SkillSoft logo

**|**[**Print**](javascript:window.print())**|**[**Contents**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t362/misc/transcript.html#contents)**|**[**Close**](javascript:parent.window.close())**|**

**Database Normalization**

**Learning Objective**

*After completing this topic, you should be able to*

* *recognize how to normalize the relations*

**1. Normalizing the database tables**

Normalization is the process of efficiently organizing data in a database by removing duplicate or redundant data from database relations. This is done to avoid storing the same data in more than one relation so that the amount of space a database consumes can be efficiently managed.

Normalization typically involves splitting existing relations into multiple smaller relations and establishing clearly defined relationships between them. These smaller relations are rejoined or linked each time a user sends a query to the database.

**Drill Down Home Page**

Normalizing database relations helps to improve data integrity, scalability, and storage efficiency.  
  
A series of guidelines have been developed to normalize database relations. These are referred to as *normal forms* or NF.

**Page 1 of 4: 1NF**

Short for first normal form, 1NF is the lowest form of normalization, and it sets the basic rules for an organized database.  
  
The main goal of 1NF is to divide the database data into logical database entities or tables. And then ensure it meets the following conditions:

* there should be no repeating groups or columns in the same table
* each row should be unique in a table, that is no duplicate rows, and
* each column should contain only one value and only one data type

**Page 2 of 4: 1NF**

Consider an example of an Employees table with a column to hold the different skills for each employee. With more than one value in a single Skills field, it would be difficult to search for all employees with a specific skill. This table is not efficient with storage, and will not be scalable.

To conform to 1NF rules, there should be only one value specified in a column, and it should only have one data type. Therefore the Skills column should be separated into another table to hold the information about skills.  
  
This table should then be linked back to the main table, Employees, using a primary key and a foreign key. Now the columns are unique in both tables because the Employee\_ID along with Skill\_ID makes each row unique.  
  
The tables are now in 1NF. Because each row can be uniquely identified, there are no repeating attributes or groups, and each attribute only contains one value.

**Page 3 of 4: 1NF**

Tables also cannot contain any repeating attributes or group of attributes, that is, for each occurrence of a row the attribute cannot have more than one occurrence or a many-to-many relationship.

For example, if you have an Employees table that contains information on the projects that are being worked on by each employee, there can be multiple projects specified for the employee if they work on more than one project. In this case, the Project\_Name, Start\_Date, End\_Date, and Budget attributes would be a repeating group.

**Page 4 of 4: 1NF**

To correct this, you must create a separate entity for the projects that are being worked on and also move the primary key of the Employees table to the new table as the foreign key to link the two tables. This will also ensure that no two rows contain the same values because an employee can only work on two separate projects at the same time, not the same project. So in the Projects table, the rows can be uniquely identified by using the columns Project\_Name and Employee\_ID. These columns are known as the candidate keys and comprise the superkey.  
  
The tables can now be considered to be in the 1NF, because there are no repeating attributes or groups. Each row is unique because no row in either table will contain the same values, and each attribute can hold only a single value.

**Page 1 of 3: 2NF**

Second normal form or 2NF further addresses the concept of removing duplicate data. To conform to 2NF, a relational database should

* meet all the requirements of the 1NF and
* each nonkey column must be fully dependent on the key column or columns

**Page 2 of 3: 2NF**

The main goal for 2NF is to ensure that each nonkey attribute is fully dependent on the key columns of the table, that is, no attribute can only be partially dependant on the key columns. You then need to move all nonkey columns that are not fully dependent on the key columns to a new table, with a copy of the key column they fully depend on. You then use a foreign key to link the two tables using the key column that is now in both the tables.  
  
To understand this, consider an example of the Animal Details table that has two candidate key columns – Animal\_ID and Date.

In this table, each nonkey column must fully depend on both the candidate key columns. However, the columns Animal\_Name and Animal\_Type only fully depend on the Animal\_ID column and do not require the Date column.

**Page 3 of 3: 2NF**

Therefore these nonkey columns need to be moved to a new table along with a copy of the key column, Animal\_ID. The Animal\_ID column will then become the primary key in the new table, called Animals, and a foreign key in the Animal Details table.

These tables are now in 2NF because they conform to 1NF and each nonkey column in each table fully depends on the key columns in that table.

**Page 1 of 2: 3NF**

A database table is in the third normal form or 3NF when it fulfills all the requirements of the 2NF. Additionally, any nonkey columns that are independent of the key column are removed. That is if a nonkey column is fully dependent on another nonkey column, rather than the key column, it should be moved to a new table, along with the nonkey column it fully depends on. The new table should have a foreign key to link to the original table.

**Page 2 of 2: 3NF**

Consider an example of the Sections table. In this table, the columns, Supervisor\_Phone, Supervisor\_FName, and Supervisor\_LName can be only determined by the Supervisor\_ID column. They cannot be determined by the Section\_Name or the Date, which are the key columns.

Therefore a new table needs to be created that includes the nonkey columns that are fully dependent on the nonkey column of Supervisor\_ID. The Supervisor\_ID column is also added to this new table as the primary key and in the Sections table as the foreign key.  
  
These tables are now in 3NF because they each conform to 2NF and each nonkey column is now dependent on the key columns in each table rather than dependent on nonkey columns.

**Page 1 of 3: BCNF**

Short for the Boyce-Codd Normal Form, BCNF requires that a table should be 3NF and that there are no overlapping candidate keys.  
  
Consider the Projects table that is used to store the projects for each employee. Each employee can work on multiple projects and each project can have multiple employees working on them.

