Relational Algebra

Motivation

- Employees(eid, ename, city, state)
  Departments(did, dname, mid)

- Select E.ename
  From Employees E, Departments D
  Where E.eid = D.mid and E.city = ‘Madison’

- How to execute this query?
  – create possible plans
  – estimate their runtimes
  – select and execute the fastest plan

Five Basic RA Operations

+ union
+ set difference
+ selection
+ projection
+ Cartesian product

Set Operations: Union

- Union: all tuples in R1 or R2
- Notation: R1 U R2
- R1, R2 must have the same schema
- R1 U R2 has the same schema as R1, R2
- Example:
  - ActiveEmployees U RetiredEmployees

Set Operations: Difference

- Difference: all tuples in R1 and not in R2
- Notation: R1 – R2
- R1, R2 must have the same schema
- R1 - R2 has the same schema as R1, R2
- Example
  - AllEmployees - RetiredEmployees

Selection

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- *c* is a condition: =, <, >, and, or, not
- Output schema: same as input schema
- Find all employees with salary more than $40,000:
  - $\sigma_{Salary > 40000}$ (Employee)

Selection Example

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

Find all employees with salary more than $40,000.

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

SSN	Name	DepartmentID	Salary
888888888	Alice	2	45,000

Example

Relation Sells:

bar	beer	price
Joe’s	Bud	2.50
Joe’s	Miller	2.75
Sue’s	Bud	2.50
Sue’s	Miller	3.00

JoeMenu := SELECT$_{bar=\text{“Joe’s”}}$(Sells):

bar	beer	price
Joe’s	Bud	2.50
Joe’s	Miller	2.75

Projection

- Returns certain columns
- Eliminates duplicate tuples!
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Input schema $R(B1,\ldots,Bm)$
- Condition: $\{A1,\ \ldots,\ An\} \subseteq \{B1,\ \ldots,\ Bm\}$
- Output schema $S(A1,\ldots,An)$
- Example: project social-security number and names:
  - $\Pi_{SSN,\ Name}(Employee)$

Projection Example

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

$\Pi_{\text{SSN, Name}}$ **(Employee)**

SSN	Name
999999999	John
777777777	Tony
888888888	Alice

Example

Relation Sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Prices := $\text{PROJ}_{\text{beer,price}}$(Sells):

beer	price
Bud	2.50
Miller	2.75
Miller	3.00

Cartesian Product

- Each tuple in *R1* with each tuple in *R2*
- Notation: *R1 x R2*
- Input schemas *R1(A1,…,An), R2(B1,…,Bm)*
- Condition: $\{A1,\ldots,An\} \cap \{B1,\ldots Bm\} = \Phi$
- Output schema is *S(A1, …, An, B1, …, Bm)*
- Notation: *R1 x R2*
- Example: **Employee x Dependents**
- Very rare in practice; but joins are very common

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

Example: R3 := R1 * R2

R1(

A,	B
1	2
3	4

)

R2(

B,	C
5	6
7	8
9	10

)

R3(

A,	R1.B,	R2.B,	C
1	2	5	6
1	2	7	8
1	2	9	10
3	4	5	6
3	4	7	8
3	4	9	10

)

Renaming

- Does not change the relational instance
- Changes the relational schema only
- Notation: $\rho_{B1,\ldots,Bn}\,(R)$
- Input schema: $R(A1, \ldots, An)$
- Output schema: $S(B1, \ldots, Bn)$
- Example:

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

Renaming Example

Employee

Name	SSN
John	999999999
Tony	777777777

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

LastName	SocSocNo
John	999999999
Tony	777777777

Derived RA Operations

1) Intersection

2) Most importantly: Join

Set Operations: Intersection

- All tuples both in R1 and in R2
- Notation: $R1 \cap R2$
- R1, R2 must have the same schema
- $R1 \cap R2$ has the same schema as R1, R2
- Example
  - UnionizedEmployees $\cap$ RetiredEmployees
- Intersection is derived:
  - $R1 \cap R2 = R1 - (R1 - R2)$ why ?

Joins

- Theta join
- Natural join
- Equi-join
- Semi-join
- Inner join
- Outer join
- etc.

