Database Management Systems (DBMS)

Lec 19: Query processing and optimization

Ramesh K. Jallu
IIIT Raichur

Date: 06/04/21

Recap

- Algorithm to compute a minimal cover
- Algorithm to find a candidate key
- Dependency-preserving
- Algorithm to test NJ property

Query Processing

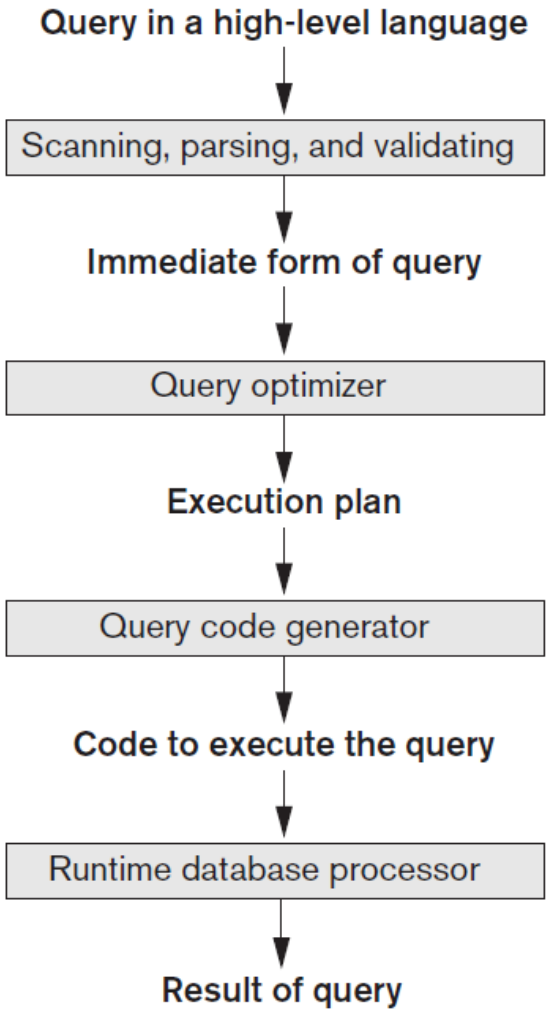
- Queries are expressed in a high-level query language such as SQL
- Processing a query we mean ***scanning***, ***parsing***, and ***validating*** the query
  - The scanner identifies the query tokens: SQL keywords, attribute names, and relation names, etc.
  - The parser checks the query syntax to determine whether it is formulated according to the syntax rules of the query language
  - The query is validated by checking that all attribute and relation names are valid

Query Processing (Contd.)

- An internal representation of the query is then created as a tree or graph
- The DBMS then devises an ***execution strategy*** or ***query plan*** for retrieving the results of the query from the database files
- An evaluation plan defines (i) what algorithm should be used for each operation, and (ii) how the execution of the operations should be coordinated
- A query has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as ***query optimization***

Query optimizaiton

- ***Query optimization*** is the process of selecting the most efficient query processing plan from among the many strategies
  - This task is done by ***query optimizer***
  - The ***code generator*** generates the code to execute that plan
  - The ***runtime database processor*** executes the query code, whether in ***compiled*** or ***interpreted mode***, to produce the query result



A few aspects of query optimization

- We do not expect users to write their queries so that they can be processed efficiently

- The DB system is expected to construct a query-evaluation plan that *minimizes the cost* of query evaluation
  - In order to do this, a query optimizer must know the cost of each operation
  - Cost is estimated using statistical information from the database catalog, e.g. number of tuples in each relation, size of tuples, etc.

  - The system attempts to find an expression that is equivalent to the given expression, but more efficient to execute

  - Also, the system selects a detailed strategy for processing the query

Goals of this chapter

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

How to measure query costs

- Many factors contribute to measure in terms of a number of different resources
  - Including disk accesses, CPU time to execute a query, the cost of communication
- Cost can be measured based on ***response time*** and ***total resource consumption***
- Databases resident on magnetic disks
  - The I/O cost to access data from disk usually dominates the other costs
  - Early cost models focused on the I/O cost when estimating the cost

How to measure query costs (Contd.)

- As flash storage becoming larger and less expensive, data can be stored on SSDs
- Flash storage achieve very fast response time, thus I/O cost does not dominate
- The CPU cost involves cost per tuple, cost for processing each index entry, cost for an arithmetic operation, comparison operation, etc.
- The main factors in estimating the I/O cost of a query-evaluation plan are (i) the number of blocks transferred from storage, and (ii) the number of random I/O accesses

How to measure query costs (Contd.)

- To estimate I/O cost we consider ***number of blocks transferred*** from storage and the ***number of random I/O accesses***

- Both the operations require a disk seek on magnetic storage
  - t_T - time to transfer one block of data
  - t_S - average block-access time (disk seek time + rotational latency)
  - An operation that transfers b blocks and performs S random I/O accesses takes $b * t_T + S * t_S$ seconds

- Although SSDs do not perform a physical seek operation, they have an overhead for initiating an I/O operation
- For data that are already present in main memory, reads happen at the unit of cache lines, instead of disk blocks

How to measure query costs (Contd.)

