Mark Biggar

Discussion of normalisation, with respect to the tables presented in the assignment.

Applied Databases

Normalisation

Contents

[Table of tables 2](#_Toc9124682)

[Table of Figures 2](#_Toc9124683)

[Introduction 3](#_Toc9124684)

[Table presented 3](#_Toc9124685)

[Database Anomalies 3](#_Toc9124686)

[Insertion anomaly: 3](#_Toc9124687)

[Deletion anomaly: 4](#_Toc9124688)

[Update anomaly: 4](#_Toc9124689)

[Normalization 4](#_Toc9124690)

[Application of normalisation to the table 4](#_Toc9124691)

[1st Normal Form 4](#_Toc9124692)

[2nd Normal Form 5](#_Toc9124693)

[3rd Normal Form and Boyce-Codd Normal Form 5](#_Toc9124694)

[4th Normal Form 5](#_Toc9124695)

[5th Normal Form 6](#_Toc9124696)

[6th Normal Form 6](#_Toc9124697)

[Domain Key Normal Form 7](#_Toc9124698)

[New schema of database 7](#_Toc9124699)

[Possible future state 7](#_Toc9124700)

[Conclusion 8](#_Toc9124701)

[Bibliography 9](#_Toc9124702)

[Appendix – SQL coding to create and assign data 10](#_Toc9124703)

# Table of tables

[Table 1: SQL database presented in Q4.2 3](#_Toc9124709)

[Table 2: Table in 1st Normal Form 4](#_Toc9124710)

[Table 3: Student table in 2nd Normal Form 5](#_Toc9124711)

[Table 4: Module table in 2nd Normal Form 5](#_Toc9124712)

[Table 5: Table in 4th Normal Form 6](#_Toc9124713)

[Table 6: Table in 4th Normal Form 6](#_Toc9124714)

[Table 7: Table linking student ID with module ID 6](#_Toc9124715)

[Table 8: Students table description 10](#_Toc9124716)

[Table 9: Data in student table 10](#_Toc9124717)

[Table 10: Description of module table 11](#_Toc9124718)

[Table 11: Data in module table 11](#_Toc9124719)

[Table 12: Description of courses table 12](#_Toc9124720)

[Table 13: Data in courses table 12](#_Toc9124721)

[Table 14: Replication of initial table 13](#_Toc9124722)

# Table of Figures

[Figure 1: Schema for proposed database 7](#_Toc9124704)

[Figure 2: Possible future high level schema (Fotache, 2006) 7](file:///C:\Users\markb\OneDrive\Desktop\Data%20Analytics\Databases\Project\G00376334\Normalisation.docx#_Toc9124705)

# Introduction

It is intended to review the table presented with respect to the rules of normalization. In the approach, each of the rules of normalization will be applied to the table, where appropriate. This will be used to both demonstrate the application of the rules, and to create new tables that are more suitable for the data being stored. Further, it is hoped to show how these tables can be used as the basis of more expansive database, which will capture more data that may be required.

# Table presented

The table presented in the question is replicated below. The table covers 5 columns of data, namely:

* StudentID – the identity of the student in the database.
* StudentName – the first name of the student enrolling in courses.
* DOB – the date of birth of the students.
* ModuleID – the identifier for the module that the college offers.
* ModuleName – the name of the module that the college offers.

Primary keys are StudentID and ModuleID.

Each row represents the course that on which a student has enrolled.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentID | StudentName | DOB | ModuleID | ModuleName |
| 1 | Sean | 03/01/2000 | 100 | Applied Databases |
| 2 | Bill | 23/04/1990 | 100 | Applied Databases |
| 3 | Tom | 10/12/1973 | 101 | Java Programming |
| 3 | Tom | 10/12/1973 | 104 | Mobile Apps |
| 4 | Mary | 12/04/1991 | 101 | Java Programming |
| 4 | Mary | 12/04/1991 | 102 | Computer Architecture |
| 5 | Joe | 29/06/1982 | 100 | Applied Databases |
| 5 | Joe | 29/06/1982 | 104 | Mobile Apps |

Table 1: SQL database presented in Q4.2

# Database Anomalies

In general, relational databases often contain a number of issues. These include unnecessary complexity, the same data being stored in multiple places, difficulty in updating information correctly, data being inserted in the incorrect places, or in the wrong form, as well as removing incorrect or redundant data (Watt & Eng, 2014). These issues are referred to as deletion, insertion, or updating anomalies (Watt & Eng, 2014).

