Normalisation – Part 1

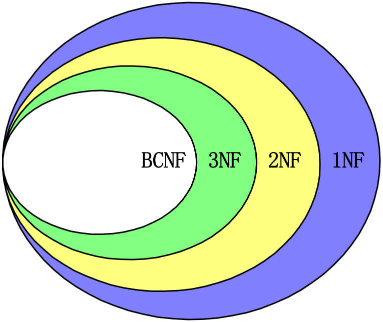
BCNF

Schema Design

- A driving force for **the study of dependencies** has been **schema design**.
- The goal of schema design is to select **the most appropriate schema** for a particular database application.
- The **choice of a schema** is guided by **semantic information** about the application data provided by users and captured by dependencies.
- A common approach starts with a **universal relation** and applies decomposition to create new relations that satisfy certain normal forms (i.e. **normalization**).

Normal Forms

Normal forms	Test criteria
1NF	
⇓	weak
2NF	
⇓	⇓
3NF	
⇓	
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**
  - Each normal form reduces certain kind of data redundancy.
  - Each normal form does not have certain types of (undesirable) dependencies.
- 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, then X is a **superkey**.
- When a relation schema is in BCNF, all data redundancy based on functional dependency are removed.
    - Note: this does not necessarily mean a good design.

Do not represent the same fact twice (within a relation)!

Normalisation to BCNF

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

TEACH

StudentID	CourseName	Instructor
u123456	Operating Systems	Jane
u234567	Operating Systems	Jane
u234567	Databases	Mark

- **Is TEACH in BCNF?**
    - Not in BCNF because of {Instructor} → {CourseName}.

Normalisation to BCNF

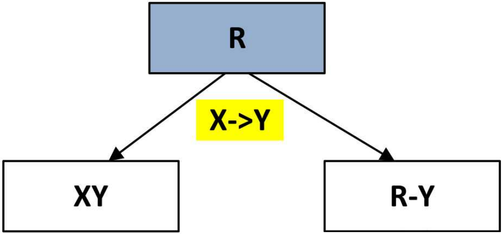
- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

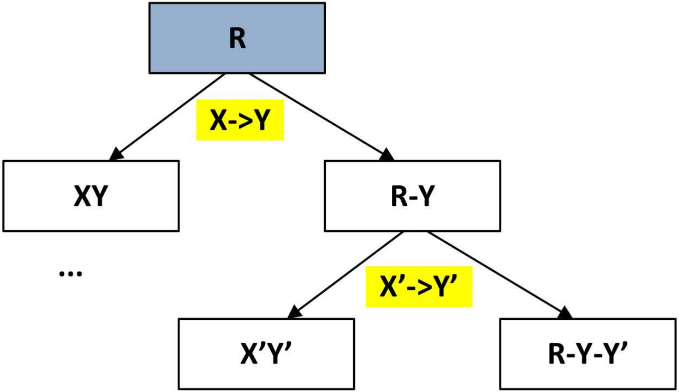
  Output: a set $\mathcal{S}$ of relation schemas in BCNF, each having a set of FDs

  - Start with $\mathcal{S} = \{R'\}$;
  - Do the following for each $R \in \mathcal{S}$ iteratively until no changes on $\mathcal{S}$:
    - Find a (non-trivial) FD $X \rightarrow Y$ on R that violates BCNF, if any;
    - 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}.

TEACH

StudentID	CourseName	Instructor
u123456	Operating Systems	Jane
u234567	Operating Systems	Jane
u234567	Databases	Mark

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

BCNF - Example

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

TEACH

StudentID	CourseName	Instructor
u123456	Operating Systems	Jane
u234567	Operating Systems	Jane
u234567	Databases	Mark

- **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:
  - $\{\text{StudentID},\text{CourseName}\} \rightarrow \{\text{Instructor}\}$;
  - $\{\text{Instructor}\} \rightarrow \{\text{CourseName}\}$.

TEACH

StudentID	CourseName	Instructor
u123456	Operating Systems	Jane
u234567	Operating Systems	Jane
u234567	Databases	Mark

R_1

CourseName	Instructor
Operating Systems	Jane
Databases	Mark

R_2

StudentID	Instructor
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}.

TEACH

StudentID	CourseName	Instructor
u123456	Operating Systems	Jane
u234567	Operating Systems	Jane
u234567	Databases	Mark

R_1

CourseName	Instructor
Operating Systems	Jane
Databases	Mark

R_2

StudentID	Instructor
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. **Lossless join** – " **capture the same data** "

   To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

2. **Dependency preservation** – " **capture the same meta-data** "

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

Two Properties

- **Facts**

(1) There exists an algorithm that can generate **a lossless decomposition into BCNF**.

(2) However, a BCNF-decomposition that is **both lossless and dependency-preserving** does not always exist.

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