|  |  |
| --- | --- |
|  |  |
| D*\_*Project |  |
|  |  |
|  | COS20015 |
|  | 102797812 David Long  103523487 Shaugato Paroi |

Table of Contents

[Abstract 3](#_Toc117429581)

[Overview of Database 3](#_Toc117429582)

[Introduction 3](#_Toc117429583)

[Main Uses of Database 3](#_Toc117429584)

[Illustration of the Design 3](#_Toc117429585)

[Normalisation 4](#_Toc117429586)

[Data Storage Solution 7](#_Toc117429587)

[Scripts For Data Storage 9](#_Toc117429588)

## Abstract

In this report, Google SQL is used to host, develop and implement the database for College Database System. Exploring the usual use cases and testing the overall design of the database system. Moreover, a software application called SQL Server Management Studio is integrated into Google SQL for easy configuration, management and administrating of all components within the SQL server and to test the full capability of the entities. We have integrated a data generator called Red Gate SQL Data Generator to generate at least a thousand dummy data. SQL Studio allows straight forward search queries with a GUI instead of generating a script to execute such as SELECT, ALTER, JOIN to fulfill the prerequisites from the outline. The implementation of the database was extremely successful, powerful, secure and very easy to use. Not only that, the implementation allows effortless migration between other databases.

## Overview of Database

The database is hosted on a cloud database which requires careful implementation and ensuring best practices are utilized. This database consists of 9 tables. There are one to one , one to many and many to many relationships. Our database has student, Enrolment, Grade,Exam, Subject, Course, Scholarship, Provider\_Scholarship, Provider tables

## Introduction

College is part of everyday life. And a good database is a very essential part of it. In our database we are trying to create a simple yet complex which can easily to implement in any kind’s of university scenario. The modification part of the

## Main Uses of Database

A college database management system is a record system which enables the record administrators to access necessary data at any place and any time through the internet. A student can receive important information and notifications in the university like courses, scholarships and exams.

## Illustration of the Design

Our database created using 10 tablets. These are orovide, provier , provide\_scholarship, svholarshio

Here is the UML diagram for our database which clearly demonstrate our plan:

Diagram, schematic

Description automatically generated

## Normalisation

Here we do the normalization of each table in our database

* Exam table

|  |  |  |  |
| --- | --- | --- | --- |
| Exam\_ID | Exam\_name | Exam\_requirements | Grade |

* + This is already in 3rd normalization form , because there is on partial dependency . So this table does not need to be changed
* Subject table

|  |  |  |
| --- | --- | --- |
| Subject\_Id | Course\_ID | Devision |

* + We can see that this table already in 3rd NF form. As there is no duplication in primary key. Also there is no functional dependency between columns except with the primary key
* Grade

|  |  |  |  |
| --- | --- | --- | --- |
| Student\_ID | Exam\_Id | Num\_Attempts | Score |

* + In this table we use a Composite Primary Key , which is consists of Student\_ID from student table and Exam\_ID from Exam Table. Here other non-primary key don’t depend on each other except depent on only composite primary key. There will be no repeated value in the composite primary key also. So, we can conclude that it’s already in 3rd normalization form
* Student table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student\_ID | FirstName | FamilyName | Gender | Date\_OF\_Birth | Phon\_No |

* + In this table We create a surrogate key called Student\_ID and make it as primary key. Here the non-primary key like FirstName,FamilyName,Gender,Date\_OF\_Birth , Phon\_No don’t depend on depend on each other , and the primarykeyStudent\_ID is unique . So it’s already meets the criteria of 1st Normalizasion, 2nd Normalization, 3rd Normaliztion.
* Enrolment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Enrolment\_ID | Student\_ID | Enrolment\_year | Finish\_year | Scholarship\_ID |

* + In this table surrogate key Enrolment\_ID works as primary Key. Student\_Id is foreign key referred from Student table , Scholarship\_ID is also a foreign key referred from Scholarship table. And other key are non- primary key. Here is no dublication in any rows in the table . So, it’s already in 1st NF. There is no partial dependency as all other colums except the primary key coloum only depend on only primary key coloum. So, we can also say there is no transitive dependency. So Enrolment is already meets 1st NF, 2nd NF and 3rd NF
* Course Table

|  |  |  |  |
| --- | --- | --- | --- |
| Couse\_ID | Course\_Name | Subject\_Id | Enrolment\_ID |

