**PES UNIVERSITY EC Campus**

**1 KM before Electronic City,**

**Hosur Road, Bangalore-560100**

A Project Report on

**TIMETABLE MANAGEMENT SYSTEM**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**Database Management system (DBMS)**

For the Academic year 2020 . by

**Aditya Abhishek (PES2201800486)**

**TITLE**

**TIMETABLE MANAGEMENT SYSTEM: A Code to implement a timetable**

**ABSTRACT**

A [**Timetable**](https://en.wikipedia.org/wiki/Timetable) is a kind of schedule that sets out times at which specific events are intended to occur. It may also refer to:

* [School timetable](https://en.wikipedia.org/wiki/School_timetable), a table for coordinating students, teachers, rooms, and other resources
* [Time horizon](https://en.wikipedia.org/wiki/Time_horizon), a fixed point of time in the future at which point certain processes will be evaluated or assumed to end
* [Timeline](https://en.wikipedia.org/wiki/Timeline), a project artifact. It is typically a graphic design showing a long bar labeled with dates alongside itself and (usually) events labeled on points where they would have happened. It is used to show events along a period of time
* "Time Table", a track on the 1972 album [*Foxtrot*](https://en.wikipedia.org/wiki/Foxtrot_(album)), by English progressive rock band Genesis
* [*Time Table* (film)](https://en.wikipedia.org/wiki/Time_Table_(film)), a 1956 American black-and-white crime film noir

The purpose of timetabling and strategic scheduling in schools is multifaceted. First, timetables give students, teachers, parents, and administrators a clear picture of what's happening in the classroom at any given time. Second, timetables assist in maximizing school resources such as classroom availability, teacher availability, and materials availability. Third, timetables add structure to the school system for planning and documentation purposes. Lastly, timetables help to ensure accountability. For example, are the students receiving sufficient instructional time in each subject area? Timetabling also helps in the facilitation of parent-teacher meetings, scheduling across multiple campuses, and assigning classes to teachers.

Timetable construction is hardworking and complicated task when there are large number of course arrays and limited resources. As a result, universities and some institutes tend to solve this issue manually even; the results may not always fully optimal. In this paper, we discuss about a framework of utilizing timetable management system to a medium scale university for resource optimization. Our endeavor through the overall research was to develop an automated timetable management system. Timetable concerns all activities with regards to producing schedule that must subjective to different constraints. Timetable can be defined as the optimization of given activities, action or events to a set of objects in space-time matrix to satisfy a set of desirable constraints.

A key factor is running an educational center or basically an academic environment is the for a well-planned, well-throughout and clash free timetable. A college timetable is a temporal arrangement of a set of lectures and classrooms in which all given constraints are defined. Creating such timetable is complex and time-consuming process. Hence we have developed a practical approach for building lecture course timetable system The transactions are between lecturers, timetable and classes to which they are allotted..

In a DBMS, a trigger is a SQL procedure that initiates an action. A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. In this project for any action that violate the trigger gives an error message.

A query is a request for data or information from a database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs or complex results, e.g., trend analyses from data-mining tools. Using queries you can get retrieve information like to know a teacher’s timetable for a day or week, which class is going etc.A timetable with mandated period lengths, and specific subjects for each period helps administrators allocate sufficient resources to the most important curriculum areas. Curriculum should be organized so that the most important subjects are at optimal times of the day. For example, if an administrator believes that the morning is the best time to schedule English language arts courses, then reading and writing should almost always be scheduled before lunch. time and duration of each class period. Without a school timetable, students would have no idea how to prepare for the day,how much transit time they have before they are late for class, and who their teachers are.