Specifically, in the table presented, a number of issues arise. These include:

## Insertion anomaly:

Insertion anomalies occur when incorrect information, which is inconsistent with the previously entered data, is entered into a table (Watt & Eng, 2014). In the table above, if students wished to add another module to their course load, the user would need to add another row of data to the table. The information would need to be copied into each table, which could give rise to both transcription errors, as well as extra work to ensure that all the rows are completed (e.g. ID, and DOB). Additionally, the more students wishing to take new courses, this issue would increase exponentially.

## Deletion anomaly:

Deletion anomalies refer deleting information of one purpose, but losing additional information that should be retained (Watt & Eng, 2014). For example, if 100 – Applied Databases was to be removed, all the information with regards to Bill would be lost. Similarly, if the Mary was to be deleted, all information with regards to 102 – Computer Architecture would no longer be available.

## Update anomaly:

Update anomalies occur when information relating to one piece of information is updated in an inconsistent manner (Watt & Eng, 2014). In the above table, if 100 – Applied Databases was to change name or code, the user would need to change all individual occurrences of the name.

# Normalization

Normalization can be defined as the process of ensuring that the information is stored in a manner that can be understood, and that reduces the amount of superfluous or redundant information that is store (Karwin, 2010). It is primarily concerned with storing data correctly, and thereby reducing the complexity of the database (Karwin, 2010), and increasing consistency (Agiledata.org, 2019). The normalization of a database, in turn, also helps reduce the cost and time associated with maintenance, coding, and interpretation (Karwin, 2010).

# Application of normalisation to the table

A database that has been normalized is referred to as a normal form database. There are generally 6 main rules of database normalization (1st Normal Form – 6th Normal Form), and two less used rules (Boyce-Codd Normal Form and Domain Key Normal Form). The normal forms will be applied to the given table, as far as is practical and makes sense.

## 1st Normal Form

In the 1st Normal Form, there should only be one piece of data where each row and column meets (Watt & Eng, 2014). For example, if a row contains column contains 2 entries for a phone number in a single row, it would violate the 1st Normal Form. The remedy for this would be to split the data into two rows, with the difference in rows being the phone numbers. Alternatively, additional columns could be introduced, with the entries being split between the new columns, instead of adding new rows.

The table currently satisfies the terms of the 1st Normal Form, as each row and column intersection contains only one piece of data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentID | StudentName | DOB | ModuleID | ModuleName |
| 1 | Sean | 03/01/2000 | 100 | Applied Databases |
| 2 | Bill | 23/04/1990 | 100 | Applied Databases |
| 3 | Tom | 10/12/1973 | 101 | Java Programming |
| 3 | Tom | 10/12/1973 | 104 | Mobile Apps |
| 4 | Mary | 12/04/1991 | 101 | Java Programming |
| 4 | Mary | 12/04/1991 | 102 | Computer Architecture |
| 5 | Joe | 29/06/1982 | 100 | Applied Databases |
| 5 | Joe | 29/06/1982 | 104 | Mobile Apps |

Table 2: Table in 1st Normal Form

## 2nd Normal Form

A table must be in 1st Normal Form in order for the 2nd Normal Form to be implemented. 2nd Normal Form deals with information in the row is not relevant to the primary key, but is a subset of the key (Kent, 1983).

In the tables above both date of birth of the student, nor the module name are relevant to the entire field. This is due to the fact that they will stay constant to both student and the module ID respectively. In order for the table to meet the 2nd Normal Form requirements, it would be necessary to split the table into 2 separate tables. These would be a student table, made of the student’s ID, student’s name, and their date of birth; a module table composed of the module ID and the module name. This would still maintain the primary keys, but split into two tables.

