# Relational Design Theory I

Functional Dependencies

## Functional Dependencies: why?

Design methodologies:
- Bottom up (e.g. binary relational model)
- Top-down (e.g. ER leads to this)

Needed: tools for analysis of quality of relational schema

Goals:
- reducing redundancy (update/deletion anomalies)
- avoiding spurious tuples
- reducing null values

## Redundancy and Anomalies I:

**Insertion, Update Anomalies**

Example (Movie database)
MOVIE(title, actorname, year, length, country, role)

Consider tasks:
- update year;
- insert actor;
- insert movie
- delete actor

## Redundancy and Anomalies II

**Deletion Anomalies**

Example (student activities)

| Student | Activity | Fee |
|---|---|---|
| Marcus Brennigan | Piano | $20 |
| Deepa Patel | Swimming | $15 |
| Marcus Brennigan | Swimming | $15 |
| Abigail Winter | Tennis | $30 |
| Prakash Patel | Skiing | $150 |

delete student Abigail Winter.

## Null values I

Example (student activities)

| SID | Activity | Fee |
|---|---|---|
| 1001 | Piano | $20 |
| 1090 | Swimming | $15 |
| 1001 | Swimming | $15 |

insert new activity Chess with a fee of $20
(what would be a better design)

## Null values II

Example

section(SecID, teacherID, GraderID)

versus

section(SecID, teacherID)
grader(SecID, studentID)

## Spurious Tuples

Loss of information through additional, spurious tuples.

Example:

Split student activity into
(SID, Fee) and (Activity, Fee)
What is the problem?

What is the real solution?

## FUNCTIONAL DEPENDENCIES

## Functional Dependencies

Example (university)

SID determines LastName
CID determines CourseName
CID determines CourseNr
{StudentID, GroupID} determines Joined

## Functional Dependencies I

$R(A_1, A_2, \ldots, A_n)$
X and Y are subsets of $\{A_1, A_2, \ldots, A_n\}$

$X \rightarrow Y$ means that fixing the values of all attributes in X determines the values of all attributes in Y

X is called a **determinant** of Y

## Dependencies as constraints

- FDs are constraints we put on the relational schema
- We can see whether a relational state violates a FD (example)
- We cannot deduce FDs from a relational state.
- A relational state fulfilling a FD is a **model** of that FD.

## Keys and Superkeys

$R(A_1, A_2, \ldots, A_n)$
X subset of $\{A_1, A_2, \ldots, A_n\}$

X is **superkey** if $X \rightarrow \{A_1, A_2, \ldots, A_n\}$

A minimal superkey is a **key**, i.e. X is a key if
1) X is a superkey, and
2) no proper subset of X is a superkey.

Examples: University Relations
Lot(Lot#, County, PropertyID)

## Inference on FDs

{SID} → {LastName, FirstName, SID, SSN, Career, Program, City, Started}

We can conclude (among others) that

{SID} → {Career}
{SID} → {LastName}
{SID} → {SSN, City}

## Inference on FDs

{StudentID, CourseID} → {Quarter, Year}
{StudentID} → {SID, SSN, City}
{SSN} → {LastName, FirstName}
{Name} → {PresidentID, Founded}

Which of the following FDs can we infer from these rules?

{Name} → {Founded}
{StudentID, CourseID} → {Year, LastName}
{SSN} → {City}

## Trivial Dependencies

X → Y is **trivial**, if Y is contained in X.

R(A, B, C, D) with FDs

AB → C (or {A, B} → {C})
C → D
D → A

- What nontrivial FDs can we deduce?
- What are the keys of R?
- Are there superkeys which are not keys?

## Inference Example

R(A, B, C, D, E) with primary key {A,C,D}

And FDs
$AB \rightarrow CD$
$B \rightarrow E$
$D \rightarrow E$

- What nontrivial FDs can we deduce?
- What are the keys of R?

## Reasoning about FDs

A FD $X \rightarrow Y$ can be **inferred** from a set F of functional dependencies, if it holds true in every relational state that satisfies all FDs in F. In other words:

every model of F is a model of $X \rightarrow Y$

Example: from $\{A \rightarrow BC, C \rightarrow D\}$ we can infer $\{A \rightarrow D\}$

## Reasoning about FDs

Two sets F and G of FDs are **equivalent**, if all FDs in F can be inferred from G, and vice versa. In other words:
every model of F is a model of G and vice versa

Example:
$\{AB \rightarrow C, A \rightarrow B\}$ and $\{A \rightarrow B, A \rightarrow C\}$ are equivalent.

## Rules

**Reflexivity (triviality)**
**Augmentation**
**Transitivity**
Decomposition
Union
Pseudotransitive rule

$\Big\}$ Armstrong's inference rules (imply other rules)

(pg. 81)

## Closure

How do we determine whether we can infer a FD $X \rightarrow Y$ from a set of FDs $\mathcal{F}$?

