ER MODEL & DATABASE DESIGN

Dr. Md Forhad Rabbi

frabbi-cse@sust.edu

Contents

Introduction

Entity Types – Entity Sets – Attributes and Keys

Introduction

- A database can be modeled as:
  - A collection of entities,
  - Relationship among entities.

Introduction

- An ***Entity–relationship model (ER model)*** describes the ***structure of a database with the help of a diagram,*** which is known as **Entity Relationship Diagram (ER Diagram)**.
- An ***ER model is a design or blueprint of a database that can later be implemented as a database***.
- The main components of E-R model are: ***entity set and relationship set.***

Introduction

- An ***ER diagram shows the relationship among entity sets.***
- An ***entity set is a group of similar entities*** and these entities can have attributes.
- In terms of ***DBMS, an entity is a table or attribute of a table in database***, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

Uses of entity relationship diagrams

- **Database design:** ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project.
- **Database troubleshooting:** ER diagrams are used to analyze existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it's going wrong.

Uses of entity relationship diagrams

- **Business information systems:** The diagrams are used to design or analyze relational databases used in business processes.
- **Education:** Databases are today’s method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.
- **Research:** Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data.

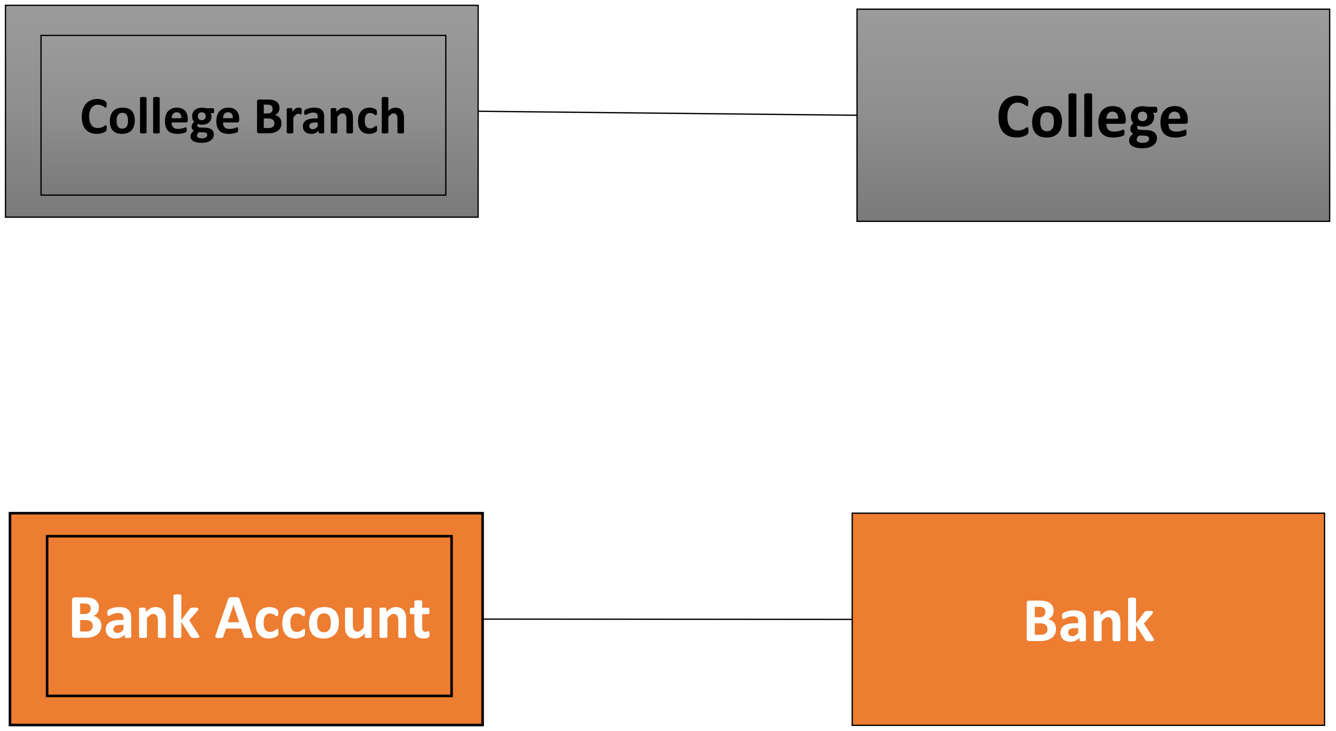
Entity

- An entity is ***an object that exists*** and is ***distinguishable from other objects.***
- ***Anything about which data are to be collected and stored.***
  - Example: specific person, company, event, plant
- An entity is ***represented as rectangle*** in an ER diagram.

Weak - Entity

- An entity that ***cannot be uniquely identified by its own attributes*** and relies on the relationship with other entity is called weak entity.
- The weak entity is ***represented by a double rectangle.***
- For example – ***a bank account cannot be uniquely identified without knowing the bank to which the account belongs,*** so bank account is a weak entity.
- A ***Engineering branch cannot be identified without knowing the College/ University*** to which branch belongs.

Weak - Entity



Attribute

- An attribute ***describes the property of an entity.***
- ***A characteristic of an entity.***
- An attribute is ***represented as Oval*** in an ER diagram.
- Entities have *attributes.*
- Each entity has a value for each of its attributes known as a Domain.
    - Example: people have *names* and *addresses*

Attribute

- Example:

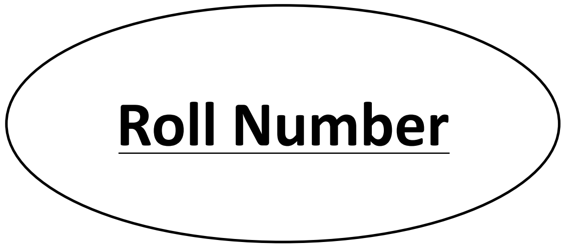
  customer = (*customer_id, customer_name, customer_street, customer_city*)
  loan = (*loan_number, amount*)

- **Domain** – the set of permitted values for each attribute.
- Attribute types:
  1. ***Key Attribute***
  2. ***composite attributes.***
  3. ***Single-valued*** and ***multi-valued*** attributes
     - Example: multivalued attribute: *phone_numbers*
  4. ***Derived attributes***
     - Can be computed from other attributes
     - Example: age, given date_of_birth

Attribute Types

1. **Key attribute:** can ***uniquely identify an entity*** from an entity set.

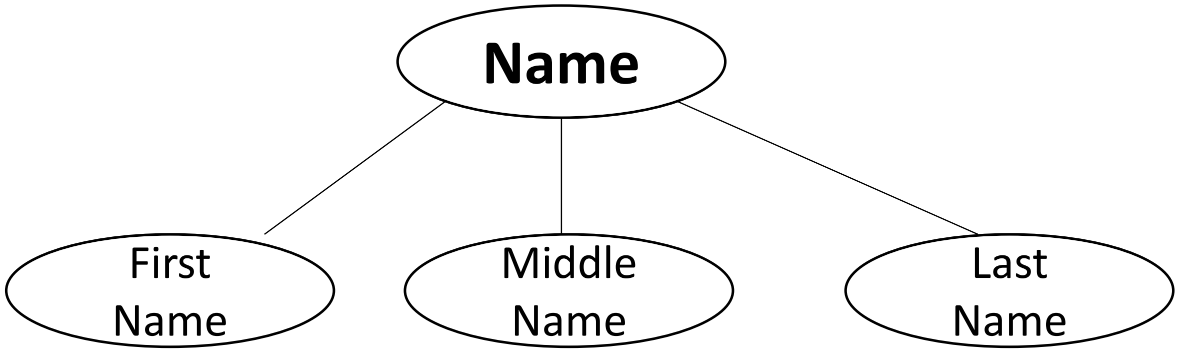
- For example, ***student roll number*** can uniquely identify a student from a set of students.
- Key attribute is ***represented by oval*** same as other attributes however the ***text of key attribute is underlined.***