**CONTENTS**

1. INTRODUCTION
   1. PROBLEM DEFINITION
   2. MINIWORLD
2. DATA MODEL
   1. ENTITY-RELATIONSHIP MODEL

2.2 RELATIONAL MODEL

2.2.1 ASSUMPTIONS AND CONSTRAINTS

1. FUNCTIONAL DEPENDENCY AND NORMALISATION
   1. NORMAL FORM 1
   2. NORMAL FORM 2
      1. VIOLATIONS OF NORMAL FORM 2
   3. NORMAL FORM 3
      1. VIOLATIONS OF NORMAL FORM 3
2. DATA DEFINITION LANGUAGE (DDL)
   1. CODE
3. QUERIES
4. TRIGGERS
5. CONCLUSION
6. **INTRODUCTION**

The timetable is a necessary tool for the efficient working of a school. It is really a mirror that reflects the entire educational programme of the school. "It is the timetable that supplies the framework within which the work of the school proceeds. It is the instrument through which the purpose of the school is to function." The values of the school timetable may be enumerated as follows:

1. It ensures smooth and orderly working of the school: The major achievement of timetable is that everything is planned in advance. All the teachers and students know their jobs as well as the time they are to devote to each activity. It is due to the timetable that smooth, orderly and regular work in the school goes on even in the absence of the Head Teacher or any one of the teachers. If there is no timetable in the school, there is always the danger of negligence of duty, duplication of effort and repetition of unnecessary items and activities. The timetable places proper persons at their proper places, at the proper time and in the proper manner.

2. It prevents wastage of time and energy: The timetable shows exactly what is to be done at a particular time. It, thus, directs the attention of both the pupil and the teacher to one thing at a time. Thus one's energy is automatically directed in a particular direction and this prevents a lot of wastage of time and energy. It also prevents confusion, duplication, overlapping and unnecessary repetition on the part of the pupil and the teacher.

3. It ensures equitable distribution of work among teachers: With the help of the timetable, the Head Teacher can keep track of the quantum of work load assigned to each teacher working under hirnlher. Thd timetable gives a summary of the work allotted to each teacher. The Head Teacher, or 'another superior off~cerc, an know at a glance the amount of work that every teacher is expected to do. Thus, the timetable helps in avoiding the allotment of too much or too little work to,one teacher. This prevents heart burning among the teachers and so unnecessary tension to the Head Teacher on this account. ~ f t earl l, the Head Teacher has to have a congenial atmosphere in the school for optimal efficiency as well as efficacy.

4. It ensures equitable distribution of time to different subjects and activities: The timetable gives due place, extension and emphasis to various subjects and activities in the school, according to their relative imqortance or difficulty. This is very essential for the all round development ofthe pupils. This ensures that while the more important subjects and activities get more attention and time, the less important ones are not neglected.

5. It helps in adjusting schoolwork according to the needs of pupils: The timetable helps the school authorities to adjust schoolwork according to the physiological needs of pupils. Fatigue, interest and freshness of mind and body are given due consideration at the time of constructing a timetable. As the Head, you should see to it that ample time is provided in the timetable for the teachers to check the pupils' notebooks and also to prepare for their lessons. As far as the pupils are concerned, the Head must ensure that co-curricular activities like sports, music, art and craft, library and other such activities get adequate time. This is very important for the all round development of the pupils..

**1.1 PROBLEM DEFINTION**

To implement a timetable management system.

**1.2 MINIWORLD**

1. A basic TIMETABLE for a student consists of numerous Period(s) which consists of the PNo, the Day of the weak and the StartTime and EndTime for the Period.

2. The LECTURER table contains the details of the lecturer in the form of his/her Name, Initials and Dept he/she is in.

3. TIMETABLE(s) are assigned to the LECTURER who teaches the SEM\_CLASS(es) for a particular SemSection and Year.

4. SemSection contains the Semester,the Section and the RoomNo for the class.

5. The LECTURER offers different COURSE(es) to different SEM\_CLASS(es).

6. A number of LECTURER(s) teaches a particular COURSE which has its unique CCode, its CName and the Credits for that particular course.

7. This COURSE is offered during a particular year for different SEM\_CLASS(es).

8. It is natural for a LECTURER to take a number of days off for HOLIDAY(s) he has planed .

9. A HOLIDAY contains the no of days off with the StartDate and the EndDate of the leave.

1. **DATA MODEL**

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

A database can be described with any one of these depending on several factors. The biggest factor is whether the database management system you are using supports a particular model. Most database management systems are built with a particular data model in mind and require their users to adopt that model, although some do support multiple models.In addition, different models apply to different stages of the database design process. High-level conceptual data models are best for mapping out relationships between data in ways that people perceive that data. Record-based logical models, on the other hand, more closely reflect ways that the data is stored on the server.

