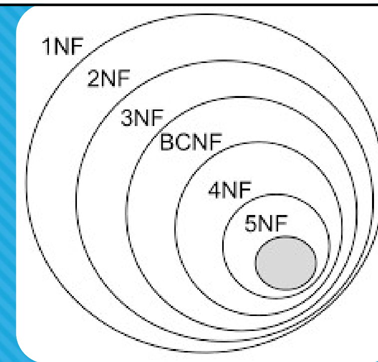
CSCI 5333: DBMS
Chapter 8



Normalization for Relational Databases

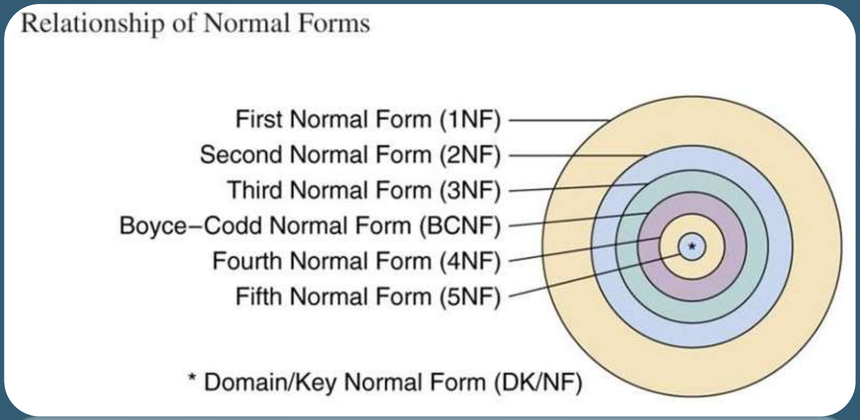
Shajadul Hasan Khondker

Department of Computing Sciences

University of Houston – Clear Lake

Chapter Outline

- Drawback in Relational Database Design
- Decomposition of Relations
- Informal guideline of Normalization
- Functional Dependencies
- Armstrong Axioms
- Attribute Closure
- Canonical Cover
- 3rd Normal Form (3NF)
- Boyce Code Normal Form (BCNF)



Drawback in Relational Database Design

- Relational database design requires that we find a “good” collection of relation schemas. A bad design may lead to:
  - Repetition of Information.
  - Inability to represent certain information.

- Design Goals:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of database integrity constraints.

Example

- **Consider the relation schema:**

Lending = (branch-name, branch-city, assets, customer-name, loan-number, amount)

branch-name	*branch-city*	*assets*	*customer-name*	*loan-number*	*amount*
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Redundancy:
  - Data for *branch-name, branch-city,* assets are repeated for each loan that a branch makes
  - Wastes space
  - Complicates updating, introduces possibility of inconsistency of *assets* value
- Null values
  - Cannot store information about a branch if no loans exist
  - Can use null values, but they are difficult to handle.

Decomposition

- The way which we eliminate inconsistency in relations is to decompose that relation into two or more *sub-relations*.
- We check to see if a relation is in a *normal form* e.g. BCNF or 3NF, and if it does not meet the criteria, we split the relation into sub-relations
- In the case that a relation R is not in "*good*" form, decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that:
  - each relation is in good form
  - the decomposition is a *lossless-join* decomposition
- Our theory is based on:
  - functional dependencies
  - multivalued dependencies

Decomposition

- Decompose the relation schema *Lending-schema* into:

  Branch = (branch-name, branch-city, assets)

  Loan-info = (loan-number, customer-name, branch-name, amount)

- All attributes of an original schema (R) must appear in the decomposed schemas (R_1, R_2):

  $R = R_1 \cup R_2$

- Lossless-join decomposition:
  For all possible relations r on schema R

  $r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$

Decomposition (Example)

R

Movie : Table

title	year	length	type	studioname	starName
Star Wars	1977	124	Sci-Fi	Fox	Carrie Fisher
Star Wars	1977	124	Sci-Fi	Fox	Mark Hamill
Braveheart	1994	280	Action	Universal	Mel Gibson
Breakfast Club	1986	200	Comedy	Paramount	Molly Ringwal
Breakfast Club	1986	200	Comedy	Paramount	Emilo Estavez

