Database Design and Implementation

CS 645

Relational algebra

Why relational algebra?

- operators that transform input data to output
- theoretical foundations for relational databases and languages
- mechanics used in processing and optimization

Relational Database = set of relations

Relation:

Schema: name of the relation, name and type of each column

Instance: a table, with rows and columns

Restriction:

All attributes are atomic, no nested tables

Arity (number of attributes) is 3

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

column, attribute, field

row, tuple

A relation is a set of tuples: no tuple can occur more than once
- Not true in practice, real systems allow duplicates

Relations

- A relation is a set of tuples
  - Sets: {a,b,c}, {a,d,e,f}, { }, . . .

- But, commercial DBMSs implement relations that are bags rather than sets
  - Bags: {a, a, b, c}, {b, b, b, b, b}, . . .

Sets vs Bags

Relational Algebra has two flavors:

- Over sets: theoretically elegant but limited
- Over bags: needed for SQL queries + more efficient
  - Example: Compute average price of all products

We discuss set semantics

- We mention bag semantics only where needed

Relational Query Languages

- Allow manipulation and retrieval of data from a database

- Query languages ≠ Programming languages
  - Not expected to be Turing complete
  - Not intended to be used for complex calculations
  - Support easy efficient access to large datasets

Preliminaries

$$Q: R_1 \cdots R_n \longrightarrow R'$$

- A query is applied to one or more relation instances
- The result is a relation instance
- The schema of the input is fixed
- The schema of the result is also fixed for a given query

Algebra

- A mathematical system consisting of:
  - Operands : variables or values from which new values are constructed
  - Operators : symbols denoting procedures that construct new values given existing values

Relational Algebra

- Query language associated with relational model
- Queries specified in an operational manner
  - A query gives a step-by-step procedure
- Operants are relations
- Relational operators
  - Take one or two relation instances as argument
  - Return one relation instance as result
  - Easy to compose into relational algebra expressions

The WHAT and the HOW

- In SQL, we write WHAT we want to get from 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**

