Sandeep Singh Bundra

CSE 3 Sem DBMS

9 Normalization

9.1 INTRODUCTION

The basic objective of logical modeling is to develop a "good" description of the data, its relationships and its constraints. For the relational model, this means that we must identify a suitable set of relations. However, the task of choosing the relations is a difficult one because there are many options for the designer to consider. This lesson is devoted to explaning some methods of improving logical design. The techniques presented here are based on a large body of research into the logical design process called normalization.

The process of normalization was first developed by **E.F. Codd**. Normalizing a logical database design involves organizing the data into more than one table. Normalization improves the performance by reducing **redundancy** in database tables. The basic objectives of normalization are to reduce redundancy, which means that information to be stored only once in relation. Another serious problem with using the relation as base relation is the problem of **modification anomalies.** These can be classified into insertion anomalies, deletion anomalies and update anomalies.

9.2 MODIFICATION ANOMALIES

Changing the data in some relations can have undesirable consequences called modification anomalies. Anomalies can be eliminated by changing the structure of the relation. There are three types of modification anomalies.

Consider the ACTIVITY relation.

The attributes of the ACTIVITY relations are SID (Student identifier), Activity, and Fee. The meaning of a row is that a student engages in the named activity for the specified fee.

ACTIVITY

SID	~~ACTIVITY~~ Game	FEE
100	SKING	200
150	SWIMMING	150
175	SQUASH	50
200	SWIMMING	150

FIGURE 9.1

1. **Deletion Anomaly**

   Suppose that each activity has a fixed fee that is the same for all students. If we delete the tuple for the student 100, we lose not only the fact that student 100 is skier, but also the fact that skiing costs $200. This is called a delation anomaly. In deleting the facts about one entity (that student 100 is a skier), we inadvertently deleted facts about another entity (that skiing cost $200). We loose facts about two entities with one deletion.

2. **Insertion Anomaly**

   Suppose we want to store the fact that scuba driving costs $175. We cannot enter this data into the ACTIVITY relation until a student takes up scuba. This is called an insertion anomaly because we cannot insert a fact about one entity until we have an additional fact about another entity.

3. **Updation anomaly**

   Suppose we want to change the fee of swimming activity from $150 to For this change, we are faced with either the problem of searching ACTIVITY relation to find every tuple with swimming activity and change the fee from $150 to $ 250 or the possiblity of producing an inconsistant result (the fee for swimming may be $150 in one place and $250 in another). This is called a Updation anomaly.

9.3 PROBLEMS ARISING OUT OF BAD DATABASE DESIGN

Problems arising out of bad database design are :

1. Redundancy
2. Wastage of storage space due to redundancy.
3. Modification anomalies.
4. Complex structure of tables

Hence, main **benefits** of normalization are :

- Reduced redundancy
- required less memory space
- Free from modification anomalies
- Simplicity
- Flexibility

9.4 NORMAL FORMS

A relation is ~~seed~~ said to be in particular normal form if it satisfies a certain specified set of constraints. Let us consider First Normal Form.

9.4.1 First Normal Form

A relation is, in first normal form if and only if all underlying domains contain atomic values only.

EXAMPLE 1

The relation ACTIVITY in fig. 9.1 is in first normal form. As we saw earlier, the ACTIVITY relation has modification anomalies. The problem with this relation is that it has a dependency involving only part of the key. The key is the combination (SID, Activity), but the relation contains a dependency, Activity → Fee. The determinant of this dependency (Activity) is only part of the key (SID, Activity). The solution of these problems replace relation ACTIVITY by the two relations STU-ACT (SID, Activity) and ACT-COST (Acativity, Fee) fig. 9.2 shows the STU-ACT & ACT-COST relations.

STU-ACT

SID	Activity
100	SKIING
150	SWIMMING
175	SQUASH
200	SWIMMING

ACT-COST

Activity	Fee
SKIING	200
SWIMMING	150
SQUASH	50

FIGURE 9.2

EXAMPLE 2

Consider the relation FIRST (S#, STATUS, CITY, P#, QTY.) shown in fig. 9.3.

FIRST

S#	STATUS	CITY	P#	QTY.
S1	30	Delhi	P1	100
S1	30	Delhi	P2	125
S1	30	Delhi	P3	130
S1	30	Delhi	P4	115
S2	10	Karnal	P1	200
S2	10	Karnal	P2	215
S3	40	Rohtak	P1	200
S4	30	Delhi	P4	200
S4	30	Delhi	P5	300

FIGURE 9.3

Here we assume that each supplier has a unique supplier number (S#), exactly one status code and location. A supplier can supply different parts. Further, we assume that STATUS is functionally dependent on CITY, therefore, all Delhi supliers must have a status of 30. Fig. 9.4 shows the functional dependency diagram for this relation. The key of FIRST relation is the combination (S#, P#)

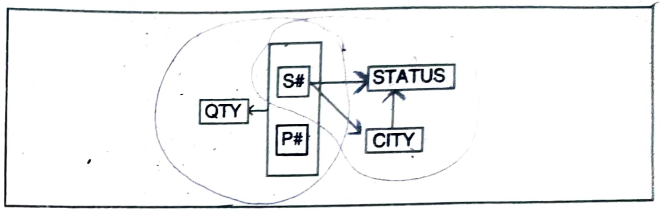


FIGURE 9.4

It is clear that relation FIRST is in first normal form. The relaion FIRST has the following modification anomalies.

