**CHAPTER ONE**

**INTRODUCTION**

**1.1 Introduction**

The project titled “Student & Library Database Management System” consists of two case studies. The first case study is creating student management system to deal with the case of each type of student courses and curriculum data and details of the batch and resources are other information related to the details of the study. This management system should cover all information related to the progress of each student. From communing his/her study in the first semester unity Graduation.We can have a design make it easier for us to explore all the ongoing activities at the college, so that we can identify any college to be appointed who, of course, the current situation of the student, and the percentage of attendance of the students and the requirements for the next student. Student management system is an automated version of the system management book by a student. It can handle all the details about the student. Manual system in case they need a lot of time, manpower. Work here almost computerized. So as to maintain the accuracy. Keep a backup copy of the very easy. It can do within a few minutes. The system has two types of access to the media, administrator and user. Student management system and managed by an administrator. It is the job of an administrator to insert update and monitor the entire process when a user logs on to the system.

The second case study is to build a database model for library management system. This model mainly focuses on basic operations in library like adding new members, new books, updating new information, searching books and members, and facilitating borrowing and returning books. The model is easy to use for both beginners and advanced users. The designed software has an attractive interface, combined with strong searching insertion and reporting capability. By using the model, a report containing a list of books borrowed and/or returned by members can be obtained.

.

**1.2 Objectives of the project**

The main objective of this project is to develop a data base model for library & student management system that can be applied in universities’ libraries or public libraries. Specific objectives of the project are summarized as follow:

1. To build logical data base model suitable for implementation in student and library database management system (S&LDMS)
2. To identify the basic data entities involved in S&LDMS
3. To identify the relationships between the data entities of the S&LDMS

**CHAPTER TWO**

**DATABASE MANAGEMENT SYSTEM**

**2.1 Introduction to Database Management System (DBMS)**

A database management system (DBMS) is system software for creating and managing [databases](http://searchsqlserver.techtarget.com/definition/database). The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage [data](http://searchdatamanagement.techtarget.com/definition/data).

A DBMS makes it possible for end users to create, read, update and delete [data](http://searchdatamanagement.techtarget.com/definition/data) in a database. The DBMS essentially serves as an interface between the [database](http://searchsqlserver.techtarget.com/definition/database) and end users or [application programs](http://searchsoftwarequality.techtarget.com/definition/application-program), ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database [engine](http://whatis.techtarget.com/definition/engine) that allows data to be accessed, locked and modified -- and the database [schema](http://searchsqlserver.techtarget.com/definition/schema), which defines the database’s logical structure. These three foundational elements help provide [concurrency](http://searchoracle.techtarget.com/definition/concurrent-processing), security, [data integrity](http://searchdatacenter.techtarget.com/definition/integrity) and uniform administration procedures. Typical database administration tasks supported by the DBMS include [change management](http://searchcio.techtarget.com/definition/change-management), performance monitoring/tuning and [backup](http://searchstorage.techtarget.com/definition/backup) and [recovery](http://searchstorage.techtarget.com/definition/recovery). Many database management systems are also responsible for automated [rollbacks](http://searchsqlserver.techtarget.com/definition/rollback), restarts and recovery as well as the [logging](http://whatis.techtarget.com/definition/log-log-file) and [auditing](http://searchcio.techtarget.com/definition/audit-trail) of activity.

The DBMS is perhaps most useful for providing a centralized view of data that can be accessed by multiple users, from multiple locations, in a controlled manner. A DBMS can limit what data the end user sees, as well as how that end user can view the data, providing many views of a single database schema. End users and software programs are free from having to understand where the data is physically located or on what type of storage media it resides because the DBMS handles all requests.

The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data ([storage](http://searchstorage.techtarget.com/definition/storage) and hardware). As long as programs use the application-programming interface ([API](http://searchexchange.techtarget.com/definition/application-program-interface)) for the database that is provided by the DBMS, developers will not have to modify programs just because changes have been made to the database.

With relational DBMSs ([RDBMSs](http://searchsqlserver.techtarget.com/definition/relational-database-management-system)), this API is [SQL](http://searchsqlserver.techtarget.com/definition/SQL), a standard programming language for defining, protecting and accessing data in a RDBMS.

Databases and database technology have a major on the growing use of computer. It is fair to say that database play a critical role in almost all areas where computers are used including business, electronic commerce engineering, medicine, genetics, law, education and library science.

**2.2 Database properties**

* A Database represents some aspect of the real ,world, sometimes called the miniworld or the universe of discourse ( UoD ). Changes to the miniworld are reflected in the database.
* A database is a logically coherent collection of data with some inherent meaning a random assortment of data cannot correctly be referred to as a database.
* A database is ,designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested. A database can be of any size and complexity and may be generated and maintained manually or it may be computerized.

In other words, a database has some source from which data is derived, some degree of interaction with event in the real world, and an audience that is actively in its contents. The end user of a database may perform business transaction or events may happen that cause the information in the database to change. in order for database to be accurate and reliable at all times, it must be a true reflection of the miniworld that it represents; therefore, changes must be reflected in the database as soon as possible.

**2.3** **Database and DBMS structures**

Units of data within a database are generally called 'record.' Each record and is unique and is further broken down into a limited number of 'fields,' which describe attributes of the record. For example, in an employee database, a record may designate each employee's name, title, and salary, date of hire, telephone extension, supervisor's name, and so forth. The fields may or may not be unique to the record, but at least one be unique for the record to be unique the fields may contain fixed or variable information, and they may contain either text or numbers. Figures in value- and date-type fields can be used for computation when the DBMS is used to analyze the data. Fields can even contain pictures, video clips, or sound if the DBMS and the computer hard ware are capable of handling such multimedia data. Records with the same set of field classifications are usually kept within one file. In a business database sets of records often exist both for concrete things, such as clients or vendors, and foe activities, such as orders, payments, and production statistic.

**2.4** **ADVANTAGES OF USING THE DBMS Approach**

In this section, we discuss some of the advantages of using a DBMS and the capabilities that a good DBMS should possess. These capabilities are in addition to the four to accomplish a variety of objectives related to the design, administration, and use of a large multiuser database.

**1- Controlling Redundancy**

In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications.

**2- Providing Persistent Storage for Program Objects**

Databases can be used to provide persistent storage for program objects and data structures. This is one of the main reason for object-oriented database systems. Programming languages typically have complex data structures, such as record types in Pascal or class definitions in C++ or Java. The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in

Permanent files, which often involves converting these complex structures into a format suitable for file storage. When the need arises to read this data once more, the programmer must convert from the file format to the program variable structure.

**3- Providing Storage Structures for Efficient Query Processing**

Database systems must provide capabilities for efficiently executing queries and updates. Because the database is typically stored on disk, the DBMS must provide specialized data structures to speed up disk search for the desired records. Auxiliary files called indexesare used for this purpose. Indexes are typically based on tree data structures or hash data structures, suitably modified for disk search. Inorder to process the database records needed by a particular query, those records.

**4- Providing Backup and Recovery**

A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystemof the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.

**5- Providing Multiple User Interfaces**

Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for stand-alone users.

**6- Representing Complex Relationships among Data**

A database may include numerous varieties of data that are interrelated in many ways.

A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.

**7- Enforcing Integrity constraints**

Most database applications have certain integrity constraints that must hold for the data. A DBM should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item. A more complex type of constraint that frequently occurs involves specifying that a record in one file must be related to records in other files we can specify, "Every section record must be related to a course record. This is known as referential integrity constraint another type of constraint specifies uniqueness on data item values, such a "every course record must have a unique value for Course Number." This is known as a key or uniqueness constraint these constraints are derived from the meaning or semantics of the data and of the miniworld, it represents. It is the database designers' responsibility to identify integrity constraints during database Design. Some constraints can be specified to the DBMS and automatically enforced. Other constraints may have to be checked by update programs or at the time of data entry.

**8- Permitting Inference and Actions Using Rules**

Some database systems provide capabilities for defining deduction rules for inference new information from the stored database facts. Such systems are called deductive database systems. For example, there may be complex rules in the mini world application for determining when a student is on probation. These can be specified declaratively as rules, which when compiled and maintained by the DBMS can determine all students on probation.

