Normalization Algorithms

SWEN304/SWEN435

Lecturer: Dr Hui Ma

Engineering and Computer Science

Normalization

- Normalization is used to design a set of relation schemas that is optimal from the point of view of database updating
- The normalization starts from a universal relation schema
- There are six normal forms, of which three are based on functional dependencies
- Normal forms define to which extent we should normalize
- The Synthesis algorithm and the Decomposition algorithm represent the formal normalization methods
- *Readings from the textbook:*
  - *Chapter 15* : *15.1-15.5,*
  - *Chapter 16* : *16.1 -16.3*

Normalization

- Normalization is a database design procedure whose input is (U, F), and the output is

$$S = \{(R_i, F_i) \mid i = 1, \ldots, n\}$$

- Desirable properties of a decomposition S are:
  - $U = \bigcup_{i=1}^{n} R_i$ (Attribute preservation)
  - $F^+ = (\bigcup_{i=1}^{n} F_i)^+$ (Dependency preservation)
  - Lossless join decomposition

Normalization

- Note, for every set

$$S = \{(R_i, F_i) \mid i = 1,\dots, n\}$$

of relation schemas, there exists one (hypothetical) universal relation schema (U, F) such that

$$U = \bigcup_{i=1}^{n} R_i \text{ , and}$$

$$F = \bigcup_{i=1}^{n} F_i$$

- So, given S, you can infer (U, F)

Third Normal Form

- A relation schema $N(R, F)$ with a set of keys $K(N)$ is in **third normal form** (3NF) if for each non-trivial functional dependency $X \to A$ holds in F, **either** X is a superkey of N, **or** A is a prime attribute of N
- X is a superkey of N: X is a superset of a key of N
- Formally

$$(\forall f: X \to A \in F)(A \in X \vee X \to R \in F^{+} \vee (\exists Y \in K(N))(A \in Y))$$

- Relation schemas being in 3NF but not in BCNF still exhibit some update anomalies

Lossless 3NF Decomposition

Synthesis Algorithm

Input: (U, F)

Output: $S = \{(R_i, K_i) | i = 1, \ldots, n\}$ (*K_i is the relation schema key*)

1. Find a minimal cover G of F
2. Group FDs from G according to the same left-hand side. For each group of FDs

$$(X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_k),$$

   make one relation schema in S

$$(\{X, A_1, A_2, \ldots, A_k\}, X)$$

3. If none of relation schemes in S contain a key of (U, F), create a new relation scheme in S that will contain only a key of (U, F)

Properties of Synthesis Algorithm

- At least third normal form
- Attribute preservation
- Functional dependency preservation
- Lossless join decomposition

- Lossless join property of *S* is the consequence of a theorem proving that *S* represents a non-additive decomposition if it contains a relation schema that contains a key of the constructed universal relation schema
- This property is valid for any set of relation schemas

Boyce-Codd Normal Form

- The Boyce-Codd normal form is the highest NF that is based on FDs
- The relation schema (R, F) is in the **Boyce-Codd Normal Form (BCNF)**, if the left-hand side of each non trivial functional dependency in F contains a relation schema key
- Formally

$$(\forall f: X \rightarrow A \in F)(A \in X \vee X \rightarrow R \in F^+)$$

- A relation in BCNF is free from update anomalies
- Ideally, relation database design should try to achieve BCNF or 3NF for every relation schema

BCNF Test

- Given R and F on R
- Relation schema (R, F) is not in BCNF if there exists a non-trivial FD $X \rightarrow A$ in F such that $R \not\subseteq X^+_F$

- Example:
  - $R = \{StudId, CourId, LecId\}$
  - $F = \{StudId + CourId \rightarrow LecId, LecId \rightarrow CourId\}$
    - $LecId \rightarrow CourId$ is a non trivial FD,
    - and $LecId$ is not a relation schema key

BCNF Decomposition

Decomposition algorithm:

Input: (U, F)

Output: $S = \{(R_i, F_i)\,|\, i = 1,\ldots, n\}$

1. Set $S := \{(U, F)\}$
2. While there is a relation schema (R, G) in S that is not in BCNF do

   2.1 Choose a functional dependency $X \rightarrow Y$ in G that violates BCNF,

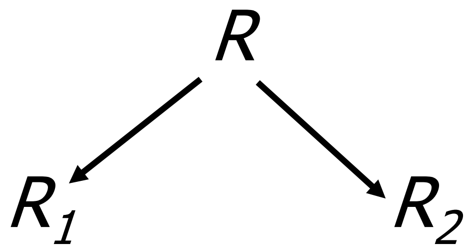
   2.2 Replace (R, G) with $(R - Y, G|_{R-Y})$ and $(XY, G|_{XY})$

- The final result will be a lossless BCNF-decomposition

BCNF Decomposition Properties

- Properties:
  - Boyce-Codd normal form
  - Attribute preservation
  - Lossless join decomposition
  - Some functional dependencies may be lost

- The decomposition algorithm is based on a step by step splitting of relations until desired normal form is achieved



