ABCDEF'
```



20

Rewrite Rule: (b)Inserting Projects

- Extra projects can be added near the leaves of the parse tree to reduce the size of tuples going up the tree:



21

II. Cost-Based Optimization

- Factors affecting query cost:
  - Access cost to secondary storage
  - Storage cost of intermediate files
  - Computation cost
  - Memory usage cost
  - Communication costs (between the DBMS server and its client)

22

Catalogs

- Catalogs in a Database store information for cost estimation
- Catalogs are meta-data that could be either:
  - Table specific
  - Field specific
  - Index specific
  - Database wide

23

Catalog Examples

- B(R) – Number of blocks taken by relation R
- T(R) – Number of tuples in relation R
- V(R,a) – Number of distinct values relation R has for value a.
  - V(R, [a1,a2,…, an]) is the number of distinct values relation R has for the combined set of attributes a1,a2,…, an.

24

Example (1) cost estimation techniques

- Estimating the cost selection.

- Consider a select of the form: $S = \sigma_{A=c}(R)$, where
  c is a constant and
  A is an attribute of R.
- T(S) = estimate of the number of tuples in S
  = $\left(\frac{T(R)}{V(R,'c')}\right)$

Gives a good estimate if all values of A have uniform probabilities of occurrence (in the select query)

25

Example (2) cost estimation techniques

- Estimating the cost selection.

- Consider a select of the form: $S = \sigma_{A\neq c}(R)$, where
  c is a constant and
  A is an attribute of R.
- T(S) = estimate of the number of tuples in S
  = $1 - \left(\frac{T(R)}{V(R,'c')}\right) = \left(V(R,\ 'c') - T(R)\right)/V(R,\ 'c')$

Gives a good estimate if all values of A have uniform probabilities of occurrence (in the select query)

26

Summary

- Query Optimization by rewriting parse tree
  - Pushing selects
  - Cascading selects
  - Pulling selects from views
  - Extra projects
- Cost estimation of query components based on catalog information

27