CS344

Schema Refinement (Continued)

Rajat Khanduja
09010137

Rovin Bhandari
09010144

Schema Refinement (Recap)

- **Need for Schema Refinement**
  - Redundancy is a primary issue in data storage.
  - Problems caused by redundancy are as follows :
    - Redundant storage
    - Update anomalies
    - Insertion anomalies
    - Deletion anomalies
  - Although decomposition can eliminate redundancy, it causes problems of its own.
  - Problems related to decomposition :-
    - Lossless vs. lossy decomposition
    - Dependency preserving vs non dependency preserving decomposition
  - Schema refinement aims at addressing these problems by proposing several 'Normal forms'.
  - Each normal form has a set of properties and if a schema is in a particular normal form it is possible to predict the problems that would not arise.

- **Functional dependencies**
  - A functional dependency is a constraint between two sets of attributes of a relation in a schema.
  - A set of attributes X in a relation R is said to functionally determine another set of attributes Y in R if each X value is associated with one Y value.
  - A trivial functional dependency is one in which the right side only contains attributes that also appear on the left side.

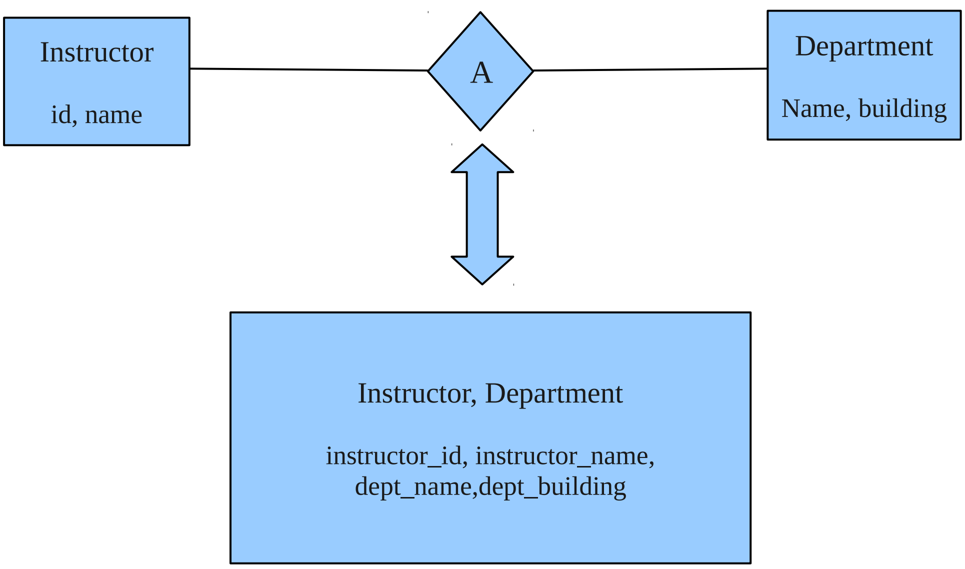
Closure of a set of FDs

- The set of all functional dependencies (FDs) is called the closure of F.
- To compute the closure of a given set of FDs, Armstrong's Axioms may be used repeatedly until no new FD is found.
- Armstrong's Axioms (AAs)
  - *Reflexivity* : $\alpha \rightarrow \beta$ holds if $\alpha \subseteq \beta$
  - *Augmentation* : $\gamma\,\alpha \rightarrow \gamma\,\beta$ holds if $\alpha \rightarrow \beta$ & $\gamma \subseteq R$ (for any relation R)
  - *Transitivity* : $\alpha \rightarrow \gamma$ holds if $\alpha \rightarrow \beta$ & $\beta \rightarrow \gamma$ hold
- Derived from Armstrong's axioms
  - *Union* : $\alpha \rightarrow \gamma\,\beta$ holds if $\alpha \rightarrow \beta$ holds & $\alpha \rightarrow \gamma$ holds
    - Proof :
      $\alpha \rightarrow \beta$ (1)
      $\alpha \rightarrow \gamma$ (2)
      Augmenting (1) with α
      $\alpha\,\alpha \rightarrow \alpha\,\beta$ i.e $\alpha \rightarrow \alpha\,\beta$ [$\alpha\,\alpha = \alpha$] (3)
      Similarly, augmenting (2) with β
      $\alpha\,\beta \rightarrow \gamma\,\beta$ (4)
      From (3) and (4)
      $\alpha \rightarrow \gamma\,\beta$ [Using transitivity]
      Hence proved
  - *Decomposition* : $\alpha \rightarrow \beta$ holds & $\alpha \rightarrow \gamma$ holds if $\alpha \rightarrow \gamma\,\beta$ holds
    - Proof :
      $\gamma\,\beta \rightarrow \gamma$ [trivial FD] (1)
      $\gamma\,\beta \rightarrow \beta$ [trivial FD] (2)
      Also,
      $\alpha \rightarrow \gamma\,\beta$ [Given] (3)
      Therefore,
      $\alpha \rightarrow \beta$ [Using (2), (3) and transitivity]
      $\alpha \rightarrow \gamma$ [Using (1), (3) and transitivity]
      Hence proved