Attribute Types

2. Composite Attribute: An attribute that is a ***combination of other attributes*** is known as composite attribute.

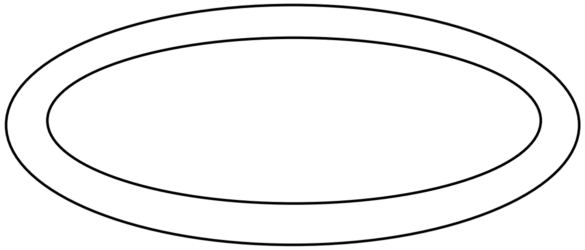
- For example: In student entity, the ***student address*** is a ***composite attribute*** as an ***address is composed of other attributes such as pin code, state, country.***



Attribute Types

3. Multivalued Attribute: An attribute that can ***hold multiple values*** is known as multivalued attribute.

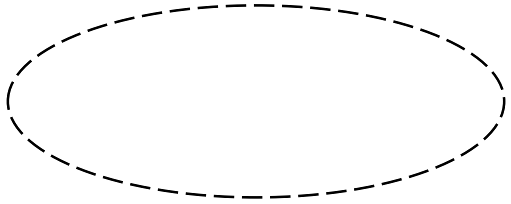
- It is ***represented with double ovals*** in an ER Diagram.
- For example – ***A person can have more than one phone numbers*** so the phone number attribute is multivalued.



Attribute Types

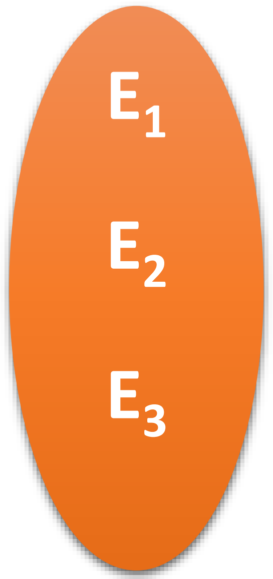
4. Derived attributes: derived attribute is one whose ***value is dynamic and derived from another attribute***.

- It is represented by ***dashed oval*** in an ER Diagram.
- For example – ***Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).***



Entity Set

- An ***entity set*** is a ***set of entities of the same type that share the same properties***.
    - Example: set of all persons, companies, trees, holidays.



Relationship Sets

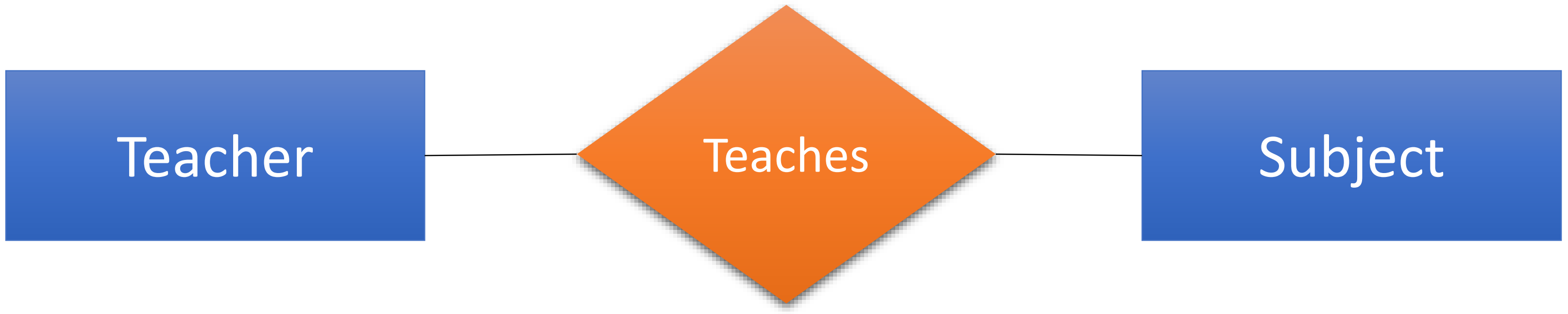
Relationship Set

- A ***relationship*** is an ***association among several entities.***
- A ***relationship set*** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

- where $(e_1, e_2, \ldots, e_n)$ is a relationship.

Relationship Set



Degree of a Relationship Set

- Refers to ***number of entity sets*** that ***participate in a relationship set.***
- Relationship sets that involve ***two entity sets are binary (or degree two).*** Generally, most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary.

Degree of a Relationship Set

- Relationship sets ***may involve more than two entity sets.***
- Example: Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee, job, and branch*

Mapping Constraints

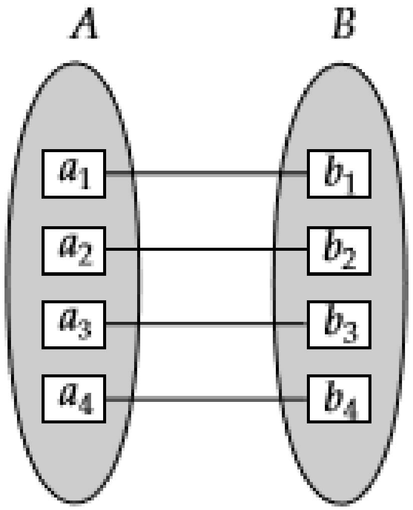
1. *One to One*
2. *One to Many*
3. *Many to One*
4. *Many to Many*

Mapping Constraints

- **Mapping Cardinalities:** express the ***number of entities to which another entity can be associated via a relationship***.
- For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:
  - One to one (1:1)
  - One to many (1:n)
  - Many to one (n:1)
  - Many to many (n:n)

Mapping Constraints

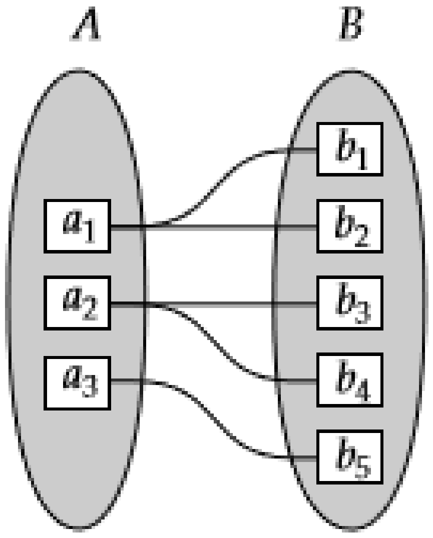
1. One to One: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



(a)

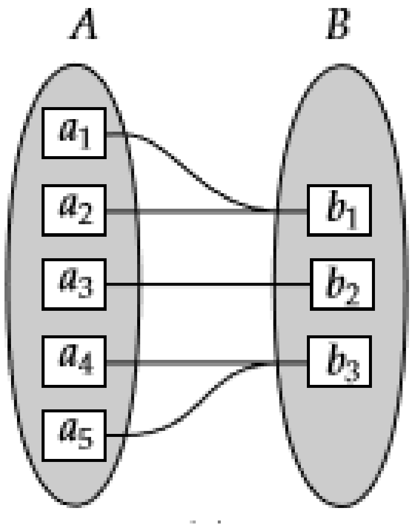
Mapping Constraints

2. 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.