In a traditional DBMS, an explicit procedural prof-,Jmm code would have to be written to support such applications.

**9- Additional implication of using the database approach**

This part discusses some additional of using the database approach that can benefit most organizations. Potential for enforcing standards. The database user in a large departments, projects, and users within the organization. Standards can be defined for names and formats of data elements, display formats, report structures, terminology, and so on. The DBA can enforce standards in a centralized database environment more easily than in an environment where each user group has control of its own data files and software.

**10- Reduced application development time**

A prime selling feature of the database approach is that developing a new application such as the retrieval of certain data from the database for printing a new report takes very little time. Designing and implementing a large multiuser database from scratch, many take more time than writing a single specialized file application. However, once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities. Development time using DBMS is estimated to be one-sixth to one-fourth of that for a traditional file system flexibility. It may be necessary to change the structure of a database as requirements change. For example, a new user group may emerge that needs information not currently in the database. In response, it may be necessary to add a file to the database or to extend the data elements in an existing file. Modern DBMSs allow certain types of evolutionary changes to the structure of the database without affecting the stored data and the existing application programs.

**11- Availability of up-to-date information**

A DBMS makes the database available to all users. As soon as one user's update is applied to the database, all other users can immediately see this update. This availability of up-to-date information is essential for many transaction-processing applications, such as reservation system or banking database, and it is made possible by the concurrency control and recovery subsystems of a DBMS.

**12- Economic of Scale**

The DBMS approach permits consolidations, thus reducing the amount of wasteful overlap between activities of data-processing personnel in different projects or departments as well as redundancies among applications. This enables the whole organization to invest in more powerful processors, storage devices, or communication gear, rather than having each department purchase its own (lower performance) equipment. This reduces overall costs of operation and management.

**2.5 (Disadvantages) When Not Use a DBMS**

In spite of the advantages of using a DBMS, a DBMS may involve unnecessary overhead cost that would not be incurred in traditional file processing in few situations. The overhead costs of using a DBMS are due to the following:

* High initial investment in hardware, and training.
* The generality that a DBMS provides for defining and processing data.
* Overhead for providing security, concurrency control, recovery, and integrity following circumstances:

1. Simple, well-defined database applications that are not expected to change all
2. Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
3. Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
4. No multiple-user access to data certain industries and applications have elected not to use general-purpose DBMS

For example, many computer-aided design (CAD) tools used by mechanical and civil engineers have proprietary file and data management software that is geared for the internal manipulations of drawings and 3D objects. Similarly, communication and switching systems designed by companies like AT&T were early manifestations of database software that was made to rum very fast with hierarchically organized data for quick access and routing of calls.

**2.6 Relational Keys**

There are no duplicate tuples within a relation. Therefore, we need to be able to identify one or more attributes (called relational keys) that uniquely identifies each.

**2.6.1 Super key**

An attribute or a set of attributes that uniquely identifies a tuple within a relation. A super key uniquely identifies each tuple within a relation. However, a super key may contain additional attributes that are not necessary for unique identification, and we are interested in identifying super keys that contain only the minimum number of attributes necessary for unique identification.

**2.6.2 Candidate key**

A candidate key K for a relation R has two properties:

* Uniqueness - In each tuple of R, the values of K uniquely identify the tuple.
* Irreducibility – no proper subset of K has the uniqueness property.

There may be several candidate keys for a relation. When a key consist of more than one attribute we call it a composite key cannot be a candidate key. Allocates each branch office a unique branch number.

**2.6.3 Primary key**

The candidate key that is selected to identify tuples uniquely within the relation.

**2.6.4 Foreign key**

An attribute, or set of attributes, within one relation that matches the matches the candidate key of some (possibly the same) relation.

**CHAPTER THREE**

**GRAPHICAL MODELING OF THE PROPOSED MODELS**

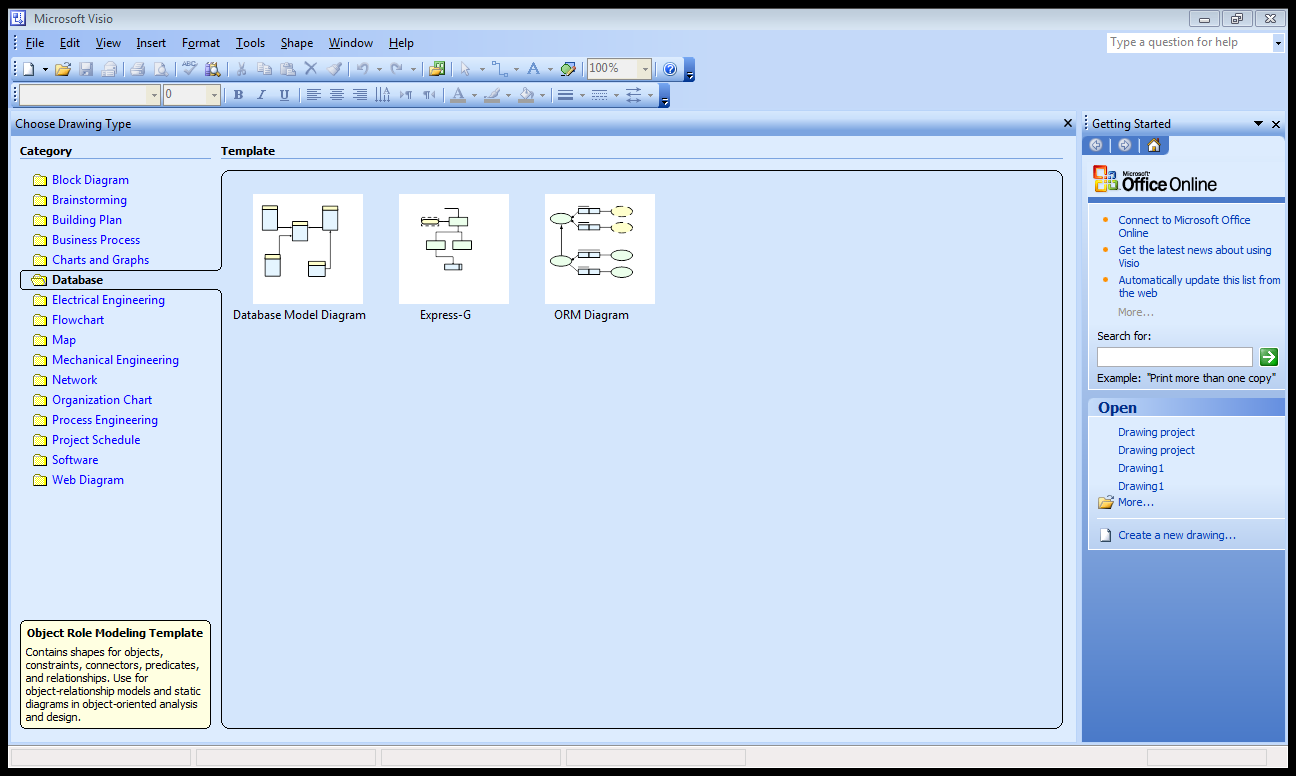
**3.1 Introduction**

This software has been used to design data. Microsoft Visio more advanced and professional user can connect his models more than one data source to display the information visually. Considered by university students around the world, particularly those who specialize in the field of engineering the first and most important tool for the work of all types of graphical models and running this program on Windows system. This program specializes various illustrative drawings of them, for example, Process Flow organizational structures entity relationship diagrams.

