

CS 541

Relational Algebra

September 9, 2002

Chris Clifton - CS541

1

“Core” Relational Algebra

A small set of operators that allow us to manipulate relations in limited but useful ways. The operators are:

1. Union, intersection, and difference: the usual set operators.
 - ◆ But the relation schemas must be the same.
2. *Selection*: Picking certain rows from a relation.
3. *Projection*: Picking certain columns.
4. *Products and joins*: Composing relations in useful ways.
5. *Renaming* of relations and their attributes.

Fall 2002

Chris Clifton - CS541

2

Relational Algebra

- limited expressive power (subset of possible queries)
- good optimizer possible
- rich enough language to express enough useful things

Finiteness

σ SELECT	} UNARY	} FUNDAMENTAL
π PROJECT		
\times CARTESIAN PRODUCT	} BINARY	
\cup UNION		
$-$ SET-DIFFERENCE		
\cap SET-INTERSECTION	} CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS	
\bowtie_{θ} THETA-JOIN		
\bowtie NATURAL JOIN		
\div DIVISION or QUOTIENT		

Fall 2002 Chris Clifton - CS541 3

Selection

$R_1 = \sigma_C(R_2)$
 where C is a condition involving the attributes of relation R_2 .

Example

Relation *Sells*:

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

$JoeMenu = \sigma_{bar=Joe's}(Sells)$

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

Fall 2002 Chris Clifton - CS541 4

Extra Example Relations

DEPOSIT(branchName, acctNo, custName, balance)

CUSTOMER(custName, street, custCity)

BORROW(branchName, loan-no, custName, amount)

BRANCH(branchName, assets, branchCity)

CLIENT(custName, empName)

Borrow	BN	L#	CN	AMT
T1	Midtown	123	Fred	600
T2	Midtown	234	Sally	1200
T3	Midtown	235	Sally	1500
T4	Downtown	612	Tom	2000

Fall 2002

Chris Clifton - CS541

5

SELECT (σ)

$\text{arity}(\sigma(R)) = \text{arity}(R)$

$0 \leq \text{card}(\sigma(R)) \leq \text{card}(R)$

$\sigma_c(R)$

$\sigma_c(R) \subseteq (R)$

c is selection condition: terms of form: attr op value attr op attr

op is one of $< = > \leq \geq$

example of term: branch-name = 'Midtown'

terms are connected by $\wedge \vee \neg$

$\sigma \text{ branchName} = \text{'Midtown'} \wedge \text{amount} > 1000$ (Borrow)

$\sigma \text{ custName} = \text{empName}$ (client)

Fall 2002

Chris Clifton - CS541

6

Projection

$$R_1 = \pi_L(R_2)$$

where L is a list of attributes from the schema of R_2 .

Example

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

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

beer	price
Bud	2.50
Miller	2.75
Coors	3.00

- Notice elimination of duplicate tuples.

Projection (π)

$$0 \leq \text{card}(\pi_A(R)) \leq \text{card}(R)$$

$$\text{arity}(\pi_A(R)) = m \leq \text{arity}(R) = k$$

$$\pi_{i_1, \dots, i_m}(R) \quad 1 \leq i_j \leq k \text{ distinct}$$

produces set of m -tuples $\langle a_1, \dots, a_m \rangle$

such that \exists k -tuple $\langle b_1, \dots, b_k \rangle$ in R where $a_j = b_{i_j}$ for $j = 1, \dots, m$

$$\pi_{\text{branchName, custName}}(\text{Borrow})$$

Midtown Fred

Midtown Sally

Downtown Tom

Product

$$R = R_1 \times R_2$$

pairs each tuple t_1 of R_1 with each tuple t_2 of R_2 and puts in R a tuple $t_1 t_2$.

Cartesian Product (\times)

$$\text{arity}(R) = k_1 \quad \text{arity}(R \times S) = k_1 + k_2$$

$$\text{arity}(S) = k_2 \quad \text{card}(R \times S) = \text{card}(R) \times \text{card}(S)$$

$R \times S$ is the set all possible $(k_1 + k_2)$ -tuples

whose first k_1 attributes are a tuple in R

last k_2 attributes are a tuple in S

R			
A	B	C	D

S		
D	E	F

R \times S						
A	B	C	D	D'	E	F

Theta-Join

$$R = R_1 \bowtie_C R_2$$

is equivalent to $R = \sigma_C(R_1 \times R_2)$.