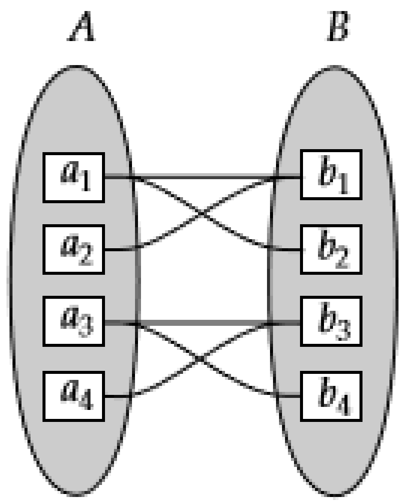
Mapping Constraints

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



Mapping Constraints

4. 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.



Keys

Keys

- We must have a way to specify how entities within a given entity set are distinguished.
- A ***key*** allows us to ***identify a set of attributes that is enough to distinguish entities from each other***.
- They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.
- A Key ***can be a single attribute or a group of attributes,*** where the combination may act as a key.

Types of Keys

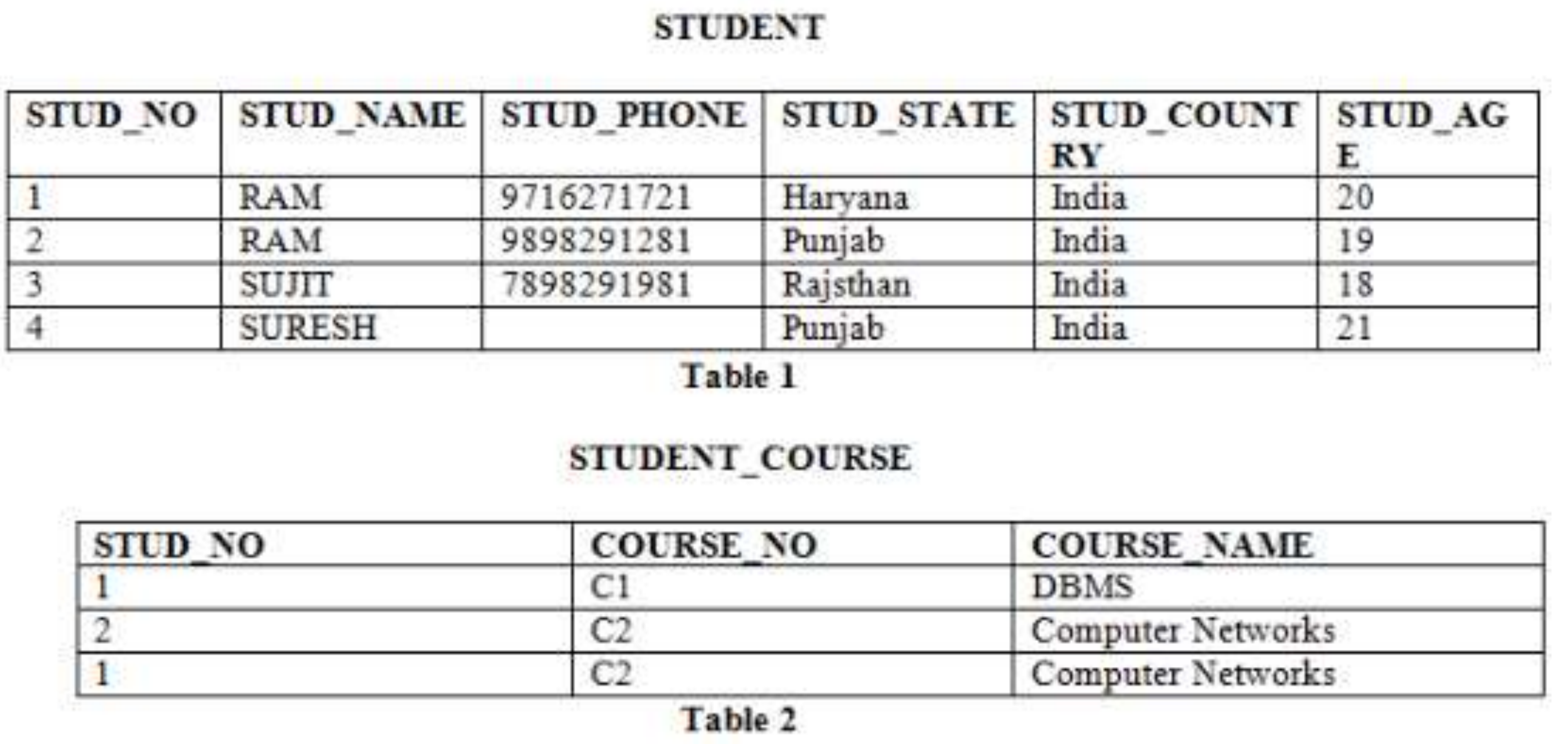
1. Primary Key
2. Super key
3. Candidate Key
4. Composite Key
5. Secondary Key
6. Foreign Key

Types of Keys

1. Candidate Key:

- The ***minimal set of attribute which can uniquely identify a tuple*** is known as candidate key.
  - For Example, STUD_NO in STUDENT relation.
- The ***value*** of Candidate Key is ***unique and not-null (Can't be empty)*** for every tuple.
- ***There can be more than one candidate key in a relation***.
  - For Example, ***STUD_NO*** as well as ***STUD_PHONE both are candidate keys*** for relation STUDENT.
- The candidate key can be simple (having only one attribute) or composite as well.
  - For Example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

Types of Keys



STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Types of Keys

2. Primary Key:

- Primary key is ***a candidate key that is most appropriate to become the main key*** for any table.
- The primary key is selected from one of the candidate keys and becomes the identifying key of a table.
- It can uniquely identify any data row of the table.
- Unique + Not Null
  - For the table Student we can make the student_id column as the primary key.

Types of Keys

3. Super Key:

- A super key is ***a set of one of more columns (attributes) to uniquely identify rows*** in a table.
- Super Key is a superset of Candidate key.
- In the table defined above super key would include ***student_id, (student_id, name), phone*** etc.
- ***Confused?***
    - The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Types of Keys

3. Super Key:

- Next comes, (***student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.***
- Similarly, ***phone number for every student will be unique***, hence again, phone can also be a key.
- So they all are super keys.

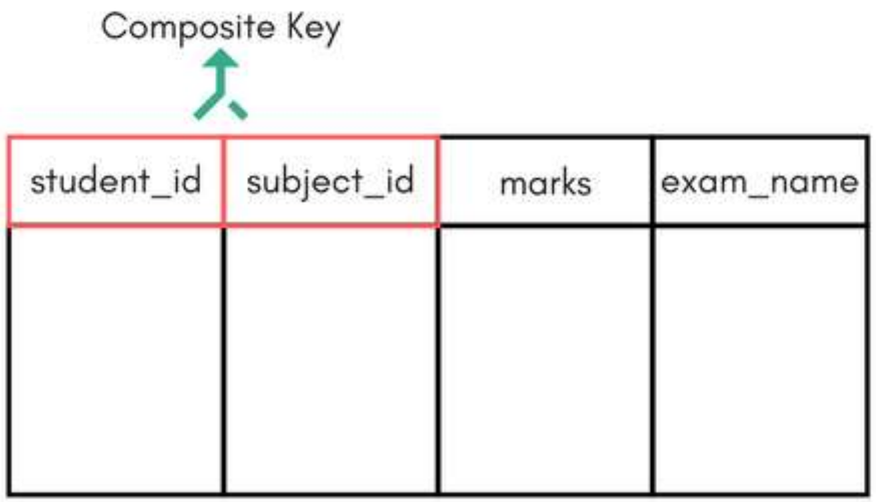
Types of Keys

4. Composite Key:

- Key that consists of ***two or more attributes that uniquely identify any record*** in a table is called Composite key.
- ***But the attributes which together form the Composite key are not a key independently or individually.***

