CptS 451- Introduction to Database Systems

Relational Design Theory

(DMS Ch-19)

Instructor: Sakire Arslan Ay

WASHINGTON STATE UNIVERSITY

World Class. Face to Face.

Some of the material in this file has been adopted from Jennifer Widom's lecture notes

WSU CptS 451

Relational Design Theory (topics)

- Motivation & Overview
- Functional Dependencies
- Boyce-Codd Normal Form
- 3NF

Database Design: A 6-Step Program

1. Requirements Analysis: data requirements, critical operations on the data
2. Conceptual DB Design: high-level description of data and constraints - typically using ER model
3. Logical DB Design: conversion into a schema
   - pick a type of DBMS, relational DBMS is most popular and is our focus
4. Schema Refinement: normalization (eliminating redundancy)
5. Physical DB Design: consider workloads, indexes and clustering of data
6. Application/Security Design

Designing/Refining a Database Schema

WSU CptS 451

- Usually many designs possible
  - Often use higher-level design tools.
  - Some designers go straight to relations
    - Useful to understand why tools produce certain schemas
- Some are (much) better than others!
- How do we choose?
  - Theory for relational database design

Designing a Database Schema

Example1: College application info.

- SSN and name
- Colleges applying to
- High schools attended (with city)
- Hobbies

Apply(SSN, sName, cName, HS, HScity, hobby)

Designing a Database Schema

WSU CptS 451

Example1: College application info.

Apply(SSN, sName, cName, HS, HScity, hobby)

Kyle (with SSN 111) from PHS (Pullman) swims and plays trumpet and he applied to WSU, UW, and OSU.

SSN	sName	cName	HS	HScity	hobby
1111	Kyle	WSU	PHS	Pullman	Swim
1111	Kyle	WSU	PHS	Pullman	Trumpet
1111	Kyle	UW	PHS	Pullman	Swim
1111	Kyle	UW	PHS	Pullman	Trumpet
1111	Kyle	OSU	PHS	Pullman	Swim
1111	Kyle	OSU	PHS	Pullman	Trumpet

Designing a Database Schema

WSU CptS 451

Example1: College application info.

Apply(SSN, sName, cName, HS, HScity, hobby)

Design "anomalies"

1) Redundancy

- Captures information multiple times
  - For example:
    - SSN, sName, HS, Hscity are repeated per (cName, hobby) pair.

SSN	sName	cName	HS	HScity	hobby
1111	Kyle	WSU	PHS	Pullman	Swim
1111	Kyle	WSU	PHS	Pullman	Trumpet
1111	Kyle	UW	PHS	Pullman	Swim
1111	Kyle	UW	PHS	Pullman	Trumpet
1111	Kyle	OSU	PHS	Pullman	Swim
1111	Kyle	OSU	PHS	Pullman	Trumpet

- There is **functional dependency** between SSN and sName, HS and HScity.

Designing a Database Schema

Example1: College application info.

Apply(SSN, sName, cName, HS, HScity, hobby)

Design "anomalies"

1) Redundancy

2) Update anomaly

– if we decide to call the instrument cornet (instead of trumpet) we need to modify it in each of the tuples in which it is stored (one per cName). Else, database will be inconsistent.

SSN	sName	cName	HS	HScity	hobby
1111	Kyle	WSU	PHS	Pullman	Swim
1111	Kyle	WSU	PHS	Pullman	~~Trumpet~~ Cornet
1111	Kyle	UW	PHS	Pullman	Swim
1111	Kyle	UW	PHS	Pullman	Trumpet
1111	Kyle	OSU	PHS	Pullman	Swim
1111	Kyle	OSU	PHS	Pullman	Trumpet

Designing a Database Schema

WSU CptS 451

Example1: College application info.

Apply(SSN, sName, cName, HS, HScity, hobby)

Design "anomalies"

1) Redundancy
2) Update anomaly
3) Deletion anomaly

– How to delete "Trumpet" hoby without deleting applicant information
  – possible solution: use **null** values in the hobby field

SSN	sName	cName	HS	HScity	hobby
1111	Kyle	WSU	PHS	Pullman	Swim
1111	Kyle	WSU	PHS	Pullman	~~Trumpet~~ NULL
1111	Kyle	UW	PHS	Pullman	Swim
1111	Kyle	UW	PHS	Pullman	~~Trumpet~~ NULL
1111	Kyle	OSU	PHS	Pullman	~~Trumpet~~ NULL

Designing a Database Schema

WSU CptS 451

Example: College application info.

- SSN and name
- Colleges applying to
- High schools attended (with city)
- Hobbies

```
Student(SSN, sName)
Apply(SSN, cName)
HighSchool(SSN, HS)
Located(HS, HScity)
Hobbies(SSN, hobby)
```

- Decompose the relation into multiple relations
  - No anomalies
  - Can reconstruct the original relations (no loss of information)

Designing a Database Schema

Example: College application info.

- SSN and name
- Colleges applying to
- High schools attended (with city)
- Hobbies

```
Student(SSN, sName)
Apply(SSN, cName)
HighSchool(SSN, HS)
Located(HS, HScity)
Hobbies(SSN, hobby)

HobbyList(hobby, desc)
CollegeList(cName)
```

- The best design, for an application for relational databases depend not only on constructing the relations well, but also in what the data is representing in the real world.

Redundancy and Anomalies in Relational Schema – Example2

CptS451 Projects	Student	Proj title	Date	Room#
	Kyle S.	Yelp	04/28/18	EME102A
	Aaron B.	Yelp	04/28/18	EME102A
	Jeromy J.	Yelp	04/28/18	EME102A
	Kelly K.	OODB	04/30/18	ETRL101

Redundancy:

— date of presentation and room# are repeated per member of project group

Redundancy and Anomalies in Relational Schema – Example2 (cont.)

WSU CptS 451

CptS451 Projects	Student	Proj title	Date	Room#
	Kyle S.	Yelp	$\boxed{04/30/14}$	EME102A
	Aaron B.	Yelp	04/28/14	EME102A
	Jeromy J.	Yelp	04/28/14	EME102A
	Kelly K.	OODB	04/30/14	ETRL101

Error in updating. Forgot to update all entries.

Update Anomaly:

– if we modify presentation date for the "yelp" project, we need to modify the date in each of the tuples in which it is stored (one per member). Else, database will be inconsistent.

