##### Designing and Implementing a Relational DBMS on

**“Title of Mini Project”**

Submitted to the

Savitribai Phule Pune University

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

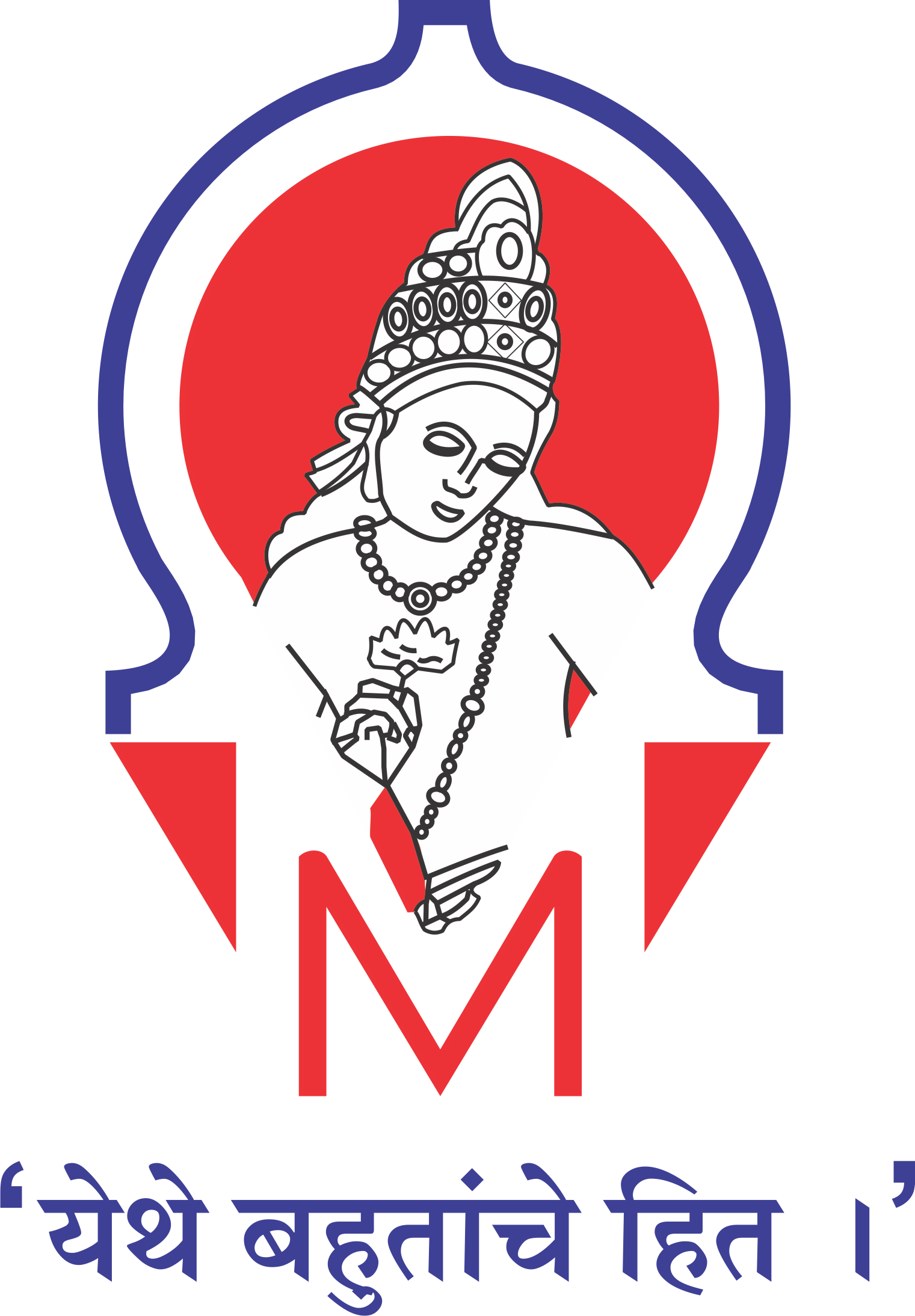
by

**FirstName LastName**

(Exam Seat No. / Roll No. & Division)

