Finding All Implied FD's

Motivation: Suppose we have a relation $ABCD$ with some FD's F. If we decide to decompose $ABCD$ into ABC and AD, what are the FD's for ABC, AD?

- Example: $F = AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$. It looks like just $AB \rightarrow C$ holds in ABC, but in fact $C \rightarrow A$ follows from F and applies to relation ABC.
- Problem is exponential in worst case.

Algorithm

For each set of attributes X compute X^+.

- Add $X \rightarrow A$ for each A in $X^+ - X$.
- Ignore or drop some "obvious" dependencies that follow from others:

1. *Trivial FD's*: right side is a subset of left side.
   - Consequence: no point in computing $\emptyset^+$ or closure of full set of attributes.
2. Drop $XY \rightarrow A$ if $X \rightarrow A$ holds.
   - Consequence: If X^+ is all attributes, then there is no point in computing closure of supersets of X.
3. Ignore FD's whose right sides are not single attributes.

- Notice that after we project the discovered FD's onto some relation, the FD's eliminated by rules 1, 2, and 3 can be inferred *in the projected relation.*

Example

Example: $F = AB \to C$, $C \to D$, $D \to A$. What FD's follow?

- $A^+ = A$; $B^+ = B$ (nothing).
- $C^+ = ACD$ (add $C \to A$).
- $D^+ = AD$ (nothing new).
- $(AB)^+ = ABCD$ (add $AB \to D$; skip all supersets of AB).
- $(BC)^+ = ABCD$ (nothing new; skip all supersets of BC).
- $(BD)^+ = ABCD$ (add $BD \to C$; skip all supersets of BD).
- $(AC)^+ = ACD$; $(AD)^+ = AD$; $(CD)^+ = ACD$ (nothing new).
- $(ACD)^+ = ACD$ (nothing new).
- All other sets contain AB, BC, or BD, so skip.
- Thus, the only interesting FD's that follow from F are: $C \to A$, $AB \to D$, $BD \to C$.

Decomposition to Reach BCNF

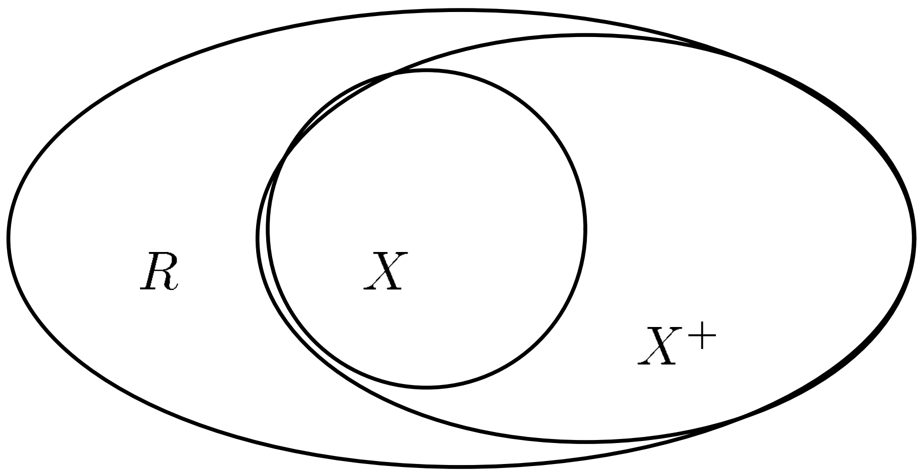
Setting: relation R, given FD's F. Suppose relation R has BCNF violation $X \to A$.

- We need only look among FD's of F for a BCNF violation.
- Proof: If $Y \to A$ is a BCNF violation and follows from F, then the computation of Y^+ used at least one FD $X \to B$ from F.
  - X must be a subset of Y.
  - Thus, if Y is not a superkey, X cannot be a superkey either, and $X \to B$ is also a BCNF violation.

