Relational Algebra

- Operators to "manipulate" tables.
- Relational operators can be looked as analogous to the arithmetic operators which are used to manipulate numbers.

"Core" Relational Algebra

1. Union, intersection, and difference: Tables are sets of rows – thus, the usual set operators can be used (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.

By default, relational algebra is set semantics – i.e., the tables are looked upon as sets (rather than bags).

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

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

Selection – key points.

$$R_1 = \sigma_C(R_2)$$

- Works by considering one row at a time of R_2
- Thus, condition C involves only attributes of R_2 or constants. No sub-queries!

Projection

$$R_1 = \pi_L(R_2)$$

where L is a *list* of attributes of R_2.

Example

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

beer	price
Bud	2.50
Miller	2.75
Coors	3.00

Notice elimination of duplicate tuples (since tables are sets of rows)

Cartesian 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 $t_1 t_2$.

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

Natural Join

$$R = R_1 \bowtie R_2$$

is the theta-join of R_1 and R_2 with:

- Condition: all attributes of the same name be equated.
- ONLY one column for each pair of equated attributes is retained.

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 ⋈ 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 $A_1,\ldots,A_n$.

Bars =

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

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

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

The name of the second relation is R.

Example

Sells(bar, beer, price)

Bars(name, addr)

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

Example

Sells(bar, beer, price)

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

Steps for Writing Queries

1. Divide and Conquer:
   - Break down the question into sub-questions.
2. Complement.
   - See if writing the "complement" is easier. E.g., if the query involves looking at an indeterminate number of tuples.
3. Visualize the required "pattern" in the table.

Examples

Sells (bar, beer).

1. Find bars that do not sell Miller.
2. Find bars that sell only Miller.
3. Find bars (b, b1) s.t. b doesn't sell any beer that b1 sells.
4. Find bars (b,b1) s.t. there is some beer that b1 sells but b doesn't sell.
5. Find bars (b, b1) s.t. b sells all beers that b1 sells.
6. Find bars (b, b1) s.t. b sells the same set of beers that b1 sells.

Linear Notation for Expressions

- Use variable names for intermediate 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

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 can be made to work on bags as well (bag-semantics).

  Just work on a tuple-by-tuple basis, and don't eliminate duplicates.

Bag Union, Intersection and Difference

- Union: Sum the number of duplicates.

  Example: {1,2,1} ∪ {1,2,3,3} = {1,1,1,2,2,3,3}.

- Intersection: Minimum of the number of copies.

  Example: {1,2,1} ∩ {1,2,3,3} = {1,2}.

- Difference: Proper-subtract the number of copies.

  Example: {1,2,1} – {1,2,3,3} = {1}.

Extended Relational Algebra

Other operators (for bags, SQL).

- Duplicate-elimination operator δ.
- Grouping-and-aggregation operator γ.
- Outerjoin, Others (skipped).

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

Aggregate Operator

$$\gamma_{A,\,F(B)}(R)$$

where A and B are attributes, and F is an aggregation operator (e.g., sum, max, ..). The above is computed as:

1. Partition *R* into groups of tuples based on A values.
2. From each group, output one tuple:

   < Common A-value, F(B values of the group)>

- Generalization: A can be a set of attributes, and we could have a lists of F(B)'s viz. f(B), g(C), etc.

Aggregate Operator: 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

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

1. Group by beer as follows:

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 avg. of price within groups:

beer	AVG(price)
Bud	2.25
Miller	3.00
Coors	3.00