

The Call for Cthulhu Project

Conor McFeely 19204085
(conor.mcfeely@ucdconnect.ie)



Table of Contents

Section 1: Introduction	4
Section 2: Schematic View	5
Section 3: Normalized View	8
Section 4: Database Views	12
Section 5: Procedural Elements	15
Section 6: Database in Action	17
Section 7: Conclusion	20
Section 8: References	22

List of Figures

Figure 1: schema based on tables on white paper.....	1
Figure 2: ER Diagram	3
Figure 3: ER Diagram 2	5
Figure 4: Description Department	8
Figure 5: Description Professor,	8
Figure 6: Description Projects	9
Figure 7: Description Project Preference:	9
Figure 8: Description Student	9
Figure 9: Description Stream	9
Figure 10: Description Allocation,	10
Figure 11: Output View 1	12
Figure 12: Output View 2	12
Figure 13: Output View 3	13
Figure 14: Output View 4	13
Figure 15: Output View 5	13
Figure 16: Output View 6,	14
Figure 17: Output Trigger 1	15
Figure 18/19: Output Trigger 2	15
Figure 20: Output Trigger 3	16
Figure 21: Output Query 1	17
Figure 22: Output Query 2	18
Figure 23: Output Query 3	18
Figure 24: Output Query 4	19
Figure 25: Output Query 5	19

1. Introduction

The project will look at the issues surrounding Philip Howard Craftlove and the school of Cthulhu Studies trying to allocate projects to final year students within the course at University Miskatonic University.

The database needs to be able to hold all of the student, professor and project information as well as being able to indict key information about each such as what stream do each belong to, does the student want to have a project pre-selected or would they like to make a proposal for the different type of project they would like to do based on their study stream.

The possibility of the course growing at University Miskatonic highlights the need for the database to be scalable and to be able to handle large data and still being able to manoeuvre and view what's needed in each database. The design needs to handle each of the problems University Miskatonic is faced and be scalable for future use.

The project allocation is not the concern for myself as the database designer, but I need to understand fully the data that is available and the best way to support the allocation by storing the data in a relational and convenient way for the users.

This report will describe each way I designed the database and the logic behind my thinking, highlighting the relationships between the entities and normalizing the design as best as possible. The report will also look at some views and queries I felt best suited for the University and to show how useful they can be.

A report from Knauff 2004 highlights the importance of designing a database without touching a computer. Planning will be a key part of the database design, by trying to make everything normalised from the start and not normalising tables as they're being built or when they're handling the data. Without proper planning the database would fail in a lot of aspects and would have an overall bad design.

The information provided will help me first create an ER diagram that will design the database before having to create it, giving me more time to look for problems and to give myself a better starting point to approach all problems and normalise the data as much as possible.

2. Database Plan: A Schematic View

The three tables provided from the white paper highlight that they would like information to be stored into a Student, Project and Preferences table. This is a possible way to store all the key information and entities and it would look like the below. (Figure 1)



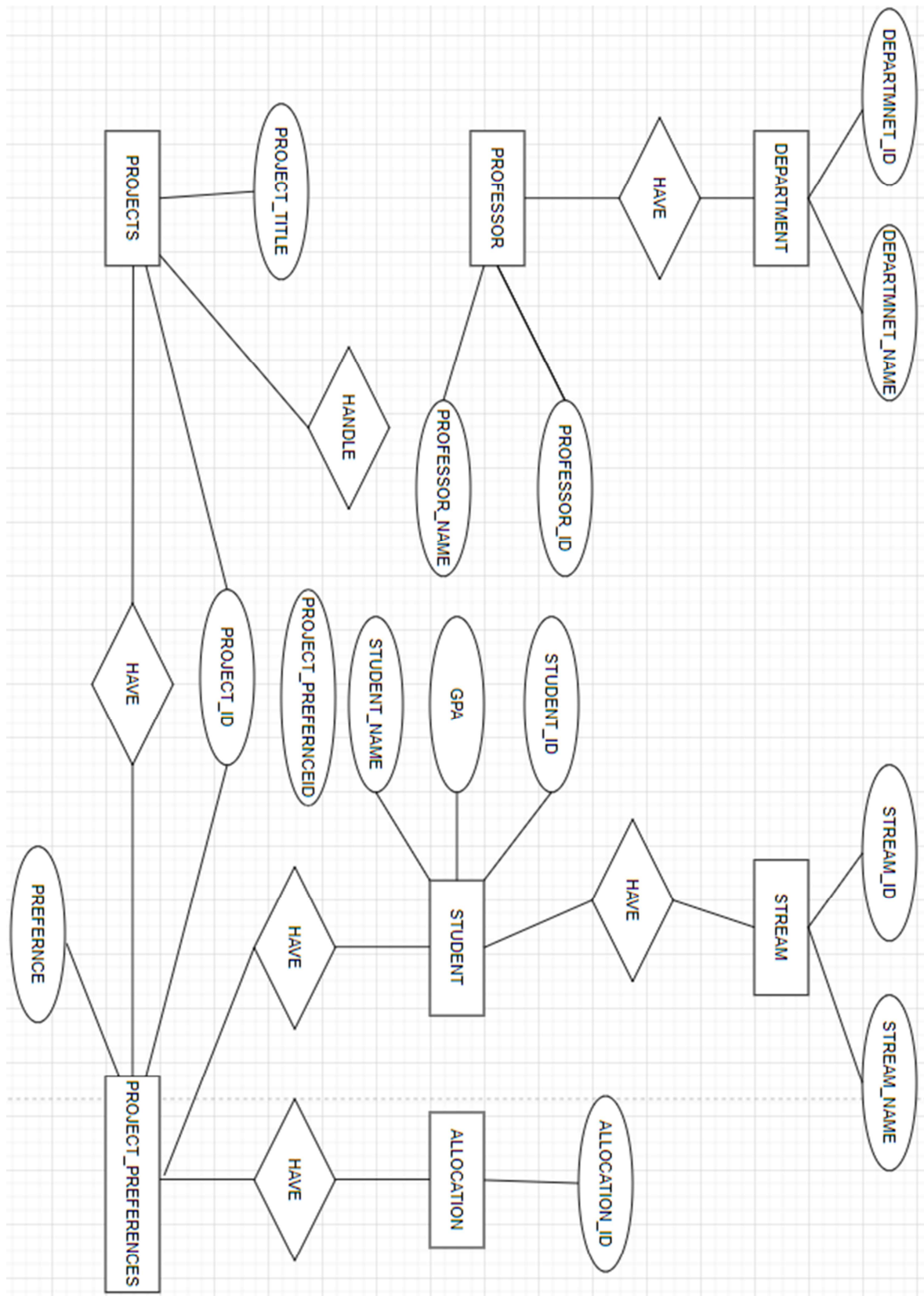
This database lacks normalisation and relationships which would compromise the data being stored and the effectiveness of queries. The possibility of redundant data would be high, and any modifications have the possibility to compromise the integrity of the database (Li, 2019).