Redundancy and Anomalies in Relational Schema – Example2 (cont.)

WSU CptS 451

CptS451 Projects	Student	Proj title	Date	Room#
	Kyle S.	Yelp	04/28/18	EME102A
	Aaron B.	Yelp	04/28/18	EME102A
	Jeromy J.	Yelp	04/28/18	EME102A
	~~Kelly K.~~ NULL	OODB	04/30/18	ETRL101

Deletion Anomaly:

- How to delete the fact that Kelly K. dropped out of the project without deleting information about the OODB project.
  - possible solution: use **null** values in the student field

Relation Decomposition

CptS451 Projects	Student	Proj title	Date	Room#
	Kyle S.	Yelp	04/28/14	EME102A
	Aaron B.	Yelp	04/28/14	EME102A
	Jeromy J.	Yelp	04/28/14	EME102A
	Kelly K.	OODB	04/30/14	ETRL101
	NULL	dDB	04/24/14	SLOAN123

Student	Proj title
Kyle S.	Yelp
Aaron B.	Yelp
Jeromy J.	Yelp
Kelly K.	OODB

Proj title	Date	Room#
Yelp	04/28/14	EME102A
OODB	04/30/14	ETRL101
dDB	04/24/14	SLOAN123

Anomalies have gone:

- No more repeated data
- Easy to update project information

Designing a Database Schema

Design by decomposition – how does it work?

- Start with "mega" relations
- Decompose into smaller, better relations with same info.
- Can do decomposition automatically

Automatic decomposition

- "Mega" relations + *properties of the data (functional dependencies)*
- System decomposes based on properties
- Final set of relations satisfy *normal form*
  - *No anomalies, no lost information*

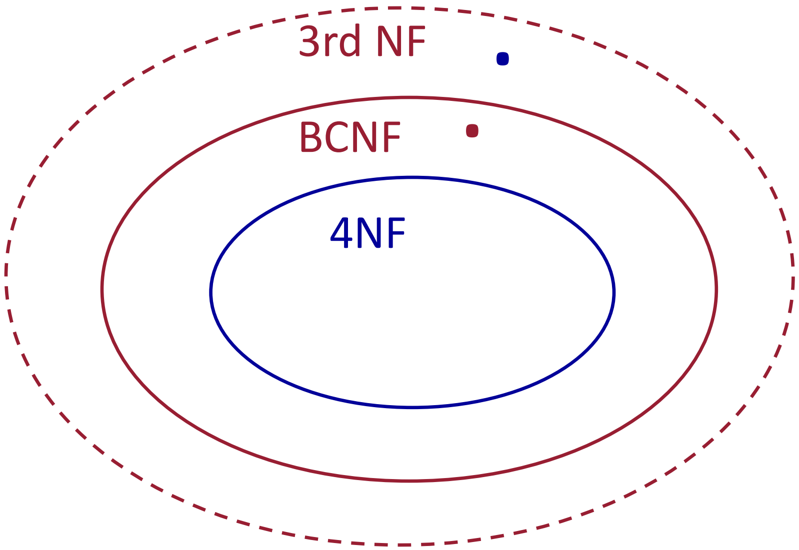
Designing a Database Schema

Normal Forms

Functional dependencies ⇒ Boyce-Codd Normal Form

+ Multivalued dependencies ⇒ Fourth Normal Form

1st NF
2nd NF
3rd NF



In CptS451, we will only cover BCNF and 3NF

What we will cover: Functional Dependencies and BCNF

WSU CptS 451

`Apply(SSN, SName, cName)`

- Redundancy: Storing **SSN-sName** pair once for each college

SSN	sName	cName
1111	Kyle	WSU
1111	Kyle	UW
1111	Kyle	OSU

Functional Dependency **SSN → sName**

- Same **SSN** always has same **sName**
- Should store each **SSN**'s **sName** only once

Boyce-Codd Normal Form If **A → B** then **A** is a key

`Apply(SSN, sName, cName) :` **SSN** is not a key
Apply is not in BCNF

Decompose: `Student(SSN, sName)` `Apply(SSN, cName)`

SSN is the key **SSN, CName is the key**

Functional Dependency

Functional Dependency - Example

WSU CptS 451

Example:

`Hourly_Emps(SSN,name,dept,rating,hourly_pay, num_hours)`

Assume there is fixed hourly pay rate for each rating.

ssn	name	dept	rating	hourly_pay	num_hours
111-11-1111	Kelly	123	5	18	40
222-11-2222	Kyle	124	4	16	40
333-11-3333	John	124	4	16	20
444-11-4444	Roseanne	123	4	16	20
555-11-5555	Ning	123	5	18	40

Problems????

Key : ssn

FDs: ssn $\rightarrow$ name,dept,rating,num_hours
rating $\rightarrow$ hourly_pay

Functional Dependency - Example

WSU CptS 451

`Hourly_Emps(SSN,name,dept,rating,hourly_pay, num_hours)`

- Suppose **`hourly_pay`** is determined by **`rating`**

  rating = 4 → hourly_pay =16
  rating = 5 → hourly_pay =18
  rating = 6 → hourly_pay =20
  … …

- Two tuples with same **`rating`** have same **`hourly_pay`**

  `rating`→ **`hourly_pay`** (rating functionally determines hourly pay)

Functional Dependency - Definition

For all tuples t, u in R,

if $t[A] = u[A] \Rightarrow t[B]=u[B]$

then $A \rightarrow B$

Definition: Given Relation R(A1,...,An,B1,...,Bm,C1,...,Cl)

A1,...,An ***functionally determine*** B1,...,Bm, i.e.,

$\boxed{(A1,...,An \rightarrow B1,...,Bm)}$

When any two tuples agree on the attributes $\boxed{A1,A2,...,An}$

Then they must also agree on the attributes $\boxed{B1,B2,...,Bn}$

Functional Dependency (FD) – Some Terminology

WSU CptS 451

- FDs are based on knowledge of real world. They generalize the concept of a key.

