Database Management Systems

Database Design (II)

Good Database Design

- Rule of the thumb: Independent facts in separate tables

- Each relation schema should consist of a primary key and a set of mutually independent attributes.

- Lazy rule: is it in normal form(s)?

Boyce-Codd Normal Form (BCNF)

Let R be a relation schema and F a set of functional dependencies.

Schema R is in BCNF if and only if whenever $(X \rightarrow Y) \in F^+$ and $XY \subseteq R$, then either

- $(X \rightarrow Y)$ is trivial (i.e., $Y \subseteq X$), or
- X is a superkey of R.

A database schema $\{R_1, \ldots, R_n\}$ is in BCNF if each relation schema R_i is in BCNF.

BCNF

- Formalized the goal that independent facts are stored in separate tables.
- What should be done if R is not in BCNF? — Decomposition.
- Identify undesirable dependencies (the ones that make R not in BCNF), and decompose the schema R using these dependencies.

Decomposition

- Definition: Let R be a relation schema. The collection $\{R_1, \ldots, R_n\}$ of relation schemas is a decomposition of R if $R = R_1 \cup R_2 \cup \ldots \cup R_n$.

- A good decomposition
  - does not lose information (most important one)
  - does not complicate checking of constraints

Lossless-Join Decomposition

- Definition: Suppose R is the Relation Schema (with instance r), and R (r) is decomposed into: R_1, R_2 (with instance: r_1, r_2). If $r = r_1 \bowtie r_2$, then this decomposition is called a Lossless-Join Decomposition.

- How to tell?
  $R_1 \cap R_2 \rightarrow R_1$
  or
  $R_1 \cap R_2 \rightarrow R_2$

Lossless-Join BCNF Decomposition

```
Set ComputeBCNF(R, F)
{
    Result = { R };
    while some Ri in Result and X->Y in F+ violate the BCNF condition
    // in other words, if X->Y in F+ makes Ri NOT in BCNF
    {
        Result = Result - {Ri};
        Add (Ri-(Y-X)) to Result;
        Add {XY} to Result;
    }
    return Result;
}
```