Under the guidance of

**Dr/Prof. Name of the Guide**

****

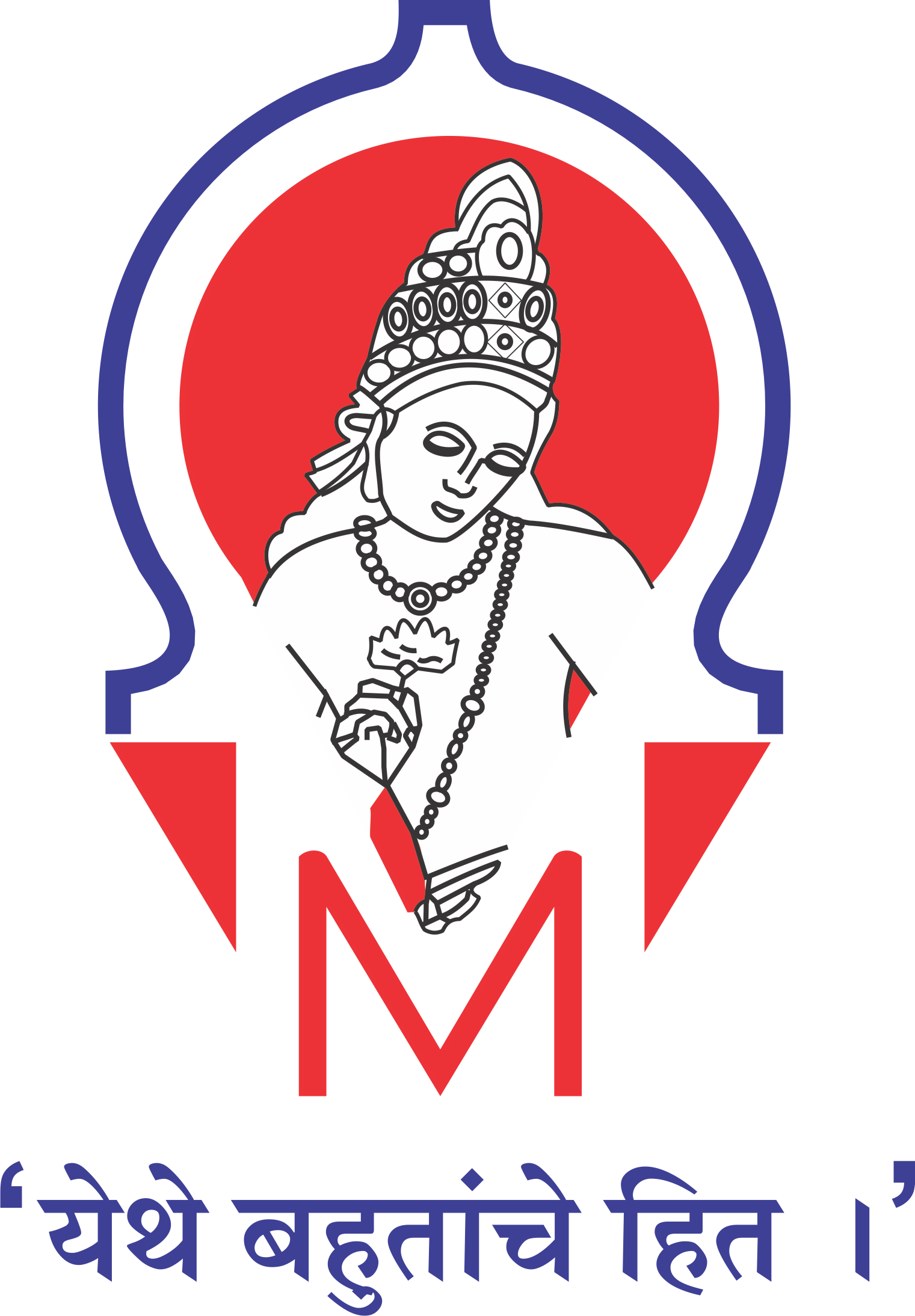
Department Of Information Technology

**Marathwada Mitra Mandal’s**

**College of Engineering**

Karvenagar, Pune - 411 052.