Now that the white paper provides a background to what Miskatonic University expects, it provides a blueprint for how to move forward. The first step is to design the database and looking at the relationships and constraints provided from the blueprint.

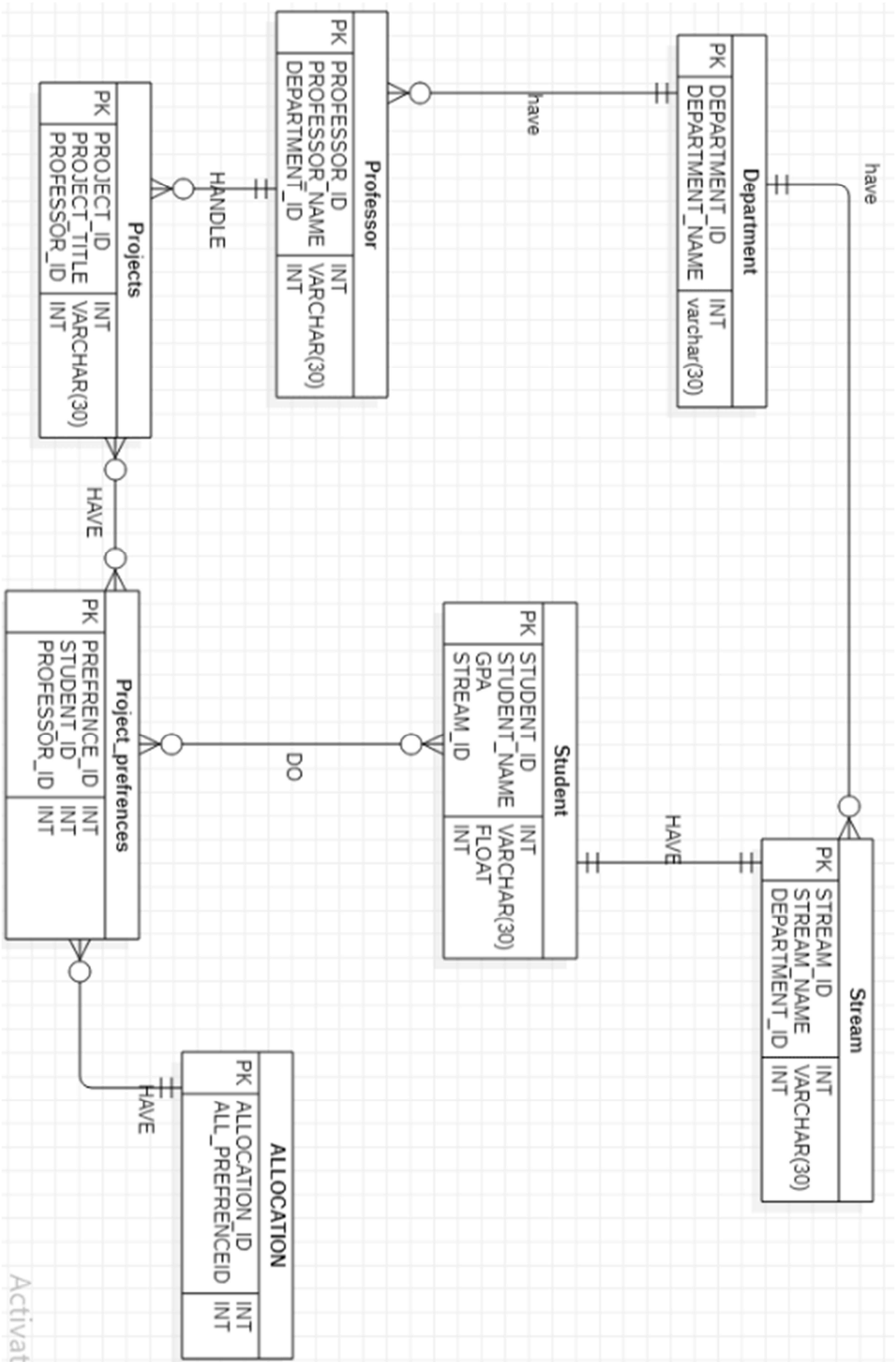
The design needs to be separated more to become more efficient and have a broader amount of entities so they're less diluted. I believe the key entities to be student, project, preference, professor and the department either Cthulhu studies or Dagon studies or both. These set of tables should be able to hold all the information needed if it is readable and functional to operate queries and multiple views.

