Normalization

Carsten Schürmann

Database Design

Quality considerations

- Integrity of the data
- Database performance

How can we affect quality?

- Choice of relations (how many and which)
- Choice of attributes
- Choice of integrity constraints

Example

Students(cpr, courseID, room, grade, name, address)

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Redundancy Problems

Update Anomalies

- If the room number changes, we need to make sure that we change all students records.

Insert Anomalies

- May not be possible to add a student unless they're enrolled in a course.

Delete Anomalies

- If all the students enrolled in a course are deleted, then we lose the room number.

Decomposition

Students(cpr, name, address)

cpr	name	address
140298-1234	Jesper	Copenhagen
041297-5367	Nikoline	Aarhus
151197-2352	Claus	Dragør
050596-1142	Martin	Copenhagen

Courses(cpr, courseID, grade)

cpr	courseID	grade
140298-1234	SIDD	12
041297-5367	SIDD	10
151197-2352	CRITSYS	12
050596-1142	CRITSYS	7

Rooms(courseID, room)

courseID	room
SIDD	AUD1
SIDD	AUD1
CRITSYS	2A12
CRITSYS	3A01

Overview

- Functional Dependencies
- Armstrong's Axioms
- Closures
- Canonical Cover
- Decomposition
- Normal Forms

Functional Dependency (FD)

A constraint is

- Part of the schema
- Defines a valid instance

Definition: Let A, B be attributes. We write $A \to B$

read as "A functionally implies B"

iff for all tuples t_1, t_2.

$t_1[A] = t_2[A]$ implies that $t_1[B] = t_2[B]$

FD Examples

1. cpr → name — YES
2. cpr → address — YES
3. name → address — NO

cpr	name	address
140298-1234	Jesper	Copenhagen
041297-5367	Nikoline	Aarhus
151197-2352	Claus	Dragør
050596-1142	Martin	Copenhagen

You can check if an FD is violated on an instance

You cannot prove that an FD is part of a schema

More on FDs

General case: We write $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$

iff for all tuples t_1, t_2.

$t_1[A_1] = t_2[A_1]$ and … $t_1[A_n] = t_2[A_n]$

implies that

$t_1[B_1] = t_2[B_1]$ and … $t_1[B_m] = t_2[B_m]$

Note $A \rightarrow B$ and $A \rightarrow C$ implies $A \rightarrow BC$

But $AB \rightarrow C$ does neither imply $A \rightarrow C$ nor $B \rightarrow C$

FDs SQL assertions

1. cpr → name
2. cpr → address

```
CREATE ASSERTION student-name-address
   CHECK (NOT EXISTS
      (SELECT *
       FROM students AS s1, students AS s2
       WHERE s1.cpr= s2.cpr
       AND ((s1.name <> s2.name
       OR (s1.address <> s2.address)))
```

Why Should I Care?

- FDs seem important, but what can we actually do with them?
- They allow us to decide whether a database design is correct.

Implied Dependencies

1. cpr → name address [Provided]
2. cpr courseID → grade [Provided]

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

3. cpr courseID → grade name address [Implied]
4. cpr courseID → cpr [Implied]

\$100 Question

Given a set of FDs, how do we decide if some other FD is implied?

Compute the closure using Armstrong's axioms:

Reflexivity: If $X \supseteq Y$ then $X \rightarrow Y$

Augmentation: If $X \rightarrow Y$ then $X\,Z \rightarrow Y\,Z$ for all Z

Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Reflexivity

If $X \supseteq Y$ then $X \to Y$

Example: cpr courseID → cpr

Definition:

Such FDs is also called trivial or unavoidable

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Augmentation

If $X \rightarrow Y$ then $X\,Z \rightarrow Y\,Z$ for all Z

Example: If cpr $\rightarrow$ name

then cpr grade $\rightarrow$ name grade

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Transitivity

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Example: If cpr → address and address → country

then cpr → country

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Addition Rules

Union:

- If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Decomposition:

- If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Pseudo-transitivity:

- If $X \rightarrow Y$ and $YW \rightarrow Z$ then $XW \rightarrow Z$

Closures

Definition: Given a set F of FDs. The closure F^+ is defined as the set of all implied FDs.

