

Lecture 2:

1. E-R Modelling

Entity–Relationship model (ER model) in software engineering is an abstract way to describe a database. Describing a database usually starts with a relational database, which stores data in tables. Some of the data in these tables point to data in other tables for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is ‘has a phone number’.

1.1. Relationship

An association among entities. *e.g.*, There is a relationship between employees and department, which can be named as Works_in.

1.2. Relationship Set

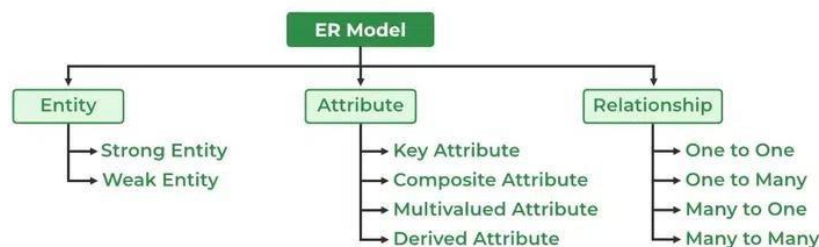
- ✓ An association of entity sets *e.g.*, Employee_Department
- ✓ A relationship instance is an association of entity instances *e.g.*, Shyam_Sales.
- ✓ Same entity set could participate in different relationship sets.
- ✓ An n -array relationship set R relates n entity sets.
- ✓ A relationship set involving three entity sets, is known as a ternary relationship

1.3. Entity

- ✓ Anything that exists and can be distinguished/ real world object which can
- ✓ be distinguished from other objects. *e.g.*, student.

1.4. Entity Set

- ✓ A group of similar entities, *e.g.*, all students.
- ✓ All entities in an entity set have the same set of attributes.
- ✓ Each entity set has a key.
- ✓ Can be mapped to a relation easily.



1.5. Weak Entity Set and Strong Entity Set

An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed a weak entity set. An entity set that has a primary key is termed a strong entity set. For a weak entity set to be meaningful, it must be associated with another entity set, called the identifying or owner entity set. Every weak entity must be associated with an identifying entity *i.e.*, the weak entity set is said to be existence dependent on the identifying entity set.

1.6. Attribute

Properties that describe an entity or we can say attributes are descriptive properties possessed by each member of an entity set and each attribute has a domain. *There are many types of attributes*

a. Composite Attribute

Attributes that can have component attribute. *e.g.*, a composite attribute name, with component attributes First_Name, Middle_Name and Last_Name.

b. Derived Attribute

The value for this type of attribute can be derived from the values of other related attributes or entities. For instance, let us say that the customer entity set has an attribute Loans_Held, which represents how many loans a customer has from the bank. We can derive the value for this attribute by counting the number of loan entities associated with that customer.

c. Descriptive Attribute

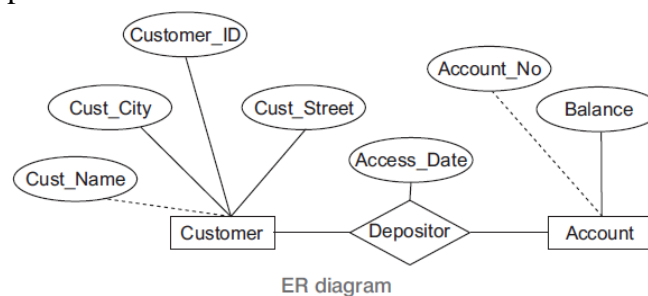
If a relationship set has also some attributes associated with it, then we link these attributes to that relationship set. *e.g.*, consider a relationship set depositor with entity sets customer and account. We could associate the attribute Access_Date to that relationship to specify the most recent date on which a customer accessed an account.












d. Single Valued Attribute

Attribute which has only one value, *e.g.*, the Employee_Number attribute for a specific Employee_entity refers to only one employee number.

e. Multi Valued Attribute

Attributes which can have 0, 1 or more than 1 values. An employee entity set with the attribute Phone_Number. An employee may have zero, one or several phone numbers and different employees may have different numbers of phones.



