Normalization and Normal Forms

- A normal form specifies a set of conditions that the relational schema must satisfy in terms of its constraints – they offer varied levels of guarantee for the design
- Normalization rules are divided into various normal forms. Most common normal forms are:
    - First Normal Form (1NF)
    - Second Normal Form (2NF)
    - Third Normal Form (3NF)
- Informally, a relational database relation is often described as "normalized" if it meets third normal form. Most 3NF relations are free of insertion, update, and deletion anomalies

1NF: First Normal Form

- A relation is in First Normal Form if and only if all underlying domains contain atomic values only (doesn't have multivalued attributes (MVA))
- **STUDENT(Sid, Sname, Cname)**

Students

SID	Sname	Cname
S1	A	C,C++
S2	B	C++, DB
S3	A	DB

SID : Primary Key

MVA exists ⇒ Not in 1NF

Students

SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB

SID, Cname : Primary Key

No MVA ⇒ In 1NF

Source: http://www.edugrabs.com/normal-forms/#fnf

2NF: Second Normal Form

- Relation $\boldsymbol{R}$ is in Second Normal Form (2NF) only if :
  - $\boldsymbol{R}$ is in 1NF and
  - $\boldsymbol{R}$ contains no Partial Dependency

Partial Dependency:

Let $\boldsymbol{R}$ be a relational Schema and $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A}$ be the attribute sets over $\boldsymbol{R}$ where $\boldsymbol{X}$: Any Candidate Key, $\boldsymbol{Y}$: Proper Subset of Candidate Key, and $\boldsymbol{A}$: Non Prime Attribute

If $Y \rightarrow A$ exists in R, then R is not in 2NF.

($\boldsymbol{Y} \rightarrow \boldsymbol{A}$) is a Partial dependency only if

- $\boldsymbol{Y}$: Proper subset of Candidate Key
- $\boldsymbol{A}$: Non Prime Attribute

A **prime attribute** *of a relation is an attribute that is a part of a candidate key of the relation*

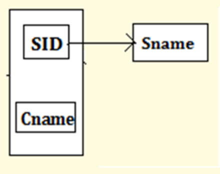
2NF (2)

- **STUDENT(Sid, Sname, Cname)** (already in 1NF)

Students:

SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB

(SID, Cname): Primary Key



Functional Dependencies:
$\{SID, Cname\} \rightarrow Sname$
$SID \rightarrow Sname$

Partial Dependencies:
$SID \rightarrow Sname$ (as SID is a Proper Subset of Candidate Key $\{SID, Cname\}$)

- Redundancy?
  - Sname
- Anomaly?
  - Yes

Key Normalization

R1:

SID	Sname
S1	A
S2	B
S3	A

{SID}: Primary Key

R2:

SID	Cname
S1	C
S1	C++
S2	C++
S2	DB
S3	DB

{SID,Cname}: Primary Key

The above two relations R1 and R2 are
1. Lossless Join
2. 2NF
3. Dependency Preserving

Source: http://www.edugrabs.com/2nf-second-normal-form/

3NF: Third Normal Form

Let $\boldsymbol{R}$ be the relational schema.

- [E. F. Codd,1971] $\boldsymbol{R}$ is in 3NF only if:
  - $\boldsymbol{R}$ should be in 2NF
  - $\boldsymbol{R}$ should not contain transitive dependencies (OR, Every non-prime attribute of R is non-transitively dependent on every key of R)
- [Carlo Zaniolo, 1982] Alternately, $\boldsymbol{R}$ is in 3NF iff for each of its functional dependencies $X \rightarrow A$, at least one of the following conditions holds:
  - X contains A (that is, A is a subset of X, meaning $X \rightarrow A$ is trivial functional dependency), or
  - X is a superkey, or
  - Every element of $A - X$, the set difference between A and X, is a *prime attribute* (i.e., each attribute in $A - X$ is contained in some candidate key)
- [Simple Statement] A relational schema $\boldsymbol{R}$ is in 3NF if for every FD $\boldsymbol{X} \rightarrow \boldsymbol{A}$ associated with R either
  - $\boldsymbol{A} \subseteq \boldsymbol{X}$ (that is, the FD is trivial) or
  - $\boldsymbol{X}$ is a superkey of $\boldsymbol{R}$ or
  - $\boldsymbol{A}$ is part of some candidate key (not just superkey!)
- **A relation in 3NF is naturally in 2NF**

3NF is 2NF without any transitive dependencies??