* + Here the Course\_ID is used as primary. The subject\_Id and Enrolment\_Id are both foreign key. And Course\_Name is a non-primary key. There is no repeated value in the rows of the table . And all the key except the primary key depends only on the primary key. So , we can conclude that there is no partial dependency or transitive dependency. So this table is already is in 3rd Normalisation form which is it’s final form
* Scholarship Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scholarship\_ID | Scholarship\_Percentage | Condition | Start\_Year | Finish\_Year |

* + In the scholarship table, Scholarship\_ID is the primary key. All other keys are non-primary key. These keys only depends on Scholarship\_ID. So , we can say that in this table there is no partial dependency and transitive dependency. So, it already fulfills 1st NF ,2nd NF , 3rd requirements criteria
* Provider(Scholarship)

|  |  |  |
| --- | --- | --- |
| Provider\_ID | Scholarshi\_ID | Description |

* + Here Provider\_ID comes from the provider table and Scholarship\_Id comes from Scholarship table . They are both foreign key. And with this we create a composite primary key and use on this table . And the remaining one is non-primary key . So there is no chance of partial and transitive dependency here. So we can conclude it’s already in it’s final form.
* Provider table

|  |  |  |
| --- | --- | --- |
| Provider\_ID | Provider\_Name | Organisation |

* + Here the primary key is Provider\_ID and others are non-primary key. So there is no partial dependency and transitive dependency. So we don’t need to do anything on this table.
* Subject\_Course table

|  |  |
| --- | --- |
| Course\_ID | Subject\_ID |

* + This table is consists of CourseID( referred from Course Table) and Subject\_ID(referred from Subject table) . These two used as foreign key and also primary key. So. It actually creates a composite primary key. So , it’s already in it’s final form.
* Enrolment\_Subject

|  |  |
| --- | --- |
| Enrolment\_ID | Subject\_Id |

* + This table only has two columns which are Enrolment\_ID(Foreign key referred from Enrolment table) and Subject\_ID(Foreign key referred from subject table). Using this two , we create a composite primary key. And there is no other key in this table. So it’s already in 3rd NF form

## Data Storage Solution

For our College Database solution, Cloud SQL is chosen for its fully-managed database that assists in setting up, maintenance, management and administration on Google cloud platform. They support SQL server which is proprietary software tool that executes the SQL statements.

The Cloud includes features for automated backups, data replication and disaster recovery to ensure high availability and resilience.