**3.2 Draw Entity Relationship Diagram (ERD)**

The following steps are followed in the graphical modeling of the database models on MS Visio:

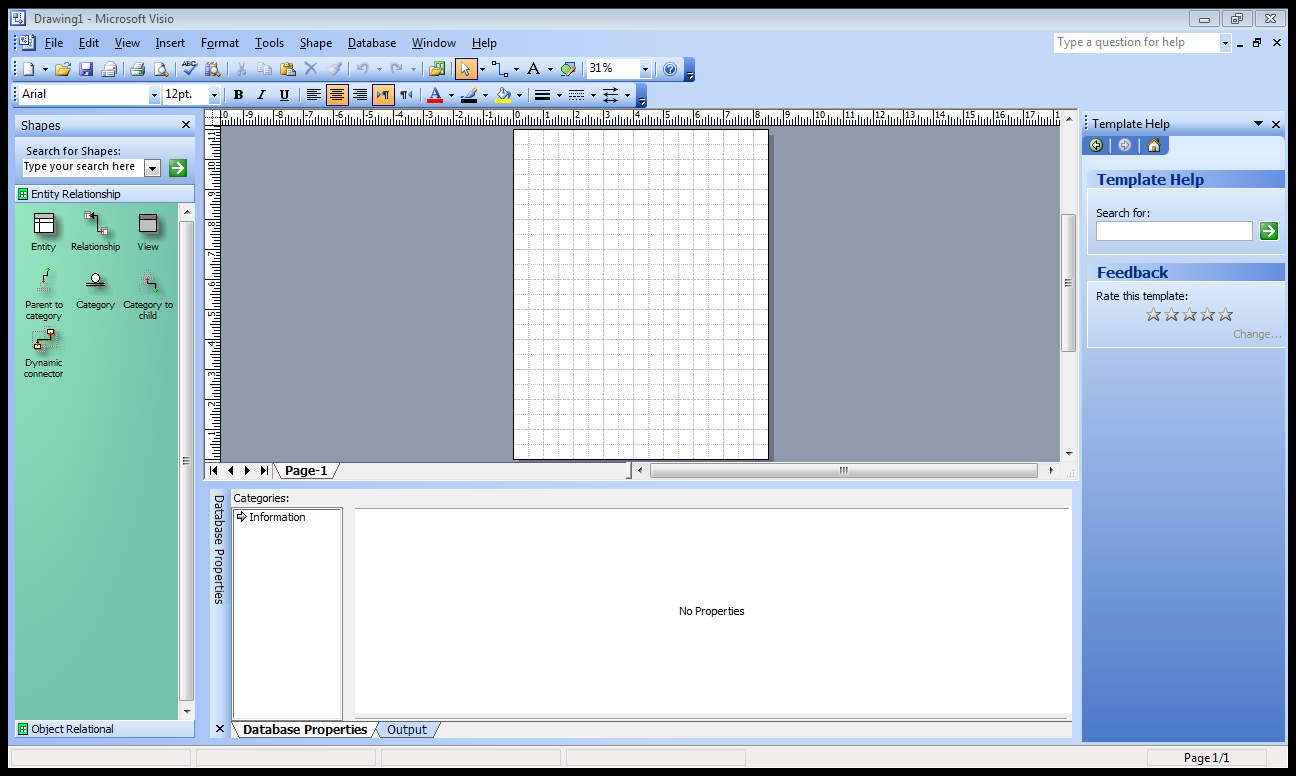
1. First open the program then click on Database Model Diagram. (as shown in figure 3.1).



**Fig, 3.1** Open the program

2. From files click on page setup and then click on the preferable size. We even could

Do customize size for the page. (as shown in figure 3.2).