The reason I went for this design as it becomes a highly normalised database and provides a lot of keys which will be easy for the users to search for the information they need. The following page shows the schema and ER diagram for the database design, highlights the keys and relations used to help make the data as useful as possible.

ER View1 (figure 2)



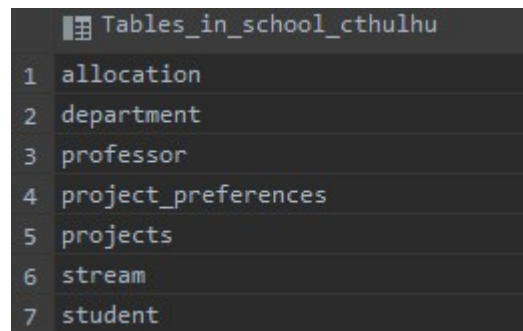
ER View 2 (figure 3)



3. Database Structure: A Normalized View

As figure 2 and 3 illustrate you can see that the database will have seven tables, which means there has been four new added tables to the proposed three tables within the white paper, the seven tables are as follows:

- Department
- Professor
- Projects
- Project_preference
- Student
- Stream
- Allocation



	Tables_in_school_cthulhu
1	allocation
2	department
3	professor
4	project_preferences
5	projects
6	stream
7	student

Department: Will have an auto incremented integer primary key for each department and will have a department name which will hold either CS or DS studies. It will have a one to many relationships with the Professor table and the stream table that'll use DEPARTMENT_ID as the foreign key. (figure 4)

	Field	Type	Null	Key	Default	Extra
1	DEPARTMENT_ID	int	NO	PRI	<null>	auto_increment
2	DEPARTMENT_NAME	varchar(30)	YES		<null>	

Professor: Will have an auto incremented integer primary key that'll create a unique ID for each professor, it will hold the Professor name and have a foreign key of DEPARTMENT_ID from Department table. It has a one to zero or many relationships with the projects table with PROFFESSOR_ID being the foreign key. (figure 5)

	Field	Type	Null	Key	Default	Extra
1	PROFESSOR_ID	int	NO	PRI	<null>	auto_increment
2	PROFESSOR_NAME	varchar(30)	YES		<null>	
3	DEPARTMENT_ID	int	YES	MUL	<null>	

Projects: The Projects table will have an auto incremented integer which will be a primary key and stored as PROJECT_ID. The Project title is made UNIQUE to make sure it can't be repeated. PROFESSOR_ID is the foreign key that references the professor table. (figure 6)

	Field	Type	Null	Key	Default	Extra
1	PROJECT_ID	int	NO	PRI	<null>	auto_increment
2	PROJECT_TITLE	varchar(30)	YES	UNI	<null>	
3	PROFESSOR_ID	int	YES	MUL	<null>	

Project Preference: This table creates an auto incremented integer as primary key and has two foreign keys PROFESSOR_ID and STUDENT_ID along with a preference column that will hold the student's preference and it has zero or many relationships with Student, Allocation and Projects. (figure 7)

	Field	Type	Null	Key	Default	Extra
1	PREFERENCE_ID	int	NO	PRI	<null>	auto_increment
2	STUDENT_ID	int	YES	MUL	<null>	
3	PROJECTS_ID	int	YES	MUL	<null>	

Student: This table will have an auto incremented int as Student ID and will be the primary key, it will have the stream name so if it's the CD or DS stream, the STREAM_ID will be a foreign key of the STREAM table and the GPA field will hold the information as a float to reflect GPA, ie 4.0, 3.8 etc. The student table has a one to one relationship with the stream table. (figure 8)

	Field	Type	Null	Key	Default	Extra
1	STUDENT_ID	int	NO	PRI	<null>	auto_increment
2	STUDENT_NAME	varchar(30)	YES		<null>	
3	GPA	float	YES		<null>	
4	STREAM_ID	int	YES	MUL	<null>	

Stream: The Stream table will have an auto incremented ID as a primary key, it will hold the name of the stream DS or CS etc and will have a foreign key of DEPARTMENTID. It has a relationship with the department, student and stream projects detail table. (figure 9)

	Field	Type	Null	Key	Default	Extra
1	STREAM_ID	int	NO	PRI	<null>	auto_increment
2	STREAM_NAME	varchar(20)	YES		<null>	
3	DEPARTMENT_ID	int	YES	MUL	<null>	

ALLOCATION: has an auto incremented primary key used as an ID and has a foreign key to the project preference table using that ID and is saved as ALL_PREFERENCE_ID. It has many to one and only one relationship with the project_preferences table. (figure 10)

	Field	Type	Null	Key	Default	Extra
1	ALLOCATION_ID	int	NO	PRI	<null>	auto_increment
2	ALL_PREFERENCE_ID	int	YES	MUL	<null>	

Normalized

The goal for the design was to have it in Boyce-Codd normal form (BCNF) is the strongest normalization of a database without having to deal with extra anomalies that are handled in 4NF and 5NF (Harrington, 2009). I will now look at each normal form and how the database achieves this.