Record: 5 of 5

S

Movie1 : Table

title	year	length	type	studioName
Star Wars	1977	124	Sci-Fi	Fox
Braveheart	1994	280	Action	Universal
Breakfast Club	1986	200	Comedy	Paramount

Record: 3 of 3

T

Movie2 : Table

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
BraveHeart	1994	Mel Gibson
Breakfast Club	1986	Molly Ringwal
Breakfast Club	1986	Emilo Estavez

Record: 6 of 6

Decomposing Relations

- In previous, we saw that we could ‘decompose’ the bad relation schema like the following:

  Data(sid, sname, addrs, cid, cname, grade)

  to a ‘better’ set of relation schema

    Student(sid, sname, addrs)
    Course(cid, cname)
    Enrolled(sid, cid, grade)

Are all Decompositions good?

- Consider our motivating example:

```
Data = (sid,sname,address,cid,cname,grade)
```

- Alternatively we could decompose into

```
R1 = (sid,sname,address)
R2 = (cid,cname,grade)
```

- But this decomposition loses information about the relationship between students and courses.
- *We require to understand the* ***data semantics as well***.

Informal Guidelines for Normalization

- Let's say we want to create a table of user information, and we want to store each users' Name, Company, Company Address, and some personal bookmarks, or urls.
- You might start by defining a table structure like this:

users				
name	company	company_address	url1	url2
Joe	ABC	1 Work Lane	abc.com	xyz.com
Jill	XYZ	1 Job Street	abc.com	xyz.com

First Normal Form

- First Normal Form: **NO REPEATING GROUPS**
  1. Eliminate repeating groups in individual tables.
  2. Create a separate table for each set of related data.
  3. Identify each set of related data with a primary key.
- Historically, it was defined to disallow multi-valued attributes, composite attributes, and their combinations
- Create another table in first normal form by eliminating the repeating group (Url#), as shown below:

users				
UserId	name	company	company_address	url
1	Joe	ABC	1 Work Lane	abc.com
1	Joe	ABC	1 Work Lane	xyz.com
2	Jill	XYZ	1 Job Street	abc.com
2	Jill	XYZ	1 Job Street	xyz.com

Second Normal Form

- Second Normal Form: **ELIMINATE REDUNDANT DATA**
  1. Create *separate tables* for sets of values that apply to multiple records.
  2. Relate these tables with a *foreign key*.

users			
userId	name	company	company_address
1	Joe	ABC	1 Work Lane
2	Jill	XYZ	1 Job Street

urls	
relUserId	url
1	abc.com
1	xyz.com
2	abc.com
2	xyz.com

- what happens when we want to add another employee of company ABC?

Third Normal Form (3NF)

- **Third Normal Form:** ELIMINATE DATA NOT DEPENDENT ON KEY
  - Eliminate fields that do not depend on the key.
  - Relate these tables with a foreign key.

users		
userId	name	relCompId
1	Joe	1
2	Jill	2

companies		
compId	company	company_address
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls	
relUserId	url
1	abc.com
1	xyz.com
2	abc.com
2	xyz.com

Functional Dependencies

- Definition of FD
- Closure of a Set of FDs
- Armstrong's Axioms
- Trivial Dependencies

Functional Dependency

- Functional dependencies are constraints on the set of legal relations.
- Requires that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key* which can be used in normalization
- **Definition:**
  - If two tuples of R agree in attribute A then they must agree in another attribute B
- Written as
  - $A \rightarrow B$
  - A *functionally determines* B
  - B is *functionally dependent on* A

Functional Dependencies (Cont.)

- Let R be a relation schema

  $$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency** $\alpha \rightarrow \beta$
  holds on R if and only if for any legal relations $r(R)$, whenever any two tuples t_1 and t_2 of r agree on the attributes α, they also agree on the attributes β. That is,

  $$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider $r(A, B)$ with the following instance of r.

1	4
1	5
3	7

- On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold.

A → B

- If tuple t and tuple u agree on the value of attribute A, then they must also agree on the value of attribute B

Example One

film : Table

title	year	length	type
Star Wars	1977	124	Sci-Fi
Lost Boys	1987	130	Horror
The Quiet Man	1965	100	Irish
	0	0	

Record: 4 of 4

title, year $\rightarrow$ length
title, year $\rightarrow$ type

- This says that if two tuples have the same value in their *title* and *year* attributes, then these two tuples **must** have the same values in their *length* and *type* attributes.