- If we know that an FD holds on all tuples, then we say that R satisfies the FD

- If we say that R satisfies an FD "F", we are stating a **constraint** on R

Functional Dependency (FD) - Some Terminology

WSU CptS 451

Let X and Y be set of attributes from relation R=(A,B,C,…)

Trivial FD

- Those that are true for every relation
- $X \rightarrow Y$ is trivial if Y is a subset of the X, i.e., $Y \subseteq X$
- Example: $AB \rightarrow A$

Nontrivial FD

- $X \rightarrow Y$ is called nontrivial if at least one of the attributes in Y is not among the attributes in X , i.e., $Y \not\subseteq X$
- Examples: $AB \rightarrow AC$

Completely nontrivial FD

- Called *completely nontrivial* if none of the attributes in Y is one of the attributes in X, i.e., $Y \cap X = \emptyset$
- Example: $AB \rightarrow C$

FD – Observation

WSU CptS 451

- If both of these FDs are true:

ssn → rating
rating → hourly_pay

- Then this FD also holds:

ssn → hourly_pay

If we find out from application domain (real world data) that a relation satisfies some FDs, it doesn't mean that we found all the FDs that it satisfies!
There could be more FDs implied by the ones we have.

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  ssn → rating and rating → hourly_pay implies ssn → hourly_pay

- An FD ***f*** is <u>*implied by*</u> a set of FDs ***F,*** if ***f*** holds whenever all FDs in ***F*** hold.
  - ***F⁺*** = *closure of F* is the set of all FDs that are implied by *F*.

- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - <u>*Reflexivity*</u>: **If $X \subseteq Y$, then $Y \rightarrow X$**
  - <u>*Augmentation*</u>: **If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z**
  - <u>*Transitivity*</u>: **If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$**

- These are ***sound*** and ***complete*** inference rules for FDs!

Additional Rules for FDs

Additional rules while reasoning about F^+

1. Combining Rule (Union)

Combining Right sides of FDs

$$\left.\begin{array}{l} A1,\ldots An \rightarrow B1 \\ A1,\ldots An \rightarrow B2 \\ \ldots. \\ A1,\ldots An \rightarrow Bm \end{array}\right\} \text{ is equivalent to } A1,\ldots An \rightarrow B1,\ldots Bm$$

Example: $A \rightarrow B$ and $A \rightarrow C$ is equivalent to $A \rightarrow BC$.

Additional Rules for FDs

2. Splitting Rule (Decomposition)

Splitting **right** sides of FDs

$A1,\ldots An \to B1,\ldots Bm$ is equivalent to $A1,\ldots An \to B1$

$A1,\ldots An \to B2$

....

$A1,\ldots An \to Bm$

Example: $A \to BC$ is equivalent to $A \to B$ and $A \to C$.

Can we also split **left**-hand-side?

- **No**. There is no splitting rule for left sides

Example?

What we talked about so far…

WSU CptS 451

1. Redundancy and Anomalies in Relational Schema
2. Design by decomposition
   - Start with “mega” relations containing everything
   - Decompose into smaller, better relations with same info.
   - Can do decomposition automatically
   - Final set of relations satisfies *normal form*
     - *No anomalies, no lost information*
     - *BCNF (will cover)*
     - *1NF, 2NF, 3NF, 4NF (won’t cover)*
3. Functional dependencies (FDs)
4. Rules for Functional Dependencies
   - Armstrong Axioms,
   - Closure of FDs
5. Closure of Attributes (next)

Closure of Attributes

- Given relation R, FDs ***F***, set of attributes X={A1,A2,…An}
- Find all Y={B1,..,Bm} such that $X \rightarrow Y$

Closure of X denoted by X^{+} ={A1,A2,…An}$^{+}$

How to calculate {A1,A2,…An}$^{+}$?

Start with X={A1,A2,…An}

Repeat until no change:

For every FD rule in ***F*** $X' \rightarrow Y'$,

if X' is in the closure set

add Y' to the set

Hourly_Emps(SSN,name,dept,
rating,hourly_pay,
num_hours)

ssn $\rightarrow$ name,dept,rating,num_hours

rating $\rightarrow$ hourly_pay

Closure Example

Hourly_Emps(SSN, name, dept, rating, hourly_pay, num_hours)

ssn $\rightarrow$ name,dept,rating,num_hours
rating $\rightarrow$ hourly_pay

$\{ssn\}^+$ = {ssn, name, dept, rating, num_hours, hourly_pay}

Hence:

ssn $\rightarrow$ ssn, name, dept, rating, num_hours, hourly_pay

Key : ssn

Armstrong's Axioms

Reflexivity: **If $X \subseteq Y$, then $Y \rightarrow X$**
Augmentation: **If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z**
Transitivity: **If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$**

Closure Example

WSU CptS 451

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

$A, B \rightarrow C$
$A, D \rightarrow E$
$B \rightarrow D$
$A, F \rightarrow B$

- Compute $\{A,B\}^+$ $\{A,B\}^+ = \{A, B, \quad\quad\quad\quad \}$
- Compute $\{A,F\}^+$ $\{A,F\}^+ = \{A, F, \quad\quad\quad\quad \}$

Closure Example

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

$A, B \rightarrow C$
$A, D \rightarrow E$
$B \rightarrow D$
$A, F \rightarrow B$

- Compute $\{A,B\}^+$ $\{A,B\}^+ = \{A, B, C, D, E\}$
- Compute $\{A,F\}^+$ $\{A,F\}^+ = \{A, F, B, C, D, E\}$

What is the key of R?

Closure Example 2

WSU CptS 451

Practice yourselves:

R(A,B,C,D)

$A, B \rightarrow C$
$A, D \rightarrow B$
$B \rightarrow D$

- Compute $\{A,B\}^+$ $\{A,B\}^+ = \{A, B, \quad \}$
- Compute $\{A,D\}^+$ $\{A,D\}^+ = \{A, D, \quad \}$
- Compute $\{B,D\}^+$ $\{B,D\}^+ = \{B, D, \quad \}$

What is the key of R?

Functional Dependencies and Keys

WSU CptS 451

- Assume R=(A1,..An,B1,..Bm,C1,..Ck) is a relation with no duplicates
- Suppose **A1,A2,…An** $\rightarrow$ all attributes
  - i.e., A1,A2,…An $\rightarrow$ A1,…An,B1,…Bm,C1,…,Ck
  - The group of attributes **A1,…,An is a key** that functionally determine the complete tuple.