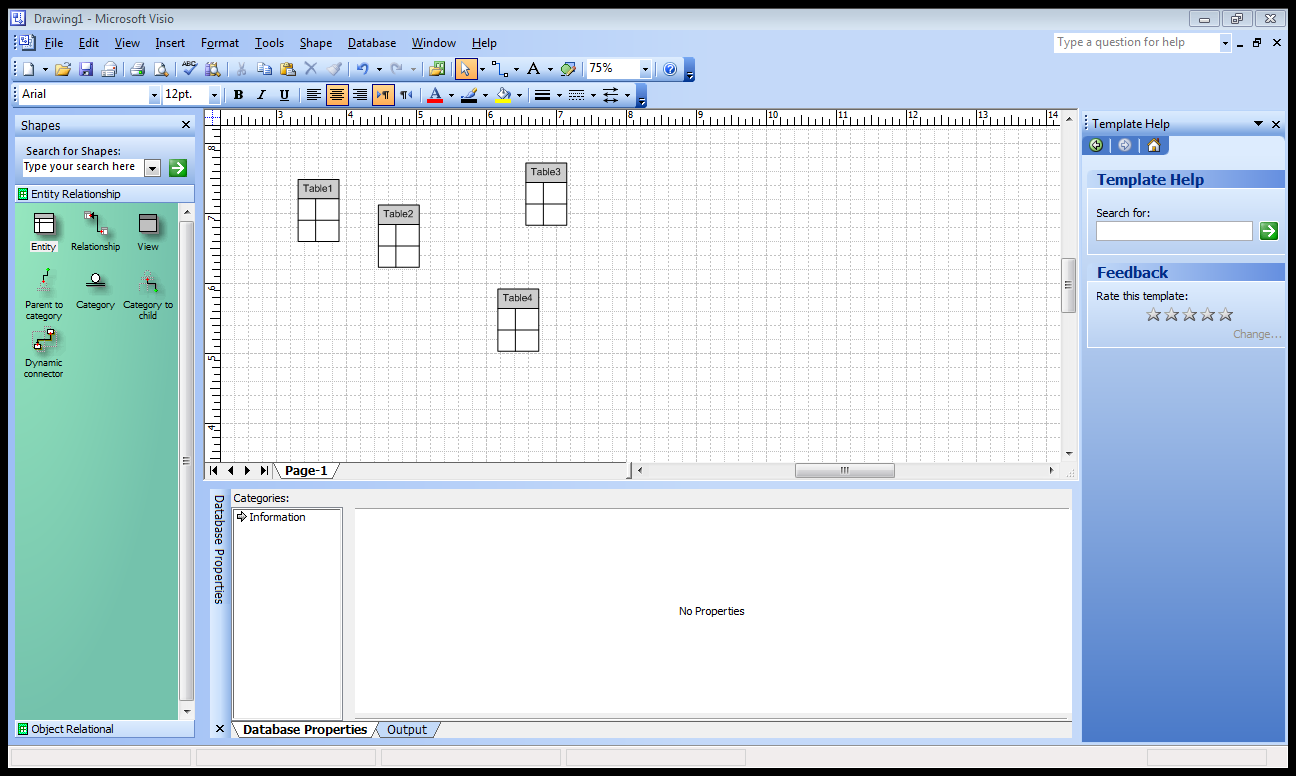
****

**Fig, 3.2** Customize size for the page

3. We click on the icon entity and then keep our right hand on the left button to draw

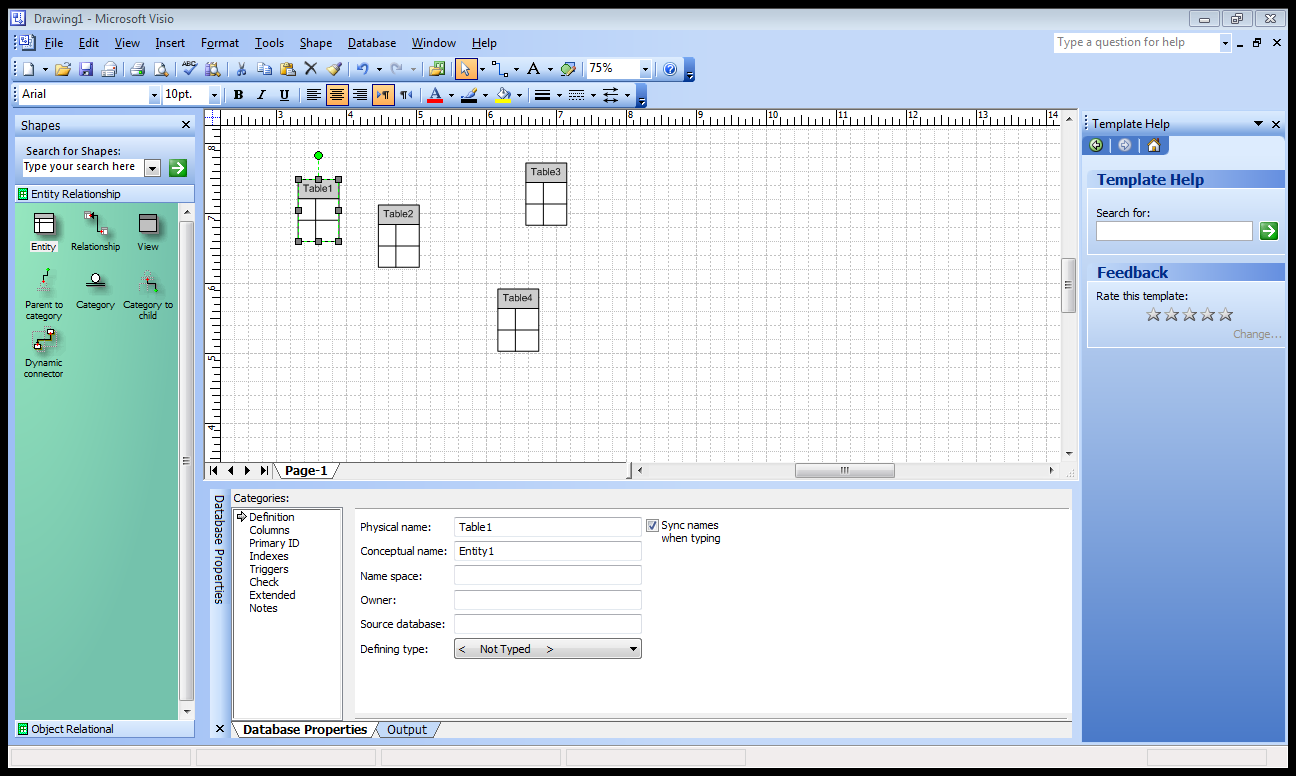
this entity into the page and then repeat this step as many entities needed.

(as shown in figure 3.3).

****

**Fig, 3.3** Drawing entities

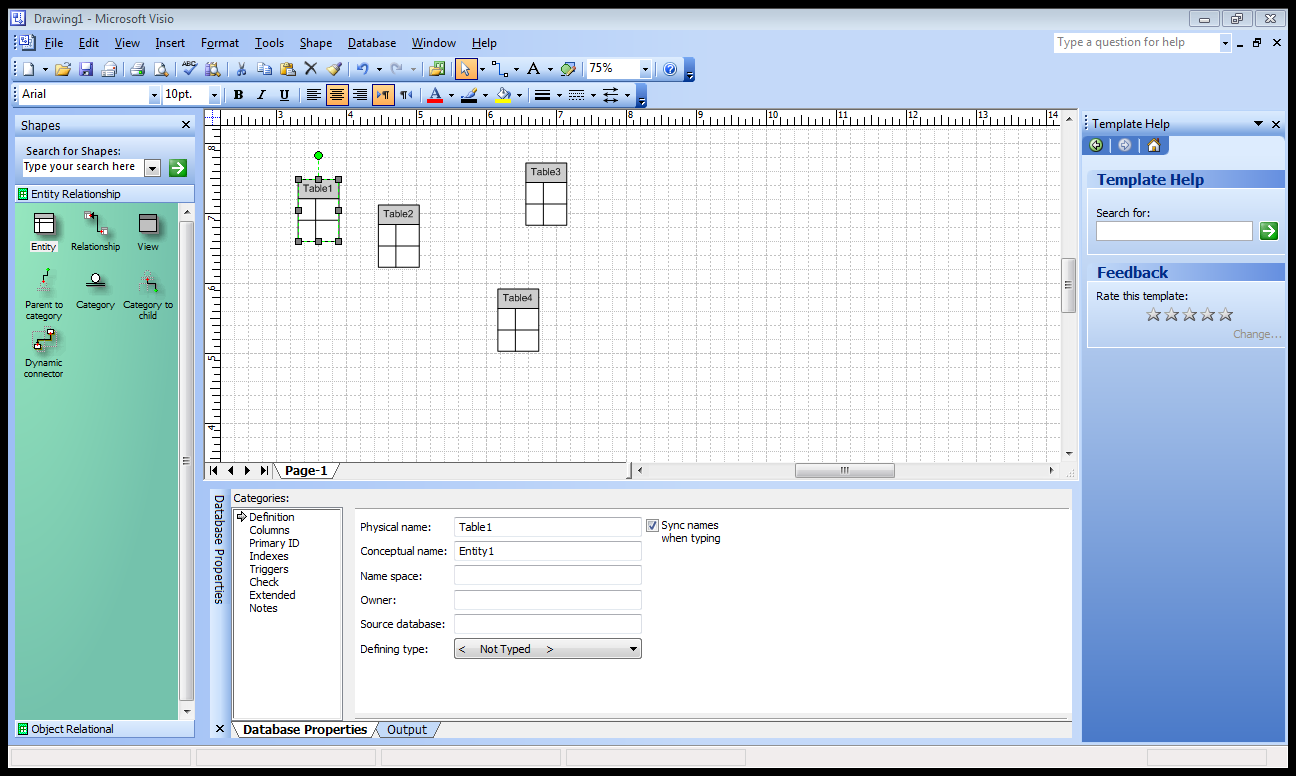
4. Click on each entity separately and right its attributes and definition.

(as shown in figure 3.4). 

**Fig, 3.4** Create entities

1. Click on definition from categories below to right the entity’s name.

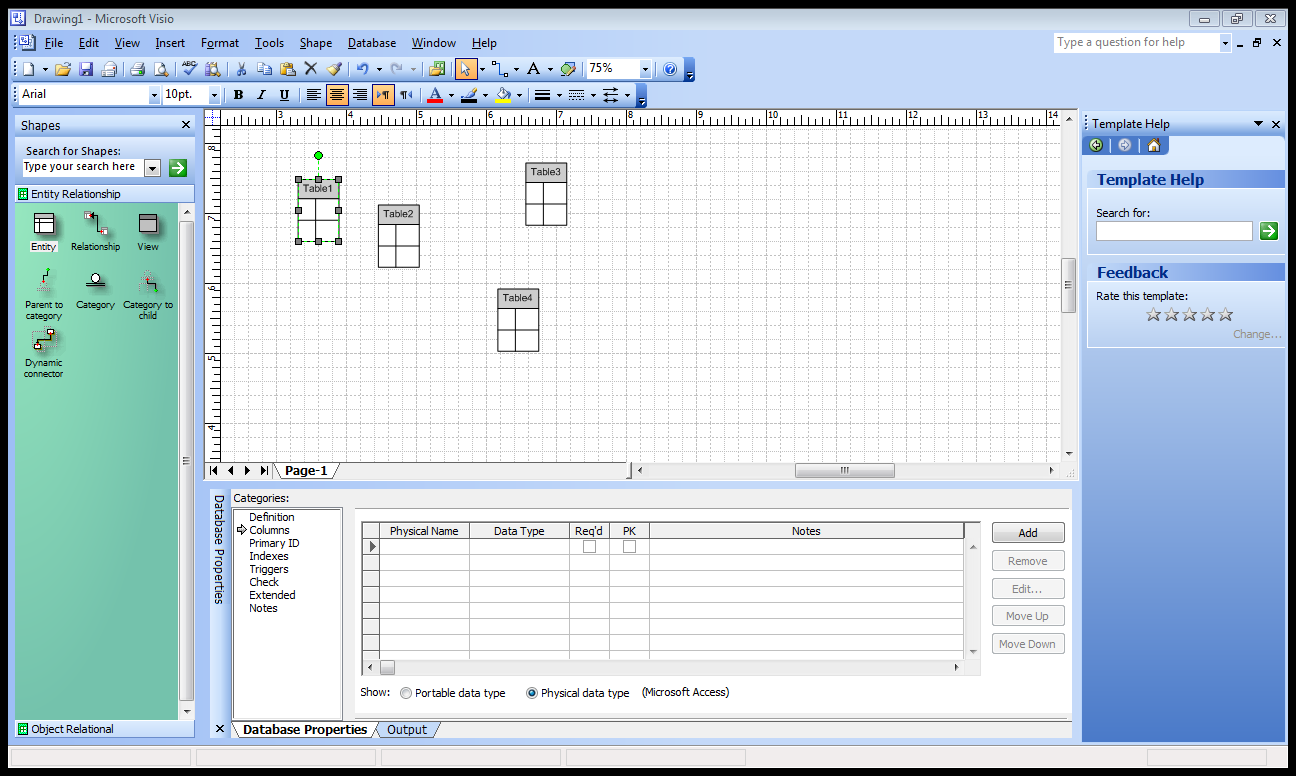
(as shown in figure 3.5).