(a) Deletion anomaly

If we delete the FIRST tuple with S# value S3 and P# value P1, we lose the information that S3 is located in Rohtak. Hence the relation FIRST has a deletion anomaly.

(b) Insertion anomaly

We cannot enter the fact that a particular supplier is located in a particular supplier city until that supplier supplies at least one part. So, we cannot enter the fact that supplier S5 is located in Jind until S5 supplies at least one part.

(c) Updation anomaly

It supplier S1 moves from Delhi to Ambala, we are faced with either the problem of searching the FIRST relation to find every tuple connecting S1 and Delhi and change it or the posiblity of producing an inconsistent result (the city for S1 may be given as Delhi in one place and Ambala in another).

The solution of these problems is to replace the relation FIRST by the two relation SECOND (S# STATUS, CITY) and SP (S#, P#, QTY). Fig. 9.5 shows sample tabulation corresponding to the data values of figure 9.3

Combined

SECOND

S#	STATUS	CITY
S1	30	Delhi
S2	10	Kamal
S3	40	Rohtak
S4	30	Delhi

SP

S#	P#	QTY
S1	P1	100
S1	P2	125
S1	P3	130
S1	P4	115
S2	P1	200
S2	P2	215
S3	P1	200
S4	P4	200
S4	P5	300

S#
P# → City

$\underline{S\#, P\#} \rightarrow QTY$
Combined

FIGURE 9.5

Fig. 9.6 shows the functional dependency diagrams for these two relations.

Transitive property $A \rightarrow B$
$B \rightarrow C$
thus
$A \rightarrow C$

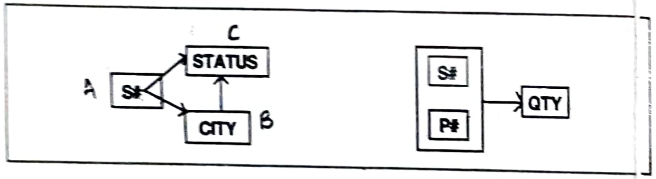


FIGURE 9.6

9.4.2 Second Normal Form

A relation is in second normal form (2 NF) if it is in INF and every nonkey attribute is fully dependent on the key.

If the key is a single attribute, then the relation is automatically in the 2 N F. For example, STU-ACT and ACT - COST relations are in 2 N F because they have single - attribute key. Relation SECOND and SP are also both is 2 N F.

The relation SECOND still causes problems due to transitive functional dependence on S#. In SECOND, S# determines CITY and CITY determines STATUS, thus, transitively, S# determines STATUS. Problems of SECOND relations are :

1. **Deletion anomaly**

If we delete the SECOND tuple for a particular city, we distroy not only information for the supplier concerned but also the information that that c has that particular status value. For example, if we delete the SECOND tuple for S2, we lose the information that the status for Kamal is 10.

2. **Insertion anomaly**

We can not enter the fact that a particular city has a particular status value. For example we can not state that any supplier in Ambala must have a status 70 until we have some supplier located in that city.

3. **Updation anomaly**

If we want to change the status value for Delhi from 30 to 50, we are faced with either the problem of searching the SECOND relation to find every tuple for Delhi or the possiblity of producing an inconsistent result. Again the solution of the problems is to replace the SECOND relation by two relations.

SC(S#, CITY) and CS (CITY, STATUS)

Fig. 9.7. shows the tabulation corresponding to the data values of SECOND.

SC

S#	CITY
S1	Delhi
S2	Kamal
S3	Rohtak
S4	Delhi

CS

CITY	STATUS
Delhi	30
Kamal	10
Rohtak	40

FIGURE 9.7

9.4.3 Third Normal Form

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

For example, Relation SC, CS and SP are in 3 NF but the relation SECOND is not in 3 NF. because it satisfies transitive property

Roll / Bvr

Roll - T... sum

9.4.4 Boyce-Codd Normal Form (BCNF)

Unfortunately, even relations in third normal form can have anomalies. Consider the ADVISOR (SID, Major, Fname) relation in Fig. 9.8. Suppose the requirements underlying this relation are that a student (SID) can have one or more majors, a major can have several faculty members (Fname) as advisors, and a faculty member (Fname) advises in only one major area.