Example Two

film : Table

title	year	length	type	star
Star Wars	1977	124	Sci-Fi	Harrison Ford
The Quiet Man	1965	100	Irish	John Wayne
Waynes World	1992	95	Comedy	Mike Meyers
Waynes World	1992	95	Comedy	Dana Carvey
	0	0		

Record: 5 of 5

title, year → length
title, year → type
title, year → star???

Example Three

- Consider the schema

  loan-info(loan-number, customer, amount)

- The following functional dependency *holds* on loan-info
  - **loan-number → amount**
- If more than one person can own a loan does
  - **loan-number → customer** hold?

Exercise

Do the following FD's hold for the above relation?

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

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

A	B	C	D
A1	B1	C1	D1
A1	B2	C1	D2
A2	B2	C2	D2
A2	B3	C2	D3
A3	B3	C2	D4

Trivial Dependencies

- A functional dependency, **$\alpha \rightarrow \beta$,** is said to be ***trivial*** if **$\beta \subseteq \alpha$**
  - *customer-name, loan-number $\rightarrow$ customer-name*
  - *customer-name $\rightarrow$ customer-name*
- ***nontrivial*** if at least one of the attributes in **β** is not in **α**
  - *customer-name, loan-number $\rightarrow$ customer-name, city*
- ***completely nontrivial*** if none of the attributes in **β** are in **α**
  - *customer-name, loan-number $\rightarrow$ city*

Closure of a Set of FDs (F^+)

- Given a set *F* set of functional dependencies, there are certain other functional dependencies that are logically implied by *F*.
  - E.g. If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that **$A \rightarrow C$.**

A	B	C
A1	B1	C3
A1	B1	C3
A2	B4	C5

Closure of a Set of FDs (F^+)

- The set of all functional dependencies logically implied by *F* is the *closure* of *F*.
- We denote the *closure* of *F* by F^+.
- We can find all of F+ by applying Armstrong's Axioms:
  - **if** $\beta \subseteq \alpha$, **then** $\alpha \rightarrow \beta$ **(reflexivity)**
  - **if** $\alpha \rightarrow \beta$, **then** $\gamma\,\alpha \rightarrow \gamma\,\beta$ **(augmentation)**
  - **if** $\alpha \rightarrow \beta$, **and** $\beta \rightarrow \gamma$, **then** $\alpha \rightarrow \gamma$ **(transitivity)**
- These rules are
  - sound (generate only functional dependencies that actually hold) and
  - complete (generate all functional dependencies that hold).

Armstrong's Axioms

- Armstrong's axioms are a set of three rules and by applying these rules repeatedly we can find all of F^+
- **Reflexivity Rule**
  - If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds
  - **Example:** α = (A, B, C, D), β = (A, B)

    (A, B, C, D) $\rightarrow$ (A, B) holds
- **Augmentation Rule**
  - If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds
  - **Example:** α = (A, B, C, D), β = (X, Y), γ = (M, N) then $\gamma\alpha \rightarrow \gamma\beta$

    (A, B, C, D, M, N) $\rightarrow$ (M, N, X, Y) holds

Armstrong's Axioms

- **Transitivity Rule**
  - If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds
  - Example: α = (A, B, C, D), β = (X, Y), γ = (M, N)

    $\alpha \rightarrow \beta$ (A, B, C, D) → (X, Y)

    $\beta \rightarrow \gamma$ (X, Y) → (M, N)

    $\alpha \rightarrow \gamma$ (A, B, C, D) → (M, N) holds
- These rules are *sound,* i.e. they do not generate any incorrect functional dependencies
- The rules are also *complete*, i.e. they generate ALL the possible functional dependencies from the original set that form F^+

Example

- $R = (A, B, C, G, H, I)$
- $F = \{\ A \rightarrow B,\ A \rightarrow C,\ CG \rightarrow H,\ CG \rightarrow I,\ B \rightarrow H\ \}$
- **Some members of F^+**
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$ and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - Augmentation of $CG \rightarrow I$ to infer $CG \rightarrow CGI$, augmentation of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity
    - from $CG \rightarrow H$ *and* $CG \rightarrow I$: "union rule" can be inferred from

Exercise

- Consider the schema
  - **R(A, B, C, D)**
- and the FD's
  - **F = {AB $\rightarrow$ C, C $\rightarrow$ D, D $\rightarrow$ A}**
- What are the **nontrivial** functional dependencies that follow from the given set?
  - **AB $\rightarrow$ D, $\boxed{\cancel{AB \rightarrow A}}$**

Closure of a Set of FDs (F^+)

- There are also some additional rules that are not part of Armstrong's axioms
- We can further simplify manual computation of F^+ by using the following additional rules.
  - If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds **(union)**
  - If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds **(decomposition)**
  - If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds **(pseudotransitivity)**
- The above rules can be inferred from Armstrong's axioms.