|  |  |  |
| --- | --- | --- |
| Table Entity Name | Entity Description | Relationships between  Table |
| Student | Stores information about student ID ,First Name ,family Name,Gender Date of Birth, Phon No. Student\_ID is the primary key here. We can get a student’sfull details from here. | * One to many relationship with Enrolment(one student can enroll into many courses in enrolment table) * One to many relationship with Grade( one student can have different gread in different exam) |
| Enrolment | Stores information about Enrolment\_ID, student\_ID,Enrolment\_year,  Finish\_Year,Course\_ID.  We can get a student’s study history from   * *Enrolment\_ID is the primary key here* * *Student\_ID is the foreign key here* * *Course\_ID is also the foreign key* | * Many to one relationship with Scholarship( one scholarship can be given to multiple student in enrolment and each student must have at least one scholarship ) * Many to one relationship with student ( multiple enrolment can be happened by one student) * Many to one relationship with course( a student can enrol in multiple course) * One to many relationship with Enrolment\_Subject |
| Exam | Stores information about the exam name , requirement to take the exam, achieved grade from the exam, which subject the exam relates to   * Exam\_ID is the primary key * Subject\_ID is the foreign key | * One to one relationship with grade * Many to one relation with subject(multipul exams can be taken for one subject) |
| Subject | This only shows us subject name with its unique ID.   * Subject\_ID is the primary key | * One to many relationship with exam |
| Course | Stores information about a unique course alongside with it’s ID   * Course\_ID is the primary key here | * Many to one relationship with Enrolment |
| Subject\_Course | It’s a weak entity. As there is a many to many relationship between subject and course (a subject can be seen on multiple course and different courses have multiple same subject).And in order to break many – to -many relationship , this weak entity is created.   * Course\_ID and Subject\_Id both act as a composite primary key |  |
| Enrolment\_Course | It’s a weak entity. As there is a many to many relationship between Enrolment and course( a course can be enrolled multiple times by different students and a student can enroll different courses at the same time).And in order to break many – to -many relationship , this weak entity is created.   * Enrolment\_ID and Course\_ID both act as a composite primary key here |  |
| Course | Stores information about course along with it’s unique ID   * Course\_ID is the primary key here | * One to many relationship with Enrolment ( one course can be enrolled multiple times by different students) * Many to many relationship with subject |
| Scholarship | Stores information about scholarship percentage , condition to hold scholarship, scholarship start and finish yead.   1. Scholarship\_Id is the primary key | * One to many relationship with enrolment( one scholarship can be given to multiple students) * Many to many relationship with provider |
| Provider\_Scholarship | It’s a weak entity. As there is a many to many relationship between Provider and Scholarship(same scholarship can be provided by multiple providers and one provider can provide multiple scholarship).And in order to break many – to -many relationship, this weak entity is created.   * The Provider\_Id and Scholarship\_ID both asct as a composite primary key here |  |
| Provider | It gives us information about provider name and organization along with it’s unique Provider\_ID   * The provider\_Id is the primary key here | * Many to many relationship with Scholarship |
| Grade | It saves information about student and exam, the number of attemps a student takes and the score a student gets   * Student\_Id and Exam\_Id both act as composite primary key here | * It has one to many relationship with Student ( a student can have multiple grades for one exam in different attempts and a student must need to take a exam at least one and must need to score more than zero ) * One to one relationship with Exam( each exam has only one grade at a time) |

## Scripts For Data Storage

#### Creating Database Command using Microsoft SQL server 2019:

use D\_Project2;

CREATE TABLE Student

(

student\_ID int NOT NULL,

FirstName VARCHAR(30) NOT NULL,

FamilyName VARCHAR(30) NOT NULL,

Gender CHAR(30) NOT NULL ,

Date\_OF\_Birth VARCHAR(20) NOT NULL,

Phon\_No INT NOT NULL,

PRIMARY KEY(student\_ID)

);

CREATE TABLE Scholarship

(

Scholarship\_ID int NOT NULL,

Scolarship\_Percentage int NOT NULL,

Condition VARCHAR(50) NOT NULL,

Start\_year INT NOT NULL,

Finish\_year INT NOT NUll,

Enrolment\_ID INT NOT NULL,

PRIMARY KEY (Scholarship\_ID)

);

CREATE TABLE Enrolment

(

Enrolment\_ID int NOT NULL,

student\_ID int NOT NULL,

Enrolment\_year INT NOT NULL,

Finish\_year INT NOT NULL,

Course\_ID INT NOT NULL,

PRIMARY KEY (Enrolment\_ID)

);

CREATE TABLE Grade

(

student\_ID INT NOT NULL,

Exam\_ID INT NOT NULL,

Num\_Attempts INT NOT NULL,

Score INT NOT NULL

);

CREATE TABLE Exam (

Exam\_ID int NOT NULL,

Exam\_Name VARCHAR(30) NOT NULL,

Exam\_requirement int NOT NUll,

Grade VARCHAR(50) NOT NULL,

Subject\_ID int NOT NULL,

PRIMARY KEY ( Exam\_ID)

);

CREATE TABLE Subject

(

Subject\_ID int NOT NULL,

Division VARCHAR(10) NOT NULL,

PRIMARY KEY(Subject\_ID)

);

CREATE TABLE Course

(

Course\_ID INT NOT NULL,

Course\_Name VARCHAR(80) NOT NULL,

PRIMARY KEY(Course\_ID)

);

CREATE TABLE Subject\_Course

(

Course\_ID int NOT NULL,

Subject\_ID int NOT NULL,

);

CREATE TABLE Enrolment\_Subject

(

Enrolment\_ID int NOT NULL,

Subject\_ID int NOT NULL,

);