These would be set up as follows:

|  |  |  |
| --- | --- | --- |
| StudentID | StudentName | DOB |
| 1 | Sean | 03/01/2000 |
| 2 | Bill | 23/04/1990 |
| 3 | Tom | 10/12/1973 |
| 4 | Mary | 12/04/1991 |
| 5 | Joe | 29/06/1982 |

Table 3: Student table in 2nd Normal Form

|  |  |
| --- | --- |
| ModuleID | ModuleName |
| 100 | Applied Databases |
| 101 | Java Programming |
| 102 | Computer Architecture |
| 104 | Mobile Apps |

Table 4: Module table in 2nd Normal Form

## 3rd Normal Form and Boyce-Codd Normal Form

The 3rd Normal Form deals with data in the table being not directly related to the primary key (Kent, 1983). In this case, the student ID, student name, and date of birth do not contain any non-directly related data, while the module table only has two columns, so it would be impossible to further split the table.

The Boyce-Codd Normal, often abbreviated as BCNF, is a special case of the 3rd Normal Form (Watt & Eng, 2014). This form relates to the case where all the columns in a table could be considered a primary key for the table. In the student table neither the student name nor date of birth would be sufficiently unique to create a primary key.

In the cases of both tables, they satisfy both the 3rd Normal Form, and the Boyce-Codd Normal Form.

## 4th Normal Form

The 4th Normal Form suggests that the tables should contain only two dependent variables per table (Kent, 1983). Further, it suggests that when there should only be one multivalued dependency per table (Geeks for Geeks, 2019). This differs from the previous 3 rules, which dealt with functional dependencies.

While there are no functional dependencies left in the table, it is possible to split the tables further. The module table already meets the requirements of the 4th Normal Form. Nevertheless, the student table could be split into 2 separate tables comprising of the student ID, and student name; and a second table with the student ID, and the date of birth. The tables are presented below, however, there is little value gained in the creation of these tables.

|  |  |
| --- | --- |
| StudentID | StudentName |
| 1 | Sean |
| 2 | Bill |
| 3 | Tom |
| 4 | Mary |
| 5 | Joe |

Table 5: Table in 4th Normal Form

|  |  |
| --- | --- |
| StudentID | DOB |
| 1 | 03/01/2000 |
| 2 | 23/04/1990 |
| 3 | 10/12/1973 |
| 4 | 12/04/1991 |
| 5 | 29/06/1982 |

Table 6: Table in 4th Normal Form

## 5th Normal Form

In order for a table to meet the required form of the 5th Normal Form, every join from the original table(s) must have been recreated in a zero sum manner, so that that no extra information has been created when the data is joined again (Geeks for Geeks, 2019). The tables created above do lack one table, in order to meet the requirements of the 5th Normal Form. This table would consist of a student ID as well as the module ID of the courses being undertaken. This table would then be used as a connector table for the two primary keys, and would be as follows:

|  |  |
| --- | --- |
| StudentID | ModuleID |
| 1 | 100 |
| 2 | 100 |
| 3 | 101 |
| 3 | 104 |
| 4 | 101 |
| 4 | 102 |
| 5 | 100 |
| 5 | 104 |

Table 7: Table linking student ID with module ID

## 6th Normal Form

The 5th Normal Form is normally considered the final form for database table (Darko Golec, 2017). However, for the sake of completeness, it is necessary to define both the 6th Normal Form, and the Dormant Key Normal Form, and determine whether they can be applied to the tables presented.

The 6th Normal Form is used to “*handle the situation where temporal data vary independently to avoid unnecessary duplication*” (Harrington, 2009). This normally results in the tables with only two columns of data, namely a column containing a primary key, and a column containing a related piece of information. All suggested tables meet the requirement of the 6th Normal Form.

## Domain Key Normal Form

Fagin (1981) states that “*a schema is in DK/NF if and only if it has no insertion or deletion anomalies*”. This would suggest that when a table has had all its anomalies removed, such as in the 1st Normal Form, it meets the minimum constraints of the Domain Key Normal Form. Additionally, the Domain Key Normal Form also suggests that there needs to be only 2 columns per table, the primary key, and a dependent variable (Fagin, 1981). Once again the suggested tables would meet the requirements of the Domain Key Normal Form.

