CS 589, Principles of Database Systems, Unit 2, Notes 2

Can Also Test if FDs imply JD

Use slightly different convention for distinguished variables (dvs)

Special variables, one in each column

- marked with an underline
- will use for a test for a JD

Tableau for JD ⋈[R1, R2, … , Rn]

Has n rows, w1, w2, …, wn

wi[Ri] = distinguished variables

wi[R - Ri] = unique non-distinguished variables (ndvs)

Example JD Tableau

`⋈[AB, BCD, DE]`

A	**B**	**C**	**D**	**E**
$\underline{a}$	$\underline{b}$	*c1*	*d1*	*e1*
a1	$\underline{b}$	$\underline{c}$	$\underline{d}$	*e2*
a2	*b1*	*c2*	$\underline{d}$	$\underline{e}$

If you want to see if FDs F imply a JD J

- Do the chase of the tableau with FD-rules for F
- Always replace ndv by dv
- See if you get the row of all dvs

Example Chase

F = {B → C, C → E, D → C, CE → D}

Apply B → C

A	B	C	D	E
$\underline{a}$	$\underline{b}$	$\underline{c}$	*d1*	*e1*
a1	$\underline{b}$	$\underline{c}$	$\underline{d}$	*e2*
a2	*b1*	*c2*	$\underline{d}$	$\underline{e}$

Final result

A	B	C	D	E
$\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{d}$	$\underline{e}$
a1	$\underline{b}$	$\underline{c}$	$\underline{d}$	$\underline{e}$
a2	*b1*	$\underline{c}$	$\underline{d}$	$\underline{e}$

We produced a row of all dvs, so the JD is implied by F.

What's going on here?

JD Rule

Can also use JDs to modify a tableau

- FD-rule: equate variables
- JD-rule: add rows

$\bowtie$[R1, R2, …, Rn]

- Find rows w1, w2, …, wn in tableau U
- and find a row v (not already in U) where
  v[R1] = w1[R1]
  v[R2] = w2[R2]
  …
  v[Rn] = wn[Rn]
- then add v to U

JD-rule Example

M = {$\bowtie$[ABC, CD], BD → A}

Implies BC → A? Implies BC → D?

	A	B	C	D
w1	*a1*	*b1*	*c1*	*d1*
w2	*a2*	*b1*	*c1*	*d2*

Can apply JD-rule two ways

w1[ABC], w2[CD]
v1 = <a1 b1 c1 d2>
w2[ABC], w1[CD]
v2 = <a2 b1 c1 d1>

JD-rule Example cont.

With additional rows:

	A	B	C	D
w1	*a1*	*b1*	*c1*	*d1*
w2	*a2*	*b1*	*c1*	*d2*
v1	*a1*	*b1*	*c1*	*d2*
v2	*a2*	*b1*	*c1*	*d1*

Apply BD → A to set a2 equal to a1

	A	B	C	D
w1	*a1*	*b1*	*c1*	*d1*
w2	*a1*	*b1*	*c1*	*d2*
v1	*a1*	*b1*	*c1*	*d2*
v2	*a1*	*b1*	*c1*	*d1*

JD-rule Example cont.

Remove duplicate rows:

	A	B	C	D
w1	*a1*	*b1*	*c1*	*d1*
w2	*a1*	*b1*	*c1*	*d2*

See that BC → A, but not BC → D

Facts about the Chase

- Order of rule application doesn't matter
  Need consistent rule for variable replacement, e.g.
  - lower # replaces higher #
  - dvs replace ndvs
- Chase can be large if there are JDs
- Complete for JD and FD inference
- Can be used to simplify queries and fill in missing instance information

Query Simplification Example

```
select a1.D, a2.P
from assign as a1, a2, a3
where a1.P = Chin and
 a1.L = a2.L and a2.D = 8may and
 a2.P = a3.P and a3.D = 8may and
 a1.T = a3.T
```

Tableau for this (allow constants)

assign(	PILOT	FLT	DATE	TIME)
a1	Chin	*f1*	*d1*	*t1*
a2	*p1*	*f1*	8may	*t2*
a3	*p1*	*f2*	8may	*t1*
result	*p1*		*d1*	

Apply Chase to Query Tableau

F = {L → T, PDT → L, LD → P}

Apply L → T

assign(	**PILOT**	**FLT**	**DATE**	**TIME)**
a1	Chin	*f1*	*d1*	*t1*
a2	*p1*	*f1*	8may	*t1*
a3	*p1*	*f2*	8may	*t1*
result	*p1*		*d1*	

Apply PDT → L

assign(	**PILOT**	**FLT**	**DATE**	**TIME)**
a1	Chin	*f1*	*d1*	*t1*
a2	*p1*	*f1*	8may	*t1*
a3	*p1*	*f1*	8may	*t1*
result	*p1*		*d1*	

Remove Duplicate Rows from Result

assign(	**PILOT**	**FLT**	**DATE**	**TIME)**
a1	Chin	*f1*	*d1*	*t1*
a2,a3	*p1*	*f1*	8may	*t1*
result	*p1*		*d1*	

As SQL:

```
select a2.P, a1.D
from assign as a1, a2
where a1.P = Chin and
  a1.L = a2.L and a2.D = 8may and
  a1.T = a2.T
```

Inclusion Dependencies

Similar to referential integrity constraints in SQL databases

Usually go between two relations: values in one must appear in the other

Example Inclusions

sched(FLIGHT	**FROM**	**TO**	**DEPART**	**ARRIVE)**
83	PDX	SEA	10:15a	11:00a
116	SEA	PDX	1:25p	2:05p
281	SEA	PDX	5:50a	6:30a
301	PDX	SEA	6:35p	7:15p
319	PDX	SFO	9:30a	11:15a
412	SFO	PDX	1:25p	3:10p

dist(AIRPORT1	**AIRPORT2**	**MILES)**
PDX	SEA	155
SEA	PDX	155
PDX	SFO	585
SFO	PDX	585

Also Recall

assigned(	PILOT	FLIGHT	DATE	TIME)
	Cushing	83	9 Aug	10:15a
	Clark	83	11 Aug	10:15a
	Chin	83	13 Aug	10:15a
	Cushing	116	10 Aug	1:25p
	Chin	116	12 Aug	1:25p
	Clark	281	8 Aug	5:50a
	Copley	281	9 Aug	5:50a
	Copley	281	13 Aug	5:50a
	Clark	301	12 Aug	6:35p
	Copley	412	15 Aug	1:25p

Inclusions Dependencies for Example

Assigned flights must be in schedule

assigned[L T] $\subseteq$ sched[L DEP]

Scheduled flights must have distances

sched[FR TO] $\subseteq$ dist[A1 A2]

Must have same airports in both distance columns

dist[A1] $\subseteq$ dist[A2], dist[A2] $\subseteq$ dist[A1]

But, not every scheduled flight is necessarily assigned

sched[L DEP] $\not\subset$ assigned[L T]

Axioms for Inclusion Dependencies

Attribute order is important

I1. Reflexivity: $r[X] \subseteq r[X]$

sched[DEP ARR] $\subseteq$ sched[DEP ARR]

I2. Permutation: Change attribute order on both sides (in the same way)

assigned[L T] $\subseteq$ sched[L DEP] implies
assigned[T L] $\subseteq$ sched[DEP L]

I3. Projection: Remove attributes on both sides (in the same places)

sched[FR TO] $\subseteq$ dist[A1 A2] implies
sched[FR] $\subseteq$ dist[A1]

Axioms Continued

I4. Transitivity: $q[X] \subseteq r[Y]$ and $r[Y] \subseteq s[Z]$ imply $q[X] \subseteq s[Z]$

sched[FR] $\subseteq$ dist[A1] and
dist[A1] $\subseteq$ dist[A2]imply
sched[FR] $\subseteq$ dist[A2]