CIS 560 – Database System Concepts

Lecture 13

Relational Algebra

September 25, 2013

Credits for slides: Suciu, Chang.

Outline

Last time:

- Transactions in SQL (6.6)

Today:

- Relational algebra (Sections 2.4 and 5.1-5.2)

Next:

- Introduction to Database Programming (Ch. 9)
  - Connect to DB and call SQL from Java

2

Review

- Transaction anomalies
- ACID properties

3

The WHAT and the HOW

- In SQL we write WHAT we want to get form the data
- The database system needs to figure out HOW to get the data we want
- The passage from WHAT to HOW goes through the Relational Algebra

4

SQL = WHAT

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

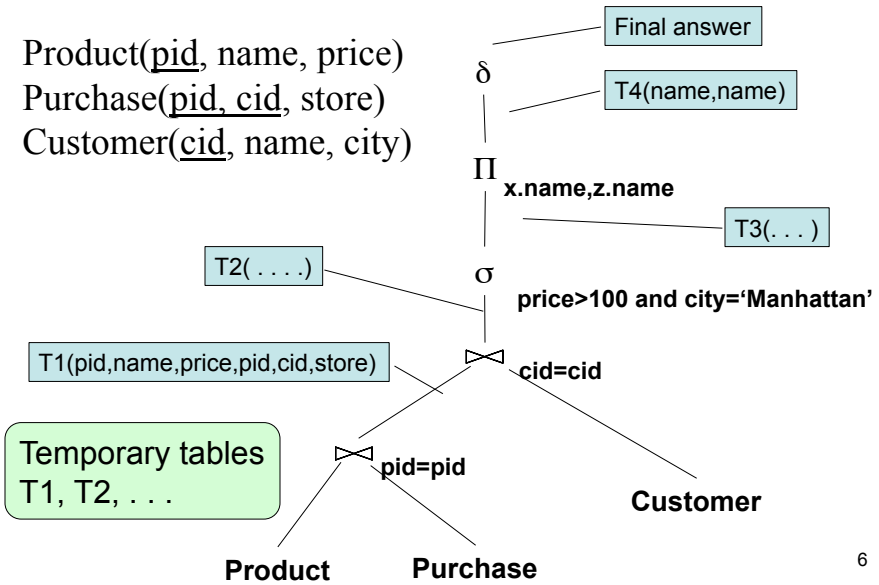
```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
      x.price > 100 and z.city = 'Manhattan'
```

It's clear WHAT we want, unclear HOW to get it

5

Relational Algebra = HOW

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)



6

Relational Algebra = HOW

The order is now clearly specified:

Iterate over PRODUCT…
…join with PURCHASE…
…join with CUSTOMER…
…select tuples with Price>100 and City='Manhattan'…
…eliminate duplicates…
…and that's the final answer !

7

Relational Algebra (1/3)

The Basic Five operators:

- Union: ∪
- Set difference: -
- Selection: $\sigma_{condition}(S)$
  - Condition is a Boolean combination (∧,∨) of terms
  - Term is: `attr op const, attr op attr`
  - Op is: <, <=, =, !=, >=, or >
- Projection: $\pi_{list\text{-}of\text{-}attributes}(S)$
- Cross-product or Cartesian product: ×

8

Relational Algebra (2/3)

Derived or auxiliary operators:

- Intersection ($\cap$)
- Join $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Variations of joins
  - natural, equijoin, theta join
  - outer-join and semi-join
- Rename $\rho_{B1,\ldots,Bn}(S)$

9

Relational Algebra (3/3)

Extensions for bags:

- Duplicate elimination: δ
- Group by: γ [Same symbol as aggregation]
  - Partitions tuples of a relation into "groups"
- Sorting: τ

Other extensions:

- Aggregation: γ (min, max, sum, average, count)

10

Different Types of Join

- Theta-join: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
- Equijoin: $R \bowtie_\theta S = \pi_A(\sigma_\theta(R \times S))$
  - Join condition θ consists only of equalities
  - Projection π_A drops all redundant attributes
- Natural join: $R \bowtie S = \pi_A(\sigma_\theta(R \times S))$
  - Equijoin
  - Equality on all fields with same name in R and in S

11

Theta-Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

age	zip	job
54	98125	lawyer
20	98120	cashier

$P \bowtie_{P.age=J.age \wedge P.zip = J.zip \wedge P.age<50} J$

P.age	P.zip	disease	job	J.age	J.zip
20	98120	flu	cashier	20	98120

12

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

age	zip	job
54	98125	lawyer
20	98120	cashier

$P \bowtie_{P.age=J.age} J$

P.age	P.zip	disease	job	J.zip
54	98125	heart	lawyer	98125
20	98120	flu	cashier	98120

13

Natural Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

age	zip	job
54	98125	lawyer
20	98120	cashier

$P \bowtie J$

P.age	P.zip	disease	job
54	98125	heart	lawyer
20	98120	flu	cashier

14

So Which Join Is It?

