# CS 589, Principles of Database Systems, Notes 2-1



**Lecture 2-1**
**Spring 2016**
**Dependencies and Inference**





## Model Theoretic View of Databases

A database schema is a theory or specification

The database state is a model or instance that satisfies the specification

- When is this view useful?
  - discussing constraints and implication
  - comparing representation capabilities of two database schemata
  - representing incomplete information: DB as partial model

relation name

## "Specification" Part

```
assigned(  PILOT    FLIGHT    DATE  TIME)
```

relation scheme

attribute

Each attribute A has a dom(A)

So the relation scheme defines attributes and allowable values for a conforming relation instance



Principles of Database Systems

relation instance

## Relation Instance

assigned(

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

Expect a relation instance to satisfy requirements of the schema

*Data dependencies*: further constraints that the schema puts on a satisfying instance

## Functional Dependency

**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 |

A generalization of keys: when tuples agree on certain attributes, must agree on others

FLIGHT → TIME (L → T)
PILOT DATE TIME → FLIGHT (PDT → L)
FLIGHT DATE → PILOT (LD → P)



Principles of Database Systems

## Functional Dependency Definition

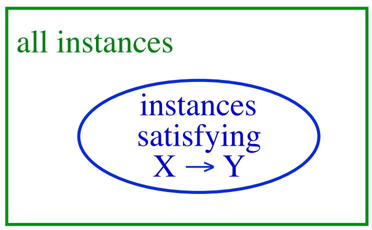
A relation instance r(R) satisfies the *functional dependency* (FD) X → Y (X and Y subsets of R), if

```
for every two tuples t,s in r,
if t[X] = s[X],
then t[Y] = s[Y]
```

## FDs as Specification

Usually, we don't ask which FDs a relation instance satisfies.

Rather, specify FDs that we expect all instances of to satisfy as part of the relation schema



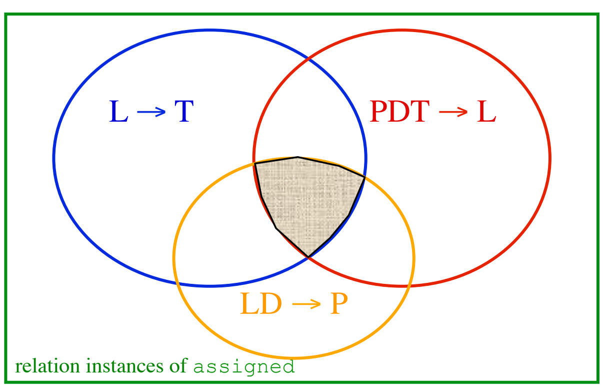





## Sets of FDs

A relation instance r satisfies a set of FDs F if it satisfies every FD in F

F = {L → T, PDT → L, LD → P}



## Implication

If an instance r satisfies FDs F, there may be another FD $X \to Y$ it necessarily satisfies

Say that F *implies* $X \to Y$ ($F \models X \to Y$)

```
Consider L → T
Let r be any relation instance
  satisfying it
take t,s in r where t[PL] = s[PL]
so t[L] = s[L]
by L → T, t[T] = s[T]
thus r also satisfies PL → T
```