Notations/Shapes in ER Modeling	
Rectangle represents entity type.	
Double/Bold rectangle represent weak entity type.	 or 
Diamond represents relationship type.	
Double/Bold diamond represents weak relationship type.	 or 
Ellipse represents attribute type.	
Double ellipse represents multivalued attribute.	
Dashed ellipse denotes derived attribute.	
Line a link attribute to entity sets and entity sets to relationship sets	
Double lines which indicate total participation of an entity in a relationship set i.e., each entity in the entity set occurs in atleast one relationship in that relationship set.	

1.7. Mapping Cardinalities/Cardinality Ratio/Types of Relationship

Expresses the number of entities to which another entity can be associated *via* a relationship set. For a binary relationship set R between entity sets A and B , the mapping cardinality must be one of the following

a. One to One

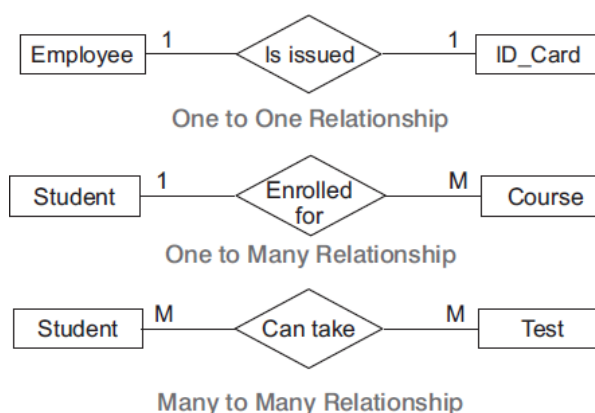
An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity in A .

b. One to Many

An entity in A is associated with any number (zero or more) of entities in B . An entity in B , however, can be associated with at most one entity in A .

c. Many to Many

An entity in A is associated with any number (zero or more) of entities in B and an entity in B is associated with any number (zero or more) of entities in A .

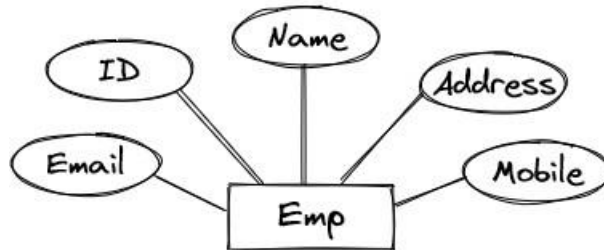


2. Converting ER-Diagram to Relational Model

Rule 1: Conversion of an entity set into a table

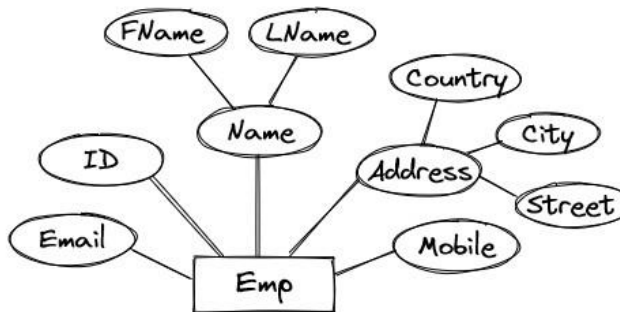
a) Representation of Strong Entity set with simple attributes.

EMP (ID, Name, Address, Mobile, Email)



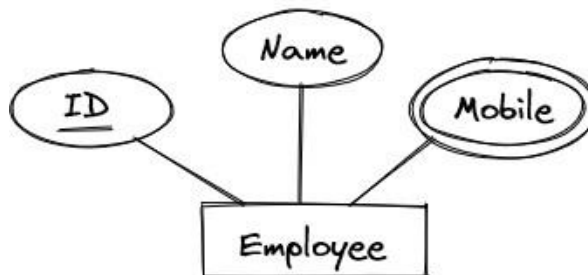
b) Representation of Strong entity set with Composite Attributes.

EMP (ID, FName, LName, Country, city, Street, Mobile, Email)



c) Representation of Strong entity set with a multi-valued attribute.

Employee (ID, Name, Mobile)

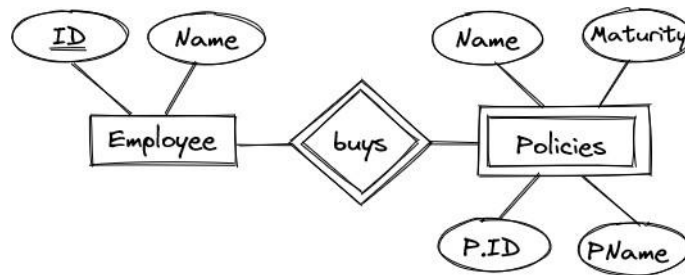


Employee table is further divided as below in order to eliminate redundancy due to multi-valued attribute.