Selecting a data model is also a matter of aligning your priorities for the database with the strengths of a particular model, whether those priorities include speed, cost reduction, usability, or something else. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

* Hierarchical Model
* Network Model
* Entity-relationship Model
* Relational Model

We here will discuss two models in details.

* 1. **ENTITY-RELATIONSHIP MODEL**

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

## **Entity**

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

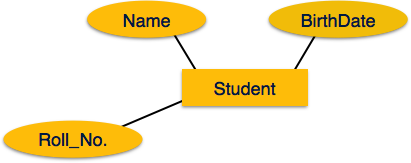
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Entities in a school database

## **Attributes**

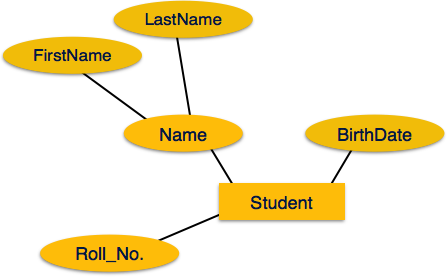
Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

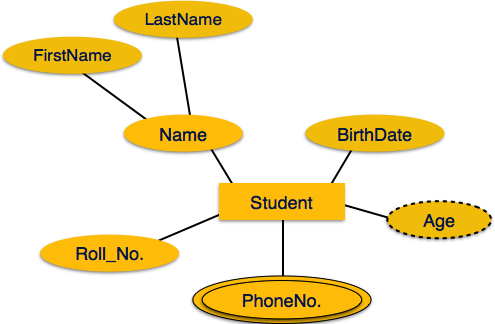


### **Types of Attributes**

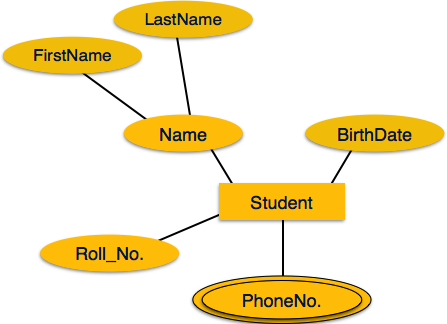
* **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
* **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.
* **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.



* **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.



* **Multi-valued attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.



**Entity-Set and Keys**

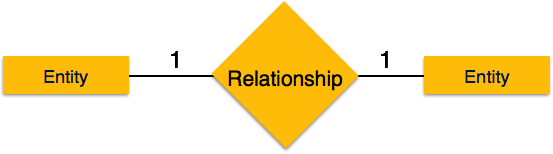
Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll\_number of a student makes him/her identifiable among students.

* **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
* **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
* **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

## **Relationship**

The association among entities is called a relationship. For example, an employee **works\_at** a department, a student **enrolls** in a course. Here, Works\_at and Enrolls are called relationships.



### **Relationship Set**

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

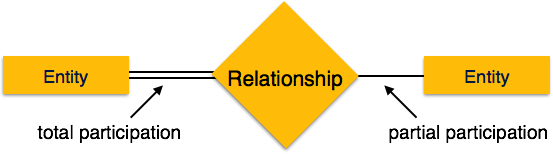
### **Degree of Relationship**

The number of participating entities in a relationship defines the degree of the relationship.

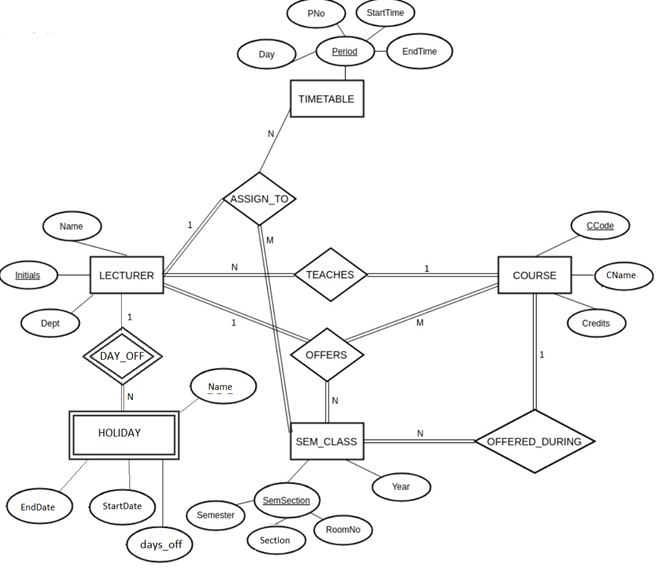
* Binary = degree 2
* Ternary = degree 3
* n-ary = degree