Types of Keys

4. Composite Kev:



Types of Keys

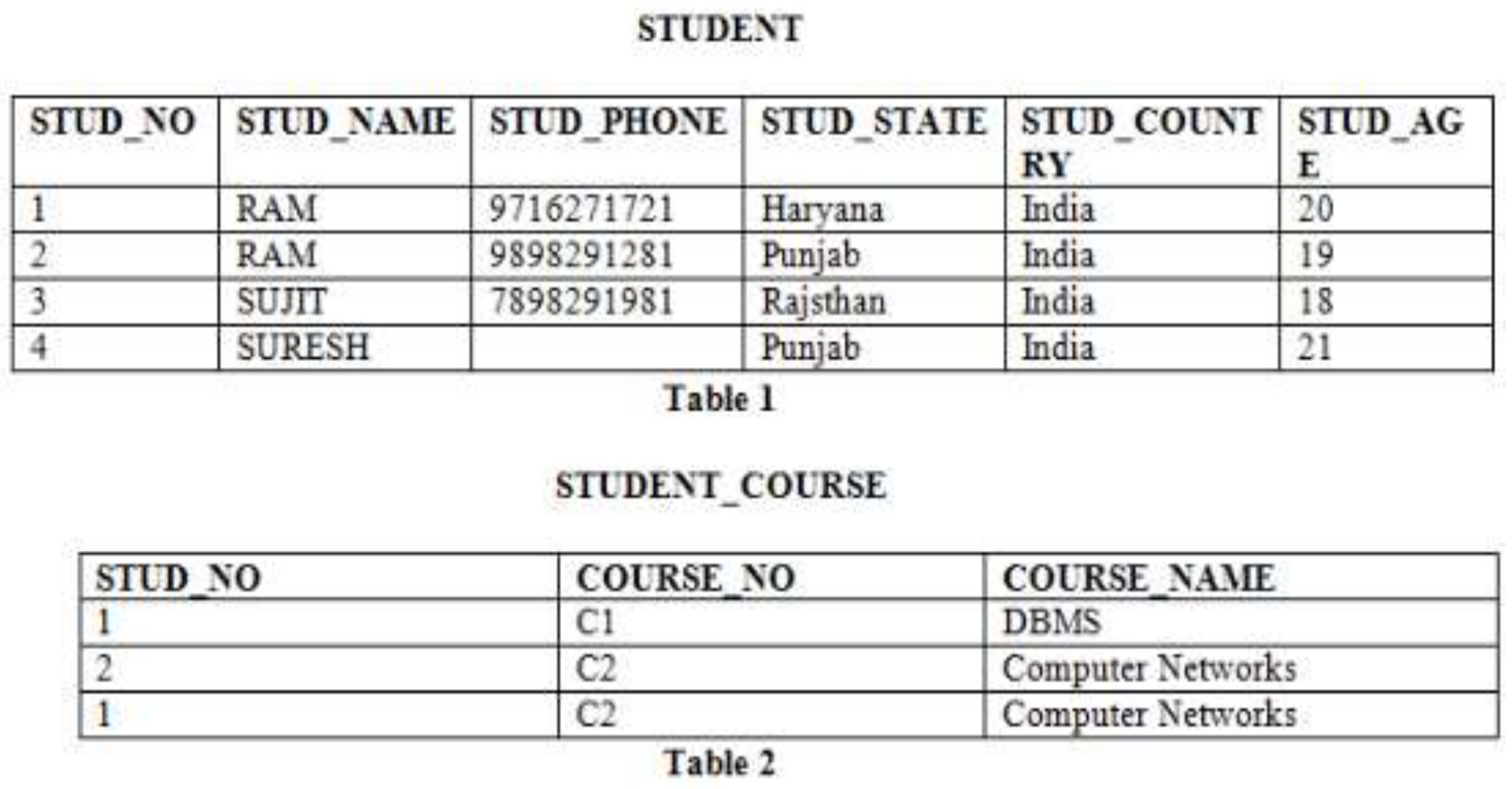
5. Secondary Key / Alternate Key: Only one of the candidate keys is selected as the primary key. The rest of them are known as secondary keys.

Types of Keys

6. Foreign Key:

- A foreign key is an attribute value in a table that acts as the primary key in another table. Hence, the foreign key is useful in linking together two tables.
- 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 STUDENT_COURSE relation is not unique. It has been repeated for the first and third tuple. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique and it cannot be null.

Types of Keys



STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

E-R Diagram

E-R Diagram

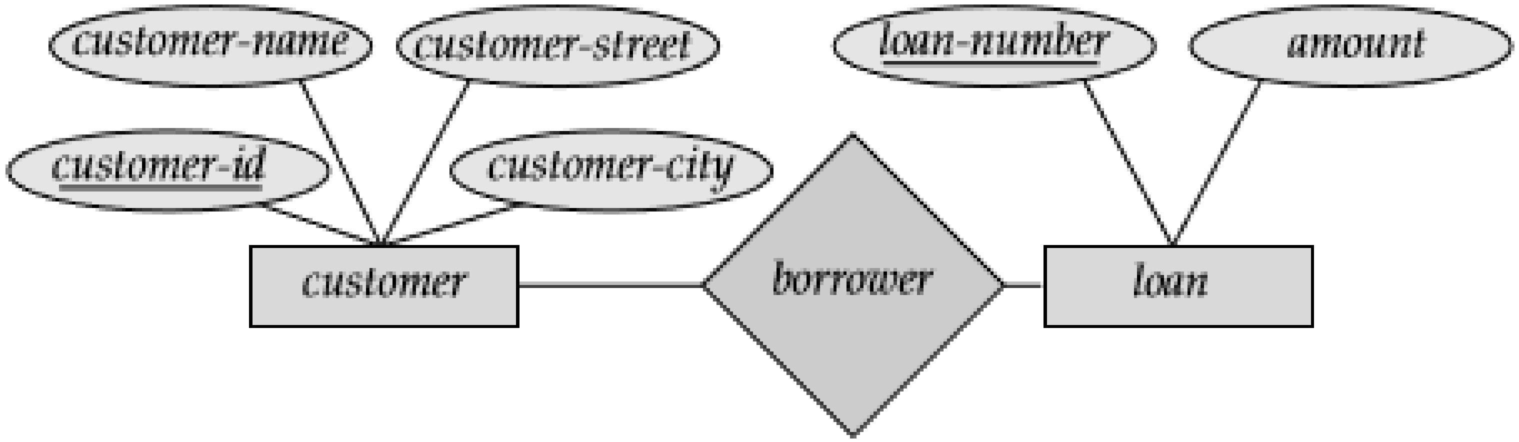
- As we know The entity-relationship (E-R) data model is based on a perception of a real world that consists of a collection of basic objects, called entities, and of relationships among these objects.
- E-R diagram can express the overall logical structure of a database graphically. E-R diagrams are simple and clear—qualities that may well account in large part for the widespread use of the E-R model.