Union Rule

If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then
$\alpha \rightarrow \beta\gamma$ holds

α = (A, B, C, D), β = (X, Y), γ = (M, N)

$\alpha \rightarrow \beta$ (A, B, C, D) $\rightarrow$ (X, Y)

$\alpha \rightarrow \gamma$ (A, B, C, D) $\rightarrow$ (M, N)

$\alpha \rightarrow \beta\gamma$ (A, B, C, D) $\rightarrow$ (X, Y, M, N)

Decomposition Rule

If $\alpha \rightarrow \beta\gamma$ holds, then
$\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds

α = (A, B, C, D), β = (X, Y), γ = (M, N)

$\alpha \rightarrow \beta\gamma$ (A, B, C, D) $\rightarrow$ (X, Y, M, N)

$\alpha \rightarrow \beta$ (A, B, C, D) $\rightarrow$ (X, Y)

$\alpha \rightarrow \gamma$ (A, B, C, D) $\rightarrow$ (M, N)

Pseudotransitivity Rule

If $\alpha \rightarrow \beta$ holds and $\gamma\, \beta \rightarrow \delta$, then
$\alpha\, \gamma \rightarrow \delta$ holds

α = (A, B, C, D), β = (X, Y), γ = (M, N), δ = (P, Q)

$\alpha \rightarrow \beta$ (A, B, C, D) $\rightarrow$ (X, Y)

$\gamma\, \beta \rightarrow \delta$ (M, N, X, Y) $\rightarrow$ (P, Q)

$\alpha\, \gamma \rightarrow \delta$ (A, B, C, D, M, N) $\rightarrow$ (P, Q)

Determining Keys

- We can also use functional dependencies to determine what are the candidate keys of a relation by using the following definition
  - A set of one or more attributes, α, is a candidate key for a relation R if and only of
    - α functionally determine all other attributes of the relation ($\alpha \rightarrow R$), i.e. it is impossible for two distinct tuples of R to agree on all attributes in the set, α (superkeys)
    - no proper subset of α functionally determines all other attributes of R (candidate keys)
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances. e.g., *loan-number* $\rightarrow$ *customer-name*

Example

(ID, Name, Address, Phone, DOB)
(ID, Name, Address, DOB)
(ID, Name, Phone, DOB)
(ID, Name)
(ID, Address)
(ID, Phone)
(ID, DOB)
(ID) → Candidate key

Superkeys

Closure of Attribute Sets

- To test whether a set α is a superkey, we must determine the set of attributes functionally determined by α
- Given a set of attributes α, define the *closure* of α under F (denoted by α^+) as the set of attributes that are functionally determined by α under F:

$$\alpha \rightarrow \beta \text{ is in } F^+ \Leftrightarrow \beta \subseteq \alpha^+$$

- Algorithm to compute α^+, the closure of α under F

result := α;
while (changes to *result*) **do**
  for each $\beta \rightarrow \gamma$ **in** *F* **do**
  begin
    if $\beta \subseteq$ *result* **then** *result* := *result* $\cup \gamma$
  end

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B,\ A \rightarrow C,\ CG \rightarrow H,\ CG \rightarrow I,\ B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $result = ABCGH$ $(CG \rightarrow H$ and $CG \subseteq AGBC)$
  4. $result = ABCGHI$ $(CG \rightarrow I$ and $CG \subseteq AGBCH)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Does $AG \rightarrow R$?
  2. Is any subset of AG a superkey?
     1. Does $A^+ \rightarrow R$?
     2. Does $G^+ \rightarrow R$?