Key (primary) : (SID, Major) ✓

Key (candidate) : (SID, Fname)

Functional dependencies : Fname Major

Sid m — Fname

ADVISOR advisor

SID	Major	Fname
100	MATH	CAUCHY
150	PHYSICS	JUNG
200	MATH	RIEMANN
250	MATH	CAUCHY
200	PHYSICS	MANISH

Tea

FIGURE 9.8

Relation in 3 NF but not in BCNF.

Since students can have several majors, SID does not determine Major. Further, since students can have several advisors, SID does not determine Fname. Thus SID cannot be a key.

The combination (SID, Major) determines Fname. the combination (SID, Fname) also determines Major. Hence, either of the combinations can be a key. Two or more attributes or attribute collection that can be a key are called candidate keys. Whichever of the candidates is selected to be the key is called primary key.

Fname determines Major. (Any faculty member advises in only one major. Therefore, given the Fname we can determine the Major.) Thus Fname is a determinant.

ADVISOR is in first normal form by definition. It is in second normal form since any non-key attributes are dependent on the entire key (no matter which composite key we select). Further, it is in 3NF because it has no transitive dependences. Inspite of all this, however, it has modification anomalies.

Suppose student 150 drops out of school. If we delete student 150's tuple, we loose the fact that JUNG advises in Physics. This is a deletion anomaly. Similarly how can we store the fact that RAKESH advises in economics ? We cannot insert this fact until a student majors in economics. This is an insertion anomaly.

Situations like this lead to the definition of Boyce-Codd normal form (BCNF) :

A Relation is in BCNF if every determinant is a candidate key.

ADVISOR is not in BCNF, it has a determinant Fname, that is not a candidate key.

ADVISOR can be decomposed into two relations STU-ADV (SID, Fname)and ADV-SUBJ (Fname, Major) which have no anomalies. Hence relations STU-ADV and ADV-SUBJ are in BCNF.

STU-ADV (SID, Fname)
Key : (SID, Fname)

Fname represents faculty name

SID	Fname
100	CAUCHY
150	JUNG
200	RIEMANN
250	CAUCHY
200	MANISH

ADV-SUBJ (Fname, Subject)
Key : (Fname)

Fname	Major
CAUCHY	MATH
JUNG	PHYSICS
RIEMANN	MATH
MANISH	PHYSICS

FIGURE 9.9 (RELATIONAL IN BCNF)

9.4.5 Fourth Normal form (4NF)

Consider the relation STUDENT (SID, Major, Activity) participate attributes. Suppose that students can enroll in several different majors and in several different activities. Since this is so, the only key is the combination of (SID, Major, Activity).

STUDENT

SID	Major	Activity
200	MUSIC	SWIMMING
200	ACCOUNTING	SWIMMING
200	MUSIC	TENNIS
200	ACCOUNTING	TENNIS
250	MATH	JOGGING

FIGURE 9.10

SID	Major	Activity
200	Music	Swimming
	Accounting	Tennis
250	Maths	Jogging

Student 200 majors in music and accounting. He also participates in swimming and tennis. Student 250 majors only in math and particularly in jogging. The relation STUDENT has **multivalued dependency**. Multivalued dependencies lead to update anomalies. Let us see why.

First, note the data redundancy in Fig. 9.10. Student 200 has four rows. Each one shows one of his majors paired with one of his activities. If the data were stored any other way we had only two rows : one for music and swimming and one for accounting and tennis then the implications would be misleading. It would appear that student 100 swam only when he was a music major and played tennis only as an accounting major. That interpretation is illogical ! His majors and his activities are completely independent of one another. So in this relation, we store all the combinations of majors and activities. But data redundancy is not the important problem with multivalued dependencies.

Suppose that student 200 decides to sign up for skiing, so we add the row [200, MUSIC, SKIING]. This implies that the skies as a music major, but not as an accounting major. In order to keep the consistent we must add one row for each of her majors paired with skiing. Thus we must also add the row [100, ACCOUNTING, SKIING]. This is an update anomaly.

In general, multivalued dependencies exists when there are atleast three attributes in a relation, atleast one of them is multivalued, and the values of the two attributes depend only on the other attributes .

In other words, in a relation R (A, B, C) a multivalued dependency exists if

(1) A leads to multiple values of C, and B, and/or

(2) A leads to multiple values of C, and

(3) B and C are independent of each other.

As we have seen in previous example that SID leads to multiple values of major and multiple value of Activity Major and Activity are independent of one another. These multivalued dependencies can be indicated as follows :

SID $\twoheadrightarrow$ Major

SID $\twoheadrightarrow$ Activity

$\twoheadrightarrow$ Symbol of multivalued dependencies

This relation is in BCNF (2NF because it is all key, 3NF because it has no non-key determinants). However, as we have seen, it has anomalies.

We can eliminate these anomalies by decomposed STUDENT relation into two relations

STU-MAJOR (SID, Major) and STU-ACT (SID, Activity) as show in fig. 9.11.

Key : (SID, Major)

SID	[redacted] Major
200	MUSIC
200	ACCOUNTING
250	MATH

Key : (SID, Activity)

SID	[redacted] Activity
200	SKING
200	SWIMMING
200	TENNIS
250	JOGGING

FIGURE 9.11 (ELIMINATION OF MULTIVALUED DEPENDENCY)

This observation leads to the definition of 4 NF :

A relation is in 4 NF if it is in BCNF and has no multivalued dependencies.

Projection Join Normal Form Or PJ/NF

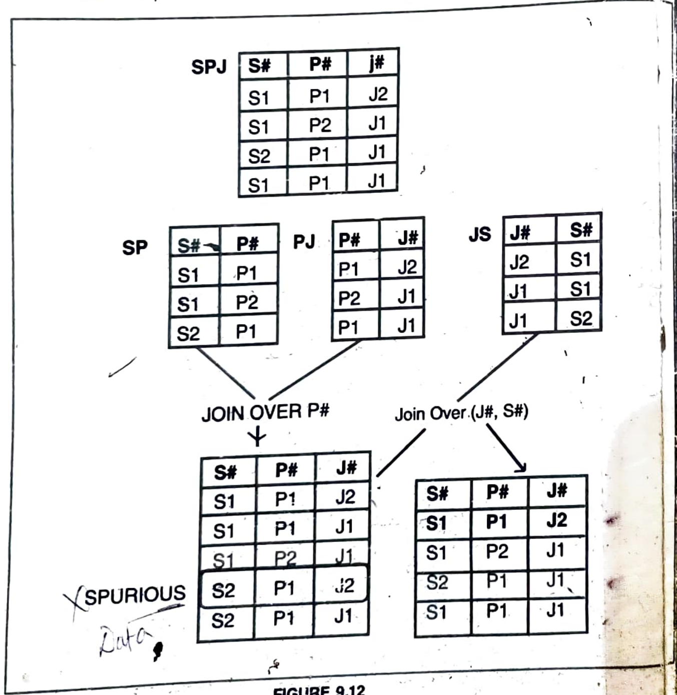
9.4.6 Fifth Normal Form (5NF)

Consider relation SPJ shown below this relation is "all key" and involves no FD, or MVDs and so is 4 NF. Fig. 9.12 shows

(a) the three Projections SP, PJ & JS of SPJ and

(b) the effect of joining SP & PJ over P# & then joining the result & JS over J#, S#). Note that the result of the first join is to produce a copy of the original SPJ plus one spurious tuple & that the effect of the second. Join is to elimir that tuple.

The statement, SPJ is equal to join of its three projections SP, PJ & JS is equivalent to the statement,



SPJ

S#	P#	j#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

SP

S#	P#
S1	P1
S1	P2
S2	P1

PJ

P#	J#
P1	J2
P2	J1
P1	J1

JS

J#	S#
J2	S1
J1	S1
J1	S2

JOIN OVER P#

S#	P#	J#
S1	P1	J2
S1	P1	J1
S1	P2	J1
S2	P1	J2
S2	P1	J1

Join Over (J#, S#)

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

FIGURE 9.12

If the pair $<s_1,p_1>$ appears in SP

and the pair $<p_1,j_1>$ appears in PJ

and the pair $<j_1, s_1>$ appears in JS

then the triple $<s_1, p_1, j_1>$ appears in SPJ

This is a constrant, just like an FD or an MVD. This constraint is called a join dependency (J.D.). So in this example, we say that SPJ satisfies the join dependency "SP, PJ, JS)".

DEFINITION OF JD

relation on R satisfies the JD * (x,y, ---, z) if and only if it is the join of its projections on x, y, ---, z. Where x, y, --- , z are subsets of the set of attributes of R.

Relations SPJ suffers from a number of problems over update operations e.g. consider.

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1

FIGURE 9.13

(1) If $<S_2, P_1, J_1>$ inserted. $<S_1, P_1, J_1>$ must also be inserted

(2) Yet converse is not true.

(Note reason is that above constraint on SPJ is equivalent to IF $<S_1, P_1, J_2>$, $<S_2, P_1, J_1>$, $<S_1, P_2, J_1>$ appear in SPJ then $<S_1,P_1, J_1>$ also appears in SPJ, so it is due to presence of JD). This problem is removed when it is three decomposed.

DEFINITION OF FIFTH NORMAL FORM (5NF)

A relation R is in fifth normal form (5 NF)- also called projection-join normal form (PJ/NF) - If only if every join dependency in R is implied by the candidate keys of R.