First Normal Form (1NF)

First normal form relates to no repeating groups, composite attributes. The attributes must all contain atomic values (Laurini and Thompson, 1992). In all the table columns they all contain the same type of information due to the pre-set parameters an example would be the ID being an auto generated int or the GPA only accepting float inputs.

Each of the columns have a unique name to avoid confusion during data retrieval and they're no duplicating rows and the order in which the data is stored in the table does not matter. The information provided already had the starter table in a good state that was easily extendable and had properties of first normal form.

Second Normal Form (2NF)

For a table to be in 2NF it must first pass 1NF and there must be no partial dependencies. This was the first step in reducing the redundant data in tables by extracting the information and placing it into a new table or creating a new table for it and establishing a relationship between those new tables (Celko, 2015). All the tables have no partial dependences and all PK and FK are correctly created or linked to a different table.

An example of this would have been creating the professor table and department table as it would be able to host information that would have been partially dependent on each other but are now linked by FK and still linked to the original project and project preferences tables.

Third Normal Form (3NF)

For a table to be in 3NF it must fulfil the constraints of 2NF and have no transitive dependency meaning that all non-primary or foreign key fields are dependent on the primary key, this is known to be where most anomalies are removed from a database (Teorey, Lightstone, Nadeau and Jagadish, 2011). This would be beneficial for when the School needs to add new students or when they need to add or remove a professor, instead of updating each individual student or lecturer because they're stored in one place, for example if a student was removed on a database it would be reflected across all the tables that use student_ID or name in their tables, Instead of having to remove them all individually.

For this database it involved splitting the tables some more and implementing more foreign keys, and really focusing on what information each table needed. Instead of having the professor ID in projects, project preferences and the student stream it is now a foreign key that each table can access.

Boyce-Codd Normal Form (BCNF)

It is necessary to normalize a database to the highest possible level for the best performance. In most cases 3NF is a good design objective but if BCNF is available it is best to seek the upmost level of normalization (Harrington, 2009). For tables to be considered BCNF it must first be 3NF and for each functional dependency the prime attribute is made into a Super key.

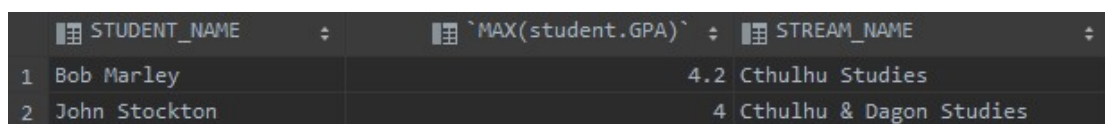
BCNF is free of most anomalies and it enforces the use of keys for the ease of functional dependencies and to limit the most redundancy in the data that it can, and it is the strongest normalization variation this database can have as there is no non-prime attributes that can identify as a prime attribute and all the prime attributes are defined or part of a primary key.

4. Database Views

Views are an important part of a database system; they provide benefits for the users in many areas. One of those areas is to hide the complexity of the database, so the user can only view the data they need and do not need to search through multiple tables to view the desired information, that can also being used as a security measure in times of general data protection regulation (GDPR) there is a high importance that users only see the information they need to see, so creating a specific view that hides all other information is key.

View 1:

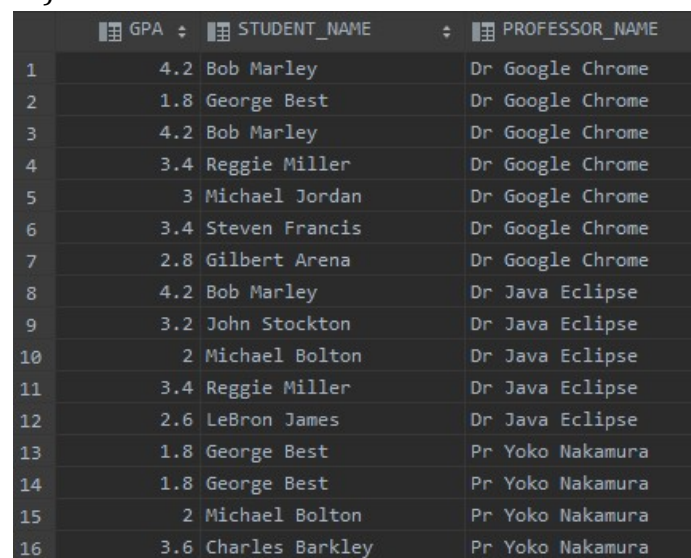
This view will be good when awarding the top performers of each stream as it outputs the name and the highest GPA from each stream and also have the stream name and will order the MAX column in descending fashion, so the top student of both streams is shown first. (figure 11)