1. Compute X^+.

   - Cannot be all attributes — why?

2. Decompose R into X^+ and $(R - X^+) \cup X$.



3. Find the FD's for the decomposed relations.

   - Project the FD's from F = calculate all consequents of F that involve only attributes from X^+ or only from $(R - X^+) \cup X$.

Example

R = `Drinkers(`<u>`name`</u>`, addr, `<u>`beersLiked`</u>`, manf, favoriteBeer)`

F =

1. `name` $\rightarrow$ `addr`
2. `name` $\rightarrow$ `favoriteBeer`
3. `beersLiked` $\rightarrow$ `manf`

Pick BCNF violation `name` $\rightarrow$ `addr`.

- Close the left side: `name`$^+$ = `name addr favoriteBeer`.
- Decomposed relations:

  `Drinkers1(`<u>`name`</u>`, addr, favoriteBeer)`
  `Drinkers2(`<u>`name`</u>`, `<u>`beersLiked`</u>`, manf)`

- Projected FD's (skipping a lot of work that leads nowhere interesting):
  - For `Drinkers1`: `name` $\rightarrow$ `addr` and `name` $\rightarrow$ `favoriteBeer`.
  - For `Drinkers2`: `beersLiked` $\rightarrow$ `manf`.

- BCNF violations?
  - For `Drinkers1`, `name` is key and all left sides of FD's are superkeys.
  - For `Drinkers2`, {`name`, `beersLiked`} is the key, and `beersLiked` → `manf` violates BCNF.

Decompose `Drinkers2`

- Close $\texttt{beersLiked}^+ = \texttt{beersLiked, manf}$.
- Decompose:

  Drinkers3(<u>beersLiked</u>, manf)
  Drinkers4(<u>name</u>, <u>beersLiked</u>)

- Resulting relations are all in BCNF:

  Drinkers1(<u>name</u>, addr, favoriteBeer)
  Drinkers3(<u>beersLiked</u>, manf)
  Drinkers4(<u>name</u>, <u>beersLiked</u>)

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

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

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.

`BarInfo = Sells` $\underset{Sells.Bar=Bars.Name}{\bowtie}$ `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 $R \cup (\sigma(S) \bowtie T)$.

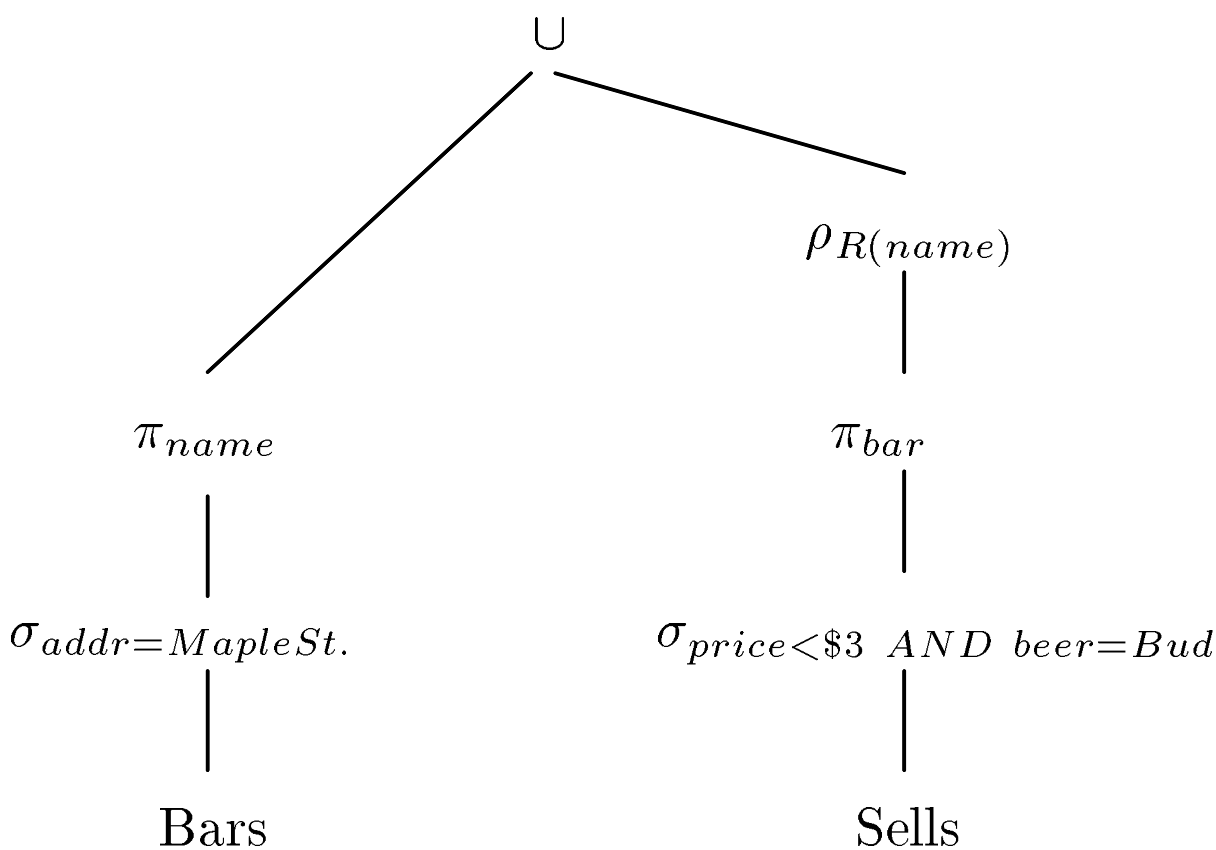
Each Expression Needs a Schema

- 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.

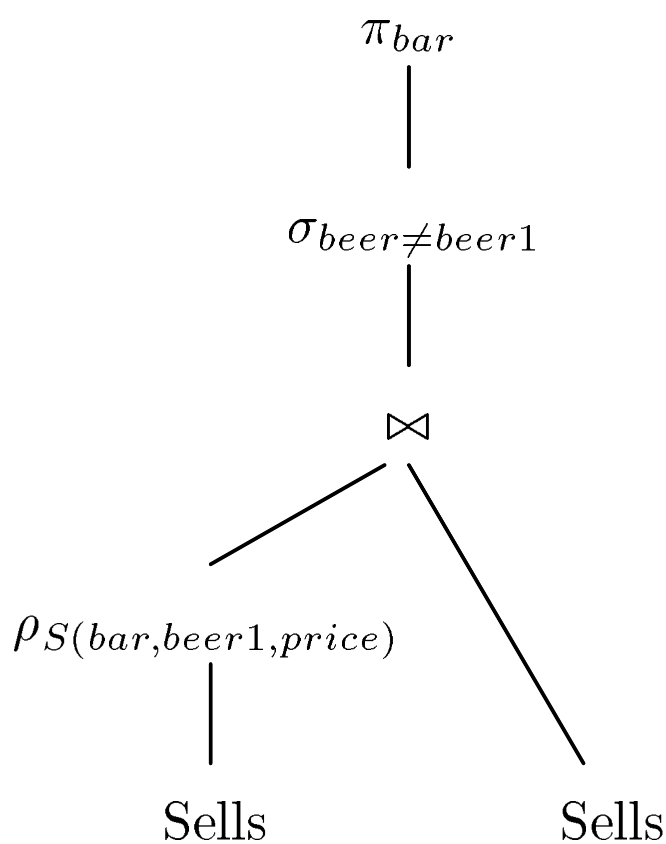
```
Sells(bar, beer, price)
Bars(name, addr)
```



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)
```

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

$\texttt{R2(bar)} :=$

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

$\texttt{R3(bar)} := \texttt{R1} \cup \texttt{R2}$

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