**2020-2021**

****

**CERTIFICATE**

This is to certify that the mini project report entitled **“Title of mini project”** being submitted by **Name of Student (Exam Seat No. / Roll No. & Division)** is a record of bonafide work carried out by him/her under the supervision and guidance of Mr. Nikhil Dhavase in partial fulfillment of the requirement for **SE (Information Technology Engineering) – 2019 course** of Savitribai Phule Pune University, Pune in the academic year 2020-2021.

Date:

Place: Pune

Guide Subject Coordinator Head of the Department

Principal

This Mini Project report has been examined by us as per the Savitribai Phule Pune University, Pune requirements at MMCOE, Pune on . . . . . . . . . . .

Internal Examiner External Examiner

**ACKNOWLEDGEMENT**

Purpose of acknowledgments page is to show appreciation to those who contributed in conducting this dissertation work / other tasks and duties related to the report writing. Therefore when writing acknowledgments page you should carefully consider everyone who helped during research process and show appreciation in the order of relevance. In this regard it is suitable to show appreciation in brief manner instead of using strong emotional phrases.

In this part of your work it is normal to use personal pronouns like “I, my, me” while in the rest of the report this articulation is not recommended. Even when acknowledging family members and friends make sure of using the wording of a relatively formal register. The list of the persons you should acknowledged, includes guide (main and second), academic staff in your department, technical staff, reviewers, companies, family and friends.

You should acknowledge all sources of funding. It’s usually speciﬁc naming the person and the type of help you received. For example, an advisor who helped you conceptualize the project, someone who helped with the actual building or procedures used to complete the project, someone who helped with computer knowledge, someone who provided raw materials for the project, etc.

(Students Name & Signature)

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **TITLE** | **Page no** |
| **1.** | Abstract | **3** |
| **2.** | Introduction | **3** |
| **3.** | Data Types | **4** |
| **4.** | Data Requirements  Entities  Attributes  Relationships- Cardinality | **4** |
| **5.** | E R Diagram | **4** |
| **6.** | Schema Diagram | **4** |
| **7** | Relational Database Design |  |
| **8.** | Creating database using MySQL | **4** |
| **9.** | Test Case Queries | **16** |
| **10.** | Conclusion | **16** |
| **11** | References |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***1. ABSTRACT***

In this project we are created one application which is easy to access user friendly.For this application we used the backend as MySQL to store the data which is used in the application and for the user interface we had used the PHP and HTML.2 kinds of people are able to use this application as manager and the customer as well. The customer is able to do the online shopping very easily from any place by reading the details of product and by seeing the product image.The manager is the only person who is able to add and remove the products from the site.

***2. INTRODUCTION***

E. F. Codd introduced the term in his seminal paper "A Relational Model of Data for Large Shared Data Banks", published in 1970. In this paper and later papers he defined what he meant by relational. One well-known definition of what constitutes a relational database system is Codd's 12 rules. However, many of the early implementations of the relational model did not conform to all of Codd's rules, so the term gradually came to describe a broader class of database systems. At a minimum, these systems: • presented the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns, can satisfy this property) • provided relational operators to manipulate the data in tabular form The first systems that were relatively faithful implementations of the relational model were from the University of Michigan; Micro DBMS (1969) and from IBM UK Scientific Centre at Peterlee; IS1 (1970–72) and its followon PRTV (1973–79). The first system sold as an RDBMS was Multics Relational Data Store, first sold in 1978. Others have been Berkeley Ingres QUEL and IBM BS12. The most popular definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules.[1] A second, theory-based school of thought argues that if a database does not implement all of Codd's rules (or the current understanding on the relational model, as expressed by Christopher J Date, Hugh Darwen and others), it is not relational. This view, shared by many theorists and other strict adherents to Codd's principles, would disqualify most DBMSs as not relational. For clarification, they often refer to some RDBMSs as Truly-Relational Database Management Systems (TRDBMS), naming others Pseudo-Relational Database Management Systems (PRDBMS).As of now, all commercial relational DBMSes employ SQL as their query language. Alternative query languages have been proposed and implemented, notably the pre-1996 implementation of Berkeley Ingres QUEL. With standardization of the SQL, both commercial and open source DBMSes have adopted some degree of standards compliance.

