

5. Database Design Theory

5.1 Drawbacks of a bad design

Consider the following database, we can find drawbacks of a bad design.

u (University)

Prof #	Pname	Grade	Salary	Course #	hours
10	Smith	5	50,000	CS100	36
10	Smith	5	50,000	CS150	36
15	Jones	4	45,000	CS240	18
15	Jones	4	45,000	CS241	18
20	James	6	55,000	CS353	36
21	Smith	6	55,000	CS436	36

- Increased space requirements.

Some information would be stored redundantly.

- Insertion anomalies.

Example:

Inserting a new professor to the table u needs to store undefined entities as Null's. Use of Null's introduces some extra complexity. (25, Johnson, 3, 40,000,Null, Null).

- Deletion anomalies.

Example:

James will no long teach CS353. We have to use null values or delete the tuple of James and CS353.

Side effect of the latter: all information about James also disappears.

- Update anomalies.

Example: The salary for grade 5 is changed into 51,000. Two tuples in the database need to be changed.

If 500 professors are at grade 5, then, 500 different tuples must be updated!

Problem: inefficiency and great risk of inconsistency.

- To avoid the problems, we should split information over a number of different good relations.

How do we know when we have a “good” set of relations?

- Basic idea:

A “good” database design is one where each relation consists of a primary key and a set of mutually independent attributes.

- Relations are classified into classes. Relations in a class satisfy some conditions: – conditions of normal form.

- What is a normal form?

A normal form is kind of relations which satisfy a given set of conditions.

- The process of producing relations in good normal forms is **normalization**.

5.2 Functional Dependencies

- **Definition:**

Attribute Y of a relation r is **functionally dependent** on attribute (set) X of r if whenever two tuples of r have the same X-value, they must also have the same Y-value.

Note: X may be a set of attributes.

- Notation: $X \rightarrow Y$. It means “X functionally determines Y” or “Y is functionally dependent on X”.
- X is the **determinant** of the functional dependency $X \rightarrow Y$, and Y is the **dependant**.
- The notation $X \rightarrow YZ$ is equivalent to the functional dependencies: $X \rightarrow Y$ and $X \rightarrow Z$.

- **Keys:**

- A set of attributes X of a relation schema R is a **superkey** of R if every attribute in R is functionally dependent on X .

Example:

In a relation schema $R(A, B, C, D, E)$, the attribute set AB is a superkey of R if $AB \rightarrow ABCDE$.

- A **candidate key** is a minimal superkey. A set of attributes X of a relation schema R is a candidate key of R if X is a superkey of R and no proper subset of X is a superkey of R .
- The **primary key** of a relation schema R is simply a candidate key designated as being the principal key.

- **Armstrong's Axioms** for FD's

Let X , Y , and Z denote sets of attributes of relation schema R .

- Reflexivity

$$Y \subseteq X \implies X \rightarrow Y$$

- Augmentation

$$X \rightarrow Y \implies XZ \rightarrow YZ$$

- Transitivity

$$(X \rightarrow Y) \ \& \ (Y \rightarrow Z) \implies X \rightarrow Z$$

- Union

$$(X \rightarrow Y) \ \& \ (X \rightarrow Z) \implies X \rightarrow YZ$$

- **(Transitive) Closure of a set of FD's**

- **Definition:**

Let F be a set of FDs. The **closure** of F , denoted by F^+ , is the set of FDs logically implied by the FD's in F .

- **(Transitive) Closure of a set of attributes**

– **Definition:**

Let X be a set of attributes, let F be a set of functional dependencies, the set of all attributes functionally determined by X under F is called **the closure of X under F** and denoted by X^+ .