1. cpr → name address [provided]
2. cpr courseID → grade [provided]
3. cpr name → cpr [by reflexivity]
4. cpr → name [by decomposition]
5. cpr → address [by decomposition]
6. cpr grade → name grade [by augmentation]
7. cpr courseID → name grade [by transitivity]

…

Algorithm to Check if an FD is Entailed

With closure we can find all FD's easily.

Definition: Attribute closure

For a given attribute X, the attribute closure X^+ is the set of all attributes such that $X \rightarrow A$ can be inferred using the Armstrong Axioms.

Algorithm:

To check if $X \rightarrow A$, compute X^+ and check if $A \in X^+$

Again, Why Should I Care?

Does the closure tell you, which constraints to add to the database?

Yes, but it is expensive to check!

Can’t we do get a away with checking fewer FDs?

Yes, by checking only the canonical cover!

Canonical Cover

Definition: Given a set F of FDs. The closure FC is defined as the minimal set of FDs.

1. cpr → name address
2. cpr courseID → grade
3. cpr name → cpr
4. cpr → name
5. cpr → address
6. cpr grade → name grade
7. cpr courseID → name grade

Canonical Cover

Definition

Let F be a set of FDs. The the canonical cover FC satisfies the following properties.

1. The RHS of every FD in FC is a single attribute
2. The closures of F and FC are equal: $F^+ = FC^+$
3. FC is minimal

Canonical Cover

Algorithm:

1. Input a set of FDs
2. For each FD in this set

   Drop extraneous attributes and redundant FDs until the remaining FDs have a single attribute on the left

Example

1. $A\ B \rightarrow C$
2. $A \rightarrow BC$
3. $B \rightarrow C$
4. $A \rightarrow B$

Split 2.

Example (Step 2)

1. A B $\rightarrow$ C
2. A $\rightarrow$ B
3. A $\rightarrow$ C
4. B $\rightarrow$ C
5. A $\rightarrow$ B

Eliminate 2 (because it is the same as 5.)

Example (Step 2)

1. $A\ B \rightarrow C$
2. $A \rightarrow C$
3. $B \rightarrow C$
4. $A \rightarrow B$

Eliminate 2 (because it is the same as 5.)

Example (Step 3)

1. A B → C
2. A → C
3. B → C
4. A → B

Eliminate 2. (because it is implied by 3. and 4. because of transitivity)

Example (Step 4)

1. A B $\rightarrow$ C
2. B $\rightarrow$ C
3. A $\rightarrow$ B

Eliminate A from 1.
(because it is implied by 3.)

Example (Step 5)

1. B $\rightarrow$ C
2. B $\rightarrow$ C
3. A $\rightarrow$ B

Eliminate 1. (because it is redundant)

Example (Step 6)

1. B → C
2. A → B

Done!

This is the canonical cover

- Nothing is extraneous
- All RHS are ok
- The closures are the same.

Keys

- Super Key: Any subset of attributes that functionally determines all other attributes of the relation
- Candidate Key: A minimal super key
- Primary Key: Just a candidate key

Example: cpr, courseID, room, grade superkey

Decomposition

Students(cpr, courseID, room, grade, name, address)

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Goal: Split a relation R into a set of relations $\{R_1 \ldots R_n\}$, such that the result has good qualities.

What does good mean?

Lossless joins

We should be able to reconstruct the original instance by joining the parts together