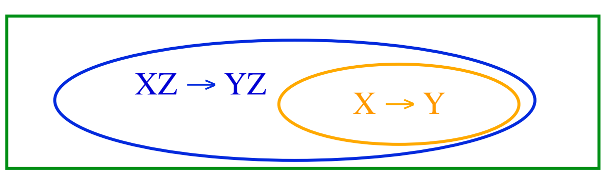


## Inference Axioms

General patterns that describe implications. For X, Y, Z, W sets of attributes

F1. Reflexivity: $X \to X$

F2. Augmentation: if $X \to Y$, then $XZ \to YZ$

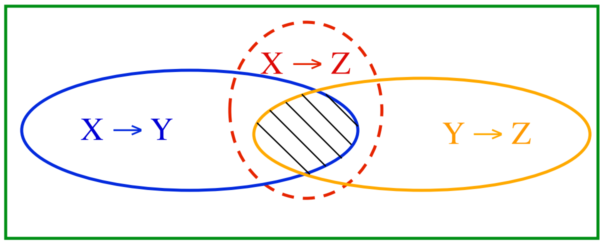



F3. Additivity: if $X \to Y$ and $X \to Z$, then $X \to YZ$

## More Axioms

F4. Projectivity: if $X \to YZ$, then $X \to Y$

F5. Transitivity: if $X \to Y$ and $Y \to Z$, then $X \to Z$




F6. Pseudotransitivity: if $X \to Y$ and $YZ \to W$, then $XZ \to W$





## Can Do Proofs

```
F = {L → T, PDT → L, LD → P}

1. L → T (in F)
2. LD → TD (augmentation from 1.)
3. LD → P (in F)
4. LD → PTD (additivity from 2.
                         and 3.)
```

## Derivation

Say that F *derives* $X \to Y$ if there is a proof of $X \to Y$ from FDs in F using inference axioms F1 - F6.

$F \vdash X \to Y$

Is implies the same as derives?

soundness

$F \models X \to Y$ if and only if $F \vdash X \to Y$

completeness





## Soundness

Prove that each inference axiom is correct. That is, it holds for any relation instance.

F5. If X → Y and Y → Z, then X → Z

Take any tuples t,s in r, where r satisfies X → Y and Y → Z.

Suppose t[X] = s[X]. Then from X → Y, must have t[Y] = s[Y].

But by Y → Z, then have t[Z] = s[Z].

So r satisfies X → Z.

## Definitions

*Closure of F*, $F^+$, is F plus all FDs that can be derived from F by F1. - F6.

So $F \vdash X \rightarrow Y$ means $X \rightarrow Y$ in $F^+$

*Closure of X*, $X^+$, is largest Z such that $X \rightarrow Z$ is in $F^+$

{L → T, PDT → L, LD → P}

What is $(LD)^+$?

What is $T^+$?





## Completeness

If it's true, you can prove it
  or, equivalently
If you can't prove it, it's not true.

### Instance to show not true

Take any $X \rightarrow Y$ over R not in $F^+$.

Consider $X^+$ and let $X^- = R - X^+$.

| | $X^+$ atts | $X^-$ atts |
|---|---|---|
| t | a1 a2 … an | b1 b2 … bm |
| s | a1 a2 … an | c1 c2 … cn |

bi ≠ ci

This relation instance violates $X \rightarrow Y$.

- $t[X] = s[X]$, but claim $t[Y] \neq s[Y]$
- Suppose $t[Y] = s[Y]$. Then $Y \subseteq X^+$
- Since $X \rightarrow X^+$ in $F^+$, so is $X \rightarrow Y$ by projectivity, a contradiction





### The instance satisfies $F^+$

Take any $W \rightarrow Z$ in $F^+$

Case 1. $W \not\subseteq X^+$.

- Then $t[W] \neq s[W]$, no problem

Case 2. $W \subseteq X^+$.

- Then $t[W] = s[W]$
- Have $X \rightarrow X^+$ in $F^+$, $X^+ \rightarrow W$ by reflexivity and projectivity, and $W \rightarrow Z$ in $F^+$.
- That gives $X \rightarrow Z$ by transitivity applied twice
- Additivity on $X \rightarrow X^+$ and $X \rightarrow Z$ gives $X \rightarrow X^+Z$

Case 2, continued

We have proved $X \rightarrow X^{+}Z$ is in $F^{+}$

But wait! $X^{+}$ is supposed to be the maximum set of attributes that X determines.

Must be that $Z \subseteq X^{+}$.

So $t[Z] = s[Z]$

Thus our instance satisfies $W \rightarrow Z$.





