

MES College of Engineering Pune-01**Department of Computer Engineering**

Name of Student:	Class:
Semester/Year:	Roll No:
Date of Performance:	Date of Submission:
Examined By:	Experiment No: A-09,B-04,C-01

ASSIGNMENT - A-09,B-04,C-01**AIM: Database Connectivity & Mini Project**

A-09: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

B-04: Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

C-01: Using the database concepts covered in Group A and Group B, develop an application with following details:

1. Follow the same problem statement decided in Assignment-1 of Group A.
2. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation.
3. Develop application considering:

- Front End: Java/Perl/PHP/Python/Ruby/.net/any other language
- Back End : MongoDB/ MySQL/Oracle

4. Test and validate application using Manual/Automation testing.
5. Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development Life Cycle:

- Title of the Project, Abstract, Introduction
- Software Requirement Specification
- Conceptual Design using ER features, Relational Model in appropriate Normalized form
- Graphical User Interface, Source Code
- Testing document
- Conclusion

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To develop basic Database administration skill.
- To develop the ability to handle databases of varying complexities.
- To use advanced database Programming concepts.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Front End: Java/Perl/PHP/Python/Ruby/.net/any other language
- Back End : MongoDB/ MySQL/Oracle

THEORY:

Data Models:

A Data Model is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constraints etc.

Data Models Types:

A. Record-based Data Models

- The Relational Model
- The Network Model
- The Hierarchical Model

B. Object-based Data Models

- The E-R Model
- The Object-Oriented Model

C. Physical Data Models

A. Relational Model:

Relational model stores data in the form of tables. This concept proposed by Dr. E.F. Codd, a researcher of IBM in the year 1960s. The relational model consists of three major components:

1. The set of relations and set of domains that defines the way data can be represented (data structure).
2. Integrity rules that define the procedure to protect the data (data integrity).
3. The operations that can be performed on data (data manipulation).

Basic Terminology used in Relational Model

The figure shows a relation with the. Formal names of the basic components marked the entire structure is, as we have said, a relation.

Attributes	Emp_Code	Name	Year
Tuples	21130	Amar Jain	1
	30745	Kuldeep	3
	41894	Manoj	2
	51207	Rita bajaj	6

Tuples of a Relation - Each row of data is a tuple.

Cardinality of a relation: The number of tuples in a relation determines its cardinality. In this case, the relation has a cardinality of 4.

Degree of a relation: Each column in the tuple is called an attribute. The number of attributes in a relation determines its degree. The relation in figure has a degree of 3.

Domains: A domain definition specifies the kind of data represented by the attribute.

Keys of a Relation

It is a set of one or more columns whose combined values are *unique* among all occurrences in a given table. A key is the relational means of specifying uniqueness. Some different types of keys are:

Primary key – It is an attribute or a set of attributes of a relation which possess the properties of uniqueness and irreducibility (No subset should be unique).

Foreign key – It is the attributes of a table, which refers to the primary key of some another table. Foreign key permit only those values, which appears in the primary key of the table to which it refers or may be null (Unknown value).

Operations in Relational Model:

The four basic operations Insert, Update, Delete and Retrieve operations.

B. ER Data Model: An Entity Relationship Model

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

E-R Model is based on –

- A. Entities and their attributes.
- B. Relationships among entities.

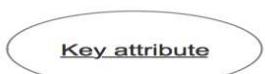
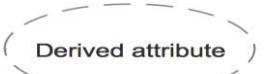
1.Entity – An entity in an ER Model is a “thing” or “object ” in the real-world having properties called attributes. An Entity set is a set of entities of the same type that share the same properties, or attributes.

2. 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.

ER diagram: An ER diagram is a pictorial representation of the information that can be captured by a database. Such a picture serves two purposes:

1. It allows database professionals to describe an overall design concisely yet accurately.
2. It can be easily transformed into the relational schema.