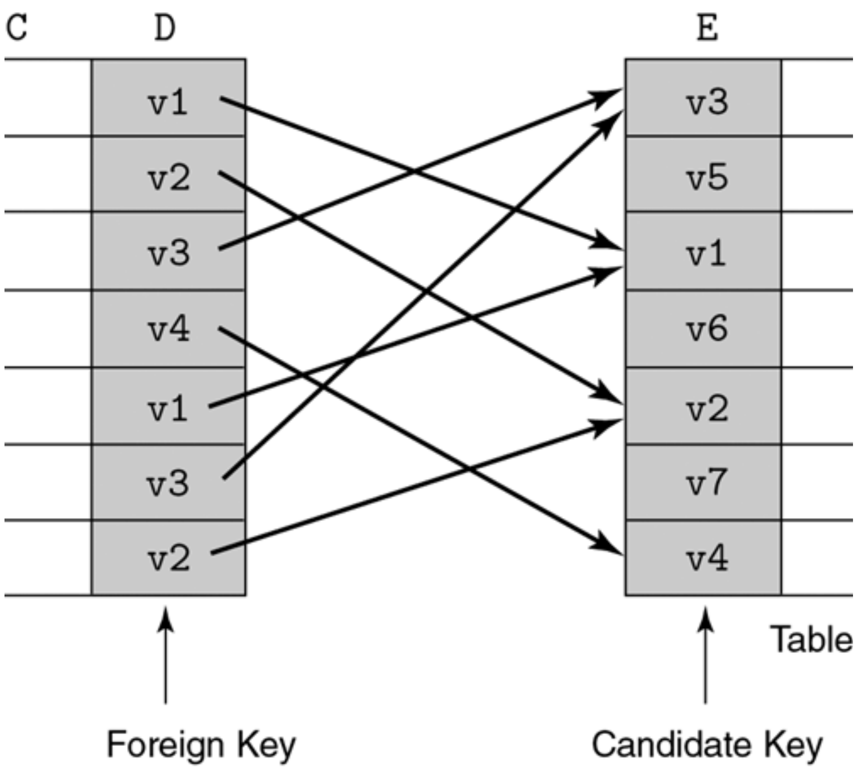
Dependency preservation

The FDs should not span multiple tables

Redundancy avoidance

Avoid unnecessary data duplication

Lossless Decomposition



Idea: Think pointers!

Split R into two relations R_1 and R_2 such that R_2's primary key is used in R_1 as foreign key.

Requirement: Reconstruct original relation by joining the parts!

Lossless Decomposition

Attempt 1: Decomposition along name address

Students(cpr, courseID, room, grade, name, address)

cpr	courseID	room	grade	name
140298-1234	SIDD	AUD1	12	Jesper
041297-5367	SIDD	AUD1	10	Nikoline
151197-2352	CRITSYS	2A12	12	Claus
050596-1142	CRITSYS	3A01	7	Martin

name	address
Jesper	Copenhagen
Nikoline	Aarhus
Claus	Dragør
Martin	Copenhagen

Bad Decomposition

What happens if Martin from Aarhus enrolls in Introduction to Databases?

cpr	courseID	room	grade	name
140298-1234	SIDD	AUD1	12	Jesper
041297-5367	SIDD	AUD1	10	Nikoline
151197-2352	CRITSYS	2A12	12	Claus
050596-1142	CRITSYS	3A01	7	Martin

name	address
Jesper	Copenhagen
Nikoline	Aarhus
Claus	Dragør
Martin	Copenhagen
Martin	Aarhus

We just broke the pointer concept!

Decomposition is not lossless!

Deep Insight for Today

Split relations along functional dependencies

Lossless Decomposition

Decomposition along: cpr → name address

Students(cpr, courseID, room, grade, name, address)

Functional Dependencies
cpr → name address
cpr courseID → grade

cpr	courseID	room	grade
140298-1234	SIDD	AUD1	12
041297-5367	SIDD	AUD1	10
151197-2352	CRITSYS	2A12	12
050596-1142	CRITSYS	3A01	7

cpr	name	address
140298-1234	Jesper	Copenhagen
041297-5367	Nikoline	Aarhus
151197-2352	Claus	Dragør
050596-1142	Martin	Copenhagen

cpr is foreign key

cpr is candidate key

Lossless Decomposition

Decomposition along: cpr courseID → grade

cpr	courseID	room	grade
140298-1234	SIDD	AUD1	12
041297-5367	SIDD	AUD1	10
151197-2352	CRITSYS	2A12	12
050596-1142	CRITSYS	3A01	7