## Armstrong Relation for F

An "exact model" for F:

- Satisfies $F^{+}$
- Violates every FD not in F ($F^{-}$)

Use the 2-tuple instances in the last proof, combined into one big instance

Example $R = ABC$, $F = \{A \rightarrow B, B \rightarrow C\}$

Only need FDs in $F^{-}$ with one attribute on right side

$F^- = \{C \rightarrow A, C \rightarrow B, B \rightarrow A, BC \rightarrow A\}$

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c1 |
| a3 | b3 | c2 |
| a4 | b4 | c2 |
| a5 | b5 | c3 |
| a6 | b5 | c3 |
| a7 | b6 | c4 |
| a8 | b6 | c4 |





## Join Dependency

Note the redundancy in the `assigned` table: repeat time for each flight.

Can split up

asgn1( PILOT FLT DATE)

| PILOT | FLT | DATE |
|---|---|---|
| Cushing | 83 | 9 Aug |
| Clark | 83 | 11 Aug |
| Chin | 83 | 13 Au |
| Cushing | 116 | 10 Aug |
| Chin | 116 | 12 Aug |
| Clark | 281 | 8 Aug |
| Copley | 281 | 9 Aug |
| Copley | 281 | 13 Aug |
| Clark | 301 | 12 Aug |
| Copley | 412 | 15 Aug |

asgn2(FLT TIME)

| FLT | TIME |
|---|---|
| 83 | 10:15a |
| 116 | 1:25p |
| 281 | 5:50a |
| 301 | 6:35p |
| 412 | 1:25p |

## Can Recover with a Join

assigned = asgn1 ⋈ asgn2

assigned satisfies a *join dependency* (JD)

```
⋈[PLD, LT]

assigned =
 πPLD(assigned) ⋈ πLT(assigned)
```

$$assigned = \pi_{PLD}(assigned) \bowtie \pi_{LT}(assigned)$$

A "lossless" join





## What about ⋈[PL, PDT]?

```
asgn3(PILOT   FLT) asgn4(PILOT    DATE   TIME)
     Cushing  83      Cushing   9 Aug 10:15a
     Cushing 116      Cushing  10 Aug  1:25p
       . . .             . . .


asgn3⋈asgn4(PILOT   FLT   DATE    TIME)
           Cushing  83   9 Aug  10:15a
           Cushing  83  10 Aug   1:25p
                . . .
```

### Join Dependency, General Form

r(R), with R1, R2, …, Rn subsets of R

r satisfies $\bowtie$[R1, R2, …, Rn] if

$r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ldots \bowtie \pi_{Rn}(r)$

notice that always have

$r \subseteq \pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ldots \bowtie \pi_{Rn}(r)$

A *multi-valued dependency (MVD)* is a special case where n = 2.



Principles of Database Systems

### Implications of FDs and JDs

There are some axioms

```
X → Y on R, Z = R - XY
FJ1. X → Y implies ⋈[XY, XZ]

Example
L → T, so ⋈[LT, LPD]
```

However, there is no finite, complete set of axioms for just FDs and JDs (or JDs alone).

### Are We Stuck?

No

- There are complete axiom sets for FDs and multi-valued dependencies.
- There are complete axiom sets for classes of constraints that include FDs and JDs.
- There is an *inference procedure* for FD and JD implication.
  not based on axioms





### Need to Use Tableaux

A tableau is a "template" for a relation

Has *rows* of <u>variables</u> instead of tuples of <u>values</u>

| U( | P | L | D | T) |
|---|---|---|---|---|
| v | *p1* | *f1* | *d1* | *m1* |
| w | *p2* | *f1* | *d2* | *m2* |

Treat tableau rows like tuples

w(L) = *f1*

## Valuation

A *valuation* ρ for tableau U maps its variables to domain values.

ρ(v) must be in dom(A) if v in A column

ρ(*p1*)=Chin ρ(*f1*)=86 ρ(*d1*)=1 June ρ(*m1*)=4p
ρ(*p2*)=Chao ρ(*d2*)=2 June ρ(*m2*)=4p





## Extend Valuation to Rows and Tableaux

ρ(w) = apply ρ to each variable in w

ρ(w) **(***A***)** = ρ(w**(***A***)**)

ρ(*p1 f1 d1 m1*) = <Chin 86 1June 4p>

ρ(U) = {ρ(w) | w in U}

| ρ(**U**) | **P** | **L** | **D** | **T** |
|---|---|---|---|---|
| ρ(v) | Chin | 86 | 1June | 4p |
| ρ(w) | Chao | 86 | 2June | 4p |

### How About This Valuation?

ρ(*p1*)=Chin ρ(*f1*)=86 ρ(*d1*)=1 June ρ(*m1*)=4p
ρ(*p2*)=Chao ρ(*d2*)=2 June ρ(*m2*)=5p

| ρ(U) ( | P | L | D | T ) |
|---|---|---|---|---|
| ρ(v) | Chin | 86 | 1June | 4p |
| ρ(w) | Chao | 86 | 2June | 5p |





### Want to Enforce L → T

Need to make sure that ρ(m1) = ρ(m2).

Can we "fix" the tableau?

| U ( | P | L | D | T ) |
|---|---|---|---|---|
| v | *p1* | *f1* | *d1* | *m1* |
| w | *p2* | *f1* | *d2* | ***m1*** |

Have "applied" L → T to tableau U.

## The Chase

Apply FDs and JDs to a tableau, to reason about an example relation (or sub-relation)

Tableau rule for FD X → Y

- For rows v and w
- If v[X] = w[X], make v[Y] = w[Y] by equating variables in Y
- [Book gives rules for replacing variables]

*FD chase* of tableau U for F: Apply FD rules for FDs in F to U until no change. (*Why must this terminate?*)





## Example

{AB → E, AG → C, BE → D, E → G, DG → H}

| A | B | C | D | E | G | H |
|---|---|---|---|---|---|---|
| *a1* | *a2* | *a3* | *a4* | *a5* | *a6* | *a7* |
| *a1* | *b2* | *b3* | *b4* | *a5* | *b6* | *b7* |

Apply E → G

| A | B | C | D | E | G | H |
|---|---|---|---|---|---|---|
| *a1* | *a2* | *a3* | *a4* | *a5* | *a6* | *a7* |
| *a1* | *b2* | *b3* | *b4* | *a5* | ***a6*** | *b7* |

Apply AG → C

| A | B | C | D | E | G | H |
|---|---|---|---|---|---|---|
| a1 | a2 | a3 | a4 | a5 | a6 | a7 |
| a1 | b2 | **a3** | b4 | a5 | a6 | b7 |





## Testing FD Implication with the Chase

Method in book is ambiguous

Want to test $F \models X \rightarrow Y$ on scheme
R = A1 A2 … An

1. Set up a tableau $U_X$ with rows v, w:
   v(Ai) = ai
   w(Ai) = ai if Ai in X
   = bi otherwise
2. Chase $U_X$ with FDs in F
3. If v[Y] = w[Y], then $F \models X \rightarrow Y$
   (if not, we have a counterexample)

## Example

F = {AB → C, B → D, CD → E, CE → GH, G → A} ⊨ BG → C?

| **A** | **B** | **C** | **D** | **E** | **G** | **H** |
|---|---|---|---|---|---|---|
| *a1* | *a2* | *a3* | *a4* | *a5* | *a6* | *a7* |



Principles of Database Systems

## You Try It

F = {AB → E, AG → C, BE → D, E → G, DG → H}

| **A** | **B** | **C** | **D** | **E** | **G** | **H** |
|---|---|---|---|---|---|---|
| *a1* | *a2* | *a3* | *a4* | *a5* | *a6* | *a7* |

Does F ⊨ BE → A?

Does F ⊨ BE → H? (can use same chase)