"Core" Relational Algebra

A small set of set-based 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.

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

Projection

$$R_1 = \pi_L(R_2)$$

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

Example

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

beer	price
Bud	2.50
Miller	2.75
Coors	3.00

- Notice elimination of duplicate tuples.

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

$$R = R_1 \underset{C}{\bowtie} R_2$$

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.

$$\texttt{BarInfo = Sells} \underset{Sells.bar=Bars.name}{\bowtie} \texttt{Bars}$$

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

$$R = R_1 \bowtie R_2$$

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.

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

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

- The name of the above relation is R.

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, $\bowtie$, $\underset{C}{\bowtie}$, 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 $R \cup (\sigma(S) \bowtie T)$.

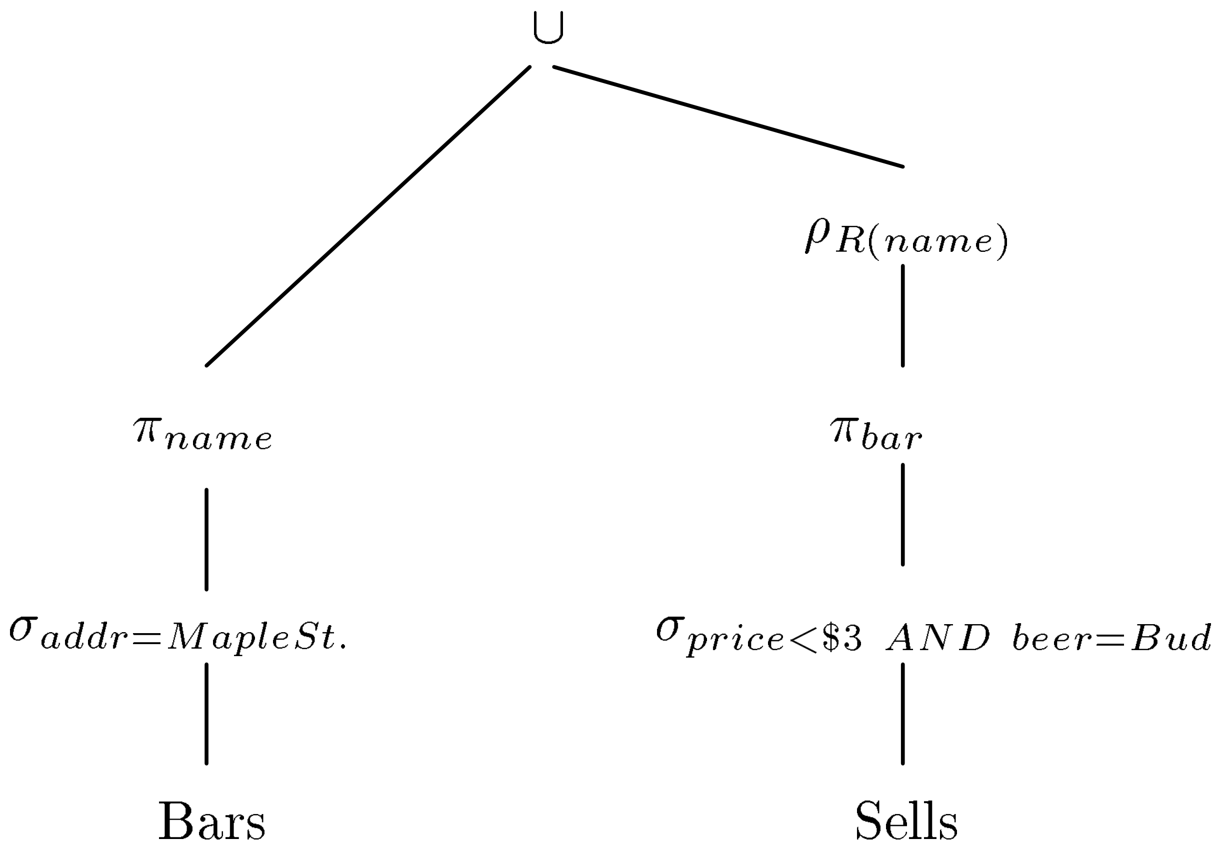
Each Expression Needs a Schema

- 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.

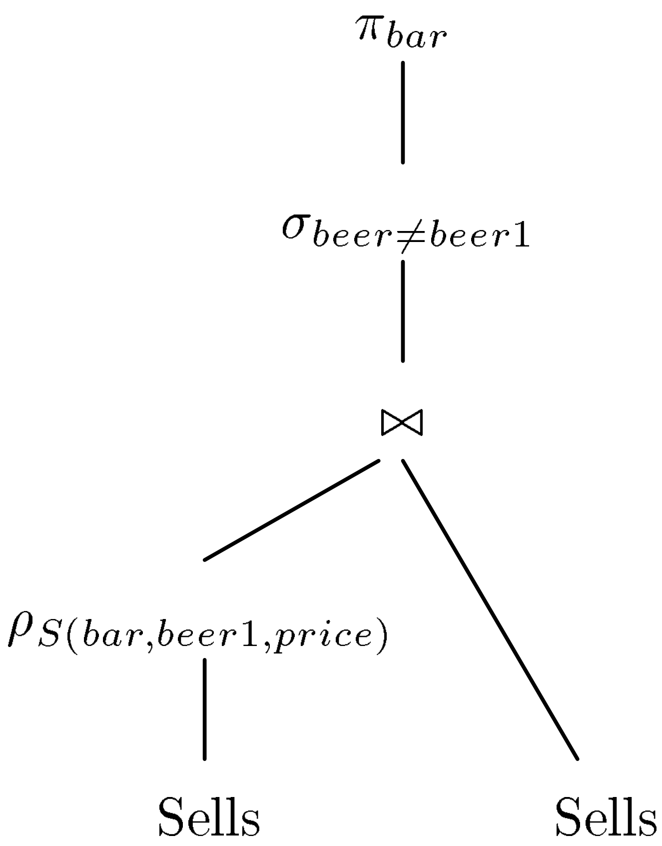
```
Sells(bar, beer, price)
Bars(name, addr)
```



Example

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

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



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) := $\pi_{name}(\sigma_{addr=Maple\ St.}(\texttt{Bars}))$
R2(name) :=
$\pi_{bar}(\sigma_{beer=Bud\ AND\ price<\$3}(\texttt{Sells}))$
Answer(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 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: $\{1,2,1\} \cup \{1,2,3\} = \{1,1,1,2,2,3\}$.

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 $\overset{\circ}{\bowtie}$.

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

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

$A + B$	$A1$	$A2$
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.

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

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 (resp., 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
1	2	$\perp$
$\perp$	6	7