# Lecture 20:
# Query Execution: Relational Algebra

Friday, November 17, 2006

1

## DBMS Architecture

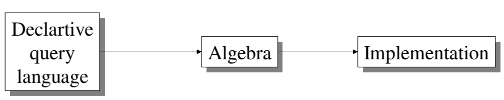
How does a SQL engine work ?

- SQL query → relational algebra plan
- Relational algebra plan → Optimized plan
- Execute each operator of the plan

2

## Relational Algebra

- Formalism for creating new relations from existing ones
- Its place in the big picture:




SQL,
relational calculus

Relational algebra
Relational bag algebra

3

## Relational Algebra

- Five operators:
  - Union: $\cup$
  - Difference: -
  - Selection: $\sigma$
  - Projection: $\Pi$
  - Cartesian Product: $\times$
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: $\rho$

4

## 1. Union and 2. Difference

- R1 $\cup$ R2
- Example:
  - ActiveEmployees $\cup$ RetiredEmployees

- R1 – R2
- Example:
  - AllEmployees -- RetiredEmployees

5

## What about Intersection ?

- It is a derived operator
- R1 $\cap$ R2 = R1 – (R1 – R2)
- Also expressed as a join (will see later)
- Example
  - UnionizedEmployees $\cap$ RetiredEmployees

6

## 3. Selection

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{"Smith"}}$ (Employee)
- The condition c can be =, <, ≤, >, ≥, <>

7

| SSN | Name | Salary |
|---|---|---|
| 1234545 | John | 200000 |
| 5423341 | Smith | 600000 |
| 4352342 | Fred | 500000 |

$\sigma_{Salary > 40000}$ (Employee)

| SSN | Name | Salary |
|---|---|---|
| 5423341 | Smith | 600000 |
| 4352342 | Fred | 500000 |

8

## 4. Projection

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Example: project social-security number and names:
  - $\Pi_{SSN, Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

9

| SSN | Name | Salary |
|---|---|---|
| 1234545 | John | 200000 |
| 5423341 | John | 600000 |
| 4352342 | John | 200000 |

$\Pi_{Name,Salary}$ (Employee)

| Name | Salary |
|---|---|
| John | 20000 |
| John | 60000 |

10

## 5. Cartesian Product

- Each tuple in R1 with each tuple in R2
- Notation: $R1 \times R2$
- Example:
  - Employee × Dependents
- Very rare in practice; mainly used to express joins

11

**Cartesian Product Example**

**Employee**

| Name | SSN |
|---|---|
| John | 999999999 |
| Tony | 777777777 |

**Dependents**

| EmployeeSSN | Dname |
|---|---|
| 999999999 | Emily |
| 777777777 | Joe |

**Employee x Dependents**

| Name | SSN | EmployeeSSN | Dname |
|---|---|---|---|
| John | 999999999 | 999999999 | Emily |
| John | 999999999 | 777777777 | Joe |
| Tony | 777777777 | 999999999 | Emily |
| Tony | 777777777 | 777777777 | Joe |

12

## Relational Algebra

- Five operators:
  - Union: $\cup$
  - Difference: -
  - Selection: $\sigma$
  - Projection: $\Pi$
  - Cartesian Product: $\times$
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: $\rho$

13

## Renaming

- Changes the schema, not the instance
- Notation: $\rho_{B1,\ldots,Bn}$ (R)
- Example:
  - $\rho_{LastName,\ SocSocNo}$ (Employee)
  - Output schema:
    Answer(LastName, SocSocNo)

14

**Renaming Example**

**Employee**

| Name | SSN |
|---|---|
| John | 999999999 |
| Tony | 777777777 |

$\rho_{LastName,\ SocSocNo}$ **(Employee)**

| LastName | SocSocNo |
|---|---|
| John | 999999999 |
| Tony | 777777777 |

15

## Natural Join

- Notation: R1 |x| R2
- Meaning: R1 |x| R2 = $\Pi_A(\sigma_C(R1 \times R2))$
- Where:
  - The selection $\sigma_C$ checks equality of all common attributes
  - The projection eliminates the duplicate common attributes

16

**Natural Join Example**

**Employee**

| Name | SSN |
|---|---|
| John | 999999999 |
| Tony | 777777777 |

**Dependents**

| SSN | Dname |
|---|---|
| 999999999 | Emily |
| 777777777 | Joe |

**Employee** $\bowtie$ **Dependents =**

$\Pi_{Name,\ SSN,\ Dname}(\sigma_{SSN=SSN2}(Employee \times \rho_{SSN2,\ Dname}(Dependents))$

| Name | SSN | Dname |
|---|---|---|
| John | 999999999 | Emily |
| Tony | 777777777 | Joe |

17

## Natural Join

- R=

| A | B |
|---|---|
| X | Y |
| X | Z |
| Y | Z |
| Z | V |

S=

| B | C |
|---|---|
| Z | U |
| V | W |
| Z | V |

- R |x| S=

| A | B | C |
|---|---|---|
| X | Z | U |
| X | Z | V |
| Y | Z | U |
| Y | Z | V |
| Z | V | W |

18

## Natural Join

- Given the schemas R(A, B, C, D), S(A, C, E), what is the schema of $R \bowtie S$ ?
- Given R(A, B, C), S(D, E), what is $R \bowtie S$ ?
- Given R(A, B), S(A, B), what is $R \bowtie S$ ?