Functional Dependencies
cpr → name address
cpr courseID → grade

cpr	courseID	room
140298-1234	SIDD	AUD1
041297-5367	SIDD	AUD1
151197-2352	CRITSYS	2A12
050596-1142	CRITSYS	3A01

cpr courseID is foreign key

cpr	courseID	grade
140298-1234	SIDD	12
041297-5367	SIDD	10
151197-2352	CRITSYS	12
050596-1142	CRITSYS	7

cpr courseID is candidate key

Lossless Decomposition

Easy to check that it is lossless:

cpr	courseID	room
140298-1234	SIDD	AUD1
041297-5367	SIDD	AUD1
151197-2352	CRITSYS	2A12
050596-1142	CRITSYS	3A01

$\bowtie$ (

cpr	courseID	grade
140298-1234	SIDD	12
041297-5367	SIDD	10
151197-2352	CRITSYS	12
050596-1142	CRITSYS	7

$\bowtie$

cpr	name	address
140298-1234	Jesper	Copenhagen
041297-5367	Nikoline	Aarhus
151197-2352	Claus	Dragør
050596-1142	Martin	Copenhagen

)

Lossless Decomposition

Property:

A decomposition of R into R_1 and R_2 is lossless if and only if

$$R_1 \cap R_2 \rightarrow R_1$$

or

$$R_1 \cap R_2 \rightarrow R_2$$

And intersecting attributes must form a super key for one of the resulting smaller relations.

Dependency Preservation

- Main Idea:

  Original FDs cannot span multiple tables.

- Why does this matter?

  It would be expensive to check (assuming that our DBMS supports ASSERTIONS).

Dependency Preservation

Loans(bname, bcity, assets, cname, loanId, amt)

bankname	city	assets	customer	loanid	amount
Østerbro	Copenhagen	9M DKK	Jensen	L-13	1000 DKK
Østerbro	Copenhagen	9M DKK	Madsen	L-23	2000 DKK
Lufthavn	Aalborg	2M DKK	Jensen	L-15	5000 DKK
Østerbro	Copenhagen	9M DKK	Tofte	L-78	3500 DKK

Functional Dependencies

bankname → city assets

loanid→ amount bankname

Dependency Presevation

bankname	city	assets	customer	loanid	amount
Østerbro	Copenhagen	9M DKK	Jensen	L-13	1000 DKK
Østerbro	Copenhagen	9M DKK	Madsen	L-23	2000 DKK
Lufthavn	Aalborg	2M DKK	Jensen	L-15	5000 DKK
Østerbro	Copenhagen	9M DKK	Tofte	L-78	3500 DKK

Functional Dependencies

bankname → city assets

loanid → amount bankname

bankname	assets	customer	loanid
Østerbro	9M DKK	Jensen	L-13
Østerbro	9M DKK	Madsen	L-23
Lufthavn	2M DKK	Jensen	L-15
Østerbro	9M DKK	Tofte	L-78

city	loanid	amount
Copenhagen	L-13	1000 DKK
Copenhagen	L-23	2000 DKK
Aalborg	L-15	5000 DKK
Copenhagen	L-78	3500 DKK

Dependency Preservation

Problem: Assertion checking very expensive

```
CREATE ASSERTION bankname-city
CHECK (NOT EXISTS
   (SELECT *
    FROM R1 AS x1, R2 AS y1,
         R1 AS x2, R2 AS y2
    WHERE x1.loanId = y1.loanId
    AND x2.loanId = y2.loanId
    AND x1.loanId = x2.loanId
    AND x1.bankname = x2.bankname
    AND y1.city <> y2.city))
```

To ensure that FD checking is efficient, only a single relation should be examined for each FD

Deep Insight for Today

Split relations along functional dependencies

Redundancy Avoidance

Main Idea:

Avoid duplicate entries in a relation for a FD.

How could this happen?