**Fig, 3.5** The definition of entities

6. Click on columns from categories below to right the entity fields, field type, field

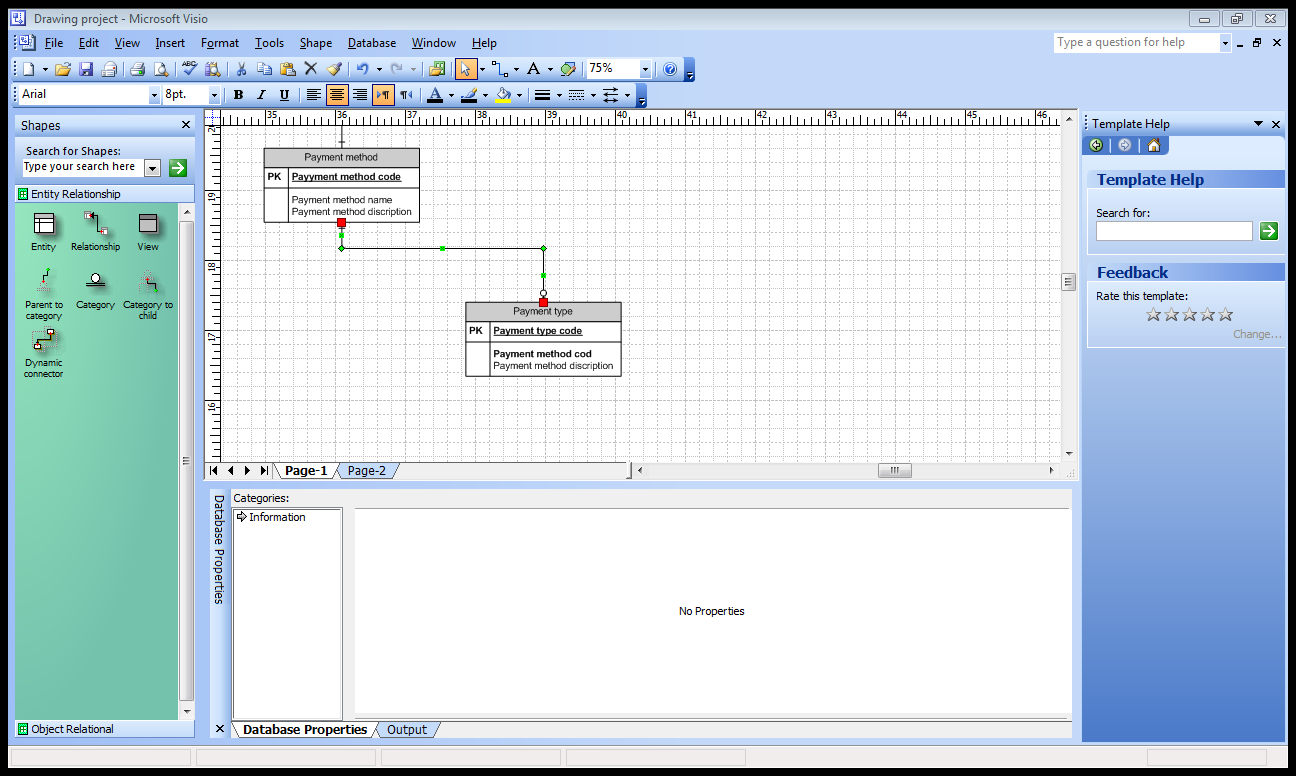
requiration and if the field is the primary key or not**.** (as shown in figure 3.6).

****

**Fig, 3.6** Definition of attributes

7. Connect the entities with icon dynamic connecter like this connection below

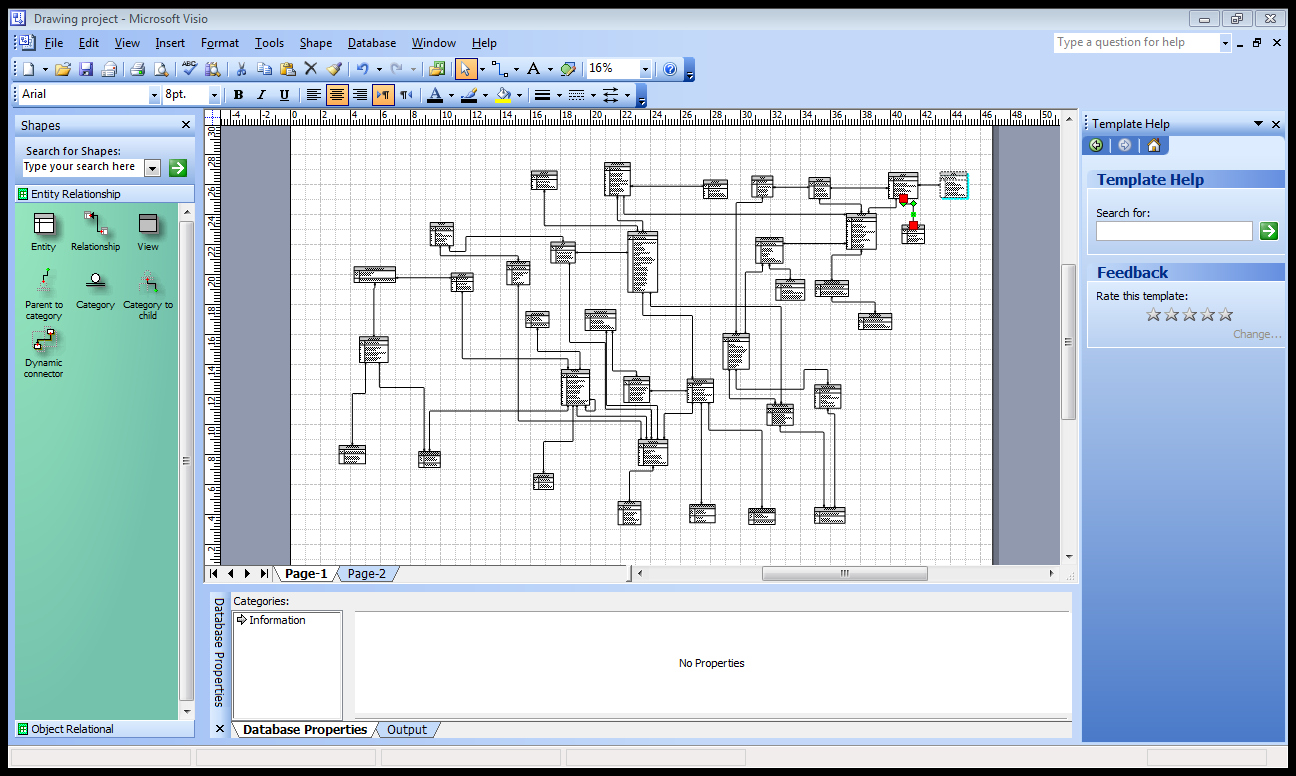
Between the student type and student address. (as shown in figure 3.7).

****

**Fig, 3.7** Connect the entities with icon dynamic

8. Repeat the last step with all the entities related together until it’s all connected and

Became one system like it’s shown below. (as shown in figure 3.8).



**Fig, 3.8** The final form

**CHAPTER FOUR**

**PROPOSED DATABASE MODEL**

**4.1 Building an Entity - Relationship Diagram (ERD)**

Entity – Relationship Diagram are used to identify the data that must be captured, stored and retrieved in order to support the business activities performed by an organization; and to identify the data required to derive and report on the performance measures that an organization should be monitoring. The components of ERD are entities, attributes and relationships.