**2.1 Problem Statement.**

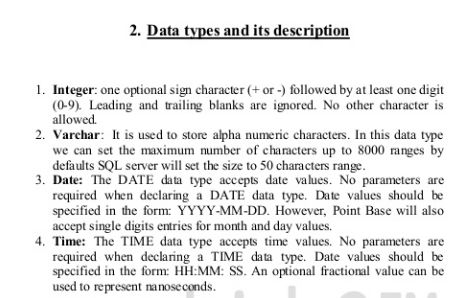
ghhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

***2.2 Motivation***

*Adv of DBMS over File Systems*

***2.3 . Objectives***

3. Data Types

******

**4. Data Modeling using ER Model**

**4.1** REQUIREMENTS COLLECTION AND ANALYSIS

We list the data requirements for the database project here, and then create its conceptual schema step-by-step as we introduce the modeling concepts of the ER model. The COMPANY database keeps track of a company’s employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the mini world—the part of the company that will be represented in the database.

The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

■ A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

■ We store each employee’s name, Social Security number,2 address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).

■ We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent’s first name, sex, birth date, and relationship to the employee.

**4.1.1 Entity Types, Entity Sets, Attributes, and Keys**

1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager\_start\_date. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.

2. An entity type PROJECT with attributes Name, Number, Location, and Controlling\_department. Both Name and Number are (separate) key attributes.

3. An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth\_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First\_name, Middle\_initial, Last\_name—or of Address.

4. An entity type DEPENDENT with attributes Employee, Dependent\_name, Sex, Birth\_date, and Relationship (to the employee).

**4.2 E-R Diagram**

An entity–relationship model (ER model) describes inter-related things of interest in a

specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database. Entity–relationship modeling was developed for database design by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously, some ER modelers show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontology. An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship

between entities is implemented by storing the primary key of one entity as a pointer or &quot;foreign key&quot; in the table of another entity There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.

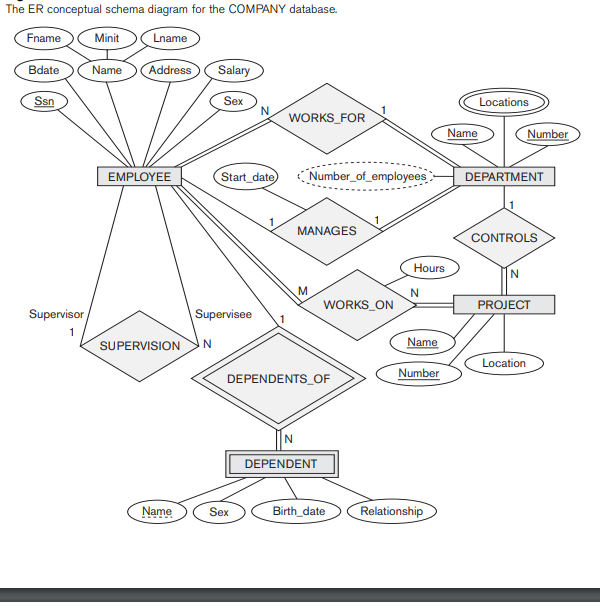
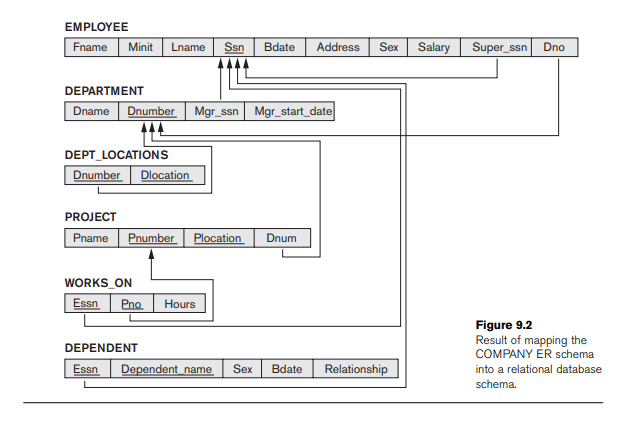


