UPPSALA
UNIVERSITET

Data Quality, Normalization and Redesign

To: Professor La Carpa and IT Personnel of the Music Department

Authors: Elza Georges, Amalia Hoff, Lina Sjögren (Group 27)

Assignment 2, Database Design I

Table of contents

1. Introduction

Since the beginning of time man has been collecting and organizing objects in a methodical and systematic way. The purpose of this methodical and systematic arranging is the same today as it was when our predecessors walked the earth — it simplifies the retrieval of stored objects and you will spend less time searching for a specific object. By utilizing a systematic approach more objects can be stored than if we store them in a random manner (which is highly impractical and often costs more time and space).

The modern day database has integrated our predecessors tendencies into the world of technology and systems. This makes it possible to store, organize and retrieve data efficiently. The Music Department of Uppsala University has until today used an ineffective way to store information about its musical scores. By implementing the data from the previously used Excel-file to a relational database in SQL we corrected several issues with the existing database and redesigned it. Therefore, the purpose of this report is to describe the problems that we have encountered and what solutions we have implemented.

2. Overview of the Previous Table

An Excel-file has previously been used to store information within the Music Department of Uppsala University. Since the amount of information stored in the Excel-file has increased over time and the semantics needed some improvement we added new attributes to the old structure (See Table 1a for old table and Table 1b for added attributes).

2.1 Problems with the Previous Table

Table 1a contains different attributes such as Title, Name, Surname, Year of birth, Year of death, Publisher, Instruments, New Location and Notes. As mentioned in the introduction, the core idea of a database is to save time and space and this was also the goal with the excel-table. When reviewing the five attributes Surname - Publisher one can notice repetition of data. This takes up excess storage space, time and also leaves room for human mistakes such as misspelling. By compressing the data through normalization in the SQL database this problem can be solved (see 4. Functional Dependencies). The attribute Instruments is not optimally designed, making it impossible for users to search for a specific instrumentation in a score. This was solved by decoding in the database (see 3. The ER diagram of our finished database). The attribute New Location had information of multiple locations stored in single fields or cells, which causes a problem when searching for scores in a specific location and was later corrected in the database (see 4. Functional Dependencies).

Title	Surname	Name	Born	Died	Publisher	Instruments	New Location	Notes

Table 1a Shows the attributes used in the old table in the Excel-file

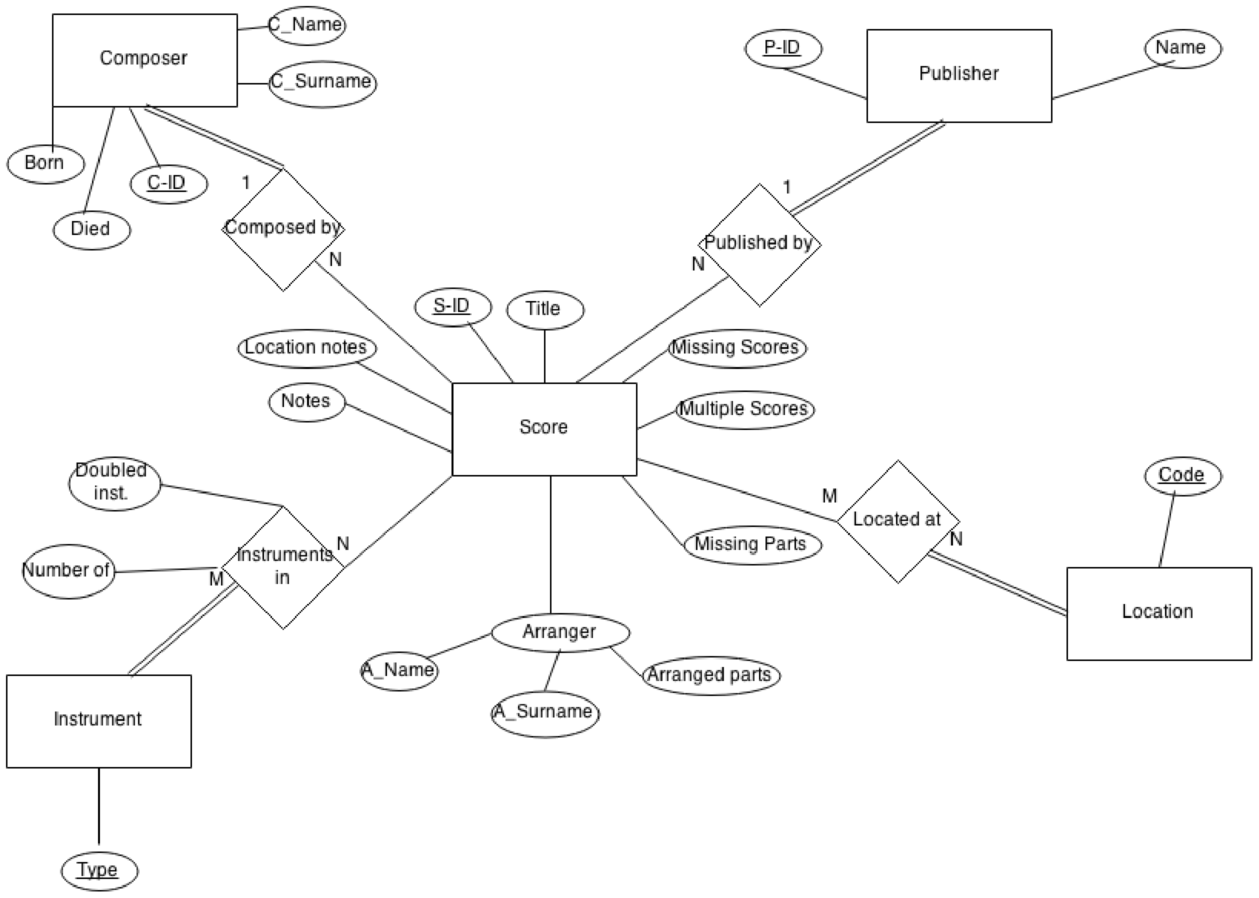
Notes was an attribute containing different types of information which could be categorized, making the information easier to retrieve. The following attributes where therefore added: Location notes, Arranger name, Arranger surname, Arranged parts, Missing parts, Missing scores, Multiple scores. The attribute ScoreID was also added, making it easier to identify the unique Scores (see Table 1b).