**4.2 Proposed Database Model For First Case Study**

The database model proposed in this project for students management system is shown in Figure 4.1. It consists of nine main entities includes

1. Students
2. Staff
3. Courses
4. Semester
5. Exam result
6. Teacher
7. Address
8. Student activities
9. Classes

The attributes of each entity and the relationship between them will be described in the next section. Every entity stated above has its attributes. These are the smallest unites of data that can be described in a meaningful manner.



**Fig, 4.1** Proposed database model ER-diagram for students relations management system.

**4.3 Database Model Entity**

Entities are named to describe the data. They are normally presented as noun or noun expression. Every entity stated above has its predefined attributes. Each attribute has some characteristics need to be defined by the developer. The following subsection will demonstrate the major entities proposed in this data model.

**4.3.1 Student entity**

The student entity consists of eleven attribute (as shown in Table 4.1). Six of them are text type and only five are numerical type. The student ID attribute is adopted as the primary key of the student entity.

**Table 4.1** Student entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Student ID | 1 |
| FK | 20 | Text | Activity ID | 2 |
| FK | 8 | Numerical | Semester ID | 3 |
| FK | 20 | Text | Roster ID | 4 |
| FK | 20 | Numerical | Address ID | 5 |
| FK | 20 | Text | Forms ID | 6 |
| FK | 10 | Numerical | Payment ID | 7 |
|  | 8 | Numerical | Student number | 8 |
|  | 10 | Text | First name | 9 |
|  | 10 | Text | Middle name | 10 |
|  | 10 | Text | Last name | 11 |

**4.3.2 Staff entity**

The staff consists of six attributes three of them considered as text and three are numerical (as shown in Table 4.2). The staff ID is adopted as the primary key.

**Table 4.2** Staff entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Staff ID | 1 |
| FK | 20 | Text | Salutation ID | 2 |
| FK | 20 | Numerical | Staff roll code | 3 |
|  | 10 | Text | First name | 4 |
|  | 10 | Text | Last name | 5 |
|  | 20 | Numerical | Mobile number | 6 |

**4.3.3 Address entity**

The address entity consists of seven attributes which seven of them are numerical (as shown in Table 4.3). The Address ID is adopted as the primary key.

**Table 4.3** Address entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Address ID | 1 |
| FK | 20 | Numerical | Student ID | 2 |
| FK | 20 | Numerical | Teacher ID | 3 |
| FK | 20 | Numerical | Staff ID | 4 |
| FK | 20 | Numerical | Landlord ID | 5 |
|  | 20 | Numerical | Address line 1 | 6 |
|  | 20 | Numerical | Address line 2 | 7 |

**4.3.4 Semester entity**

The semester entity has four attributes two are date type and the other one are text type and the one numerical type (as shown in Table 4.4). The semester ID is adopted to be as the primary key.

**Table 4.4** Semester entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Semester ID | 1 |
|  | 8 | Text | Semester name | 2 |
|  | 8 | Date/Time | Semester star date | 3 |
|  | 8 | Date/Time | Semester end date | 4 |

**4.3.5 Exam result entity**

The exam result entity has two attributes one of are text type and one numerical type and the exam result -is adopted to be the primary key (as shown in Table 4.5).

**Table 4.5** Exam result entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Exam result ID | 1 |
|  | 9 | Text | Grade | 2 |

**4.3.6 Courses entity**

The courses entity has four attributes two of them are numerical type and two is text type. The course ID is adopted as the primary key (as shown in Table 4.6).

**Table 4.6** courses attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Field**  **No.** |
| PK | 20 | Numerical | Course ID | 1 |
| FK | 8 | numerical | Level code | 2 |
|  | 20 | Text | Course name | 3 |
|  | 8 | Text | E.g. French | 4 |

**4.3.7 Teachers entity**

The teacher's entity has four attributes three of them are text type and the one is numerical type. The teachers ID is adopted as the primary key (as shown in Table 4.7).

**Table 4.7** Teachers entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Teacher ID | 1 |
|  | 10 | Text | Frist name | 2 |
|  | 10 | Text | Last name | 3 |
|  | 250 | Text | Other details | 4 |

**4.3.8 Classes entity**

The classes' entity has five attributes: four of them are considered numerical type and one are considered a text type. The class ID is adopted as the primary key (as shown in Table 4.8).

**Table 4.8** Classes entity attribute

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Class ID | 1 |
|  | 3 | Numerical | Room ID | 2 |
|  | 8 | Numerical | Course ID | 3 |
|  | 20 | Numerical | Teacher ID | 4 |
|  | 20 | Text | Class description | 5 |

**4.3.9 Student activities entity**

The student activities entity has six attributes: four of them are considered numerical type, other one of them are text type and the last one is considered date type. The student and activity ID is adopted as the primary key (as shown in Table 4.9).

**Table 4.9** Student activities entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Student ID | 1 |
| FK | 20 | Text | Activity ID | 2 |
|  | 8 | Date/Time | Date from | 3 |
|  | 20 | Numerical | Attainment level code | 4 |
|  | 20 | Numerical | Payment status code | 5 |
|  | 20 | Numerical | Activity cost due | 6 |

**4.4 Database Entity Relationships**

Frequently, a meaningful relationship exists between two different types of entity. There are three types of relationship; which can exist between two different entities, which are one-to-one relationships, one-to-many relationships and many-to-many relationships. One–to-many relationships take place when a single occurrence of an entity is related to just one occurrence of a second entity. One-to-many relationships take place when a single occurrence of an entity is related to many occurrences of a second entity.

Many-to-many relationships take place when many occurrences of an entity are related to many occurrences of a second entity. Many-to-many relationship conceals a hidden entity. For this reason, many-to-many relationships are eliminated by identifying and adding the hidden entity. The new entity is related to the two original entities by a pair of one-to-many relationships.

Relationships are normally present as verb or verb expressions. it must be closely examining each relationship to ensure that it is true representation of a relationship that exists in the real world.

The cardinality and participation constraints for each relationship need to be identified. First, consider the cardinality ratio whether one-to-one relationship, one-to-many relationship or many relationship. Then, the participation constraints are considered whether it is mandatory or optional. Optional participation in a relationship is shown by a circle placed on the relationship line next to the entity that optionally participated in the relationship. While mandatory participation is shown in a similar manner by placing a bar crossing the relationship line.

**4.4.1 Student relationship**

As shown in Figure 4.2, the student relationship with student classes is one-to-many. As a given student can have many student classes, but a given student classes is associated with one student. The student relationship with the semester is also one-to-many relationship. A given student can do or have many semesters but a certain semester can only be done by only one student. The relationship of the student entity is one-to-many with the roster, student address and exam result, payment, student course registration, student activity. The student relationship with courses is many-to-one.



**Fig, 4.2** Student relationship

**4.4.2 Staff relationship**

The staff relationship with scheduled activities is one-to-many. A given staff can have many scheduled activities, but a given scheduled activities is associated with one staff as shown in Figure 4.3. The staff entity has also relationship with activities run by staff and year group.



**Fig, 4.3** Staff relationship

**4.4.3 Address relationship**

The address relationship with the staff, teachers one-to-many (as shown in Figure 4.4)

The addresses relationship with property owner is many-to-one.

 **Fig, 4.4** Address relationship

**4.4.4 Exam result and semester relationship**

The exam result relationship with the exam, result category is one-to-many as shown in Figure 4.5. The semester relationship with exam result is one-to-many.

**Fig, 4.5** Exam result and semester relationship



**4.4.5 Student activities relationships**

The student activities relationship with the activities, scheduled activities is many-to-one as shown in Figure 4.6.



**Fig, 4.6** student activities relationship

**4.4.6 Classes relationship**

The classes relationship with the teachers, courses is many-to-one (as shown in Figure 4.7). The classes relationship with rooms, payments is one-to-many. The Courses type relationship is many-to-one with level student.