Fig 9.1 The ER conceptual schema diagram for the COMPANY database.

**4.2.2 Relational Database Design Using ER-to-Relational Mapping**

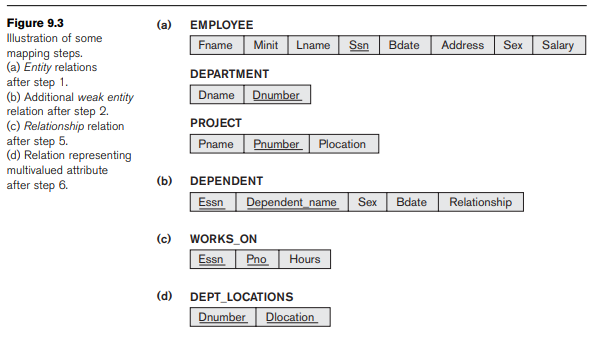
The COMPANY ER schema is shown again in Figure 9.1, and the corresponding COMPANY relational database schema is shown in Figure 9.2.

****

**ER-to-Relational Mapping Algorithm**

The relational model constraints, which include primary keys, unique keys (if any), and referential integrity constraints on the relations, will also be specified in the mapping results.

**Step 1:** Mapping of Regular Entity Types. For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R. If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation R. Knowledge about keys is also kept for indexing purposes and other types of analyses. In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.2 to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.1. The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the attributes Super\_ssn and Dno of EMPLOYEE, Mgr\_ssn and Mgr\_start\_date of DEPARTMENT, and Dnum of PROJECT. In our example, we choose Ssn, Dnumber, and Pnumber as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively. Knowledge that Dname of DEPARTMENT and Pname of PROJECT are secondary keys is kept for possible use later in the design. The relations that are created from the mapping of entity types are sometimes called entity relations because each tuple represents an entity instance. The result after this mapping step is shown in Figure 9.3(a).

****

**Step 2: Mapping of Weak Entity Types.** For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any. If there is a weak entity type E2 whose owner is also a weak entity type E1, then E1 should be mapped before E2 to determine its primary key first. In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT (see Figure 9.3(b)). We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type—as a foreign key attribute of DEPENDENT; we rename it Essn, although this is not necessary.

The primary key of the DEPENDENT relation is the combination {Essn, Dependent\_name}, because Dependent\_name (also renamed from Name in Figure 9.1) is the partial key of DEPENDENT.

**Step 3:** **Mapping of Binary 1:1 Relationship Types**. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross reference or relationship relation approach. The first approach is the most useful and should be followed unless special conditions exist, as we discuss below.

**1. Foreign key approach:** Choose one of the relations—S, say—and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

In our project, we map the 1:1 relationship type MANAGES from Figure 9.1 by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it Mgr\_ssn. We also include the simple attribute Start\_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr\_start\_date (see Figure 9.2). Note that it is possible to include the primary key of S as a foreign key in T instead. In our example, this amounts to having a foreign key attribute, say Department\_managed in the EMPLOYEE relation, but it will have a NULL value for employee tuples who do not manage a department. If only 2 percent of employees manage a department, then 98 percent of the foreign keys would be NULL in this case. Another possibility is to have foreign keys in both relations S and T redundantly, but this creates redundancy and incurs a penalty for consistency maintenance