E-R Diagram

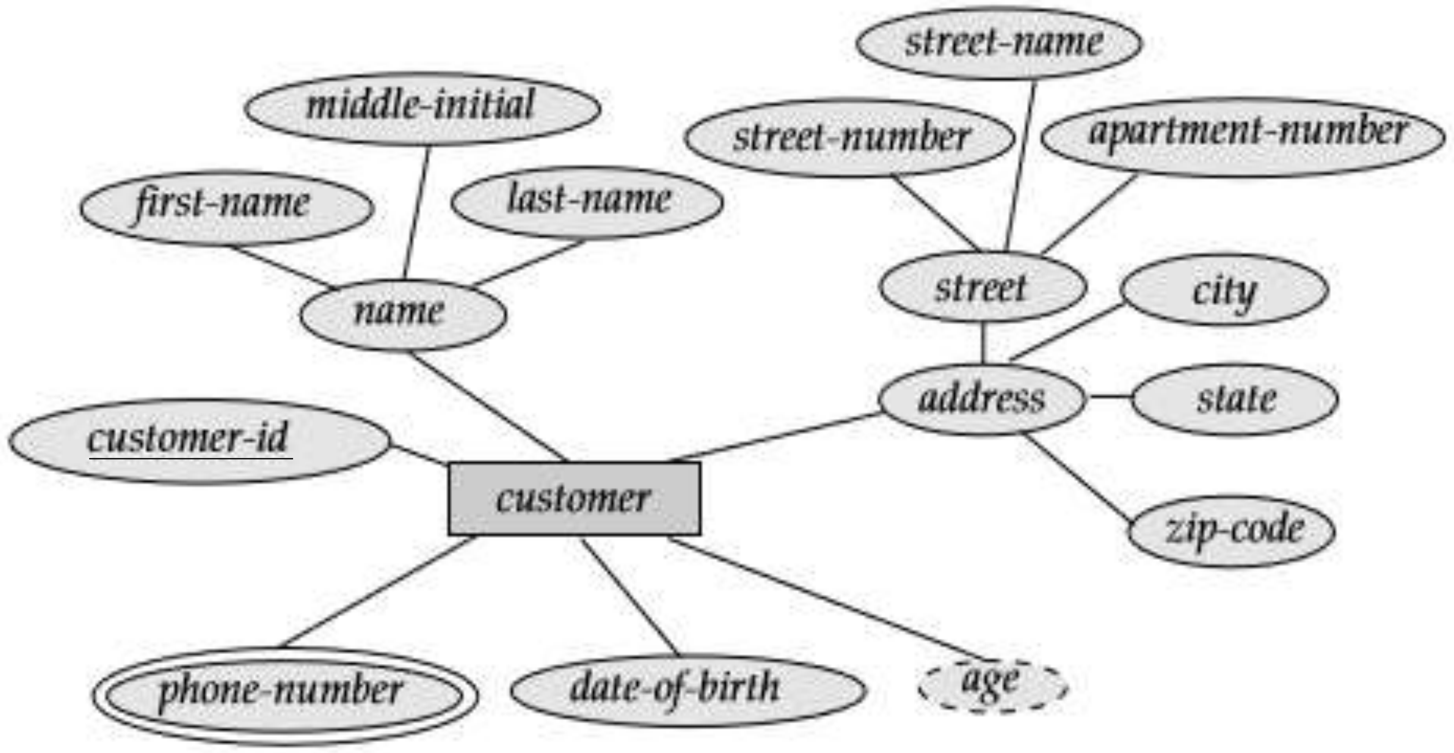
Such a diagram consists of the following major components:

- **Rectangles:** which represent entity sets
- **Ellipses/ Oval:** which represent attributes
  - **Double ellipses:** which represent multivalued attributes
  - **Dashed ellipses:** which denote derived attributes
- **Diamonds:** which represent relationship sets
- **Lines:** which link attributes to entity sets and entity sets to relationship sets
- **Double lines:** which indicate total participation of an entity in a relationship set
- **Double rectangles:** which represent weak entity sets

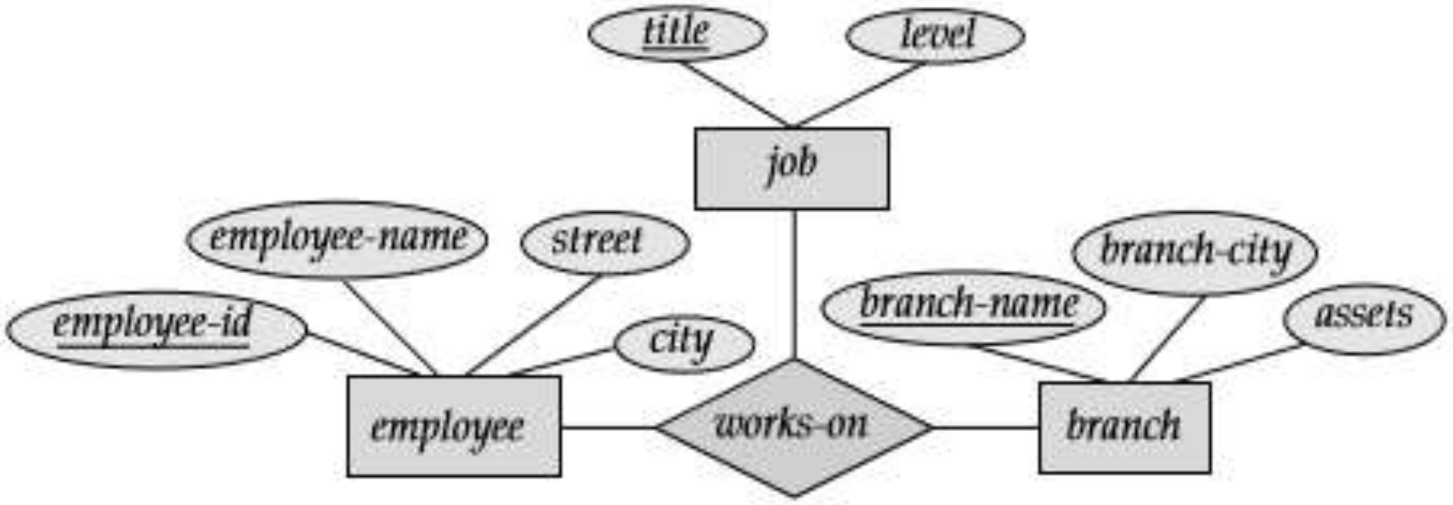
E-R Diagram – Example



E-R Diagram – Example
E-R diagram with composite, multivalued, and derived attributes

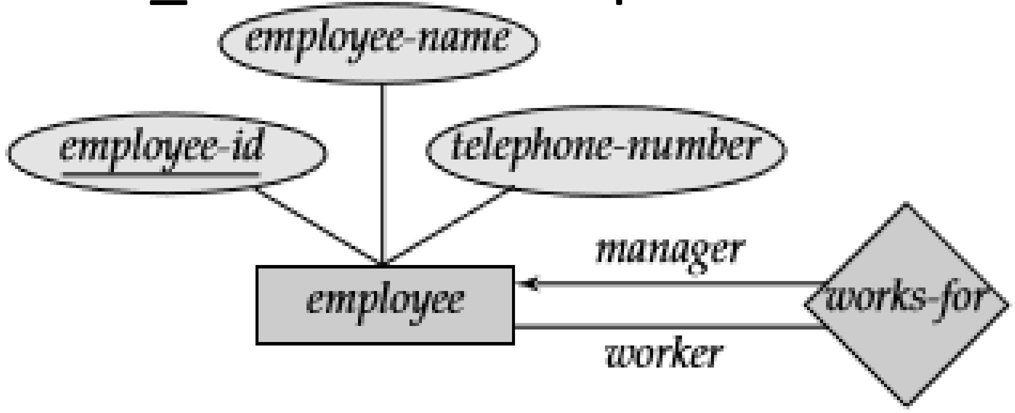


E-R Diagram – Example
E-R diagram with Ternary relationship set



Roles

- ***Roles*** are ***indicated*** in E-R diagrams ***by labeling the lines*** that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship.
- The ***labels "manager"*** and ***"worker"*** are called ***roles;*** they specify how employee entities interact via the works_for relationship set.

