LUTON'S SECONDARY SCHOOL DATABASE MANAGEMENT SYSTEM

Table of Contents

Introduction and background

The goal of this report is to design a database solution for a secondary school located in Luton. The report utilizes a Use-Case diagram and an Entity Relationship Diagram (ERM or ERD) to develop a database that provides a detailed understanding of the database and its relationships between various entities. The database is designed to accommodate three main tables for pupils, subjects, and staff, and three different queries are demonstrated to ensure the database's functionality. These queries are generated by combining any two or three tables, such as queries between pupils and subjects, subjects and staff, and all three tables.

In addition to outlining the general structure and queries, the report also details the tools and techniques used to develop the database and queries. The APACHE server is utilized to create a local server, allowing for the creation of the database under PHP Admin (Kostakis, 2019). The ER diagram is created using draw.io. This is used as a free online tool to develop the Use-Case diagram. PHP Admin is helpful in developing the query-specific language SQL or MYSQL (Schwab *et al.*, 2019). Using MYSQL offers the advantage of being a browser-free and system-free language, allowing for the outcomes and database to be checked on any machine or system.

In summary, this engineering report provides a comprehensive overview of the process used to design and implement a database solution for a local secondary school in Luton. By utilizing Use-Case and Entity Relationship diagrams, the report demonstrates the database's relationships between different entities and outlines the tools and techniques used to create and query the database (Fadli and Sunardi, 2018).

The primary purpose of this report is to provide a comprehensive understanding of data management systems, data system design, data governance practice, and data modeling. It explains the concept of Relational Database Management Systems (RDBMS) using structured SQL queries (Kolonko, 2018). The report also includes a use case diagram and entity relationship model for a local secondary school in Luton. The project background identifies three tables, namely pupils, subjects, and staff, with their respective attributes. The report ensures that all necessary queries can be generated by making the required changes, including adding additional attributes to establish relationships between the entities. For instance, the pupils table includes the name of students studying in secondary school, the staff table includes information about teachers, and the subjects table provides information about the subjects taught in the school.

Tools and techniques to design the system

This study uses several key tools, including the APACHE server to create a local server with the database for the secondary school, the XAMPP Control Panel Version 3.3.0 to control the APACHE server, and the open-source and free administration tool PHP MyAdmin to develop the database and execute queries (Kadir *et al.*, 2021). Additionally, the study utilizes Draw.io, an open-source and free software application to develop the Use-Case diagram of the software to create the Entity Relationship Diagram (Jansen and Tyler, 2021). Finally, the study employs SQL/MySQL, an open-source RDBMS, to write and execute queries.

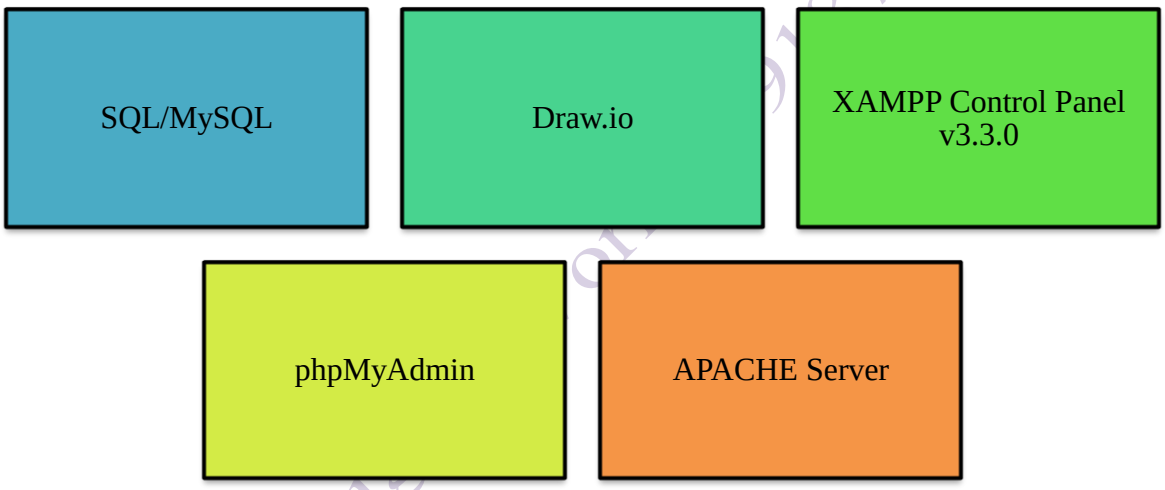


Figure 1: Applied tools

(Source: Self-developed)

The tools used in the project are further elaborated in the next section, where their roles and functionalities are explained. The APACHE server, which is a cross-platform open-source web server, is used to create a local server on the system where the database is developed. This server is one of the most commonly used HTTP servers for SQL projects and DBMS globally. It is referred to as a web server, but it is actually an application that runs on HTTP, and it helps to establish a connection between a web browser and the server. One of the significant advantages of using APACHE is that it is compatible with any operating system, including Unix and Windows (Richard, 2022).

The XAMPP Control Panel is another tool used in the project, and it is utilized to manage the APACHE server and MySQL. This tool ensures the efficient management of the software applications that are included in the XAMPP package. The phpMyAdmin cannot be started on the localhost without starting both the APACHE server and MySQL through the XAMPP Control Panel (Azad *et al.*, 2019). Therefore, before the practical creation of the database, it is essential to ensure that the MySQL and APACHE servers are running in the control panel. The Start/Stop button under the Actions section in the control panel can be used to initiate these servers, and once they are running, the APACHE server and MySQL are confirmed to be active (Setiawan *et al.*, 2019). The XAMPP Control Panel is an important tool that ensures the smooth running of the project by enabling the easy management of the necessary software applications (West and Prettyman, 2018).

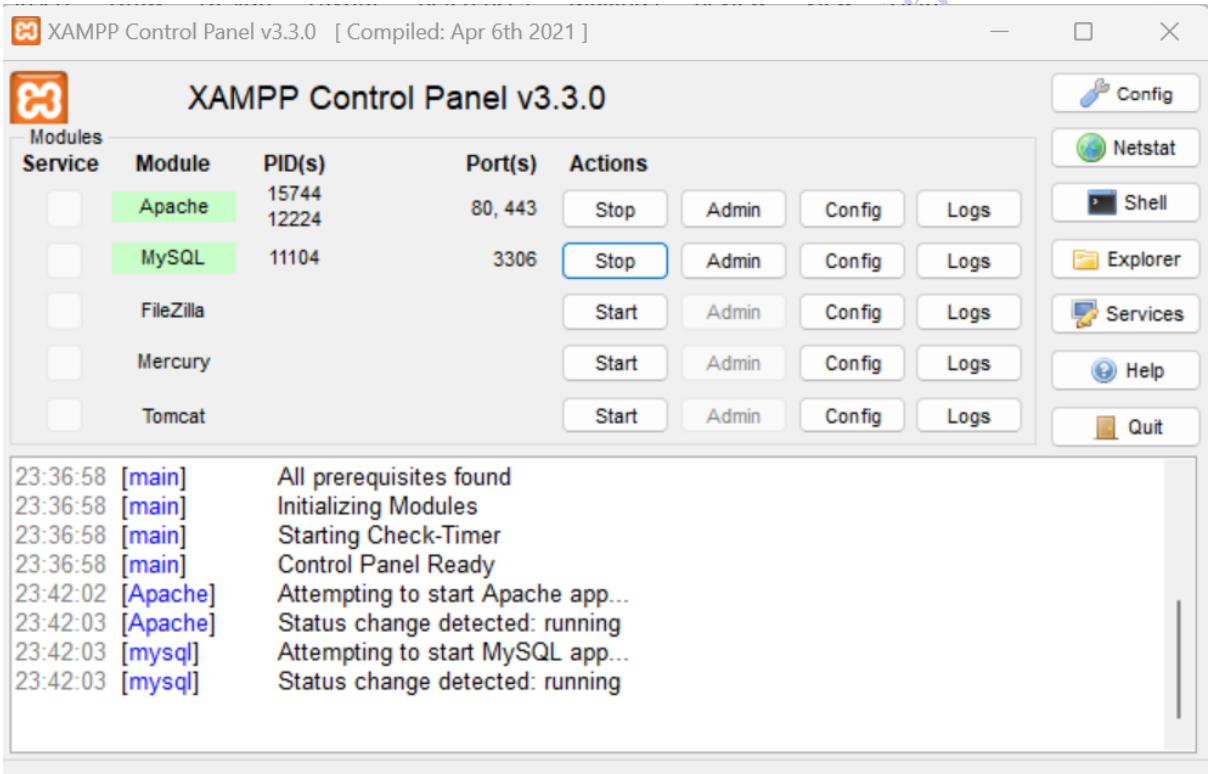


Figure 2: XAMPP

(Source: Application on the system)