R(A,B,C,D)

$\{A,B\}^+ = \{A, B, C, D\}$

A, B $\rightarrow$ C
A, D $\rightarrow$ B
B $\rightarrow$ D

Keys

- A **superkey is a set of attributes A1, ..., An such that** they functionally determine the complete tuple.
  - **Ex:** R(A,B,C,D,E,F) with FDs {AB→C; AD→E; B → DA; F →B}

    {A,C,F} is a superkey for R

- A **key is a minimal key**
  - A superkey and for which no subset is a key
  - **Ex:** {A,F} is a minimal key for R

Computing (Super)Keys

- Given R=(A1,..,An,B1,..Bm,C1,..Ck), let **X** be a subset of the attributes for R, i.e.,
  $\mathbf{X} \subseteq \{A1,..,An,B1,..Bm,C1,..Ck\}$
- For all subsets **X**, compute $\mathbf{X}^+$
- If **$\mathbf{X}^+$ is equal to [all attributes]**, then **X** is a **superkey**
- Minimal **X**(s) is/are the **key**(s)
  - Can we have more than one key ?
    - YES

Computing Keys – Example1

Product(name, price, category, color)

FDs:

name, category → price
category → color

Closures:

$\{name\}^+ = \{name\}$
$\{price\}^+ = \{price\}$
$\{category\}^+ = \{category,color\}$
$\{color\}^+ = \{color\}$

- What is the key?

$\{name,price\}^+ = \{name,price\}$

Key $\boxed{\{name,category\}^+ =\{name,category,price,color\}}$

$\{name,color\}^+ = \{name,color\}$
$\{price,category\}^+ =\{price,category,color\}$
$\{price,color\}^+ = \{price,color\}$
$\{category,color\}^+ =\{category,color\}$

Superkey $\boxed{\{name,price,category\}^+ = \{name,price,category,color\}}$

$\{name,price,color\}^+ =\{name,price,color\}$

Superkey $\boxed{\{name,category,color\}^+ = \{name,price,category,color\}}$

$\{price,category,color\}^+ = \{price,category,color\}$

Superkey $\boxed{\{name,price,category,color\}^+ =\{name,price,category,color\}}$

Functional Dependency

- Back to Example:

```
Student(SSN, sName, address,
         HScode, HSname, HScity, GPA, priority)
```

- **Functional dependencies:**

  SSN → sName
  SSN → address
  SSN → HScode
  HScode → HSname, HScity
  HSname, HScity → HScode
  SSN → GPA
  GPA → priority

We assume that each student has one (current) address, and has graduated from one high schools.

From the last2 above we can derive SSN → priority

Computing Keys – Example3

WSU CptS 451

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

- **FDs:**

  SSN → sName, address, HScode, GPA

  HScode → HSname, HScity

  HSname, HScity → HScode

  GPA → priority

- What is the key?

{SSN}$^+$ = {SSN,sName,address,HSname,Hscity,GPA,priority}
{sName}$^+$ = {sName}
{HScode}$^+$ = {HScode,HSname,HScity}
{HSname}$^+$ = {HSname}
{HScity}$^+$ = {HScity}
{GPA}$^+$ = {GPA,priority}
{priority}$^+$ = {priority}

{SSN,sName}$^+$ ={SSN,sName,address,GPA,priority}
{SSN,address}$^+$ = {SSN,sName,address, ,Hscity,GPA,priority}
{SSN,HScode}$^+$ = {SSN,sName,address,HScode,HSname,Hscity,GPA,priority}
{SSN,HSname}$^+$ = {SSN,sName,address,HSname,GPA,priority}
{SSN,HScity}$^+$ = {SSN,sName,address,HScity,GPA,priority}
{SSN,GPA}$^+$ = {SSN,sName,address,GPA,priority}
{SSN,priority}$^+$ = {SSN,sName,address,GPA,priority}

Computing Keys – Example3

WSU CptS 451

$\{sName, address\}^+ = \{sName, address\}$
$\{sName, HScode\}^+ = \{sName, Hscode, HSname, Hscity\}$
$\{sName, HSname\}^+ = \{sName, HSname\}$
$\{sName, HScity\}^+ = \{sName, HScity\}$
$\{sName, GPA\}^+ = \{sName, GPA, priority\}$
$\{sName, priority\}^+ = \{sName, priority\}$

$\{address, HScode\}^+ = \{address, HScode, HSname, Hscity\}$
$\{address, HSname\}^+ = \{address, HSname\}$
$\{address, HScity\}^+ = \{address, HScity\}$
$\{address, GPA\}^+ = \{address, GPA, priority\}$
$\{address, priority\}^+ = \{address, priority\}$

$\{HScode, HSname\}^+ = \{HScode, HSname, Hscity\}$
$\{HScode, HScity\}^+ = \{HScode, HSname, Hscity\}$
$\{HScode, GPA\}^+ = \{HScode, HSname, Hscity, GPA, priority\}$
$\{HScode, priority\}^+ = \{HScode, HSname, Hscity, priority\}$

$\{HSname, HScity\}^+ = \{HScode, HSname, Hscity\}$
$\{HSname, GPA\}^+ = \{Hsname, GPA, priority\}$
$\{HSname, priority\}^+ = \{Hsname, priority\}$

$\{HScity, GPA\}^+ = \{HScity, GPA, priority\}$
$\{HScity, priority\}^+ = \{HScity, priority\}$
$\{GPA, priority\}^+ = \{GPA, priority\}$

Complete the rest:

- Closures for subsets with three attributes
- Closures for subsets with four attributes
- Closures for subsets with five attributes
- Closures for subsets with six attributes
- Closures for subsets with seven attributes
- Closure for the complete set

Key or Keys?

- Can a relation have more than one key?
  - Yes.
- Examples:

For R(A,B,C)

$A \rightarrow B$,
$B \rightarrow C$
$C \rightarrow A$

3 Keys:
A or B or C

$AB \rightarrow C$,
$BC \rightarrow A$

2 Keys:
AB or BC

$A \rightarrow BC$,
$B \rightarrow AC$

2 Keys:
A or B

Back to the main problem...

