Design Problem

OK; let's design a relational DB schema for beers-bars-drinkers.

- Drinkers have unique names and addresses. They like one or more beers and frequent one or more bars. They have phones, usually one but sometimes more or none.
- Bars have unique names and addresses. They serve one or more beers and are frequented by one or more drinkers. They charge a price for each beer they serve, which may vary from beer to beer.
- Beers have unique names and manufacturers. Manufacturers have unique names and addresses. Beers are served by one or more bars and are liked by one or more drinkers.

Relational Algebra

A small set of operators that allow us to manipulate relations in limited, but easily implementable and 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 $R \cup (\sigma(S) \bowtie T)$.

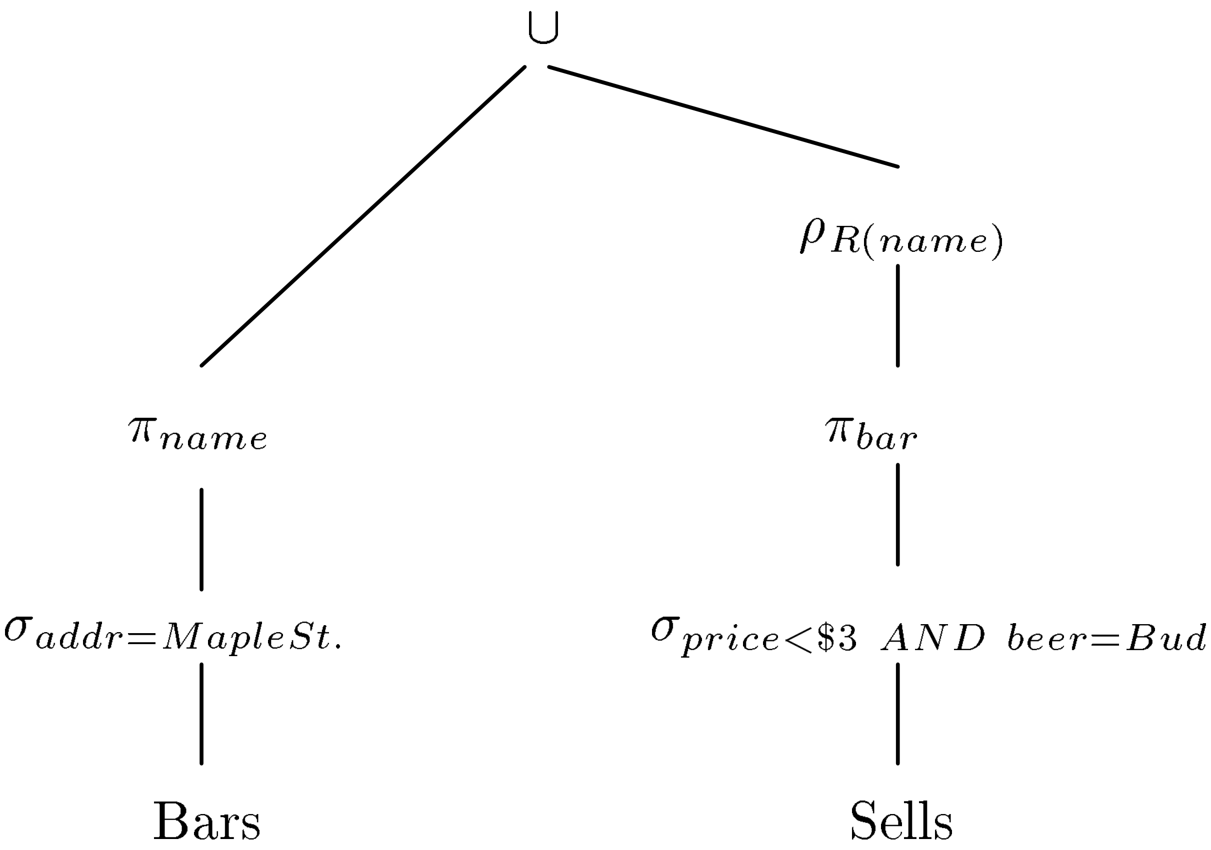
Each Expression Needs a Schema

- 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.

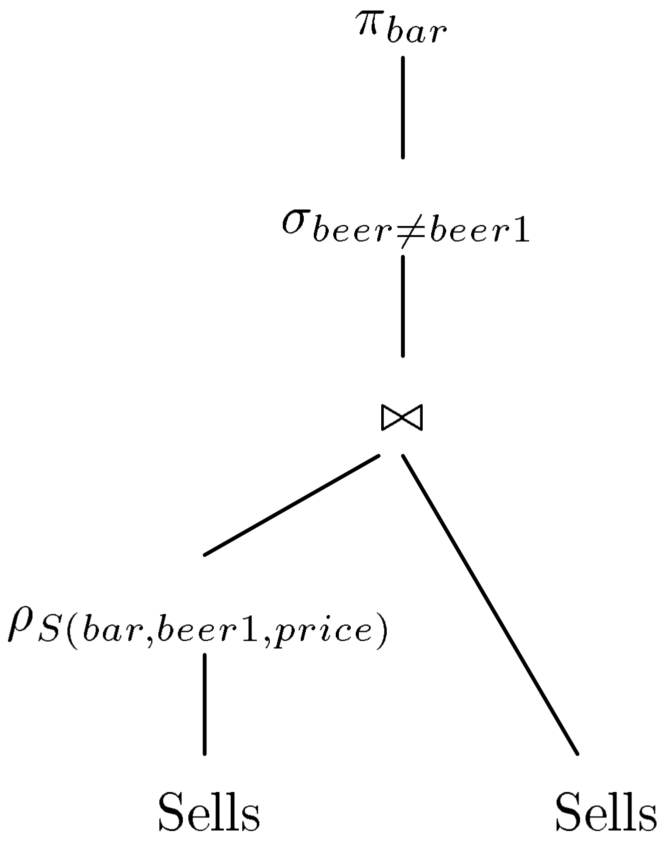
```
Sells(bar, beer, price)
Bars(name, addr)
```



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

R2(bar) :=

$\quad\pi_{bar}(\sigma_{beer=Bud\ AND\ price<\$3}(\texttt{Sells}))$

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

Example

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

```
Sells(bar, beer, price)

S1(bar,beer1,price) := Sells
S2(bar,beer,price,beer1) :=
    S1 ⋈ Sells
```

S3(bar) = $\pi_{bar}(\sigma_{beer \neq beer1}(S2))$