19

## Theta Join

- A join that involves a predicate
- $R1 \bowtie_{\theta} R2 = \sigma_{\theta} (R1 \times R2)$
- Here $\theta$ can be any condition

20

## Eq-join

- A theta join where $\theta$ is an equality
- $R1 \bowtie_{A=B} R2 = \sigma_{A=B} (R1 \times R2)$
- Example:
  - Employee $\bowtie_{SSN=SSN}$ Dependents
- Most useful join in practice

21

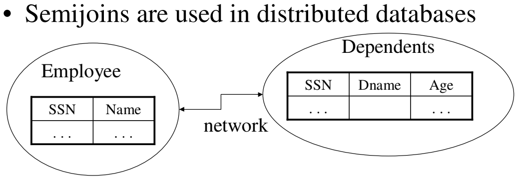
## Semijoin

- $R \ltimes S = \Pi_{A1,\ldots,An} (R \bowtie S)$
- Where $A_1, \ldots, A_n$ are the attributes in R
- Example:
  - Employee $\ltimes$ Dependents

22

## Semijoins in Distributed Databases

- Semijoins are used in distributed databases



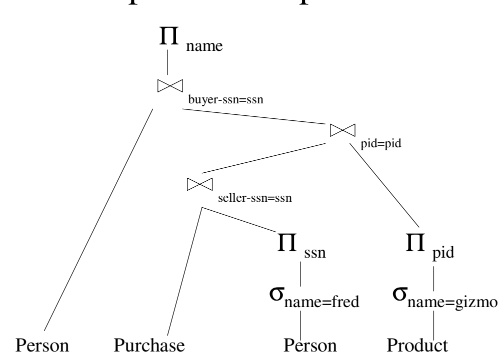

Employee $\bowtie_{ssn=ssn} (\sigma_{age>71}$ (Dependents))

R = Employee $\ltimes$ T

T = $\Pi_{SSN}\ \sigma_{age>71}$ (Dependents)

Answer = R $\bowtie$ Dependents

## Complex RA Expressions




24

## Operations on Bags

A **bag** = a set with repeated elements

All operations need to be defined carefully on bags

- {a,b,b,c}∪{a,b,b,b,e,f,f}={a,a,b,b,b,b,b,c,e,f,f}
- {a,b,b,b,c,c} – {b,c,c,c,d} = {a,b,b,d}
- $\sigma_C$(R): preserve the number of occurrences
- $\Pi_A$(R): no duplicate elimination
- Cartesian product, join: no duplicate elimination

Important ! Relational Engines work on bags, not sets !

Reading assignment: 5.3 – 5.4

25

## Note: RA has Limitations !

- Cannot compute "transitive closure"

| Name1 | Name2 | Relationship |
|---|---|---|
| Fred | Mary | Father |
| Mary | Joe | Cousin |
| Mary | Bill | Spouse |
| Nancy | Lou | Sister |

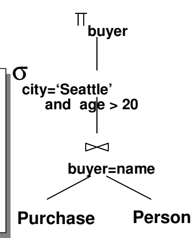
- Find all direct and indirect relatives of Fred
- Cannot express in RA !!! Need to write C program

26

## From SQL to RA

Purchase(buyer, product, city)
Person(name, age)

```
SELECT DISTINCT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
      P.city='Seattle' AND
     Q.age > 20
```

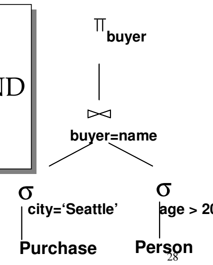



27

## Also…

Purchase(buyer, product, city)
Person(name, age)

```
SELECT DISTINCT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
      P.city='Seattle' AND
     Q.age > 20
```




28

## Non-monontone Queries (in class)

Purchase(buyer, product, city)
Person(name, age)

```
SELECT DISTINCT P.product
FROM Purchase P
WHERE P.city='Seattle' AND
   not exists (select *
               from Purchase P2, Person Q
               where P2.product = P.product
                  and P2.buyer = Q.name
                  and Q.age > 20)
```

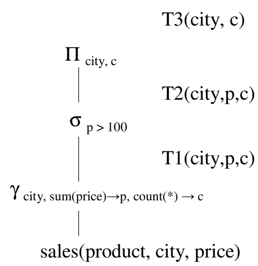
## Extended Logical Algebra Operators (operate on Bags, not Sets)

- Union, intersection, difference
- Selection $\sigma$
- Projection $\Pi$
- Join |x|
- Duplicate elimination $\delta$
- Grouping $\gamma$
- Sorting $\tau$

30

## Logical Query Plan

SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100




T1, T2, T3 = temporary tables

31

## Logical v.s. Physical Algebra

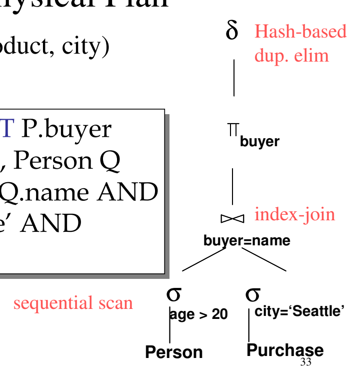
- We have seen the logical algebra so far:
  - Five basic operators, plus group-by, plus sort
- The Physical algebra refines each operator into a concrete algorithm

32

## Physical Plan

Purchase(buyer, product, city)
Person(name, age)

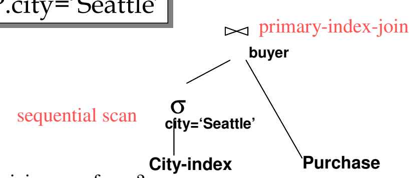
SELECT DISTINCT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
P.city='Seattle' AND
Q.age > 20




33

## Physical Plans Can Be Subtle

SELECT *
FROM Purchase P
WHERE P.city='Seattle'




Where did the join come from ?

34