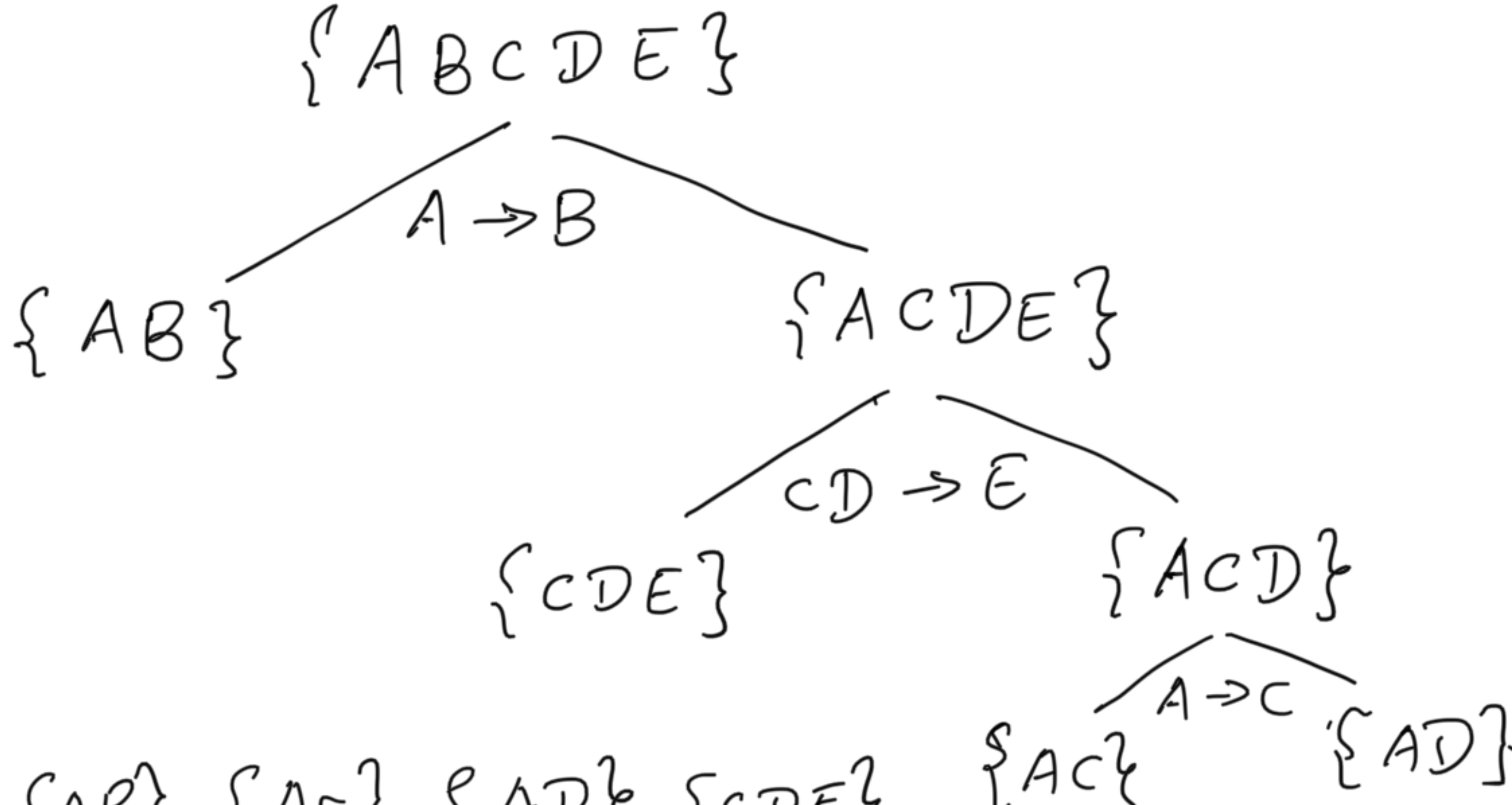
BCNF Decomposition

- There is always a lossless-join decomposition.
- Results depend on sequence of functional dependencies used in the decomposition.

BCNF Example (I)