	STUDENT_NAME	MAX(student.GPA)	STREAM_NAME
1	Bob Marley	4.2	Cthulhu Studies
2	John Stockton	4	Cthulhu & Dagon Studies

View 2:

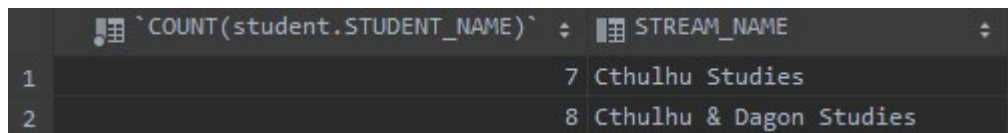
This view shows all the possible professors for each student based on their project and stream choices. This can be useful for professors looking at who their potential students are and what are their capabilities based on the GPA. It can also be useful for students when they want to check which professor will oversee their project. (figure 12)



	GPA	STUDENT_NAME	PROFESSOR_NAME
1	4.2	Bob Marley	Dr Google Chrome
2	1.8	George Best	Dr Google Chrome
3	4.2	Bob Marley	Dr Google Chrome
4	3.4	Reggie Miller	Dr Google Chrome
5	3	Michael Jordan	Dr Google Chrome
6	3.4	Steven Francis	Dr Google Chrome
7	2.8	Gilbert Arena	Dr Google Chrome
8	4.2	Bob Marley	Dr Java Eclipse
9	3.2	John Stockton	Dr Java Eclipse
10	2	Michael Bolton	Dr Java Eclipse
11	3.4	Reggie Miller	Dr Java Eclipse
12	2.6	LeBron James	Dr Java Eclipse
13	1.8	George Best	Pr Yoko Nakamura
14	1.8	George Best	Pr Yoko Nakamura
15	2	Michael Bolton	Pr Yoko Nakamura
16	3.6	Charles Barkley	Pr Yoko Nakamura

View 3:

This view will let the school see how many students are enrolled in each stream and will work even if more students are added and will also work if new streams are added. This will be good for when there will be up to 200 students applying for the course in the future and they'll need to manage the capacity of each stream. (figure 13)

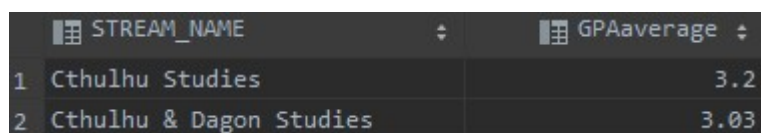


The screenshot shows a database query result with two columns: a count of students and the stream name. The query is `COUNT(student.STUDENT_NAME)`` and the stream names are `STREAM_NAME`. The results are as follows:

1	7	Cthulhu Studies
2	8	Cthulhu & Dagon Studies

View 4:

The fourth view helps the university know what the strongest stream is in terms of GPA, by averaging the GPA of each student from each student and outputting the results. This is also helpful for prospective students as they will be able to get an idea of what course is more difficult and what course they have possibility of getting the best grade from. (figure 14)

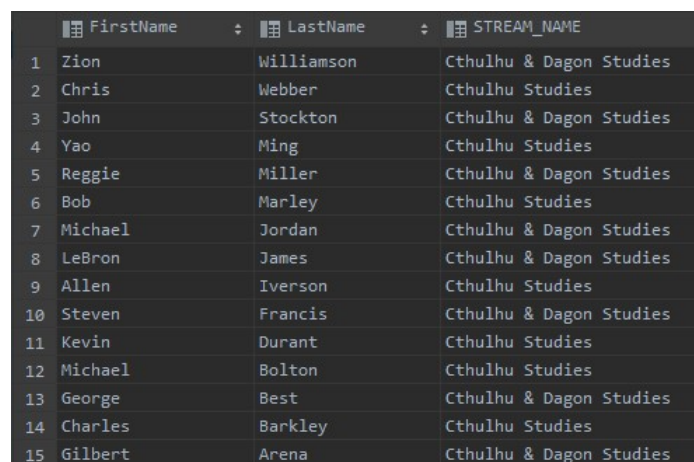


The screenshot shows a database query result with two columns: the stream name and the average GPA. The columns are `STREAM_NAME` and `GPAaverage`. The results are as follows:

1	Cthulhu Studies	3.2
2	Cthulhu & Dagon Studies	3.03

View 5:

This view outputs the names of the students and the Stream name but instead of outputting the full name it will parse the text with a space and save it as substring first name and substring last name. This could be helpful when the university is creating emails and they only want to attach the first name for the "Hi <Insert Name>" and if it's just the first name it may feel more personal. (figure 15)



The screenshot shows a database query result with three columns: first name, last name, and stream name. The columns are `FirstName`, `LastName`, and `STREAM_NAME`. The results are as follows:

	FirstName	LastName	STREAM_NAME
1	Zion	Williamson	Cthulhu & Dagon Studies
2	Chris	Webber	Cthulhu Studies
3	John	Stockton	Cthulhu & Dagon Studies
4	Yao	Ming	Cthulhu Studies
5	Reggie	Miller	Cthulhu & Dagon Studies
6	Bob	Marley	Cthulhu Studies
7	Michael	Jordan	Cthulhu & Dagon Studies
8	LeBron	James	Cthulhu & Dagon Studies
9	Allen	Iverson	Cthulhu Studies
10	Steven	Francis	Cthulhu & Dagon Studies
11	Kevin	Durant	Cthulhu Studies
12	Michael	Bolton	Cthulhu Studies
13	George	Best	Cthulhu & Dagon Studies
14	Charles	Barkley	Cthulhu Studies
15	Gilbert	Arena	Cthulhu & Dagon Studies

View 6:

The final view that has been created is similar to view 5, but this time it parses the professor names as they all have titles or either professor, this is useful for when they don't want to send full titled emails or for students to be able to contact the head of the class in a better manner understanding their full name. It also helps for when the system is telling each student what project they got the professor can chose if they want their full title, or just title and surname as what the student will see.