Entity Relationship Diagram Symbols — Chen notation

Symbol	Shape Name	Symbol Description
Entities		
	Entity	An entity is represented by a rectangle which contains the entity's name.
	Weak Entity	An entity that cannot be uniquely identified by its attributes alone. The existence of a weak entity is dependent upon another entity called the owner entity. The weak entity's identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.
	Associative Entity	An entity used in a many-to-many relationship (represents an extra table). All relationships for the associative entity should be many
Attributes		
	Attribute	In the Chen notation, each attribute is represented by an oval containing attributes name
	Key attribute	An attribute that uniquely identifies a particular entity. The name of a key attribute is underscored.
	Multivalued attribute	An attribute that can have many values (there are many distinct values entered for it in the same column of the table). Multivalued attribute is depicted by a dual oval.
	Derived attribute	An attribute whose value is calculated (derived) from other attributes. The derived attribute may or may not be physically stored in the database. In

		the Chen notation, this attribute is represented by dashed oval.
Relationships		
	Strong relationship	A relationship where entity is existence-independent of other entities and PK of Child doesn't contain PK component of Parent Entity. A strong relationship is represented by a single rhombus
	Weak (identifying) relationship	A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double rhombus.

Entity Relationship Diagram Symbols — Crow's Foot notation

Symbol	Meaning
+○	Zero or One
※	One or More
	One and only One
≥○	Zero or More
※ M:1	a one through many notation on one side of a relationship and a one and only one on the other
≥○ M:1	a zero through many notation on one side of a relationship and a one and only one on the other
※ M:1 ○+	a one through many notation on one side of a relationship and a zero or one notation on the other
≥○ M:1 ○+	a zero through many notation on one side of a relationship and a zero or one notation on the other
≥○ M:M ○≤	a zero through many on both sides of a relationship
≥○ M:M ≤	a zero through many on one side and a one through many on the other
M:M ≤	a one through many on both sides of a relationship
1:1 ○+	a one and only one notation on one side of a relationship and a zero or one on the other
1:1 +	a one and only one notation on both sides

Transform ER Diagram into Tables:

Since ER diagram gives us the good knowledge about the requirement and the mapping of the entities in it, we can easily convert them as tables and columns. i.e.; using ER diagrams one can easily create relational data model, which nothing but the logical view of the database.

There are various steps involved in converting it into tables and columns. Each type of entity, attribute and relationship in the diagram takes their own depiction here. Consider the ER diagram below and will see how it is converted into tables, columns and mappings.

The basic rule for converting the ER diagrams into tables is

- Convert all the Entities in the diagram to tables.

All the entities represented in the rectangular box in the ER diagram become independent tables in the database. In the below diagram, STUDENT, COURSE, LECTURER and SUBJECTS forms individual tables.

- All single valued attributes of an entity is converted to a column of the table

All the attributes, whose value at any instance of time is unique, are considered as columns of that table. In the STUDENT Entity, STUDENT_ID, STUDENT_NAME form the columns of STUDENT table. Similarly, LECTURER_ID, LECTURER_NAME form the columns of LECTURER table. And so on.

- Key attribute in the ER diagram becomes the Primary key of the table.

In diagram above, STUDENT_ID, LECTURER_ID, COURSE_ID and SUB_ID are the key attributes of the entities. Hence we consider them as the primary keys of respective table.

- Declare the foreign key column, if applicable.

In the diagram, attribute COURSE_ID in the STUDENT entity is from COURSE entity. Hence add COURSE_ID in the STUDENT table and assign it foreign key constraint. COURSE_ID and SUBJECT_ID in LECTURER table forms the foreign key column. Hence by declaring the foreign key constraints, mapping between the tables are established.

- Any multi-valued attributes are converted into new table.

A hobby in the Student table is a multivalued attribute. Any student can have any number of hobbies. So we cannot represent multiple values in a single column of STUDENT table. We need to store it separately, so that we can store any number of hobbies, adding/ removing / deleting hobbies should not create any redundancy or anomalies in the system. Hence we create a separate table STUD_HOBBY with STUDENT_ID and HOBBY as its columns.