CREATE TABLE ProviderInfo\_Scholarship

(

Provider\_ID INT NOT NULl,

Scholarship\_ID int NOT NULL,

Description Text NOT NULl

);

CREATE TABLE ProviderInfo

(

Provider\_ID INT NOT Null,

Provider\_name VARCHAR(80) NOT NULL,

Organisation VARCHAR(80) NOT NULL,

PRIMARY KEY(Provider\_ID)

);

ALTER TABLE Grade

ADD CONSTRAINT fk\_grade1

FOREIGN KEY (student\_ID) REFERENCES Student(student\_ID);

ALTER TABLE Grade

ADD CONSTRAINT fk\_grade2

FOREIGN KEY (Exam\_ID) REFERENCES Exam(Exam\_ID);

ALTER TABLE Grade

ADD CONSTRAINT Composite\_Key\_grade

PRIMARY KEY (student\_ID,Exam\_ID);

ALTER TABLE Enrolment

ADD CONSTRAINT fk\_enrolment

FOREIGN KEY (student\_ID) REFERENCES Student(student\_ID);

ALTER TABLE Enrolment

ADD CONSTRAINT fk\_enrolment1

FOREIGN KEY (Course\_ID) REFERENCES Course(Course\_ID);

ALTER TABLE ProviderInfo\_Scholarship

ADD CONSTRAINT fk\_ProviderInfo\_scholarship

FOREIGN KEY (Scholarship\_ID) REFERENCES Scholarship(Scholarship\_ID);

ALTER TABLE ProviderInfo\_Scholarship

ADD CONSTRAINT fk\_ProviderInfo\_scholarship1

FOREIGN KEY (Provider\_ID) REFERENCES ProviderInfo(Provider\_ID);

ALTER TABLE ProviderInfo\_Scholarship

ADD CONSTRAINT Composit\_ProviderInfo\_scholarship1

PRIMARY KEY(Scholarship\_ID,Provider\_ID);

ALTER TABLE Subject\_Course

ADD CONSTRAINT fk\_Subject\_Course

FOREIGN KEY (Subject\_ID) REFERENCES Subject(Subject\_ID);

ALTER TABLE Subject\_Course

ADD CONSTRAINT fk\_Subject\_Course1

FOREIGN KEY (Course\_ID) REFERENCES Course(Course\_ID);

ALTER TABLE Subject\_Course

ADD CONSTRAINT Comp\_KEY\_Sub\_Course

PRIMARY KEY (Subject\_ID ,Course\_ID);

ALTER TABLE Enrolment\_Subject

ADD CONSTRAINT fk\_Enrolment\_Subject

FOREIGN KEY (Subject\_ID) REFERENCES Subject(Subject\_ID);

ALTER TABLE Enrolment\_Subject

ADD CONSTRAINT fk\_Enrolment\_Subject1

FOREIGN KEY (Enrolment\_ID) REFERENCES Enrolment(Enrolment\_ID);

ALTER TABLE Enrolment\_Subject

ADD CONSTRAINT Comp\_KEY\_Enrolment\_Subject

PRIMARY KEY (Subject\_ID ,Enrolment\_ID);

ALTER TABLE Scholarship

ADD CONSTRAINT fk\_Scholarship

FOREIGN KEY (Enrolment\_ID) REFERENCES Enrolment(Enrolment\_ID);

ALTER TABLE Exam

ADD CONSTRAINT fk\_Exam

FOREIGN KEY (Subject\_ID) REFERENCES Subject(Subject\_ID);

Search Queries

QUERY 1:

SELECT [dbo].[Grade].Num\_Attempts, [dbo].[Grade].Score, [dbo].[Grade].Exam\_ID, [dbo].[Student].FirstName, [dbo].[Student].FamilyName, [dbo].[Student].student\_ID

FROM [dbo].[Grade]

INNER JOIN [dbo].[Student]

ON [dbo].[Grade].student\_ID=[dbo].[Student].student\_ID

WHERE [dbo].[Grade].Score > 90

ORDER BY [dbo].[Grade].Score DESC;

Purpose:

* Ordering students score from descending order, joining the student class and score class to determine who scored the highest and listing their name.