- We can reduce disk I/O cost by using extra buffer space
  - Amount of real memory available to buffer depends on other concurrent queries and OS processes
- The response time depends on the contents of the buffer when the query begins execution
  - This information is not available when the query is optimized and is hard to account for even if it were available
- In a system with multiple disks, the response time depends on how accesses are distributed among disks
  - Which is hard to estimate without detailed knowledge of data layout on disk

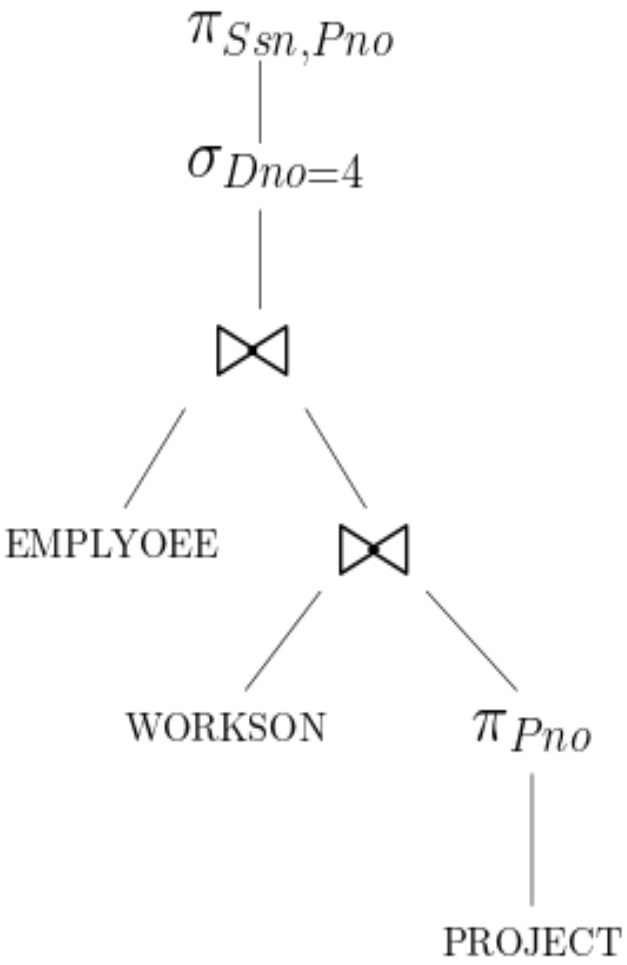
Translating SQL Queries into RA expression

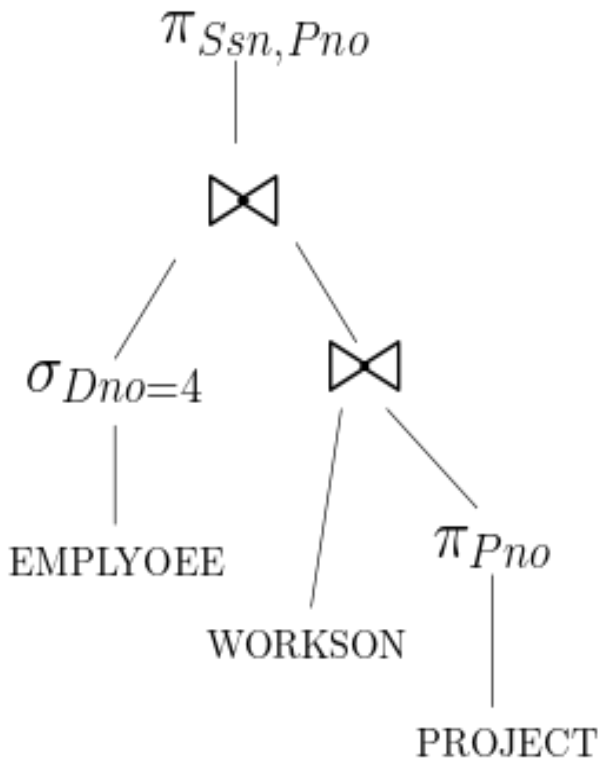
- We will concentrate on describing query processing and optimization in the context of an RDBMS
- The SQL is used in most commercial RDBMS
- An SQL query is first translated into an equivalent extended relational algebra expression and then optimized
  - The expression is represented as a ***query tree*** data structure

