THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

# 5a. Functional Dependencies

CSCI 2541W
Database Systems & Team Projects

Gabe

Slides adapted from Profs. Bhagi Narahari, Tim Wood, and Rahul Simha, and book by Silberschatz, Korth, and Sudarshan

Good and Bad Schemas

**Functional Dependencies**

**Normal Forms based on Functional Dependencies**

# Normal Form Examples: 1NF

**1NF:** Attributes should be atomic and tables should have no repeating groups

| Customer ID | First Name | Surname | Telephone Number |
|---|---|---|---|
| 123 | Pooja | Singh | 555-861-2025, 192-122-1111 |
| 456 | San | Zhang | (555) 403-1659 Ext. 53; 182-929-2929 |
| 789 | John | Doe | 555-808-9633 |

**BAD:** One cell should not have multiple values in it!

| Customer ID | First Name | Surname | TNumber1 | TNumber2 |
|---|---|---|---|---|
| 123 | Pooja | Singh | 555-861-2025 | 192-122-1111 |
| 456 | San | Zhang | (555) 403-1659 Ext. 53 | 182-929-2929 |
| 789 | John | Doe | 555-808-9633 | |

**BAD:** Should not have repeating columns!

# Normal Form Examples: 2NF

**2NF:** No value in a table should be dependent on only **part** of a key that uniquely identifies a row

1NF

| Customer ID | First Name | Surname | Telephone Number |
|---|---|---|---|
| 123 | Pooja | Singh | 555-861-2025 |
| 123 | Pooja | Singh | 192-122-1111 |
| 456 | San | Zhang | 182-929-2929 |
| 456 | San | Zhang | (555) 403-1659 Ext. 53 |
| 789 | John | Zhang | 555-808-9633 |

**BAD:** First/Last name depend on only Customer ID

**VS**

2NF

| Customer ID | First Name | Surname |
|---|---|---|
| 123 | Pooja | Singh |
| 456 | San | Zhang |
| 789 | John | Zhang |

| Customer ID | Telephone Number |
|---|---|
| 123 | 555-861-2025 |
| 123 | 192-122-1111 |
| 456 | (555) 403-1659 Ext. 53 |
| 456 | 182-929-2929 |
| 789 | 555-808-9633 |

**Meets 2NF**

# Normal Form Examples: 3NF

**3NF:** No value should be able to be dependent on another non-key field

| <u>Customer ID</u> | First Name | Surname | Birthday | Age | Fav Color |
|---|---|---|---|---|---|
| 123 | Pooja | Singh | 1/4/1984 | 37 | Blue |
| 456 | San | Zhang | 3/15/2001 | 19 | Blue |
| 789 | John | Zhang | 11/12/2006 | 14 | Buff |

**BAD:** Age is based on Birthday (non-key)

| <u>Tournament</u> | <u>Year</u> | Winner | Winner's Birthplace |
|---|---|---|---|
| Indiana Invitational | 1998 | Al Fredrickson | Ohio |
| Cleveland Open | 1999 | Bob Albertson | New York |
| Des Moines Masters | 1999 | Al Fredrickson | Ohio |
| Indiana Invitational | 1999 | Chip Masterson | Kentucky |