Algorithm: compute the closure of X under F

Input: $X, F = \{Z_1 \rightarrow Y_1, \dots, Z_n \rightarrow Y_n\}$

Output: X^+

$X^+ := X;$

While (X^+ changes)

 For ($i = 1$ to n)

 If Z_i of $Z_i \rightarrow Y_i$ is in X^+

$X^+ := X^+ \cup Y_i;$

 EndIf

 EndFor

EndWhile

• **Lossless Decompositions**

– **Definition:**

Let U be a relation schema (i.e. a set of attributes). A set of relation schemas R_1, R_2, \dots, R_n is a **decomposition** of U if

$$R_1 \cup R_2 \cup \dots \cup R_n = U$$

(that is, every attribute of U occurs in at least one R_i).

Let u be a relation instance over schema U and

$$r_i = \Pi_{R_i}(u), i = 1, 2, \dots, n.$$

then $\{R_1, R_2, \dots, R_n\}$ is a **lossless decomposition** of U if

$$u = r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$$

for every instance u of U .

Example:

Consider the schema $U(A,B,C)$ and the decomposition $R_1(A,B), R_2(A,C)$.

u

A	B	C
5	10	6
7	10	4

r_1

A	B
5	10
7	10

r_2

A	C
5	6
7	4

$$r_1 \bowtie r_2 = u$$

A	B	C
5	10	6
7	10	4

– It is always the case that

$$r_1 \bowtie r_2 \bowtie \dots \bowtie r_n \supseteq u$$

that is, for every tuple $t \in u$, it holds that

$$t \in r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$$

– It is not always the case that

$$u \supseteq r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$$

In other words, there may exist $t \in r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$ but $t \notin u$. Such tuples are called **spurious** tuples and the decomposition is said to be **lossy**.

Example:

Consider the schema $U(A,B,C)$ and the decomposition $R'_1(A,B), R'_2(B,C)$.

A	B	C
5	10	6
7	10	4

A	B
5	10
7	10

B	C
10	6
10	4

u

r'_1

r'_2

$$r'_1 \bowtie r'_2 \neq u$$

A	B	C
5	10	6
5	10	4
7	10	6
7	10	4

- **Lossless Decomposition Theorem:**

It is impossible to exam **all** the instances of a relation schema for testing if a decomposition is lossless.

We can use the following method for the test.

Let $\{R_1, R_2\}$ be a decomposition of U . The decomposition is lossless if and only if at least one of the following FD's is in F^+

$$R_1 \cap R_2 \rightarrow R_1,$$

or

$$R_1 \cap R_2 \rightarrow R_2.$$

Example:

$U = (A, B, C)$, $R_1 = (A, B)$, $R_2 = (B, C)$, and $F = \{ A \rightarrow B, A \rightarrow C \}$.

We have

$$R_1 \cap R_2 \rightarrow R_1 : B \rightarrow AB$$

$$R_1 \cap R_2 \rightarrow R_2 : B \rightarrow BC$$

Neither of the two FD's are in F^+ (because $B^+ = \{B\}$) and hence the decomposition is not lossless.

Example:

$U = (A, B, C)$, $R'_1 = (A, B)$, $R'_2 = (A, C)$, and $F = \{A \rightarrow B, A \rightarrow C\}$.

$$R'_1 \cap R'_2 \rightarrow R'_1 : A \rightarrow AB$$

$$R'_1 \cap R'_2 \rightarrow R'_2 : A \rightarrow AC$$

Both FD's are in F^+ , and hence the decomposition is lossless.

• **Dependency Preserving Decompositions**

Example:

$U(A,B,C)$, $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$

- Consider the decomposition $R_1(A,B)$, $R_2(B,C)$.
 - * The FD's $F_1 = \{A \rightarrow B\}$ and $F_3 = \{B \rightarrow C\}$ can easily be enforced when modifying instances of R_1 or R_2 .
 - * The FD $A \rightarrow C$ is automatically satisfied when $A \rightarrow B$ and $B \rightarrow C$ are satisfied, and require no further checking.
 - * Such a decomposition is **dependency preserving**.
- Consider the decomposition $R_3(A,B)$, $R_4(A,C)$.
 - * The FD's $F_1 = \{A \rightarrow B\}$ and $F_2 = \{A \rightarrow C\}$ can easily be enforced.
 - * However, $B \rightarrow C$ is not automatically satisfied when $A \rightarrow B$ and $A \rightarrow C$ are satisfied.
 - * $B \rightarrow C$ has become an **interrelational constraint** (or unenforceable FD), which cannot be tested without a join.