SQL = WHAT

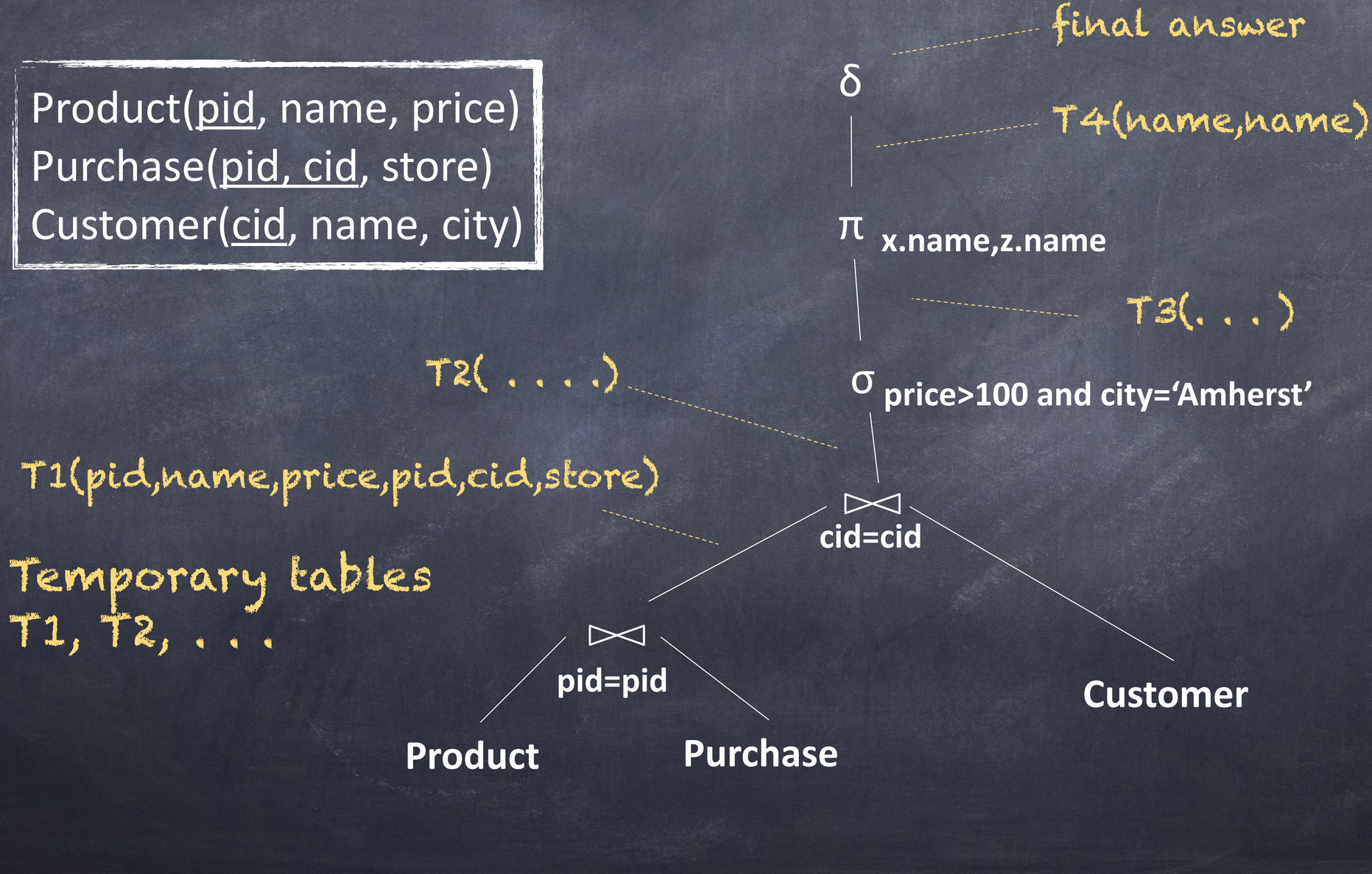
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

```
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 = 'Amherst'
```

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

Relational Algebra = HOW

Product(<u>pid</u>, name, price)
Purchase(<u>pid, cid</u>, store)
Customer(<u>cid</u>, name, city)



Temporary tables
T1, T2, . . .

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='Amherst'...
- ...eliminate duplicates...
- ...and that's the final answer !

Relational Algebra (1/3)

Five basic operators:

- Union ($\cup$) and Set difference ($-$)
- Selection: $\sigma_{\text{condition}}(S)$
  - Condition is Boolean combination ($\wedge, \vee$) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>=$, or $>$
- Projection: $\pi_{\text{list-of-attributes}}(S)$
- Cross-product or cartesian product ($\times$)

Relational Algebra (2/3)

Derived or auxiliary operators:

- Intersection ($\cap$), Division (R/S)
- Join: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B1,\cdots,Bn}(S)$

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)

Union and Difference

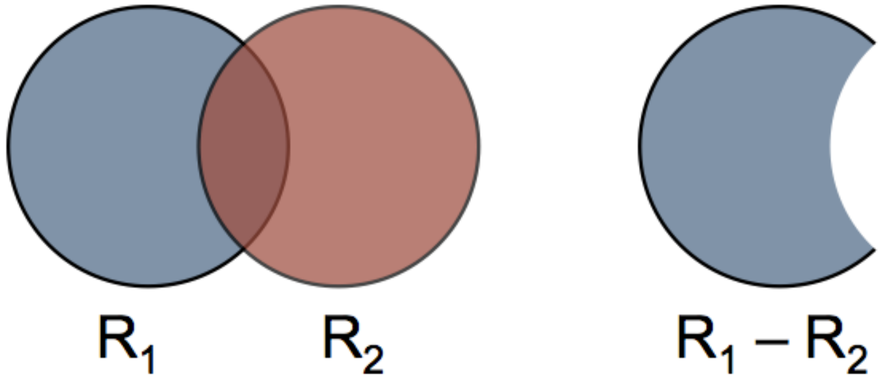
- R1 ∪ R2
- Example:
  - ActiveEmployees ∪ RetiredEmployees

- R1 – R2
- Example:
  - AllEmployees – RetiredEmployees

Be careful when applying to bags!

