CS54100: Database Systems

Relational Algebra

3 February 2012

Prof. Walid Aref

Indiana Center for Database Systems ™

"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.

Relational Algebra

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

Finiteness

σ SELECT
π PROJECT
} UNARY

X CARTESIAN PRODUCT
U UNION
– SET-DIFFERENCE
} BINARY

} FUNDAMENTAL

$\cap$ SET-INTERSECTION
$\bowtie_{\theta}$ THETA-JOIN
$\bowtie$ NATURAL JOIN
÷ DIVISION or QUOTIENT
} CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS

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

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

Extra Example Relations

DEPOSIT(<u>branchName, acctNo,</u>custName,balance)
CUSTOMER(<u>custName,</u>street,custCity)
BORROW(<u>branchName,loan-no,</u>custName,amount)
BRANCH(<u>branchName,</u>assets, branchCity)
CLIENT(<u>custName,</u>emplName)

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

Selection

SELECT (σ)

arity(σ(R)) = arity(R)

$0 \leq$ card(σ(R)) $\leq$ card(R)

σ_c (R)

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

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

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

example of term: branch-name = ‘Midtown’

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

$\sigma_{\text{branchName = ‘Midtown’} \wedge \text{amount > 1000}}$ (Borrow)

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

Projection

$R_1 = \pi_L(R_2)$

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

Example

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

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

beer	price
Bud	2.50
Miller	2.75
Coors	3.00

- Notice elimination of duplicate tuples.

Projection

Projection **(π)**

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

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

$\pi_{i_1,\ldots,i_m}(R) \quad 1 \le i_j \le k$ distinct

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

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

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

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 $t_1 t_2$.

Product

Cartesian Product (×)

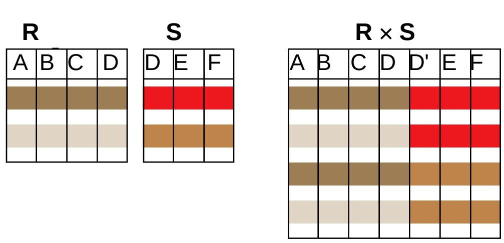
arity(R) = k1 arity(R × S) = k1 + k2

arity(S) = k2 card(R × S) = card(R) × card(S)

R × S is the set all possible (k1 + k2)-tuples

whose first k1 attributes are a tuple in R

last k2 attributes are a tuple in S



Theta-Join

$R = R_1 \bowtie_C R_2$

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

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` $\bowtie_{\texttt{Sells.Bar=Bars.Name}}$ `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.

Theta-Join $R \underset{i\,\theta\,j}{\bowtie} S$

$\sigma_{\$i\ \theta\ \$(r+j)}\ (R \times S)$

R: 1 . . . r, **i**

S: 1 . . . s, **j**

arity(R) = r

arity(S) = s

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

$0 \leq card(R \underset{\theta}{\bowtie} S) \leq card(R) \times card(S)$

θ can be $< > = \neq \leq \geq$

If equal (=), then it is an EQUIJOIN

$R \underset{c}{\bowtie} S = \sigma_c (R \times S)$

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

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

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

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`.

`BarInfo = Sells` $\bowtie$ `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.

Renaming

$\rho_{S(A_1,\ldots,A_n)}(R)$ produces a relation identical to R but named S and with attributes, in order, named $A_1,\ldots,A_n$.

Example

Bars =

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

$\rho_{R(\texttt{bar},\texttt{addr})}(\texttt{Bars})$ =

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

- The name of the second relation is R.

Set Operations: Union

Union ($R \cup S$) **arity(R) = arity(S) = arity($R \cup S$)**

max(card(R),card(S)) $\leq$ card($R \cup S$) $\leq$ card(R) + card(S)

set of tuples in R or S or both **$R \subseteq R \cup S$**

$S \subseteq R \cup S$

Find customers of Perryridge Branch

$\pi_{\text{Cust-Name}}\ (\sigma_{\text{Branch-Name = "Perryridge"}}\ (\text{BORROW} \cup \text{DEPOSIT})\ )$

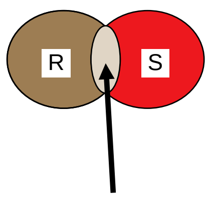
Set Operations: Intersection

SET INTERSECTION **arity(R) = arity(S) = arity ($R \cap S$)**

($R \cap S$) **$0 \le$ card ($R \cap S$) $\le$ min (card(R), card(S))**

$\varnothing \subseteq R \cap S \subseteq R$

tuples both in R and in S $\varnothing \subseteq R \cap S \subseteq S$



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

Set Operations: Difference

Difference($R - S$)

arity(R) = arity(S) = arity($R - S$)

$0 \le$ card($R - S$) $\le$ card(R) $\varnothing \subseteq R - S \subseteq R$

is the tuples in R not in S

Depositors of Perryridge who aren't borrowers of Perryridge

$\pi_{custName}$ (σ $_{branchName = \text{'Perryridge'}}$ **(DEPOSIT – BORROW))**

Deposit < Perryridge, 36, Pat, 500 >

Borrow < Perryridge, 72, Pat, 10000 >

$\pi_{custName}$ (σ $_{branchName = \text{'Perryridge'}}$ (DEPOSIT)) —
$\pi_{custName}$ (σ $_{branchName = \text{'Perryridge'}}$ (BORROW))

Does σ (π (D) – π (B)) work?

PURDUE
UNIVERSITY

CS54100: Database Systems

Relational Algebra

6 February 2012

Prof. Chris Clifton

Indiana
Center for
Database
Systems
™

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.