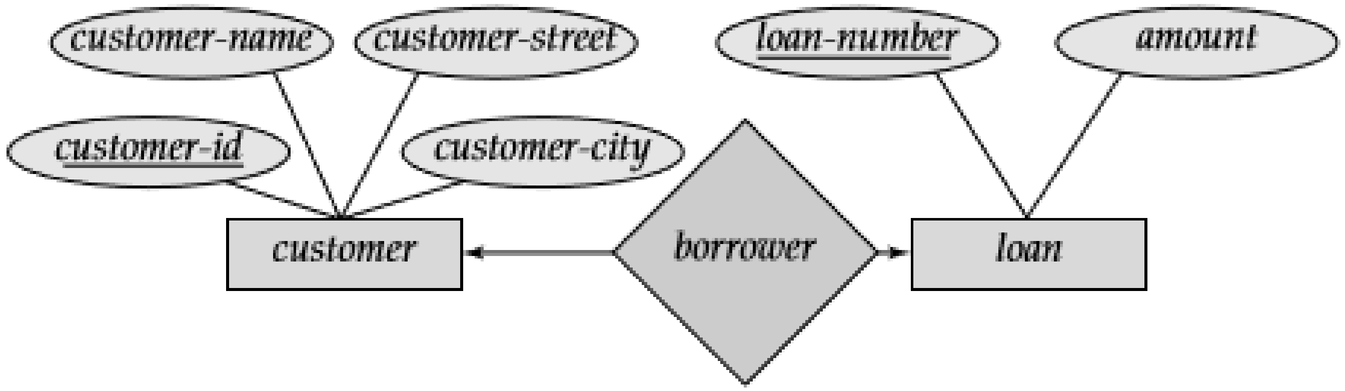


Cardinality Constraints

- We express cardinality constraints by drawing either a ***directed line (→), signifying "one,"*** or an ***undirected line (—), signifying "many,"*** between the relationship set and the entity set.

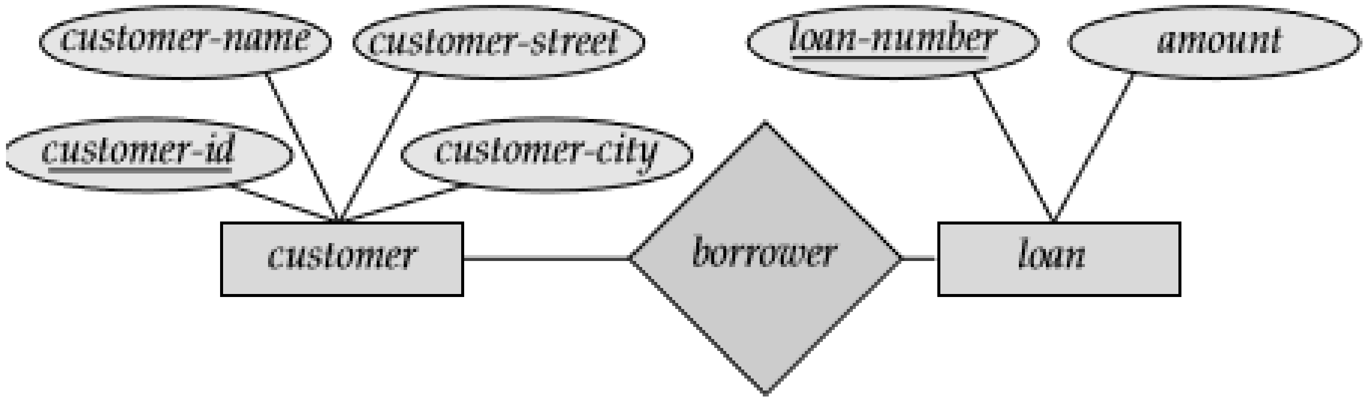
Cardinality Constraints

- **One-to-one relationship:**
  - A customer is associated with at most one loan via the relationship *borrower*
  - A loan is associated with at most one customer via *borrower*



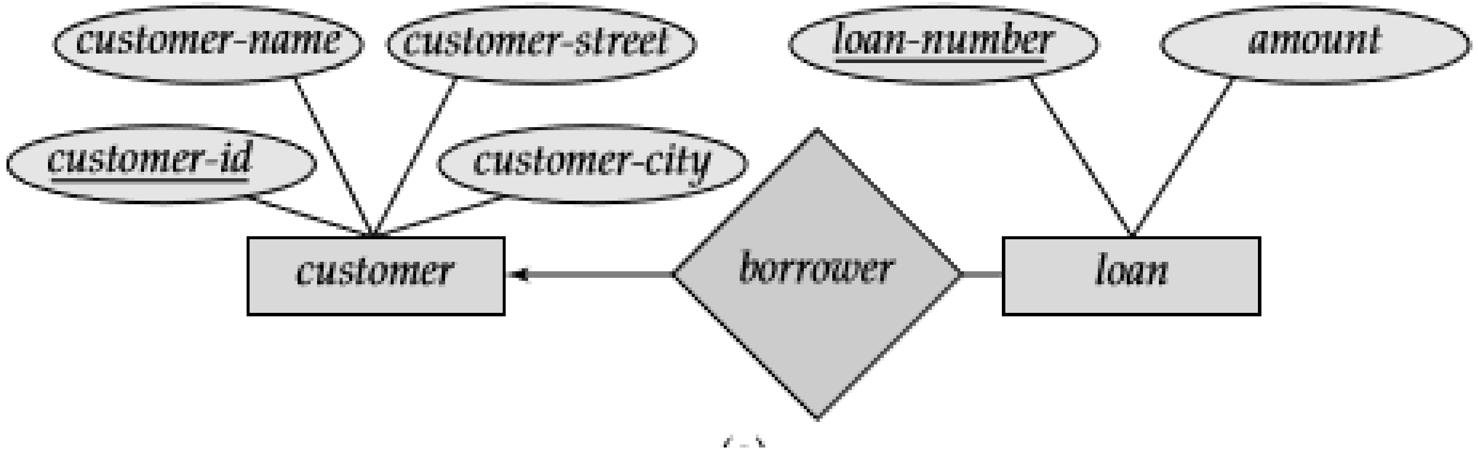
Cardinality Constraints

- **Many-to-Many:**
  - A customer is associated with several (possibly 0) loans via borrower
  - A loan is associated with several (possibly 0) customers via borrower