What about Intersection?

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



Relational Algebra (1/3)

Five basic operators:

- Union ($\cup$) and Set difference ($-$)
- Selection: $\sigma_{\text{condition}}(S)$
  - Condition is Boolean combination ($\wedge, \vee$) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>=$, or $>$
- Projection: $\pi_{\text{list-of-attributes}}(S)$
- Cross-product or cartesian product ($\times$)

Selection

- Returns all tuples that satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{“Smith”}}$ (Employee)
- The condition c can be
  - Boolean combination ($\wedge, \vee$) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>=$, or $>$

Maps to the WHERE clause in SQL!

Selection example

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

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

SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

Projection

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

Semantics differs over set or over bags

set semantics:
duplicate elimination automatic

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

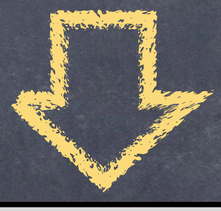
$\pi_{Name,Salary}(Employee)$

Name	Salary
John	20000
John	60000

bag semantics:
no duplicate elimination; need explicit δ

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

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



Name	Salary
John	20000
John	60000
John	20000

Selection & Projection Examples

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{zip,disease}$(Patient)

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{disease='heart'}$(Patient)

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

π_{zip} ($\sigma_{disease='heart'}$(Patient))

zip
98120
98125

Relational Algebra (1/3)

Five basic operators:

- Union ($\cup$) and Set difference ($-$)
- Selection: $\sigma_{\text{condition}}(S)$
  - Condition is Boolean combination ($\wedge, \vee$) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>=$, or $>$
- Projection: $\pi_{\text{list-of-attributes}}(S)$
- Cross-product or cartesian product ($\times$)

Cartesian Product

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Rare in practice; mainly used to express joins

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

Relational Algebra (2/3)

Derived or auxiliary operators:

- Intersection ($\cap$), Division (R/S)
- Join: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename: $\rho_{B1,\cdots,Bn}(S)$

Renaming

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

Relational Algebra (2/3)

Derived or auxiliary operators:

- Intersection ($\cap$), Division (R/S)
- Join: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B1,\cdots,Bn}(S)$

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

Theta-Join Example

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

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

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

Equijoin Example

$$R \bowtie_\theta S = \pi_A (\sigma_\theta(R \times S))$$

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

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

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

Natural Join Example

$$R \bowtie S = \pi_A (\sigma_\theta(R \times S))$$

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

AnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

P ⋈ J

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

So, which join is it ?