**BAD:** Birthplace based on Winner (non-key)

# Summary

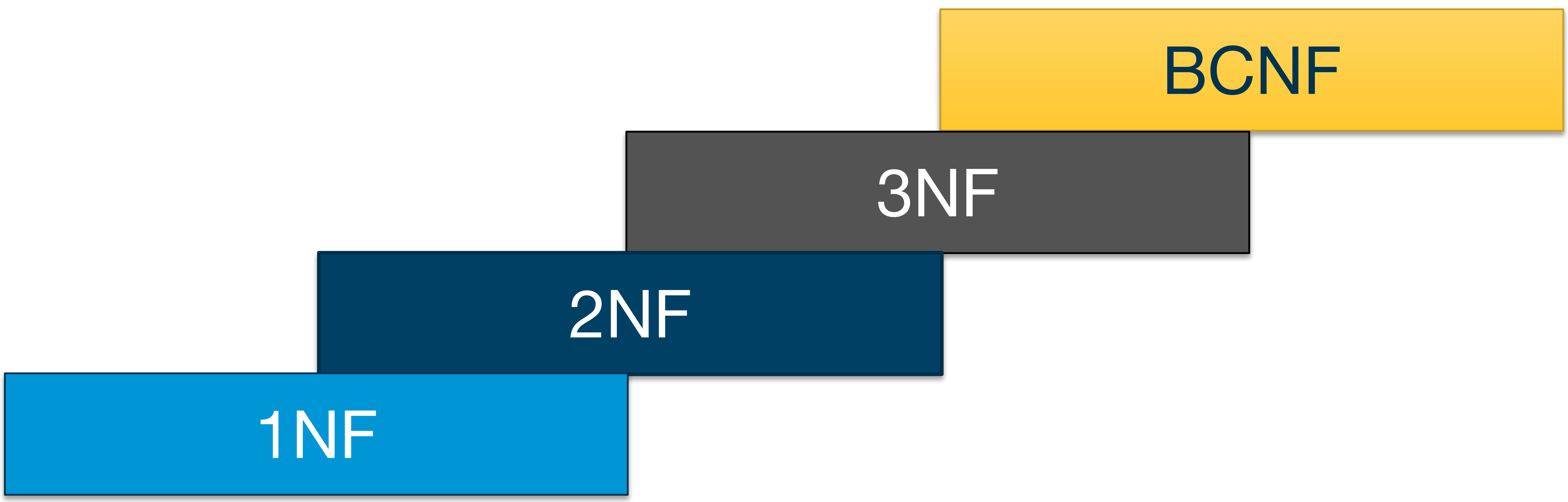
1NF: ensures atomicity of cells and prevents repetition of identical column types

2NF: prevents data across rows

3NF: prevents repetition of data within+across rows

# Dependencies

How can we **formally represent dependencies** between Attributes in a Relation?



# Functional Dependencies

Use **functional dependencies!** (abbreviated **FD**)

We say a set of attributes **X** functionally determines an attribute **Y** if *given the values of* ***X*** *we always know the only possible value of* ***Y***.

- Notation: **X** → **Y**
- X **functionally determines** Y
- Y is **functionally dependent** on X

Example:

- GWID → Name
- {GWID, CourseID, Semester, Year} → Grade

# Sets of Functional Dependencies

Some more functional dependencies

- {GWID} → {NAME, ADDRESS, MAJOR}
- {MAJOR} → {DEPT_NAME, DEPT_CHAIR}

From above dependencies, we can infer

- {GWID} → {DEPT_NAME, DEPT_CHAIR}

We can do math on functional dependencies!

A functional dependency "holds" if it must be true for all legal relations

# Functional Dependency Ops

**Armstrong's Axioms:** where A, B, C are sets of attributes

- **Reflexive rule**: if B $\subseteq$ A, then A $\rightarrow$ B (if B is subset of A)
- **Augmentation rule**: if A $\rightarrow$ B, then C $\cup$ A $\rightarrow$ C $\cup$ B
- **Transitivity rule**: if A $\rightarrow$ B, and B $\rightarrow$ C, then A $\rightarrow$ C

These rules are

- Sound and complete — generate all functional dependencies that hold.

{GWID} $\rightarrow$ {Name, Address, Major}

{Major} $\rightarrow$ {Dept_Name, Dept_Chair}

{GWID, CourseID, Semester, Year} $\rightarrow$ Grade

# Functional Dependency Ops

**Armstrong's Axioms:** where A, B, C are sets of attributes

- **Reflexive rule**: if B ⊆ A, then A → B (if B is subset of A)
- **Augmentation rule**: if A → B, then C ∪ A → C ∪ B
- **Transitivity rule**: if A → B, and B → C, then A → C

*Given:*

{GWID} → {Name, Address, Major}

{Major} → {Dept_Name, Dept_Chair}

{GWID, CourseID, Semester, Year} → Grade

R:{GWID, CourseID, Semester, Year} → {GWID, Year}

A:{Major, Name, Address} →

{Dept_Name, Dept_Chair, Name, Address}

T:{GWID} → {Name, Address, Major} →

{Dept_Name, Dept_Chair, Name, Address}

# Functional Dependency Ops

**Armstrong's Axioms:** where A, B, C are sets of attributes

- **Reflexive rule**: if $B \subseteq A$, then $A \rightarrow B$ (if B is subset of A)
- **Augmentation rule**: if $A \rightarrow B$, then $C \cup A \rightarrow C \cup B$
- **Transitivity rule**: if $A \rightarrow B$, and $B \rightarrow C$, then $A \rightarrow C$

These rules are

- Sound and complete — generate all functional dependencies that hold.

Bonus rules to make life easier:

- **Union rule**: If $A \rightarrow B$ holds and $A \rightarrow C$ holds, then $A \rightarrow B \cup C$ holds.
- **Decomposition rule**: If $A \rightarrow B \cup C$ holds, then $A \rightarrow B$ holds and $A \rightarrow C$ holds.
- **Pseudotransitivity rule**:If $A \rightarrow B$ holds and $C \cup B \rightarrow D$ holds, then $A \cup C \rightarrow D$ holds.

# Functional Dependency Ops

**Armstrong's Axioms:** where A, B, C are sets of attributes

- **Reflexive rule**: if B ⊆
- **Augmentation rule**:
- **Transitivity rule**: if A

These rules are

- Sound and complete —
  hold.

From now on:
will *shorthand* B ∪ C as BC

Bonus rules to make life easier:

- **Union rule**: If A → B holds and A → C holds, then A → B ∪ C holds.
- **Decomposition rule**: If A → B ∪ C holds, then A → B holds and A → C holds.
- **Pseudotransitivity rule**: If A → B holds and C ∪ B → D holds, then A ∪ C → D holds.

# Definition: Closure of a Set of FD's

Defn. Let F be a set of FD's.

Its **closure**, **F+**, is the set of all FD's:

**{$X \rightarrow Y$ | $X \rightarrow Y$ is derivable from F by Armstrong's Axioms}**

**Two sets of dependencies F and G are equivalent if F+=G+**

- i.e., their closures are equal
- i.e., the same sets of FDs can be inferred from each

# Example Closure

What FDs can we infer?

$R = (A, B, C, G, H, I)$

$F = \{ A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H\}$

**Reflexive rule**: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

**Augmentation rule**: if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$

**Transitivity rule**: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

# Example Closure

$R = (A, B, C, G, H, I)$

$F = \{ A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H\}$

**Reflexive rule**: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

**Augmentation rule**: if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$

**Transitivity rule**: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

A few members of F+ include:

- $A \rightarrow H$
- $AG \rightarrow I$
- $CG \rightarrow HI$

# Example Closure

R = (A, B, C, G, H, I)

F = { $A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H$}

**Reflexive rule**: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

**Augmentation rule**: if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$

**Transitivity rule**: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

A few members of F+ include:

- $A \rightarrow H$
  by transitivity from $A \rightarrow B$ and $B \rightarrow H$
- $AG \rightarrow I$
- $CG \rightarrow HI$

# Example Closure

$R = (A, B, C, G, H, I)$

$F = \{ A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H\}$

**Reflexive rule**: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

**Augmentation rule**: if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$

**Transitivity rule**: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

A few members of F+ include:

- $A \rightarrow H$
  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 $CG \rightarrow I$
- $CG \rightarrow HI$

# Example Closure

$R = (A, B, C, G, H, I)$

$F = \{ A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H\}$

**Reflexive rule**: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

**Augmentation rule**: if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$

**Transitivity rule**: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

A few members of F+ include:

- $A \rightarrow H$
  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 $CG \rightarrow I$
- $CG \rightarrow HI$
  by augmenting $CG \rightarrow I$ to infer $CG \rightarrow CGI$ (note $CGCG \rightarrow CGI$ is $CG \rightarrow CGI$),
  and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$,
  and then transitivity ($CG \rightarrow CGI \rightarrow HI$).
  This derives the *Union* rule!

# Functional Dependencies and Keys

A Candidate Key is a minimal set of attributes which are sufficient to uniquely identify each tuple in a relation

- All other attributes must be functionally dependent on the set of attributes that make up the Candidate Key.

Thus a candidate key must be a minimal set of attributes which can appear on the left hand side of functional dependencies, but will produce a closure that includes all other attributes on the right hand side

$$\{A, B\} \rightarrow \{C, D, E\}$$

# Functional Dependencies and Keys

A candidate key must be a minimal set of attributes which can appear on the left hand side of functional dependencies, but will produce a closure that includes all other attributes on the right hand side

What is a candidate key for R?

$R = (A, B, C, G, H, I)$

$$
\begin{aligned}
F = \{ & A \rightarrow B \\
& A \rightarrow C \\
& CG \rightarrow H \\
& CG \rightarrow I \\
& B \rightarrow H \}
\end{aligned}
$$

# Functional Dependencies and Keys

A candidate key must be a minimal set of attributes which can appear on the left hand side of functional dependencies, but will produce a closure that includes all other attributes on the right hand side

What is a candidate key for R?

R = (A, B, C, G, H, I)

$$
\begin{aligned}
F = \{ & A \rightarrow B \\
& A \rightarrow C \\
& CG \rightarrow H \\
& CG \rightarrow I \\
& B \rightarrow H \}
\end{aligned}
$$

Candidate Key: (A, G)
Non-Prime Attribs:
(B,C,G,H,I)

# Functional De

A candidate key must b
appear on the left hand
produce a closure that
hand side

What is a candidate key

R = (A, B, C, G, H, I)

F = { $A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H$}

**<u>Pseudotransitivity rule</u>:**

If $A \rightarrow B$ holds and $CB \rightarrow D$ holds, then $AC \rightarrow D$ holds.

<u>How?</u>

$A \rightarrow C$ and $CG \rightarrow H$

Augmentation: $AG \rightarrow CG$

Transitivity: $AG \rightarrow CG \rightarrow H$

Candidate Key: (A, G)
Non-Prime Attribs:
(B,C,G,H,I)

Good and Bad Schemas

**Functional Dependencies**

**Normal Forms based on Functional Dependencies**

# Redefining 2NF

No value in a table should be dependent on only **part** of a key that uniquely identifies a row

Using Functional Dependencies and Closures lets us more precisely define our Normal Forms

**Second Normal Form**: For every **X** → **A** that holds over relationship schema **R**, where A is a non-prime attribute (i.e., A is not an attribute in any candidate key)

1. either **A** ∈ **X** (it is trivial), or
2. **X** is a superkey for **R**, or
3. **X** is transitively dependent on a super key R

*Easier to think of the opposite*: There cannot be **X** → **A** where X is a partial candidate key for R

– Says nothing about non-prime to non-prime dependencies!

# 2NF Violations

| ID | First Name | Surname | Telephone Number |
|---|---|---|---|
| 123 | Pooja | Singh | 555-861-2025 |
| 123 | Pooja | Singh | 192-122-1111 |
| 456 | San | Zhang | 182-929-2929 |
| 456 | San | Zhang | (555) 403-1659 Ext. 53 |
| 789 | John | Zhang | 555-808-9633 |

| Tournament | Year | Winner | Winner's Birthplace |
|---|---|---|---|
| Indiana Invitational | 1998 | Al Fredrickson | Ohio |
| Cleveland Open | 1999 | Bob Albertson | New York |
| Des Moines Masters | 1999 | Al Fredrickson | Ohio |
| Indiana Invitational | 1999 | Chip Masterson | Kentucky |

# 2NF Violations

| <u>ID</u> | First Name | Surname | <u>Telephone Number</u> |
|---|---|---|---|
| 123 | Pooja | Singh | 555-861-2025 |
| 123 | Pooja | Singh | 192-122-1111 |
| 456 | San | Zhang | 182-929-2929 |
| 456 | San | Zhang | (555) 403-1659 Ext. 53 |
| 789 | John | Zhang | 555-808-9633 |

