Contents

[1. Abstract 1](#_Toc54271566)

[2. Overview of Database 1](#_Toc54271567)

[3. Illustration of the design 2](#_Toc54271568)

[4. Normalisation 3](#_Toc54271569)

[5. Data Storage Solution 6](#_Toc54271570)

[6. Scripts to create Data Storage 7](#_Toc54271571)

[7. Main Usage and Scripts for typical use cases 13](#_Toc54271572)

# **Abstract**

In this report, MySQL is used to develop and implement a University Database System. Exploring typical use cases, and the overall design of the database system. The report also outlines various MySQL queries such as SELECT, ALTER, DELETE, JOIN to tackle the requirements from the project. The implementation of database was successful and is extremely powerful, secure, and easy to use.

# **Overview of Database**

The database will be used to store information relating to the University.

* **Introduction**

University is a key institutions in the society. A good university will persuade people to join them. As a result of this, there is a need for a system that enables university management in making effective and efficient decision for students.

* **Main Uses of the Database**

A university database management system is a record management which enables the user to access necessary data at any place and any time through internet. A student can receive important information and notifications in the university like the timetable, workshops, fee payment, exam registration. The students can check their attendance and marks for every unit. The management staff can also make changes in marks, or some necessary changes in case of any mistake that are made. The tutors and teachers can also retrieve information about the students records and allows them to make changes in student academic or personal details.

# **Illustration of the design**

The following figure is an Entity-Relationship Diagram showing the relationships between each table.

Diagram

Description automatically generated

# **Normalisation**

The following figure is the normalisation for each entities drawn in ER diagram. This will include the first normal form, second normal form, and third normal form.

**Subject Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Subject\_id | Subject\_name | Prerequisite | Credit |

* This table doesn’t need to be normalized.

**Scholarship Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Scholarship\_id | Scholarship\_value | Scholarship\_condition | Scholarship\_desc |

* This table doesn’t need to be normalized.

**Provider Table:**

|  |  |  |
| --- | --- | --- |
| Provider\_id | Provider\_name | Company\_no |

* This table doesn’t need to be normalized.

**Student Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student\_id | Scholarship\_id | Firstname | Lastname | Gender | Date\_of\_birth |

* This table is in First Normal Form (1NF).

**Normalise to Second Normal Form (2NF)** **:**

|  |  |
| --- | --- |
| Student\_id | Scholarship\_id |

student\_id 🡪 scholarship\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_id | Firstname | Lastname | Gender | Date\_of\_birth |

student\_id 🡪 firstname, lastname, gender, date\_of\_birth

* In this 2NF, there are no more partial dependencies that exist in the table. It has met all the requirements for 2NF.

**From 2NF above, we normalise to Third Normal Form (3NF) :**

Scholarships

|  |  |  |  |
| --- | --- | --- | --- |
| Scholarship\_id | Scholarship\_value | Scholarship\_condition | Scholarship\_desc |

Scholarship\_id 🡪 scholarship\_value, scholarship\_condition, scholarship\_desc

* The purpose in this 3NF is to check for transitive dependencies and eliminate them if found from the 2NF table. This table are now in 3NF.

**Result Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Student\_id | Subject\_id | Attempt | Score |

* This table is in First Normal Form (1NF).

**Normalise to Second Normal Form (2NF) :**

|  |  |
| --- | --- |
| Student\_id | Subject\_id |

Student\_id 🡪 subject\_id

|  |  |
| --- | --- |
| Student\_id | Attempt |

Student\_id 🡪 attempt

|  |  |
| --- | --- |
| Student\_id | Score |

Student\_id 🡪 score

* In this 2NF, there are no more partial dependencies that exist in the table. It has met all the requirements for 2NF.

**From 2NF above, we normalise to Third Normal Form (3NF) :**

|  |  |  |  |
| --- | --- | --- | --- |
| Subject\_id | Subject\_name | Prerequisite | Credit |

Subject\_id 🡪 subject\_name, prerequisite, credit

* The purpose in this 3NF is to check for transitive dependencies and eliminate them if found from the 2NF table. This table are now in 3NF.