ScoreID	Location notes	Arranger name	Arranger surname	Arranged parts	Missing parts

Missing scores	Multiple scores

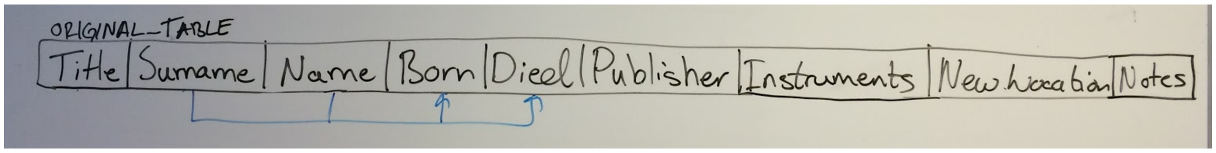
Table 1b (two rows) Shows the added attributes to the Excel-file

3. The ER-Diagram of Finished Database



4. Functional Dependencies

4.1 The Original Table



Picture 1a Shows the functional dependencies identified in the original table.

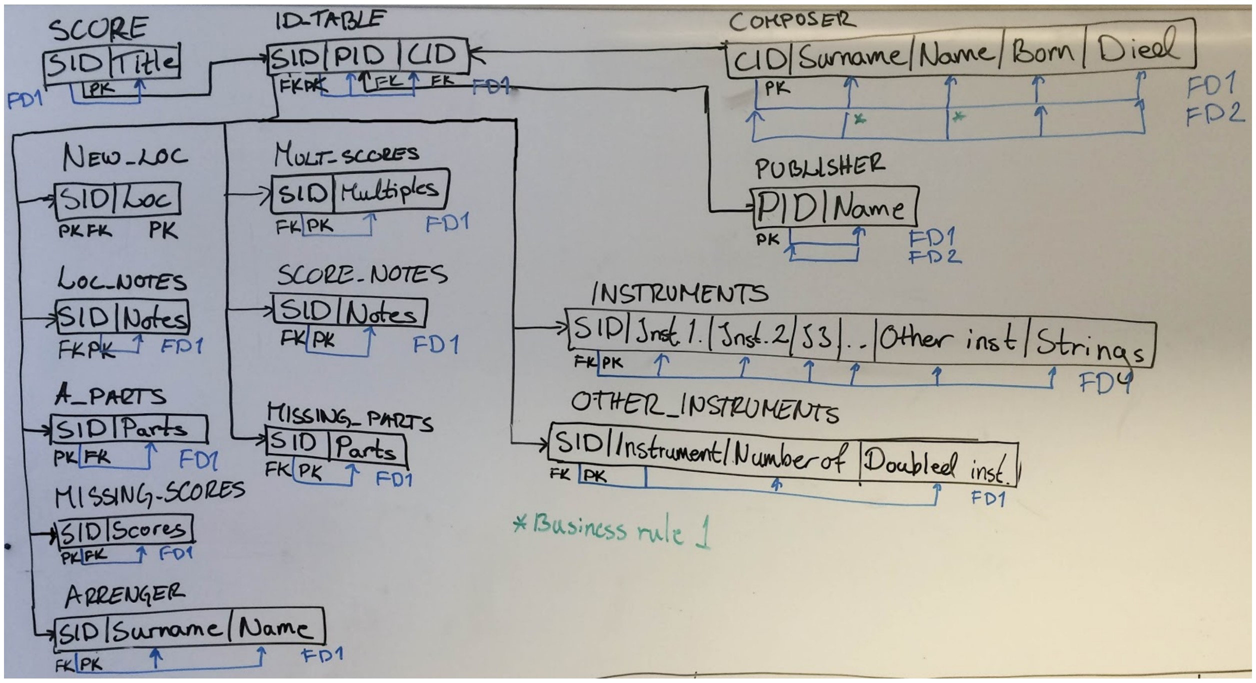
When analyzing the original table seen above, it soon became clear that it was impossible to uniquely determine some of the attributes. For example, there were scores published by different publishers and some scores have multiple locations. As mentioned before, this is not a very efficient way to store data. Further, complications can arise when trying to retrieve the right information about a specific score.

