LECTURE 4

Normalization

Learning Goals

What is normalization?

Anomalies

- Normalization: 1NF
- Normalization: 2NF
- Normalization: 3NF

What is normalization?

What are the goals of normalization?

- Eliminate redundant data.
- Ensure data dependencies make sense.

What is the result of normalization?

What are the levels of normalization?

Always follow purist's approach of normalization?

NO

Normalization

Before we begin our discussion about the normal forms, it's important to understand that what we will discuss are guidelines and guidelines only i.e. not a dejure standard. Sometimes (especially in a DSS environment), it becomes necessary to drift from this purist approach to meet practical business requirements of performance. However, when deviation take place, it's extremely important to evaluate any possible consequences these deviations would cause and the corresponding inconsistencies and anomalies this is what de-normalization is about.

Normalization is the process of efficiently organizing data in a database by decomposing (splitting) a relational table into smaller tables by projection. There are basically two goals of normalization as follows:

1. Eliminate redundant data (for example, storing the same data in more than one table)

2. Ensuring data dependencies make sense (only storing related data in a table).

Both of these are worthy goals, as they reduce the amount of space a database consumes, and ensure that data is logically stored and is in third normal form (3NF). The database community has developed a series of guidelines or benchmarks for ensuring that databases are indeed normalized. These goals are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). The first two normal forms are Intermediate steps to achieve the goal of having all tables in 3NF. Note that in order to be correct, decomposition must be lossless i.e. the new tables can be recombined by a natural join to recreate the original table without creating any false or redundant data and without any loss of any data/information.

Consider a student database system to be developed for a multi-campus university, such that it specializes in one degree program at a campus i.e. BS, MS or PhD.

SID	Degree	Campus	Course	Marks
1	BS	Islamabad	CS-101	30
1	BS	Islamabad	CS-102	20
1	BS	Islamabad	CS-103	40
1	BS	Islamabad	CS-104	20
1	BS	Islamabad	CS-105	10
1	BS	Islamabad	CS-106	10
2	MS	Lahore	CS-101	30
2	MS	Lahore	CS-102	40
3	MS	Lahore	CS-102	20
4	BS	Islamabad	CS-102	20
4	BS	Islamabad	CS-104	30
4	BS	Islamabad	CS-105	40

SID: Student ID

Degree: Registered as BS or MS student

Campus: City where campus is located

Course: Course taken

Marks: Score out of max of 50

Part of example student database system

We consider the case of development of a student DBMS for a multi-campus university i.e. with campuses in many cities. There is a head office of the University (say) in Islamabad which would

like to get an overall picture of all the students at all the campuses of the university. A typical detail table is shown with a brief description of the fields. These are in no way the only fields, but sufficient to briefly review the basic forms of normalization.

In order to uniquely associate marks with course and the student, a composite primary key composed of SID and campus is used.

Database system in 1NF

SID	Degree	Campus	Course	Marks
1	BS	Islamabad	CS-101	30
1	BS	Islamabad	CS-102	20
1	BS	Islamabad	CS-103	40
1	BS	Islamabad	CS-104	20
1	BS	Islamabad	CS-105	10
1	BS	Islamabad	CS-106	10
2	MS	Lahore	CS-101	30
2	MS	Lahore	CS-102	40
3	MS	Lahore	CS-102	20
4	BS	Islamabad	CS-102	20
4	BS	Islamabad	CS-104	30
4	BS	Islamabad	CS-105	40

Table FIRST

Part of example student database system in 1NF

Although the table is shown in 1NF i.e. it contains atomic values, there are no repeating values, there is no aggregation, yet it contains redundant data. For example, information about the student's degree and campus location is repeated for every course taken.

Redundancy causes what are called *update anomalies*. Update anomalies are problems

Those arise when records are inserted, deleted, or updated in the database. For example, the following anomalies can occur:

Normalization: 1NF

Update Anomalies

INSERT. Certain student with SID 5 got admission in a different campus (say) Karachi cannot be added until the student registers for a course.

DELETE. If student graduates and his/her corresponding record is deleted, then all information about that student is lost.

UPDATE. If student migrates from Islamabad campus to Lahore campus (say) SID = 1, then six rows would have to be updated with this new information.