**Enrolment Table :**

|  |  |  |
| --- | --- | --- |
| Enrolment\_id | Student\_id | Description |

* This table is in First Normal Form (1NF).

**Normalise to Second Normal Form (2NF) :**

|  |  |
| --- | --- |
| Enrolment\_id | Student\_id |

Enrolment\_id 🡪 student\_id

|  |  |
| --- | --- |
| Enrolment\_id | Description |

Enrolment\_id 🡪 description

* In this 2NF, there are no more partial dependencies that exist in the table. It has met all the requirements for 2NF.

**From 2NF above, we normalise to Third Normal Form (3NF) :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student\_id | Scholarship\_id | Firstname | Lastname | Gender | Date\_of\_birth |

Student\_id 🡪 scholarship\_id, firstname, lastname, gender, date\_of\_birth

* The purpose in this 3NF is to check for transitive dependencies and eliminate them if found from the 2NF table. This table are now in 3NF.

**Scholarship\_Provider Table :**

|  |  |  |
| --- | --- | --- |
| Scholarship\_id | Provider\_id | Note |

* This table is in First Normal Form (1NF).

**Normalise to Second Normal Form (2NF) :**

|  |  |
| --- | --- |
| Scholarship\_id | Provider\_id |

Scholarship\_id 🡪 provier\_id

|  |  |
| --- | --- |
| Scholarship\_id | Note |

Scholarship\_id 🡪 note

* In this 2NF, there are no more partial dependencies that exist in the table. It has met all the requirements for 2NF.

**From 2NF above, we normalise to Third Normal Form (3NF) :**

|  |  |  |
| --- | --- | --- |
| Provider\_id | Provider\_name | Company\_no |

Provider\_id 🡪 provider\_name, company\_no

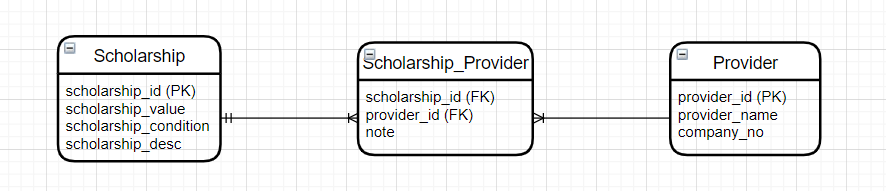
* The purpose in this 3NF is to check for transitive dependencies and eliminate them if found from the 2NF table. This table are now in 3NF.

# **Data Storage Solution**

The following table outlines the explanation of each strong entities in the database:

|  |  |
| --- | --- |
| Entity name | Justification |
| Student | Storing student personal details, enrolment, and scholarship. |
| Subject | This table stores the name and ID of a subject, the prerequisite so that we know if a student could enroll in a subject or not, and also the credit worth in each subject. |
| Result | This table store grades of students in every subject they enrolled in and the number of attempt they took. |
| Scholarship | Scholarship reward to students with certain condition. The details of the scholarship such as value and description also exist. |
| Enrolment | Enroll for academic courses and monitor the progress of students through the semester. |
| Class | Class id for identification, 1 subject can have many classes. |
| Provider | This table stores provider of the scholarship which raising fund for the students. One provider can grant multiple scholarship for students. |

The following figure outlines many to many relationship:



For many to many relationship, there is a weak entity in between 2 strong entities, so it becomes 2 entities of one to many relationship. For example, in this project, Scholarship\_Provider is a weak entity. A scholarship can have many scholarship provider and scholarship are provided by many company.

