Normalization

Dr. Shrutilipi Bhattacharjee, Assistant Professor, Dept. of IT, NIT Karnataka, India

Second Normal Form

- A relation is in the second normal form if it fulfills the following two requirements:
  - It is in first normal form
  - It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation
- **A non-prime attribute of a relation** is an attribute that is not a part of any candidate key of the relation

- Put simply, a relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key
- Note that it does not put any restriction on the non-prime to non-prime attribute dependency; that is addressed in third normal form

Second Normal Form

- Example:
  - Consider following functional dependencies in relation R (<u>A, B</u>, C, D)
  - AB $\rightarrow$ C [A and B together determine C]
  - C $\rightarrow$ D [C determines D]
  - In the above relation, AB is the only candidate key and there is no partial dependency, i.e., any proper subset of AB doesn’t determine any non-prime attribute

- **A normal form of historical significance:**

  “
  - You may have noted that we skipped second normal form
  - It is of historical significance only and, in practice, one of third normal form or BCNF is always a better choice
  - First normal form pertains to attribute domains, not decomposition

  ”

Lossless Decomposition

- We can use functional dependencies to show when certain decomposition are lossless

- For the case of $R = (R_1, R_2)$, we require that for all possible relations r on schema R

$$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

- A decomposition of R into R_1 and R_2 is lossless decomposition if at least one of the following dependencies is in F^+:

$$R_1 \cap R_2 \rightarrow R_1$$

$$R_1 \cap R_2 \rightarrow R_2$$

- The above functional dependencies are a sufficient condition for lossless join decomposition
- The dependencies are a necessary condition only if all constraints are functional dependencies

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
- $R_1 = (A, B), \quad R_2 = (B, C)$
  – Lossless decomposition:
  $$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$
- $R_1 = (A, B), \quad R_2 = (A, C)$
  – Lossless decomposition:
  $$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$
- *Note:*
  – $B \rightarrow BC$ is a shorthand notation for $B \rightarrow \{B, C\}$

Boyce-Codd Normal Form

- A relation schema *R* is in BCNF with respect to a set *F* of functional dependencies if for all functional dependencies in F^+ of the form

$$\alpha \to \beta$$

where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

– $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)

– α is a superkey for *R*

- *Example schema not in BCNF*
  – *in_dep(<u>ID</u>, name, salary, <u>dept_name</u>, building, budget)*
  – *Because dept_name* $\to$ *building, budget* holds on *in_dep*, but *dept_name* is not a superkey

Decomposing a Schema into BCNF

- Let R be a schema *R* that is not in BCNF
- Let $\alpha \rightarrow \beta$ be the FD that causes a violation of BCNF

- We decompose *R* into:
  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$

- In our example of *in_dep*,
  - α = *dept_name*
  - β = *building, budget*
  - *dept_name* $\rightarrow$ *building, budget*

- and *in_dep* is replaced by
  - $(\alpha \cup \beta)$ = (*dept_name, building, budget*)
    - *dept_name* $\rightarrow$ *building, budget*
  - $(R - (\beta - \alpha))$ = (*ID, name, dept_name, salary*)
    - *ID* $\rightarrow$ *name, salary, dept_name*

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
- $R_1 = (A, B), \quad R_2 = (B, C)$
  - Lossless-join decomposition:
    $$R_1 \cap R_2 = \{B\} \quad \text{and } B \rightarrow BC$$
  - Dependency preserving
- $R_1 = (A, B), \quad R_2 = (A, C)$
  - Lossless-join decomposition:
    $$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$
  - Not dependency preserving
    (cannot check $B \rightarrow C$ without computing $R_1 \bowtie R_2$)

BCNF and Dependency Preservation

- It is not always possible to achieve both BCNF and dependency preservation
- Consider a schema: *dept_advisor(s_ID, i_ID, department_name)*
- With function dependencies:

$$i\_ID \rightarrow dept\_name$$
$$s\_ID, dept\_name \rightarrow i\_ID$$

- In the above design, we are forced to repeat the department name once for each time an instructor participates in a *dept_advisor* relationship
- *dept_advisor* is not in BCNF
  - *i_ID* is not a superkey
- To fix this, we need to decompose *dept_advisor*
- Any decomposition of *dept_advisor* will not include all the attributes in

$$s\_ID, dept\_name \rightarrow i\_ID$$

- Thus, the composition is NOT be dependency preserving

Dependency Preservation

- Testing functional dependency constraints each time the database is updated can be costly

- It is useful to design the database in a way that constraints can be tested efficiently

- If testing a functional dependency can be done by considering just one relation, then the cost of testing this constraint is low

- When decomposing a relation it is possible that it is no longer possible to do the testing without having to perform a Cartesian Produced

- A decomposition that makes it computationally hard to enforce functional dependency is said to be NOT **dependency preserving**

BCNF and Dependency Preservation

- Constrains, including functional dependencies, are costly to check in practice unless they pertain to only one relation

- If it is sufficient to test only those dependencies on each individual relation of a decomposition in order to ensure that ***all*** functional dependencies hold, then the decomposition is dependency preserving

- Because it is not always possible to achieve both BCNF and dependency preservation, we consider a weaker normal form, known as *third normal form*

Next Lecture

Normalization

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in