ID	Name
001	Teena
002	Alina
003	Maryam

ID	Mobile1	Mobile2	Mobile3
001	1234	5678	
002	3578		
003	0214	9514	3574

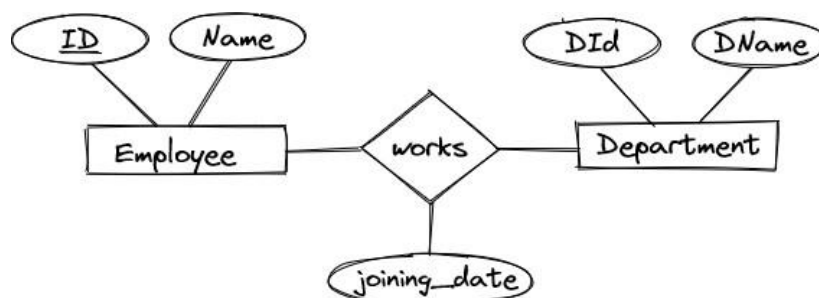
d) Representation of Weak entity set.



1) Employee(ID, Name)

2) Policies(P.ID, ID, Name, PName, Maturity, PName)

Rule 2: Conversion of Relationship into Relation.



1. Employee (Name, ID)

2. Department (DId, DName)

3. works (ID, DId, joining_date)

3. Keys:

Keys are one of the basic requirements of a relational database model. **keys** are fundamental components that ensure data integrity, uniqueness, and efficient access. It is widely used to identify the tuples(rows) uniquely in the table.

Different Types of Database Keys

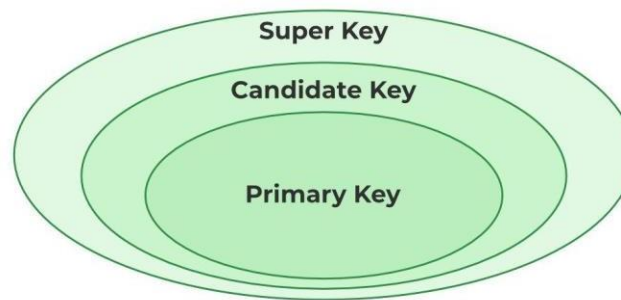
a. Super Key

The set of one or more attributes (columns) that can uniquely identify a tuple (record) is known as Super Key. It may include extra attributes that aren't essential for uniqueness but still uniquely identify the row.

Example: Consider the STUDENT table

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

A super key could be a combination of STUD_NO and PHONE, as this combination uniquely identifies a student.



Relation between Primary Key, Candidate Key, and Super Key

b. Candidate Key

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.

- A candidate key is a minimal super key, meaning it can uniquely identify a record but contains no extra attributes.
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- A candidate key must contain unique values, ensuring that no two rows have the same value in the candidate key's columns.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key.

Example: For the **STUDENT** table below, **STUD_NO** can be a candidate key, as it uniquely identifies each record.

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Table **STUDENT_COURSE**

STUD_NO	TEACHER_NO	COURSE_NO
1	001	C001
2	056	C005

A composite candidate key example: {**STUD_NO**, **COURSE_NO**} can be a candidate key for a **STUDENT_COURSE** table.

c. Primary Key

There can be more than one candidate key in relation out of which one can be chosen as the primary key.

- A **primary key** is a **unique key**, meaning it can uniquely identify each record (tuple) in a table.
- It must have **unique values** and cannot contain any **duplicate** values.
- A **primary key cannot be NULL**, as it needs to provide a valid, unique identifier for every record.
- A primary key does not have to consist of a single column. In some cases, a **composite primary key** (made of multiple columns) can be used to uniquely identify records in a table.
- Databases typically store rows ordered in memory according to primary key for fast access of records using primary key.