There exists some FD $X \rightarrow Y$ covered by relation and X is not a super key

Deep Insight for Today

Split relations along functional dependencies

Normal Forms

Now we know how to derive FDs, we can then:

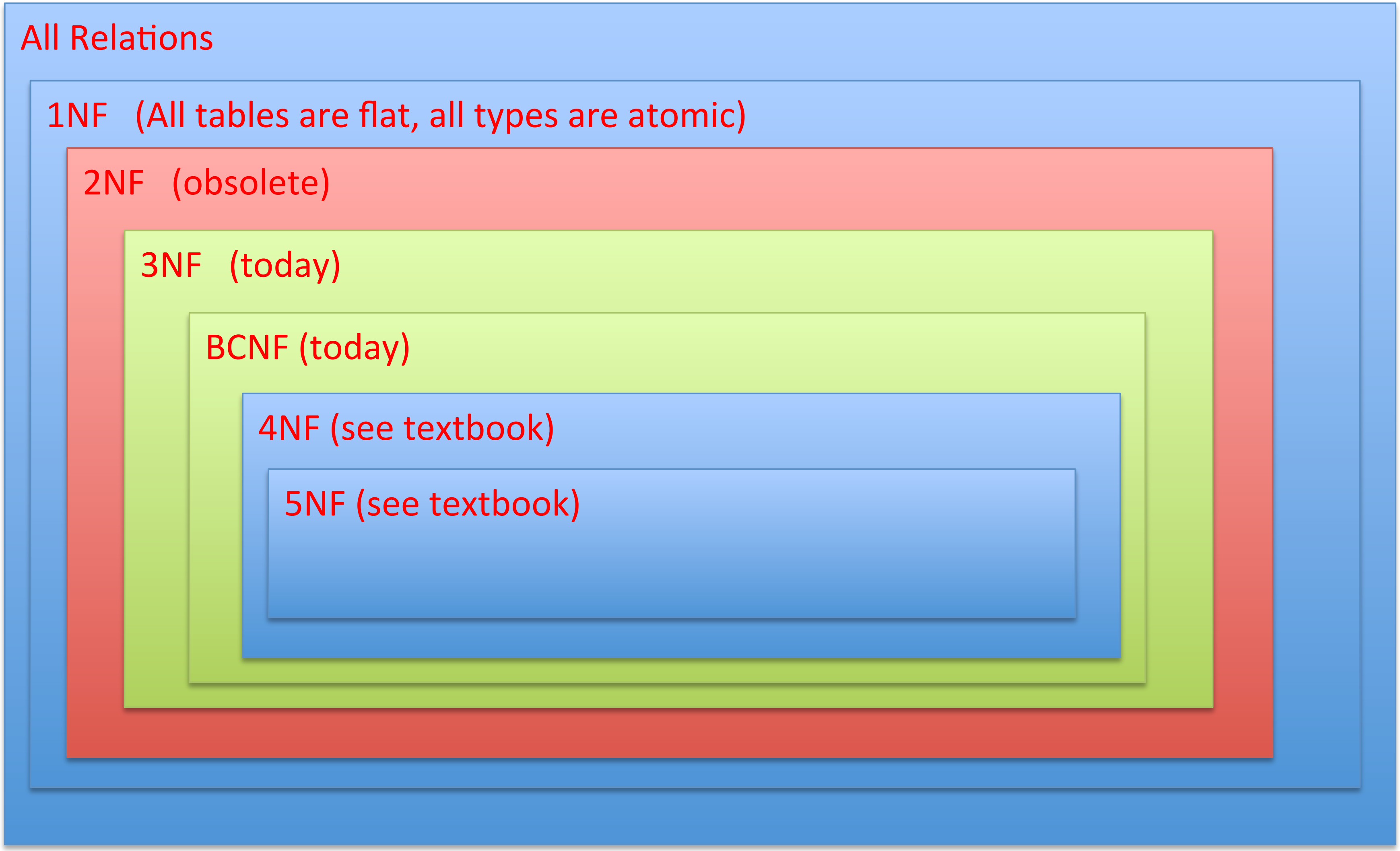
- Search for “bad” FDs
- If there are such, then decompose the table into two tables, repeat for the sub-tables.
- When done, the database schema is normalized

Normal Forms

Definition:

A normal form is a characterization of a schema decomposition in terms of the properties that satisfies.