**Page 2 of 3: BCNF**

In this table, the candidate keys are Employee\_ID, Project\_Name, First\_Name, and Last\_name. This is because you need to use each of these columns together to uniquely identify a row in the table.   
  
In this case, the Project\_Name is the overlapping candidate key. The table is not in BCNF because after you remove the Project\_Name from the candidate keys, they only describe the employee – Employee\_ID, First\_Name, and Last\_Name – that is they are determinants of each other and do not qualify as the candidate key on their own.  
  
The First\_Name and Last\_Name columns are considered to be dependent because if you change the first name of one of the employees in one row of the table, the other rows will no longer be correct. Also if you delete a row because the employee does not work with the company any more, you may lose the Employee\_ID record.

**Note**

*A determinant is a column that partially determines the value in another column.*

**Page 3 of 3: BCNF**

To put this table into BCNF, you need to create a new table that moves the dependent columns – First\_Name and Last\_Name, along with the Employee\_ID column – to be used as a link to relate the tables together.

The candidate keys for the Projects table are Project\_Name and Employee\_ID, and the only candidate key in the Employees table is Employee\_ID. You can now insert, update, and delete any of the employee records without affecting other rows in the Employees table or the Projects table.

**Page 1 of 2: 4NF**

The fourth normal form or 4NF deals with problems associated with normalizing tables that contain complex composite primary keys. A table is in 4NF when it meets all the requirements of BCNF, and doesn't have any multivalued dependencies. Neither should the dependent attributes be a subset of the attributes they depend on. And the dependent attributes combined with the attributes they depend on should not constitute the entire entity.  
  
In a multivalued dependency, two or more columns depend on a determinant, and each dependent column has a particular set of values. The values in the dependent columns are independent of each other.

**Page 2 of 2: 4NF**

Consider an example of the Employees table that stores information about employees who travel to different offices. The Skill and Office columns are both dependent on Employee\_ID, but regardless of their skills employees can work at any office, that is they don't need a specific skill to work at an office.  
  
Therefore, to uniquely identify each row, you need to use all the columns in the table, so there are no nonkey columns. And the table currently conforms to all normal forms up to and including BCNF. However there is redundant data.

To conform to 4NF, you need to move the dependent columns to a new table, along with the column it depends on so the tables can be related back.

**Page 1 of 2: 5NF**

A table is in fifth normal form or 5NF, which is also known as Project-Join Normal form if, and only if, it is in 4NF and every join dependency on the table is related to its candidate key. The requirements of the 5NF help you to design complex relationships that involve multiple tables.

When you decompose tables through normalization, you should be able to reconstruct the original table by doing joins between the resulting tables without losing data and without generating extra rows. This is a lossless join.  
  
When you can't decompose a table into smaller tables, which have different keys from the original without data losses, the table is in 5NF.

**Page 2 of 2: 5NF**

Using the same example from 4NF, you can add another table that combines the Offices with the Skills, so only applicable skills are located in a specific office.  
  
So an employee can work in specific offices, each employee has specific skills, and only certain skills are at certain offices.

Database normalization is a progressive activity. That means you can't have a 3NF normalized database until the database meets the 1NF and 2NF requirements.  
  
Most databases only need to be normalized to 3NF. At 3NF most tables are free of anomalies relating to inserting, updating, and deleting data from the tables. And typically when normalized to 3NF, the tables will also adhere to the higher normal forms of BCNF, 4NF, and 5NF.  
  
Usually databases normalized to 3NF will not conform to BCNF, 4NF, and 5NF when the database is highly complex and where nonnormalized tables may cause serious problems.

A well-normalized database provides users with several benefits:

* updates data quickly without introducing any redundancy or inconsistencies
* minimizes errors that are often caused by duplicated data
* avoids unnecessary coding needed to keep duplicated date in sync, and
* allows you to create short and narrow indexes that help in faster index searching

**Question**

Which statements are correct about normalizing database relations?

**Options:**

1. A normalized relation doesn't have repeating groups
2. Columns that are dependent on the primary key are removed from the relations
3. A normalized relation has several multivalued dependencies
4. A normalized relation has a set of columns that can uniquely identify a record

**Answer**

***Option 1:*** *This option is correct. A normalized relation should have no repeating groups. This ensures that no two columns of the relation store similar data.*

***Option 2:*** *This option is incorrect. A relation is normalized when columns that are not dependent on the primary key are removed.*

***Option 3:*** *This option is incorrect. According to 4NF rules, a relation should have no multivalued dependencies.*

***Option 4:*** *This option is correct. As per the 1NF requirement, a normalized relation should have a key column that together can be used to uniquely identify a record.*

**Correct answer(s):**

1. A normalized relation doesn't have repeating groups  
4. A normalized relation has a set of columns that can uniquely identify a record

**Summary**

In the relational database design, you need to normalize the relations or tables by eliminating inconsistencies and minimizing redundant data. A normalized relation ensures that only related data is stored in a relational database table. 1NF, 2NF and 3NF are frequently used normal forms for normalization. In highly complex databases, you may need to normalize relations to BCNF, 4NF, and 5NF.

**Table of Contents**

|  **[Top of page](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t362/misc/transcript.html" \l "pagetop)** |

|  [**Learning Objective**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t362/misc/transcript.html#objective) |

| [**1.Normalizing the database tables**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t362/misc/transcript.html#section_1) |

| [**Summary**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t362/misc/transcript.html#section_2) |

© 2011 SkillSoft Ireland Limited.