Example:

STUDENT table -> Student(**STUD_NO**, SNAME, ADDRESS, PHONE) , **STUD_NO** is a primary key

Table **STUDENT**

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789

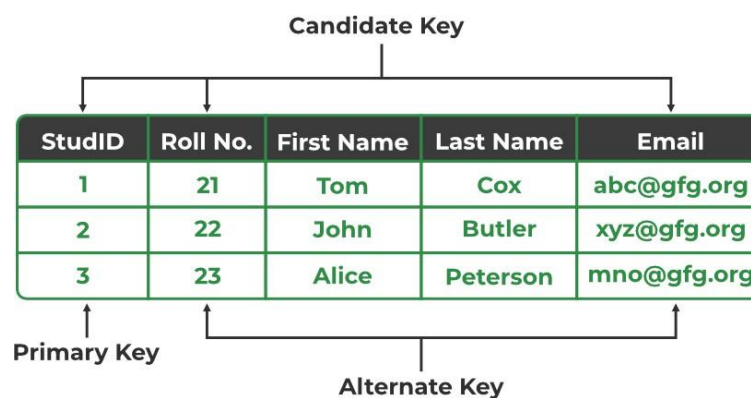
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

d. Alternate Key

An **alternate key** is any candidate key in a table that is **not** chosen as the **primary key**. In other words, all the keys that are not selected as the primary key are considered alternate keys.

- An alternate key is also referred to as a **secondary key** because it can uniquely identify records in a table, just like the primary key.
- An alternate key can consist of **one or more columns** (fields) that can uniquely identify a record, but it is not the primary key
- Eg:- SNAME, and ADDRESS is Alternate keys

Example: In the STUDENT table, both STUD_NO and PHONE are candidate keys. If STUD_NO is chosen as the primary key, then PHONE would be considered an alternate key.



Primary Key, Candidate Key, and Alternate Key

e. Foreign Key

A **foreign key** is an attribute in one table that refers to the **primary key** in another table. The table that contains the foreign key is called the **referencing table**, and the table that is referenced is called the **referenced table**.

- A **foreign key** in one table points to the **primary key** in another table, establishing a relationship between them.
- It helps **connect two or more tables**, enabling you to create relationships between them. This is essential for maintaining data integrity and preventing data redundancy.
- They act as a cross-reference between the tables.
- For example, DNO is a primary key in the DEPT table and a non-key in EMP

Example: Consider the STUDENT_COURSE table

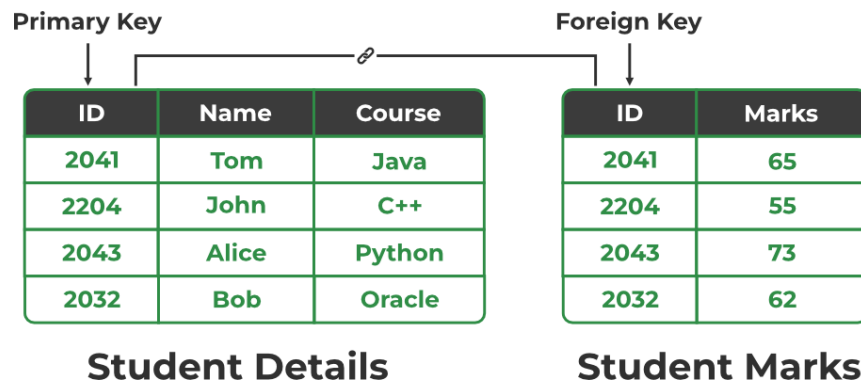
STUD_NO	TEACHER_NO	COURSE_NO
1	005	C001
2	056	C005

Here, STUD_NO in the STUDENT_COURSE table is a **foreign key** that references the STUD_NO primary key in the STUDENT table.

Explanation:

- Unlike the Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint. For Example, STUD_NO in the STUDENT_COURSE relation is not unique.

- It has been repeated for the first and third tuples. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique, and it cannot be null.