- *Pseudo-transitivity* : $\gamma\,\alpha \rightarrow \delta$ holds if $\alpha \rightarrow \beta$ & $\gamma\,\beta \rightarrow \delta$ hold
  - Proof :
    Augmenting $(\alpha \rightarrow \beta)$ with γ
    $\gamma\,\alpha \rightarrow \gamma\,\beta$ (1)
    Also,
    $\gamma\,\beta \rightarrow \delta$ [Given] (2)
    By transitivity,
    $\gamma\,\alpha \rightarrow \delta$
    <u>Hence proved</u>

Schema Normalization

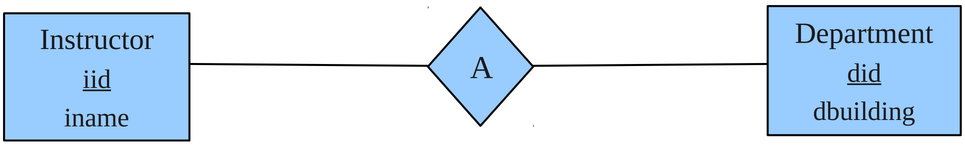
- Normal Form of a schema is an indicator of the quality (and redundancy of data that might be involved) of the schema.
- Order of weakest to strongest normal forms :- 1,2,3, BCNF.
  - Two extremes:
    - One big table: results in data redundancy
    - Many (smaller) tables: little or no redundancy
  - Higher the normal form, less the redundancy, more the number of tables.
  - Keep all entities separate to ensure minimum redundancy.



- These forms have increasingly restrictive requirements. Every relation in a higher normal form is also in all of the lower forms.
- BCNF allows only 2 types of FDs:
  - trivial
  - implied by super keys
- BCNF ensures that no redundancy can be detected using FDs

  [since, a value can be stored twice only if the key is defined twice]

  Example:



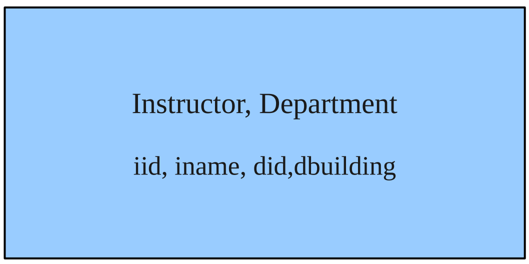
Here,

the FDs are trivial:

iid $\rightarrow$ iname

Therefore, the schema is in BCNF.

But if the two tables are combined to form a single table,



the FDs are no longer derivable from the super key:

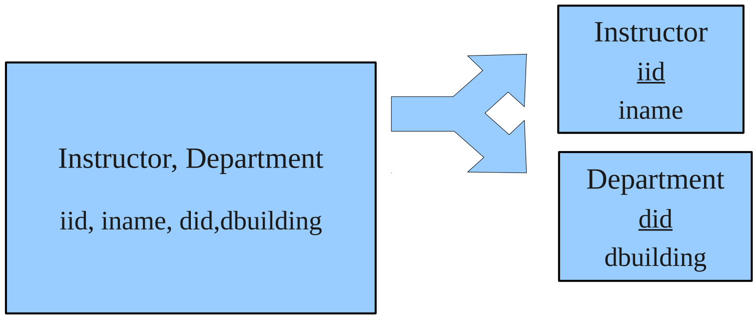
only iid cannot identify a unique row of it

(primary could be iid, did)

Therefore, the schema is not in BCNF.

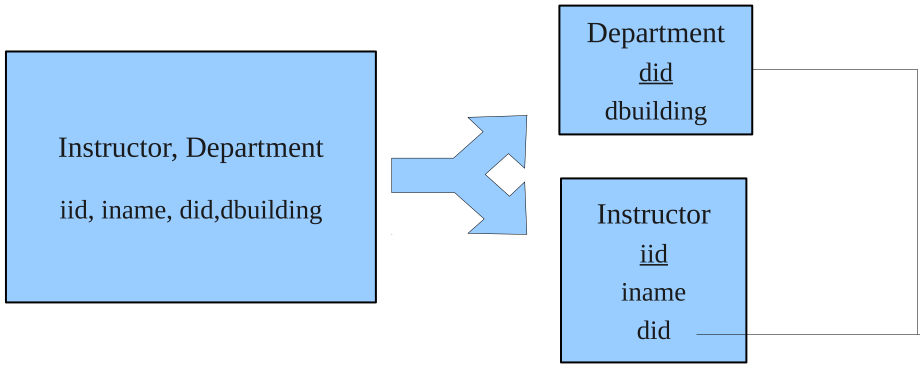
- To have a schema satisfy BCNF, we need to ensure more number of (hence, smaller sized) tables by decomposing the original table (lossless decomposition, that is).
- Decomposition of R into R1 and R2 is lossless iff R1 $\cap$ R2 is a key for R1 or R2.

e.g.,



is not lossless,

while,



is lossless.