# **Scripts to create Data Storage**

**\*\*\*\*\*\*\*\*\*\* MySQL CREATE TABLE STATEMENT \*\*\*\*\*\*\*\*\*\***

CREATE TABLE Scholarship (

scholarshipID INT UNSIGNED,

scholarship\_value INT(10) NOT NULL,

scholarship\_condition TEXT NOT NULL,

scholarship\_desc TEXT NOT NULL,

PRIMARY KEY(scholarshipID)

);

CREATE TABLE Student (

studentID INT UNSIGNED NOT NULL,

scholarshipID INT UNSIGNED,

firstname VARCHAR(35) NOT NULL,

lastname VARCHAR(35) NOT NULL,

gender CHAR(1) CHECK (gender IN ('M','F','U')),

date\_of\_birth DATE NOT NULL,

PRIMARY KEY(studentID)

);

CREATE TABLE Subject (

subjectID CHAR(4) NOT NULL UNIQUE,

subject\_name VARCHAR(40) NOT NULL,

prerequisite VARCHAR(40) DEFAULT NULL,

credit INT(5) NOT NULL,

PRIMARY KEY(subjectID)

);

CREATE TABLE Result (

studentID INT UNSIGNED NOT NULL,

subjectID CHAR(4) NOT NULL,

score INT(5) NOT NULL,

attempt INT(5) NOT NULL

);

CREATE TABLE Enrolment (

enrolmentID INT UNSIGNED NOT NULL UNIQUE,

studentID INT UNSIGNED NOT NULL,

description TEXT DEFAULT NULL,

PRIMARY KEY(enrolmentID)

);

CREATE TABLE Class (

classID INT UNSIGNED NOT NULL UNIQUE,

subjectID CHAR(4) NOT NULL,

note TEXT DEFAULT NULL,

PRIMARY KEY(classID)

);

CREATE TABLE Provider (

providerID CHAR(5) NOT NULL,

provider\_name VARCHAR(40) NOT NULL,

company\_no INT(5) NOT NULL,

PRIMARY KEY(providerID)

);

CREATE TABLE Scholarship\_Provider (

scholarshipID INT UNSIGNED,

providerID CHAR(5) NOT NULL,

note TEXT DEFAULT NULL

);

**\*\*\*\*\*\*\*\*\*\* FOREIGN KEY CONSTRAINTS \*\*\*\*\*\*\*\*\*\***

ALTER TABLE Student

ADD CONSTRAINT fk\_student\_scholarship

FOREIGN KEY(scholarshipID) REFERENCES Scholarship(scholarshipID);

ALTER TABLE Result

ADD CONSTRAINT fk\_result\_student

FOREIGN KEY(studentID) REFERENCES Student(studentID);

ALTER TABLE Result

ADD CONSTRAINT fk\_result\_subject

FOREIGN KEY(subjectID) REFERENCES Subject(subjectID);

ALTER TABLE Enrolment

ADD CONSTRAINT fk\_enrolment\_student

FOREIGN KEY(studentID) REFERENCES Student(studentID);

ALTER TABLE Class

ADD CONSTRAINT fk\_class\_subject

FOREIGN KEY(subjectID) REFERENCES Subject(subjectID);

ALTER TABLE Scholarship\_Provider

ADD CONSTRAINT fk\_scholarship\_provider

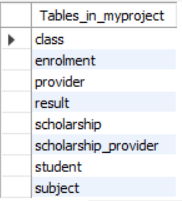
FOREIGN KEY(scholarshipID) REFERENCES Scholarship(scholarshipID);

ALTER TABLE Scholarship\_Provider

ADD CONSTRAINT fk\_provider\_scholarship

FOREIGN KEY(providerID) REFERENCES Provider(providerID);

Show tables;



**\*\*\*\*\*\*\*\*\*\* MySQL INSERT STATEMENT \*\*\*\*\*\*\*\*\*\***

INSERT INTO Subject (subjectID, subject\_name, credit)

VALUES ('A101', 'Data Management', 15);

INSERT INTO Subject (subjectID, subject\_name, credit)

VALUES ('B202', 'Statistics', 15);

INSERT INTO Subject (subjectID, subject\_name, credit)

VALUES ('C303', 'Data Visualisation', 20);