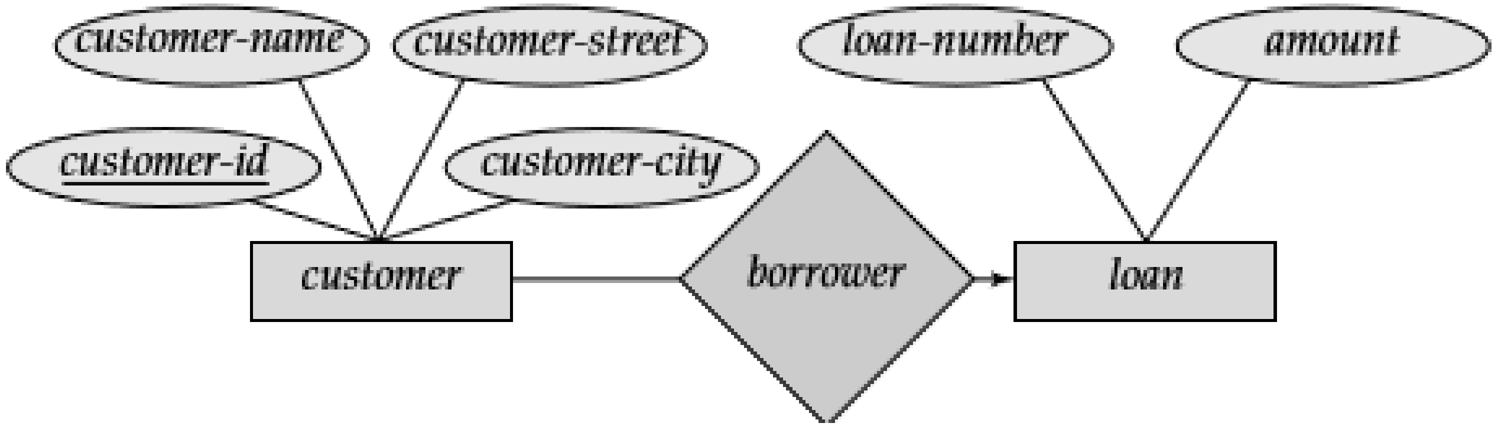
Cardinality Constraints

- **One-to-Many Relationship:**
  - In the one-to-many relationship a loan is associated with at most one customer via *borrower*, a customer is associated with several (including 0) loans via *borrower.*



Cardinality Constraints

- **Many-to-One:**
  - In a many-to-one relationship a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*

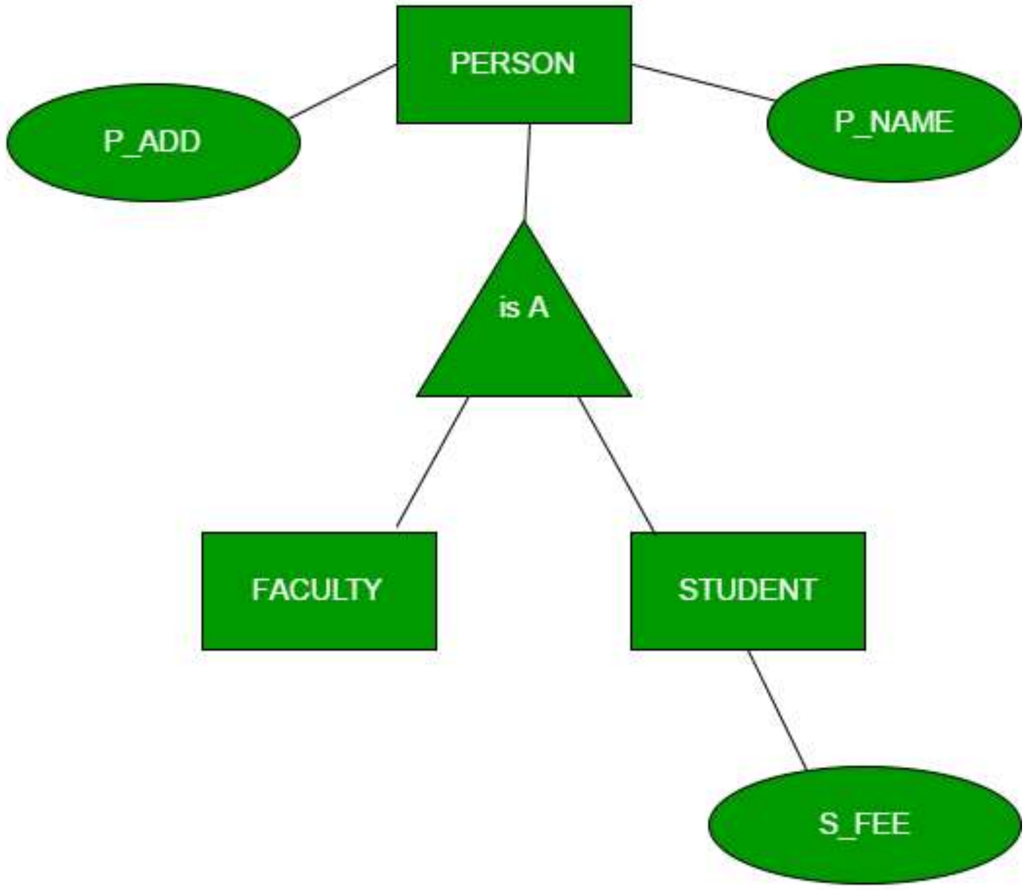


ER Model: Generalization, Specialization and Aggregation

Generalization

- Generalization is the process of ***extracting common properties from a set of entities and create a generalized entity from it.***
- It is ***a bottom-up approach*** in which two or more entities can be generalized to a higher level entity if they have some attributes in common.
  - For Example*, STUDENT and FACULTY can be generalized to a higher level entity called PERSON* as shown in next slide.
  - In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).

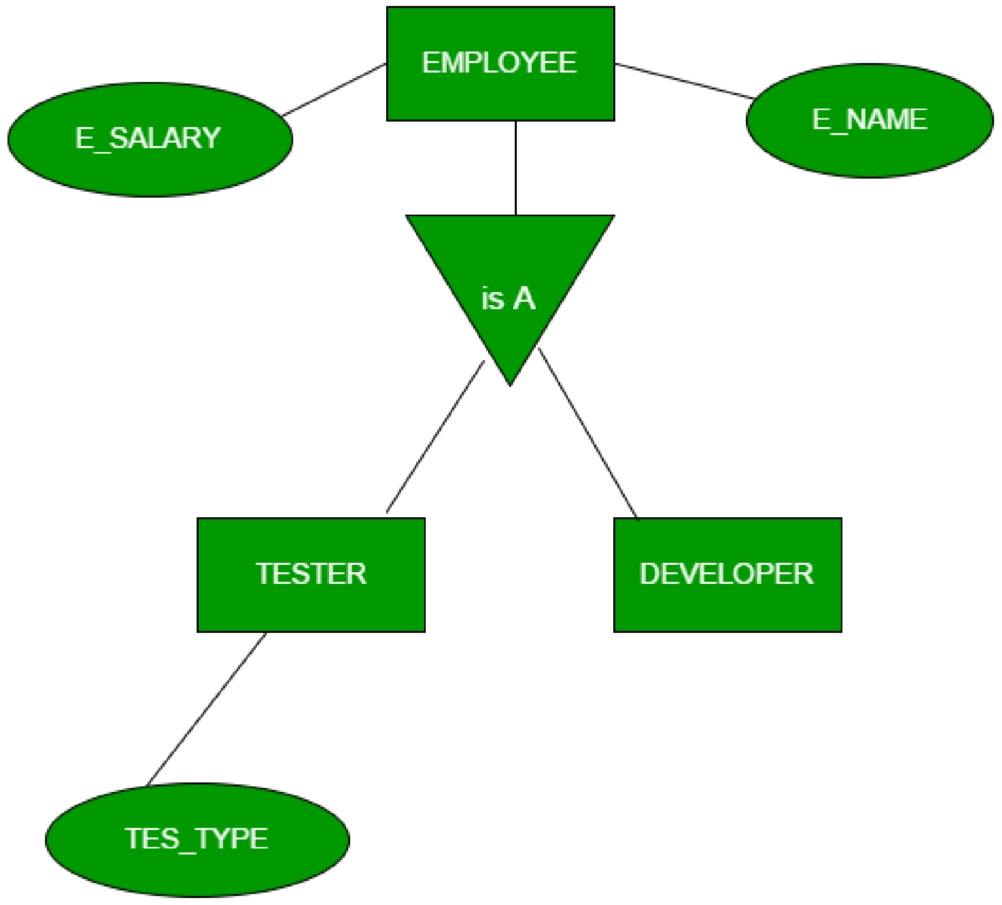
Generalization



Specialization

- In specialization, ***an entity is divided into sub-entities based on their characteristics***.
- It is a ***top-down approach*** where higher level entity is specialized into two or more lower level entities.
  - For Example, *EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc.* as shown in next slide.
  - In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).