The next step involves setting up the platform for developing the database in the local host using the following tool:

phpMyAdmin: A widely-used administration tool that is both free and open-source, phpMyAdmin is used for database creation and management using MariaDB or MySQL in this project. It is written in PHP and serves as a third-party application for managing databases. This tool also enables the display of multiple query results and offers the

advantage of converting databases and query results into PDF graphics, which are included in this project. Like the APACHE server, phpMyAdmin is also an OS-free application that can be run on any operating system, and it is compatible with different.

Table	Action	Rows	Type	Collation	Size	Overhead
administrasi	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
berita	Browse Structure Search Insert Empty Drop	7	InnoDB	latin1_swedish_ci	16 KiB	-
evaluasi	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
harga	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
kualifikasi	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
paket	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB	-
pemenang	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
perusahaan	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	32 KiB	-
peserta	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
teknis	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
user	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
11 tables	Sum	37	InnoDB	latin1_swedish_ci	192 KiB	0 B

Figure 3: phpMyAdmin portal

(Source: Rukmana and Muslim, 2017)

Draw.io is a free and open-source software provided by diagrams.net and it can be used to develop various types of diagrams such as flow charts, graphs, and use-case diagrams. The tool is used in this project to create the use-case diagram. The use of free and open-source software tools did not increase the cost of the project. Draw.io is an online application that does not require installation into the system and does not consume system space. The tool offers different Use-Case diagrams that can be selected from the list (Escalona *et al.*, 2021). Overall, Draw.io is a versatile and convenient tool for creating diagrams, which makes it a valuable addition to the project without any extra cost.

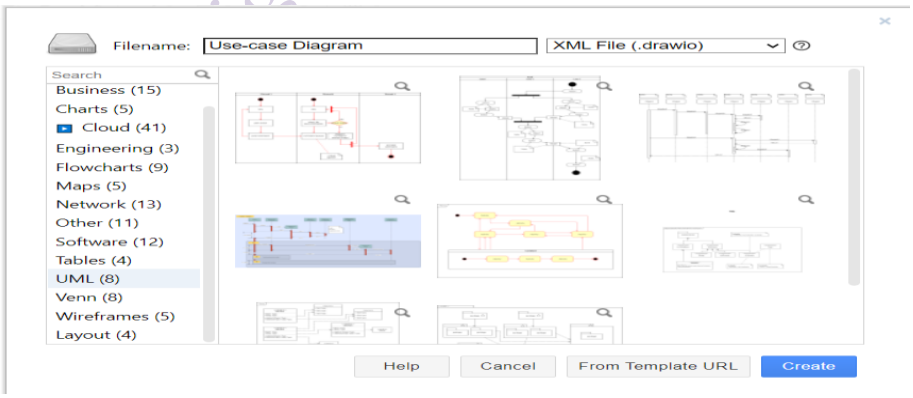


Figure 4: Overview of Draw.io

(Source: Draw.io, 2023)

MySQL is a popular RDBMS (relational database management system) used to manage and store data in a structured format. It uses SQL (Structured Query Language) to write and execute queries, which is a standard language used in the database industry. SQL comprises of DDL (data definition language) and DML (data manipulation language), among other

query-specific statements. DDL is used to create, modify or delete database objects, such as tables, indexes, and views. For example, the CREATE command is used to create a table, ALTER to modify a table, DROP to delete a table, and RENAME to rename a table. Deleting or truncating data from a table is also considered an example of DDL. On the other hand, DML is used to manipulate data within the tables. Inserting, deleting, updating, and merging rows are considered examples of DML (Ozgur *et al.*, 2018).

MySQL is known for its ease of installation and accessibility. It can be used across multiple operating systems, including Windows, macOS, and Linux, and can be installed on a local machine or a remote server. Its popularity is also attributed to its speed, reliability, and robustness in handling large databases with millions of rows. MySQL is used by many popular websites and applications, such as WordPress, Facebook, and Twitter.

In addition to its features, MySQL has an active and supportive community that contributes to its development and maintenance. MySQL's open-source nature means that developers worldwide can contribute to its improvement and add new features, making it more adaptable to users' needs.

MySQL is not just limited to database management. Its usage extends to business intelligence, data warehousing, and other big data applications. It can be integrated with other software tools like Hadoop and Spark to manage and store data at a larger scale.

In conclusion, MySQL is a versatile, user-friendly, and robust RDBMS system, making it a popular choice for developers and businesses worldwide (Kheirabadi *et al.*, 2019). Its compatibility, ease of use, and active community make it an excellent tool for managing and storing data.

Database Design Decision

The database design process involves two main stages: logical design and physical design. The logical design is first created using a Use-Case diagram and an ERD to identify any queries related to individual entities or combinations of entities. The Use-Case diagram helps to understand the activities between staff and pupils, while the ERD provides the relationships between the entities and their cardinalities. The physical design of the database is then developed based on the logical design, which determines the structure of the tables, primary keys, and other options. Each entity in the ERD corresponds to a table in the database, with each attribute becoming a column of the table. The Use-Case actions provide an idea about the tuples and additional attributes required to make the query results practical.

Finally, queries are executed to ensure the database is functioning properly. The steps involved in the database design process are depicted in the accompanying image.

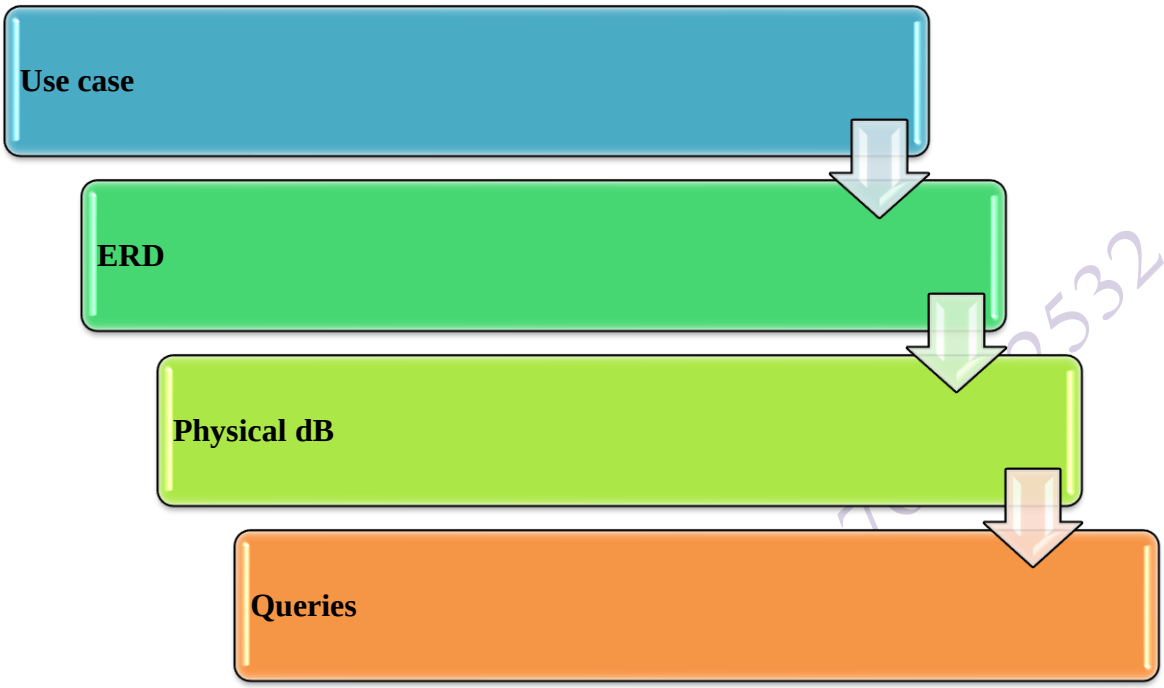


Figure 5: Steps to design the database

(Source: Self-developed)

The Use-Case diagram is a graphical representation that displays the interactions between actors and the system. The diagram uses circular or oval boxes to represent the use cases and arrows to show how they are connected to the actors. The designer created a Use-Case diagram to understand how users interact with the database and what tasks they perform. This information is then used to develop a quality DBMS that meets the needs of the users. The Use-Case diagram for this project was developed based on data provided by the client regarding a secondary school in Luton. For example, the client indicated that the subject entity must be cross-referenced with the pupils-entity using student-id. This means that students can select at least one subject from the subject's table, and these two tables will be associated with each other using student-id. The Use-Case diagram was created using draw.io and is presented in the figure below. A well-designed Use-Case diagram is crucial in developing an effective database system.

