Database Design: Third Normal Form

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Learning Objectives

After this lecture, you should be able to:

- Decompose a database schema into a set of relations obeying 3NF

Normal Forms

First Normal Form = all attributes are atomic

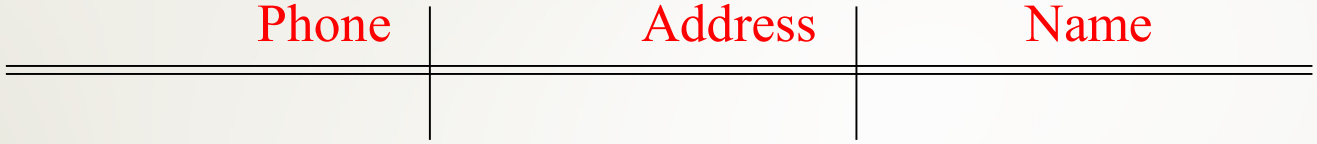
Second Normal Form (2NF) = old and obsolete

Boyce Codd Normal Form (BCNF)

Third Normal Form (3NF)

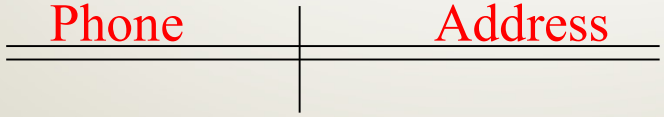
Others...

3NF: A Problem with BCNF



FD's: Phone → Address; Address, Name → Phone

So, there is a BCNF violation (Phone ➔ Address), and we decompose.



Phone -> Address

No FDs

So where's the problem?

Phone	Address
1234	10 Downing
5678	10 Downing

Phone	Name
1234	John
5678	John

FD's: Phone → Address; Address, Name → Phone

No problem so far. All *local* FD's are satisfied.

Let's put all the data into a single table:

Phone	Address	Name
1234	10 Downing	John
5678	10 Downing	John

Violates the dependency: Address, Name → Phone

Preserving FDs

- Thus, if the X and Y of a FD X->Y do not both end up in the same decomposed relation:
  - Such a decomposition is not "dependency-preserving."
  - No way to force BCNF to preserve dependencies
- Thus, while BCNF gives us **lossless join** and **less redundancy**, it doesn't give us **dependency preservation**

An alternative: 3rd Normal Form (3NF)

Definition. A relation R is in 3rd normal form if :

Whenever there is a nontrivial dependency $A_1, A_2, \ldots, A_n \rightarrow B$ for R, then $\{A_1, A_2, \ldots, A_n\}$ is a super-key for R, OR B is part of a key.

Prevents the "Phone ➔ Address" FD from causing a decomposition

Textbook uses rule with many B_i on the RHS, if so, then each one must be part of some key.

3NF vs. BCNF

- R is in BCNF if whenever X→ A holds, then X is a superkey.
  - Slightly stricter than 3NF.
    - Doesn't let R get away with it if A is part of some key
  - Thus, BCNF "more aggressive" in splitting

- Example: R(A,B,C) with AB → C; C → A
  - 3NF but not BCNF

Decomposing R into 3NF
Preliminaries: Minimal basis

Given a set of FDs: F.

Say the set F' is *equivalent* to F, in the sense that F' can be inferred from F and v. versa.

- Any such F' is said to be a *basis* for F.

- "Minimal basis"
  - A basis with all RHS singletons, where any modifications lead to no longer a basis, including:
    - Dropping attribute from LHS of a rule: compact rules
    - Dropping a rule: small # of rules

Example of minimal basis

- R(A, B, C) with FDs:
  - A → BC; B → AC; C → AB
- A basis:
  - A→ B; A → C; B → A; B → C; C → A; C → B
- One minimal basis:
  - A → B
  - B → C
  - C → A

Conversion into minimal basis

- "Algorithm for converting F to a minimal basis
  - R = F with all RHS singletons:
  - Repeat until convergence:
    - If a rule minus an attribute from LHS is inferred from F, replace rule with rule minus attribute from LHS
    - If a rule is inferred from rest, drop it

Minimal basis example

Given R (A B C D E) and

F = { **A->D, BC->AD, C->B, E->A, E->D**}

Find F', the minimal basis for F.

Algorithm:

1- Only singleton in RHS

2- Remove unnecessary att. from LHS

3- Remove FDs that can be inferred from the rest

① $BC \rightarrow A$; $BC \rightarrow D$

② $BC \rightarrow A \Rightarrow C \rightarrow A$

$BC \rightarrow D \Rightarrow C \rightarrow D$

$E^+ = \{E, A, D\}$

$E \rightarrow D$ can be inferred

③ $F = \{A \rightarrow D, C \rightarrow A, \cancel{C \rightarrow D}, C \rightarrow B, E \rightarrow A, \cancel{E \rightarrow D}\}$

$F' = \{A \rightarrow D, C \rightarrow A, C \rightarrow B, E \rightarrow A\}$

Decomposing R into 3NF

1. Get a "minimal basis" *G* of given FDs
2. For each FD A ➔ B in the minimal basis *G*,
   use AB as the schema of a new relation.
3. If none of the schemas from Step 2 is a superkey,
   add another relation whose schema is a key for the original relation.

Result will be lossless, will be dependency-preserving, 3NF; might not be BCNF

Decomposing R into 3NF

1. Get a "minimal basis" G of given FDs
2. For each FD A ➔ B in the minimal basis G,
   use AB as the schema of a new relation.
3. If none of the schemas from Step 2 is a superkey,
   add another relation whose schema is a key for the original relation.

Implicitly this is connecting all the LHSs with the remaining attributes

Result will be lossless, will be dependency-preserving,

Basically every minimal FD is preserved somewhere

Example

- R(A, B, C) with FDs:
  - A → BC; B → AC; C → AB

Minimal Basis: A → B; B → C; C → A

So, first cut:

R1(A, B), R2(B, C), R3(C, A)

Any attributes left? Nope ➔ done

Example

- R(A, B, C, D, E) with FDs:
  - A → B; CD → B; DA → C

  BCNF Decomp:

  (AB), (ACD), (ADE) or:

  (BCD), (ACD), (ADE)

  Which FDs do each of these not preserve?

Minimal Basis:

A → B; CD → B; DA → C

3NF Decomp: (AB), (BCD), (ACD), (**ADE**)

Desirable Properties of Schema Refinement

1) minimize redundancy

✓ **2) avoid info loss**

✓ **3) preserve dependency**

3NF

4) ensure good query performance

Caveat

- Normalization is not the be-all and end-all of DB design
- Example: suppose attributes A and B are always used together, but normalization theory says they should be in different tables.
  - decomposition might produce unacceptable performance loss (extra disk reads)

Overview of Database Design

- Conceptual design: (ER & UML Models are used for this.)
  - What are the **entities** and **relationships** we need?

- Logical design:
  - Transform ER design to Relational Schema

- Schema Refinement: (Normalization)
  - Check relational schema for redundancies and related anomalies.
- Physical Database Design and Tuning:
  - Consider typical workloads; (sometimes) modify the database design; select file types and indexes.

We'll discuss indexing next.