Uses of Attribute Closure

- There are several *uses* of the attribute closure algorithm:
- Testing for superkey:
  - To test if α is a superkey, we compute $\alpha^{+,}$ and check if α^{+} contains all attributes of *R*.
- Testing for functional dependencies
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in F^{+}), just check if $\beta \subseteq \alpha^{+}$.
  - That is, we compute α^{+} by using attribute closure, and then check if it contains β.
  - Is a simple and cheap test, and very useful
- Computing closure of F
  - For each $\gamma \subseteq R$, we find the closure γ^{+}, and for each $S \subseteq \gamma^{+}$, we output a functional dependency $\gamma \rightarrow S$.

Normalization of the Relations

- Canonical Cover
- Properties of Decomposition
- Boyce-Codd Normal Form (BCNF)
- Third Normal Form (3NF)

Canonical Cover

- Sets of functional dependencies may have redundant dependencies that can be inferred from the others
  - Parts of a functional dependency may be redundant
    - E.g. on RHS: $\{A \to B,\ \ B \to C,\ \ A \to CD\}$ can be simplified to $\{A \to B,\ \ B \to C,\ \ A \to D\}$
    - E.g. on LHS: $\{A \to B,\ \ B \to C,\ \ AC \to D\}$ can be simplified to $\{A \to B,\ \ B \to C,\ \ A \to D\}$
- Intuitively, a canonical cover of F is a "minimal" set of functional dependencies equivalent to F, with no redundant dependencies or having redundant parts of dependencies

Extraneous Attributes

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.
  - Attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow C\}$
  - B is extraneous in $AB \rightarrow C$ because $A \rightarrow C$ logically implies $AB \rightarrow C$.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - C is extraneous in $AB \rightarrow CD$ since $A \rightarrow C$ can be inferred even after deleting C

Testing if an Attribute is Extraneous

- Consider a set F of functional dependencies and the functional dependency $\alpha \to \beta$ in F.
  - To test if attribute $A \in \alpha$ is extraneous in α
    1. compute $(\alpha - \{A\})^+$ using the dependencies in F
    2. check that $(\alpha - \{A\})^+$ contains β; if it does, A is extraneous in α
  - To test if attribute $A \in \beta$ is extraneous in β
    1. compute α^+ using only the dependencies in
       $F = (F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$,
    2. check that α^+ contains A; if it does, A is extraneous

Canonical Cover (F_c)

- To compute a canonical cover for *F*:

  $F_c = F$

  repeat
  Use the union rule to replace any dependencies in *F*
  $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$
  Find a functional dependency $\alpha \rightarrow \beta$ with an
  extraneous attribute either in α or in β
  If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$
  until *F* does not change

- Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

Example of Computing a Canonical Cover

- $R = (A, B, C)$
- $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$ *(union rule)*
  - Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A is extraneous in $AB \rightarrow C$ because $B \rightarrow C$ logically implies $AB \rightarrow C$.
  - *Compute B^+ and check it contains C or not. True, it contains C*
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow BC$ since $A \rightarrow BC$ is logically implied by $A \rightarrow B$ and $B \rightarrow C$.
  - Update F and then compute A^+ contains C or not. True, it contains C; so C is redundant
- The canonical cover is:

$$F_C = \{ A \rightarrow B$$
$$B \rightarrow C \}$$

Canonical Cover

- We compute the canonical cover of F by
  - using the union rule to combine FD's with the same left hand side
  - finding and removing extraneous attributes
- A *canonical cover* for F is a set of dependencies F_c such that:
  - F logically implies all dependencies in F_c, and
  - F_c logically implies all dependencies in F, and
  - No functional dependency in F_c contains an extraneous attribute, and
  - Each left side of functional dependency in F_c is unique.

Desirable Properties of Decomposition

- Desirable properties of a decomposition are:
  - Attribute preservation
  - Lossless-join decomposition
  - Dependency preservation
  - No Redundancy (i.e., 3NF or BCNF)
- Attribute preservation:
  - When we decompose a relation schema *R* into several sub-schemas R_1, R_2,.., R_n we want
  - R = $R_1 \cup R_2 \cup \ldots \cup R_n$
  - That is, all attributes of an original schema (*R*) must appear in the decomposition (R_1, $R_{2, \ldots,} R_n$):