### **Participation Constraints**

* **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
* **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



**E-R DIAGRAM FOR THE PROJECT**



* 1. **RELATOINAL MODEL**

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

## **Concepts**

**Tables** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

**Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.

**Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

**Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.

**Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

**Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain.

## **Constraints**

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints −

* Key constraints
* Domain constraints
* Referential integrity constraints

### **Key Constraints**

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

* in a relation with a key attribute, no two tuples can have identical values for key attributes.
* a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

### **Domain Constraints**

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

### **Referential integrity Constraints**

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

RELATIONAL SCHEMA FOR THIS PROJECT

# 

**2.2.1 ASSUMPTIONS AND CONSTRAINTS**·

1. For TIMETABLE, ***period*** is composite primary key . Hence, day, P\_No, Start\_time, End\_time are also primary key.
2. For LECTURER, ***initials*** is a primary key as it is unique for all tuples.
3. For SEM\_CLASS ***Sem\_section*** is composite primary key hence semester, section and room\_no are also primary key.
4. For COURSE ***C\_Code*** is primary key as it is unique for every subject.
5. For ASSIGN\_TO ***initals*** is foreign key referencing to relation lecturer and is a primary key of its own relation.

Similarly sem, section, room\_no, day, p\_no, start\_t, end\_t are also foreign keys referencing to timetable and sem\_class and primary of its own relation.

1. For OFFERS ***initials, c\_code, semester, section and room\_no*** are foreign keys and primary keys of its own relation.
2. For HOLIDAY ***h\_initial*** is a foreign key referencing to lecturer and also primary key and h\_name is primary key.
3. Most of the attributes have datatype varchar only ***days\_off*** has datatype integer.

# **FUNCTIONAL DEPENDENCY AND NORMALIZATION**

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

1. X   →   Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**For example:**

Assume we have an employee table with attributes: Emp\_Id, Emp\_Name, Emp\_Address.

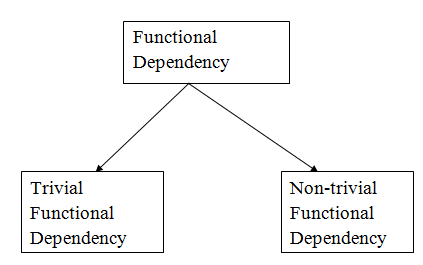
Here Emp\_Id attribute can uniquely identify the Emp\_Name attribute of employee table because if we know the Emp\_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

1. Emp\_Id → Emp\_Name

We can say that Emp\_Name is functionally dependent on Emp\_Id.

## **Types of Functional dependency**



### **1. Trivial functional dependency**

* A → B has trivial functional dependency if B is a subset of A.
* The following dependencies are also trivial like: A → A, B → B

**Example:**

1. Consider a table with two columns Employee\_Id and Employee\_Name.
2. {Employee\_id, Employee\_Name}   →    Employee\_Id is a trivial functional dependency as
3. Employee\_Id is a subset of {Employee\_Id, Employee\_Name}.
4. Also, Employee\_Id → Employee\_Id and Employee\_Name   →    Employee\_Name are trivial dependencies too.

### **2. Non-trivial functional dependency**

* A → B has a non-trivial functional dependency if B is not a subset of A.
* When A intersection B is NULL, then A → B is called as complete non-trivial.

**Example:**

1. ID   →    Name,
2. Name   →    DOB

**3.1 1 NORMAL FORM**

As evident from the ER diagram, there are no multivalued attributes. In relations timetable and sem\_class, there are composite attributes. We need to expand these attributes to make them atomic. The results can be seen after comparing the ER model with the new schema diagram. It is the first normal form.

**3.2 2 NORMAL FORM**

In relation timetable all are primary keys hence it is in second normal form.

In relation lecturer

initials-> name, dept, c\_code

Since there is no partial dependency, hence it is in second normal form.