WSU CptS 451

- Given a relation schema, we need to decide whether it is a good design
  - Such a decision must be guided by an understanding of what problems, if any, arise from the current schema.
    - Redundancy, Update/Insert/Delete anomalies
  - FDs that hold on the relation may cause such problems.
  - Example: hourly_emp relation

ssn	name	dept	rating	hourly_pay	num_hours
111-11-1111	Kelly	123	5	18	40
222-11-2222	Kyle	124	4	16	40
333-11-3333	John	124	4	16	20
444-11-4444	Roseanne	123	4	16	20
555-11-5555	Min	123	5	18	40

Key : ssn FDs: ssn $\rightarrow$ name,dept,rating,num_hours

rating $\rightarrow$ hourly_pay

Back to the main problem…

WSU CptS 451

- Several "Normal Forms" have been proposed. If a relation is in one of these normal forms, we know that certain kinds of problems does not exist.
  - BCNF (most important normal forms from DB design standpoint)
  - BCNF - Main Idea: We want all attributes in every tuple to be determined by the tuple's key attributes, i.e. part of a *superkey*
    - X → A is OK if X is a (super)key
    - X → A is not OK otherwise

What does this say about redundancy?

ssn	name	dept	rating	hourly_pay	num_hours
111-11-1111	Kelly	123	5	18	40
222-11-2222	Kyle	124	4	16	40
333-11-3333	John	124	4	16	20
444-11-4444	Roseanne	123	4	16	20
555-11-5555	Min	123	5	18	40

Key : ssn **FDs**: ssn → name,dept,rating, num_hours
rating → hourly_pay

Design by Decomposition – Eliminate Anomalies

- Main idea:
  - X → A is OK if X is a (super)key
  - X → A is not OK otherwise
  - Need to decompose the table, but how?

Decompositions in General

R(A1, ..., An, B1, ..., Bm, C1, ..., Cp)

R1(A1, ..., An, B1, ..., Bm)

R2(A1, ..., An, C1, ..., Cp)

R1 = projection of R on A1, ..., An, B1, ..., Bm

R2 = projection of R on A1, ..., An, C1, ..., Cp

Lossless Decomposition

R(A1, ..., An, B1, ..., Bm, C1, ..., Cp)

R1(A1, ..., An, B1, ..., Bm)

R2(A1, ..., An, C1, ..., Cp)

- Decomposition of R into R1 and R2 is lossless if,
  1. {A1, ..., An, B1, ..., Bm} U {A1, ..., An, C1, ..., Cp} = {A1, ..., An, B1, ..., Bm, C1, ..., Cp)
  2. $R1 \bowtie R2 = R$

Lossless Decomposition

ssn	name	dept	rating	hourly_ pay	num_ hours
111-11-1111	Kelly	123	5	18	40
222-11-2222	Kyle	124	4	16	40
333-11-3333	John	124	4	16	20
444-11-4444	Roseanne	123	4	16	20
555-11-5555	Min	123	5	18	40

Key : ssn
FDs: ssn $\rightarrow$ name,dept,rating,num_hours
rating $\rightarrow$ hourly_pay

Hourly_Emps

H1

ssn	name	dept	rating	num_hours
111-11-1111	Kelly	123	5	40
222-11-2222	Kyle	124	4	40
333-11-3333	John	124	4	20
444-11-4444	Roseanne	123	4	20
555-11-5555	Min	123	5	40

H2

rating	hourly_pay
4	16
5	18

1. {ssn, name,dept,rating,num_hours} U {rating_hourly_pay} = {ssn, name,dept,rating,num_hours,rating_hourly_pay }

2. H1⋈H2 = Hourly_Emps

Boyce-Codd Normal Form

Definition. A relation R is in BCNF if:
Whenever $X \rightarrow B$ is a non-trivial dependency, then X is a key or superkey.

– i.e., X^+={all attributes}

There are no "bad" FDs

Boyce-Codd Normal Form

WSU CptS 451

Definition. A relation R is in BCNF if:
Whenever X →B is a non-trivial dependency, then X is a key or superkey.

– i.e., X^+={all attributes}

hourly_emp (SSN, name, dept, rating, hourly_pay, num_hours)

ssn → name,dept,rating, num_hours
rating → hourly_pay

Key: {SSN}

hourly_emp is not in BCNF.
Not every FD has a key on the left hand side

Student(SSN, sName, address,HScode, HSname, HScity, GPA, priority)

SSN → sName, address, GPA
HScode → HSname, HScity
HSname, HScity → Hscode
GPA → priority

Key: {SSN, HScode}

Student is not in BCNF.
Not every FD has a key on the left hand side

Boyce-Codd Normal Form – Example1

Given R(A,B,C,D)

$AB \rightarrow C$

$BC \rightarrow D$

$CD \rightarrow A$

$AD \rightarrow B$

Calculate:

$\{A,B\}^+ = \{A,B,C,D\}$ key

$\{B,C\}^+ = \{B,C,D,A\}$ key

$\{C,D\}^+ = \{C,D,A, B\}$ key

$\{A,D\}^+ = \{A,D,B,C\}$ key

In BCNF

Boyce-Codd Normal Form – Example2

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

AB → C

DE → C

B → D

~~BC → B~~ trivial FD

Calculate:

$\{A,B\}^+ = \{A,B,C,D\}$ X

$\{D,E\}^+ = \{D,E,C\}$ X

$\{B\}^+ = \{B,D\}$ X

Not in BCNF

Boyce-Codd Normal Form – Example3

WSU CptS 451

Given R(A,B,C,D)

AB → C

AB → D

C → A

D → B

Calculate:

$\{A,B\}^+ = \{A,B,C,D\}$ key

$\{C\}^+ = \{C,A\}$ X

$\{D\}^+ = \{D,B\}$ X

Not in BCNF

Lossless BCNF Decomposition

WSU CptS 451

- Next we will study algorithm to decompose a relational schema into sub-schemas which are in BCNF such that the decomposition is lossless
- Setting:

  Given relation **R**, and

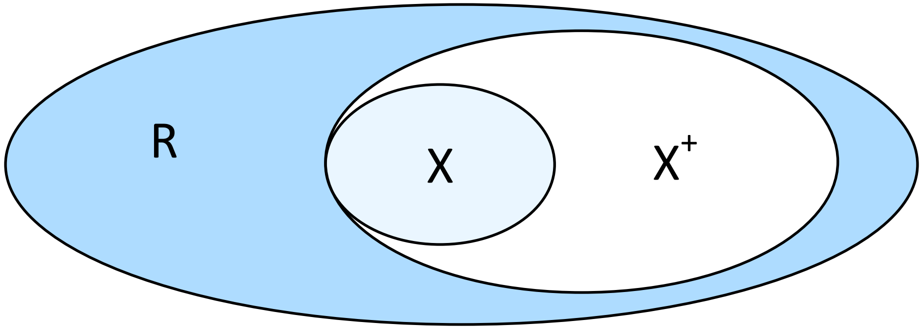
  F, FDs for R

  Suppose relation R has BCNF violation $\mathbf{X} \rightarrow \mathbf{Y}$.