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

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

Outer Join Example

AnonPatient P

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

AnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

P ⟗ J

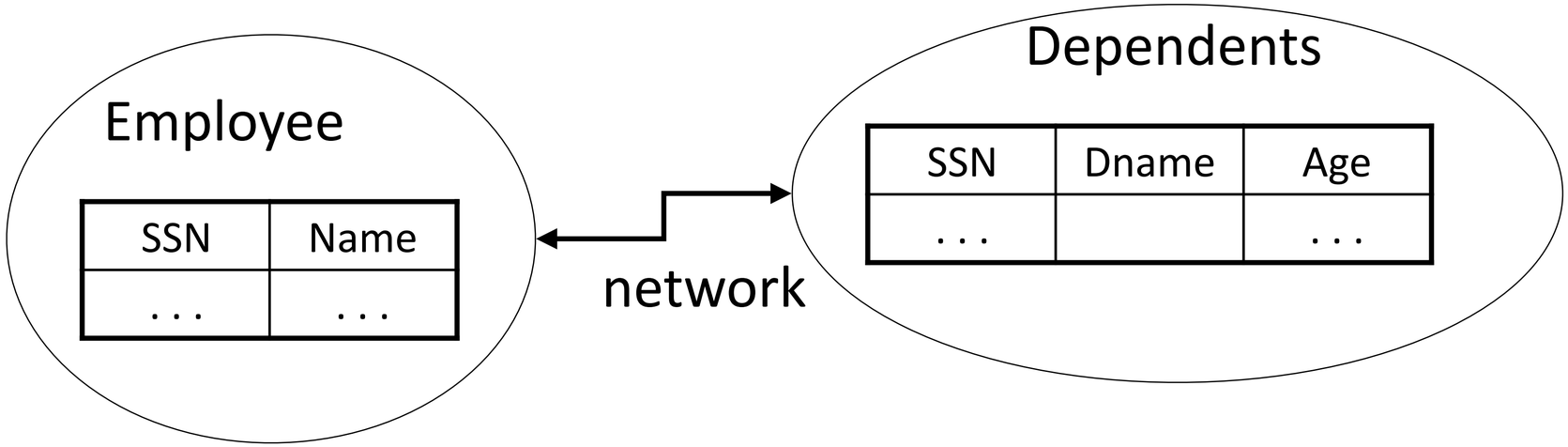
age	zip	disease	job
54	98125	heart	lawyer
20	98120	flu	cashier
33	98120	lung	null

Semijoin

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

Semijoins in Distributed Databases

- Semijoins are used in distributed databases



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

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

$$R = \text{Employee} \ltimes T$$

$$\text{Answer} = R \bowtie \text{Dependents}$$

Division

- A derived operator, used for queries like:
- “Find students who have enrolled in all systems courses”

- Take relations R(x,y) and S(y)

$$R/S = \{(x)|\forall(y) \in S, \exists(x,y) \in R\}$$

- Attributes of S must be a subset of attributes of R.