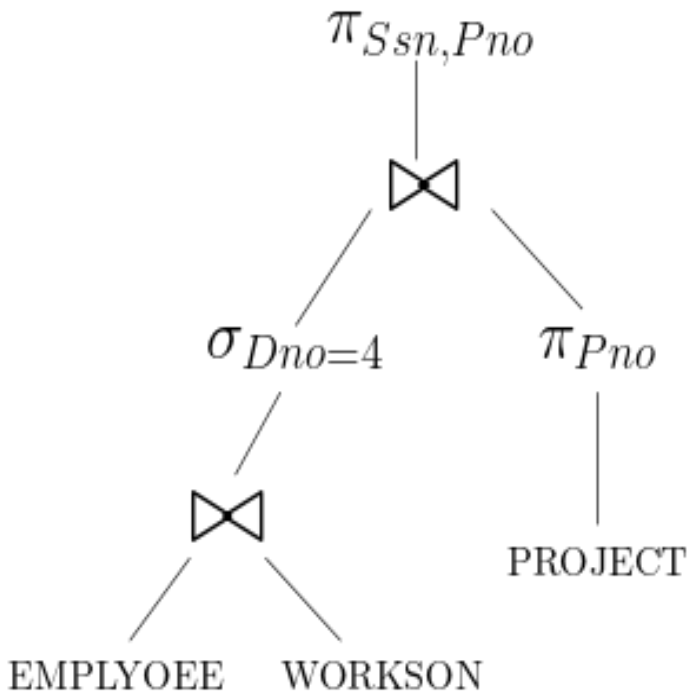
Translating SQL Queries into RA expression

- Find SSNs of all employees in department number 4 together with the project numbers of all the projects that the employee works on
- $Q_1 : \pi_{\text{Ssn,Pno}}(\sigma_{\text{Dno} = 4}\ (\text{EMPLYOEE} \bowtie (\text{WORKSON} \bowtie \pi_{\text{Pno}}(\text{PROJECT}))))$
- $Q_2 : \pi_{\text{Ssn,Pno}}(\sigma_{\text{Dno} = 4}\ (\text{EMPLYOEE}) \bowtie (\text{WORKSON} \bowtie \pi_{\text{Pno}}\ (\text{PROJECT})))$
- $Q_3 : \pi_{\text{Ssn,Pno}}((\sigma_{\text{Dno} = 4}\ (\text{EMPLYOEE}) \bowtie \text{WORKSON}) \bowtie \pi_{\text{Pno}}\ (\text{PROJECT}))$

Query tree


$\pi_{Ssn,Pno}$
$\sigma_{Dno=4}$
EMPLYOEE
WORKSON
π_{Pno}
PROJECT


$\pi_{Ssn,Pno}$
$\sigma_{Dno=4}$
EMPLYOEE
WORKSON
π_{Pno}
PROJECT


$\pi_{Ssn,Pno}$
$\sigma_{Dno=4}$
π_{Pno}
EMPLYOEE
WORKSON
PROJECT

Additional relational operations

1. Generalized projection

- It extends the projection operation by allowing functions of attributes to be included in the projection list
- The generalized form can be expressed as: $\pi_{F1, F2, ..., Fn}(R)$
- F1, F2, … , Fn are functions over the attributes in relation R and may involve arithmetic operations and constant values

Example

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show

Net Salary = Salary – Deduction,
Bonus = 2000 * Years_service, and
Tax = 0.25 * Salary

Then a generalized projection combined with renaming may be used as follows:

$$\text{REPORT} \leftarrow \rho_{(\text{Ssn, Net\_salary, Bonus, Tax})}(\pi_{\text{Ssn, Salary} - \text{Deduction, } 2000 * \text{Years\_service, } 0.25 * \text{Salary}}(\text{EMPLOYEE}))$$

Additional relational operations (Contd.)

1. Aggregate Functions and Grouping

- Aggregate functions are used in simple statistical queries that summarize information from the database tuples
  - For example, SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT
- We can group the tuples in a relation by the attribute value and then applying an aggregate function independently to each group
- Aggregate functions are denoted by <grouping attributes> ℑ <function list>®
- The resulting relation has the grouping attributes plus one attribute for each element in the function list

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(\text{Dno } \mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary } (\text{EMPLOYEE}))$

b. $\text{Dno } \mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$

c. $\mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$

Additional relational operations (Contd.)

2. OUTER JOIN operation

- It is a join operation with an extension
- The JOIN operation matches tuples that satisfy the join condition
- In NATURAL JOIN operation R * S, only tuples from R that have matching tuples in S and vice versa appear in the result
- Tuples without a matching tuple or tuples with NULL values in the join attributes are also eliminated
- This type of join is known as an inner join

Additional relational operations (Contd.)

- OUTER JOIN keeps all the tuples in R, or all those in S, or all those in both relations in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation
- The LEFT OUTER JOIN, denoted by ⟕, operation keeps every tuple in the first, or left, relation R in R⟕S
- The RIGHT OUTER JOIN, denoted by ⟖, keeps every tuple in the second, or right, relation S in the result of R⟖S
- The FULL OUTER JOIN, denoted by ⟗, keeps all tuples in both the left and the right relations
- When no matching tuples are found in R and/or S, padding them with **NULL** values as needed

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

list of all employee names as well as the name of the departments they manage if they happen to manage a department; if they do not manage one, we can indicate it with NULL

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

$$TEMP \leftarrow (EMPLOYEE \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)$$
$$RESULT \leftarrow \pi_{Fname,\ Minit,\ Lname,\ Dname}(TEMP)$$

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Thank you!