Example:

$$r_3 \qquad r_4 \qquad r_3 \bowtie r_4$$

$A \rightarrow B$ and $A \rightarrow C$ are satisfied but $B \rightarrow C$ is not.

Note that the decomposition is lossless but not dependency preserving.

A	B
5	10
6	10

A	C
5	15
6	30

A	B	C
5	10	15
6	10	30

- **Desirable Properties of a Decomposition**

1. Lossless
2. Dependency preserving
3. No redundancy
4. Minimal number of relations

5.3 Boyce-Codd Normal Form (BCNF)

- **Definition:**

A relation schema R is in BCNF if, for every FD $X \rightarrow Y$ on R (i.e. $X, Y \subseteq R$), at least one of the following conditions holds:

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

- **Definition*:**

A relation schema R is in BCNF if and only if every determinant of its non-trivial FDs is a superkey of R .

- A relation is in BCNF if its schema is in BCNF.
- A database is in BCNF if all of its relations are in BCNF.
- If a relation schema R is not in BCNF it can be decomposed into a set of BCNF relation schemas by the following algorithm:

Algorithm: decomposition into BCNF schemas

Input: R, F
Output: S = a set of BCNF relation schemas

```
S := {R};  
While (∃Ri ∈ S | Ri is a non-BCNF schema)  
  If ( X → Y is a nontrivial FD on Ri such  
    that X → Ri is not in F+,  
    and X ∩ Y = ∅)  
    S := (S - Ri) ∪ (Ri - Y) ∪ (X, Y);  
  EndIf  
EndWhile
```

Example:

Consider relation R(A,B,C,D,E,F) and FD's F = {A → BC, D → AEF }.

- R is not in BCNF because A is not a superkey of R (A⁺={ABC}).
- R can be Decomposed using the above algorithm

$$S = \{R_1(A, D, E, F), R_2(A, B, C)\}$$

- R₁ is in BCNF. The only nontrivial FD on R₁ is D → AEF, and D is a superkey of R₁.
- R₂ is in BCNF. The only nontrivial FD on R₂ is A → BC, and A is a superkey of R₂.
- The algorithm above guarantees losslessness, but not dependency preservation. As shown by the example below, a dependency preserving BCNF decomposition may not exist.
- The number of relations in a decomposition may vary depending on what FD's are used for generating the decomposition.

Example:

Consider relation R(A,B,C) and FD's F = {AB → C, C → B }.

R is not in BCNF because C is not a superkey of it.

decomposition:

$$\begin{aligned}
S &= \{R\} \\
\text{For } R, \text{ use } C \rightarrow B \\
S &= (S-R) \cup (R-B) \cup (C, B) \\
&= \phi \cup (A, C) \cup (C, B) \\
&= \{(A, C), (C, B)\}
\end{aligned}$$

- Every decomposition of R must fail to preserve $AB \rightarrow C$, because it involves all attributes of R and is not implied by $C \rightarrow B$.
- We cannot always achieve lossless join, dependency preservation and BCNF simultaneously.
- Therefore, a weaker form of normalization – 3NF is introduced.

5.4 Third Normal Form (3NF)

- 3NF is weaker than BCNF but a lossless, dependency preserving decomposition into 3NF relations always exists.

- **Definition:**

Let R be a relation schema, let X be any set of attributes of R, and let Y be any single attribute of R. Then R is in 3NF if and only if, for every FD $X \rightarrow Y$ on R (i.e. $X, Y \subseteq R$), at least one of the following conditions holds:

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

- **Definition*:**

A relation schema is in 3NF if and only if the **nonkey** attributes (if any) are

- mutually independent, and
- irreducibly dependent on the primary key.

- **Example:**

$R(A, B, C), F = \{ AB \rightarrow C, C \rightarrow B \},$

R is in 3NF, because AB is a candidate key of R .

- The following algorithm computes a lossless, dependency preserving decomposition into 3NF relation schemas:

Algorithm: production of 3NF relation schemas

Input: R, F

Output: $S =$ a set of 3NF relation schemas

```
i := 0; S := { };
For each FD  $X \rightarrow Y$  in  $F$  do
  If ( $\nexists R_j \in S \mid XY \subseteq R_j$  for  $1 \leq j \leq i$ )
    i := i+1;
     $R_i := (XY)$ ;
     $S := S \cup \{R_i\}$ ;
  EndIf
EndFor
If ( $\nexists R_j \in S \mid R_j$  contains a candidate key for  $R$ )
  i := i+1;
   $R_i :=$  (any candidate key for  $R$ );
   $S := S \cup \{R_i\}$ ;
EndIf
```

5.5 First Normal Form and Second Normal Form

- **First Normal Form (1NF)**

- **Definition:**

A relation schema R is in the first normal form (1NF) if and only if all underlying domains contains single atomic values only, (not arrays, structures, etc.).

- **Example:**

before normalization:

after normalization:

Prof #	Pname	Grade	Salary	Course #	hours
10	Smith	5	50,000	CS100	36
				CS150	36
15	Jones	4	45,000	CS240	18
				CS241	18
20	James	6	55,000	CS353	36
21	Smith	6	55,000	CS436	36

Prof#	Pname	Grade	Salary	Course #	hours
10	Smith	5	50,000	CS100	36
10	Smith	5	50,000	CS150	36
15	Jones	4	45,000	CS240	18
15	Jones	4	45,000	CS241	18
20	James	6	55,000	CS353	36
21	Smith	6	55,000	CS436	36

- A 1NF relation may have **partial dependency** which may cause anomalies:
 - * Insertion anomalies:

Insertion of information about a new professor who has no course to teach will result in an incomplete primary key.
 - * Deletion anomalies:

Professor James no longer teaches CS353. The whole tuple of James may be deleted.

• **Second Normal Form (2NF)**

- **Definition:**

A relation schema R is in the second normal form (2NF) if and only if it is in 1NF and every nonkey attribute is fully dependent on the primary key.

- **Example**

Prof#	Pname	Grade	Salary
10	Smith	5	50,000
15	Jones	4	45,000
20	James	6	55,000
21	Smith	6	55,000

Prof#	Course #	hours
10	CS100	36
10	CS150	36
15	CS240	18
15	CS241	18
20	CS353	36
21	CS436	36

- Second normal form is violated when a nonkey attribute depends on a proper subset of a key.
- In a 2NF relation, partial dependency has been removed. However, it may have **transitive dependency**, which may cause anomalies.
 - * Insertion anomalies:

We can not insert a new grade and its salary until we have a professor at that grade.
 - * Deletion anomalies:

If all the professor at a certain grade have been retired, the information about that grade disappears.

* Update anomalies:

If the salary of a grade is changed, we have to update all the tuples with that grade.

- **Third Normal Form (3NF)**

- **Definition:**

A relation R is in third normal form (3NF) if and only if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key.

- The third normal form is violated when a nonkey attribute is a fact about another nonkey attribute.

- **Example**

Prof#	Pname	Grade
10	Smith	5
15	Jones	4
20	James	6
21	Smith	6

Grade	Salary
4	45,000
5	50,000
6	55,000

Prof#	Course #	hours
10	CS100	36
10	CS150	36
15	CS240	18
15	CS241	18
20	CS353	36
21	CS436	36

5.6 Merging Relations

- Relations having the same primary key should be merged into one relation.

- When merging relations we must consider the problems of the meanings of data, mainly:
 - **Synonyms:** Two or more attributes have different names but the same meaning.
 - **Homonyms:** A single attribute may have different meaning in different relations.
- When two 3NF relations are merged to form a single relation, transitive dependencies may result.

Reading: Sections 3.4, 3.5 and 3.6.