**Fig, 4.7** Classes relationship

**4.5 Proposed Database Model Second Case Study**

The database model proposed in this project for library management system is (shown in Figure 4.8). It consists of eight main entities includes:

1. Books
2. Members
3. Assets
4. Pupils
5. Reference
6. Types
7. Reservations
8. Libraries



**Fig, 4.8** Proposed database model ER-diagram for library relations management system.

**4.6 Database Model Entity**

**4.6.1 Books entity**

The book consists of nine attributes: two of them are considered as numerical; six of them are text and last one is considered as date type (shown in Table 4.10). The book and genre code, ISBN ID are adopted as the primary key.

**Table 4.10** Books entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Filed**  **No.** |
| PK | 20 | Text | Book ID | 1 |
| PK | 15 | Numerical | Genre code | 2 |
| PK | 25 | Numerical | ISBN | 3 |
|  | 15 | Text | Reference subject | 4 |
|  | 15 | Text | Book title | 5 |
|  | 15 | Numerical | Book price | 6 |
|  | 25 | Text | Data of publication | 7 |
|  | 15 | Text | Other first name | 8 |
|  | 25 | Numerical | Number of publication | 9 |

**4.6.2 Members entity**

The members entity has seven attributes: one of them is considered to be numerical type and six of them are text type (shown in Table 4.11). The member ID is adopted as the primary key.

**Table 4.11** Members entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Filed**  **no** |
| PK | 20 | Numerical | Member ID | 1 |
| FK | 20 | Numerical | Member address ID | 2 |
|  | 10 | Text | Member first name | 3 |
|  | 10 | Text | Member last name | 4 |
|  | 10 | Numerical | Phone number | 5 |
|  | 20 | Text | Email address | 6 |
|  | 250 | Text | Other name details | 7 |

**4.6.3 Assets entity**

The assets entity has six attributes: one of them is considered to be numeric type and the other five are text type. The asset and asset type code ID are adopted as the primary key (as shown in Table 4.12).

**Table 4.12** Assets entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Filed**  **No** |
| PK | 20 | Text | Assets | 1 |
| PK | 20 | Numerical | Assets type code | 2 |
| FK | 20 | Text | Event | 3 |
| FK | 20 | Numerical | Format type code | 4 |
|  | 20 | Text | Data of pupation | 5 |
|  | 250 | Text | Other details | 6 |

**4.6.4 Pupils entity**

The pupils entity has seven attributes: one of them are considered to be numeric type and the other six are text type. (Table 4.13 shows) attributes of pupils entity. The pupil ID is adopted as the primary key.

**Table 4.13** Pupils entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Filed**  **No** |
| PK | 20 | Numerical | Pupil ID | 1 |
|  | 10 | Text | First name | 2 |
|  | 10 | Text | Middle name | 3 |
|  | 10 | Text | Last name | 4 |
|  | 10 | Numerical | Home phone number | 5 |
|  | 20 | Text | Email address | 6 |
|  | 250 | Text | Other pupil details | 7 |

**4.6.5 References entity**

The References entity has six attributes: one of them is considered to be numerical type and other five are text type. The reference ID is adopted as the primary key (shown in Table 4.14).

**Table 4.14** References entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Reference ID | 1 |
|  | 20 | Numerical | Call number | 2 |
|  | 20 | Text | Ref title | 3 |
|  | 20 | Text | Data of publication | 4 |
|  | 20 | Text | Place of publication | 5 |
|  | 250 | Text | Other ref details | 6 |

**4.6.6 Types entity**

The Types entity has four attributes; one of them is considered to be numerical type and the others are text type. The library ID is adopted as the primary key (Table 4.15).

**Table 4.15** Types entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No** |
| PK | 20 | Text | Library ID | 1 |
|  | 20 | Text | Category | 2 |
|  | 20 | Text | Film ID | 3 |
|  | 20 | Text | Type details | 4 |

**4.6.7 Reservations entity**

The Reservations entity has five attributes: two of them are considered to be numerical type and others are type text. The Reservations ID is adopted as the primary key (as shown in Table 4.16).

**Table 4.16** Reservation entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No** |
| PK | 20 | Numerical | Reservations ID | 1 |
| FK | 20 | Numerical | Asset ID | 2 |
| FK | 20 | Numerical | Member ID | 3 |
|  | 20 | Text | Data reserved | 4 |
|  | 20 | Text | Data located | 5 |

**4.6.8 Libraries entity**

The libraries entity has four attributes: one of them is considered to be numerical type and the other three are text type. The library and address ID are adopted as the primary key (as shown in Table 4.17).

**Table 4.17** Libraries entity attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No** |
| PK | 20 | Text | Library ID | 1 |
| PK | 20 | Numerical | Address ID | 2 |
|  | 20 | Text | Library name | 3 |
|  | 20 | Text | Library details | 4 |

**4.7 Database Entity Relationships**

**4.7.1 Books relationship**

The books relationship with member request, book publisher is many-to-one (as shown in figure 4.9). The books relationship with reference, book at library, assigned books, books by category, book out on loan is one-to-many.



**Fig, 4.9** Books relationship

**4.7.2 Members relationship**

The members relationship with the member request, reservations, assets in circulation is one-to-many (as shown in figure 4.10). The members relationship with addresses is many-to-one.



**Fig, 4.10** Members relationship

**4.7.3 Assets relationship**

The assets relationship with the event, format type, reference type, assets in circulation is many-to-one (as shown in figure 4.11). The assets relationship with reservation is one-to-many.



**Fig, 4.11** Assets relationship

**4.7.4 pupils relationship**

The pupils relationship with the book out on loan, pupils on courses is one-to-many as shown in figure 4.12.



**Fig, 4.12** Pupils relationship

**4.7.5 References relationship**

The references relationship with the books, topic, reference type is many-to-one as shown in figure 4.13.



**Fig, 4.13** References relationship

**4.7.6 Types relationship**

The type's relationship with the reference type, film, type categories is many-to-one as shown in figure 4.14.



**Fig, 4.14** Types relationship

**4.7.7 Reservations relationship**

The reservation relationship with the members, and assets is many-to-one (as shown in figure 4.15).



**Fig, 4.15** Reservation relationship

**4.7.8 Libraries relationship**

The libraries relationship with the music at libraries is one-to-many (as shown in Figure 4.16). The libraries relationship with address is many-to-one.

**Fig, 4.16** Libraries relationship



**CHAPTER FIVE**

**CONCLUSION AND RECOMMENDATIONS**

**5.1 Conclusion**

This project presented a database design model for student and library relationships management system. Seventeen main entities are considered in this data base model. These entities are students, staff, courses, semester, exam result, teacher, address, student activities, classes, books, members, assets, pupils, reference, types, reservation, and libraries.

* The main entities were divided to seventeen sub-entities from which we managed to build the system.
* The system was successfully integrated by linking primary and foreign keys relations; this will help in making the system programmable.
* The proposed database model in this project is considered software independent, which can be implemented to create any student and library relationship management system (S&LRMS. (
* The proposed database model in this project is a multifunctional system; the reason behind this can be attributed to the integration between the different entities such as the integration between orders and invoices. Such interaction would help in processing student and library requirements, however, from the implementation point of view; this software could be seen as analytical system.
* The proposed database model is one-to-many system, it is one major advantage, in which it can facilitate and promote the inter-relationships within entities.
  1. **Recommendation for Future Work**

The following recommendation may improve the development of the persuaded data model.

* The data should be more wide and comprehensive so it is highly recommended to working data entity to the model.
* The system presented should be implemented on more than one program for example Microsoft Access and Microsoft Visio to compare the results and test the efficiency of the proposed data.

**REFERENCES**

1. Student & library <http://www.databaseanswers.org/index.htm> [18-4-2014]
2. database management system (DBMS) definition . [2000] [http://searchsqlserver.techtarget.com/definition/database-management-system [11-10-2015](http://searchsqlserver.techtarget.com/definition/database-management-system%20%5b11-10-2015)]
3. Gay dash A (1988). Effective database management. USA: prentice hall, Inc.
4. Connolly T.M and beg C.E (1999). Database System :Approach to Design Implementation and Management USA : Addison-Wesley.