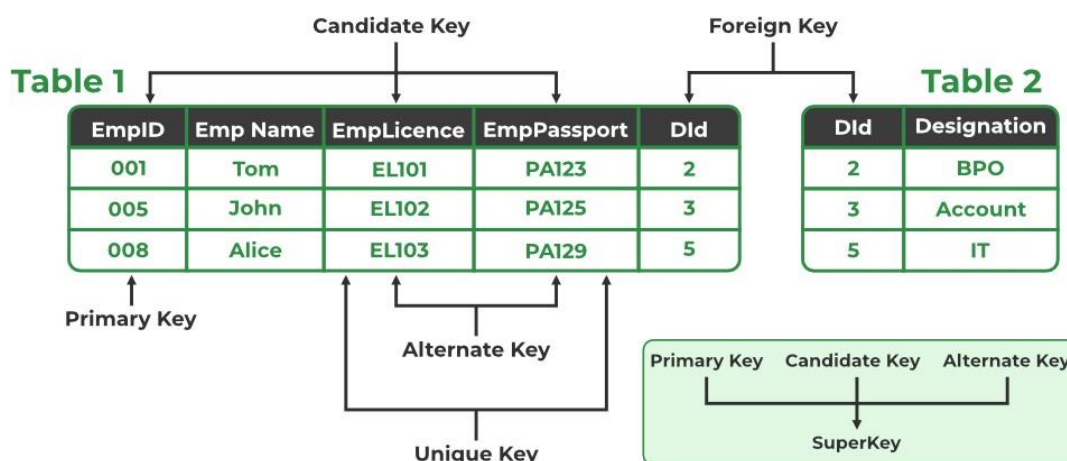
Relation between Primary Key and Foreign Key

f. Composite Key

Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used. It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

- It acts as a primary key if there is no primary key in a table
- Two or more attributes are used together to make a composite key.
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

Example: In the STUDENT_COURSE table, {STUD_NO, COURSE_NO} can form a composite key to uniquely identify each record.



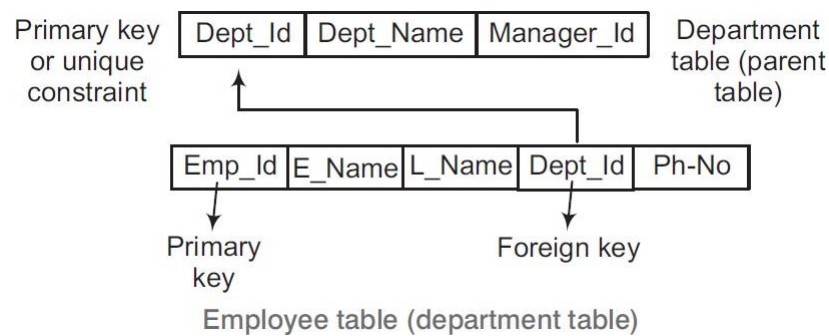
Different Types of Keys

4. Integrity Constraints

Necessary conditions to be satisfied by the data values in the relational instances so that the set of data values constitute a meaningful database. *There are four types of integrity constraints*

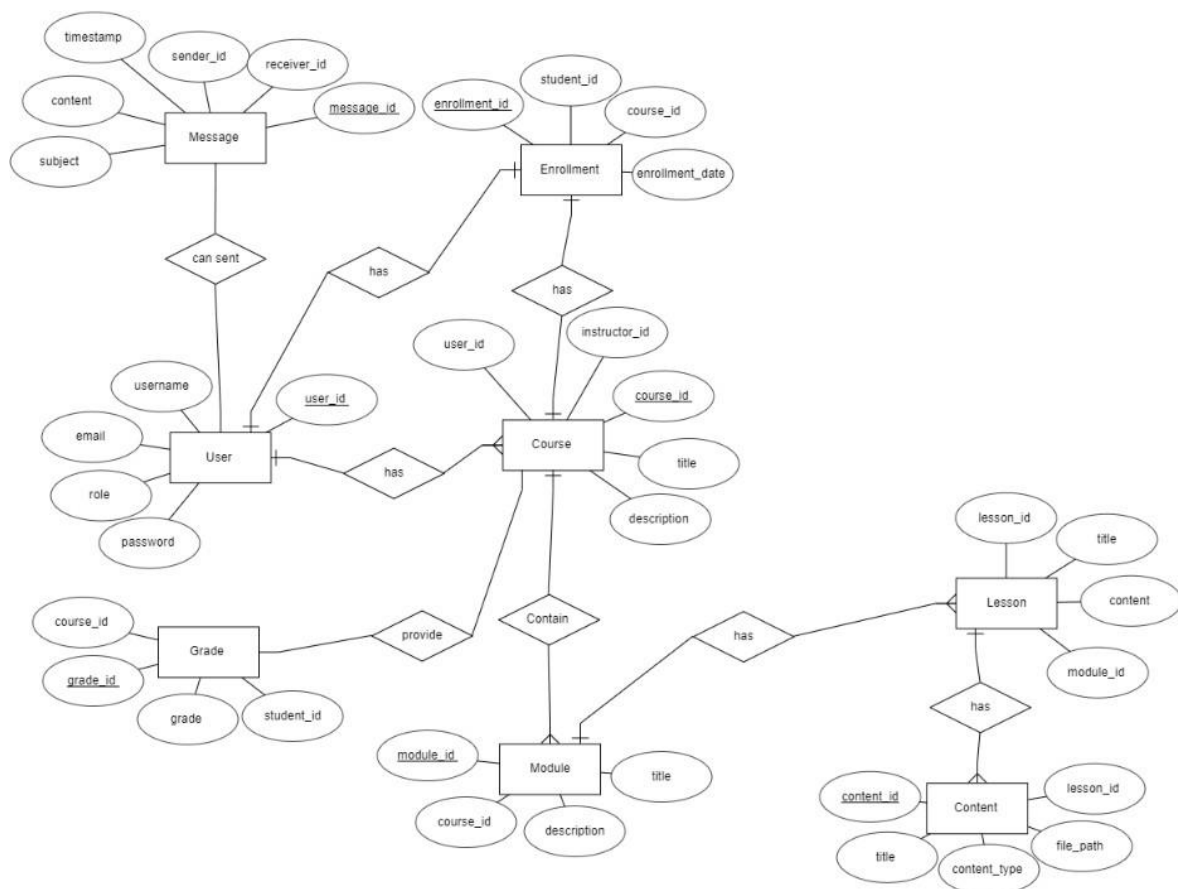
- Domain Constraint** The value of attribute must be within the domain.