INSERT INTO Subject (subjectID, subject\_name, credit)

VALUES ('D404', 'Artificial Intelligence', 30);

INSERT INTO Subject (subjectID, subject\_name, credit)

VALUES ('E505', 'Data Analytics', 20);

INSERT INTO Scholarship(scholarshipID, scholarship\_value, scholarship\_condition, scholarship\_desc)

VALUES(1111, 15000, 'No fail unit', 'Available');

INSERT INTO Scholarship(scholarshipID, scholarship\_value, scholarship\_condition, scholarship\_desc)

VALUES(2222, 20000, 'At least 2 HD', 'Available');

INSERT INTO Scholarship(scholarshipID, scholarship\_value, scholarship\_condition, scholarship\_desc)

VALUES(3333, 25000, 'At least 3 HD', 'Available');

INSERT INTO Scholarship(scholarshipID, scholarship\_value, scholarship\_condition, scholarship\_desc)

VALUES(4444, 26000, 'HD on every subject', 'Available');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1024, 2222, 'Scorlib', 'Lexrin', 'M', '2002-01-05');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1025, 1111, 'Raymond', 'Andilsim', 'M', '2002-01-28');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1026, 3333, 'Edward', 'Apriandy', 'M', '2001-04-11');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1027, 4444, 'Jackson', 'Timmer', 'M', '2001-05-09');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1028, 4444, 'Wilsen', 'Marchlen', 'M', '2001-08-9');

INSERT INTO Student(studentID, scholarshipID, firstname, lastname, gender, date\_of\_birth)

VALUES(1029, 2222, 'Angela', 'Chua', 'F', '2001-01-31');

INSERT INTO Student(studentID, firstname, lastname, gender, date\_of\_birth)

VALUES(1030, 'Putri', 'Fellany', 'F', '2002-01-05');

INSERT INTO Student(studentID, firstname, lastname, gender, date\_of\_birth)

VALUES(1031, 'Rudy', 'Santoso', 'M', '1970-05-24');

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1027, 'A101', 1, 75);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1027, 'B202', 3, 60);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1028, 'C303', 1, 90);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1028, 'D404', 2, 55);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1029, 'A101', 1, 80);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1024, 'A101', 1, 100);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1025, 'E505', 4, 30);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1025, 'B202', 3, 45);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1026, 'C303', 2, 70);

INSERT INTO Result(studentID, subjectID, attempt, score)

VALUES(1026, 'E505', 1, 95);

INSERT INTO Provider(providerID, provider\_name, company\_no)

VALUES('PT666', 'Microsoft', 116118);

INSERT INTO Provider(providerID, provider\_name, company\_no)

VALUES('TJ777', 'Government', 113115);

INSERT INTO Provider(providerID, provider\_name, company\_no)

VALUES('SA888', 'Swinburne', 114116);

INSERT INTO Provider(providerID, provider\_name, company\_no)

VALUES('CO999', 'Amazon', 115117);

INSERT INTO Scholarship\_Provider(scholarshipID, providerID, note)

VALUES(4444, 'CO999', 'Congratulations');

INSERT INTO Scholarship\_Provider(scholarshipID, providerID)

VALUES(2222, 'TJ777');

INSERT INTO Scholarship\_Provider(scholarshipID, providerID)

VALUES(3333, 'SA888');

# **Main Usage and Scripts for typical use cases**

* **Retrieve data**

*Show all information of student ORDER BY studentID :*

SELECT studentID AS 'Student ID', CONCAT(firstname, ' ', lastname) AS 'Student Name',

gender AS 'Gender', date\_of\_birth AS 'DOB', scholarshipID AS 'Scholarship ID'

FROM Student ORDER BY studentID;



*Show all details information of scholarships :*

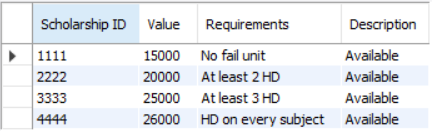
SELECT scholarshipID AS 'Scholarship ID',

scholarship\_value AS 'Value',

scholarship\_condition AS 'Requirements',

scholarship\_desc AS 'Description'