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

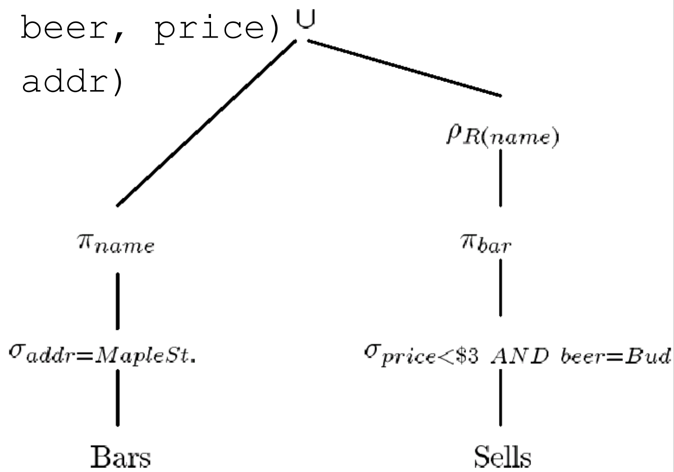
Each Expression Needs a Schema

- 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.

Example

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

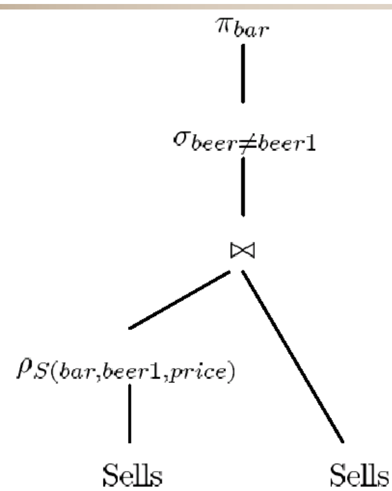
```
Sells(bar, beer, price)
Bars(name, addr)
```



Example

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

```
Sells(bar, beer,
  price)
```



CS54100

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

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

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

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*.

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 → addr
  - name → favoriteBeer
  - beersLiked → manf
- Decompose:

Project onto `Drinkers1(name, addr, favoriteBeer)`:

name	addr	favoriteBeer
Janeway	Voyager	WickedAle
Spock	Enterprise	Bud

Project onto `Drinkers3(beersLiked, manf)`:

beersLiked	manf
Bud	A.B.
WickedAle	Pete's
Bud	A.B.

Project onto `Drinkers4(name, beersLiked)`:

name	beersLiked
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` =

name	beersLiked	manf
Janeway	Bud	A.B.
Janeway	WickedAle	Pete's
Spock	Bud	A.B.

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

Theorem: Lossless Join

- Decomposition of *XYZ* into *XY* and *XZ*:
  - Let $XY = \Pi\ XYZ$; $XZ = \Pi\ XYZ$
  - $XY \bowtie XZ$ guaranteed to reconstruct *XYZ* if and only if $X \rightarrow\rightarrow Y$
    - Remember that $X \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow Y$
  - 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 $X \rightarrow\rightarrow Y$ or $X \rightarrow\rightarrow Z$ that violates 4NF.
  - Again, we can reconstruct *XYZ* from its projections onto *XY* and *XZ*.

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.

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: $R \cap S = R \cap T = \{1\}$;
  $(R \cap S) \cup (R \cap T) = \{1\} \cup \{1\} = \{1,1\} \neq \{1\}$.

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 $\overset{o}{\bowtie}$.

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

$R =$

A	*B*
1	2
3	4

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

A+B	*A*1	*A*2
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*

A	*B*
1	2
3	4

$\pi_{Max(A),\ Min(B)}(R) =$

Max(A)	*Min(B)*
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 $\gamma_{beer,AVG(price)}(R)$.

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

CS54100: Database Systems

Relational Algebra

8 February 2012

Prof. Chris Clifton

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 $\overset{\circ}{\bowtie}$.
- 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 $\overset{\circ}{\bowtie}$ *S* =

A	*B*	*C*	
3	4	5	*part of natural join*
1	2	$\perp$	*part of right-outerjoin*
$\perp$	6	7	*part of left-outerjoin*

Division Operator

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

"Breaking" the Model

- Some SQL constructs break the traditional relational model

```
select bar
from sells
where beer in
   (select favorite_beer from drinkers);
```

- What is the equivalent relational algebra?
  - Why does it break the model?

CS54100

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	UNARY	FUNDAMENTAL
X CARTESIAN PRODUCT	BINARY	FUNDAMENTAL
U UNION	BINARY	FUNDAMENTAL
– SET-DIFFERENCE	BINARY	FUNDAMENTAL

$\cap$ SET-INTERSECTION	CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS
$\bowtie_\theta$ THETA-JOIN	CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS
$\bowtie$ NATURAL JOIN	CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS
$\div$ DIVISION or QUOTIENT	CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS

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 $\overset{o}{\bowtie}$.

Relational Calculus

- Two flavors
  - Domain Relational Calculus
  - Tuple Relational Calculus

CS54100

Tuple Relational Calculus

- Query: $\{ t \mid P(t) \}$
  - All tuples such that P is true for t
  - t[A] denotes value of attribute t for a
  - $t \in r$ denotes t is in relation r
  - *P* similar to predicate calculus
- Quantifiers
  - $\exists t \in r(Q(t))$
    - There is a tuple in *r* such that *Q(t)* holds
  - $\forall t \in r(Q(t))$
    - *Q(t)* holds for all tuples in *r*

CS54100

Domain Relational Calculus

- Query: $\{ <x_1, \ldots, x_n> \mid P(x_1, \ldots, x_n)\}$
  - x_i are domain variables
  - P is a predicate

CS54100

Safety of Expressions

- Calculus expressions meeting certain conditions are “safe”
  - Processing clearly defined
  - Essentially requires that all values in an expression appear in predicate or relation

CS54100