$R = \{ABCDE\}$

$F = \{A \to B,\ B \to C,\ CD \to E\}$

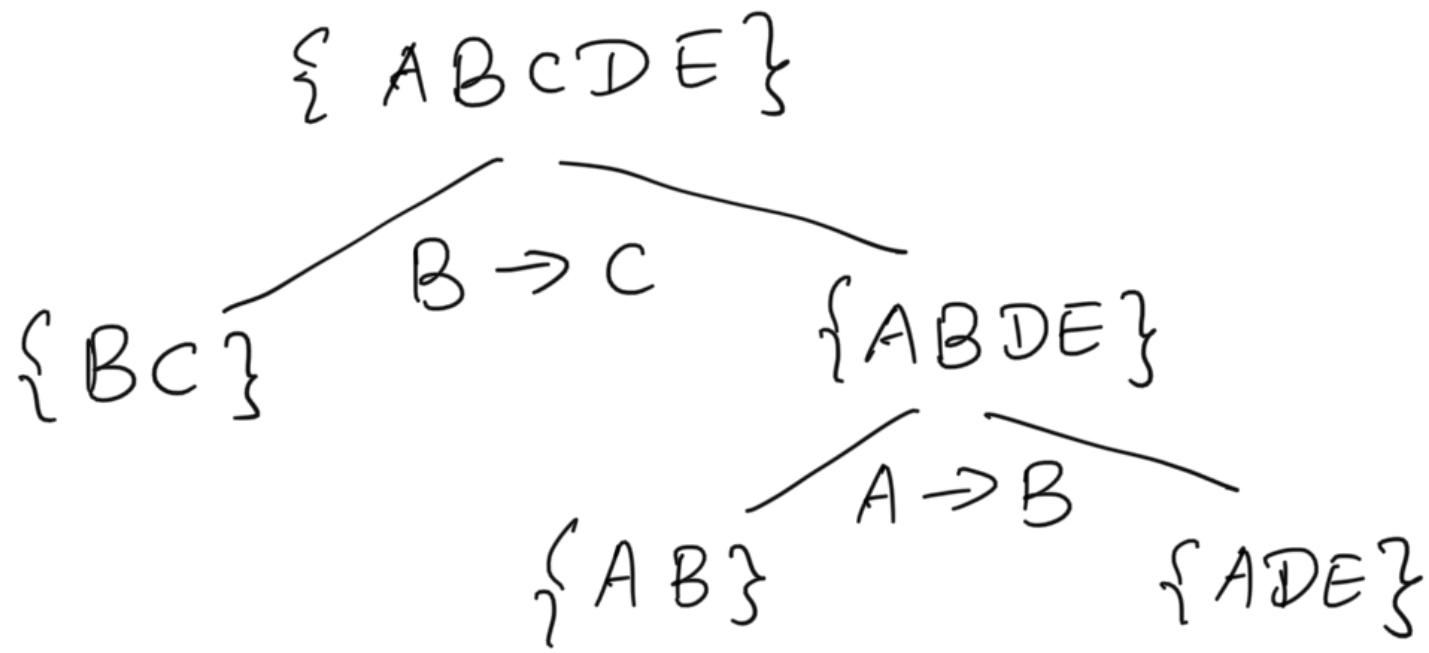


$BCNF = \{AB\}, \{AC\}, \{AD\}, \{CDE\}$

BCNF Example (II)

$R = \{ABCDE\}$

$F = \{A \to B,\ B \to C,\ CD \to E\}$



$\underline{BCNF = \{BC\},\ \{AB\},\ \{ADE\}}$

Dependency Preserving Decomposition

- A functional dependency is inter-relational if it requires joining two tables in order to test it.

- A decomposition $D = \{R_1, \ldots, R_n\}$ of R is dependency preserving if there is an equivalent set F' of FDs, none of which is inter-relational in D. (Note that the assumption here is that none of the functional dependencies in the original set F is inter-relational on original relation R.)

- It is possible that no dependency preserving BCNF decomposition exists.

- Example: $R = \{ABC\}$, and $F = \{AB \rightarrow C, C \rightarrow B\}$.

Third Normal Form

Let R be a relation schema and F a set of functional dependencies.

Schema R is in 3NF if and only if whenever $(X \rightarrow Y) \in F+$ and $XY \subseteq R$, then one of the following conditions is true:

- $(X \rightarrow Y)$ is trivial (i.e., $Y \subseteq X$), or
- X is a superkey of R, or
- each attribute of Y is contained in a candidate key of R.

A database schema $\{R_1, \ldots, R_n\}$ is in 3NF if each relation schema R_i is in 3NF.

3NF Decomposition

- 3NF is looser than BCNF (it allows more redundancy).
- For any relation, there always exists a lossless-join, dependency-preserving decomposition into 3NF relation schema.

Minimal Cover

- A set of functional dependencies G is minimal if
  - every right-hand side of an FD in G is a single attribute.
  - for no $X \rightarrow A$ in G is the set $(G - \{X \rightarrow A\})$ equivalent to G.
  - for no $X \rightarrow A$ in G and $Z \subset X$ is the set $(G - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ equivalent to G.
- For every set of FDs F, there is an equivalent minimal set of FDs (called the minimal cover of F).
- G is the minimal cover of F if G is equivalent to F and G is minimal.

Finding Minimal Cover

- Replace $X \rightarrow YZ$ with the pair $X \rightarrow Y$ and $X \rightarrow Z$.
- Remove $X \rightarrow A$ from F if
  $A \in$ ComputeXClosure(X, F- $\{X \rightarrow A\}$).
- Remove A from the left-hand-side of $X \rightarrow B$ in F
  if $B \in$ ComputeXClosure($X - \{A\}$, F).

Lossless-join and Dependency-preserving 3NF Decomposition

```
// compute the minimal cover of F, each functional dependency forms a relation
// in the decomposed set; If the candidate key is missing, add it to the decomposition
Set Compute3NF(R, F)
{
    Result = { };  // empty set of Result to start with
    G = a minimal cover for F;
    for each (X->Y) in G {
        Add {XY} to Result;
    }
    if none of Ri in Result contains a candidate key of R {
        compute a candidate key K of R;
        Add K to Result;
    }
    return Result;
}
```