**ID** -> {**First Name, LastName**} Violates 2NF since **ID** is a partial Candidate Key

| <u>Tournament</u> | <u>Year</u> | Winner | Winner's Birthplace |
|---|---|---|---|
| Indiana Invitational | 1998 | Al Fredrickson | Ohio |
| Cleveland Open | 1999 | Bob Albertson | New York |
| Des Moines Masters | 1999 | Al Fredrickson | Ohio |
| Indiana Invitational | 1999 | Chip Masterson | Kentucky |

No 2NF violation

# Redefining 3NF

No value should be able to be dependent on another non-key field

Third Normal Form (3NF): For every $\mathbf{X} \rightarrow \mathbf{A}$ that holds over relationship schema **R**,

1. either $\mathbf{A} \in \mathbf{X}$ (it is trivial), or
2. **X** is a superkey for **R**, or
3. **A** is a member of some key for **R**

*Easier to think of*: **X** must be a full candidate key, unless A itself is a part of a candidate key

"Every non-key attribute must provide a fact about the Key, the whole Key, and nothing but the Key… so help me Codd"

# 3NF Violations

| <u>Customer ID</u> | First Name | Surname | Birthday | Age | Fav Color |
|---|---|---|---|---|---|
| 123 | Pooja | Singh | 1/4/1984 | 37 | Blue |
| 456 | San | Zhang | 3/15/2001 | 19 | Blue |
| 789 | John | Zhang | 11/12/2006 | 14 | Buff |

| <u>Tournament</u> | <u>Year</u> | Winner | Winner's Birthplace |
|---|---|---|---|
| Indiana Invitational | 1998 | Al Fredrickson | Ohio |
| Cleveland Open | 1999 | Bob Albertson | New York |
| Des Moines Masters | 1999 | Al Fredrickson | Ohio |
| Indiana Invitational | 1999 | Chip Masterson | Kentucky |

# 3NF Violations

C ->
F,S,B,A,F
B->A

| <u>Customer ID</u> | First Name | Surname | Birthday | Age | Fav Color |
|---|---|---|---|---|---|
| 123 | Pooja | Singh | 1/4/1984 | 37 | Blue |
| 456 | San | Zhang | 3/15/2001 | 19 | Blue |
| 789 | John | Zhang | 11/12/2006 | 14 | Buff |

**Birthday->Age** holds, but **Birthday** is not a superkey

T,Y -> W, WB
W -> WB

| <u>Tournament</u> | <u>Year</u> | Winner | Winner's Birthplace |
|---|---|---|---|
| Indiana Invitational | 1998 | Al Fredrickson | Ohio |
| Cleveland Open | 1999 | Bob Albertson | New York |
| Des Moines Masters | 1999 | Al Fredrickson | Ohio |
| Indiana Invitational | 1999 | Chip Masterson | Kentucky |

**Winner -> Birthplace** holds, but **Winner** is not a superkey

# Normal Forms 1-3

**1NF**: Attributes should be atomic and tables should have no repeating groups

- *Prevents messiness within a cell and repetition of rows*

**2NF**: There cannot be **X** → **A** where X is a partial candidate key for R

- Doesn't forbid non-prime to non-prime dependencies
- *Prevents repetition of cells across rows*

**3NF**: There cannot be **X** → **A** where X is not a full candidate key for R (unless A is a Key)

- Only allows dependencies on Keys
- *Prevents repetition of data within a row*

Good and Bad Schemas

**Functional Dependencies**

**Even more normal forms!**

# Normal Forms 1-3

**1NF**: Attributes should be atomic and tables should have no repeating groups

- *Prevents messiness within a cell and repetition of rows*

**2NF**: There cannot be **X** → **A** where X is a partial candidate key for R

- Doesn’t forbid non-prime to non-prime dependencies
- *Prevents repetition of cells across rows*