APPENDICES

**1. Student classes entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Student class ID | 1 |
| FK | 20 | Numerical | Class ID | 2 |
| FK | 20 | Numerical | Status code | 3 |
| FK | 20 | Numerical | Student ID | 4 |
|  | 20 | Text | Attendance from date | 5 |
|  | 25 | Numerical | Skill score | 6 |
|  | 25 | Text | Retest status | 7 |

**2. Property owner entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 25 | Text | Landlord ID | 1 |
|  | 25 | Text | Landlord name | 2 |
|  | 25 | Date/Time | Date first rental | 3 |
|  | 250 | Text | Other details | 4 |

**3. Student address entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Student address | 1 |
| FK | 20 | Numerical | Address type code | 2 |
| FK | 20 | Numerical | Student ID | 3 |
| FK | 20 | Numerical | Address ID | 4 |
|  | 35 | Date/Time | Date address from | 5 |
|  | 35 | Date/Time | Date address to | 6 |
|  | 25 | Text | Monthly rental | 7 |
|  | 250 | Text | Other details | 8 |

**4. Payment entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Payment ID | 1 |
| FK | 20 | Numerical | Student ID | 2 |
| FK | 20 | Text | Student payment method ID | 3 |
|  | 20 | Numerical | Current balance | 4 |
|  | 20 | Numerical | Amount due | 5 |
|  | 20 | Numerical | Amount paid | 6 |

**5. Rooms entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Filed name** | **Field**  **No.** |
| PK | 20 | Numerical | Room ID | 1 |
|  | 20 | Text | Location | 2 |
|  | 20 | Text | Description | 3 |

**6. Exam entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 25 | Text | Exam ID | 1 |
|  | 25 | Time/Date | Exam date | 2 |

**7. Result category entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Result category ID | 1 |
|  | 2 | Numerical | Result mark high | 2 |
|  | 2 | Numerical | Result mark low | 3 |
|  | 2 | Numerical | Result pass ect | 4 |

**8**. **Scheduled activities entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Activity ID | 1 |
| FK | 20 | Numerical | Academic year code | 2 |
| FK | 25 | Numerical | Led by staff ID | 3 |
|  | 8 | Date/Time | Star date | 4 |
|  | 8 | Date/Time | End date | 5 |
|  | 20 | Text | Contact details | 6 |
|  | 250 | Text | Other details | 7 |

**9. Roster entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Roster ID | 1 |
| FK | 20 | Numerical | Class ID | 2 |
| FK | 20 | Numerical | Student ID | 3 |
| FK | 20 | Numerical | Period ID | 4 |

**10. Period entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Period ID | 1 |
|  | 10 | Text | Period name | 2 |

**11. Student payment methods entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Student payment method ID | 1 |
| FK | 20 | Numerical | Payment method code | 2 |
| FK | 20 | Text | Student ID | 3 |
|  | 20 | Text | Bank details | 4 |
|  | 20 | Numerical | Card details | 5 |

**12. Student course attendance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 8 | Date/Time | Date of attendance | 1 |
| FK | 20 | Numerical | Student ID | 2 |
| FK | 20 | Numerical | Course ID | 3 |

**13. Year group entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Year group code | 1 |
| FK | 20 | Numerical | Year group staff code | 2 |
| FK | 20 | Text | First from in group | 3 |
| FK | 20 | Text | Second from in group | 4 |

**14. Activities run by staff entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Activity ID | 1 |
| FK | 20 | Numerical | Academic year code | 2 |
| FK | 20 | Numerical | Staff ID | 3 |

**15. Student course registration entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Student ID | 1 |
| FK | 20 | Numerical | Class ID | 2 |
| FK | 8 | Date/Time | Date of registration | 3 |
|  | 8 | Date/Time | Date of first class | 4 |
|  | 8 | Date/Time | Date of last class | 5 |

**16. Level student entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Level code | 1 |
|  | 20 | Text | Level name | 2 |

**17. Ref payment methods entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Ref payment method code | 1 |
|  | 20 | Numerical | Payment method description | 2 |
|  | 20 | Numerical | E.g. credit card | 3 |

**18. Activities entity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Activity ID | 1 |
|  | 20 | Text | Activity name | 2 |
|  | 20 | Numerical | Activity cost | 3 |
|  | 20 | Text | Activity description | 4 |
|  | 250 | Text | Other details | 5 |

**19. Music at libraries**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Sheet music ID | 1 |
| PK | 20 | Text | Library ID | 2 |
|  | 20 | Text | Gantry in stock | 3 |

**20. Sheet music1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Sheet music ID | 1 |
|  | 20 | Numerical | Gere code | 2 |
|  | 20 | Numerical | ISBN | 3 |

**21. Event**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Event ID | 1 |
|  | 20 | Text | Event name description | 2 |

**22. Assets in circulation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Assets ID | 1 |
|  | 20 | Numerical | Member ID | 2 |

**23. Member requests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Requests ID | 1 |
| PK | 20 | Numerical | Member ID | 2 |
|  | 20 | Text | Sheet music ID | 3 |
|  | 8 | Date/time | Date request | 4 |
|  | 8 | Date/time | Date located | 5 |
|  | 250 | Text | Other request details | 6 |

**24. Recommended event music**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Event ID | 1 |
| PK | 20 | Text | Sheet music | 2 |

**25. Recording devices**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Recording ID | 1 |
| PK | 20 | Numerical | Format type code | 2 |
|  | 20 | Text | Device name | 3 |

**26. Reference type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Reference type | 1 |
|  | 20 | Numerical | Reference code | 2 |
|  | 20 | Numerical | Assets code | 3 |
|  | 20 | Text | Type details | 4 |
|  | 20 | Text | Ref type description | 5 |

**27. Publishing devices**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Device ID | 1 |
| PK | 20 | Numerical | Format type code | 2 |
|  | 20 | Text | Device name | 3 |

**28. Format type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Format type code | 1 |
| PK | 20 | Text | Format type description | 2 |

**29. Reference**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Reference ID | 1 |
| PK | 20 | Numerical | Call number | 2 |
|  | 20 | Text | Ref title | 3 |
|  | 20 | Date/time | Date of publication | 4 |
|  | 20 | Text | Place of publication | 5 |
|  | 250 | Text | Other ref details | 6 |

**30. Production companies**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Pro company ID | 1 |
|  | 250 | Text | Other details | 2 |

**31. Film**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Film ID | 1 |
|  | 20 | Text | Prod company ID | 2 |
|  | 15 | Text | Film name | 3 |
|  | 35 | Text | Film description | 4 |

**32. Books out on loan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Book borrowing ID | 1 |
|  | 20 | Numerical | Pupil ID | 2 |
|  | 20 | Numerical | Borrower teacher ID | 3 |
|  | 8 | Date/time | Date issued | 4 |
|  | 35 | Text | ISBN | 5 |
|  | 20 | Numerical | Fine paid in | 6 |
|  | 8 | Date/time | Date due for term | 7 |
|  | 8 | Date/time | Date returned | 8 |

**33. Categories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Category ID | 1 |
|  | 20 | Text | Category name | 2 |
|  | 20 | Text | Category short name | 3 |
|  | 20 | Text | Category description | 4 |

**34. Book publisher**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | book publisher ID | 1 |
|  | 20 | Text | publisher ID | 2 |
|  | 20 | Text | publisher name | 3 |
|  | 20 | Text | Publisher details | 4 |

**35. Assigned book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Course code | 1 |
|  | 35 | Numerical | ISBN | 2 |

**36. Type categories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Category code | 1 |
|  | 20 | Text | Category description | 2 |

**37. Pupils on course**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Numerical | Pupil ID | 1 |
| PK | 20 | Numerical | Course code | 2 |
|  | 8 | Date/ time | Date from | 3 |
|  | 8 | Date/time | Date to | 4 |

**38. Book by category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | category ID | 1 |
|  | 20 | Numerical | ISBN | 2 |
|  | 20 | Text | Category short name | 3 |

**39. Book at library**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Length** | **Type** | **Field name** | **Field**  **No.** |
| PK | 20 | Text | Book at library | 1 |
|  | 20 | Text | Library ID | 2 |