Normal Forms (NFs)



1 Normal Form

All fields are atomic

This relation is not in 1NF

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus, Carsten	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Boyce-Codd Normal Form

- BCNF guarantees no redundancies and no lossless joins (but not Dependency Preservation)
- A relation R with FD set F is in BCNF if for all non-trivial $X \rightarrow Y$ in F^+ the following holds:

  $X \rightarrow \text{Attributes}(R)$ (i.e., X is a super key)

Example 1

R(A, B, C)
$A \rightarrow B$
$B \rightarrow C$

Is R in BCNF? NB A is a candidate key

Let's check some unavoidable FDs:

- $A \rightarrow B$
- $A \rightarrow C$
- $B \rightarrow C$

R is not in BCNF!

Example 2

$R_1(A, B)$ $R_2(B, C)$

$A \rightarrow B$

$B \rightarrow C$

Are R_1, R_2 in BCNF? NB A, B are candidate keys

Let's check some unavoidable FDs:

- If $A \rightarrow B$ then $A \rightarrow$ Attributes (R_1)
- If $B \rightarrow C$ then $B \rightarrow$ Attributes (R_2)

R_1, R_2 are in BCNF!

Boyce Codd Normal Form

Given a schema R and a set of FDs F, we can always decompose R into a set of relations $\{R_1, \ldots R_n\}$ such that

- $\{R_1, \ldots R_n\}$ are in BCNF
- The decompositions are lossless.

Remark: Some BCNF decompositions might not preserve dependencies.

BCNF Decomposition Algorithm

Given a relation R and a set of FDs F:

1. Compute F^+
2. Result $\leftarrow$ {R}
3. While some $R_i \in$ Result not in BCNF, do:
   - Choose $(X \rightarrow Y) \in F^+$ such that $(X \rightarrow Y)$ is covered by R_i and X is not a superkey. ($X \rightarrow$ Attributes(R_i) not valid)
   - Decompose R_i on $(X \rightarrow Y)$:
   - $R_{i,1} \leftarrow X \cup Y$
   - $R_{i,2} \leftarrow Ri - Y$
   - Result $\leftarrow$ (Result - {Ri}) $\cup$ $\{R_{i,1}, R_{i,2}\}$

Backto the Example: Closure

1. cpr → name address [provided]
2. cpr courseID → grade [provided]
3. cpr name → cpr [by reflexivity]
4. cpr → name [by decomposition]
5. cpr → address [by decomposition]
6. cpr grade → name grade [by augmentation]
7. cpr courseID → name grade [by transitivity]

Step 1

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Functional Dependencies
cpr → name address
cpr courseID → grade

- Compute F^+ already done!

Step 2

cpr	courseID	room	grade	name	address
140298-1234	SIDD	AUD1	12	Jesper	Copenhagen
041297-5367	SIDD	AUD1	10	Nikoline	Aarhus
151197-2352	CRITSYS	2A12	12	Claus	Dragør
050596-1142	CRITSYS	3A01	7	Martin	Copenhagen

Functional Dependencies
cpr → name address
cpr courseID → grade

- Choose cpr → name address
- cpr is not a superkey

Step 3

Functional Dependencies
cpr → name address
cpr courseID → grade

- R is not in BCNF
- Decompose R into the two smaller relations R_1 and R_2

cpr	courseID	room	grade
140298-1234	SIDD	AUD1	12
041297-5367	SIDD	AUD1	10
151197-2352	CRITSYS	2A12	12
050596-1142	CRITSYS	3A01	7

cpr	name	address
140298-1234	Jesper	Copenhagen
041297-5367	Nikoline	Aarhus
151197-2352	Claus	Dragør
050596-1142	Martin	Copenhagen

DONE!

Problem

- We started with a relation R and its dependency set FD.
- We decomposed R into BCNF relations $\{R_1,\ldots R_n\}$ with their own $\{FD_1,\ldots,FD_n\}$.
- We can reconstruct R from $\{R_1,\ldots R_n\}$
- But we cannot necessarily reconstruct FD from $\{FD_1,\ldots,FD_n\}$

More about this next time.