(figure 16)

	Title	FirstNAme	lastName	DEPARTMENT_NAME
1	Pr	Yoko	Nakamura	Teachers of Cthulhu Studies
2	Pr	Yoko	Nakamura	Teachers of Cthulhu & Dagon Studies
3	Dr	Hugh	Larsson	Teachers of Cthulhu Studies
4	Dr	Hugh	Larsson	Teachers of Cthulhu & Dagon Studies
5	Dr	Java	Eclipse	Teachers of Cthulhu Studies
6	Dr	Java	Eclipse	Teachers of Cthulhu & Dagon Studies
7	Dr	Google	Chrome	Teachers of Cthulhu Studies
8	Dr	Google	Chrome	Teachers of Cthulhu & Dagon Studies
9	Pr	Stephen	Chambers	Teachers of Cthulhu Studies
10	Pr	Stephen	Chambers	Teachers of Cthulhu & Dagon Studies
11	Pr	Martin	Adams	Teachers of Cthulhu Studies
12	Pr	Martin	Adams	Teachers of Cthulhu & Dagon Studies

5. Procedural Elements

Triggers are a pre-stored procedure that will run when an event occurs on the database, these can happen when a user tries to update, insert, delete statements in a view or a table and they will automatically operate when the tables affected are being called on. This helps enforce data integrity and provide some security to transactions and logging of events so the database actions can be traced back (Oracle PL/SQL Trigger Tutorial, 2020).

Trigger 1:

The first trigger designed makes sure that the only streams that can be entered into the STREAM table are “Cthulhu Studies” and “Cthulhu & Dagon Studies”, also the department ID related to these two streams have to link with a foreign key for the department table, It will throw an Error condition if the information is attempted to be inputted incorrectly. (figure 17)

```
school_cthulhu> INSERT INTO STREAM VALUES (2,'Apple & Pineapple Studies',2)
[2020-05-04 12:40:45] [02000][1643] Unhandled user-defined not found condition
```

Trigger 2:

The second trigger assigns a GPA to a new student if they have no GPA currently set this would be useful for a transfer student or for students that are entering the first year of the program it will set every null GPA to a 3.0, meaning they have a good starting grade and it can be updated throughout the year. Below shows me creating a student with a null GPA and then a screenshot looking for the student to see if the GPA has auto populated to a 3.0. (figure 18)

```
INSERT INTO STUDENT VALUES (18, 'Gilbert a', null, 2);
SELECT * FROM STUDENT;
```

(figure 19)

STUDENT_ID	STUDENT_NAME	GPA	STREAM_ID
18	Gilbert a	3	2

The same was used in the professors table, if a new professor was added and they haven't decided on their name title yet, it would return Awaiting Professor details.

PROFESSOR_ID	PROFESSOR_NAME	DEPARTMENT_ID
8	18 Awaiting Professor details	2

Trigger 3:

A trigger to limit the amount of projects a professor can be in charge of and if the amount of projects are greater than 5 then the SQL will output a signal state with the message notifying that the professors max project oversight limit has been reached and the professor will have to be changed for that project. (figure 20)

```
school_cthulhu> INSERT INTO PROJECTS VALUES (271, 'kjakdsa', 6)
[2020-05-05 03:47:51] [12345][1644] A professor can only concentrate on atmost five projects
[2020-05-05 03:47:51] [HY000][1644] A professor can only concentrate on atmost five projects
```


6. Example Queries: Your Database in Action

Queries in a database help you achieve a complex task or complex view of the information in a database, a database isn't well-intentioned unless information can not be gathered from it. A query is one way of extracting the information from the database (Chamberlin, 2020).

The queries I created I believe will help the school gather information they will need everyday but in a shrunken format and will only show the information that is needed and with the queries they can easily be edited to include more information if need be.

Query 1:

The below is a result of a query that lists all students for there selected streams and ranks the students based on their GPA in descending order. This will be effective for the school and professors being able to see the top performers of each stream and what students may need help and it is a query that could be used throughout a school year.