Desirable Properties of Decomposition

- Lossless-Join Decomposition
  - An important property of decomposition is that it should be possible to retrieve exactly the same information from the resulting relations as was present in the original relation.
  - Computed using the *natural join*
  - A decomposition of a relation R into relations $R_1, R_2, \ldots, R_n$ is called a *lossless-join* decomposition if the relation R is always the natural join of the relations $R_1, R_2, \ldots, R_n$

$$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

Desirable Properties of Decomposition

- Lossless-Join Decomposition (Cont….)
  - Given R decomposed into R_1 and R_2, the decomposition is lossless if one of two conditions hold;
    - $R_1 \cap R_2 \rightarrow R_1$
    - $R_1 \cap R_2 \rightarrow R_2$
  - That is, the common attributes of R_1 and R_2 must include a candidate key of either R_1 or R_2
  - That is, you can check whether $R_1 \subseteq (R_1 \cap R_2)^+$ or $R_2 \subseteq (R_1 \cap R_2)^+$

Desirable Properties of Decomposition

- Lossless-Join Decomposition Example

Movie : Table

title	year	length	type	studioname	starName
Star Wars	1977	124	Sci-Fi	Fox	Carrie Fisher
Star Wars	1977	124	Sci-Fi	Fox	Mark Hamill
Braveheart	1994	280	Action	Universal	Mel Gibson
Breakfast Club	1986	200	Comedy	Paramount	Molly Ringwal
Breakfast Club	1986	200	Comedy	Paramount	Emilo Estavez

Record: 5 of 5

Movie1 : Table

title	year	length	type	studioName
Star Wars	1977	124	Sci-Fi	Fox
Braveheart	1994	280	Action	Universal
Breakfast Club	1986	200	Comedy	Paramount

Record: 3 of 3

Movie2 : Table

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
BraveHeart	1994	Mel Gibson
Breakfast Club	1986	Molly Ringwal
Breakfast Club	1986	Emilo Estavez

Record: 6 of 6

Desirable Properties of Decomposition

- Example of Lossy-Join Decomposition
  - Lossy-join decompositions result in information loss.
  - Example: Decomposition of *R = (A, B, C, D)*
    S = (A, B)   *T = (C, D)*

R

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3

S

A	B
a_1	b_1
a_2	b_2
a_3	b_3

T

C	D
c_1	d_1
c_2	d_2
c_3	d_3

$S \bowtie T$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_3	d_3
a_2	b_2	c_1	d_1
a_2	b_2	c_2	d_2
a_2	b_2	c_3	d_3
a_3	b_3	c_1	d_1
a_3	b_3	c_2	d_2
a_3	b_3	c_3	d_3

Desirable Properties of Decomposition

- Lossy-Join Decomposition Example

Enrol : Table

student	course	date-enrolled	room	instructor
James	C++	12FEB	101	Coburn
Sarah	C++	20JAN	101	Coburn
Fred	Java	12FEB	200	Smith
Mary	Networks	23JAN	104	Park
James	Maths	12FEB	100	Oshea

Record: 1 of 5

enrol1 : Table

student	course	date-enrolled
James	C++	12FEB
James	Maths	12FEB
Sarah	C++	20JAN
Fred	Java	12FEB
Mary	Networks	23JAN

Record: 3 of 5

enrol2 : Table

date-enrolled	room	instructor
12FEB	101	Coburn
12FEB	200	Smith
12FEB	100	Oshea
20JAN	101	Coburn
23JAN	104	Park

Record: 5 of 5

Desirable Properties of Decomposition

- Dependency Preservation
  - If we decompose a relation, then the set of functional dependencies that held on the original set
    - should also hold on the resulting relations as a whole
    - no joins should be needed to when testing the FD's
  - Let F be the dependencies on a relation R which is decomposed in relations $R_1, R_2, \ldots, R_n$
  - We have to partition the dependencies given by F to $F_1, F_2, \ldots, F_n$ such that
  - $F_1, F_2, \ldots, F_n$ are dependencies that only involve attributes from relations $R_1, R_2, \ldots, R_n$ respectively
  - Need to ensure that $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

Desirable Properties of Decomposition

- Dependency Preservation
  - If the union of dependencies F_i imply all the dependencies in F, then we say that the decomposition has preserved dependencies, otherwise the dependencies has not been preserved
  - If dependencies not preserved then checking updates for violation of functional dependencies may require computing joins, which is expensive
- Example:

  $R = (A, B, C, G, H, I)$, $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, G \rightarrow I, C \rightarrow H\}$
  - If we decompose R to R_1=(A, B, C) and R_2=(G, H, I) then we need to decompose F to
  - $F_1 = \{ A \rightarrow B, A \rightarrow C \}$
  - $F_2 = \{G \rightarrow I \}$
  - $F_1 \cup F_2 \neq F$, *Hence, dependency not preserved*

Desirable Properties of Decomposition

student-no $\rightarrow$ instructor
instructor $\rightarrow$ office
student-no $\rightarrow$ office

subject(student-no, instructor, office)

subject1(student-no, instructor)   subject2(student-no, office)

F_1 = student-no $\rightarrow$ instructor
F_2 = student-no $\rightarrow$ office

- Does $F_1 \cup F_2$ imply all of F?

Boyce-Codd Normal Form

- A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies 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 (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- When we have a FD in the form $\alpha \rightarrow \beta$ on a relation R that violates BCFN we will decompose the schema into several sub-schemas.
- The resulting relations themselves are not guaranteed to be in BCNF. We may need to further decompose the resulting relations

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C\}$
- *Check whether R is in BCNF or not?*
  - *No, R* is not in BCNF
- Decomposition $R_1 = (A, B)$, $R_2 = (B, C)$
- After decomposing R to R_1 and R_2, you need to check the following decomposition properties.
  - Attribute Preserving
  - Lossless-join decomposition
  - Dependency preserving
  - R_1 and R_2 in BCNF
- If any one of the above doesn't satisfy then the decomposition is not correct. Need to decompose the schema in different way

Testing for BCNF

- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
  1. compute α^+ (the attribute closure of α), and
  2. 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* with a given set of functional dependencies F 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^+.
  - We can show that 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.

Testing for BCNF

- However, using only F is incorrect when testing a relation in a decomposition of R
  - E.g. Consider $R\ (A, B, C, D)$, with

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

    - Decompose R into $R_1(A,B)$ and $R_2(A,C,D)$
    - F1 = { A →B}, F2 = { ∅ }
    - Neither of the dependencies in F contain only attributes from (A,C,D) so we might be mislead into thinking R_2 satisfies BCNF.
    - In fact, dependency $A \rightarrow C$ in F^+ shows R_2 is not in BCNF.

BCNF Example

- State why the following relation is not in BCNF and decompose it so that the resulting tables are in BCNF

  Movie(title, year, stuidoName, president, presAdd)

  - **title, year → studioName**
  - **studioName → president**
  - **president → presAdd**

- **Movie1(title, year, stuidoName) , Check in BCNF? (Yes)**
- **Movie2(stuidoName, president, presAdd), Check in BCNF?(Yes)**

BCNF and Dependency Preservation

- It is not always possible to get a BCNF decomposition that is dependency preserving

- ***$R = (J, K, L)$***
  $F = \{JK \rightarrow L, L \rightarrow K\}$

- R is not in BCNF
- Any decomposition of R will fail to preserve

$$JK \rightarrow L$$

Third Normal Form: 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.
  - Allows some redundancy (with resultant problems; we will see examples later)
  - But FDs can be checked on individual relations without computing a join.
  - There is always a lossless-join, dependency-preserving decomposition into 3NF.

Third Normal Form(3NF)

- A relation schema *R* is in third normal form (3NF) if for all $\alpha \rightarrow \beta$ in F^+, at least one of the following holds:
  - **$\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)**
  - **α is a superkey for *R***
  - **Each attribute *A* in $\beta - \alpha$ is contained in a candidate key for *R*.** *That is, A is a member of a candidate key.*

  (NOTE: each attribute may be in a different candidate key)

- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold).
- Third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later).

Third Normal Form(3NF)

- **Example**
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L, L \rightarrow K\}$
  - Two candidate keys: JK and JL
  - R is in 3NF

    $JK \rightarrow L$ — JK is a superkey
    $L \rightarrow K$ — K is contained in a candidate key of R
  - BCNF decomposition has (JL) and (LK)
    - Testing for $JK \rightarrow L$ requires a join
- There is some redundancy in this schema

Testing for 3NF

- **Optimization:** Need to check only FDs in *F*, need not check all FDs in F^+.
- Use attribute closure to check, for each dependency $\alpha \rightarrow \beta$, if α is a superkey.
- If α is not a superkey, we have to verify if each attribute in β is contained in a candidate key of *R*
  - this test is rather more expensive, since it involve finding candidate keys