Use-case diagram

Use case diagrams have several advantages in software development:

Communication: Use case diagrams provide a clear and concise way to communicate the functionality of a system to stakeholders, including developers, testers, and end-users. The visual representation of the system's functionality helps everyone to understand what the system is supposed to do, and how the various components interact with each other.

Requirement gathering: Use case diagrams are an effective tool for gathering requirements. They provide a structured way to capture the functional requirements of the system, and help ensure that all necessary use cases are identified.

Analysis: Use case diagrams can be used to analyse and refine the functionality of a system. By reviewing the use cases and identifying any missing or unnecessary features, developers can streamline the system and ensure that it meets the needs of the end-users.

Testing: Use case diagrams can also be used as the basis for test cases. By defining the use cases and their associated scenarios, testers can develop comprehensive test plans that ensure the system functions as intended.

System design: Use case diagrams can help guide the design of the system. By identifying the actors and their interactions with the system, developers can design a system that meets the needs of the end-users.

Overall, use case diagrams are an essential tool for software development. They provide a clear and concise way to communicate, gather requirements, analyse, test, and design the functionality of a system.

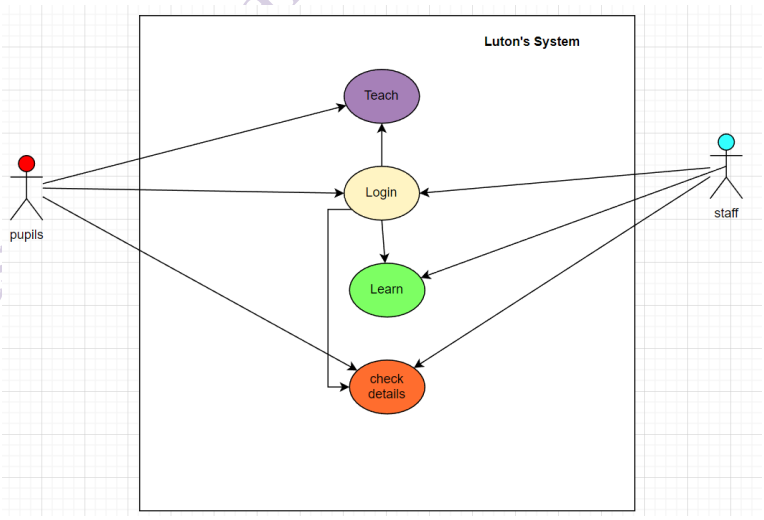


Figure 6: Use-case diagram

(Source: Draw.io, 2023)

The given information defines two players, staff and pupils, with their associated activities in the system. The use-case diagram provides a visual representation of how these players interact with the system and what activities they can perform.

For staff, although no activities are explicitly mentioned, it is inferred that they teach subjects to pupils based on the information provided about the subject entity and the pupils table. Therefore, the "teach" activity is added as part of the improvisation. Additionally, staff can check their own details as well as the details of the pupils they teach, so the "check details" Use-Case is also included.

For pupils, the given information specifies that they are students who are taught one or more subjects by the staff. Pupils can also check their own details, which is represented by the "check details" Use-Case.

Use-Case diagrams have several advantages in designing a database system. They provide a clear understanding of the roles and responsibilities of the users or actors in the system, which helps in defining the scope and requirements of the system. Use-Case diagrams also help in identifying the activities that need to be performed by the actors and the interactions between the actors and the system. This information can be used to design a database that meets the needs of the users and is efficient in performing the required activities.

In conclusion, the use-case diagram provides a clear understanding of the activities that can be performed by the staff and pupils in the system. While the activities for staff were inferred based on the information provided, the use-case diagram helps in justifying the activities associated with the players. Use-Case diagrams have several advantages in designing a database system and help in defining the scope and requirements of the system.

ER diagram

The entity-relationship model or ERD is a diagram that shows the entire database, including entities and the relationships between them. In this case, the information provided mentioned three entities from a local secondary school in Luton: subjects, pupils, and staffs. The ERD shows how these entities are interrelated, with rectangular boxes representing the entities and diamond shapes representing the relationships. Arrows indicate which entities hold the relationship in the diamond-shaped boxes (Rashkovits and Lavy, 2021).

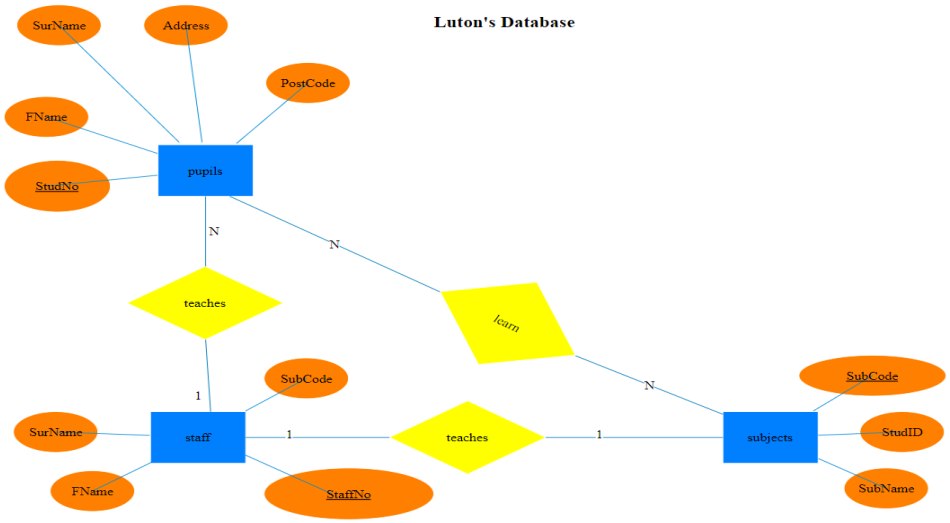


Figure 7: ERD

(Source: Self-developed using Draw.io)

The above diagram illustrates an Entity Relationship Diagram (ERD) that shows the relationships between three entities, namely "pupils", "subjects", and "staff". Each entity is enclosed in a rectangular box, and the attributes associated with each entity are shown inside an oval-shaped box. The entities are connected through lines that indicate the cardinality of the relationship between them, as represented by a diamond-shaped box.

Cardinality refers to the number of occurrences of one entity that are associated with the number of occurrences of another entity (Arruda and Madhavji, 2017). This concept can be better understood with the help of the examples provided in the ERD. For instance, the "pupils" entity is related to the "subjects" entity through the "study" relationship. As shown in the diagram, many students can take many subjects simultaneously, indicating a many-to-many or N-to-N cardinality between the "pupils" and "subjects" entities. This is represented by an 'N' on the relationship line between the "pupils" entity and "study" relationship, and another 'N' on the relationship line between the "study" relationship and "subjects" entity.