Division Examples

R

qno	pno
q1	p1
q1	p2
q1	p3
q1	p4
q2	p1
q2	p2
q3	p2
q4	p2
q4	p4

S1

pno
p2

R/S1

qno
q1
q2
q3
q4

S2

pno
p2
p4

R/S2

qno
q1
q4

S3

pno
p1
p2
p4

R/S3

qno
q1

Division with Basic Operators

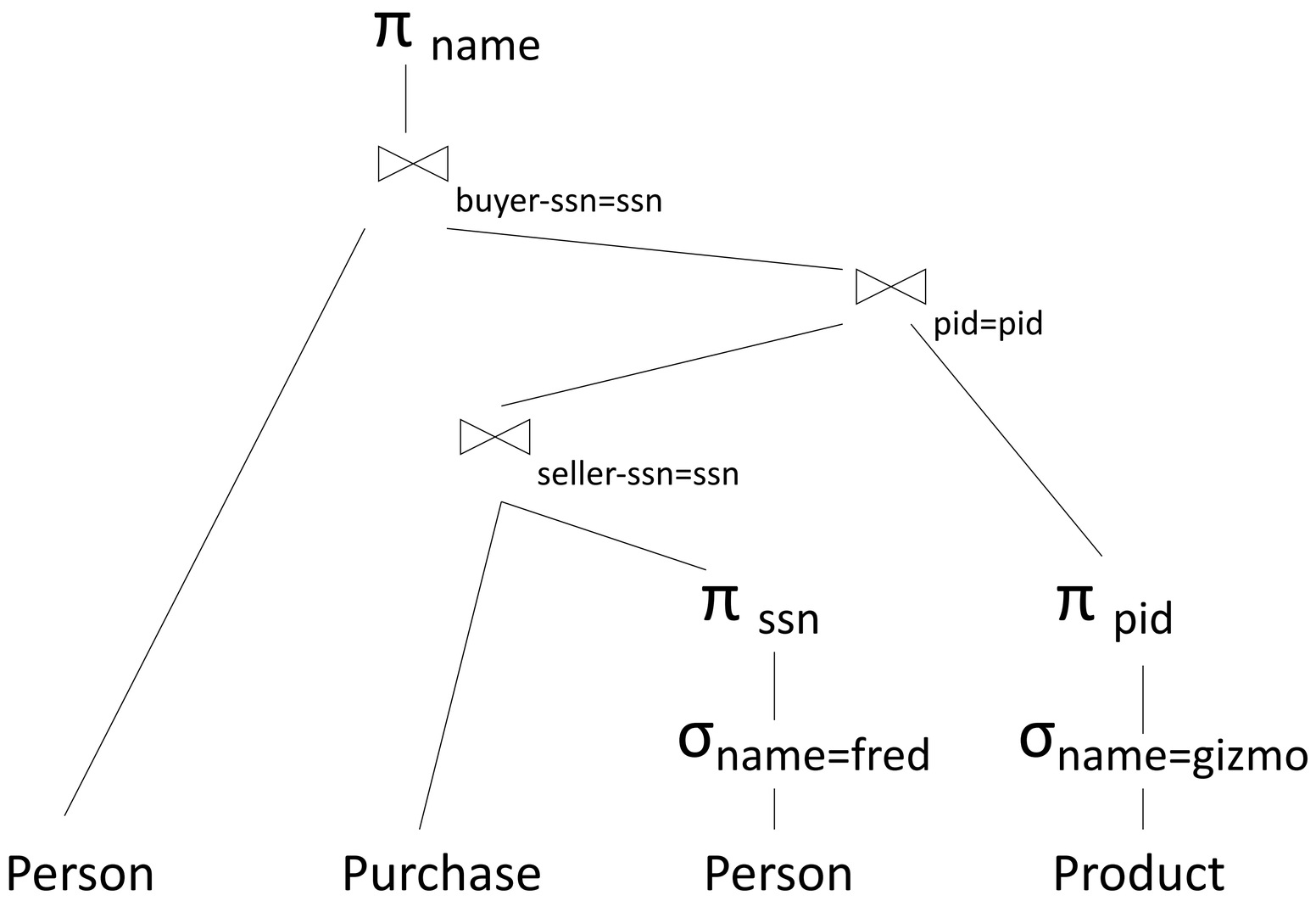
- Compute "missing tuples"

$$\pi_x((\pi_x(R) \times S) - R)$$

- Division result:

$$\pi_x(R) - \pi_x((\pi_x(R) \times S) - R)$$

Complex RA Expressions



Algebraic Equivalence

- Relational algebra has laws of associativity, commutativity, etc., that imply that certain expressions are equivalent

Query Equivalence

Two queries are equivalent if the produce the same result for all data instances

$$Q(D) = Q'(D), \forall D$$

Equivalence ⇒ Query Optimization

- Equivalent expressions may differ in cost of evaluation!

$$\sigma_{c \wedge d}(R) \equiv \sigma_c(\sigma_d(R))$$ cascading selection

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$ join associativity

$$\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$$ pushing selections

- Query optimization tries to find the most efficient representation to evaluate

Bag Semantics

- bag = set with repeated elements

- Operations need to be defined carefully:
  - {a,b,b,c} ∪ {a,b,e,f,f} = {a,a,b,b,b,c,e,f,f}
  - {a,b,b,b,c,c} − {b,c,c,c,d} = {a,b,b}
  - σ_c: preserves the number of occurrences
  - $\pi_A(R)$: does not eliminate duplicates
  - Cross-product, join: keep duplicates

Bag Laws $\neq$ Set Laws

- Commutativity of union holds

$$R \cup S = S \cup R$$

- Idempotence

$$S \cup S \neq S$$

RA and Transitive Closure

- 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!!!