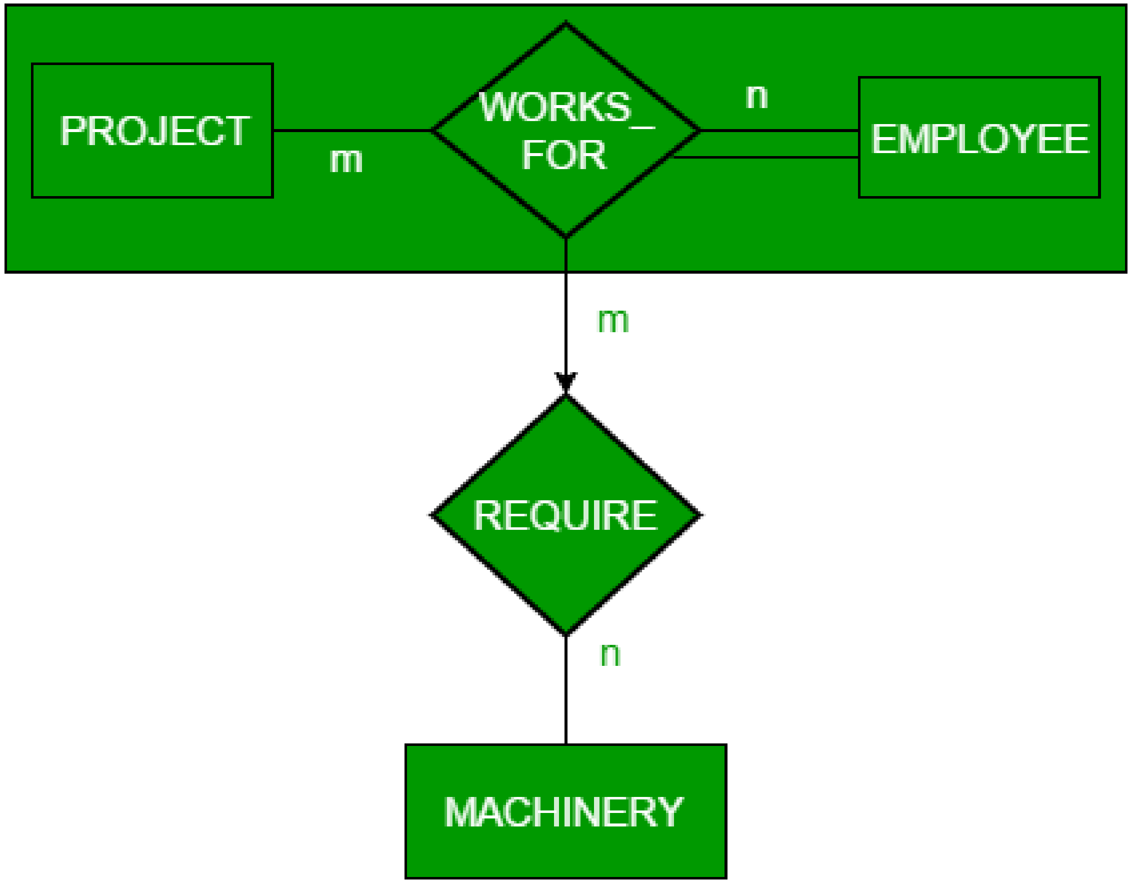
Specialization



Aggregation

- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.
  - For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY.
  - Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.

Aggregation



Relational Model

Relational Model – Introduction

- The **relational model** represents the ***database as a collection of relations***.
- Informally, each relation resembles ***a table of values*** or, to some extent, a flat file of records. It is called a ***flat file*** because ***each record has a simple linear or flat structure.***
- When a relation is thought of as a table of values – ***each row in the table represents a collection of related data values.***
- ***A row represents a fact*** that typically ***corresponds to a real-world entity or relationship***.
- The table name and column names are used to help to interpret the meaning of the values in each row.

Relational Model – Introduction

- For example, this table is called STUDENT because each row represents facts about a particular student entity.
- Student Table:

USN	Name	Year	Class
12CT12	ABC	1	CTIS
12CT15	CDE	1	CTIS
13BT13	XYZ	2	DS
13BT13	HDG	3	CSE
13BT56	HJL	3	CSE

Relational Model – Introduction

- The column names—USN, Name, year, and Class — specify how to interpret the data values in each row, based on the column each value is in.
- ***All values in a column are of the same data type***.
- In the formal relational model terminology –
  - ***A row is called a tuple***
  - ***A column header is called an attribute***, and
  - ***The table is called a relation.***
- The ***data type describing the types of values*** that can appear in each column is ***represented by a domain of possible values.***

Relational Model – Introduction

- The relational model is today the primary data model for commercial data-processing applications. It has attained its primary position because of its simplicity, which eases the job of the programmer, as compared to earlier data models such as the network model or the hierarchical model.
- ***Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).***

Relational Model – Concepts and Characterstics

1. **Attribute:** Attributes are the properties that define a relation. e.g.; **ROLL_NO**, **NAME**
2. **Relation Schema:** A relation schema represents name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE and AGE) is relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.
3. **Tuple:** Each row in the relation is known as tuple.
4. **Relation Instance:** The set of tuples of a relation at a particular instance of time is called as relation instance.

Relational Model – Concepts

5. Degree: The ***number of attributes in the relation*** is known as degree of the relation. The **STUDENT** relation defined above has degree 4.

6. Cardinality: The ***number of tuples*** in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 5.

7. Column: Column represents the set of values for a particular attribute.

8. NULL Values: The value which is not known or unavailable is called NULL value.

Constraints in Relational Model

- **Domain Constraints:** These are ***attribute level constraints***. An attribute can only take values which lie inside the domain range. e.g,; If a constrains AGE>0 is applied on STUDENT relation, inserting negative value of AGE will result in failure.
- **Key Integrity:** Every relation in the database should have atleast one set of attributes which defines a tuple uniquely. Those set of attributes is called key. e.g.; USN in STUDENT is a key. No two students can have same roll number. So a key has two properties:
  - It should be unique for all tuples.
  - It can’t have NULL values.

Constraints in Relational Model

- **Referential Integrity:** When one attribute of a relation can only take values from other attribute of same relation or any other relation, it is called referential integrity.

Relational Model – Advantages

- **Structural independence:** In relational model, changes in the database structure do not affect the data access. When it is possible to make change to the database structure without affecting the DBMS's capability to access data, we can say that structural independence has been achieved. So, relational database model has structural independence.
- **Structural independence:** In relational model, changes in the database structure do not affect the data access. When it is possible to make change to the database structure without affecting the DBMS's capability to access data, we can say that structural independence has been achieved. So, relational database model has structural independence.
- **Design, implementation, maintenance and usage ease:** The relational database model achieves both data independence and structure independence making the database design, maintenance, administration and usage much easier than the other models.

Thank You