INSERT. The fact that a certain student with SID 5 got admission in a different campus (say) Karachi cannot be added until the student registers for a course.

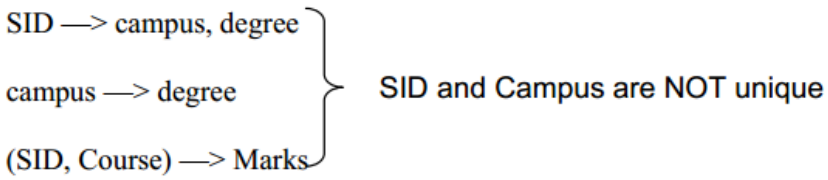
DELETE. If student graduates and his/her corresponding record is deleted, then all information about that student is lost.

UPDATE. If student migrates from Islamabad campus to Lahore campus (say) SID = 1, then six rows would have to be updated with this new information.

Normalization: 2NF

Every non-key column is fully dependent on the PK.

FIRST is in 1NF but not in 2NF because degree and campus are functionally dependent upon only on the column SID of the composite key (SID, course). This can be illustrated by listing the functional dependencies in the table:



To transform the table FIRST into 2NF we move the columns SID, Degree and Campus to a new table called REGISTRATION. The column SID becomes the primary key of this new table.

The definition of second normal form states that only tables with composite primary keys can be in 1NF but not in 2NF.

A relational table is in second normal form 2NF if it is in 1NF and every non-key column is fully

dependent upon the primary key. That is, every non-key column must be dependent upon the entire primary key. The table in the last slide is in 1NF but not in 2NF because degree, course and even marks are functionally dependent upon the SID and the campus.

FIRST is in 1NF but not in 2NF because degree and campus are functionally dependent upon only on the column SID of the composite key (SID, course). This can be illustrated by listing the functional dependencies in the table:

SID —> campus, status
Campus —> degree
(SID, Course) —> Marks
PERFORMANCE in 2NF as (SID, Course) uniquely identify Marks

Part of example student database system in 2NF

To transform the table FIRST into 2NF we move the columns SID, Degree and city to a new table called REGISTRATION. The column SID becomes the primary key of this new table.

Normalization: 2NF

Presence of modification anomalies for tables in 2NF. For the table REGISTRATION, they are:

INSERT: Until a student gets registered in a degree program, that program cannot be offered!

DELETE: Deleting any row from REGISTRATION destroys all other facts in the table.

Why there are anomalies?

The table is in 2NF but NOT in 3NF

Tables in 2NF but not in 3NF still contain modification anomalies. In the example of REGISTRATION, they are:

INSERT. The fact that a particular campus has a certain degree (Peshawar campus decides to run a PhD program) cannot be inserted until there is a student registered for PhD in the campus. DELETE. Deleting any row from REGISTRATION table destroys the degree information about

the campus as well as the association between student and campus.

Normalization: 3NF

All columns must be dependent only on the primary key.

Table PERFORMANCE is already in 3NF. The non-key column, marks, is fully dependent upon the primary key (SID, degree).

REGISTRATION is in 2NF but not in 3NF because it contains a transitive dependency.

A transitive dependency occurs when a non-key column that is a determinant of the primary key is the determinate of other columns.

The concept of a transitive dependency can be illustrated by showing the functional dependencies:

REGISTRATION.SID —> REGISTRATION. Degree

REGISTRATION.SID —> REGISTRATION. Campus

REGISTRATION. Campus —> REGISTRATION. Degree

Note that REGISTRATION. Degree is determined both by the primary key SID and the non-key column campus.

For a relational table to be in third normal form (3NF) all columns must be dependent only upon the primary key. More formally, a relational table is in 3NF if it is already in 2NF and every non-key column is non transitively dependent upon its primary key. In other words, all non-key attributes are functionally dependent only upon the primary key. Or put another way, all attributes must be directly dependent on the primary key without implied dependencies through other attributes of the relation.