Comparison of BCNF and 3NF

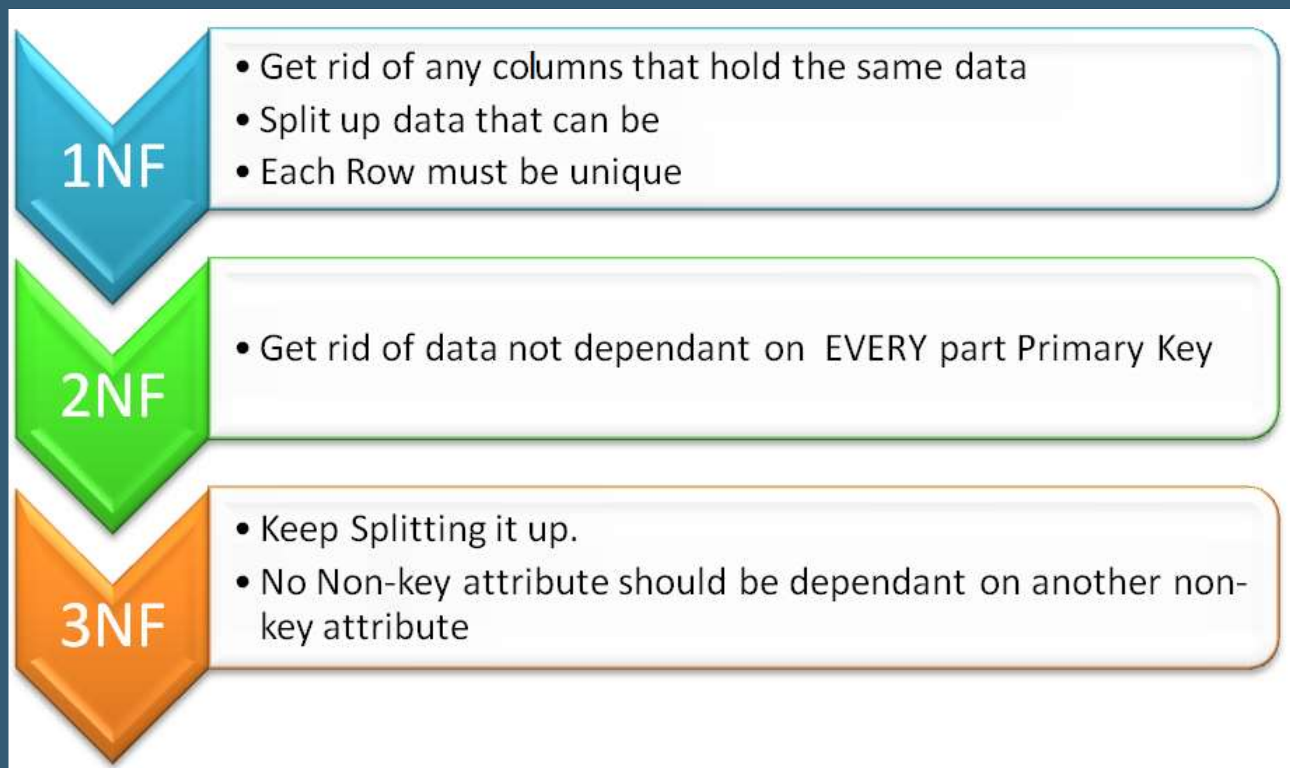
- It is always possible to decompose a relation into relations in 3NF and
  - attribute preserved
  - the decomposition is lossless
  - the dependencies are preserved

- It is always possible to decompose a relation into relations in BCNF and
  - attribute preserved
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

Design Goals

- Goal for a relational database design is:
  - Boyce-Codd Normal Form (BCNF).
  - Lossless join.
  - Attribute preservation
  - Dependency preservation.
- If we cannot achieve this, we accept one of:
  - Lack of dependency preservation (need to perform join)
  - Redundancy due to use of 3NF

Summary


1NF
• Get rid of any columns that hold the same data
• Split up data that can be
• Each Row must be unique
2NF
• Get rid of data not dependant on EVERY part Primary Key
3NF
• Keep Splitting it up.
• No Non-key attribute should be dependant on another non-key attribute

Thank You for your Attention