- When we write R ⋈ S we usually mean an equijoin - we often omit the equality predicate when it is clear from the context.

15

More Joins

- Outer join
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes
- Variants
  - Left outer join
  - Right outer join
  - Full outer join

16

Outer Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

AnonJob J

age	zip	job
54	98125	lawyer
20	98120	cashier

P ⟗ J

P.Age	P.zip	disease	job
54	98125	heart	lawyer
20	98120	flu	cashier
33	98120	lung	NULL

17

Joins

- The join operation and all its variants (equijoin, natural join, outer-join) are at the *heart* of relational database systems
- WHY?

18

Example of Algebra Queries

- Q1: Jobs of patients who have heart disease

19

Example of Algebra Queries

- Q1: Jobs of patients who have heart disease
  - π_{job}(AnonJob $\bowtie$ ($\sigma_{disease='heart'}$ (AnonPatient))

20

More Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Q2: Name of supplier of parts with size greater than 10

21

More Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Q2: Name of supplier of parts with size greater than 10
π_{sname}(Supplier ⋈ Supply ⋈($\sigma_{psize>10}$(Part))

22

More Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Q2: Name of supplier of parts with size greater than 10
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}))$

Q3: Name of supplier of red parts or parts with size greater than 10

23

More Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Q2: Name of supplier of parts with size greater than 10
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}))$

Q3: Name of supplier of red parts or parts with size greater than 10
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}) \cup \sigma_{pcolor='red'}(\text{Part})))$

24

Regular Algebra Expressions versus Programs

- An Algebra Expression is like a program
  - Several operations
  - Strictly specified order
- But relational algebra expressions have limitations

25

Limitations of Relational Algebra

- 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 Java program

26

From SQL to Relational Algebra

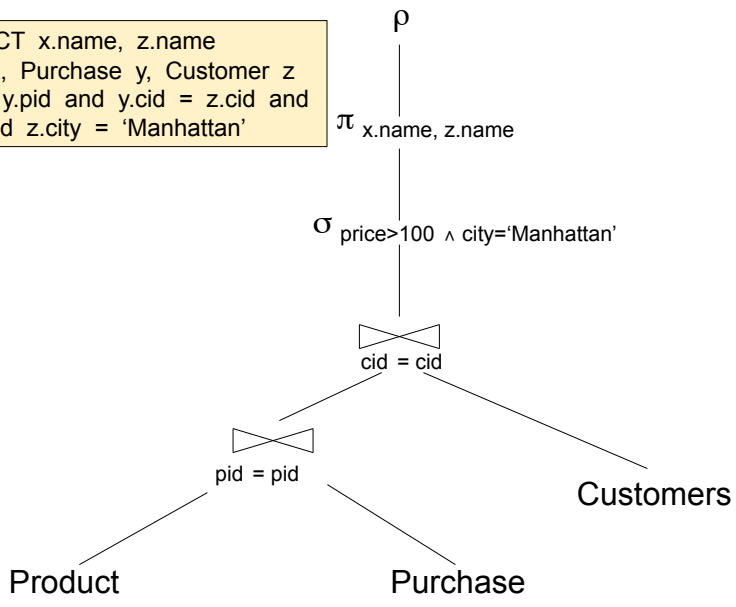
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
x.price > 100 and z.city = 'Manhattan'
```

27

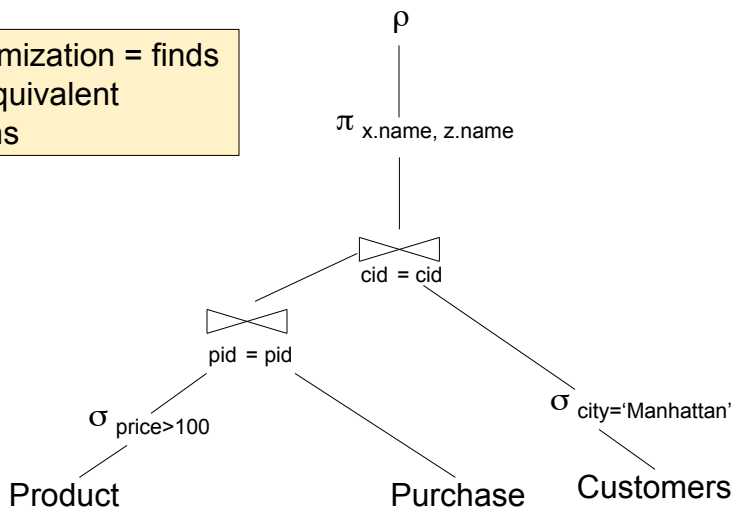
From SQL to Relational Algebra

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
x.price > 100 and z.city = 'Manhattan'
```

