Unit 2
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
  - analyzing representation equivalence of two schemata
  - modeling incomplete information

Relational Notation

relation name

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

relation scheme

attribute

A *relation scheme* R is a set of attributes

Each attribute A ∈ R has an associated *domain* dom(A) of possible values

```
dom(PILOT) = string
dom(FLIGHT) = integer
```


tuple

Tuples

```
assigned(  PILOT     FLIGHT    DATE  TIME)
           Cushing    83       9 Aug 10:15a
```

A *tuple* t on relation scheme R is a function on R where t(A) ∈ dom(A) for A ∈ R

```
t(PILOT) = Cushing
t(DATE) = 9 Aug
```

By a stretch of notation

```
t(FLIGHT TIME) = <83 10:15a>
```

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

A *relation instance* r on scheme R is a set of tuples on R. Denote r(R).

relation = name + scheme + 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)

Functional Dependency Definition

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

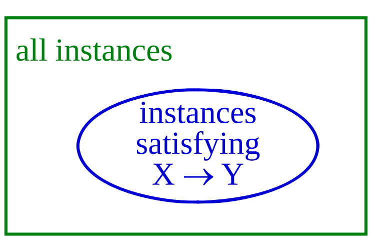
```
for any two tuples t,s in r,
if t(X) = s(X),
then t(Y) = s(Y)
```


Principles of Database Systems

FDs as Specification

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

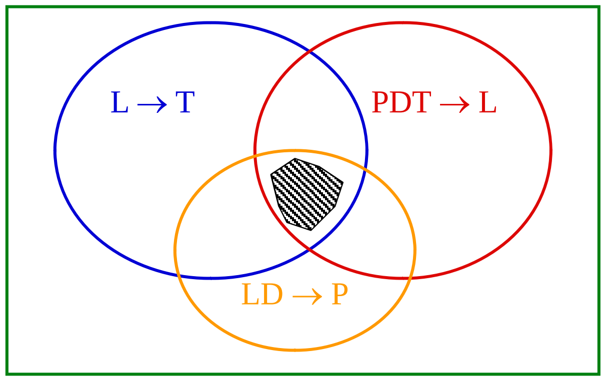
Rather, specify FDs that we expect all instances of to satisfy as part of the relation scheme



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 → Y it necessarily satisfies

Say that F *implies* X → Y (F ⊨ X → Y)

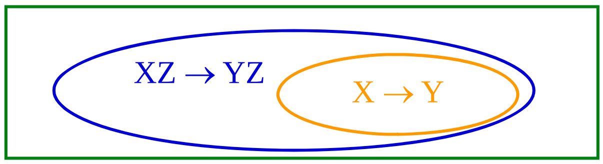
```
Consider L → T
Let r be a relation satisfying it
take t,s in r where t(PL) = s(PL)
so t(L) = s(L)
by L → T, t(T) = s(T)
thus r also satisfies PL → T
```

Inference Axioms

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

F1. Reflexivity: $X \rightarrow X$

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

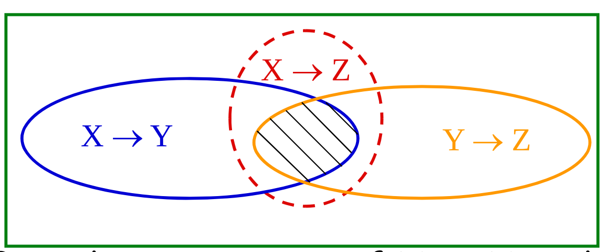


F3. Additivity: if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

More Axioms

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

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



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

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 → Y if there is a proof of X → Y from FDs in F using inference axioms F1. - F6.

$F \vdash X \rightarrow Y$

Is implication the same as derives?

soundness

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

completeness

Soundness

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

F5. If X → Y and Y → Z, then X → Z

Take 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

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

Instance to show not true

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

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

$bi \neq ci$

This table violates $X \to Y$.

- $t(X) = s(X)$, but claim $t(Y) \neq s(Y)$
- if $t(Y) = s(Y)$, then $Y \subseteq X^+$
- then since $X \to X^+$ in F^+, so is $X \to Y$ by projectivity, a contradiction

The instance satisfies F^+

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

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

- Then $t(W) \neq s(W)$, no problem

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

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

Principles of Database Systems

Case 2, continued

- We have proved $X \to X^+Z$ 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 $W \to Z$.

Armstrong Relation for F

An "exact model" for F:

- Satisfies F^+
- Violates every FD in F^-

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

Example R = ABC, F = {A → B, B → C}

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

Armstrong Relation

C → A, C → B, B → A, BC → 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 `assign` 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 Aug
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

assign = asgn1 $\bowtie$ asgn2

assign satisfies a *join dependency* (JD)

$\bowtie$[PLD, LT]

assign = π_{PLD}(assign) $\bowtie$ π_{LT}(assign)

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 ⋈[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 *multivalued dependency* is a special case where n = 2.

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.

Need to Use Tableaux

A "template" for a relation

Has *rows* of variables instead of tuples of values

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) 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 $\rho(m1) = \rho(m2)$.

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.

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**	a7

Apply AG → C

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

Testing FD Implication with the Chase

Warning! Method in book is wrong!

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

1. Set up a tableau 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

$F = \{AB \rightarrow C,\ B \rightarrow D,\ CD \rightarrow E,\ CE \rightarrow GH,\ G \rightarrow A\} \models BG \rightarrow C?$

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

You Try It

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

A	B	C	D	E	G	H
a1	*b1*	*c1*	*d1*	*e1*	*g1*	*h1*

Does BE → A?

Does BE → H? (can use same chase)