Similarly, the "staff" entity is related to the "subjects" entity through the "teaches" relationship, which has a cardinality of 1-to-1. This means that one teacher can teach only one subject, and one subject can be taught by only one staff member. Hence, the 1-to-1 cardinality between the "subjects" and "staffs" entities is justified. The "teaches" relationship between the "staffs" and "pupils" entities has a 1-to-N cardinality, indicating that one staff member can teach many students about one subject, but one student cannot study the same subject from multiple staff members.

The attributes of each entity are also shown in the diagram. For instance, the "pupils" entity has attributes such as "StudNo", which is a unique student number and a primary key to identify each student. Other attributes of the "pupils" entity include "FName", "SurName", "Address", and "PostCode". Similarly, the "subjects" entity has attributes such as "SubCode", which is a unique key and a primary key, and "SubName", which can be identified with its associated "SubCode". The "subjects" entity also has a foreign key attribute which is "StudID". It provides cross-references between the "subjects" and "pupils" entities.

The "staff" entity has three main attributes, namely "StaffNo", "FirstName", and "SurName". The "StaffNo" attribute is a primary key to identify each staff member, and each "staffs" entity can be identified with this attribute. The "staff" entity also has one foreign key attribute, "SubCode", which refer to the "SubCode" attribute of the "subjects" entity, respectively. These modifications are necessary to enable the database to work for queries related to combined entities.

In summary, the ERD provides a visual representation of the relationships between three entities, along with their attributes and cardinality. It helps to understand the complex relationships between different entities and their associations. The attributes and foreign key constraints play an essential role in defining the relationships between entities, and the cardinality helps to understand the number of occurrences of one entity that are associated with the number of occurrences of another entity.

The physical design of the logical implementation

This section involves implementing the logical design presented through an ERD and Use-Case diagram into a physical design, which includes creating the physical database, tables, rows (tuples), and columns (attributes) within the tables. The database is created through the phpMyAdmin console's graphical user interface, but SQL statements are also provided. The resulting database comprises three tables (entities) and is shown below.

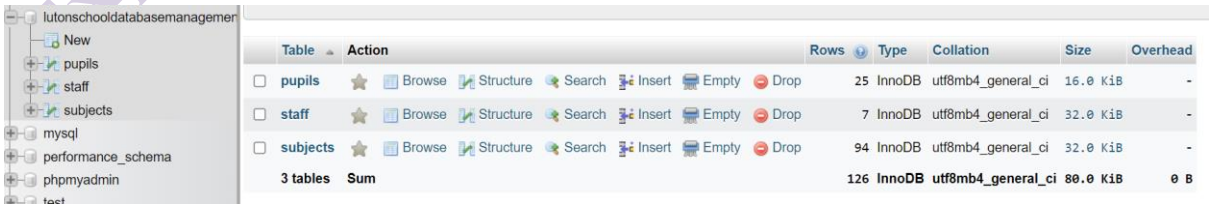


Figure 8: Luton's School Database Management System

(Source: Self-developed using phpMyAdmin)

The image displays a MySQL database for secondary schools containing three entities - "pupils", "staff", and "subjects". The following is the DDL statement to create the database.

CREATE DATABASE LutonSchoolDatabaseManagementSystem;

Shown below is the elaboration of each table's development. The table's schema with its respective columns or attributes, as depicted in the ERD above, is presented.

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0011 seconds.)

```
CREATE TABLE pupils ( StudNo INT PRIMARY KEY, FName VARCHAR(50), SurName VARCHAR(50), Address VARCHAR(100), PostCode VARCHAR(10) );
```

#	Name	Type	Collation	Attributes	Null	Default
1	**StudNo**	int(11)			No	*None*
2	**FName**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*
3	**SurName**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*
4	**Address**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*
5	**PostCode**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*

Figure 9: pupils-schema

(Source: Self-developed using phpMyAdmin)

The size of the fields for each column is indicated by the numbers in the parentheses. Following that, the structure of the "subjects" table is displayed, where the primary key is the unique "SubCode" attribute, which is underlined in the ERD. To establish the cross-reference between these two tables, the foreign key of the table is "StudID" to the “StudNo” of "pupils" table, as requested.

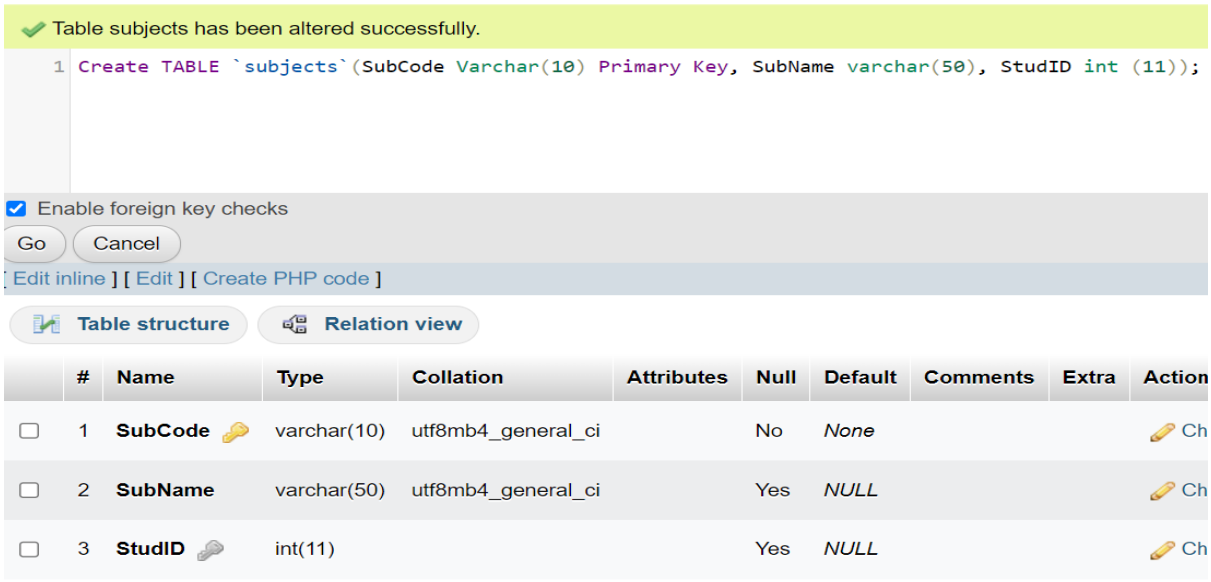


Figure 10: "subjects" table schema

(Source: Self-developed using phpMyAdmin)

Now, the "StudID" in this table must be the foreign key to the "StudNo" of the "pupils" table. This is done using the graphical user interface of the phpMyAdmin. Though. To it can be done by altering the "subjects" table with the following code:

```
ALTER TABLE subjects
ADD CONSTRAINT fk_subjects_pupils
FOREIGN KEY (StudID)
REFERENCES pupils(StudNo);
```

The graphical user interface is used in the following way:

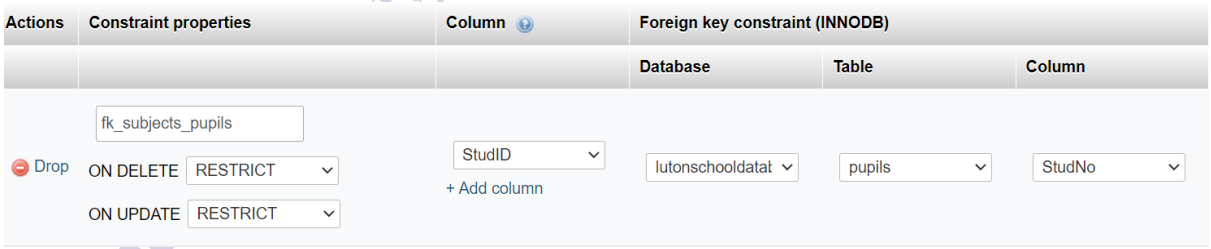


Figure 11: Foreign key creation in the "subjects" table

(Source: Self-developed using phpMyAdmin)

Similarly, the "staff" table is developed in the following way:

Name	Type	Collation	Attributes	Null	Default
StaffNo	int(10)			No	*None*
FName	varchar(50)	utf8mb4_general_ci		Yes	*NULL*
SurName	varchar(50)	utf8mb4_general_ci		Yes	*NULL*
SubCode	varchar(10)	utf8mb4_general_ci		Yes	*NULL*

