



Database Management Systems - I, CS 157A

Relational Algebra



Agenda

- Relational Algebra Basic Operations
- Algebra of Bags



Relational Algebra

What is an “Algebra”

- Mathematical system consisting of:
 - *Operands* --- variables or values from which new values can be constructed.
 - *Operators* --- symbols denoting procedures that construct new values from given values.

What is Relational Algebra?

- An algebra whose operands are relations or variables that represent relations.
- Operators are designed to do the most common things that we need to do with relations in a database.
 - The result is an algebra that can be used as a *query language* for relations.

Core Relational Algebra

- Union, intersection, and difference:
 - Usual set operations, but *both operands must have the same relation schema*.
- Selection: picking certain rows.
- Projection: picking certain columns.
- Products and joins: compositions of relations.
- Renaming of relations and attributes.

Selection

■ $R1 := \sigma_C(R2)$

- C is a condition (as in “if” statements) that refers to attributes of $R2$.
- $R1$ is all those tuples of $R2$ that satisfy C .

Example: Selection

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 := $\sigma_{\text{bar}=\text{"Joe's"}}(\text{Sells})$:

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

Projection

■ $R1 := \pi_L(R2)$

- L is a list of attributes from the schema of $R2$.
- $R1$ is constructed by looking at each tuple of $R2$, extracting the attributes on list L , in the order specified, and creating from those components a tuple for $R1$.
- Eliminate duplicate tuples, if any.

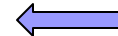
Example: Projection

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 := $\pi_{\text{beer}, \text{price}}(\text{Sells})$:

beer	price
Bud	2.50
Miller	2.75
Miller	3.00



Extended Projection

- Using the same Π_L operator, we allow the list L to contain arbitrary expressions involving attributes:
 1. Arithmetic on attributes, e.g., $A+B \rightarrow C$.
 2. Duplicate occurrences of the same attribute.

Example: Extended Projection

$R =$ (

A	B
1	2
3	4

)

$\pi_{A+B \rightarrow C, A, A}(R) =$

C	A1	A2
3	1	1
7	3	3

Product (Cartesian Product)

■ $R3 := R1 \times R2$

- Pair each tuple $t1$ of $R1$ with each tuple $t2$ of $R2$.
- Concatenation $t1t2$ is a tuple of $R3$.
- Schema of $R3$ is the attributes of $R1$ and then $R2$, in order.
- But be aware an attribute A of the same name in $R1$ and $R2$: use $R1.A$ and $R2.A$

Example: $R3 := R1 \times R2$

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

Theta-Join

- $R3 := R1 \bowtie_C R2$
 - Take the product $R1 \times R2$.
 - Then apply σ_C to the result.
- As for σ , C can be any boolean-valued condition:
 - Historic versions of this operator allowed only $A \theta B$, where θ is $=$, $<$, etc.; hence the name “theta-join.”

Example: Theta Join

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 $\bowtie_{\text{Sells.bar} = \text{Bars.name}}$ 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

- A useful join variant (*natural* join) connects two relations by:
 - Equating attributes of the same name, and
 - Projecting out one copy of each pair of equated attributes.
- Denoted $R3 := R1 \bowtie R2$.

Example: Natural Join

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 ⋈ 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	Miller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

)

Renaming

- The ρ operator gives a new schema to a relation.
- $R1 := \rho_{R1(A1, \dots, An)}(R2)$ makes R1 be a relation with attributes $A1, \dots, An$ and the same tuples as R2.
- Simplified notation: $R1(A1, \dots, An) := R2$.

