



Relational Database Design

Instructor : Nitesh Kumar Jha

niteshjha@soa.ac.in

ITER,S'O'A(DEEMED TO BE UNIVERSITY)

Sept 2018

Multivalued Dependencies

- Suppose we record names of children, and phone numbers for instructors:
 - *inst_child*(ID, child_name)
 - *inst_phone*(ID, phone_number)
- If we were to combine these schemas to get
 - *inst_info*(ID, child_name, phone_number)
 - Example data:
 - (99999, David, 512-555-4321)
 - (99999, William, 512-555-1234)
 - (99999, David, 512-555-1234)
 - (99999, William, 512-555-4321)
- This relation is in BCNF
 - Why?

Multivalued Dependencies (MVDs)

- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

$$\alpha \twoheadrightarrow \beta$$

holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$$\begin{aligned} t_1[\alpha] &= t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \\ t_3[\beta] &= t_1[\beta] \\ t_3[R - \beta - \alpha] &= t_2[R - \beta - \alpha] \\ t_4[\beta] &= t_2[\beta] \\ t_4[R - \beta - \alpha] &= t_1[R - \beta - \alpha] \end{aligned}$$

MVD (Cont.)

- Tabular representation of $\alpha \twoheadrightarrow \beta$

	α	β	$R - \alpha - \beta$
t_1	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$a_{j+1} \dots a_n$
t_2	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$b_{j+1} \dots b_n$
t_3	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$b_{j+1} \dots b_n$
t_4	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$a_{j+1} \dots a_n$

Example

- Let R be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.

Y, Z, W

- We say that $Y \twoheadrightarrow Z$ (Y **multidetermines** Z) if and only if for all possible relations $r(R)$

$\langle y_1, z_1, w_1 \rangle \in r$ and $\langle y_1, z_2, w_2 \rangle \in r$

then

$\langle y_1, z_1, w_2 \rangle \in r$ and $\langle y_1, z_2, w_1 \rangle \in r$

- Note that since the behavior of Z and W are identical it follows that

$Y \twoheadrightarrow Z$ if $Y \twoheadrightarrow W$

Example (Cont.)

- In our example:

$ID \twoheadrightarrow child_name$

$ID \twoheadrightarrow phone_number$

- The above formal definition is supposed to formalize the notion that given a particular value of Y (ID) it has associated with it a set of values of Z ($child_name$) and a set of values of W ($phone_number$), and these two sets are in some sense independent of each other.
- Note:
 - If $Y \rightarrow Z$ then $Y \twoheadrightarrow Z$
 - Indeed we have (in above notation) $Z_1 = Z_2$
The claim follows.

Use of Multivalued Dependencies

- We use multivalued dependencies in two ways:
 1. To test relations to **determine** whether they are legal under a given set of functional and multivalued dependencies
 2. To specify **constraints** on the set of legal relations. We shall thus concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation r fails to satisfy a given multivalued dependency, we can construct a relations r' that does satisfy the multivalued dependency by adding tuples to r .

Theory of MVDs

- From the definition of multivalued dependency, we can derive the following rule:

- If $\alpha \rightarrow \beta$, then $\alpha \twoheadrightarrow \beta$

That is, every functional dependency is also a multivalued dependency

- The **closure** D^+ of D is the set of all functional and multivalued dependencies logically implied by D .
 - We can compute D^+ from D , using the formal definitions of functional dependencies and multivalued dependencies.
 - We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice

Fourth Normal Form

- A relation schema R is in **4NF** with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \twoheadrightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
 - $\alpha \twoheadrightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
 - α is a superkey for schema R
- If a relation is in 4NF it is in BCNF

4NF Decomposition Algorithm

result := {*R*};

done := false;

compute D^+ ;

Let D_i denote the restriction of D^+ to R_i

while (**not** *done*)

if (there is a schema R_i in *result* that is not in 4NF) **then**

begin

 let $\alpha \twoheadrightarrow \beta$ be a nontrivial multivalued dependency
 that holds

 on R_i such that $\alpha \rightarrow R_i$ is not in D_i , and $\alpha \cap \beta = \emptyset$;

result := (*result* - R_i) \cup ($R_i - \beta$) \cup (α, β);

end

else *done* := true;

Note: each R_i is in 4NF, and decomposition is lossless-join

Example

■ $R = (A, B, C, G, H, I)$

$F = \{ A \twoheadrightarrow B$

$B \twoheadrightarrow HI$

$CG \twoheadrightarrow H \}$

■ R is not in 4NF since $A \twoheadrightarrow B$ and A is not a superkey for R

■ Decomposition

a) $R_1 = (A, B)$

(R_1 is in 4NF)

b) $R_2 = (A, C, G, H, I)$
 R_4)

(R_2 is not in 4NF, decompose into R_3 and

c) $R_3 = (C, G, H)$

(R_3 is in 4NF)

d) $R_4 = (A, C, G, I)$
 R_6)

(R_4 is not in 4NF, decompose into R_5 and

• $A \twoheadrightarrow B$ and $B \twoheadrightarrow HI \Rightarrow A \twoheadrightarrow HI$, (MVD transitivity), and