Figure 12: "staff" table schema

(Source: Self-developed using phpMyAdmin)

Now, the "StudID" in this table must be the foreign key to the "StudNo" of the "pupils" table. This is done using the graphical user interface of the phpMyAdmin. Though. To it can be done by altering the "subjects" table with the following code:

ALTER TABLE staff

ADD CONSTRAINT fk_staff_subjects

FOREIGN KEY (SubCode)

REFERENCES subjects(SubCode);

The graphical user interface is used in the following way:

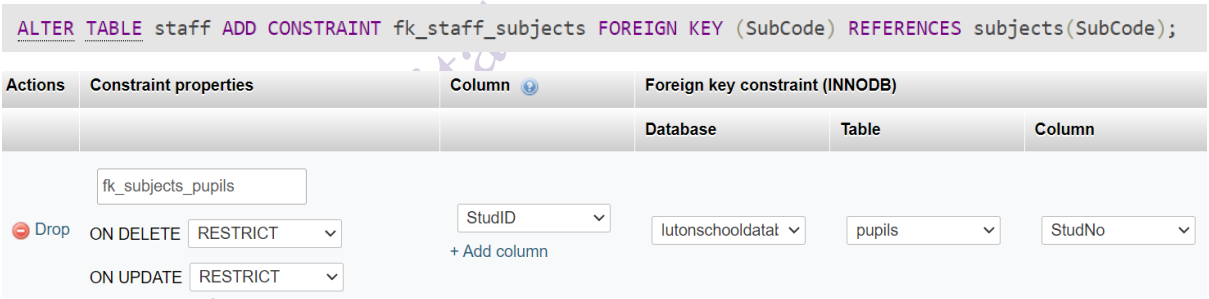


Figure 13: Foreign key creation for the "staff" table

(Source: Self-developed using phpMyAdmin)

However, the ON DELETE and ON UPDATE options are updated to CASCADE with the following code:

ALTER TABLE `subjects` DROP FOREIGN KEY `fk_subjects_pupils`; ALTER TABLE `subjects` ADD CONSTRAINT `fk_subjects_pupils` FOREIGN KEY (`StudID`) REFERENCES `pupils`(`StudNo`) ON DELETE CASCADE ON UPDATE CASCADE;

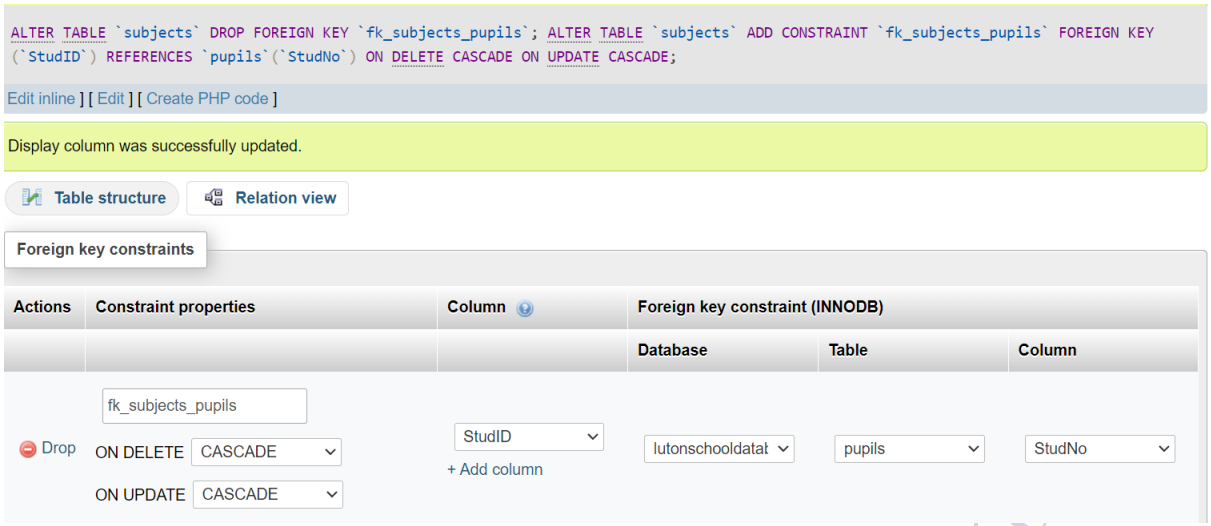


Figure 14: CASCADING – "subject" table

(Source: Self-developed using phpMyAdmin)

Similarly, cascading is done for the "staff" table with the following way:

ALTER TABLE `staff` DROP FOREIGN KEY `fk_staff_subjects`; ALTER TABLE `staff` ADD CONSTRAINT `fk_staff_subjects` FOREIGN KEY (`SubCode`) REFERENCES `subjects`(`SubCode`) ON DELETE CASCADE ON UPDATE CASCADE;

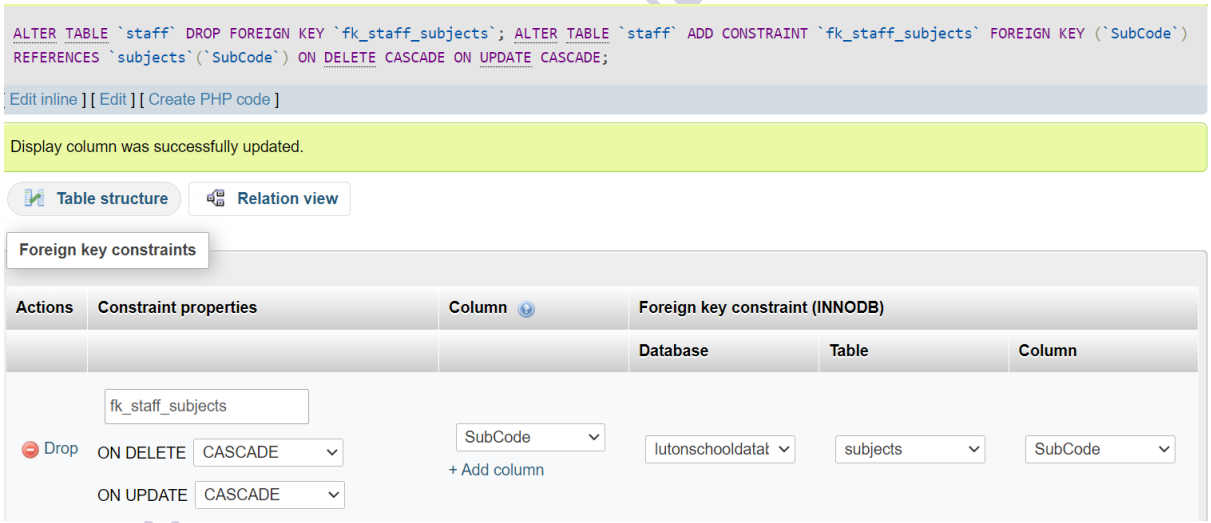


Figure 15: Cascading "staff" table

(Source: Self-developed using phpMyAdmin)

The next task is to insert data into the tables as following:

Once the tables have been created in the database, the next step is to insert data into the tables. This is done using the INSERT SQL command. To view the data that has been inserted into the tables, the SELECT SQL command is used.

The insert command used to input data into the "pupils" table is following:

INSERT INTO pupils (StudNo, FName, SurName, Address, PostCode)

VALUES

(1, 'Jones', 'Smith', 'Liverpool', 'WC2N 5DU'),

(2, 'Smith', 'Richard', 'London', 'L2 2DP');

For instance, if we take the "pupils" table, the data inserted into it can be viewed by executing the appropriate SELECT command. The resulting table can then be viewed, as shown in Figure 16.

