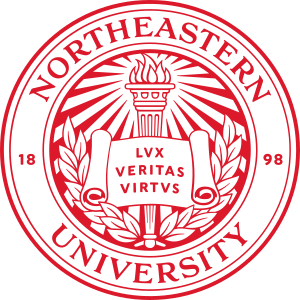
ITC 6000: Database Management Systems

0



Normalization

1NF, 2NF, 3NF

**ITC 6000 - Database Management Week 4**

College of Professional Studies Northeastern University - Vancouver

## REPRESENTATIVES

Smit Parmar Abhilash Dikshit Shamim Sherafati Nastaran Zamanian

# Abstract

The use of relational database (RDBMS) technology and different levels of normalization (1st, 2nd, 3rd normal data structures) is proliferating throughout the data processing industry. RDBMS systems are valued for their ability to maintain the integrity of data, reduce unnecessary data redundancy, and provide maximum flexibility in retrieval. Well-designed RDBMS applications typically result in normalized tables that remove repeating data groups, duplicated data, and establishes appropriate key to data associations within a table. Many of the powerful features in the SAS system utilize data sets that are not normalized, thus presenting several interfacing problems. This dichotomy in each products usage need not force a compromise in products strengths. In this report we will evaluate 3 different normalization forms used in our database bank\_ca and if it is not in the required normal form, we will perform the normalization steps.

# Introduction

One of the problems that has persisted throughout the development of relational technology is the lack of procedural data analysis and reporting tools for application developers and end-users to interface with a relational database. SQL is the industry standard data manipulation language (DML) for Selecting, Inserting, Updating, and Deleting rows and columns of data from a relational database. SQL is also a data definition language (DDL) for creating and deleting database objects (table1s, views, indexes, synonyms, etc.).

Normalization is a formal process of organizing data into table~ (relations) of logically related information that satisfy conditions defined for the various normal forms (Kemm 1989). Provide a base of all data elements relevant to the business requirements.  
Provide a processing environment that makes these data elements easily available to all appropriate users, both current and future. Ensure data integrity in all the data elements. Also, to provide a stable, reproducible, highly flexible, and standardized data architecture to meet the clearly defined business needs. Optimize the performance of the database.

Graphical user interface, text

Description automatically generated

Primary Key

We have a total of 6 tables in our database with 6 unique primary keys individually.

Table

Description automatically generated

Foreign Key

Table

Description automatically generated

Account

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the column **acc\_balance** is fully dependent on the primary key of that table, which is **account\_acc\_no**. Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. In the table, columns **acc\_balance,**and **acc\_type** is only dependent on **acc\_no**. Hence, it is in 3rd normal form.

Loan

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and each and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the columns are fully dependent on the primary key of that table, which is **loan\_customer\_c\_id\_branch\_b\_id**. Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. Hence, it is in 3rd normal form.

Bank

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the column **ba\_name, ba\_address** and **ba\_type** is fully dependent on the primary key of that table, which is **bank\_ba\_code**. Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. Hence, it is in 3rd normal form.

Branch

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and each and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the columns **b\_name, b\_address** and **ba\_code** is fully dependent on the primary key of that table, which is **branch\_b\_id.** Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. In the first table, columns **b\_name, b\_address** and **ba\_code** is only dependent on **b\_id**. Hence, it is in 3rd normal form.

Customer

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the columns are fully dependent on the primary key of that table, which is **customer\_cid**. Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. Hence, it is in 3rd normal form.

Employee

Table

Description automatically generated

1NF- Check if no Multi value attributes and one primary key?

In the above table, we can clearly see that the columns have one values and one primary key. Thus, it does not violate the 1st NF. By this, we have achieved atomicity and each and every column have unique values. Hence, it is in 1st normal form.

2NF- Check for no partial functional dependency?

As you can see, we have no partial functional dependency. In the table, the column is fully dependent on the primary key of that table, which is **employee\_eid**. Hence, it is in 2nd normal form.

3NF- Check for no Transitive functional dependency for non-prime attributes?

As you can see from the above tables all the non-key attributes are fully functional dependent only on the primary key. Hence, it is in 3rd normal form.

# Summary

In this paper a discussion has been presented on the use of different degrees of normalization using a relational database. The use of a relational database can improve the quality and accessibility of information requirements in many different applications. By applying normalization techniques, application developers and end-users alike can take advantage of the strengths of storing data in the various normal forms (1NF, 2NF, 3NF). These strengths include, but are not limited to, the elimination of redundant.

# Bibliography

1. Reddy, J. (2019, July 10). *Normalization in SQL Server*. SQLServerCentral. [https://www.c-sharpcorner.com/UploadFile/44fb93/normalization-in-sql-server/Links to an external site.](https://www.c-sharpcorner.com/UploadFile/44fb93/normalization-in-sql-server/)

2. Shankar, G. (2020, February 18). *Database Normalization in SQL with Examples –*. SQLServerCentral. [https://www.sqlservercentral.com/articles/database-normalization-in-sql-with-examplesLinks to an external site.](https://www.sqlservercentral.com/articles/database-normalization-in-sql-with-examples)

3. Chauhan, S. (2022, August 31). *Understanding Database Normalization with Example*. <https://www.dotnettricks.com/learn/sqlserver/database-normalization-basics>