3NF (2): Transitive Dependency

- A **transitive dependency** is a functional dependency which holds by virtue of transitivity. A transitive dependency can occur only in a relation that has three or more attributes.
- Let A, B, and C designate three distinct attributes (or distinct collections of attributes) in the relation. Suppose all three of the following conditions hold:
  - $A \rightarrow B$
  - It is not the case that $B \rightarrow A$
  - $B \rightarrow C$
- Then the functional dependency $A \rightarrow C$ (which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency

3NF Decomposition: Motivation

- There are some situations where
  - BCNF is not dependency preserving, and
  - Efficient checking for FD violation on updates is important
- Solution: define a weaker normal form, called Third Normal Form (3NF)
  - Allows some redundancy (with resultant problems; as seen above)
  - But functional dependencies can be checked on individual relations without computing a join
  - **There is always a lossless-join, dependency-preserving decomposition into 3NF**

3NF Decomposition (4): Algorithm

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Given: relation R, set F of functional dependencies
- Find: decomposition of R into a set of 3NF relation R_i
- Algorithm:
  a) Eliminate redundant FDs, resulting in a canonical cover F_c of F
  b) Create a relation $R_i = XY$ for each FD $X \rightarrow Y$ in F_c
  c) If the key K of R does not occur in any relation R_i, create one more relation $R_i = K$

3NF Decomposition (6): Algorithm

Module 27
Partha Pratim Das
Objectives & Outline
Decomposition to 3NF
Test
Algorithm
Practice Problem
Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison
Module Summary

- Upon decomposition:
  - Each relation schema R_i is in 3NF
  - Decomposition is
    - ▷ Dependency Preserving
    - ▷ Lossless Join
- Prove these properties

BCNF Decomposition: BCNF Definition

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- A relation schema R is in BCNF with respect to a set F of FDs if for all FDs in F^+ of the form

  $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$ at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (that is, $\beta \subseteq \alpha$)
  - α is a superkey for R

BCNF Decomposition (2): Testing for BCNF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
  a) Compute α^+ (the attribute closure of α), and
  b) Verify that it includes all attributes of R, that is, it is a superkey of R.
- **Simplified test**: To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than checking all dependencies in F^+.
  - If none of the dependencies in F causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.
- However, **simplified test using only F is incorrect when testing a relation in a decomposition of R**
  - Consider $R = (A, B, C, D, E)$, with $F = \{A \rightarrow B, BC \rightarrow D\}$
    - Decompose R into $R_1 = (A, B)$ and $R_2 = (A, C, D, E)$
    - Neither of the dependencies in F contain only attributes from (A, C, D, E) so we might be mislead into thinking R_2 satisfies BCNF.
    - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.

BCNF Decomposition (6): Example

- $R = (A, B, C)$

  $$F = \{A \rightarrow B, \; B \rightarrow C\}$$

  $Key = \{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not superkey)
- Decomposition
  - $R_1 = (B, C)$
  - $R_2 = (A, B)$

BCNF Decomposition (8): Dependency Preservation

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
- Test
- Algorithm
- Practice Problem

Decomposition to BCNF
- Test
- Algorithm
- Practice Problem
- Comparison

Module Summary

- It is not always possible to get a BCNF decomposition that is dependency preserving
- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $\quad L \rightarrow K\}$
  Two candidate keys $= JK$ and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve
  $$JK \rightarrow L$$
  This implies that testing for $JK \rightarrow L$ requires a join

Comparison of BCNF and 3NF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

S#	3NF	BCNF
1.	It concentrates on Primary Key	It concentrates on Candidate Key
2.	Redundancy is high as compared to BCNF	0% redundancy
3.	It preserves all the dependencies	It may not preserve the dependencies
4.	A dependency $X \rightarrow Y$ is allowed in 3NF if X is a super key or Y is a part of some key	A dependency $X \rightarrow Y$ is allowed if X is a super key

MVD: Definition

- Let R be a relation schema and let $\alpha \subseteq$ R and $\beta \subseteq$ R. The **multivalued dependency**

  $\alpha \twoheadrightarrow \beta$

  holds on R if in any legal relation *r(R)*, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2\ [\alpha]$, there exist tuples t_3 and t_4 in r such that:

$t_1[\alpha] = t_2\ [\alpha] = t_3\ [\alpha] = t_4\ [\alpha]$
$t_3[\beta] = t_1\ [\beta]$
$t_3[R - \beta] = t_2[R - \beta]$
$t_4\ [\beta] = t_2[\beta]$
$t_4[R - \beta] = t_1[R - \beta]$

Note:
- *The total number of attributes should be more than two.*
- *If there exists 3 attributes, then 2 attributes must be independent of each other.*

Example: A relation of university courses, the books recommended for the course, and the lecturers who will be teaching the course:

- **course** $\twoheadrightarrow$ **book**
- **course** $\twoheadrightarrow$ **lecturer**

Test: **course** $\twoheadrightarrow$ **book**

Course	Book	Lecturer	Tuples
AHA	Silberschatz	John D	t1
AHA	Nederpelt	William M	t2
AHA	Silberschatz	William M	t3
AHA	Nederpelt	John D	t4
AHA	Silberschatz	Christian G	
AHA	Nederpelt	Christian G	
OSO	Silberschatz	John D	
OSO	Silberschatz	William M	

MVD: Theory

Module 29

Partha Pratim Das

Objectives & Outline

Multivalued Dependency

Definition

Example

Use

Theory

Decomposition to 4NF

Module Summary

	Name	Rule
C-	Complementation	If $X \twoheadrightarrow Y$, then $X \twoheadrightarrow (R - (X \cup Y))$.
A-	Augmentation	If $X \twoheadrightarrow Y$ and $W \supseteq Z$, then $WX \twoheadrightarrow YZ$.
T-	Transitivity	If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow (Z - Y)$.
	Replication	If $X \rightarrow Y$, then $X \twoheadrightarrow Y$ but the reverse is not true.
	Coalescence	If $X \twoheadrightarrow Y$ and there is a W such that $W \cap Y$ is empty, $W \rightarrow Z$ and $Y \supseteq Z$, then $X \rightarrow Z$.