**3NF**: There cannot be **X** → **A** where X is not a **full** candidate key for R (unless A is a Key)

- Only allows dependencies on Keys
- *Prevents repetition of data within a row*

# Normal Forms 1-3

**1NF**: Attributes should be atomic and tables should have no re

- *Prevent* ... *of rows*

**2NF**: Ther ... artial candidate

- Doesn'... encies
- *Prevent*

**3NF**: Ther ... a **full** candidate

- Only all
- *Prevents repetition of data within a row*

Preview of the (sometimes unrealistic) goal:

$$R = (\underline{A, B, C}, D, E, F)$$

has FD+:

$$ABC \rightarrow DEF$$

# Normal Form

Normal form reference:

- 2NF: Cannot have partial Key on left hand side (LHS)
- 3NF: Meet 2NF and LHS must be full Candidate Key or RHS must be a key

| **<u>ID</u>** | **First name** | **<u>Cid</u>** | **Subj** | **Num** | **Grade** |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

**Functional Dependencies**

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

What normal form is this?

# Normal Form

Normal form reference:

- 2NF: Cannot have partial Key on left hand side (LHS)
- 3NF: Meet 2NF and LHS must be full Candidate Key or RHS must be a key

| <u>**ID**</u> | **First name** | <u>**Cid**</u> | **Subj** | **Num** | **Grade** |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

**Functional Dependencies**

$ID \rightarrow FirstName$ partial key ID violates 2NF!

$ID, Cid \rightarrow Num, Grade$

$Num \rightarrow Subj$

non-prime LHS would also violate 3NF!

Only meets 1NF

# How to Judge Decomposition?

R1 R2

| ID | First name | Cid | Subj | Num | Grade |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

Lossless Decomposition test:

– **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**
– **(R1 ∩ R2) → R1 – R2**
– **(R1 ∩ R2) → R2 – R1**

# Lossless Decomposition

R1 R2

| ID | First name | Cid | Subj | Num | Grade |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

R1 ∩ R2 = CID

R1 - R2 = ID, First

R2 - R1 = Subj, Num, Grade

**Lossless?**

Lossless Decomposition test:

– **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**

– **(R1 ∩ R2) → R1 – R2**

– **(R1 ∩ R2) → R2 – R1**

# Lossless Decomposition

R1 R2

| <u>ID</u> | First name | <u>Cid</u> | Subj | Num | Grade |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550 | | | |

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

R1 ∩ R2 = CID

R1 - R2 = ID, First

R2 - R1 = Subj, Num, Grade

*Not true:* CID → First

*Not true:* CID → Grade

In fact: CID → *nothing*

**Not lossless!**

Lossless Decomposition

– **R1**, **R2** is a lossless join
  **F iff** at least one of the fo

– **(R1 ∩ R2) → R1 – R2**

– **(R1 ∩ R2) → R2 – R1**

# Dependency Preservation

We also must maintain dependences

After decomposition from **R** to **R1 … Rn**, the closure of FDs of all **R1…Rn** must be equivalent to that of **R**

**R1 = <u>ID</u>,** FirstName, CID
**R2 = <u>CID</u>,** Sub, Num, Grade

or

**R3 = <u>ID</u>,** FirstName
**R4 = <u>ID, CID</u>,** Sub, Num, Grade

ID → FirstName

ID, Cid → Num, Grade

Num → Subj

# Dependency Preservation

We also must maintain dependences

After decomposition from **R** to **R1 … Rn**, the closure of FDs of all **R1…Rn** must be equivalent to that of **R**

**R1 = <u>ID</u>,** FirstName, CID

**R2 = <u>CID</u>,** Sub, Num, Grade

or

R1,R2 will lose the FD:
ID,CID -> Num, Grade

ID → FirstName

ID, Cid → Num, Grade

Num → Subj

**R3 = <u>ID</u>,** FirstName

**R4 = <u>ID, CID</u>,** Sub, Num, Grade

R3,R4 will maintain all FDs (why Sub?)

# 3NF

It is **always possible** to decompose a relation R into a set of relations R1…Rn which is **dependency preserving** and **lossless** that is in 3NF

3NF is the baseline for acceptable DB normalization in practice!
Required!

but 3NF is not perfect…

# When does 3NF fail?

Suppose we want to store addresses:

ADDR_INFO( **<u>CITY</u>**, **<u>ADDRESS</u>**, **ZIP**)

{CITY, ADDRESS} → ZIP

{ZIP} → {CITY}

Meets 3NF since LHS is a full Key **or** RHS is a Key

3NF: There cannot be **X** → **A** where X is not a full candidate key for R **(unless A is a Key)**

# When does 3NF fail?

ADDR_INFO( **<u>CITY</u>**, **<u>ADDRESS</u>**, **ZIP**)

{CITY, ADDRESS} → ZIP

{ZIP} → {CITY}

| City | Address | Zip |
|---|---|---|
| Washington | 800 22nd St NW | 20052 |
| Washington | 1600 Pennsylvania Avenue NW | 20050 |
| | | |
| | | |

3NF: There cannot be **X** → **A** where X is not a full candidate key for R **(unless A is a Key)**

# When does 3NF fail?

ADDR_INFO( **CITY**, **ADDRESS**, **ZIP**)

{CITY, ADDRESS} → ZIP

{ZIP} → {CITY}

| City | Address | Zip |
|---|---|---|
| Washington | 800 22nd St NW | 20052 |
| Washington | 1600 Pennsylvania Avenue NW | 20050 |
| Philadelphia | 101 South Street | 20050 |
| | | |

3NF does not prevent insertion/update of tuples which violate our FDs!

# 3NF vs BCNF

Third Normal Form (3NF): For every **X** → **A** that holds over relationship schema **R**,

1. either **A** ∈ **X** (it is trivial), or
2. **X** is a superkey for **R**, or
3. **A** is a member of some key for **R**

Option 3 can result in update anomalies!

**Boyce-Codd Normal Form** (BCNF) resolves this issue:

For every **X** → **A** that holds over relationship schema **R**,

1. either **A** ∈ **X** (it is trivial), or
2. **X** is a superkey for **R**

# 3NF vs BCNF

Third Normal Form (3NF): For every **X** → **A** that holds

over relati

1. either **A**
2. **X** is a su
3. **A** is a m

Option 3 c

**Boyce-Co** s this issue:

For every schema **R**,

1. either **A**
2. **X** is a superkey for **R**

Recall the (sometimes unrealistic) goal:

R = (<u>A, B, C</u>, D, E, F)

has FD+:

ABC → DEF

# BCNF

BCNF is stricter than 3NF

- If a relation is in BCNF, it is also in 3NF;
- if it is not in 3NF, it is not in BCNF

Note:

- There are polynomial time algorithms **guaranteed to provide a lossless, dependency preserving decomposition** into 3NF
- **but** a **dependency preserving** decomposition into BCNF **may not exist**, and no polynomial time algorithm for **lossless decomposition** is known.

# Normalization Summary

**Functional Dependencies**: Capture the dependencies between attributes

**Normalization**: Provides a schema that ensures functional dependencies will be kept consistent, without losing data

**Normal Forms**: Try to achieve BCNF, but 3NF is OK in some cases (1NF/2NF -> bad design!)

# 2NF vs 3NF vs BCNF

Second Normal Form (2NF): For every **X** $\longrightarrow$ **A** that holds over relationship schema **R**,

1. If **A** is a non-prime attribute, then **X** cannot be a partial Candidate Key

Third Normal Form (3NF): For every **X** $\longrightarrow$ **A** that holds over relationship schema **R**,

1. either **A** $\in$ **X** (it is trivial), or
2. **X** is a superkey for **R**, or
3. **A** is a member of some key for **R**

Option 3 can result in update anomalies!

**Boyce-Codd Normal Form** (BCNF) resolves this issue:

For every **X** $\longrightarrow$ **A** that holds over relationship schema **R**,

1. either **A** $\in$ **X** (it is trivial), or
2. **X** is a superkey for **R**