Example: Renaming

Bars(

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

)

$R(\text{bar}, \text{addr}) := \text{Bars}$

R(

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

)

Building Complex Expressions

- Combine operators with parentheses and precedence rules.
- 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:** $R3 := R1 \bowtie_C R2$ can be written:
 $R4 := R1 \times R2$ ← cartesian product
 $R3 := \sigma_C(R4)$ ← selection / filtering

Expressions in a Single Assignment

- **Example:** the theta-join $R3 := R1 \bowtie_C R2$
can be written: $R3 := \sigma_C (R1 \times R2)$
- **Precedence of relational operators:**
 1. $[\sigma, \pi, \rho]$ (highest).
 2. $[\times, \bowtie]$.
 3. \cap .
 4. $[\cup, -]$

Expression Trees

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

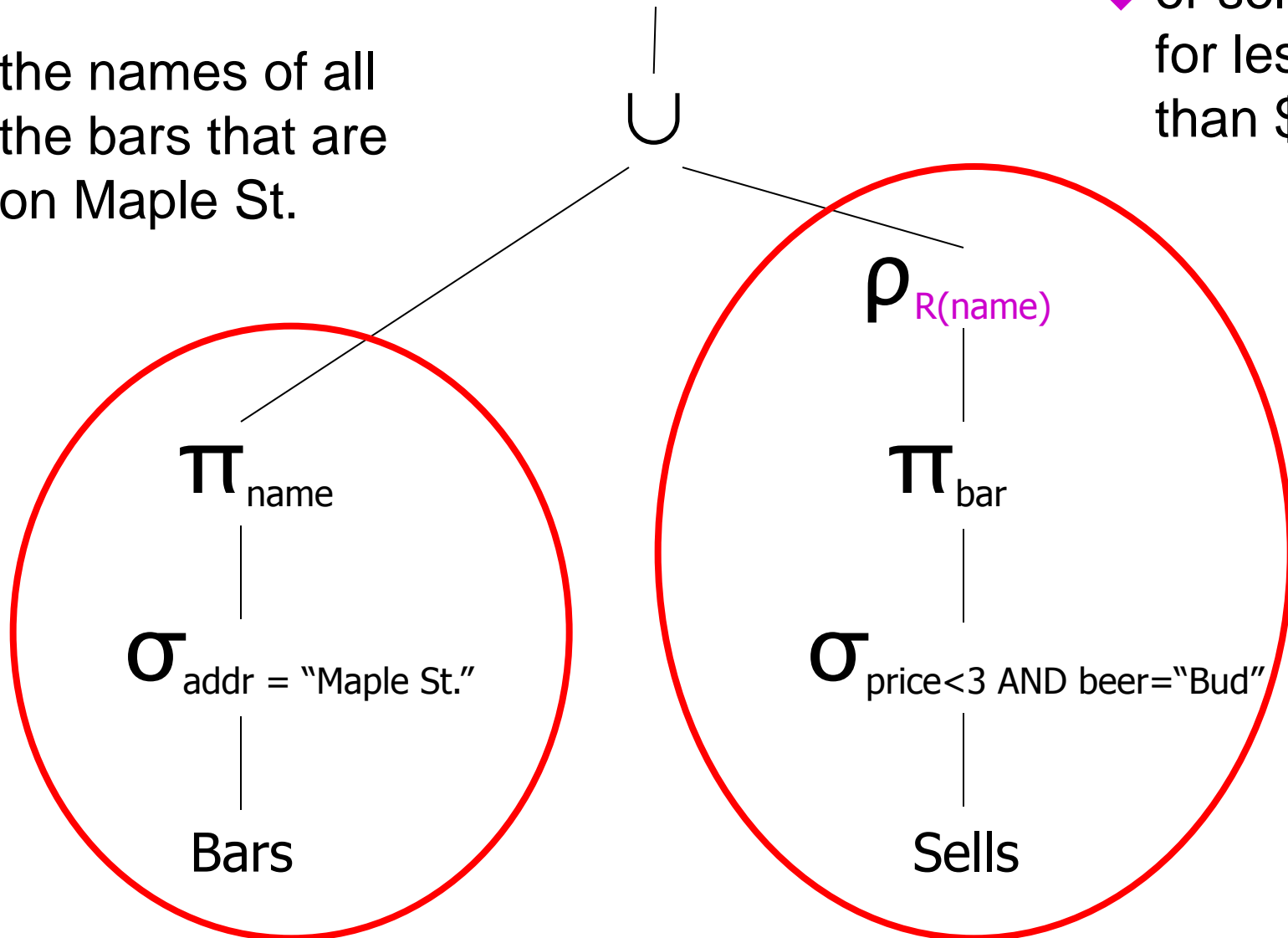
Example: Tree for a Query

- 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:

◆ the names of all the bars that are on Maple St.