Decomposition to Reach BCNF (II)

The FD $X \rightarrow Y$ violates BCNF.

1. Expand X to include X^+. (Cannot be all attributes - why?)
2. Decompose R into $R1(X^+)$ and R2(X,rest), i.e., $X \cup (R-X^+)$.



3. Find the FD's for the decomposed relations.
   – Project the FD's from F^+ on R1 and R2:
     • calculate all consequents of *F* that involve only attributes from R1 and R2
4. Iterate over all the resulting sub schemes until all in BCNF

Note: Any table with only 2 attributes is always in BCNF!!!

BCNF Decomposition Algorithm

WSU CptS 451

Input: relation R and F (FDs for R)
Output: decomposition of R into BCNF relations with "lossless join"

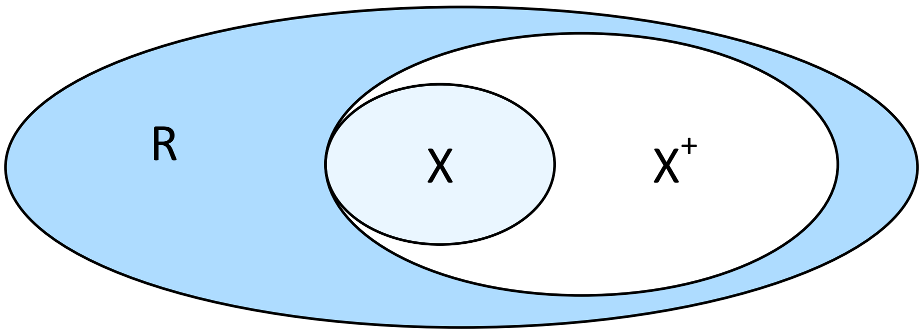
- Compute keys for R
- Repeat until all relations are in BCNF:
  - Pick any R' with $X \rightarrow Y$ that violates BCNF
  - Decompose R' into $R_1(X^+)$ and $R_2(X, rest)$
  - Compute FDs for R_1 and R_2
  - Compute keys for R_1 and R_2

Decomposition to Reach BCNF (II)

WSU CptS 451

The FD X -> Y violates BCNF.

1. Expand X to include X^+.
   – Cannot be all attributes - why?
2. Decompose R into R1(X^+) and R2(X,rest), i.e., X U (R-X^+).



How do we calculate this?

3. Find the FD's for the decomposed relations.
   – Project the FD's from F on R1 and R2:
     - calculate all consequents of F that involve only attributes from R1 and R2
4. Iterate over all the resulting sub schemes until all in BCNF

How to find the FD's for the decomposed relations?

Let R have a schema **R(A,B,C,D)**

R1 have a schema **R1(A,C)**

FD over R be:

A → B and B → C

How to find the FD's for the decomposed relations?

Algorithm to compute the set of FD's that hold on R1

Input: Relation R
R1, a sub-schema of R.
Set of FDs , ***F*** that hold in R

Output: The set of FDs that hold in R1.

Method:

Let **T** be the set of FDs that hold in **R1** (initially empty).

- For each **X** that is a subset of **R1,** do
  - Compute X^+
  - For each attribute **B** in $\mathbf{X^+}$ in such that:
    - ⇒B is in R1
    - ⇒B is not in X
      - Add X → B to **T** (i.e., the functional dependency **X → B** holds in **R1)**
- Eliminate trivial dependencies from **T**

Example - 1

Let R have a schema **R(A,B,C,D)**
R1 have a schema **R1(A,C)**
FD over R be:

A → B and B → C

- Compute $\{A\}^+ = \{A,B,C\}$
  - hence dependency A → C holds in R1
- Compute $\{C\}^+ = \{C\}$
  - no new dependency gets added.
- Compute $\{AC\}^+ = \{ABC\}$
  - AC → AC holds, but it is a trivial dependency. Therefore, no new dependency gets added.

In general you can limit search as follows:

1. It is not necessary to consider the closure of the set of **all attributes**
   - For example, {AC}+ need not have been considered in the above example
2. Not necessary to consider a set of attributes that does not contain the "left hand side" of any dependency.
   - {C}+ need not have been considered in the above example
3. Not necessary to consider a set that contains an attribute that is not in the "left hand side" of any functional dependency
   - {AC}+ need not have been considered in the above example.

Example - 2

Consider R(A,B,C,D,E) and R1(A,B,C)

FD on R be A → D, B → E, DE → C

- Compute $\{A\}^+$ == {A,D}
  - no dependency gets added.
- Compute $\{B\}^+$ == {B,E}
  - no dependency gets added
- $\{C\}^+$ does not need to be considered since {C} not in the left hand side of any FD
- Compute $\{AB\}^+$ == {A,B,C,D,E}
  - add dependency AB → C
- {AC}+ and {BC}+ do not need to be considered since {C} not in the left hand side of any FD
- Since {AB}+ == all attributes in R, {ABC} need not be considered.
- Hence, the only dependency on R1 is: AB → C

Example - 2

Consider R(A,B,C,D,E) and R1(A,B,C)

FD on R be $A \rightarrow D$, $B \rightarrow E$, $DE \rightarrow C$

BCNF Decomposition – Example1

Given R(A,B,C,D,E)
with functional dependencies:
D → B
CE → A

a. In BCNF?
{D}+ = {D,B}   D is not a key and D → B violates BCNF.
{CE}+ = {C,E,A}   CE is not a key and CE → A violates BCNF.
The Key for R is {CDE}

BCNF Decomposition – Example1 (cont.)

Given R(A,B,C,D,E) with functional dependencies:

D → B

CE → A

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

decompose R (apply D → B)

R1(B,D)

R2(A,C,D,E)

Compute FDs for R1: D → B

Compute key for R1: D

(see next slide)

(in BCNF)

Compute FDs for R2: CE → A, CDE → A (see next slide)

Compute keys for R2: CDE (see next slide)

(not in BCNF) : CE → A violates BCNF, CE is not a key

decompose R2 (apply CE → A)

Any table with only 2 attributes is always in BCNF!!!

R3(A,C,E)

R4(C,D,E)

Compute FDs for R3: CE → A

Compute keys for R3: CE

(see next-next slide)

(in BCNF)

Compute FDs for R4: None

Compute keys for R3: CDE

(see next-next slide)

(in BCNF)

R(A,B,C,D,E) is decomposed into R1(B,D), R3(A,C,E), and R4(C,D, E)

The decomposition is **dependency-preserving. Why?**

BCNF Decomposition – Example1(cont.)

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

D → B

CE → A

Compute FD's for R1(B,D):

$\{B\}^+$
$\{D\}^+$
$\{B,D\}^+$

{D} is the key

Compute FD's for R2(A,C,D,E):

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

{CDE} is the key

BCNF Decomposition – Example1(cont.)

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

D → B

CE → A

Compute FD's for R1(B,D):

~~{B}+~~
{D}+= {B,D} D → B added
~~{B,D}+~~

{D} is the key

Compute FD's for R2(A,C,D,E):

~~{A}+~~	{A,C,D}+
{C}+= {C}	~~{A,C,E}+~~
{D}+= {B,D}	~~{A,D,E}+~~
~~{E}+= {E}~~	{C,D,E}+= {C,D,E,B,A} CDE → A added
~~{A,C}+~~	~~{A,C,D,E}+~~
~~{A,D}+~~	
~~{A,E}+~~	
{C,D}+= {C,D,B}	
{C,E}+= {C,E,A} CE → A added	
{D,E}+= {D,E,B}	

{CDE} is the key

BCNF Decomposition – Example1(cont.)

Given R2(A,C,D,E)

CE → A

CDE → A

Compute FD's for R3(A,C,E):

$\{A\}^+$
$\{C\}^+$
$\{E\}^+$
$\{A,C\}^+$
$\{A,E\}^+$
$\{C,E\}^+$
$\{A,C,E\}^+$

{CE} is the key

Compute FD's for R4(C,D,E):

$\{C\}^+$
$\{D\}^+$
$\{E\}^+$
$\{C,D\}^+$
$\{C,E\}^+$
$\{D,E\}^+$
$\{C,D,E\}^+$

BCNF Decomposition – Example1(cont.)

Given R2(A,C,D,E)

CE → A

CDE → A

Compute FD's for R3(A,C,E):

~~{A}⁺~~
$\{C\}^+ = \{C\}$
$\{E\}^+ = \{E\}$
~~{A,C}⁺~~
~~{A,E}⁺~~
$\{C,E\}^+ = \{C,E,A\}$ CE → A added
~~{A,C,E}⁺~~

{CE} is the key

Compute FD's for R4(C,D,E):

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

{CDE} is the key

BCNF Decomposition – Example3

- Consider a relation with schema **R(A,B,C,D)** and **FD**'s:

  $BC \rightarrow D$,
  $BC \rightarrow A$,
  $D \rightarrow B$,
  $A \rightarrow C$

a. Find the minimal key(s) for this relation.
b. Is R in BCNF?
c. If not in BCNF, decompose the relation into collections of relations that are in BCNF.
d. Are the functional dependencies preserved in the BCNF decomposition?

BCNF Decomposition – Example2

Given **Hourly_Emps**(ssn, name,dept,rating,num_hours,rating_hourly_pay)
with functional dependencies:

ssn → name,dept,rating,num_hours
rating → hourly_pay

a. In BCNF?
{ssn}+ = {ssn,name,dept,rating,num_hours}
ssn is a key and first functional dependency doesn't violate BCNF.
{rating}+ = {rating,hourly_pay}
rating is not a key and second functional dependency violates BCNF.

Therefore, Hourly_Emps is not in BCNF.

BCNF Decomposition – Example2 (cont.)

WSU CptS 451

Hourly_Emps(ssn,name,dept,rating,num_hours,rating,hourly_pay)
ssn → name,dept,rating,num_hours
rating → hourly_pay

Hourly_Emps(ssn,name,dept,rating,num_hours,rating,hourly_pay)

decompose Hourly_Emps
(rating → hourly_pay)

H2(rating,hourly_pay)
Compute FDs for H2: rating → hourly_pay
Compute key for H2: rating

(in BCNF)

H1(ssn,name,dept,rating,num_hours,rating)
Compute FDs for H1: ssn → name,dept,rating,num_hours
Compute key for H1: ssn

(in BCNF)

Hourly_Emps is decomposed into:
H1(ssn,name,dept,rating,num_hours,rating) and H2(rating,hourly_pay)
The BCNF decomposition is dependency-preserving.

BCNF Decomposition – Example4

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

FDs:

SSN → sName, address, GPA GPA → priority

HScode → HSname, HScity HSname, HScity → Hscode

Key : SSN,HScode

a. In BCNF?

No. Several violations.

For example:

$\{HSCode\}^+ = \{HScode, HSname, HScity\}$

HSCode is not a key , HScode → HSname, Hscity violates BCNF.

$\{SSN\}^+ = \{SSN, sName, address, GPA, priority\}$

SSN is not a key , SSN → sName and SSN → address violates BCNF.

BCNF Decomposition – Example2 (cont.)

WSU CptS 451

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

SSN → sName, address, GPA   GPA → priority

HScode → HSname, Hscity   HSname, HScity → Hscode

---

b. Decompose Student (use HScode → HSname, HScity)

S1(HScode,HSname,HScity)

FDs: HScode → HSname, Hscity ; HSname, HScity → Hscode

Keys: {HScode} and {HSname, Hscity}

(in BCNF)

S2(SSN, sName, address,HScode,GPA, priority)

FDs: SSN → sName,address,GPA,priority ; GPA→ priority

Key: {SSN,HScode}

(not in BCNF)

Decompose S2 (use GPA → priority)

S3(GPA, priority) **(in BCNF)**

S4(SSN, sName, address, HScode, GPA) **(not in BCNF)**

(next slide)

BCNF Decomposition – Example2 (cont.)