Fall 2002

Chris Clifton - CS541

11

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.

$$\text{BarInfo} = \text{Sells} \bowtie_{\text{Sells.Bar=Bars.Name}} \text{Bars}$$

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.

Fall 2002

Chris Clifton - CS541

12

Theta-Join $R \bowtie_{\theta} S$

$i \theta j$

$\sigma_{\theta}(R \times S)$

R

1 ... r

i

S

1 ... s

j

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

$R(A B C) \bowtie_{R.A < S.D} S(C D E)$

result has schema $T(A B C C' D E)$

arity(R) = r
 arity(S) = s
 arity($R \bowtie_{\theta} S$) = r + s

$0 \leq \text{card}(R \bowtie_{\theta} S) \leq \text{card}(R) \times \text{card}(S)$

θ can be $< > = \neq \leq \geq$

If equal (=), then it is an EQUIJOIN

R(ABC)	S(CDE)	T(ABCC'DE)
1 3 5	2 1 1	1 3 5 1 2 2
2 4 6	1 2 2	1 3 5 3 3 4
3 5 7	3 3 4	1 3 5 4 4 3
4 6 8	4 4 3	2 4 6 3 3 4
		2 4 6 4 4 3
		3 5 7 4 4 3

Fall 2002 Chris Clifton - CS541 13

Natural Join

$R = R_1 \bowtie R_2$

calls for the theta-join of R_1 and R_2 with the condition that all attributes of the same name be equated. Then, one column for each pair of equated attributes is projected out.

Example

Suppose the attribute name in relation Bars was changed to bar, to match the bar name in Sells.

$\text{BarInfo} = \text{Sells} \bowtie \text{Bars}$

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.

Fall 2002 Chris Clifton - CS541 14

Difference($R - S$)

$$\text{arity}(R) = \text{arity}(S) = \text{arity}(R - S)$$

$$0 \leq \text{card}(R - S) \leq \text{card}(R) \quad \emptyset \subseteq R - S \subseteq R$$

is the tuples in R not in S

Depositors of Perryridge who aren't borrowers of Perryridge

$\pi_{\text{custName}} (\sigma_{\text{branchName} = \text{'Perryridge'}} (\text{DEPOSIT} - \text{BORROW}))$

Deposit < Perryridge, 36, Pat, 500 >

Borrow < Perryridge, 72, Pat, 10000 >

$\pi_{\text{custName}} (\sigma_{\text{branchName} = \text{'Perryridge'}} (\text{DEPOSIT})) -$

$\pi_{\text{custName}} (\sigma_{\text{branchName} = \text{'Perryridge'}} (\text{BORROW}))$

Does $\sigma (\pi (D) - \pi (B))$ work?

Combining Operations

Algebra =

1. Basis arguments +
2. Ways of constructing expressions.

For relational algebra:

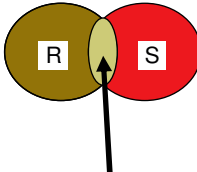
1. Arguments = variables standing for relations + finite, constant relations.
2. Expressions constructed by applying one of the operators + parentheses.
 - Query = expression of relational algebra.

$\pi_{\text{custName, custCity}}$
 $(\sigma_{\text{Client.Banker-Name} = \text{'Johnson'}}$
 $(\text{Client} \times \text{Customer})) =$
 $\pi_{\text{cust-Name, custCity}}(\text{Customer})$
 • Is this always true? Is this what we wanted?
 $\pi_{\text{Client.custName, Customer.custCity}}$
 $(\sigma_{\text{Client.bankerName} = \text{'Johnson'}}$
 $\wedge \text{Client.custName} = \text{Customer.custName}$
 $(\text{Client} \times \text{Customer}))$
 $\pi_{\text{Client.custName, Customer.custCity}}$
 $(\sigma_{\text{Client.custName} = \text{Customer.custName}}$
 $(\text{Customer} \times \pi_{\text{custName}}$
 $(\sigma_{\text{Client.bankerName} = \text{'Johnson'}}(\text{Client})))$