◆ or sell Bud for less than \$3.

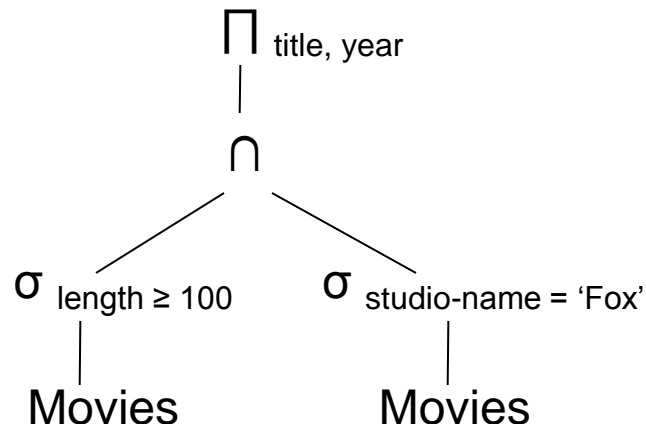


SQL: An Algebraic Query Language

■ Combining operations to form queries:

- Relational algebra allows us to form expressions of arbitrary complexity by applying operations on the result of other operations
- What are titles and years of movies made by Fox that are at least 100 minutes long
- `select title, year from Movies where length >= 100 and studio-name = 'Fox';`

Expression tree:



Linear notations:

$\Pi_{\text{title, year}} (\sigma_{\text{length} \geq 100}(\text{MOVIES}) \cap \sigma_{\text{studio-name} = \text{'Fox'}}(\text{MOVIES}))$

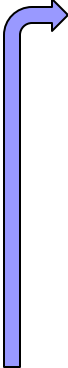
SQL: An Algebraic Query Language

■ Naming & renaming ($\rho_{S(A_1, A_2, \dots, A_n)}(R)$):

- Rename relation (R) to be relation (S) and rename its attributes to be A_1, A_2, \dots, A_n in that order

■ Relationship among operations:

- $R \cap S = R - (R - S)$



- $R \bowtie_c S = \sigma_c(R \times S)$

- $R \bowtie S = \Pi_L(\sigma_c(R \times S)),$

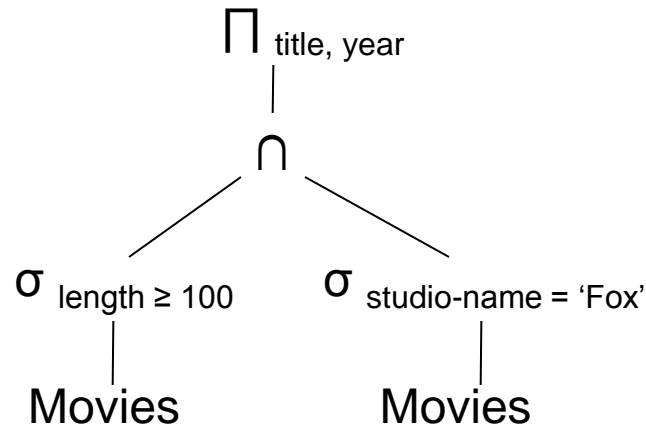
where “L” is list of attributes in “R” followed by those of “S” that are not also in “R”

- $U \bowtie_{A < D \text{ AND } U.B \neq V.B} V = \sigma_{A < D \text{ AND } U.B \neq V.B}(U \times V)$