Figure 16: Data in "pupils"

(Source: Self-developed using phpMyAdmin)

Similarly, more data inputted into the table using the same command and the result is as follows:

StudNo	FName	SurName	Address	PostCode
1	Jones	Smith	Liverpool	WC2N 5DU
2	Smith	Richard	London	L2 2DP
3	Doe	Jane	Manchester	M1 2NG
4	Adams	John	Bristol	BS1 1LT
5	Taylor	Peter	Birmingham	B1 1BB
6	Brown	Sarah	Oxford	OX1 1AA
7	Lee	David	Cambridge	CB1 1BB
8	Wilson	Helen	Glasgow	G1 1PP
9	Khan	Amina	Leeds	LS1 1XX
10	Baker	Luke	Sheffield	S1 2GU
11	Wang	Ming	Nottingham	NG1 1XX
12	Nguyen	Linh	Cardiff	CF10 1XX
13	Fernandez	Carlos	Newcastle	NE1 1DE
14	Gupta	Priya	Belfast	BT1 1XX
15	Chen	Wei	Southampton	SO14 1AA
16	Hernandez	Diego	Edinburgh	EH1 1XX

StudNo	FName	SurName	Address	PostCode
17	Perez	Rafael	Brighton	BN1 1XX
18	Kim	Min-Jae	Bournemouth	BH1 1XX
19	Garcia	Alejandro	Plymouth	PL1 1XX
20	Nakamura	Takashi	Dundee	DD1 1XX
21	O'Connor	Sean	Aberdeen	AB10 1XX
22	Hassan	Ali	Derby	DE1 1XX
23	Murphy	Patrick	York	YO1 1XX
24	Chang	Liu	123 Main St	90210
25	Garcia	Martinez	456 Elm St	12345

Figure 17: All data in "pupils" table

(Source: Self-developed using phpMyAdmin)

Hence, 25 students are registered in the secondary school till now.

In the same way, data is inserted into the "subjects" table:

```
INSERT INTO subjects (SubCode, SubName, StudId)
VALUES
('ENG', "ENGLISH', 5),
……
;
```

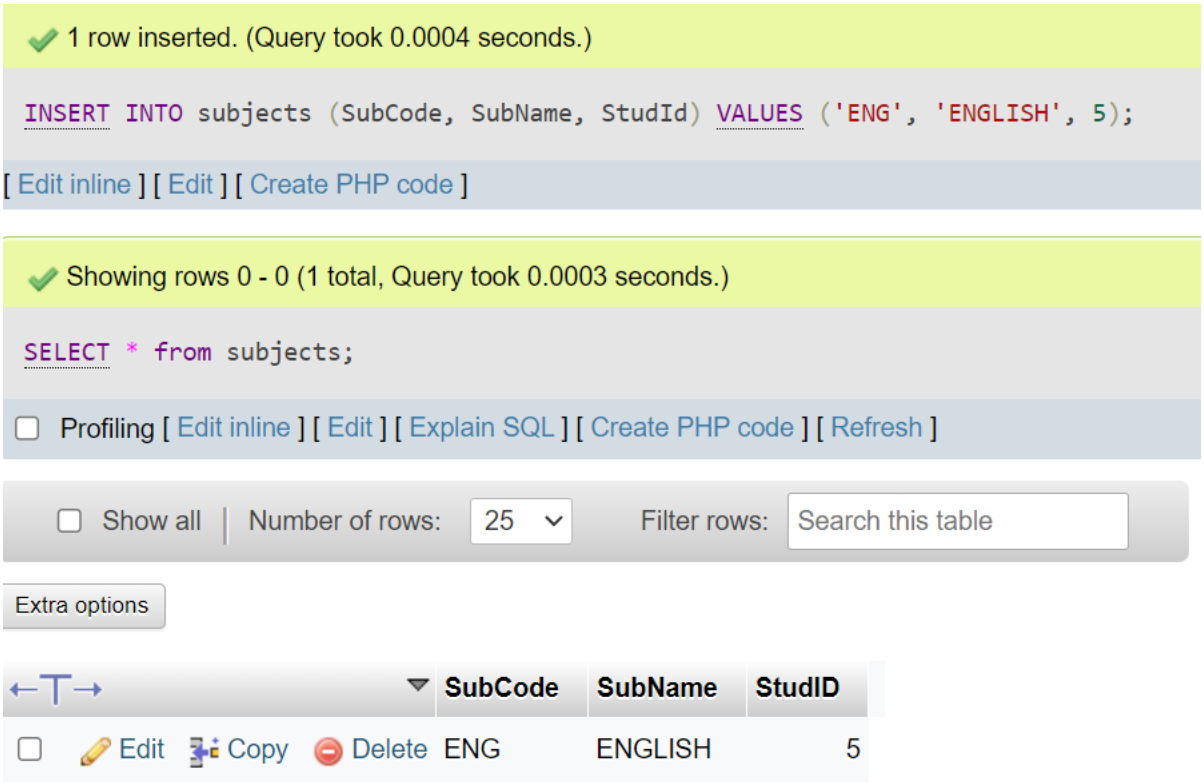


Figure 18: Successful insertion of data into "subjects" table

(Source: Self-developed)

In the same way, all of the data are inserted into this table and the result is as followed:

	SubCode	SubName	StudID
te	ENG	ENGLISH	5
te	ENG	ENGLISH	6
te	ENG	ENGLISH	7
te	ENG	ENGLISH	8
te	ENG	ENGLISH	9
te	ENG	ENGLISH	10
te	ENG	ENGLISH	11
te	ENG	ENGLISH	12
te	ENG	ENGLISH	13
te	ENG	ENGLISH	14
te	ENG	ENGLISH	15
te	ENG	ENGLISH	16
te	ENG	ENGLISH	17
te	ENG	ENGLISH	18
te	ENG	ENGLISH	19

SubCode	SubName	StudID
ENG	ENGLISH	20
ENG	ENGLISH	21
GEO	Geography	2
GEO	Geography	5
GEO	Geography	7
GEO	Geography	9
GEO	Geography	11
GEO	Geography	13
GEO	Geography	16
GEO	Geography	19
GEO	Geography	21
GEO	Geography	23
GEO	Geography	25
HIST	History	1
HIST	History	2
HIST	History	5
HIST	History	7

Figure 19: All data in "subjects" table

(Source: Self-developed)

There are 94 rows of data in the "subjects" table 7 different subjects selected by multiple students with different StudID. All of them cannot fitted into same screen so part of the table is shown in Figure 19.

Now, it is time to insert data into the staff table.

```
INSERT INTO staff (StaffNo, FName, SurName, SubCode)
VALUES
(1, 'Martini', 'Butcher', 'ENG'),
…….
;
```

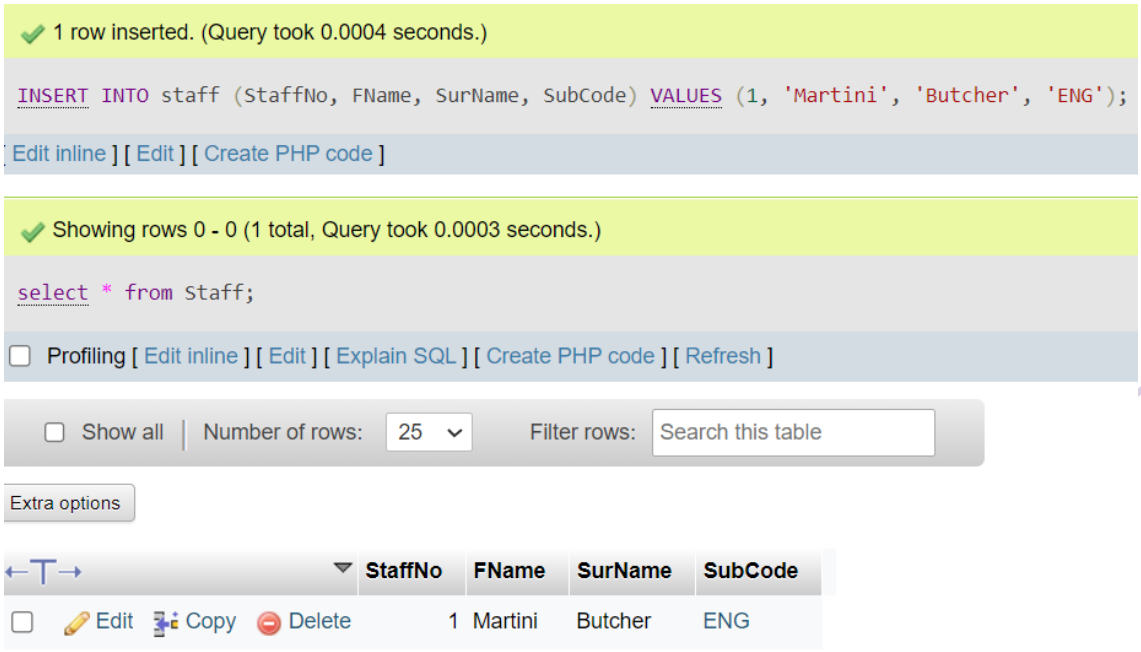


Figure 20: Successful insertion of data into "staff" table using the command above

