Normal Forms. BCNF and 3NF Decompositions

Slides based on “Database Management Systems” 3rd ed, Ramakrishnan and Gehrke
Decomposition of a Relation Schema

- A **decomposition** of R replaces it by two or more relations
  - Each new relation schema contains a subset of the attributes of R
  - Every attribute of R appears in one of the new relations
  - E.g., SNLRWH decomposed into SNLRH and RW

- Decompositions should be used only when needed
  - Cost of join will be incurred at query time

- Problems may arise with (improper) decompositions
  - Reconstruction of initial relation may not be possible
  - Dependencies cannot be checked on smaller tables
Lossless Join Decompositions

- Decomposition of $R$ into $X$ and $Y$ is **lossless-join** if:
  \[ \pi_X (r) \bowtie \pi_Y (r) = r \]

- It is always true that $r \subseteq \pi_X (r) \bowtie \pi_Y (r)$
  - In general, the other direction does not hold!
  - If it does, the decomposition is lossless-join.

- *It is essential that all decompositions used to deal with redundancy be lossless!*
Incorrect Decomposition

Natural Join

A  B  C
1  2  3
4  5  6
7  2  8

B  C
2  3
5  6
2  8

A  B  C
1  2  3
4  5  6
7  2  8
1  2  8
7  2  3
Condition for Lossless-join

- The decomposition of R into X and Y is lossless-join wrt F if and only if the closure of F contains:
  - X ∩ Y → X, or
  - X ∩ Y → Y

- In particular, the decomposition of R into UV and R - V is lossless-join if U → V holds over R.
Dependency Preserving Decomposition

- Consider CSJDPQV, C is key, JP → C and SD → P.
- Consider decomposition: CSJQDV and SDP
- Problem: Checking JP → C requires a join!

Dependency preserving decomposition (Intuitive):
- If R is decomposed into X and Y, and we enforce the FDs that hold on X, Y then all FDs that were given to hold on R must also hold.

Projection of set of FDs F: If R is decomposed into X, ... projection of F onto X (denoted \( F_X \)) is the set of FDs \( U \rightarrow V \) in \( F^+ \) (closure of F) such that U,V are in X.
Dependency Preserving Decompositions

- Decomposition of R into X and Y is *dependency preserving* if 
\[(F_X \cup F_Y)^+ = F^+\]

- Dependencies that can be checked in X without considering Y, and in Y without considering X, together represent all dependencies in \(F^+\)

- Dependency preserving does not imply lossless join:
  - ABC, \(A \rightarrow B\), decomposed into AB and BC.
Normal Forms

- If a relation is in a certain normal form (BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimized.

- Role of FDs in detecting redundancy:
  - Consider a relation R with attributes AB
    - No FDs hold: There is no redundancy
    - Given A → B:
      - Several tuples could have the same A value
      - If so, they’ll all have the same B value!
Boyce-Codd Normal Form (BCNF)

- Relation R with FDs F is in BCNF if, for all $X \rightarrow A$ in $F^+$
  - $A \subseteq X$ (called a trivial FD), or
  - $X$ contains a key for R

- The only non-trivial FDs allowed are key constraints

- BCNF guarantees no anomalies occur
Decomposition into BCNF

Consider relation R with FDs F. If X → Y violates BCNF, decompose R into R - Y and XY.

Repeated application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.

e.g., CSJDPQV, key C, JP → C, SD → P, J → S

To deal with SD → P, decompose into SDP, CSJLQV.

To deal with J → S, decompose CSJLQV into JS and CJDQV
Decomposition into BCNF

- In general, several dependencies may cause violation of BCNF. The order in which we “deal with” them could lead to very different sets of relations!
BCNF and Dependency Preservation

- In general, there may not be a dependency preserving decomposition into BCNF
  - e.g., ABC, AB → C, C → A
  - Can’t decompose while preserving first FD; not in BCNF
Third Normal Form (3NF)

- Relation $R$ with FDs $F$ is in 3NF if, for all $X \rightarrow A$ in $F^+$
  - $A \subseteq X$ (called a trivial FD), or
  - $X$ contains a key for $R$, or
  - $A$ is part of some key for $R$ ($A$ here is a single attribute)

- *Minimality* of a key is crucial in third condition above!

- If $R$ is in BCNF, it is also in 3NF.

- If $R$ is in 3NF, some redundancy is possible
  - compromise used when BCNF not achievable
  - e.g., no "good" decomposition, or performance considerations

- *Lossless-join, dependency-preserving decomposition of $R$ into a collection of 3NF relations always possible.*
Decomposition into 3NF

- Lossless join decomposition algorithm also applies to 3NF
- To ensure dependency preservation, one idea:
  - If \( X \rightarrow Y \) is not preserved, add relation \( XY \)
  - Refinement: Instead of the given set of FDs \( F \), use a \textit{minimal cover for} \( F \)

Example: \( CSJDPQV, JP \rightarrow C, SD \rightarrow P, J \rightarrow S \)
- Choose \( SD \rightarrow P \), result is \( SDP \) and \( CSJDPQV \)
- Choose \( J \rightarrow S \), result is \( JS \) and \( CJDQV \), all 3NF
- Add \( CJP \) relation
Summary of Schema Refinement

- **BCNF**: relation is free of FD redundancies
  - Having only BCNF relations is desirable
  - If relation is not in BCNF, it can be decomposed to BCNF
    - Lossless join property guaranteed
    - But some FD may be lost

- **3NF is a relaxation of BCNF**
  - Guarantees both lossless join and FD preservation

- **Decompositions may lead to performance loss**
  - *performance requirements* must be considered when using decomposition