In relation sem\_class

(semester,section,room\_no)-> year

Since there is no partial dependency, hence it is in second normal form.

In relation course

c\_code->c\_name,credits

Since there is no partial dependency, hence it is in second normal form.

In relation assign\_to, since all are primary keys, hence it is in second normal form.

In relation offers, since all are primary keys, hence it is in second normal form.

In relation holiday

(h\_initials,h\_name)->StartDate,EndDate

Since there is no partial dependency, hence it is in second normal form.

**3.2.1 CASES OF VIOLATIONS OF SECOND NORMAL FORM**

Let us discuss a couple of cases when the second normal form would be violated

1. If in the table offers cname is added too,then, even if we c\_code is removed, the name attribute would still be dependent on the second normal form

Thus it’s a case of partial dependency which needs to be resolved.

1. If in the table Assign\_to, we add an attribute Year then, even on removing day, p\_no, start\_t, end\_t the Year attribute would still be dependent on sem, section, room\_no, thus making a case of partial dependency.

**3.3 3 NORMAL FORM**

Since there is no case of transitive dependency, each table is in 3 normal form.

**3.3.1 VIOLATIONS OF 3 NORMAL FORM**

If C\_Name is added to the table Lectures then there is a case of transitive dependency which would need to be resolved.

The other example can be when we add Years to the table Assign\_To.

**4.0 DATA DEFINITION LANGUAGE (DDL)**

DDL stands for "Data Definition Language." A DDL is a language used to define data structures and modify [data](https://techterms.com/definition/data). For example, DDL commands can be used to add, remove, or modify [tables](https://techterms.com/definition/table) within in a [database](https://techterms.com/definition/database). DDLs used in database applications are considered a subset of [SQL](https://techterms.com/definition/sql), the Structured Query Language. However, a DDL may also define other types of data, such as [XML](https://techterms.com/definition/xml).

A Data Definition Language has a pre-defined [syntax](https://techterms.com/definition/syntax) for describing data. For example, to build a new table using SQL syntax, the CREATE command is used, followed by parameters for the table name and [column](https://techterms.com/definition/column) definitions. The DDL can also define the name of each column and the associated [data type](https://techterms.com/definition/datatype). Once a table is created, it can be modified using the ALTER command. If the table is no longer needed, the DROP command can be used to delete the table.

Since DDL is a subset of SQL, it does not include all the possible SQL commands. For example, commands such as SELECT and INSERT are considered part of the Data Manipulation Language (DML), while access commands such as CONNECT and EXECUTE are part of the Data Control Language (DCL). The DDL, DML, and DCL languages include most of the commands supported by SQL.

Data definition or data description language (DDL) is a syntax similar to a computer [programming language](https://en.wikipedia.org/wiki/Programming_language) for defining [data structures](https://en.wikipedia.org/wiki/Data_structure), especially [database schemas](https://en.wikipedia.org/wiki/Database_schema). DDL statements create and modify database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP

The ***CREATE*** statement in [SQL](https://en.wikipedia.org/wiki/SQL) creates a component in a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS).

The ***DROP*** statement destroys an existing database, table, index, or view.

The ***ALTER*** statement modifies an existing database object.

Another type of DDL sentence in SQL is used to define [referential integrity](https://en.wikipedia.org/wiki/Referential_integrity) relationships, usually implemented as [primary key](https://en.wikipedia.org/wiki/Primary_key) and [foreign key](https://en.wikipedia.org/wiki/Foreign_key) tags in some columns of the tables. These two statements can be included in a***CREATE TABLE*** or an ***ALTER TABLE*** sentence;

***CODE:***

CREATE TABLE timetable

(

Day VARCHAR(15) NOT NULL,

PNo INT NOT NULL,

Start\_time TIME NOT NULL,

End\_time TIME NOT NULL ,

PRIMARY KEY (Day,PNo,Start\_time,End\_time),

CHECK (Start\_time < End\_time));

CREATE TABLE sem\_class

( Sem INT NOT NULL CHECK(sem<=8),

Section CHAR NOT NULL,

Room\_No VARCHAR(12) NOT NULL,

Year INT NOT NULL,

PRIMARY KEY (Sem,Section,Room\_No));