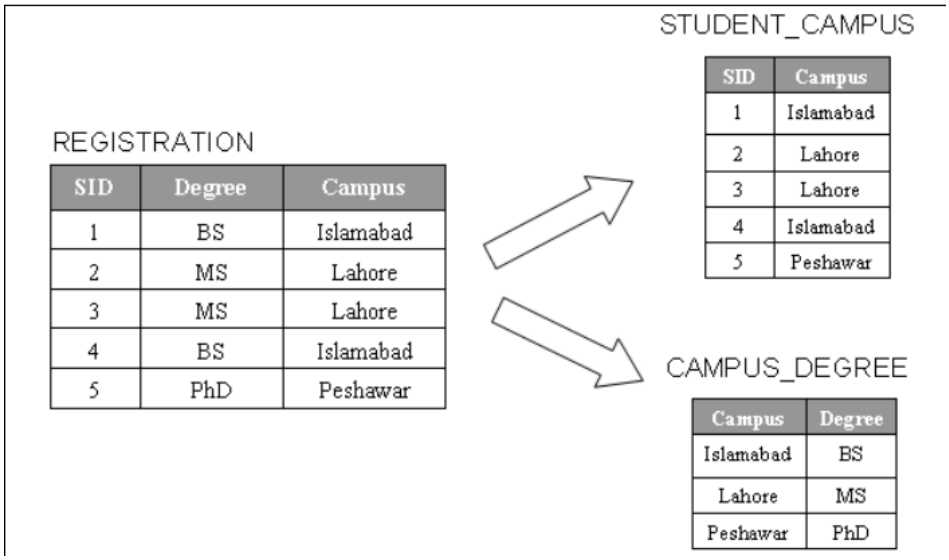
Table PERFORMANCE is already in 3NF. The non-key column marks is fully dependent and identified using the primary key SID and COURSE. However, table REGISTRATION is still only in 2NF as there is a *transitive dependency*. A transitive

dependency arises when a non-key column that is a determinant of the primary key is the determinate of other columns.

To transform REGISTRATION into 3NF, a new table is created called CAMPUS_DEGREE and the columns campus and degree are moved into it. Degree is deleted from the original table, campus is left behind to serve as a foreign key to table CAMPUS_DEGREE, and the original table is renamed to STUDENT_CAMPUS to reflect its semantic meaning.

Normalization: 3NF

To transform REGISTRATION into 3NF, we create a new table called CAMPUS_DEGREE and move the columns campus and degree into it. Degree is deleted from the original table, campus is left behind to serve as a foreign key to CAMPUS_DEGREE, and the original table is renamed to STUDENT_CAMPUS to reflect its semantic meaning.



Part of example student database system in 3NF

Shows that the table REGISTRATION is transformed into two tables i.e. Student Campus and Campus Degree. If we look at it in the context of memory, we observe that the storage space requirement has increased, for this particular example by about 7%.

Normalization: 3NF

Removal of anomalies and improvement in queries as follows:

1. **INSERT:** Able to first offer a degree program, and then students registering in it.
2. **UPDATE:** Migrating students between campuses by changing a single row.
3. **DELETE:** Deleting information about a course, without deleting facts about all columns in the record.

Observe that by virtue of bringing a relational table into 3NF, storage of redundant data is further eliminated, that not only results in saving of space, but also reduces manipulation anomalies. For example, as a consequence following improvements have taken place:

INSERT: If Peshawar campus decides to offer a PhD program, this can be reflected even
Though there is no student currently registered in that campus. Similarly, facts about new students can be added even though they may not have registered for a course.
UPDATE: Changing the campus of a student (by migration) or a degree program of a Campus requires modification of only a single row.

DELETE: Information about courses taken can be deleted without destroying Information about a student or a campus.

Conclusions

1. Normalization guidelines are cumulative.
2. Generally a good idea to only ensure 2NF.
3. 3NF is at the cost of simplicity and performance.
4. There is a 4NF with no multi-valued dependencies.
5. There is also a 5NF.

Normalization guidelines are cumulative. For a database to be in 2NF, it must first fulfill all the criteria of a 1NF database.

It is generally a good idea to ensure that the relational table is at least in 2NF. The goal is to avoid data redundancy so as to prevent slow corruption of data due to DML anomalies and make

the best possible use of storage.

Although 3NF removes even more data redundancy, but this is at the cost of simplicity and performance. Hence it is up to the designer to find a balance between normalization and speed/simplicity.

There is a fourth normal form (4NF) with one additional requirement i.e. meet all the requirements of 3NF and there must be no multi-valued dependencies.

There is also a 5NF, but that is more of academic interest.