(figure 21)

	STUDENT_ID	STUDENT_NAME	GPA	STREAM_NAME
1	1	Bob Marley	4.2	Cthulhu Studies
2	14	Yao Ming	3.8	Cthulhu Studies
3	10	Charles Barkley	3.6	Cthulhu Studies
4	6	Chris Webber	3.2	Cthulhu Studies
5	12	Lenovo Vision	3	Cthulhu Studies
6	7	Allen Iverson	2.6	Cthulhu Studies
7	4	Michael Bolton	2	Cthulhu Studies

(figure 22)

	STUDENT_ID	STUDENT_NAME	GPA	STREAM_NAME
1	8	Zion Williamson	4	Cthulhu & Dagon Studies
2	9	Reggie Miller	3.4	Cthulhu & Dagon Studies
3	13	Steven Francis	3.4	Cthulhu & Dagon Studies
4	2	John Stockton	3.2	Cthulhu & Dagon Studies
5	3	Michael Jordan	3	Cthulhu & Dagon Studies
6	15	Dell Latitude	2.8	Cthulhu & Dagon Studies
7	11	Steven Plus	2.6	Cthulhu & Dagon Studies
8	5	George Best	1.8	Cthulhu & Dagon Studies

Query 2:

The below view returns the students from each stream along with their student name and ID, this is relevant when identifying what class each student is taking or If the class is taking a roll and need to know all the relevant details for each student and it will help identify which is the most popular student stream. (figure 23)

	STUDENT_ID	STUDENT_NAME	STREAM_NAME
1	1	Bob Marley	Cthulhu Studies
2	4	Michael Bolton	Cthulhu Studies
3	6	Chris Webber	Cthulhu Studies
4	7	Allen Iverson	Cthulhu Studies
5	10	Charles Barkley	Cthulhu Studies
6	12	Kevin Durant	Cthulhu Studies
7	14	Yao Ming	Cthulhu Studies

	STUDENT_ID	STUDENT_NAME	STREAM_NAME
1	2	John Stockton	Cthulhu & Dagon Studies
2	3	Michael Jordan	Cthulhu & Dagon Studies
3	5	George Best	Cthulhu & Dagon Studies
4	8	Zion Williamson	Cthulhu & Dagon Studies
5	9	Reggie Miller	Cthulhu & Dagon Studies
6	11	LeBron James	Cthulhu & Dagon Studies
7	13	Steven Francis	Cthulhu & Dagon Studies
8	15	Gilbert Arena	Cthulhu & Dagon Studies

Query 3:

This inner join query will help the school see the project titles that each student are interested in completing, It will have the student_ID, Student_Name and the title and it will provide each preference from 1 upwards and will also show the GPA for each of those projects to help with allocating preferences as they could rank each preference by the GPA. (figure 24)

	STUDENT_ID	STUDENT_NAME	GPA	PROJECT_TITLE
1	1	Bob Marley	4.2	Difference between CS & DS
2	1	Bob Marley	4.2	Cthulhu skiing apps
3	1	Bob Marley	4.2	Gamestation
4	1	Bob Marley	4.2	Toyota Corolla
5	2	John Stockton	3.2	Alexion Pharma
6	2	John Stockton	3.2	Cthulhu skiing apps
7	2	John Stockton	3.2	Asahi
8	3	Michael Jordan	3	State Street
9	4	Michael Bolton	2	Playstation
10	4	Michael Bolton	2	Moretti

Query 4:

This query counts how many student's have selected a specific project title as their preferred choice as a project, this will be useful for the school to identify which project type is most commonly chosen by the students so they can possibly create projects that are similar to the popular titles and cut the project titles that aren't popular in future. The query sorts the count column in descending order to make it easier to view the most popular topics.

	PROJECT_TITLE	PROJECT_ID	Popular_preference
1	State Street	24	3
2	Alexion Pharma	25	2
3	Asahi	18	2
4	Cthulhu skiing apps	2	2
5	Gamestation	12	2
6	Huawei	14	2
7	Moretti	17	2
8	Playstation	13	2
9	The Return	20	2
10	Toyota Corolla	23	2
11	What even is CS	6	2
12	Where is Miskatonic	8	2
13	Dagan Marathon app	3	1
14	Difference between CS & DS	1	1
15	Eldritch Fitness monitor	4	1

Query 5:

I created this query for the school and professors are able to see all of the students, their steams, project preferences and what lecturer would be teaching them based on the stream and based on the project preference that each student has made so there is so repeated information because of the multiple preferences.

	STREAM_NAME	STUDENT_ID	STUDENT_NAME	PROJECT_TITLE	PROFESSOR_NAME
12	Cthulhu Studies	10	Charles Barkley	The Return	Pr Stephen Chambers
13	Cthulhu Studies	10	Charles Barkley	Huawei	Pr Yoko Nakamura
14	Cthulhu Studies	12	Kevin Durant	Moretti	Dr Hugh Larsson
15	Cthulhu Studies	14	Yao Ming	The Return	Pr Stephen Chambers
16	Cthulhu Studies	14	Yao Ming	Alexion Pharma	Dr Hugh Larsson
17	Cthulhu & Dagon Studies	2	John Stockton	Alexion Pharma	Dr Hugh Larsson
18	Cthulhu & Dagon Studies	2	John Stockton	Cthulhu skiing apps	Dr Java Eclipse
19	Cthulhu & Dagon Studies	2	John Stockton	Asahi	Pr Martin Adams
20	Cthulhu & Dagon Studies	3	Michael Jordan	State Street	Dr Google Chrome
21	Cthulhu & Dagon Studies	5	George Best	Dagan Marathon app	Pr Yoko Nakamura
22	Cthulhu & Dagon Studies	5	George Best	Eldritch Fitness monitor	Dr Google Chrome
23	Cthulhu & Dagon Studies	5	George Best	Water resistant laptops	Pr Yoko Nakamura
24	Cthulhu & Dagon Studies	8	Zion Williamson	What even is DS	Pr Martin Adams
25	Cthulhu & Dagon Studies	8	Zion Williamson	One plus	Dr Hugh Larsson
26	Cthulhu & Dagon Studies	9	Reggie Miller	Gamestation	Dr Google Chrome

7. Conclusions

In creating this database I've tried to make it normalized, efficient, intuitive, and have it open to expansion. I created the queries, views and triggers based on the requirements given and for them to work in the future when the database grows. Despite this I did feel like I could have improved on the work in some areas and have more ideas for how the database can improve when it grows.