CREATE TABLE course

( C\_Code VARCHAR(10) NOT NULL,

C\_Name VARCHAR(15) NOT NULL,

Credits INT NOT NULL CHECK(Credits<=4),

PRIMARY KEY (C\_Code),

UNIQUE (C\_Name));

CREATE TABLE lecturer

( Initials VARCHAR(10) NOT NULL,

Name VARCHAR(15) NOT NULL,

Dept VARCHAR(15) NOT NULL,

C\_Code VARCHAR(10) NOT NULL,

PRIMARY KEY (Initials),

FOREIGN KEY (C\_Code) REFERENCES course(C\_Code)

ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE assign\_to

( Initials VARCHAR(10) NOT NULL ,

C\_Code VARCHAR(10) NOT NULL,

Sem INT NOT NULL CHECK(sem<=8),

Section CHAR NOT NULL,

Room\_No VARCHAR(12) NOT NULL,

Day VARCHAR(15) NOT NULL,

PNo INT NOT NULL,

Start\_time TIME NOT NULL ,

End\_time TIME NOT NULL ,

PRIMARY KEY (Initials,Sem,Section,Room\_No,Day,PNo,Start\_time,End\_time),

FOREIGN KEY (Initials) REFERENCES lecturer(Initials)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Sem,Section,Room\_No) REFERENCES sem\_class(Sem,Section,Room\_No)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Day,PNo,Start\_time,End\_time) REFERENCES timetable(Day,PNo,Start\_time,End\_time)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (C\_Code) REFERENCES course(C\_Code));

CREATE TABLE offers

( Initials VARCHAR(10) NOT NULL,

C\_Code VARCHAR(10) NOT NULL,

Sem INT NOT NULL CHECK(sem<=8),

Section CHAR NOT NULL,

Room\_No VARCHAR(12) NOT NULL,

PRIMARY KEY (Initials,C\_Code,Sem,Section,Room\_No),

FOREIGN KEY (Initials) REFERENCES lecturer(Initials),

FOREIGN KEY (C\_Code) REFERENCES course(C\_Code),

FOREIGN KEY (Sem,Section,Room\_No) REFERENCES sem\_class(Sem,Section,Room\_No));

CREATE TABLE Holiday

( Initials VARCHAR(10) NOT NULL,

Name VARCHAR(15) NOT NULL,

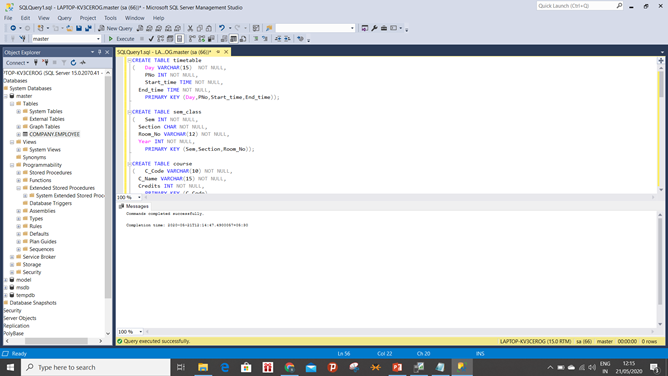
Startdate DATE NOT NULL,

EndDate DATE NOT NULL,

days\_off INTEGER NOT NULL,

PRIMARY KEY(Initials, Name),

FOREIGN KEY(Initials) REFERENCES lecturer(Initials));



# Triggers

When a lecturer resigns from his/her post, all the details of that lecturer should automatically be deleted from all other tables.

CREATE TRIGGER trgAfterDelete ON offers

After DELETE

AS

declare @Initials VARCHAR(10);

select @Initials=i.Initials from deleted i;

DELETE FROM Holiday WHERE Initials=@Initials;

DELETE FROM lecturer WHERE Initials=@Initials;

DELETE FROM assign\_to WHERE Initials=@Initials;

# GO

# SQL Queries

--to display timetable for a lecturer for whole week

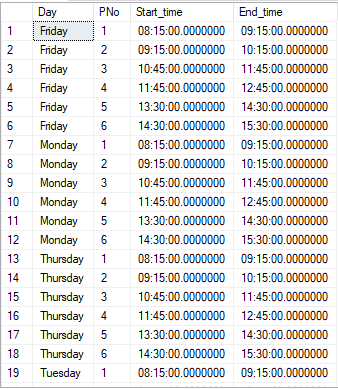
SELECT \*

FROM timetable

where EXISTS(SELECT pno,day

FROM assign\_to

WHERE initials='VRB');



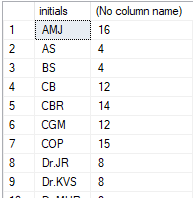
-- To calculate workload for a single lecturer

SELECT COUNT(\*) FROM assign\_to WHERE initials='SVI';



-- To calculate workload of each lecturer

SELECT initials,COUNT(\*) FROM assign\_to GROUP BY initials;



-- To find vacant classes

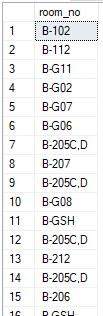
SELECT room\_no

FROM sem\_class

WHERE room\_no NOT IN (SELECT room\_no

FROM assign\_to

WHERE start\_time='14:30:00' AND end\_time='15:30:00' AND day='Monday');

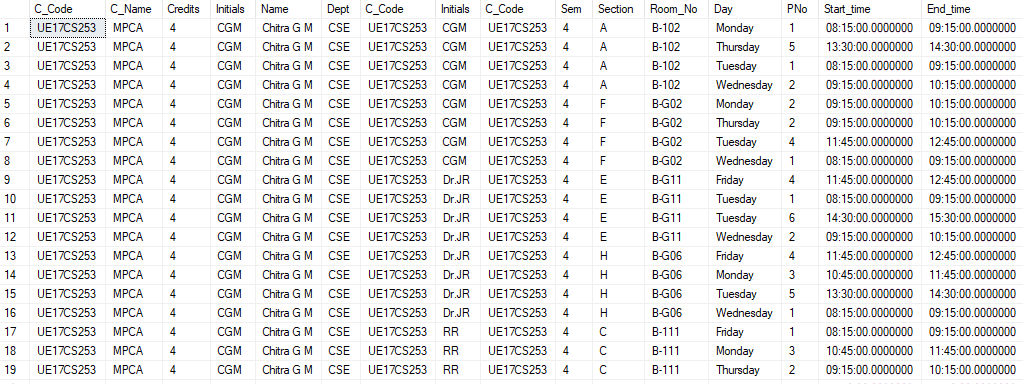


-- display all the detalis of a particular subject

SELECT \*

FROM ((course AS c JOIN lecturer AS l ON l.C\_Code=c.C\_Code) JOIN assign\_to AS a ON l.C\_Code=a.C\_Code)

WHERE l.C\_Code='UE17CS253';



**CONCLUSION**

Eliciting user requirements it very important process in order to develop system meets the user needs. Ineffective requirements methods or models lead to failure system development. Eliciting user requirements sometimes treated as an unimportant process. This attitude is changing as collecting requirements is increasingly recognized as a critically important activity in any system development process. The novelty of many services and applications, the speed with which they need to be developed, and benefits expected from the system all play a significant role in choosing the suitable development process. will continue, and Eliciting requirements technique will evolve to cope with the change in practice and technologies developments. Requirements elicitation models or techniques is essential element in determining the success or failure of projects, and in defining the attributes of the good system. In this paper, we have presented a case study describing users’ needs and factors related to the development of TMS. As we shall see, developing system for automated university timetable is not just a matter of technological platform. On one hand, it requires a deeper understanding of user needs to propose solutions that fulfil these needs and, thus, has a better chance to get adopted by the community. On the other hand, there are many requirements models available and the best system design depends on the adaptation of system according to system constraints. The model suggested above will contribute to a better understanding of the requirements of endusers in adopting TMS in general. It is our opinion that the proposed model will produce general knowledge needed to improve TMS in future. Developer must followed the requirements and needs of users in details to ensure a fruitful outcome, the designers must satisfy the needs and wants of the user when the development is complete. To achieve this, users’ needs should not only be elicited by techniques such as surveys, focus groups, interviews, etc., but should also be reflected back to users via validating the requirements model in order to prototype the user requirements. Then, the requirements will, of course, evolve as the system develops and a more formal user evaluation will take place. The next face of this research develops TMS and evaluates it.