After further analysis one Functional Dependency (FD) was established. Information about composers is relatively easy to retrieve if you have the composer's surname and first name, while other data requires far more time and work.

Examples of the identified problems with the old database are stated below:

- There will be complications when searching for a score using the title, since composers might use the same title, or even reuse a title. Scores might also exist in different versions if they are released by different publishers.
- As some of the scores exist in different locations but the information were stored in the same cell, it would be difficult to sort the scores according to location, since some of the data would be lost.
- The column labelled noted contained different types of information, making the table hard to read and retrieve.

4.2 The Created Database



Picture 1b Shows the functional dependencies identified in the updated tables.

When categorizing notes into different necessary attributes, fields containing no information, so called NULL values, was created. Since the NULL values use space in the database, the tables was split into several smaller tables to remove these values and therefore reduce space needed for storage. The Score ID (seen as SID in the picture) was used to link information in the fields to the corresponding score, which leads to a number of links between the different tables. Within the different tables it is now possible to uniquely identify each tuple, using the different functional dependencies, as seen in Picture 1b.

To reduce the possibility of human errors, a composer ID (CID) and publisher ID (PID) was added to each composer respectively publisher to facilitate the process of adding new scores to the database.

Note: in the table named COMPOSER there is a so called business rule: if you want to add a composer with a name and surname identical to another composer already existing in the database, you will have to add some kind of specification (eg. Johann Strauss the younger and Johann Strauss the older) to distinguish them apart.

5. Keys and Tables

ID_TABLE 3NF

SID	Composer ID	Publisher ID
FK PK	FK	FK

SCORE 3NF

SID	Title
PK	

COMPOSER 3NF

Composer ID	Surname	Name	Born	Died
PK				

PUBLISHER 3NF

Pub ID	Pub name
PK	

INSTRUMENTS 3NF

SID	Flute	Oboe	Clarinette	Bassoon	Horn	Trumpet	Trombone	Tuba	Timpani	Percussionist	Harp	Other Instruments	I Violin	II Violin	Viola	Celli	Bass
FK PK																	

OTHER_INSTRUMENTS 3NF

SID	Instrument	Number_of	Doubled_Inst
FK PK		PK	

NEW_LOCATION 3NF

SID	Location

FK PK PK

LOCATION_NOTES 3NF

SID	Notes

FK PK

Arranger 3NF

SID	A_Surname	A_Name

FK PK

ARRANGED_PARTS 3NF

SID	Parts

FK PK

MISSING_PARTS 3NF

SID	Parts

FK PK

MISSING_SCORES 3NF

SID	Score

FK PK

MULTIPLE_SCORES 3NF

SID	Multiple

FK PK

SCORE_NOTES 3NF

SID	Notes

FK PK

6. Further Development

To complete the construction of the database according to plan, a number of steps must be taken. A new entity type and table should be created called OTHER_INSTRUMENT, to store the name of each instrument unequivocally and to further limit the possibility of human error. This table should be linked to each score via our relation OTHER_INSTRUMENTS, but with an added foreign key constraint on the existing attribute INSTRUMENT.

Furthermore, in the current INSTRUMENT table, only certain fields with data are populated. To populate the rest, two separate functions are needed. One function is needed to convert the data containing the shorthand for the orchestration into integers, and then insert them into the appropriate column in the INSTRUMENTS relation. This can be done using substring and pattern-recognition functions, as seen in the script (where a shorthand for the orchestration is converted and placed into the corresponding instrument columns in the mentioned table). The second function has a more complicated objective: To transform the instrumentations for small ensembles, that are not coded, into usable data. This requires considerably more thought and work, but would be based on the same principles as the first function. A similar substring extraction needs to be done to separate the multiple locations in the original New Location column.

In the section Keys and Tables above, which describes the schema for our database, there are several tables that are not yet created. Additionally, many of the current tables do not as yet contain any data. The completion and population of tables according to our schema is a repetitive task that can be easily extrapolated from the existing SQL code.
Finally, once all the data has been transferred to the new tables, the original ARCHIVE table should be deleted.