We create a composite key using both the columns

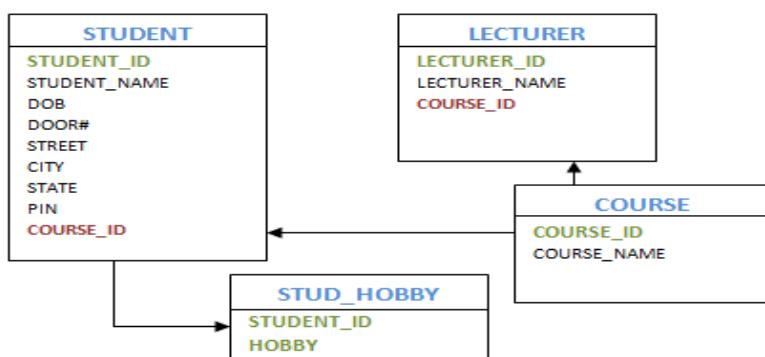
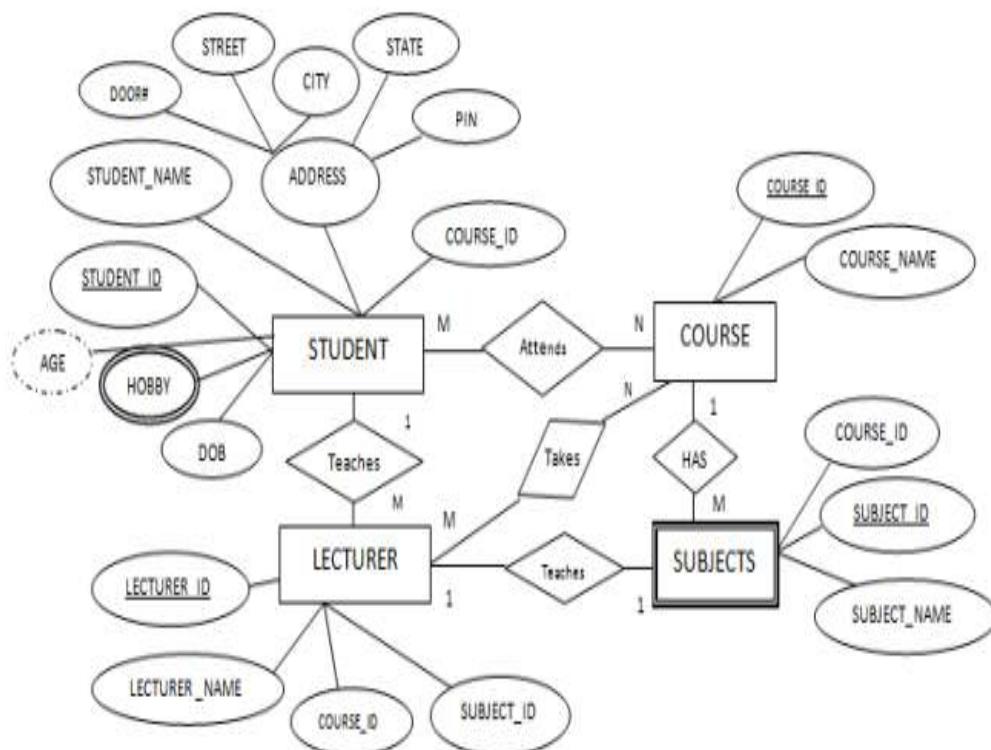
Any composite attributes are merged into same table as different columns.

In the diagram above, Student Address is a composite attribute. It has Door#, Street, City, State and Pin. These attributes are merged into STUDENT table as individual columns.

- One can ignore derived attribute, since it can be calculated at any time.

In the STUDENT table, Age can be derived at any point of time by calculating the difference between DateOfBirth and current date. Hence we need not create a column for this attribute. It reduces the duplicity in the database.

These are the very basic rules of converting ER diagram into tables and columns, and assigning the mapping between the tables. Table structure at this would be as below:



CONCLUSION: