# Relational Design Theory II

Normalization

## Detecting Anomalies

| SID | Activity | Fee | Tax |
|---|---|---|---|
| 1001 | Piano | $20 | $2.00 |
| 1090 | Swimming | $15 | $1.50 |
| 1001 | Swimming | $15 | $1.50 |

- Why is this bad design?
- Can we capture this using FDs?

## Normal Forms

- Requirements on relational schemas
- Initiated by Codd (1NF, 2NF, 3NF)

| | |
|---|---|
| 1NF (First NF) | no multivalued attributes |
| 2NF (Second NF) | no partial dependencies |
| 3NF (Third NF) | no bad transitive dependencies |
| BCNF (Boyce-Codd NF) | strengthening of 3NF |
| 4NF (Fourth NF) | extends BCNF to multivalued dependencies |

- there's more …

## BCNF

If $X \to Y$ is not trivial, then X has to be a superkey.

Has to be true for all valid FDs $X \to Y$

**Example:**

Activity(SID, Activity, Fee, Tax)

SID, Activity $\to$ Fee, Tax

Activity $\to$ Fee

Fee $\to$ Tax

How to decompose?

## BCNF Decomposition

What do we want?

- Relations are in BCNF
- We can reconstruct data in original relation
- Keep functional dependencies?

Note: Relations on two attributes are always BCNF.

## BCNF-Normalization

**Algorithm (BCNF Normalization)**

Input: Relation R, FDs $\mathcal{F}$

Output: BCNF-decomposition D of R

D := {R}

While $X \to Y$ holds in some $Q(A_1, \ldots, A_n)$ in D, and

$X \to Y$ not trivial, X not a superkey of Q

add $Q_1(X^+ \cap (\{A_1, \ldots, A_n\})$ and

$Q_2(X \cup (\{A_1, \ldots, A_n\} - X^+))$

remove Q.

## BCNF-Example

D := {R}

While $X \rightarrow Y$ holds in some $Q(A_1, \ldots, A_n)$ in D, and $X \rightarrow Y$ not trivial, X not a superkey of Q

add $Q_1(X^+ \cap (\{A_1, \ldots, A_n\})$ and

$Q_2(X \cup (\{A_1, \ldots, A_n\} - X^+))$

remove Q.

**Examples**:

R(A, B, C, D), FDs: $A \rightarrow B$, $C \rightarrow D$

R(A, B, C, D), FDs: $AC \rightarrow B$, $C \rightarrow D$

R(A,B,C,D,E), FDs: $A \rightarrow BE$, $E \rightarrow D$

## BCNF-Normalization Caveat

Checking whether $X \rightarrow Y$ holds in some Q in D refers to $\mathcal{F}$, not just D.

Example:

R(A,B,C,D,E)

FDs: $A \rightarrow B$

$BC \rightarrow D$ (implies $AC \rightarrow D$)

- naïve implementation of algorithm requires exponential time
- can be improved to polynomial time

(Tsou, Fischer, *Decomposition of a relation scheme into Boyce-Codd Normal Form*, ACM, SIGACT News,1982)

## Testing for BCNF

- single R with FDs $\mathcal{F}$

  testing for BCNF can be done in polynomial time, it is sufficient to test dependencies in $\mathcal{F}$
- this is not true for decompositions, e.g.

R(A,B,C,D,E)

FDs: $A \rightarrow B$

$BC \rightarrow D$

decompose into

$R_1$(A,B)

$R_2$(A,C,D,E)

## More BCNF-Examples

phone(Name, City, AreaCode, PhoneNumber, Extension)
R(A,B,C,D), keys: AB, CD, FDs: $A \to C$, $D \to B$
R(A,B,C,D,E), FDs: $A \to B$, $C \to D$, $AC \to E$

## Lossless Join Property

$D = \{R_1, R_2, \ldots, R_n\}$ decomposition of R.

If $R_1 * R_2 * \ldots * R_n = R$, then
D has the **lossless join property**.

BCNF decomposition has lossless join property.

## Lossless Join Property

**Test** lossless join property for binary decomposition
Given: R, $D = \{R_1, R_2\}$, FDs $\mathcal{F}$
D is a lossless join decomposition of R, if and only if
$R = R_1 \cup R_2$, and either
$R_1 \cap R_2 \to R_1 - R_2$ holds in $\mathcal{F}$ or
$R_1 \cap R_2 \to R_2 - R_1$ holds in $\mathcal{F}$ .

**Example**: Activity(SID, Activity, Fee)

- necessary?
- sufficient?
- implies correctness of BCNF algorithm

## Lossless Join Property

**Algorithm (Chase Test)**

Input: relation $R(A_1, \ldots, A_n)$, FDs $\mathcal{F}$

decomposition $D = \{R_1, R_2, \ldots, R_m\}$

Output: Is D a lossless join decomposition of R?

T := table with columns $A_1, \ldots, A_n$, rows $R_1, R_2, \ldots, R_m$

$$T[i,j] := \begin{cases} a_{i,j} & \text{if } A_i \text{ not in } R_j \\ a_i & \text{if } A_i \text{ in } R_j \end{cases}$$

instead of $A_i$, $a_{i,j}$, $a_i$

use: A, $a_j$, a

B, $b_j$, b, …

Apply FDs in $\mathcal{F}$ to identify elements until

- there is a row $(a_1, \ldots, a_n)$: lossless join
- no more changes are possible: not lossless join

## Chase Test Examples

R(A,B,C), FDs: $A \rightarrow B$,

$D_1$ = {P(A,B), Q(A,C)}

$D_2$ = {P(B,C), Q(A,C)}

R(A,B,C,D), FDs: $A \rightarrow B$, $C \rightarrow D$,

$D$ = {P(A,B), Q(B,C),T(C,D)}

R(A,B,C,D,E,F,G), FDs: $A \rightarrow G$, $B \rightarrow A$, $BCE \rightarrow ADF$, $C \rightarrow EF$, $F \rightarrow CD$, $G \rightarrow BF$,

$D$= {P(A,F,G), Q(B,E,F), S(C,D,G)}

## Dependency Preservation

banker(BranchName, CustomerName, BankerName)

BankerName → BranchName

BranchName, CustomerName → BankerName

R(A,B,C), FDs: $A \rightarrow B$, $BC \rightarrow A$

- why not in BCNF? (Keys?)
- what are possible BCNF decompositions?
- what happens to dependencies?

Deciding whether a given relation has a dependency preserving BCNF decomposition is NP-complete

Tsou, Fischer, *Decomposition of a relation scheme into Boyce-Codd Normal Form*, ACM, SIGACT News,1982

## Prime Attributes

**prime attribute**: part of some key

**Examples:**

R(A,B,C,D,E), AB is key, C is key, B→D, D→E
A,B,C are prime,
D,E are nonprime

R(A,B,C,D,E)
AC→D, BD→E, E→AC

## Prime Attributes

**prime attribute**: part of some key

- How many keys can there be on n attributes?
- How hard is it to find all keys? Algorithm?

## Prime Attributes

**prime attribute**: part of some key

- How many keys can there be on n attributes?
- How hard is it to find all keys? Algorithm?

Determining primality of an attribute is NP-complete.

(Lucchesi, Osborne, *Candidate keys for relations*, J. Comput. System Sci. 17, 1978

## 3NF

If X→Y is not trivial, then X has to be a superkey, *or*, all attributes in Y-X are prime.

Has to be true for all valid FDs X→Y

Violated in

Book(Author, Title, PriceCategory, Price)
Movie(Title, Year, MPAA, MinimumAge)

## 3NF-Examples

R(A,B,C,D) with key A and
B→CD, C→D, D→C

Can we find a decomposition of these relations that contains the same information?

## 3NF-Normalization

**Algorithm (3NF Normalization):**

Input: Relation R with FDs $\mathcal{F}$

Output: 3NF decomposition D of R

1. Compute canonical cover C of $\mathcal{F}$
2. D = {}
3. For every X→Y in C add Q(XY) to D, unless
   a) some S in D already contains all of XY: don't add Q
   b) some S in D is contained in XY: replace S with Q(XY)
4. If no relation in D contains a key of R, then add new relation Q(X) on some key X of R

## 3NF-Examples

- R(A,B,C,D) with A key, B→CD, C→D, D→C
- R(A,B,C,D) with AB key, A→C, B→D
- R(A,B,C,D,E) with AB and AC keys, BC→D, C→E
- R(A,B,C,D,E) with AB key, A→E, BC→D, D→E
- R(A,B,C,D,E,F) with ABC key, A→E, AC→F, EF→G
- R(A,B,C,D,E,F) with A and BC keys, B→D, D→F
- R(A,B,C,D,E) with AB and CD keys, A→E, C→E

Find 3NF normalization

Results can depend on canonical cover, and order of execution

## 3NF Algorithm

- 3NF Normalization Algorithm is loss-less join (chase test)
- It is dependency preserving (obviously)
- The resulting relations are in 3NF (not trivial).

## 3NF vs BCNF: properties

- BCNF is stronger than 3NF
- BCNF and 3NF are loss-less join (no spurious tuples)
- 3NF preserves dependencies
- BCNF does not always preserve dependencies

# 3NF vs BCNF: algorithmics

- Normalization Algorithms:
  - naïve algorithm for 3NF in polynomial time
  - naïve algorithm for BCNF in exponential time, but can be done in polynomial time
- Recognition Algorithms:
  - BCNF is easy to recognize (polynomial time)
  - Recognizing 3NF is NP-complete

    (Jou, Fischer, *The complexity of recognizing 3NF relation schemes*, Information Processing Letters 14, 1982)