Result:

Graphical user interface, application, Word

Description automatically generated

QUERY 2:

use D\_Project3;

SELECT [dbo].[Scholarship].[Scolarship\_Percentage], [dbo].[Scholarship].[Condition],[dbo].[ProviderInfo].[Provider\_name],[dbo].[ProviderInfo].[Organisation],[dbo].[ProviderInfo\_Scholarship].[Description]

FROM [dbo].[Scholarship]

INNER JOIN [dbo].[ProviderInfo\_Scholarship] ON [dbo].[ProviderInfo\_Scholarship].[Scholarship\_ID] = [dbo].[Scholarship].[Scholarship\_ID]

INNER JOIN [dbo].[ProviderInfo] ON [dbo].[ProviderInfo\_Scholarship].[Provider\_ID] = [dbo].[ProviderInfo].[Provider\_ID]

ORDER BY [dbo].[ProviderInfo\_Scholarship].Scholarship\_ID ASC;

### RESULT:

Text

Description automatically generated with medium confidence

#### Purpose:

The purpose of this query is to find out information about the scholarship percentage, it’s maintenance condition , it’s provider and a bit of description about the scholarship . We get it by scholarship ID through ascending order. And we get 1000 results out of it cause we have 1000 rows

QUERY 3:

SELECT [dbo].[Grade].Score, [dbo].[Student].student\_ID, [dbo].[Student].FirstName, [dbo].[Student].FamilyName, [dbo].[Enrolment].Enrolment\_ID, [dbo].[Scholarship].[Scholarship\_ID], [dbo].[Scholarship].Condition

FROM [dbo].[Grade]

INNER JOIN [dbo].[Student] ON [dbo].[Grade].student\_ID = [dbo].[Student].student\_ID

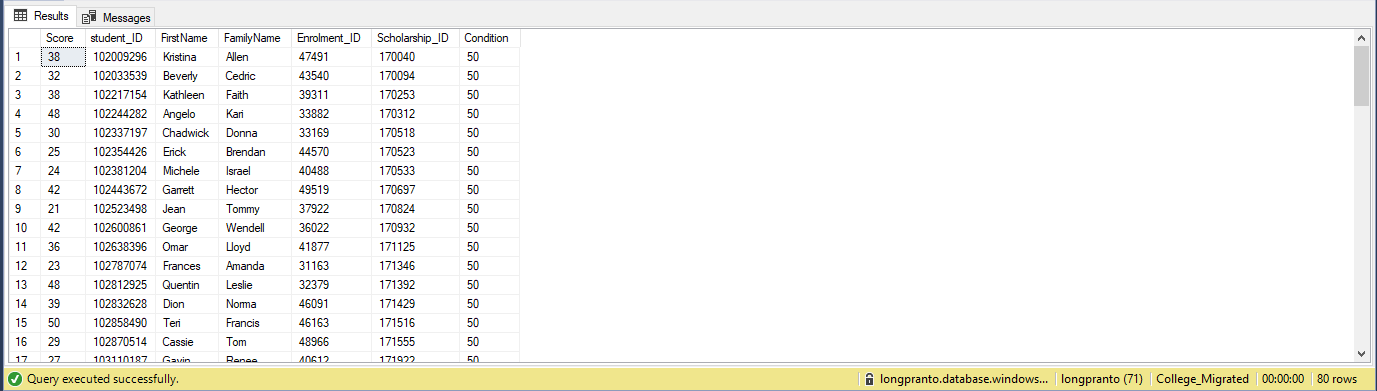
INNER JOIN [dbo].[Enrolment] ON [dbo].[Student].student\_ID = [dbo].[Enrolment].student\_ID

INNER JOIN [dbo].[Scholarship] ON [dbo].[Scholarship].Scholarship\_ID = [dbo].[Enrolment].Scholarship\_ID

WHERE [dbo].[Grade].[Score] <= 50 AND [dbo].[Scholarship].[Condition] = 50;

Purpose:

The purpose is to check which student has fallen below the university’s scholarship threshold or at risk for their scholarship. This will allow the College to make follow-up calls to warn students if they have fallen in this query. In this query, out of the 1000 students. 80 of the students has fallen below the condition threshold.

Result: