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

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 $\bowtie[R_1, R_2, \ldots, R_n]$

Has n rows, $w_1, w_2, \ldots, w_n$

$w_i[R_i]$ = distinguished variables

$w_i[R - R_i]$ = unique *non-distinguished* variables (ndvs)

Example JD Tableau

$\bowtie$[AB, BCD, DE]

A	B	C	D	E
<u>a</u>	<u>b</u>	c1	d1	e1
a1	<u>b</u>	<u>c</u>	<u>d</u>	e2
a2	b1	c2	<u>d</u>	<u>e</u>

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 a row of all dvs

Example Chase

F = {B → C, C → E, D → C, CE → D}

Apply B → C

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e2
a2	b1	c2	d	e

Final result

A	B	C	D	E
a	b	c	d	e
a1	b	c	d	e
a2	b1	c	d	e

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

JD Rule

Can also use JDs to modify a tableau
- FD-rule: equate variables
- JD-rule: add rows

$\bowtie[R_1, R_2, \ldots, R_n]$
- Find rows $w_1, w_2, \ldots, w_n$ in tableau U
- and find a row v (not already in U) where
  - $v[R_1] = w_1[R_1]$
  - $v[R_2] = w_2[R_2]$
  - …
  - $v[R_n] = w_n[R_n]$
- then add v to U

JD-rule Example

$M = \{\bowtie[ABC, CD], BD \rightarrow A\}$

Implies $BC \rightarrow A$? Implies $BC \rightarrow D$?

	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 implication
- Can be used to simplify queries and fill in missing instance information

Query Simplification Example

```
select as1.D, as2.P
from assigned as as1, as2, as3
where as1.P = Chin and
  as1.L = as2.L and as2.D = 8may and
  as2.P = as3.P and as3.D = 8may and
  as1.T = as3.T
```

Tableau for this (allow constants)

assigned(	PILOT	FLT	DATE	TIME)
as1	Chin	*f1*	*d1*	*t1*
as2	*p1*	*f1*	8may	*t2*
as3	*p1*	*f2*	8may	*t1*
result	*p1*		*d1*	

Apply Chase to Query Tableau

F = {L → T, PDT → L, LD → P}

Apply L → T

assigned(	PILOT	FLT	DATE	TIME)
as1	Chin	*f1*	*d1*	*t1*
as2	*p1*	*f1*	8may	*t1*
as3	*p1*	*f2*	8may	*t1*
result	*p1*		*d1*	

Apply PDT → L

assigned(	PILOT	FLT	DATE	TIME)
as1	Chin	*f1*	*d1*	*t1*
as2	*p1*	*f1*	8may	*t1*
as3	*p1*	*f1*	8may	*t1*
result	*p1*		*d1*	

Remove Duplicate Rows from Result

assigned(	PILOT	FLT	DATE	TIME)
as1	Chin	*f1*	*d1*	*t1*
as2,as3	*p1*	*f1*	8may	*t1*
result	*p1*		*d1*	

As SQL:

```
select as2.P, as1.D
from assigned as as1, as2
where as1.P = Chin and
  as1.L = as2.L and as2.D = 8may and
  as1.T = as2.T
```

Inclusion Dependencies

- Generalizes foreign keys 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 this Example

Assigned flights must be in schedule

```
assigned[L T] ⊆ sched[L DEP]
```

Scheduled flights must have distances

```
sched[FR TO] ⊆ dist[A1 A2]
```

Must have same airports in both distance columns

```
dist[A1] ⊆ dist[A2],
 dist[A2] ⊆ dist[A1]
```

But, not every scheduled flight is necessarily assigned

```
sched[L DEP] ⊄ assigned[L T]
```

Axioms for Inclusion Dependencies

Attribute order is important

I1. Reflexivity: r[X] ⊆ r[X]

sched[DEP ARR] ⊆ sched[DEP ARR]

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

assigned[L T] ⊆ sched[L DEP] implies
assigned[T L] ⊆ sched[DEP L]

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

sched[FR TO] ⊆ dist[A1 A2] implies
sched[FR] ⊆ dist[A1]

Axioms Continued

I4. Transitivity: q[X] ⊆ r[Y] and r[Y] ⊆ s[Z] imply q[X] ⊆ s[Z]

sched[FR] ⊆ dist[A1] and
dist[A1] ⊆ dist[A2]imply
sched[FR] ⊆ dist[A2]