# New schema of database

Based on the normalization rules discussed above, it is possible to create a new schema for the proposed database. This new schema would be made up of the 3 tables; the student table (Table 3), the courses table (Table 7), and the module table (Table 4). The schema would be as follows:

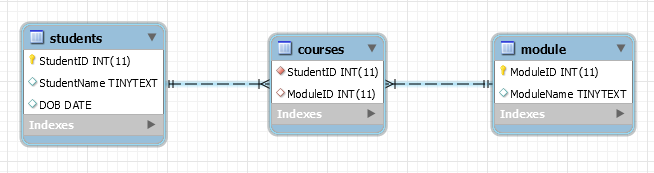


Figure 1: Schema for proposed database

A check to enable the replication of the initial table shows that it is possible to recreate the table (see Appendix A).

# Possible future state

The new schema allows for the creation of additional tables which will support further expansion of the data collected. An example of a possible example is provided by Fotache (2006), who provides the diagram at the right.

From this diagram, it is clear that a well organised schema can allow an orderly increase in the way the data is collected and stored. As can be seen, if the college wishes to expand the number of students, courses, lecturers, or facilities, these can be incorporated into the schema with only minor alterations. Additional information, such as finanical income (from students), or financial expenditure (on staff) could easily be incorporated, without the need to create new databases.

Figure 2: Possible future high level schema (Fotache, 2006)

# Conclusion

It is clear that the table that was presented in the initial question is not a well thought out or suitable table, that fails in its purpose. The table incorporates serious defecenies around the way the data is laid out. These deficenices give rise to possible issues with insertion, updating, and deletion anomalies. In addition, the table encompasses two primary keys that are not related. It is clear, as demonstrated above, that with the application of the normalization rules, it is possible to create a simpler, yet more adaptable database schema. This schema can be adapted, and upgraded, without encountering the anomaly issues of the current database/table.

In conclusion, the current table is not a good table.

# Bibliography

* Agiledata.org, 2019. *Introduction to Data Normalization: A Database "Best" Practice.* [Online]   
  Available at: http://agiledata.org/essays/dataNormalization.html  
  [Accessed 10 May 2019].
* Codd, E. F., 1970. A relational model of data for large shared data banks. *Communications of the ACM,* 13(6), pp. 377 - 387.
* Darko Golec, V. M. T. K., 2017. Relational model of temporal data base on 6th Normal Form. *Tehnički vjesnik,* 24(5), pp. 1479 - 1489.
* Eessaar, E., 2016. The database normailzation theory and they of normalized systems: Finding a common ground. *Baltic Journal of Modern Computing,* 4(1), pp. 5 - 33.
* Fagin, R., 1981. A normal form for relational database that is based on domains and keys. *ACM Transactions on Database Systems.*
* Fotache, M., 2006. *Why normalisation failed to become the ultimate guide for database designers?.* [Online]   
  Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=905060  
  [Accessed 10 May 2019].
* Geeks for Geeks, 2019. *DBMS | Introduction of 4th and 5th Normal form.* [Online]   
  Available at: https://www.geeksforgeeks.org/dbms-introduction-of-4th-and-5th-normal-form/  
  [Accessed 10 May 2019].
* Harrington, J. L., 2009. Normalization. In: *Relational Database Design.* s.l.:ScienceDirect, p. 440.
* Karwin, B., 2010. *SQL Anitmatters - Avoiding the pitfalls of database programming.* s.l.:The Pragmatic Programmers.
* Kent, W., 1983. A simple guide to five normal forms in relational database theory. *Communications of the ACM,* 26(2), pp. 120 - 125.
* Papenbrock, T. & Naumann, F., 2017. *Data-driven schema normalization.* Venice, Proceedings of the 20th International Conference on Extending Database Technology.
* Watt, A. & Eng, N., 2014. *Database design.* 2nd ed. Victoria, B.C.: s.n.

# Appendix – SQL coding to create and assign data

#### Using database called “College”

#### To create and populate the “Students” table (Table 3):

* **Table creation:**

CREATE TABLE Students (

StudentID INT NOT NULL AUTO\_INCREMENT,

StudentName TINYTEXT,

DOB DATE,

PRIMARY KEY (StudentID)

);

* **Description of table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| StudentID | int(11) | NO | PRI | NULL | auto\_increment |
| StudentName | tinytext | YES |  | NULL |  |
| DOB | date | YES |  | NULL |  |

Table 8: Students table description

* **To populate table:**

INSERT INTO Students

(StudentName, DOB)

VALUES

('Sean', '2000-01-03'),

('Bill', '1990-04-23'),

('Tom', '1973-12-10'),

('Mary', '1991-04-12'),

('Joe', '1982-06-29');

* **To show entries in table:**

Select \*

From students

* **Output**

|  |  |  |
| --- | --- | --- |
| StudentID | StudentName | DOB |
| 1 | Sean | 03/01/2000 |
| 2 | Bill | 23/04/1990 |
| 3 | Tom | 10/12/1973 |
| 4 | Mary | 12/04/1991 |
| 5 | Joe | 29/06/1982 |

Table 9: Data in student table

#### To create and populate the “Module” table (Table 4):

* **Table creation:**

CREATE TABLE Module (

ModuleID INT AUTO\_INCREMENT,

ModuleName TINYTEXT,

PRIMARY KEY (ModuleID)

);

* **Description of table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| ModuleID | int(11) | NO | PRI | NULL | auto\_increment |
| ModuleName | tinytext | YES |  | NULL |  |

Table 10: Description of module table

* **To populate table:**

INSERT INTO Module

(ModuleID, ModuleName)

VALUES

('100', 'Applied Databases'),

('101', 'Java Programming'),

('102', 'Computer Architecture'),

('104', 'Mobile Apps');

* **To show entries in table:**

Select \*

From Module

* **Output**

|  |  |
| --- | --- |
| ModuleID | ModuleName |
| 100 | Applied Databases |
| 101 | Java Programming |
| 102 | Computer Architecture |
| 104 | Mobile Apps |

Table 11: Data in module table

#### To create and populate the “Courses” table (Table 7):

* **Table creation:**

CREATE TABLE COURSES (

StudentID INT NOT NULL,

ModuleID INT,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)

);

* **Description of table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| StudentID | int(11) | NO | MUL | NULL |  |
| ModuleID | int(11) | YES | MUL | NULL |  |

Table 12: Description of courses table

* **To populate table:**

INSERT INTO courses

(StudentID, ModuleID)

VALUES

(1, 100),

(2, 100),

(3, 101),

(3, 104),

(4, 101),

(4, 102),

(5, 100),

(5, 104);

* **To show entries in table:**

Select \*

From Courses

* **Output**

|  |  |
| --- | --- |
| StudentID | ModuleID |
| 1 | 100 |
| 2 | 100 |
| 3 | 101 |
| 3 | 104 |
| 4 | 101 |
| 4 | 102 |
| 5 | 100 |
| 5 | 104 |

Table 13: Data in courses table

#### Check to replicate original table (Table 1):

* **Table replication**

Select s.StudentID,

s.StudentName,

s.DOB,

m.ModuleID,

m.ModuleName

From Students s

Join Courses c

Ons.StudentID = c.StudentID

Join Module m

Onc.ModuleID = m.ModuleID

Order by s.StudentID ASC, m.ModuleID ASC;

* **Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentID | StudentName | DOB | ModuleID | ModuleName |
| 1 | Sean | 03/01/2000 | 100 | Applied Databases |
| 2 | Bill | 23/04/1990 | 100 | Applied Databases |
| 3 | Tom | 10/12/1973 | 101 | Java Programming |
| 3 | Tom | 10/12/1973 | 104 | Mobile Apps |
| 4 | Mary | 12/04/1991 | 101 | Java Programming |
| 4 | Mary | 12/04/1991 | 102 | Computer Architecture |
| 5 | Joe | 29/06/1982 | 100 | Applied Databases |
| 5 | Joe | 29/06/1982 | 104 | Mobile Apps |

Table 14: Replication of initial table