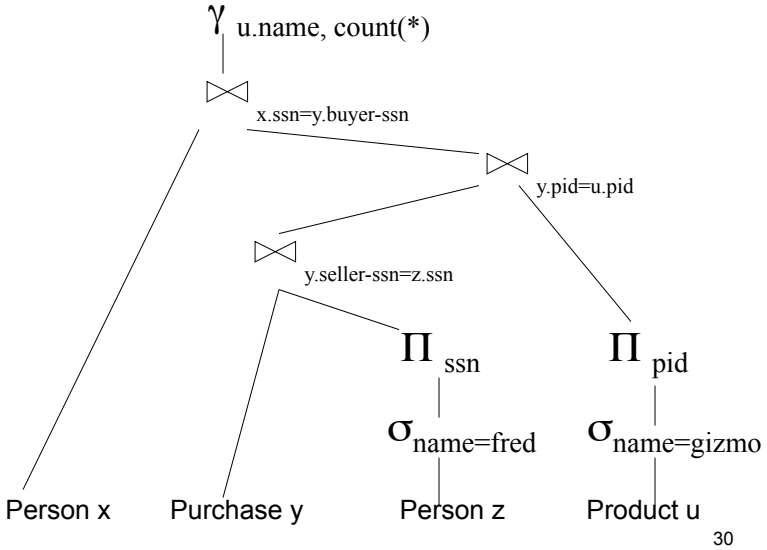


An Equivalent Expression

Query optimization = finds cheaper equivalent expressions



Complex RA Expressions – SQL?



30

Example Database Schema

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)
```

View: Suppliers in Manhattan, KS

```
CREATE VIEW NearbySupp AS
SELECT sno, sname
FROM Supplier
WHERE scity='Manhattan' AND sstate='KS'
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

Exercise – Translate to RA

Find the names of all suppliers in Manhattan who supply part number 2

```
SELECT sname FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

Rewritten Version of Our Query

Original query:

```
SELECT sname
FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```

View:

```
CREATE VIEW NearbySupp AS
SELECT sno, sname
FROM Supplier
WHERE scity='Manhattan' AND
sstate='KS'
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

Rewritten Version of Our Query

Original query:

```
SELECT sname
FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```

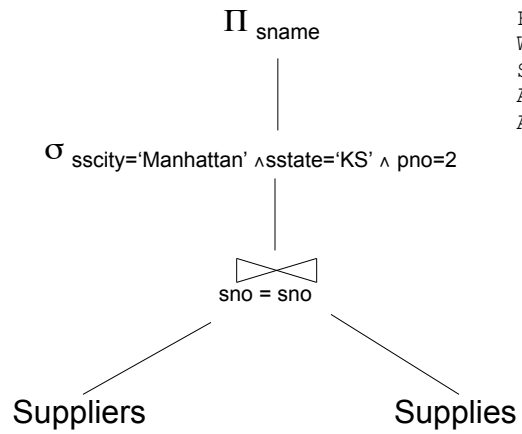
View:

```
CREATE VIEW NearbySupp AS
SELECT sno, sname
FROM Supplier
WHERE scity='Manhattan' AND
sstate='KS'
```

Rewritten query:

```
SELECT S.sname
FROM Supplier S, Supplies U
WHERE S.scity='Manhattan' AND S.sstate='KS'
AND S.sno = U.sno
AND U.pno = 2;
```

Logical Query Plan



Rewritten query:

```
SELECT S.sname
FROM Supplier S, Supplies U
WHERE S.scity='Manhattan' AND
S.sstate='KS'
AND S.sno = U.sno
AND U.pno = 2;
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

Exercise – Translate to RA

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and not exists
      SELECT *
      FROM Supplies P
      WHERE P.sno = Q.sno
            and P.price > 100
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

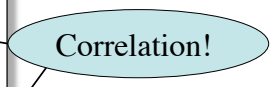
How about Subqueries?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and not exists
      SELECT *
      FROM Supplies P
      WHERE P.sno = Q.sno
            and P.price > 100
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

How about Subqueries?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and not exists
      SELECT *
      FROM Supplies P
      WHERE P.sno = Q.sno
            and P.price > 100
```



Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

How about Subqueries?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and not exists
      SELECT *
      FROM Supplies P
      WHERE P.sno = Q.sno
            and P.price > 100
```

De-Correlation

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and Q.sno not in
      SELECT P.sno
      FROM Supplies P
      WHERE P.price > 100
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

How about Subqueries?

Un-nesting

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'KS')
   EXCEPT
 (SELECT P.sno
  FROM Supplies P
  WHERE P.price > 100)
```

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'KS'
   and Q.sno not in
      SELECT P.sno
      FROM Supplies P
      WHERE P.price > 100
```

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)

How about Subqueries?

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'KS')
   EXCEPT
 (SELECT P.sno
  FROM Supplies P
  WHERE P.price > 100)
```

Finally…