Theta Join

- A join that involves a predicate
- Notation: $R1 \bowtie_{\theta} R2$ where θ is a condition
- Input schemas: $R1(A1,\ldots,An)$, $R2(B1,\ldots,Bm)$
- $\{A1,\ldots An\} \cap \{B1,\ldots,Bm\} = \phi$
- Output schema: $S(A1,\ldots,An,B1,\ldots,Bm)$
- Derived operator:

$$R1 \bowtie_{\theta} R2 = \sigma_{\theta}(R1 \; x \; R2)$$

Example

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

name,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells JOIN $_{\text{Sells.bar = Bars.name}}$ Bars

BarInfo(

bar,	beer,	price,	name,	addr
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

)

Natural Join

- Notation: $R1 \bowtie R2$
- Input Schema: $R1(A1, \ldots, An)$, $R2(B1, \ldots, Bm)$
- Output Schema: $S(C1,\ldots,Cp)$
  - Where $\{C1, \ldots, Cp\} = \{A1, \ldots, An\} \cup \{B1, \ldots, Bm\}$
- Meaning: combine all pairs of tuples in R1 and R2 that agree on the attributes:
  - $\{A1,\ldots,An\} \cap \{B1,\ldots, Bm\}$ (called the join attributes)
- Equivalent to a cross product followed by selection
- Example **Employee** $\bowtie$ **Dependents**

Natural Join Example

Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

SSN	Dname
999999999	Emily
777777777	Joe

Employee $\bowtie$ Dependents =

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

Name	SSN	Dname
John	999999999	Emily
Tony	777777777	Joe

Natural Join

- R=

A	B
X	Y
X	Z
Y	Z
Z	V

S=

B	C
Z	U
V	W
Z	V

- $R \bowtie S$ =

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

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 $R \bowtie S$?

Example

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells JOIN Bars

Note Bars.name has become Bars.bar to make the natural join "work."

BarInfo(

bar,	beer,	price,	addr
Joe's	Bud	2.50	Maple St.
Joe's	Milller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

)

Equi-join

- Most frequently used in practice:

$$R1 \bowtie_{A=B} R2$$

- Natural join is a particular case of equi-join
- A lot of research on how to do it efficiently

Relational Algebra

- Five basic operators, many derived
- Combine operators in order to construct queries: relational algebra expressions, usually shown as trees

Building Complex Expressions

- Algebras allow us to express sequences of operations in a natural way.
- Example
  - in arithmetic algebra: $(x + 4)*(y - 3)$
- Relational algebra allows the same.
- Three notations, just as in arithmetic:
  1. Sequences of assignment statements.
  2. Expressions with several operators.
  3. Expression trees.

Sequences of Assignments

- Create temporary relation names.
- Renaming can be implied by giving relations a list of attributes.
- Example: R3 := R1 JOIN$_C$ R2 can be written:

  R4 := R1 * R2

  R3 := SELECT$_C$ (R4)

Expressions with Several Operators

- Example: the theta-join R3 := R1 JOIN$_C$ R2 can be written: R3 := SELECT$_C$ (R1 * R2)
- Precedence of relational operators:
  1. Unary operators --- select, project, rename --- have highest precedence, bind first.
  2. Then come products and joins.
  3. Then intersection.
  4. Finally, union and set difference bind last.
- But you can always insert parentheses to force the order you desire.