I would look at building the allocation system as well as the database as I would have a better understanding of the task, yes it would be a lot more time consuming but I would get to see a lot more rounded system and I would be able to optimize the views, queries and triggers to best suit the database and allocation system and fulfill the white papers requirements to the best of my ability. This would have helped for the next step in the white paper in the measuring satisfaction section and would have built off the database and the preference matrix.

For when the University is increasing the amount of students it will be intaking, I would look at a way for a trigger, or function would load in a csv file of the student information and update the student tables and the corresponding tables based on the information provided.

I couldn't create a query or trigger that would rank the preferences of each student and determine which student would get the chosen project and which student would have to make do with their second choice option. I tried to implement this by having the highest-ranking GPA student always getting their first preference. So, this is something I would have linked to implement and would make the database a lot easier. I didn't correctly align students to the correct streams and what professors would be overseeing each group project.

It's very common know for universities to have a ranking system and especially with the increase of potential students in the following year, it would be a good idea to add a ranking table for each department they study so it will be easier for the university to review the current programs and what they need to do to make them more successful and what prospective students might base their stream decisions off.

There was limits to what I could do in terms of triggers because the tables were in

BCNF so there were limited opportunities as there weren't many columns on each table and even those were primary key or unique. I attempted a trigger query that would put the information into the allocation table but didn't work well. I considered creating a table that would output the time and date of when each table was updated and who by, but the table wouldn't be in BCNF>

In future planned implementations a GUI will be used, so I believe it would be optimal for the database column names match the GUI inputs for students and also for the staff member it will be easier for them to designate which information links to the GUI and the table.

A timetable table holding each of the streams timetables and list the times each professor is scheduled to be teaching so the free time can be organized to set-up meetings with students that will be overseen by the specific lecturer.

I'm happy with what I've accomplished in this project and of course have highlighted the problems that I haven't been able to solve. The database can currently serve a purpose based on the white paper and will be able to hold up even with the influx of students. For the database to really succeed Miskatonic University will need to build on the database and implement more features and tables to store more information so more queries can be run, and the information will be stored on a safe space and the access can be restricted.

References

- Celko, J., 2015. *Joe Celko's SQL For Smarties, 5Th Edition*. 5th ed. Morgan Kaufmann Publishers.
- Chamberlin, D., 2020. *SQL*. [online] Researcher.watson.ibm.com. Available at: <<https://researcher.watson.ibm.com/researcher/files/us-dchamber/SQL-encyclopedia-entry.pdf>> [Accessed 3 May 2020].
- Smartdraw.com. 2020. *Entity Relationship Diagram (ERD) - What Is An ER Diagram?*. [online] Available at: <<https://www.smartdraw.com/entity-relationship-diagram/>> [Accessed 27 April 2020].
- Harrington, J., 2009. *Relational Database Design*. 3rd ed. Amsterdam: Morgan Kaufmann.
- Knauff, B., 2004. *Design Your Own Database: Concept To Implementation*. [online] Dartmouth.edu. Available at: <<https://www.dartmouth.edu/~bknauff/dwebd/2004-02/DB-intro.pdf>> [Accessed 23 April 2020].
- Laurini, R. and Thompson, D., 1992. *Fundamentals Of Spatial Information Systems*. London: Academic Press.
- Li, L., 2019. *Database Normalization Explained*. [online] Medium. Available at: <<https://towardsdatascience.com/database-normalization-explained-53e60a494495>> [Accessed 1 May 2020].
- Guru99.com. 2020. *Oracle PL/SQL Trigger Tutorial: Instead Of, Compound [Example]*. [online] Available at: <<https://www.guru99.com/triggers-pl-sql.html>> [Accessed 3 May 2020].
- Tutorialspoint.com. 2020. *PL/SQL - Triggers - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/plsql/plsql_triggers.htm> [Accessed 4 May 2020].
- GeeksforGeeks. 2020. *SQL Trigger / Student Database - Geeksforgeeks*. [online] Available at: <<https://www.geeksforgeeks.org/sql-trigger-student-database/>> [Accessed 3 May 2020].

Teorey, T., Lightstone, S., Nadeau, T. and Jagadish, H., 2011. *Database Modeling And Design, 5Th Edition*. 5th ed. Morgan Kaufmann.

Jcsites.juniata.edu. 2020. *Three Level Database Architecture*. [online] Available at: <<http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm>> [Accessed 28 April 2020].

Watt, A., 2020. *Chapter 8 The Entity Relationship Data Model*. [online] Opentextbc.ca. Available at: <<https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/>> [Accessed 1 May 2020].

Stack Overflow. 2020. *Why Do You Create A View in A Database?* [online] Available at: <<https://stackoverflow.com/questions/1278521/why-do-you-create-a-view-in-a-database>> [Accessed 27 April 2020].