SQL: An Algebraic Query Language

■ Linear notation for Algebraic Expressions:

- **SELECT** title, year **FROM** Movies
 WHERE length \geq 100 **AND** studio-name = 'Fox';
- Assume attributes of Movies relation are: (t,y,l,i,s,p)
- Expression tree:



- Then we can represent the tree graph as follows:
 - $R(t,y,l,i,s,p) \quad := \quad \sigma_{l \geq 100} (Movies)$
 - $S(t,y,l,i,s,p) \quad := \quad \sigma_{s='Fox'} (Movies)$
 - $T(t,y,l,i,s,p) \quad := \quad R \cap S$
 - **Answer(title,year)** $:= \Pi_{t,y} (T)$

Constraints on Relations

■ Relational Algebra as a Constraint Language

- Two ways to use expressions of relational algebra to express constraints:
 - If R is an expression, then $R = 0$ is a constraint – no tuples in the result R
 - If R and S are expressions of relational algebra, then $R \subseteq S$ is a constraint that says every tuple in R must also be in S . Of course S may contain additional tuples not in R

■ Referential Integrity Constraints

- Referential Integrity constraint asserts that a value appearing in one context also appears in another related context
- $\Pi_A(R) - \Pi_B(S) = 0$ or equivalently $\Pi_A(R) \subseteq \Pi_B(S)$

Constraints on Relations (Contd.)

■ Key Constraints

- To express algebraically that an attribute or set of attributes is a key for a relation R
- Let us use hypothetical two names for the above relation: R and S and the attribute key is **name**, and another random attribute is **address**
- $\sigma_{R.name=S.name \text{ AND } R.address \neq S.address} (R \times S) = 0$