Expression Trees

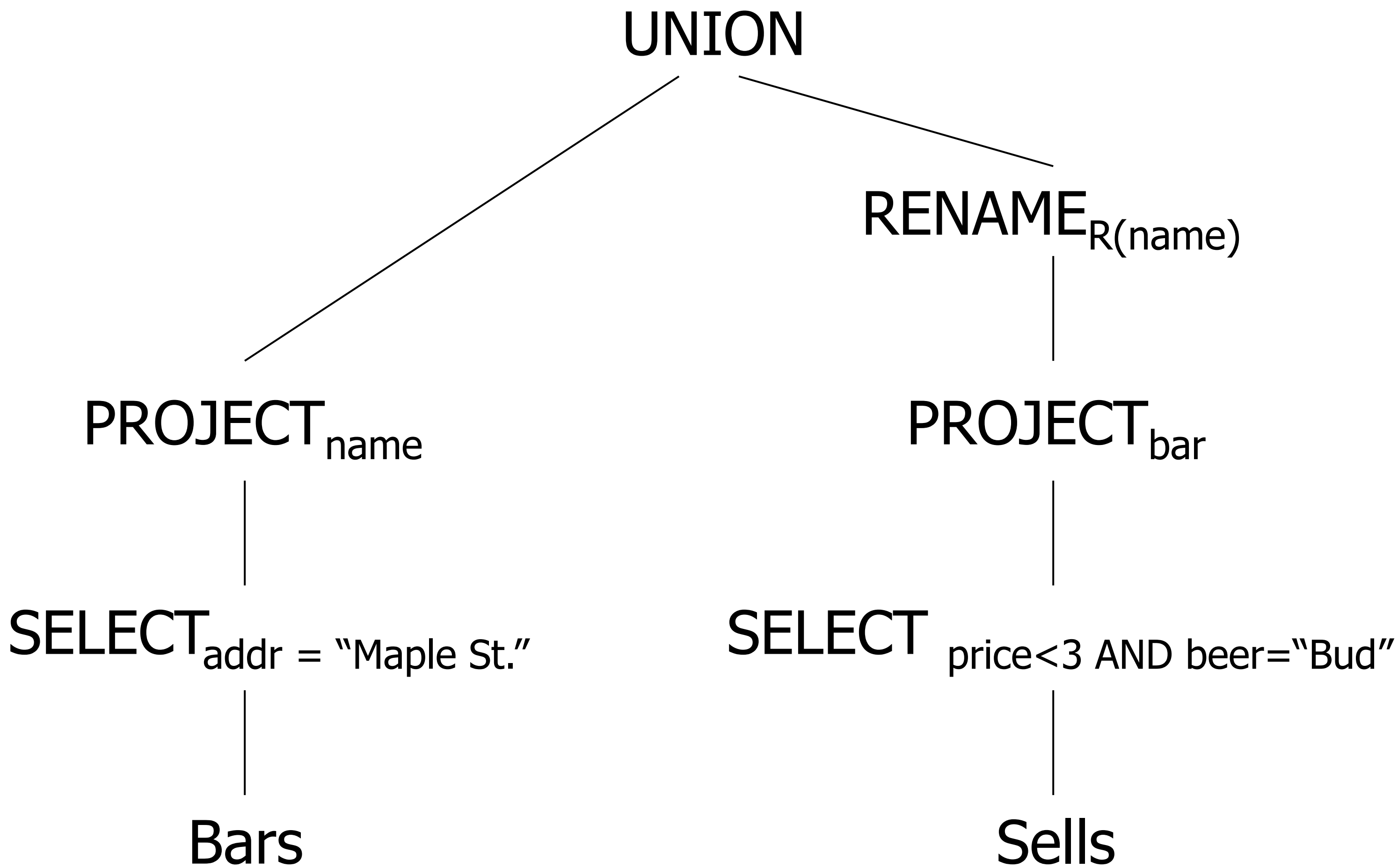
- Leaves are operands --- either variables standing for relations or particular, constant relations.
- Interior nodes are operators, applied to their child or children.

Example

- Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.

As a Tree:

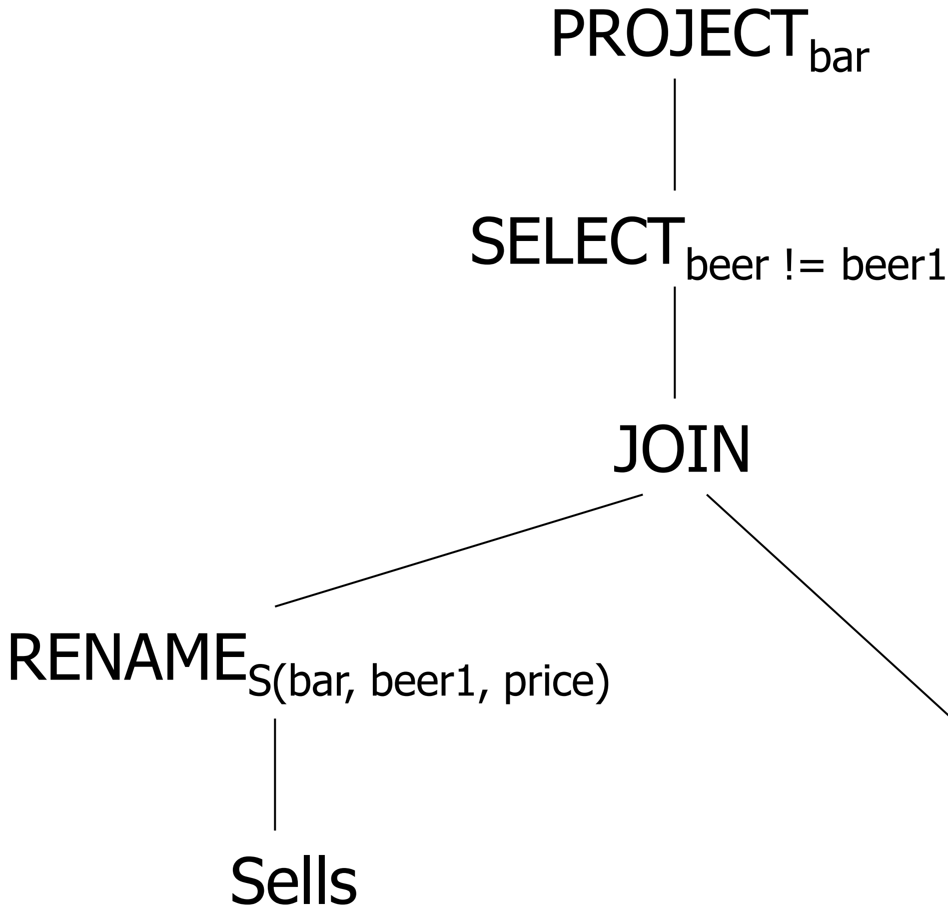
- Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.



Example

- Using Sells(bar, beer, price), find the bars that sell two different beers at the same price.
- Strategy: by renaming, define a copy of Sells, called S(bar, beer1, price). The natural join of Sells and S consists of quadruples (bar, beer, beer1, price) such that the bar sells both beers at this price.

The Tree



PROJECT$_{bar}$
SELECT$_{beer\ !=\ beer1}$
JOIN
RENAME$_{S(bar,\ beer1,\ price)}$
Sells
Sells

Complex Queries

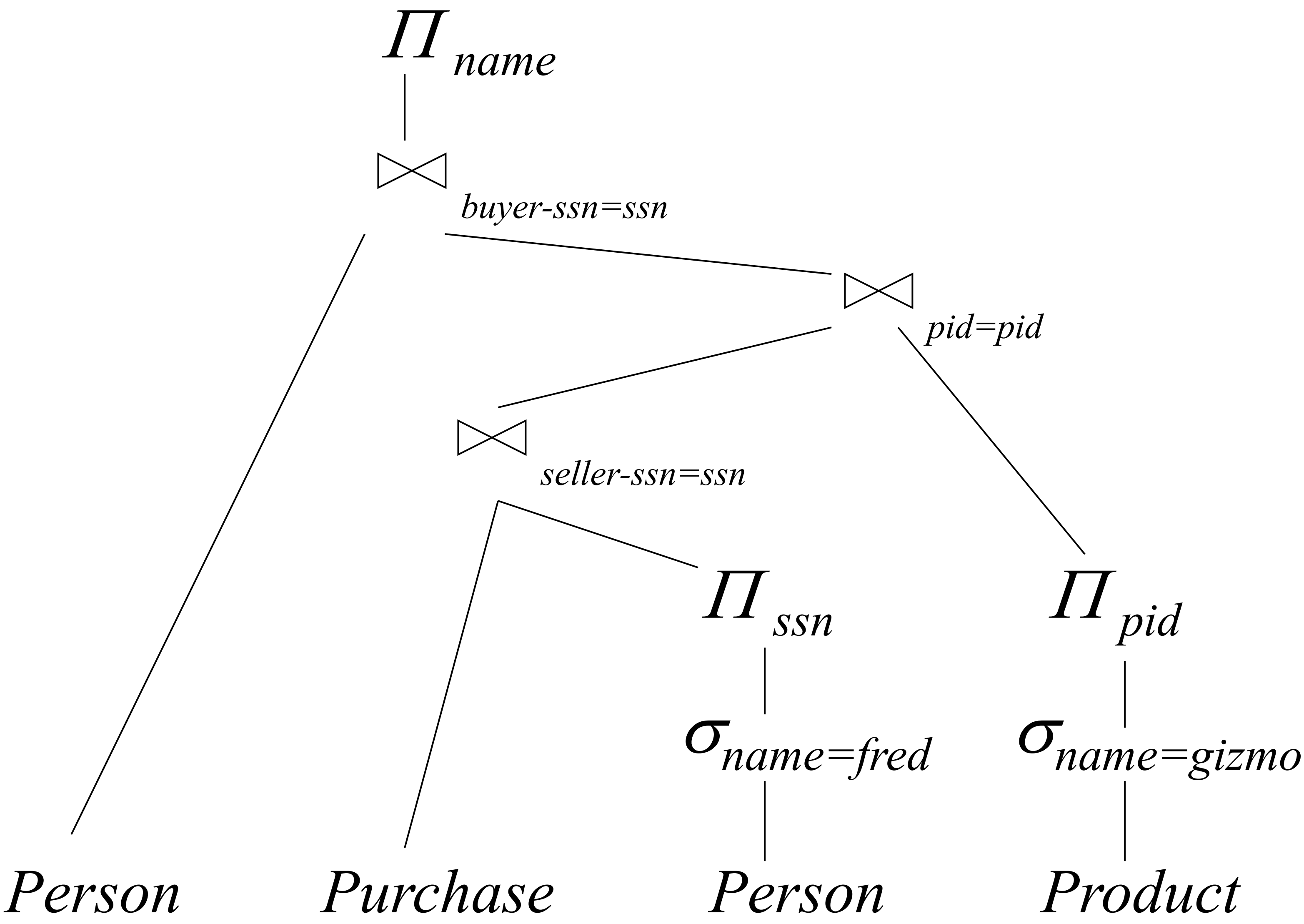
Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Note:
•in Purchase: buyer-ssn, seller-ssn are **foreign keys** in Person, pid is **foreign key** in Product;
•in Product maker-cid is a **foreign key** in Company