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

SSN → sName, address, GPA　　GPA → priority

HScode → HSname, Hscity　　HSname, HScity → Hscode

S4(SSN, sName, address,HScode,GPA) **(not in BCNF)**

FDs: SSN → sName,address,GPA,priority

Key: {SSN,HScode}

Decompose (use SSN → sName,address,GPA,priority)

S5(SSN, sName, address,GPA, priority) **(in BCNF)**

FDs: SSN → sName,address,GPA,priority

Key: {SSN,HScode}

S6(SSN, HScode) **(in BCNF)**

Student is decomposed into S1, S3, S5, and S6

BCNF Decomposition for Student

Student

SSN	sName	address	HSCode	HSname	HScity	GPA	Priority
1111	Kyle	Everett	PHS	Pullman High School	Pullman	3.4	2
1111	Kyle	Everett	EHS	Everett High School	Everett	3.4	2
2222	John	Pullman	POHS	Potlatch High School	Potlatch	3.0	3
2222	John	Pullman	MHS	Moscow High School	Moscow	3.0	3
2222	John	Pullman	PHS	Pullman High School	Pullman	3.0	3

S1

HSCode	HSname	HScity
PHS	Pullman High School	Pullman
EHS	Everett High School	Everett
POHS	Potlatch High School	Potlatch
MHS	Moscow High School	Moscow

In BCNF

S2

SSN	sName	address	HSCode	GPA	Priority
1111	Kyle	Everett	PHS	3.4	2
1111	Kyle	Everett	EHS	3.4	2
2222	John	Pullman	POHS	3.0	3
2222	John	Pullman	MHS	3.0	3
2222	John	Pullman	PHS	3.0	3

Not in BCNF

BCNF Decomposition for Student

S2

SSN	sName	address	HSCode	GPA	Priority
1111	Kyle	Everett	PHS	3.4	2
1111	Kyle	Everett	EHS	3.4	2
2222	John	Pullman	POHS	3.0	3
2222	John	Pullman	MHS	3.0	3
2222	John	Pullman	PHS	3.0	3

S3

GPA	Priority
3.4	2
3.0	3

In BCNF

S4

SSN	sName	address	HSCode	GPA
1111	Kyle	Everett	PHS	3.4
1111	Kyle	Everett	EHS	3.4
2222	John	Pullman	POHS	3.0
2222	John	Pullman	MHS	3.0
2222	John	Pullman	PHS	3.0

Not in BCNF

BCNF Decomposition for Student

WSU CptS 451

S4

SSN	sName	address	HSCode	GPA
1111	Kyle	Everett	PHS	3.4
1111	Kyle	Everett	EHS	3.4
2222	John	Pullman	POHS	3.0
2222	John	Pullman	MHS	3.0
2222	John	Pullman	PHS	3.0

S5

SSN	sName	address	GPA
1111	Kyle	Everett	3.4
2222	John	Pullman	3.0

In BCNF

S6

SSN	HSCode
1111	PHS
1111	EHS
2222	POHS
2222	MHS
2222	PHS

In BCNF

BCNF Decomposition for Student

WSU CptS 451

S1

HSCode	HSname	HScity
PHS	Pullman High School	Pullman
EHS	Everett High School	Everett
POHS	Potlatch High School	Potlatch
MHS	Moscow High School	Moscow

S3

GPA	Priority
3.4	2
3.0	3

S5

SSN	sName	address	GPA
1111	Kyle	Everett	3.4
2222	John	Pullman	3.0

S6

SSN	HSCode
1111	PHS
1111	EHS
2222	POHS
2222	MHS
2222	PHS

BCNF Decomposition

- **Claim**: The BCNF decomposition algorithm described results in lossless decompositions (i.e., the original can be reconstructed by joining decomposed relations).
- **Proof**. We will not cover. Check the book (section 19.5) for a detailed discussion on lossless decompositions.

Schema Normalization

WSU CptS 451

- So we have learnt that, if a relation R contains redundancy, we need to decompose it into sub-relations R1, R2, …, Rn such that
  - each Ri is in BCNF, and
  - the decomposition of R into R1, R2, …, Rn is a lossless decomposition.
- We also learnt an algorithm for BCNF decomposition that is guaranteed to be lossless.
- **Does this also guarantee dependency preservation???**
- **Not exactly….**
  - In some cases it is not possible to decompose a relation into BCNF relations that have both the lossless-join and the dependency preservation.
  - **Example**: see next-next slide

Example of a Non-Dependency Preserving Decomposition

street	city	zip
1025 S Main	Pullman	91163
925 S Main	Pullman	99163
925 N Main	Pullman	99164

FD:

street,city → zip

zip → city

There are 2 keys :
{street,city} and {street,zip}

zip → city is a BCNF vialotion, so we must decompose into:
R1(street,zip) and R2(city,zip)

R1

street	zip
1025 S Main	91163
925 S Main	99163
925 N Main	99164

FD: none
Key: street,zip

R2

city	zip
Pullman	91163
Pullman	99164

FD: zip → city
Key: zip

- Decomposition of address into R1 and R2 is lossless.
- Furthermore, R1 and R2 are in BCNF and doesn't contain any redundancy
- However, the decomposition does not preserve FD street,city → zip .
  - we cannot enforce the FD street,city → zip by checking FD’s in these decomposed relations.

Third Normal Form (3NF)

WSU CptS 451

- The solution to the problem in the previous slide is to relax the BCNF requirement slightly and allow for **some** redundancy.

- This relaxed condition is called Third Normal Form - 3NF.
  - Disadvantage: storage overhead, anomalies.
  - Advantage: preserve dependencies

Third Normal Form (3NF)

- Relation R with FDs ***F*** is in 3NF if, for all **X** $\rightarrow$ **A** in ***F,*** one of the following is true:
  - A $\in$ X (it is a *trivial* FD), or
  - X is a superkey, or
  - **A** is part of some (minimal candidate) key for R.
- *Minimality* of a key is crucial in third condition above!
- If R is in BCNF, obviously in 3NF.
- If R is in 3NF, some redundancy is possible. It is a compromise, used when BCNF not achievable (e.g., no ``good'' decomposition, or performance considerations).
  - *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible.*

Third Normal Form (3NF) - Example

street	city	zip
1025 S Main	Pullman	91163
925 S Main	Pullman	99163
925 N Main	Pullman	99164

FD:

street,city → zip

zip → city

There are 2 keys :
{street,city} and {street,zip}

Not in BCNF
But in 3NF

Summary of Schema Refinement

- If a relation is in BCNF, it is free of redundancies that can be detected using FDs. Thus, trying to ensure that all relations are in BCNF is a good heuristic.
- If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations.
  - Must consider whether all FDs are preserved. If a lossless-join, dependency preserving decomposition into BCNF is not possible (or unsuitable, given typical queries), should consider decomposition into 3NF.
  - Decompositions should be carried out and/or re-examined while keeping *performance requirements* in mind.