# CS 589, Principles of Databases, Notes 2-3



### Implications of FDs and IDs

Haven't discussed how infinite relations affect implication

- Examples have all been finite sets of tuples
- Hasn't mattered for combinations of dependencies seen so far

It does make a difference when dealing with FDs and IDs together





### SAT

Go back to implication as containment of sets of instances

SAT(M, R) = all instances over scheme R that satisfy all dependencies in M

R will usually be understood, so we write SAT(M)

Have: M implies dependency d if and only if SAT(M) $\subseteq$ SAT({d})

### Finite Implication

Let FSAT(M) = all finite instances that satisfy all dependencies in M

**Definition**: M *finitely implies* dependency d if FSAT(M) $\subseteq$ FSAT({d}).

Can have "finitely implies" without (regular) "implies"

But not vice versa





### First, a Fact

If r is finite and r satisfies A → B, then the number of distinct A-values in r is greater or equal to the number of distinct B-values.

$|r[A]| \geq |r[B]|$

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |
| a3 | b1 |

## Finite Implication without Implication

### Relation r(AB)

M = {A → B, r[A] ⊆ r[B]}
d = r[B] ⊆ r[A]

### Part 1. Assume r is finite

Since A → B, then |r[A]| ≥ |r[B]|
Since r[A] ⊆ r[B], |r[A]| ≤ |r[B]|

So, |r[A]| = |r[B]| and the inclusion must be an equality: r[A] = r[B]

Thus, r[B] ⊆ r[A]





### Part 2. Allow r to be infinite

Here is a relation instance that satisfies {A → B, r[A] ⊆ r[B]}, but not r[B] ⊆ r[A]

| A | B |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 2 |
| … | |
| i | i-1 |
| … | |

### What Gives?

It is hard to express “r is finite” in normal logic

Thus generally hard to define FSAT[M] and reason about finite implication

Finite implication is often undecidable





### Null Values

Used to represent missing information

At least two kinds

- dne — does not exist
- unk — value exists but unknown

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | **unk** |
| **dne** | 704 | 6Oct | 5:50a |
| Cook | **dne** | **dne** | **dne** |

### “Does Not Exist” Null

Like adding an additional value to domain of an attribute

So it enlarges the set of possible relation instances





### Issue: dne = dne?

Maybe no: LD → P

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cook | **dne** | **dne** | **dne** |
| Cadiz | **dne** | **dne** | **dne** |

Maybe yes: FIRST MI LAST → AGE

| FIRST | MI | LAST | AGE |
|---|---|---|---|
| Alex | **dne** | Cook | 37 |
| Alex | **dne** | Cook | 39 |

## Unknown Null

More like variables

Can view an instance with **unk** nulls as a set of possible fully-defined instances

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | **unk** |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | **unk** | 6Oct | **unk** |





## Some Possible Instances

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | 5:00p |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | 872 | 6Oct | 11:00p |

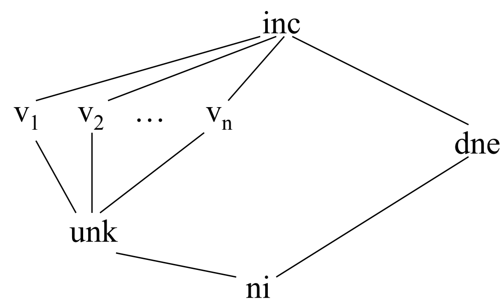
| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | 6:00p |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | 615 | 6Oct | 6:00p |

### Also, “No-Information” and “Inconsistent”

$x \sqsubseteq y$: y more informative than x

ni $\sqsubseteq$ unk $\sqsubseteq$ $v_i$ $\sqsubseteq$ inc

ni $\sqsubseteq$ dne $\sqsubseteq$ inc








### Extend Order to Tuples

Use $\sqsubseteq$ on each component

<Fred, unk, 6> $\sqsubseteq$ <Fred, 10.25, 6>
<Fred, unk, 6> $\sqsubseteq$ <Fred, unk, 6>
<ni, dne, 6> $\sqsubseteq$ <unk, inc, 6>
<ni, dne, 6> $\sqsubseteq$ <inc, inc, inc>

### Extending to Relations

Concentrate on unk null

```
ord1(Person Price Amt)
     Fred   unk   6
     Fred   10.25 unk
     Fritz  unk   unk

ord2(Person Price Amt)
     Fred   11.50 6
     unk    10.25 7

ord3(Person Price Amt)
     Fred   11.50 6
     Fred   10.25 7
     Fritz  12.25 7
```





### What Does a Tuple Mean?

For partial relations r, q, what should $r \sqsubseteq q$ require?

For tuple t in r

- exactly one s in q with $t \sqsubseteq s$?
- at least one s in q with $t \sqsubseteq s$?

For tuples t1, t2 in r, can there be a single s in q with $t1 \sqsubseteq s$ and $t2 \sqsubseteq s$?

Can there be s in q where there is no t in r such that $t \sqsubseteq s$?