Find phone numbers of people who bought gizmos from Fred.

Find telephony products that somebody bought

Expression Tree



Exercises

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Ex #1: Find people who bought telephony products.

Ex #2: Find names of people who bought American products

Exercises

Product (<u>pid</u>, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (<u>cid</u>, name, stock price, country)
Person(<u>ssn</u>, name, phone number, city)

Ex #3: Find names of people who bought American products and did not buy French products

Ex #4: Find names of people who bought American products and they live in Champaign.

Exercises

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Ex #5: Find people who bought stuff from Joe or bought products from a company whose stock prices is more than $50.

Operations on Bags (and why we care)

- Union: {a,b,b,c} U {a,b,b,b,e,f,f} = {a,a,b,b,b,b,b,c,e,f,f}
  - *add* the number of occurrences
- Difference: {a,b,b,b,c,c} – {b,c,c,c,d} = {a,b,b,d}
  - subtract the number of occurrences
- Intersection: {a,b,b,b,c,c} {b,b,c,c,c,c,d} = {b,b,c,c}
  - minimum of the two numbers of occurrences
- Selection: preserve the number of occurrences
- Projection: preserve the number of occurrences (no duplicate elimination)
- Cartesian product, join: no duplicate elimination

Read the book for more detail

Summary of Relational Algebra

- Why bother ? Can write any RA expression directly in C++/Java, seems easy.
- Two reasons:
  - Each operator admits sophisticated implementations (think of $\bowtie$, σ_C)
  - Expressions in relational algebra can be rewritten: optimized

Glimpse Ahead: Efficient Implementations of Operators

- $\sigma_{(\textbf{age} >= 30 \text{ AND } \textbf{age} <= 35)}$**(Employees)**
  - Method 1: scan the file, test each employee
  - Method 2: use an index on **age**
  - Which one is better ? Depends a lot…
- **Employees** $\bowtie$ **Relatives**
  - Iterate over Employees, then over Relatives
  - Iterate over Relatives, then over Employees
  - Sort Employees, Relatives, do “merge-join”
  - “hash-join”
  - etc

Glimpse Ahead: Optimizations

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Person(ssn, name, phone number, city)

- Which is better:

$$\sigma_{price>100}(\text{Product}) \bowtie (\text{Purchase} \bowtie \sigma_{city=sea}\text{Person})$$

$$(\sigma_{price>100}(\text{Product}) \bowtie \text{Purchase}) \bowtie \sigma_{city=sea}\text{Person}$$

- Depends ! This is the optimizer’s job…

Finally: 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