- A MVD $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ in $\mathbf{R}$ is called a trivial MVD is
  - $\mathbf{Y}$ is a subset of $\mathbf{X}$ $(\mathbf{X} \supseteq \mathbf{Y})$ or
  - $\mathbf{X} \cup \mathbf{Y} = \mathbf{R}$. Otherwise, it is a non trivial MVD and we have to repeat values redundantly in the tuples.

MVD: Theory (2)

- From the definition of multivalued dependency, we can derive the following rule:
  - If $\alpha \rightarrow \beta$, then $\alpha \twoheadrightarrow \beta$

  That is, every functional dependency is also a multivalued dependency
- The **closure** D^+ of D is the set of all functional and multivalued dependencies logically implied by D.
  - We can compute D^+ from D, using the formal definitions of functional dependencies and multivalued dependencies.
  - We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice
  - For complex dependencies, it is better to reason about sets of dependencies using a system of inference rules

Fourth Normal Form

- A relation schema R is in **4NF** with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \twoheadrightarrow \beta$, where $\alpha \subseteq$ R and $\beta \subseteq$ R, at least one of the following hold:
  - $\alpha \twoheadrightarrow \beta$ is trivial (that is, $\beta \subseteq \alpha$ or $\alpha \cup \beta =$ R)
  - α is a superkey for schema R
- If a relation is in 4NF it is in BCNF

Restriction of Multivalued Dependencies

- The restriction of D to R_i is the set D_i consisting of
  - All functional dependencies in D^+ that include only attributes of R_i
  - All multivalued dependencies of the form
    $\alpha \twoheadrightarrow (\beta \cap R_i)$
    where $\alpha \subseteq R_i$ and $\alpha \twoheadrightarrow \beta$ is in D^+

Temporal Data

Module 30

Partha Pratim Das

Objectives & Outline

Database Design Process
Normal Forms
Normalization & De-Normalization
Bad Design
LIS Example

Temporal Databases
Temporal Data
Uni / Bi Temporal
Example

Module Summary

- **Temporal data** have an association time interval during which the data are valid.
- A **snapshot** is the value of the data at a particular point in time
- In practice, database designers may add start and end time attributes to relations
- For example, *course(course_id, course_title)* is replaced by
  course(course_id, course_title, start, end)
  - Constraint: no two tuples can have overlapping valid times and are Hard to enforce efficiently
  - Foreign key references may be to current version of data, or to data at a point in time
    - For example, student transcript should refer to course information at the time the course was taken

crosstab where values for one attribute become column names

Modeling Temporal Data: Uni / Bi Temporal

- There are **two different aspects** of time in temporal databases.
  - **Valid Time**: Time period during which a fact is true in real world, provided to the system.
  - **Transaction Time**: Time period during which a fact is stored in the database, based on transaction serialization order and is the timestamp generated automatically by the system.
- Temporal Relation is one where each tuple has associated time; either valid time or transaction time or both associated with it.
  - **Uni-Temporal Relations**: Has one axis of time, either *Valid Time* or *Transaction Time*.
  - **Bi-Temporal Relations**: Has both axis of time – *Valid time* and *Transaction time*. It includes Valid Start Time, Valid End Time, Transaction Start Time, Transaction End Time.

Source: https://www.mytecbits.com/oracle/oracle-database/what-is-temporal-database

Modeling Temporal Data: Summary

Module 30

Partha Pratim Das

Objectives & Outline

Database Design Process
Normal Forms
Normalization & De-Normalization
Bad Design
LIS Example

Temporal Databases
Temporal Data
Uni / Bi Temporal
Example

Module Summary

- **Advantages**
  - The main advantages of this bi-temporal relations is that it provides historical and roll back information.
    - ▷ Historical Information – Valid Time.
    - ▷ Rollback Information – Transaction Time.
  - *For example*, you can get the result for a query on John's history, like: Where did John live in the year 2001?. The result for this query can be got with the valid time entry. The transaction time entry is important to get the rollback information.
- **Disadvantages**
  - More storage
  - Complex query processing
  - Complex maintenance including backup and recovery

Source: https://www.mytecbits.com/oracle/oracle-database/what-is-temporal-database

Let R(A, B, C, D) be a relation(all attributes have atomic values only) with the following functional dependencies:
FDs={B → C, D → A}

Find the highest normal form in which the relation R is in

1NF

Candidate key is (BD)

$A \rightarrow B$

A proper subset of C.K

B set non-prime attribute

$B \rightarrow C$