$X^+$, the closure of X, is the set of all attributes determined by X under $\mathcal{F}$.

$X \rightarrow Y$ follows from $\mathcal{F}$
if and only if
Y is in $X^+$ (with regard to $\mathcal{F}$)

Example: R(A,B,C,D,E) with FDs {$AB \rightarrow DE$, $A \rightarrow E$
$C \rightarrow BD$, $D \rightarrow E$}, does $A \rightarrow D$? Does $AC \rightarrow D$?

## Computing the Closure

Given: set of FDs $\mathcal{F}$
set of attributes X
Goal: $X^+$, the set of all attributes determined by X

**Algorithm**:
$X^+ := X$
while $Y \rightarrow Z$ in $\mathcal{F}$ with
$Y \subseteq X^+$, and
Z not contained in $X^+$
$X^+ := X^+ \cup Z$

## Closure Examples

R(A,B,C,D,E,F,G,H)

$\mathcal{F}$ = {AE→B, BH→C, CDE→F, G→EH, GH→D}

Compute

$\{A\}^+$

$\{AG\}^+$

$\{B\}^+$

$\{AEH\}^+$

## Why closure?

X → Y follows from $\mathcal{F}$
if and only if
Y is in $X^+$ (with regard to $\mathcal{F}$)

X is superkey for $R(A_1, A_2, \ldots, A_n)$ with FDs $\mathcal{F}$
if and only if
$X^+ = \{A_1, A_2, \ldots, A_n\}$ (with regard to $\mathcal{F}$)

Allows us to test for

- (Candidate) key
- Inference
- Equivalence of systems of FDs

## Cover and Equivalence

$\mathcal{F}$, $\mathcal{G}$ : sets of FDs

$\mathcal{F}$ **covers** $\mathcal{G}$,
if all FDs in $\mathcal{G}$ can be inferred from $\mathcal{F}$.
in other words:
every model of $\mathcal{F}$ is a model of $\mathcal{G}$

$\mathcal{F}$ and G are **equivalent**,
if $\mathcal{F}$ covers $\mathcal{G}$ and $\mathcal{G}$ covers $\mathcal{F}$.

## Minimal Sets of Dependencies

$\mathcal{F}$, a set of FDs is **minimal,** if

1. The rhs of every dependency in $\mathcal{F}$ is a single attribute (a *singleton*).
2. No dependency X→A in $\mathcal{F}$ can be replaced by Y→A, where Y is a proper subset of X, such that the new system of dependencies is equivalent to $\mathcal{F}$.
3. No dependency can be removed from $\mathcal{F}$ such that the new system of dependencies is still equivalent to $\mathcal{F}$.

Canonical form with no redundancies.

## Minimal Cover

$\mathcal{G}$ is minimal cover of $\mathcal{F}$,
if $\mathcal{G}$ is minimal, and it covers $\mathcal{F}$.

**Algorithm:**

1. Use decomposition rule to split all rhs.
2. Sequentially try removing each attribute from each rule, and retain new rule if system is still equivalent.
3. If removing a dependency leaves the system equivalent to the old system, then remove it.

## Minimal Cover Algorithm

To test whether
$\mathcal{G}$ – {X→A} is equivalent to $\mathcal{G}$
we only need to test whether
X→A can be inferred from $\mathcal{G}$ - {X→A}.

To test whether
$\mathcal{G}$ – {X→A} ∪ {(X-{B})→A} is equivalent to $\mathcal{G}$
we only need to test whether
X-{B}→A can be inferred from $\mathcal{G}$.

## Minimal Cover Examples

R(A,B,C) with FDs
{AB→C, A→B}

R(A,B,C,D) with FDs
{A→BC, B→AC, D→ABC}

R(A,B,C,D,E,F,G) with FDs
{BCD→A, BC→E, A→F, F→G, C→D, A→G}

## Canonical Cover

$\mathcal{F}$, a set of FDs is **canonical,** if

1. No dependency X→Y in $\mathcal{F}$ can be replaced by X'→Y, where X' is a proper subset of X, such that the new system of dependencies is equivalent to $\mathcal{F}$.
2. No dependency X→Y in $\mathcal{F}$ can be replaced by X→Y', where Y' is a proper subset of Y, such that the new system of dependencies is equivalent to $\mathcal{F}$.
3. Every lhs of a dependency occurs at most once.

## Canonical Cover Algorithm

**Algorithm (Canonical Cover):**
1. Calculate Minimal Cover
2. Recombine rules with identical lhs

Example: If we have a minimal cover
$\mathcal{F}$ = {A→B, A→C, B→E, BC→D, BC→F, C→E},
then the canonical cover is
$\mathcal{G}$ = {A→BC, B→E, BC→DF, C→E}