(Source: Self-developed)

Similarly, data for all of the subjects are inserted into the table and viewed using the same SELECT command as below:

StaffNo	FName	SurName	SubCode
1	Martini	Butcher	ENG
2	John	Doe	MFL
3	Jane	Smith	GEO
4	Michael	Johnson	HIST
5	David	Brown	MATH
6	Jennifer	Davis	PE
7	William	Wilson	SCI

Figure 21: All data in "staff" table

(Source: Self-developed)

Figure 21 showed that there are 7 rows which is supported by the data in the "subjects" table as there are 7 subjects listed in the table as well.

Queries

During the initial stages of system design, it is often assumed that a database will need to perform various functions such as queries that involve tables with data related to pupils, subjects, and staff. In this context, it is important to consider the relationships between these

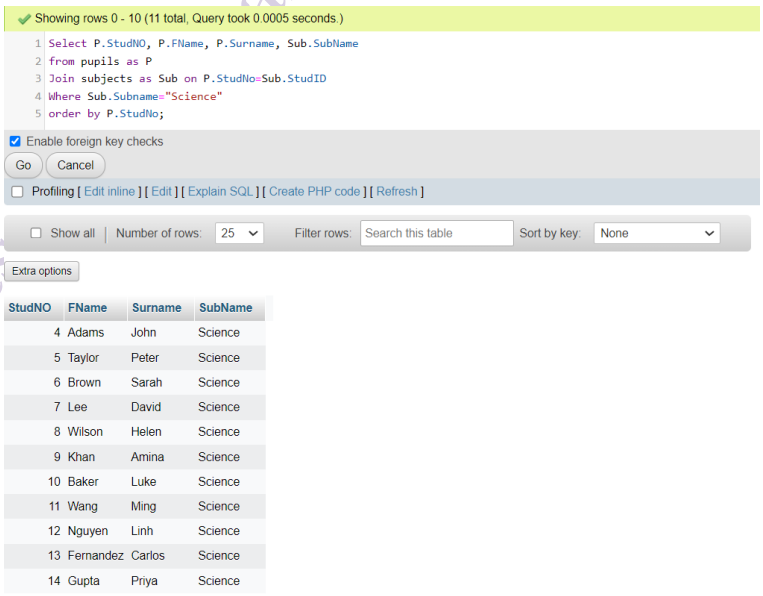
tables in order to effectively design and implement the database. One such query that can be performed is the one between the "pupils" and "subjects" tables, which can provide useful insights and information about the subjects that each pupil is enrolled in. The results of this query can be analyzed and explained in detail to gain a deeper understanding of the data and its relationships. This information can then be used to optimize the database's performance and functionality, as well as to develop more efficient and effective queries in the future.

Query including pupils and subjects

This query is done to get the name of the pupils who are enrolled from the “science” subject. For this the query include joining of the “pupils” table and “subjects” table based on “StudID” in “staff” table as the foreign key to the “StudNo” attribute as the primary key of the “pupils” table. The query is as follows:

```
Select P.StudNO, P.FName, P.Surname, Sub.SubName
from pupils as P
Join subjects as Sub on P.StudNo=Sub.StudID
Where Sub.Subname="Science"
order by P.StudNo;
```

Let’s see what are the result of the query and if it together with the tables and databases are working fine.



StudNO	FName	Surname	SubName
4	Adams	John	Science
5	Taylor	Peter	Science
6	Brown	Sarah	Science
7	Lee	David	Science
8	Wilson	Helen	Science
9	Khan	Amina	Science
10	Baker	Luke	Science
11	Wang	Ming	Science
12	Nguyen	Linh	Science
13	Fernandez	Carlos	Science
14	Gupta	Priya	Science

Figure 22: Result of the first query

(Source: Self-developed)

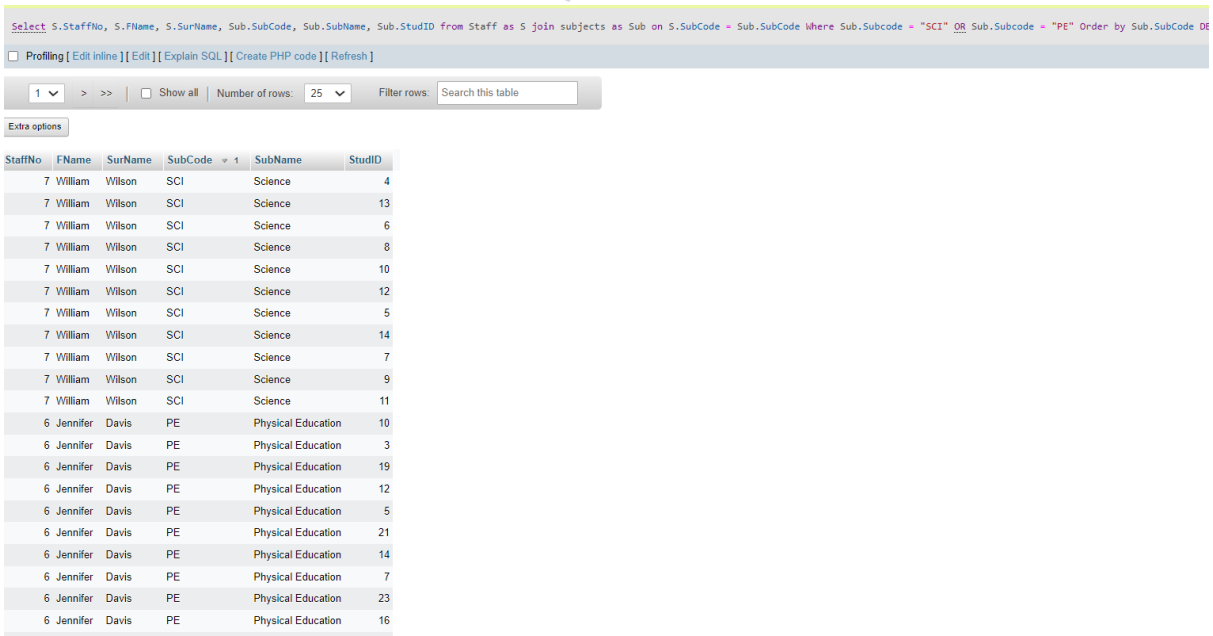
The result showed that there are 11 records which matched the query and “Science” is the subject name for all of them. Hence, it showed the join between the two tables are working fine. That also means, selection of foreign key and primary of between the tables and code to create the tables are also working. Therefore, the logical implementation of the physical database is successful.