FROM Scholarship ORDER BY scholarshipID;



* **Update information**

*Change the lastname of a student :*

UPDATE Student SET lastname = 'Ng' WHERE studentID = '1027';

Before :



After :



*Change student score from Result table :*

UPDATE Result SET score = 40 WHERE studentID = 1025 AND subjectID = 'E505';

Before :



After :



* **SQL JOIN Queries**

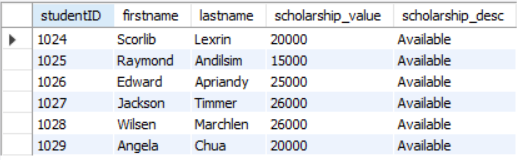
*The following command shows the details of scholarship for each students by joining 2 tables:*

SELECT s.studentID, s.firstname, s.lastname,

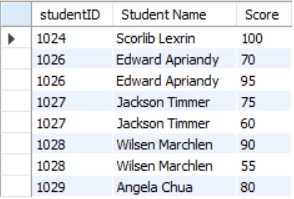
sc.scholarship\_value, sc.scholarship\_desc

FROM Student s NATURAL JOIN Scholarship sc

ORDER BY s.studentID;



*The following command shows every student that receive Score more than 50 :*

SELECT s.studentID, CONCAT(s.firstname, ' ' ,s.lastname)

AS 'Student Name', r.score AS 'Score'

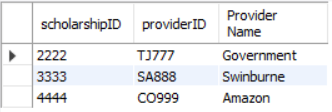
FROM Student s JOIN Result r

ON s.studentID = r.studentID

WHERE r.score > 50

ORDER BY s.studentID;

*The following command shows the provider details for each scholarship given :*

SELECT sp.scholarshipID, sp.providerID, p.provider\_name AS 'Provider Name'

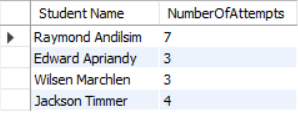
FROM Scholarship\_Provider sp LEFT JOIN Provider p

ON sp.providerID = p.providerID

ORDER BY scholarshipID;

*The following command returns the number of attempts student did on all subject :*

SELECT CONCAT(s.firstname,' ',s.lastname) AS 'Student Name',

SUM(r.attempt) AS 'NumberOfAttempts'

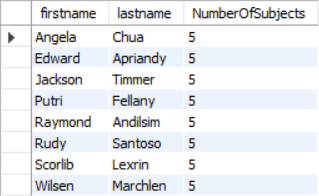
FROM Student s

INNER JOIN Result r

ON s.studentID = r.studentID

GROUP BY s.lastname HAVING COUNT(r.attempt)>1;

*The following command returns the number of subjects each student enrolled in :*

SELECT s.firstname, s.lastname, COUNT(sj.subjectID)

AS 'NumberOfSubjects'

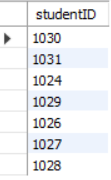
FROM Student s

NATURAL JOIN Subject sj

GROUP BY s.firstname;

* **Checking non-existing data in a table**

*The following commands returns the student id of all students that pass every unit(score more than 50). Studentid 1025 is missing because the student have score less than 50 so it means he/she didn’t pass the unit :*

SELECT studentID FROM Student s

WHERE NOT EXISTS(

SELECT \* FROM Result

WHERE Score <= 50 AND studentID = s.studentID );

* **Create VIEW**

*A view is a virtual table created by query by joining one or more tables. The following commands creates a VIEW called No\_Scholarship that contains information about students that didn’t receive any scholarship.*

CREATE VIEW No\_Scholarship AS

SELECT studentID AS 'Student ID',

CONCAT(firstname, ' ' ,lastname) AS 'Student Name',

scholarshipID FROM Student

WHERE scholarshipID IS NULL;