**Step 4:** Mapping of Binary 1:N Relationship Types.

For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S. In our example, we now map the 1:N relationship types WORKS\_FOR, CONTROLS, and SUPERVISION from Figure 9.1. For WORKS\_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno. For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super\_ssn. The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation. These foreign keys are shown in Figure 9.2. An alternative approach is to use the relationship relation (cross-reference) option as in the third option for binary 1:1 relationships. We create a separate relation R whose attributes are the primary keys of S and T, which will also be foreign keys to S and T. The primary key of R is the same as the primary key of S. This option can be used if few tuples in S participate in the relationship to avoid excessive NULL values in the foreign key

**Step 5: Mapping of Binary M:N Relationship Types**. For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate relationship relation S. In our project, we map the M:N relationship type WORKS\_ON from Figure 9.1 by creating the relation WORKS\_ON in Figure 9.2. We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS\_ON and rename them Pno and Essn, respectively. We also include an attribute Hours in WORKS\_ON to represent the Hours attribute of the relationship type. The primary key of the WORKS\_ON relation is the combination of the foreign key attributes {Essn, Pno}. This relationship relation is shown in Figure 9.3(c).

**Step 6: Mapping of Multivalued Attributes.** For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components. In our example, we create a relation DEPT\_LOCATIONS (see Figure 9.3(d)). The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation. The primary key of DEPT\_LOCATIONS is the combination of {Dnumber, Dlocation}. A separate tuple will exist in DEPT\_LOCATIONS for each location that a department has.

**5. CREATING DATABASE USING MYSQL**

An SQL schema is identified by a schema name, and includes an authorization identifier to indicate the user or account who owns the schema, as well as descriptors for each element in the schema. Schema elements include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema. A schema is created via the CREATE SCHEMA statement, which can include all the schema elements’ definitions. Alternatively, the schema can be assigned a name and authorization identifier, and the elements can be defined later. The CREATE TABLE command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and any attribute constraints, such as NOT NULL. The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command (see Chapter 5). Figure 4.1 shows sample data definition statements in SQL for the COMPANY relational database schema.

CREATE TABLE EMPLOYEE ( Fname VARCHAR(15) NOT NULL, Minit CHAR, Lname VARCHAR(15) NOT NULL, Ssn CHAR(9) NOT NULL, Bdate DATE, Address VARCHAR(30), Sex CHAR, Salary DECIMAL(10,2), Super\_ssn CHAR(9), Dno INT NOT NULL, PRIMARY KEY (Ssn), FOREIGN KEY (Super\_ssn) REFERENCES EMPLOYEE(Ssn), FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE DEPARTMENT ( Dname VARCHAR(15) NOT NULL, Dnumber INT NOT NULL, Mgr\_ssn CHAR(9) NOT NULL, Mgr\_start\_date DATE, PRIMARY KEY (Dnumber), UNIQUE (Dname), FOREIGN KEY (Mgr\_ssn) REFERENCES EMPLOYEE(Ssn) );

CREATE TABLE DEPT\_LOCATIONS ( Dnumber INT NOT NULL, Dlocation VARCHAR(15) NOT NULL, PRIMARY KEY (Dnumber, Dlocation), FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE PROJECT ( Pname VARCHAR(15) NOT NULL, Pnumber INT NOT NULL, Plocation VARCHAR(15), Dnum INT NOT NULL, PRIMARY KEY (Pnumber), UNIQUE (Pname), FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE WORKS\_ON ( Essn CHAR(9) NOT NULL, Pno INT NOT NULL, Hours DECIMAL(3,1) NOT NULL, PRIMARY KEY (Essn, Pno), FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn), FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );

CREATE TABLE DEPENDENT ( Essn CHAR(9) NOT NULL, Dependent\_name VARCHAR(15) NOT NULL, Sex CHAR, Bdate DATE, Relationship VARCHAR(8), PRIMARY KEY (Essn, Dependent\_name), FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );

**Snapshots**

**6. Test Queries(Minimum 25 Queries)**

Screen Shots

**7. Conclusion**

**8. References.**