Fall 2002 Chris Clifton - CS541 19

SET INTERSECTION $\text{arity}(R) = \text{arity}(S) = \text{arity}(R \cap S)$
 $(R \cap S)$ $0 \leq \text{card}(R \cap S) \leq \min(\text{card}(R), \text{card}(S))$

tuples both in R and in S



$\emptyset \subseteq R \cap S \subseteq R$
 $\emptyset \subseteq R \cap S \subseteq S$

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

Fall 2002 Chris Clifton - CS541 20

Operator Precedence

The normal way to group operators is:

1. Unary operators σ , π , and ρ have highest precedence.
 2. Next highest are the “multiplicative” operators, \bowtie , \bowtie_C , and \times .
 3. Lowest are the “additive” operators, \cup , \cap , and $-$.
- But there is no universal agreement, so we always put parentheses *around* the argument of a unary operator, and it is a good idea to group all binary operators with parentheses *enclosing* their arguments.

Example

Group $R \cup \sigma S \bowtie T$ as $R \cup (\sigma(S) \bowtie T)$.

Fall 2002

Chris Clifton - CS541

21



CS 541 Relational Algebra

September 11, 2002

Chris Clifton - CS541

22

Each Expression Needs a Schema

- If \cup , \cap , $-$ applied, schemas are the same, so use this schema.
- Projection: use the attributes listed in the projection.
- Selection: no change in schema.
- Product $R \times S$: use attributes of R and S .
 - ◆ But if they share an attribute A , prefix it with the relation name, as $R.A$, $S.A$.
- Theta-join: same as product.
- Natural join: use attributes from each relation; common attributes are merged anyway.
- Renaming: whatever it says.

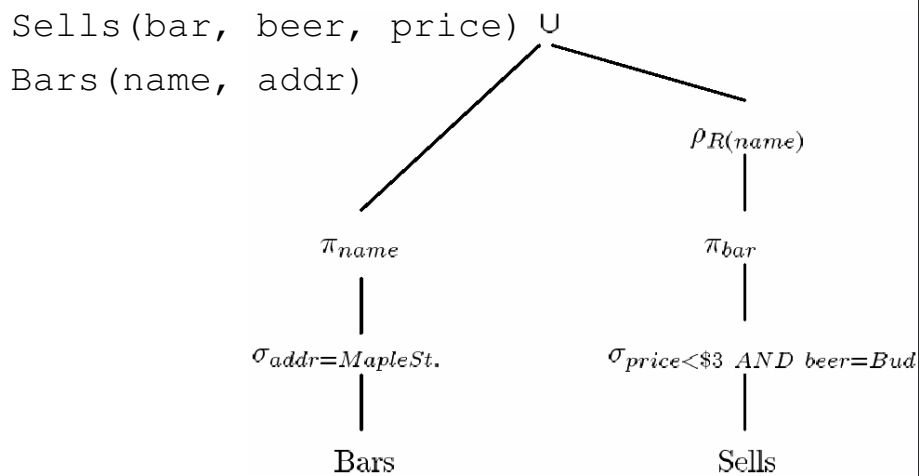
Fall 2002

Chris Clifton - CS541

23

Example

- Find the bars that are either on Maple Street or sell Bud for less than \$3.



Fall 2002

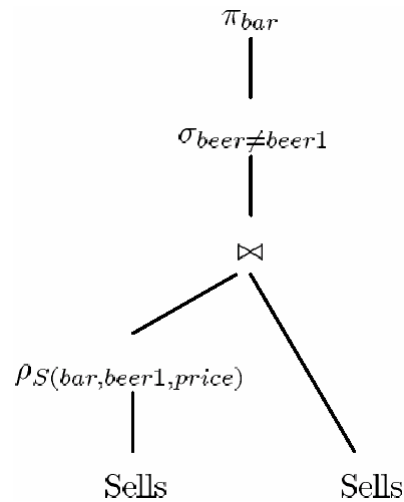
Chris Clifton - CS541

24

Example

Find the bars that sell
two different beers at
the same price.

Sells(bar, beer,
price)



Fall 2002

Chris Clifton - CS541

25

Linear Notation for Expressions

- Invent new names for intermediate relations, and assign them values that are algebraic expressions.
- Renaming of attributes implicit in schema of new relation.

Example

Find the bars that are either on Maple Street or sell Bud for less than \$3.

Sells(bar, beer, price)

Bars(name, addr)

R1(name) := $\pi_{\text{name}}(\sigma_{\text{addr} = \text{Maple St.}}(\text{Bars}))$

R2(name) := $\pi_{\text{bar}}(\sigma_{\text{beer} = \text{Bud AND price} < \$3}(\text{Sells}))$

R3(name) := $R1 \cup R2$

Fall 2002

Chris Clifton - CS541

26

Why Decomposition “Works”?

What does it mean to “work”? Why can’t we just tear sets of attributes apart as we like?

- Answer: the decomposed relations need to represent the same information as the original.
 - ◆ We must be able to reconstruct the original from the decomposed relations.

Projection and Join Connect the Original and Decomposed Relations

- Suppose R is decomposed into S and T . We project R onto S and onto T .

Fall 2002

Chris Clifton - CS541

27

Example

$$R =$$

name	addr	beersLiked	manf	favoriteBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

- FDs:
 - ◆ name \rightarrow addr
 - ◆ name \rightarrow favoriteBeer
 - ◆ beersLiked \rightarrow manf
- Decompose:

Fall 2002

Chris Clifton - CS541

28

Project onto Drinkers1 (name, addr, favoriteBeer):

<u>name</u>	<u>addr</u>	<u>favoriteBeer</u>
Janeway	Voyager	WickedAle
Spock	Enterprise	Bud

Project onto Drinkers3 (beersLiked, manf):

<u>beersLiked</u>	<u>manf</u>
Bud	A.B.
WickedAle	Pete's
Bud	A.B.

Project onto Drinkers4 (name, beersLiked):

<u>name</u>	<u>beersLiked</u>
Janeway	Bud
Janeway	WickedAle
Spock	Bud

Reconstruction of Original

Can we figure out the original relation from the decomposed relations?

- Sometimes, if we natural join the relations.

Example

Drinkers3 \bowtie Drinkers4 =

<u>name</u>	<u>beersLiked</u>	<u>manf</u>
Janeway	Bud	A.B.
Janeway	WickedAle	Pete's
Spock	Bud	A.B.

- Join of above with Drinkers1 = original *R*.

Theorem

Suppose we decompose a relation with schema XYZ into XY and XZ and project the relation for XYZ onto XY and XZ . Then $XY \bowtie XZ$ is *guaranteed* to reconstruct XYZ if and only if $X \twoheadrightarrow Y$ (or equivalently, $X \twoheadrightarrow Z$).

- Usually, the MVD is really a FD, $X \rightarrow Y$ or $X \rightarrow Z$.
- BCNF: When we decompose XYZ into XY and XZ , it is because there is a FD $X \rightarrow Y$ or $X \rightarrow Z$ that violates BCNF.
 - ◆ Thus, we can always reconstruct XYZ from its projections onto XY and XZ .
- 4NF: when we decompose XYZ into XY and XZ , it is because there is an MVD $X \twoheadrightarrow Y$ or $X \twoheadrightarrow Z$ that violates 4NF.
 - ◆ Again, we can reconstruct XYZ from its projections onto XY and XZ .

Fall 2002

Chris Clifton - CS541

31

Bag Semantics

A relation (in SQL, at least) is really a *bag* or *multiset*.

- It may contain the same tuple more than once, although there is no specified order (unlike a list).
- Example: $\{1,2,1,3\}$ is a bag and not a set.
- Select, project, and join work for bags as well as sets.
 - ◆ Just work on a tuple-by-tuple basis, and don't eliminate duplicates.

Fall 2002

Chris Clifton - CS541

32

Bag Union

Sum the times an element appears in the two bags.

- Example: $\{1,2,1\} \cup \{1,2,3,3\} = \{1,1,1,2,2,3,3\}$.

Bag Intersection

Take the minimum of the number of occurrences in each bag.

- Example: $\{1,2,1\} \cap \{1,2,3,3\} = \{1,2\}$.

Bag Difference

Proper-subtract the number of occurrences in the two bags.

- Example: $\{1,2,1\} - \{1,2,3,3\} = \{1\}$.



Laws for Bags Differ From Laws for Sets

- Some familiar laws continue to hold for bags.
 - ◆ Examples: union and intersection are still commutative and associative.
- But other laws that hold for sets do *not* hold for bags.

Example

$R \cap (S \cup T) \equiv (R \cap S) \cup (R \cap T)$ holds for sets.

- Let R , S , and T each be the bag $\{1\}$.
- Left side: $S \cup T = \{1,1\}$; $R \cap (S \cup T) = \{1\}$.
- Right side: $R \cap S = R \cap T = \{1\}$;
 $(R \cap S) \cup (R \cap T) = \{1\} \cup \{1\} = \{1,1\} \neq \{1\}$.



CS 541
Extended Relational Algebra

September 13, 2002

Chris Clifton - CS541

35

**Extended (“Nonclassical”)
Relational Algebra**

Adds features needed for SQL, bags.

1. Duplicate-elimination operator δ .
2. Extended projection.
3. Sorting operator τ .
4. Grouping-and-aggregation operator γ .
5. Outerjoin operator \bowtie .

Fall 2002

Chris Clifton - CS541

36

Duplicate Elimination

$\delta(R)$ = relation with one copy of each tuple that appears one or more times in R .

Example

$R =$

A	B
1	2
3	4
1	2

$\delta(R) =$

A	B
1	2
3	4

Sorting

- $\tau_L(R)$ = list of tuples of R , ordered according to attributes on list L .
- Note that result type is outside the normal types (set or bag) for relational algebra.
 - ◆ Consequence: τ cannot be followed by other relational operators.

Example

$R =$

A	B
1	3
3	4
5	2

$\tau_B(R) = [(5,2), (1,3), (3,4)]$.

Extended Projection

Allow the columns in the projection to be functions of one or more columns in the argument relation.

Example

$R =$	<u>A</u>	<u>B</u>	
	1	2	
	3	4	
$\pi_{A+B,A,A}(R) =$			
	<u>A+B</u>	<u>A1</u>	<u>A2</u>
	3	1	1
	7	3	3

Aggregation Operators

- These are not relational operators; rather they summarize a column in some way.
- Five standard operators: Sum, Average, Count, Min, and Max.
- Use with grouping (see next slide) or shorthand as “special” projection:

$R:$	A	B
	1	2
	3	4
$\pi_{Max(A), Min(B)}(R) =$		
	<u>Max(A)</u>	<u>Min(B)</u>
	3	2

- Remember: Aggregations return a single row – can’t combine with non-aggregates in projection

Grouping Operator

$\gamma_L(R)$, where L is a list of elements that are either

- a) Individual (*grouping*) attributes or
- b) Of the form $\theta(A)$, where θ is an aggregation operator and A the attribute to which it is applied,

is computed by:

1. Group R according to all the grouping attributes on list L .
2. Within each group, compute $\theta(A)$, for each element $\theta(A)$ on list L .
3. Result is the relation whose columns consist of one tuple for each group. The components of that tuple are the values associated with each element of L for that group.

Example

Let $R =$

bar	beer	price
Joe's	Bud	2.00
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00
Mel's	Miller	3.25

Compute $\gamma_{\text{beer,AVG(price)}}(R)$.

1. Group by the grouping attribute(s), beer in this case:

bar	beer	price
Joe's	Bud	2.00
Sue's	Bud	2.50
Joe's	Miller	2.75
Mel's	Miller	3.25
Sue's	Coors	3.00

2. Compute average of `price` within groups:

beer	AVG(price)
Bud	2.25
Miller	3.00
Coors	3.00

Outerjoin

The normal join can “lose” information, because a tuple that doesn’t join with any from the other relation (*dangles*) has no vestage in the join result.

- The *null value* \perp can be used to “pad” dangling tuples so they appear in the join.
- Gives us the *outerjoin* operator \bowtie° .
- Variations: theta-outerjoin, left- and right-outerjoin (pad only dangling tuples from the left (respectively, right)).

Example

$R =$	A	B	
	1	2	
	3	4	

$S =$	B	C	
	4	5	
	6	7	

$R \bowtie^o S =$	A	B	C	
	3	4	5	<i>part of natural join</i>
	1	2	⊥	<i>part of right-outerjoin</i>
	⊥	6	7	<i>part of left-outerjoin</i>

Division Operator

- Let $R=XY, S=Y$. Then $R \div S$ produces a relation X where
 - ◆ $x \in R$
 - ◆ $\forall y \in S, xy \in R$
- Example: Bars that serve everyone's favorite beers
 - ◆ $\Pi_{\text{bars,beers}}(\text{Sells}) \div \Pi_{\text{favoriteBeer}}(\text{Drinkers})$
- Division isn't a fundamental operator
 - ◆ $R \div S = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$