• and hence $A \twoheadrightarrow I$ (MVD restriction to R_4)

e) $R_5 = (A, I)$

(R_5 is in 4NF)

f) $R_6 = (A, C, G)$

(R_6 is in 4NF)

Further Normal Forms

- **Joint dependencies** generalize multivalued dependencies
 - lead to **project-join normal form (PJNF)** (also called **fifth normal form**)
- A class of even more general constraints, leads to a normal form called **domain-key normal form(also called sixth Normal form)**.
- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists.
- Hence rarely used

Overall Database Design Process

- We have assumed schema R is given
 - R could have been generated when converting E-R diagram to a set of tables.
 - R could have been a single relation containing *all* attributes that are of interest (called **universal relation**).
 - Normalization breaks R into smaller relations.
 - R could have been the result of some ad hoc design of relations, which we then test/convert to normal form.

ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization.
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
 - Example: an *employee* entity with attributes *department_name* and *building*, and a functional dependency $department_name \rightarrow building$
 - Good design would have made department an entity
- Functional dependencies from non-key attributes of a relationship set possible, but rare --- most relationships are binary

Denormalization for Performance

- Occasionally database designers choose a schema that has redundant information
- They use the redundancy to improve performance for specific applications.
- The penalty paid for not using a normalized schema is the extra work (in terms of coding time and execution time) to keep redundant data consistent.
- The process of taking a normalized schema and making it non-normalized is called **denormalization**
- Designers use it to tune performance of systems to support time-critical operations.
- A better alternative is to use the normalized schema, and additionally store the join of them as a **materialized view**.

Denormalization for Performance

- May want to use non-normalized schema for performance
- For example, displaying *prereqs* along with *course_id*, and *title* requires join of *course* with *prereq*
- Alternative 1: Use denormalized relation containing attributes of *course* as well as *prereq* with all above attributes
 - faster lookup
 - extra space and extra execution time for updates
 - extra coding work for programmer and possibility of error in extra code
- Alternative 2: use a materialized view defined as $course \bowtie prereq$
 - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

Other Design Issues

- Some aspects of database design are not caught by normalization
- Examples of bad database design, to be avoided:

Instead of *earnings* (*company_id*, *year*, *amount*), use

- *earnings_2004*, *earnings_2005*, *earnings_2006*, etc., all on the schema (*company_id*, *earnings*).
 - ▶ Above are in BCNF, but make querying across years difficult and needs new table each year
- *company_year* (*company_id*, *earnings_2004*, *earnings_2005*, *earnings_2006*)
 - ▶ Also in BCNF, but also makes querying across years difficult and requires new attribute each year.
 - ▶ Is an example of a **crosstab**, where values for one attribute become column names
 - ▶ Used in spreadsheets, and in data analysis tools

END OF CHAPTER

PROOF OF CORRECTNESS OF 3NF ALGORITHM

Correctness of 3NF Decomposition Algorithm

- 3NF decomposition algorithm is dependency preserving (since there is a relation for every FD in F_c)
- Decomposition is lossless
 - A candidate key (C) is in one of the relations R_i in decomposition
 - Closure of candidate key under F_c must contain all attributes in R .
 - Follow the steps of attribute closure algorithm to show there is only one tuple in the join result for each tuple in R_i

Correctness of 3NF Decomposition Algorithm (Cont'd.)

Claim: if a relation R_i is in the decomposition generated by the

above algorithm, then R_i satisfies 3NF.

- Let R_i be generated from the dependency $\alpha \rightarrow \beta$
- Let $\gamma \rightarrow B$ be any non-trivial functional dependency on R_i . (We need only consider FDs whose right-hand side is a single attribute.)
- Now, B can be in either β or α but not in both. Consider each case separately.

Correctness of 3NF Decomposition (Cont'd.)

■ Case 1: If B in β :

- If γ is a superkey, the 2nd condition of 3NF is satisfied
- Otherwise α must contain some attribute not in γ
- Since $\gamma \rightarrow B$ is in F^+ it must be derivable from F_c , by using attribute closure on γ .
- Attribute closure not have used $\alpha \rightarrow \beta$. If it had been used, α must be contained in the attribute closure of γ , which is not possible, since we assumed γ is not a superkey.
- Now, using $\alpha \rightarrow (\beta - \{B\})$ and $\gamma \rightarrow B$, we can derive $\alpha \rightarrow B$ (since $\gamma \subseteq \alpha \beta$, and $B \notin \gamma$ since $\gamma \rightarrow B$ is non-trivial)
- Then, B is extraneous in the right-hand side of $\alpha \rightarrow \beta$; which is not possible since $\alpha \rightarrow \beta$ is in F_c .
- Thus, if B is in β then γ must be a superkey, and the second condition of 3NF must be satisfied.

Correctness of 3NF Decomposition (Cont'd.)

- Case 2: B is in α .
 - Since α is a candidate key, the third alternative in the definition of 3NF is trivially satisfied.
 - In fact, we cannot show that γ is a superkey.
 - This shows exactly why the third alternative is present in the definition of 3NF.

Q.E.D.

Thank You