Query including staff and subjects

“subjects” table and the “staff” table can be joined with each other using the “SubCode” attribute presents in both of them. Now, the name of the subjects with SubCode and StudID can be retrieved. To make it complex, let’s to it for two subjects which are “Physical Education” and “Science”. The query for this is:

```
Select S.StaffNo, S.FName, S.SurName, Sub.SubCode, Sub.SubName, Sub.StudID
from Staff as S
join subjects as Sub on S.SubCode = Sub.SubCode
Where Sub.Subcode = "SCI" OR Sub.Subcode = "PE"
Order by Sub.SubCode DESC;
```

The result is as follows:



StaffNo	FName	SurName	SubCode	SubName	StudID
7	William	Wilson	SCI	Science	4
7	William	Wilson	SCI	Science	13
7	William	Wilson	SCI	Science	6
7	William	Wilson	SCI	Science	8
7	William	Wilson	SCI	Science	10
7	William	Wilson	SCI	Science	12
7	William	Wilson	SCI	Science	5
7	William	Wilson	SCI	Science	14
7	William	Wilson	SCI	Science	7
7	William	Wilson	SCI	Science	9
7	William	Wilson	SCI	Science	11
6	Jennifer	Davis	PE	Physical Education	10
6	Jennifer	Davis	PE	Physical Education	3
6	Jennifer	Davis	PE	Physical Education	19
6	Jennifer	Davis	PE	Physical Education	12
6	Jennifer	Davis	PE	Physical Education	5
6	Jennifer	Davis	PE	Physical Education	21
6	Jennifer	Davis	PE	Physical Education	14
6	Jennifer	Davis	PE	Physical Education	7
6	Jennifer	Davis	PE	Physical Education	23
6	Jennifer	Davis	PE	Physical Education	16

Figure 23: Result of Query 2

(Source: Self-developed)

The result showed how only subject code with PE and SCI are selected. Moreover, the StaffNo is only 6 and 7 realted to PE and SCI respectively. SubCode can be found in both

tables though, here it is fetched from the “subjects” table while StaffNo, FName and SurName of staff are fetched from staff table. Staff table doesn’t include name of the subjects and StudID which are fetched again from the “subjects” table. Hence, the query, joining between the tables, primary keys, foreign keys and the database are working fine.

Both of the queries and results of them ensured the effectiveness and validity of the database.

Query 3: Counting number of students for PE and Science subjects

Moreover, if counting the number of students for each subject, the code can be changed like below and the result as follows:

Code: Select S.StaffNo, S.FName, S.SurName, Sub.SubCode, Sub.SubName, Count(Sub.StudID) as NoOfStudents
from Staff as S
join subjects as Sub on S.SubCode = Sub.SubCode
Where Sub.Subcode = "SCI" OR Sub.Subcode = "PE"
Group by Sub.SubName
Order by Sub.SubCode DESC;

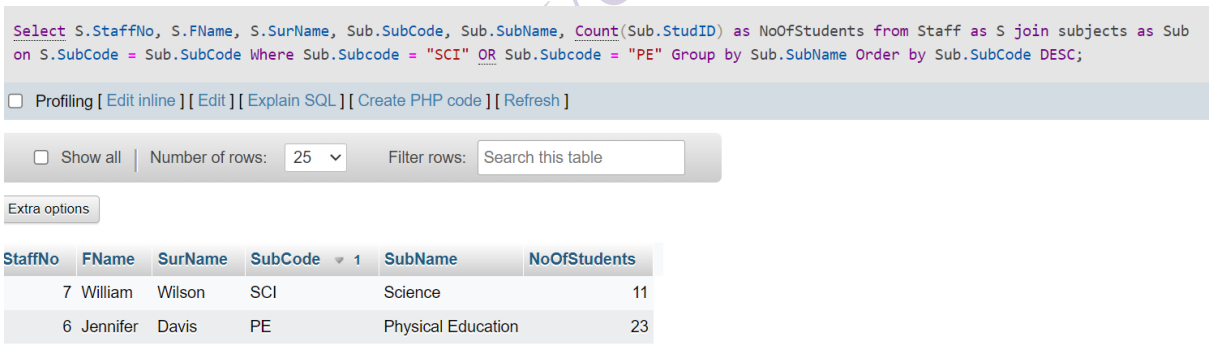


Figure 24: Result of Query 3

(Source: Self-developed)

This showed there are 11 students learning science under William Wilson and 23 students in PE classes under Jennifer Davis.

Conclusion

The process of designing a database can be a complex and iterative one, requiring careful consideration of many factors. The designer for the local secondary school in Luton has taken a thoughtful approach, using a variety of tools and techniques to create an effective database that meets the school's needs.

The first step in the design process was to develop a logical design using entity relationship and use-case diagrams. The entity relationship diagram provided a visual representation of the data entities and their relationships to each other, while the use-case diagram helped identify the queries needed to test the database and what types of data to insert into the tables. This stage of the design process is critical, as it sets the foundation for the physical design of the database.

The designer then moved on to the physical design of the database, converting the entities from the logical design into tables in the physical database and their attributes into columns. This is where the power of SQL and other database tools comes into play. By using tools such as phpMyAdmin and Apache server, the designer was able to create a robust and functional database that would serve the needs of the school.

However, as with any complex project, there were some challenges that needed to be addressed. In this case, the designer encountered issues with cross-referencing the "staff" table with the other tables in the database. To solve this problem, the designer added a foreign key from the "subjects" table to the "staff" table. This modification ensured that any queries involving both tables would produce the expected results.

While the designer could have created a separate entity or table for the foreign keys, doing so would have created unnecessary pressure on the system and consumed additional memory. By limiting the modifications to the "staff" table, the designer was able to maintain the integrity and performance of the database.

Overall, the designer for the local secondary school in Luton has done an excellent job of creating an effective database that meets the needs of the school. By using a variety of tools and techniques and taking a thoughtful and iterative approach to the design process, the designer has created a database that will serve the school well for years to come.

References

Arruda, D. and Madhavji, N.H., 2017, December. Towards a requirements engineering artefact model in the context of big data software development projects: Research in progress. In *2017 IEEE international conference on big data (big data)* (pp. 2314-2319). IEEE.

Azad, B.A., Laperdrix, P. and Nikiforakis, N., 2019. Less is more: quantifying the security benefits of debloating web applications. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 1697-1714).

Draw.io, 2023, *interface*. Available at: https://app.diagrams.net/ [Accessed on 3rd February 2022]

Escalona, M.J., García-Borgoñón, L. and Koch, N., 2021. Don't Throw your Software Prototypes Away. Reuse them!.

Fadli, S. and Sunardi, S., 2018. Perancangan Sistem Dengan Metode Waterfall Pada Apotek Xyz. *Jurnal Manajemen Informatika Dan Sistem Informasi*, *1*(2), pp.29-35.

Jansen, F.M. and Tyler, S., 2021. Young gastrointestinal angle: How to create figures for your journal articles. *United European Gastroenterology Journal*, *9*(7), p.872.

Kadir, A.Z.A., Norodzi, N.A., Aziz, N.S. and Mukhtar, N.Z., 2021. Digital System of Evaluation and Calculation. *International Journal of Synergy in Engineering and Technology*, *2*(2), pp.56-64.

Kheirabadi, M.A., Jafari, M.H., Alizadeh, F., Basiri, Z. and Oveisi, K., 2019. Making an entity-relationship model of the knowledge management process with strategic thinking. *Revista Latinoamericana de Hipertension*, *14*(1), pp.55-62.

Kolonko, K., 2018. Performance comparison of the most popular relational and non-relational database management systems.

Ozgur, C., Coto, J. and Booth, D., 2018. A comparative study of network modeling using a relational database (eg Oracle, mySQL, SQL server) vs. Neo4j. In *CONFERENCE PROCEEDINGS BY TRACK* (p. 156).

Rashkovits, R. and Lavy, I., 2021. Mapping Common Errors in Entity Relationship Diagram Design of Novice Designers. *International Journal of Database Management Systems*, *13*(1), pp.1-19.

Richard, B., 2022, *What Is Apache? An In-Depth Overview of Apache Web Server*. Available at: https://www.hostinger.in/tutorials/what-is-apache [Accessed on 1st July 2022]

Schwab, P.K., Langohr, M.S., Röckl, J., Vöhringer, D.E., Wahl, A.M. and Meyer-Wegener, K., 2019. Query-Driven Enforcement of Rule-Based Policies for Data-Privacy Compliance. In *LWDA* (pp. 42-53).

Setiawan, E.B., Setiyadi, A. and Wahdiniwaty, R., 2019, November. Quality Analysis of Mobile Web Server. In *IOP Conference Series: Materials Science and Engineering* (Vol. 662, No. 2, p. 022043). IOP Publishing.

West, A.W. and Prettyman, S., 2018. Create and test a database and table. In *Practical PHP 7, MySQL 8, and MariaDB Website Databases* (pp. 1-31). Apress, Berkeley, CA.