- b. Key Constraint** Every relation must have a primary key.
- c. Entity Integrity Constraint** Primary key of a relation should not contain NULL values.
- d. Referential Integrity Constraint** In the relational model, two relations are related to each other over the basis of attributes. Every value of referencing attributes must be NULL or be available in the referenced attribute.



Case Study:

1. Draw ER diagram for Online Learning Management Systems



❖ **Entities in Online LMS**

- I. **User**
Represents all system users: admins, instructors, and students.
Attributes: User ID, username, password, role, email.
- II. **Course**
Represents individual courses.
Attributes: Course ID, title, description, instructor.
- III. **Module**
Sections within a course.
Attributes: Module ID, title, description, course ID.
- IV. **Lesson**
Learning units under modules.
Attributes: Lesson ID, title, content, module ID.
- V. **Content**
Materials linked to lessons (videos, documents, quizzes).
Attributes: Content ID, title, type, file path, lesson ID.
- VI. **Enrollment**
Tracks student enrollments in courses.
Attributes: Enrollment ID, student ID, course ID, date.
- VII. **Grade**
Stores student performance records.
Attributes: Grade ID, student ID, course ID, grade.
- VIII. **Message**
Communication between users.
Attributes: Message ID, sender ID, receiver ID, subject, content, timestamp.