### Book's Definition

"Fill in the blanks"

$r \sqsubseteq q$ means there is a total, onto map $\theta$ from r to q where for every tuple t in r

$$t \sqsubseteq \theta(t)$$





### Examples

| ord1 (Person | Price | Amt) | $\theta$ | ord3 (Person | Price | Amt) |
|---|---|---|---|---|---|---|
| Fred | unk | 6 | → | Fred | 11.50 | 6 |
| Fred | 10.25 | unk | → | Fred | 10.25 | 7 |
| Fritz | unk | unk | → | Fritz | 12.25 | 7 |

| ord1 (Person | Price | Amt) | ord4 (Person | Price | Amt) |
|---|---|---|---|---|---|
| Fred | unk | 6 | Fred | 10.25 | 6 |
| Fred | 10.25 | unk | Fritz | 12.25 | unk |
| Fritz | unk | unk | | | |

$\theta$

## But Consider

```
ord2(Person Price Amt)  ord3(Person Price Amt)
     Fred   11.50 6   ------> Fred   11.50 6
     unk    10.25 7   ------> Fred   10.25 7
                    ? ------> Fritz  12.25 7
                        θ
```





## Possible Worlds

View a partial relation as specifying a set of possible worlds

POSS(r) = {q | r $\sqsubseteq$ q and q is fully defined}

Can include constraints C

POSS(r, C) = {q | q in POSS(r) and q in SAT(C)}

### What Happens if We Have Constraints?

Recall

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | **unk** |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | **unk** | 6Oct | **unk** |





### Not All Satisfy the FDs

$F = \{L \rightarrow T, PDT \rightarrow L, LD \rightarrow P\}$

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | | 6Oct | |

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | | 6Oct | |

## Use FD-Chase on Instances with Unknowns

**r**

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 704 | 5Oct | **unk** |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | **unk** | 6Oct | **unk** |

**r'**

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 704 | 5Oct | 5:50a |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | **unk** | 6Oct | **unk** |

POSS(r, F) = POSS(r', F)





## Querying Relations with Unknowns

Let r be a relation with unknowns, and Q a query on full relations

Would like Q' on partial relations

Q'(r) represents {Q(q) | q in POSS(r)}

When r is fully defined, would like Q'(r) and Q(r) to agree. (Faithful)

## Relational Algebra

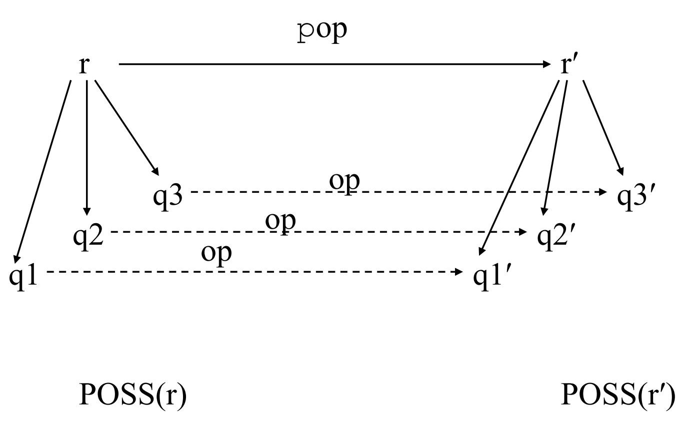
One approach: Multi-relation

- Let pop be partial-relation version of operator op
- $p\pi$ and $\pi$
- POSS(pop(r)) = {op(q) | q in POSS(r)}





## Diagram



## Sometimes Exists

r

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | **unk** |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | **unk** | 6Oct | **unk** |

$p\pi_{DT}(r)$

| DATE | TIME |
|---|---|
| 5Oct | **unk** |
| 6Oct | 5:50a |
| 6Oct | **unk** |





## Not Always

$\sigma_{L=872}(r)$

$r \sqsubseteq r1$

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | 5:00p |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | 872 | 6Oct | 11:00p |

ans(r1)

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 872 | 6Oct | 11:00p |

## Not Always

$\sigma_{L=872}(r)$

$r \sqsubseteq r2$

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| Cushing | 615 | 5Oct | 5:00p |
| Cook | 704 | 6Oct | 5:50a |
| Cushing | 615 | 6Oct | 5:00p |

ans(r2)

| PILOT | FLIGHT | DATE | TIME |
|---|---|---|---|
| ∅ | | | |





## Selection Conditions

t.B = 6

| A B | |
|---|---|
| <Fred 6> | true |
| <Fred 4> | false |
| <Fred unk> | maybe |

3-valued logic

| or | T | F | M |
|---|---|---|---|
| T | T | T | T |
| F | T | F | M |
| M | T | M | M |

## Example

```
     A     B
t1   <Fred unk>
t2   <Fritz unk>

     t.A = Fritz or t.B = 6
t1         F     or   M     = M
t2         T     or   M     = T
```





## But

```
t1   <Fred unk>

     t.B < 6 or t.B ≥ 6
t1         M     or   M     = M
```

But it would be T for any value for unk