Normalisation – Part 1

Schema Design

- A driving force for **the study of dependencies** has been **schema design**.
- T **ma** for a p
- The **choice of a schema** is guided by **semantic information** about the application data provided by users and captured
- A common approach starts with a **univer** decomposition to create new relations that satisf **normalization**).

Normal Forms

Normal forms	Test criteria
1NF	
⇓	
BCNF	strong
...	

- **Note that:**
  - 1NF is not based on any constraints.
  - 2NF, 3NF and BCNF are based on keys and functional dependencies.
  - 4NF and 5NF are based on other constraints (will not be covered).

Normalisation

- Decomposing a relation into **smaller relations in a certain normal form**
  -
- What normal forms will we learn?
  1. Boyce-Codd normal form (**BCNF**)
  2. Third normal form (**3NF**)

BCNF - Definition

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, t
- w
fu

  - Note: this does not necessarily mean a good

Do not represent the same fact twice (

Normalisation to BCNF

- Consider the relation schema TEACH with the following FDs:
  -
  -

StudentID	CourseName	Instructor
u123456	Operating Sy	
u234567	Operating Sy	
u234567	Databas	

- **Is TEACH in BCNF?**
  - Not in BCNF because of $\{\text{Instructor}\} \rightarrow \{\text{CourseName}\}$.

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

In

- https://eduassistpro.github.i

- Do the following for each $R \in \mathcal{S}$ iter :

  - Find a (non-trivial) FD $X \to Y$
  - Replace R in $\mathcal{S}$ by two relation schemas XY and $(R - Y)$ and project the FDs to these two relation schemas.

Normalisation to BCNF

Normalisation to BCNF

BCNF - Example

- Consider TEACH with the following FDs again:
  - {StudentID,CourseName} → {Instructor};
  - Instructor → CourseName .

u234567	Operating Systems	Jane
u234567	Databas	

- **Can we normalise TEACH into BCNF?**

BCNF - Example

- Consider TEACH with the following FDs again:
  - {StudentID,CourseName} → {Instructor};
  - **Instructor** **CourseName** .

u234567	Operating Sy	
u234567	Databas	

- **Replace TEACH with R_1 and R_2:**

R_1

CourseName	Instructor
Operating Systems	Jane
Databases	Mark

R_2

StudentID	Instructor
u123456	Jane
u234567	Jane
u234567	Mark

BCNF - Example

- Consider the relation schema TEACH with the following FDs:
  - {StudentID,CourseName} → {Instructor};
  - Instructor → CourseName .

u234567	Operating Sy	
u234567	Databas	

R_1

CourseName	Instructor
Operating Systems	Jane
Databases	Mark

u123456	Jane
u234567	Jane
u234567	Mark

- **Does this decomposition preserve all FDs on TEACH?**

BCNF - Example

- Consider the relation schema TEACH with the following FDs:
  - {StudentID,CourseName} → {Instructor}; Lost!
  - Instructor → CourseName .

u234567	Operating Sy	
u234567	Databas	

R_1

CourseName	Instructor
Operating Systems	Jane
Databases	Mark

u123456	Jane
u234567	Jane
u234567	Mark

- **No. We only have {Instructor} → {CourseName} on R_1.**

Two Properties

- We need to consider the following properties when decomposing a relation:

1.

   NATURAL JOIN operation is applied to th

2. **Dependency preservation** – "**cap**

   To ensure that each functional dependency can be inferred from functional dependencies after decomposition.

Two Properties

- **Facts**

(2) However, a BCNF-decomposition that is
and dependency-preserving do

- Does there exist **a less restrictive normal form** such that a lossless and dependency preserving decomposition can always be found?