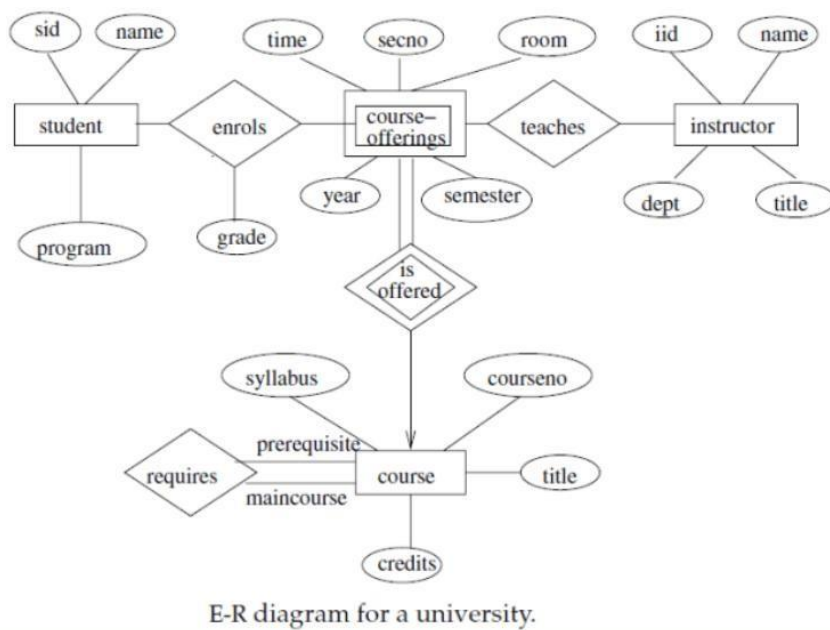
❖ **Key Relationships**

- I. **User ↔ Course (1:M):** One user (instructor) can teach multiple courses.
- II. **Course ↔ Module (1:M):** One course has multiple modules.
- III. **Module ↔ Lesson (1:M):** One module includes multiple lessons.
- IV. **Lesson ↔ Content (1:M):** One lesson has multiple content items.
- V. **User ↔ Course (via Enrollment) (M:M):** Students enroll in many courses; courses have many students.
- VI. **User ↔ User (via Message) (M:M):** Users can message each other.

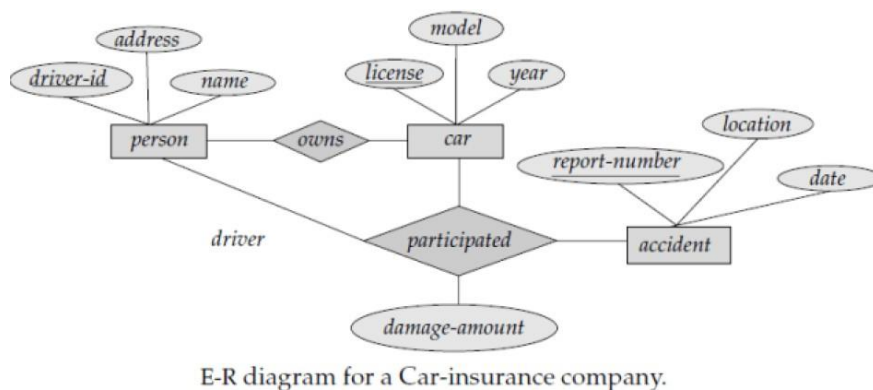
❖ **Database Design Tips**

- I. Understand data requirements early.
- II. Use suitable data types for attributes.
- III. Apply indexes to frequently searched columns.
- IV. Design with scalability in mind (e.g., partitioning, sharding).

2: A university registrar's office maintains data about the following entities: 1. courses, including number, title, credits, syllabus, and prerequisites; 2. course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; 3. students, including student-id, name, and program; 4. instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.



- 3: (a) Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
 (b) Construct appropriate tables for the above ER Diagram ?



Car insurance tables:

person (driver-id, name, address)

car (license, year, model)

accident (report-number, date, location)

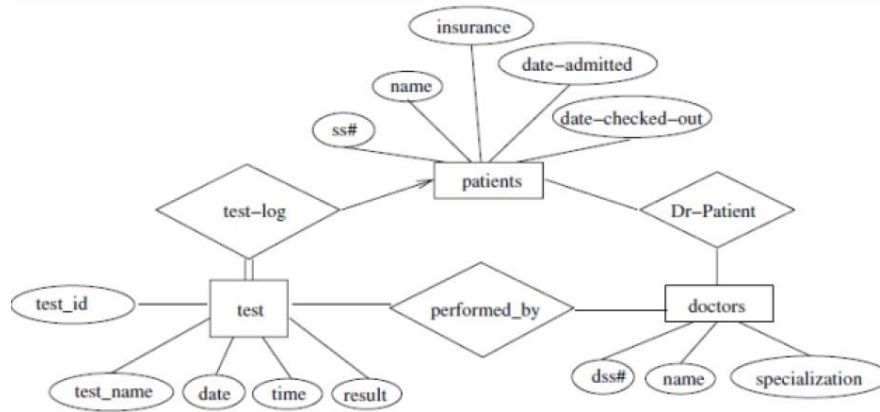
participated (driver-id, license, report-number, damage-amount)

Tables

4:

(a) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.

(b) Construct appropriate tables for the above ER Diagram :



E-R diagram for a hospital.

Patient(SS#, name, insurance)

Physician (name, specialization)

Test-log(SS#, test-name, date, time)

Doctor-patient (physician-name, SS#)

Patient-history(SS#, test-name, date)

Tables