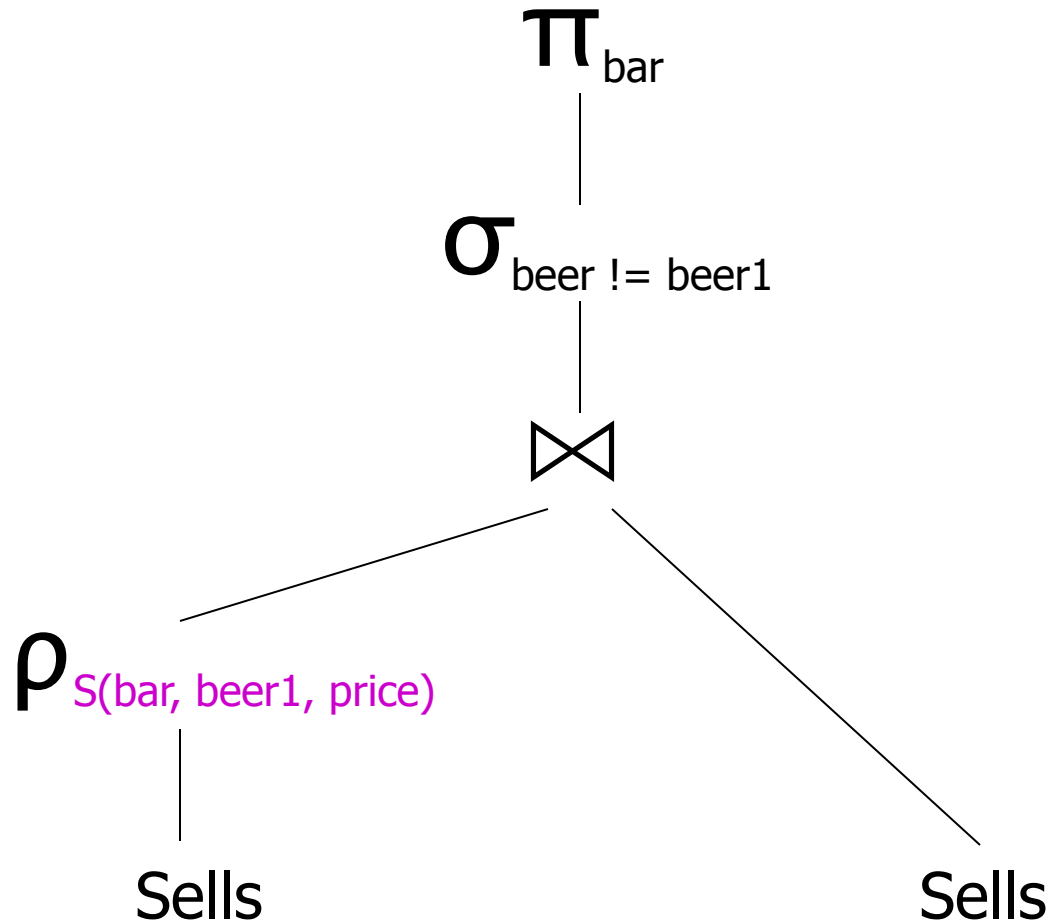
■ Additional Constraints

- Assume gender of movie star has to be 'M' or 'F' only
- $\sigma_{gender \neq 'M' \text{ AND } gender \neq 'F'} (\text{MovieStar}) = 0$

Example: Self-Join

- 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)
 - ➔ *The bar sells both beers (beer and beer1) at this price.*

The Tree



Schemas for Results

- **Union, intersection, and difference:** the schemas of the two operands must be the same, so use that schema for the result.
- **Selection:** schema of the result is the same as the schema of the operand.
- **Projection:** list of attributes tells us the schema.

Schemas for Results (cont.)

- **Product:** schema is the attributes of both relations:
 - Use $R.A$, etc., to distinguish two attributes named A .
- **Theta-join:** same as product.
- **Natural join:** union of the attributes of the two relations (no duplicate attributes).
- **Renaming:** the operator tells the schema.



Algebra of Bags

Relational Algebra on Bags

- A *bag* (or *multiset*) is like a set, but an element may appear more than once.
- **Example:** $\{1,2,1,3\}$ is a bag and is not a set.
- **Example:** $\{1,2,3\}$ is also a bag that happens to be a set.

Why Bags?

- SQL, the most important query language for relational databases, is actually a bag language.
- Some operations, like projection, are more efficient to produce bags than sets.

Example: Bag Selection

R(

A,	B
1	2
5	6
1	2

)

$\sigma_{A+B < 5} (R) =$

A	B
1	2
1	2

Example: Bag Projection

$R($

A,	B
1	2
5	6
1	2

)

$\pi_A(R) =$

A
1
5
1

Example: Bag Product

R(

A,	B
1	2
5	6
1	2

)

S(

B,	C
3	4
7	8

)

R X S =

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	3	4
5	6	7	8
1	2	3	4
1	2	7	8

Example: Bag Theta-Join

R(

A,	B
1	2
5	6
1	2

)

S(

B,	C
3	4
7	8

)

$R \bowtie_{R.B < S.B} S =$

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	7	8
1	2	3	4
1	2	7	8

Bag Union

- An element appears in the result of the union of two bags is the sum of the number of times it appears in each bag.
- **Example:** $\{1,2,1\} \cup \{1,1,2,3,1\} = \{1,1,1,1,1,2,2,3\}$

Bag Intersection

- An element appears in the intersection of two bags is the minimum of the number of times it appears in either.
- **Example:** $\{1,2,1,1\} \cap \{1,2,1,3\} = \{1,1,2\}$.

Bag Difference

- An element appears in the difference $A - B$ of bags as many times as it appears in A , minus the number of times it appears in B .
 - But never less than 0 times.
- **Example:** $\{1, 2, 1, 1\} - \{1, 2, 3\} = \{1, 1\}$.



Summary

- Relational Algebra
- Selection / Projection / Join
- Expression Tree
- Relational Algebra on Bags



END