Projection of a Set of FDs

- Given U, F and $W \subseteq U$, projection of F onto W is

$$F|_W = \{X \to A \in F^+ \mid AX \subseteq W\}$$

All the FDs in the closure of F that have both LHS and LHS as subsets of W

- When decomposing one relation schema (R, F) onto two new relation schemas (R_1, F_1) and (R_2, F_2), then

$$F_1 = F|_{R1} \text{ and } F_2 = F|_{R2}$$

A Question

- Let $min(F|_W)$ denote a minimal cover of $F|_W$

- Given $F = \{A \rightarrow B, B \rightarrow C\}$
- Which answer is correct:

a) $min(F|_{AC}) = \{ \}$

b) $min(F|_{AC}) = \{A \rightarrow B\}$

c) $min(F|_{AC}) = \{A \rightarrow C\}$

Lossless Join Decomposition Property 1

- A decomposition $D(R) = \{R_1, R_2\}$ is a lossless join decomposition of R with respect to F if

$$R_1 \cap R_2 \rightarrow R_1 \in F^+ \vee R_1 \cap R_2 \rightarrow R_2 \in F^+$$

- That property leads to a conclusion:

  Given R and $F = \{X \rightarrow Y, ...\}$ set of FDs in R, a decomposition

$$R_1 = R - Y,\ F_1 = F|_{R-Y}$$

$$R_2 = XY,\ F_2 = F|_{XY}$$

  is a non-additive (lossless join) decomposition

A Question

- Given $R = \{A, B, C\}$ and $F = \{B \to C\}$
- Is the decomposition $D = \{R_1, R_2\}$ with

  $R_1 = \{A, B\}$, $F_1 = \{\ \}$ and

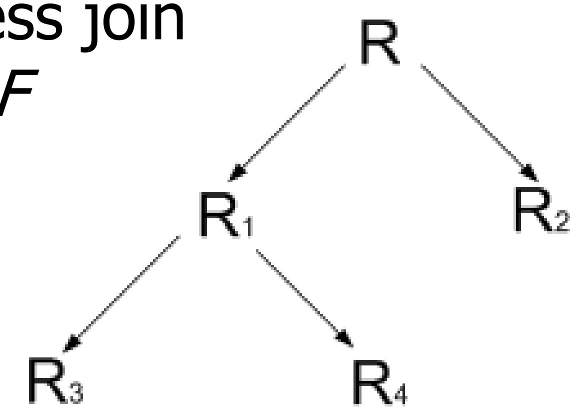
  $R_2 = \{B, C\}$, $F_2 = \{B \to C\}$

  lossless?

- Yes,
- because $\{A, B\} \cap \{B, C\} = \{B\}$ and if $B \to C$ belongs to F_2, then B is a key of R_2, i.e.,$B \to R_2$

Lossless Join Decomposition Property 2

- If $D(R) = \{R_1, R_2\}$ is a lossless join decomposition of R with respect to F, and
- $D(R_1) = \{R_3, R_4\}$ is a lossless join decomposition of R_1 with respect to $F_1 = F|_{R1}$
- So is $D(R) = \{R_2, R_3, R_4\}$ a lossless join decomposition of R with respect to F



- Property 2 says that the decomposition process may be continued until the desired normal form is achieved and that the resulting decomposition will be the lossless one

Finishing Database Design

- After the normalization, one has also to define interrelation constraints (referential integrity constraints)

Checking FD Satisfaction

- When a database schema is in BCNF, all nontrivial functional dependencies, embedded in a relation schema, contain a key on their left-hand side,

- Only then, by means of SQL DDL CREATE TABLE key definition, a DBMS becomes able to check satisfaction of functional dependencies
  - Since keys are unique, no FD left-hand side can have duplicate values, hence no FD violation

BCNF Decomposition: An Example

Input: (U, F)

Output: $S = \{(R_i, F_i) | i = 1,\dots, n\}$

1. Set $S := \{(U, F)\}$
2. While there is a relation schema (R, G) in S that is not in BCNF do
   2.1 Choose a functional dependency $X \to Y$ in G that violates BCNF,
   2.2 Replace (R, G) with $(R - Y, G|_{R-Y})$ and $(XY, G|_{XY})$

- For a relation N
  - let $R = ABCD$
  - let $F = \{A \to B, B \to C, CD \to A, AC \to D\}$
- Compute $B^+ = BC$, so B is not a superkey
- Decomposition along $B \to C$ gives

  $$R_1 = ABD \text{ and } R_2 = BC$$
- In addition we get $F_1 = \{A \to B, A \to D, BD \to A\}$ and $F_2 = \{B \to C\}$

BCNF Decomposition: An Example

- Check R_1 and R_2 to see if they are in BCNF
  - R_2 is in BCNF because $(B)^+ = BC = R_2$
  - Compute $A^+ = ABD$ and $(BD)^+ = ABD$. So, R_1 is in BCNF
- Hence, obtained lossless BCNF-decomposition
- However, $CD \to A \in F^+$, but $CD \to A \notin (F_1 \cup F_2)^+$
- In this lossless BCNF-decomposition we lost dependencies

Summary

- The Synthesis algorithm is based on finding a minimal cover of the given FD set
  - It guaranties third normal form, lossless join decomposition, attribute and FD preservation
- The Decomposition algorithm is based on a gradual splitting of non-BCNF relation schemas onto two new relation schemas
  - Splitting is made using functional dependencies that violate BCNF
  - It guaranties a BCNF lossless join decomposition, and attribute preservation, but preservation of FDs is not guaranteed

Summary

- Normalization results in a set of relation schema
  - That design is suitable for efficient database update
  - But, it can slow down execution of queries
  - Sometimes, it is advisable to undertake controlled denormalization