Chapter-3

Database Designs

Relational Constraints/Integrity Rules

- **The different Relational Integrity Rules**
  - **Domain Integrity**
    - No value of the attribute should be beyond the allowable limits
  - **Entity Integrity**
    - In a base relation, no attribute of a Primary Key can assume a value of NULL
  - **Referential Integrity**
    - If a Foreign Key exists in a relation, either the Foreign Key value must match a Candidate Key value in its home relation or the Foreign Key value must be NULL
  - **Enterprise Integrity**
    - Additional rules specified by the users or database administrators of a database are incorporated

Key constraints

- **Super Key**:
- set of attribute or attributes that uniquely identifies a tuple within a relation.
- A key which we just understand or think at the beginning and they can even be every key in a table
- **Candidate:**
- It is minimal subset of super key
- If any proper subset of a super key is super key then that key cannot be a candidate key.
- **Primary keys**
- The candidate key chosen to uniquely identify each row of data in a table
- The DBA Picks any one of the candidate key to be the primary key
  - The rest one of the candidate key become alternate keys
- **Foreign key**
- Primary key of another table used to define relationship with another table and is used to maintain integrity

Key constraint Example

Student Table

SID	REG_ID	NAME	BRANCH	EMAIL
1	CS-2019-37	John	CS	john@xyz.com
2	CS-2018-02	Adam	CS	adamcool@xyz.com
3	IT-2019-01	Adam	IT	adamnerd@xyz.com
4	ECE-2019-07	Elly	ECE	elly@xyz.com

- Possible combination of Super keys
  - SID
  - REG_ID
  - EMAIL
  - SID + REG_ID
  - REG_ID + EMAIL
  - EMAIL + SID
  - SID + REG_ID + EMAIL
- The following key are combinations of candidate keys
  - SID
  - REG_ID
  - EMAIL
- REG_ID is the primary key
- SID and EMAIL become alternate keys

Relational Views

- There are two kinds of relation in relational database.

1. Base Relation(*Named Relation)*

- corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

2. View (Unnamed Relation)

- A View is the dynamic result of one or more relational operations operating on the base relations to produce another virtual relation that does not actually exist as presented.
- **View is a virtually derived relation** that does not necessarily exist in the database but can be produced upon request by a particular user at the time of request.

Purpose of a view

- Hides unnecessary information from users:
  - since only part of the base relation (Some collection of attributes, not necessarily all) are to be included in the virtual table.
- Provide powerful flexibility and security:
  - since unnecessary information will be hidden from the user there will be some sort of data security.
- Provide customized view of the database for users:
  - each user is going to be interfaced with his own preferred data set and format by making use of the Views.

Purpose of a view

- Update on views derived from various relations is not allowed since it may violate the integrity of the database.
- Since aggregation and summary results are computed from a base relation and does not exist actually.
  - Update on view with aggregation and summary is not allowed.

Schemas and Instances

- When a database is designed using a Relational data model
  - all the data is represented in a form of a table.
    - In such definitions and representation
      - there are two basic components of the database.
        - » Schemas
        - » Instances

Schemas and Instances

- **Database Schema**:
  - specifies name of relation and the collection of the attributes (specifically the Name of attributes).
  - refer to a description of database or describes how data is to be structured
  - specified during database design
  - should not be changed unless during maintenance
- **Schema Diagrams**
  - convention to display some aspect of a schema visually
- **Schema Construct**
  - refers to each object in the schema (e.g. STUDENT)
    - E.g.: STUDENT (FName, LName, Id, Year, Dept, Sex)

Schemas and Instances

Instances

- is the collection of data in the database at a particular point of time (snap-shot).
- Also called **State or Snap Shot or Extension of the** database
- Refers to the actual data in the database at a specific point in time
- State of database is changed any time we add, delete or update an item.
- Since Instance is actual data of database at some point in time, changes rapidly

- To define a new database, we specify its database schema to the DBMS (database is empty)
- database is initialized when we first load it with data

Database design

3. Conceptual Database Design(using ERD)

Database design is the process of coming up with different kinds of specification for the data to be stored in the database. The database design part is one of the middle phases we have in information systems development where the system uses a database approach. Design is the
part on which we would be engaged to describe how the data should be perceived at different levels and finally how it is going to be stored in a computer system.
Information System with Database application consists of several tasks which include:

- ☐ *Planning of Information systems Design*
- ☐ *Requirements Analysis,*
- ☐ ***Design (Conceptual, Logical and Physical Design)***
- ☐ *Tuning*
- ☐ *Implementation*
- ☐ *Operation and Support*

From these different phases, the prime interest of a database system will be the Design part which is again sub divided into other three sub-phases.
These sub-phases are:

1. ***Conceptual Design***
2. ***Logical Design, and***
3. ***Physical Design***

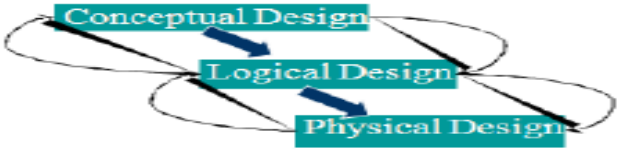
In general, one has to go back and forth between these tasks to refine a database design, and decisions in one task can influence the choices in another task.

☐ In developing a good design, one should answer such questions as:

- What are the relevant Entities for the Organization
- What are the important features of each Entity
- What are the important Relationships
- What are the important queries from the user
- What are the other requirements of the Organization and the Users

Database design

3.1 The Three levels of Database Design



Conceptual Database Design

- Conceptual design is the process of constructing a model of the information used in an enterprise, *independent of any physical considerations.*
- It is the source of information for the logical design phase.
- Mostly uses an Entity Relationship Model to describe the data at this level.
- After the completion of Conceptual Design one has to go for refinement of the schema, which is verification of Entities, Attributes, and Relationships

Logical Database Design

- Logical design is the process of constructing a model of the information used in an enterprise based on a specific data model (e.g. relational, hierarchical or network or object), but independent of a particular DBMS and other physical considerations.
- Normalization process
- Collection of Rules to be maintained
- Discover new entities in the process
- Revise attributes based on the rules and the discovered Entities

Physical Database Design

- Physical design is the process of producing a description of the implementation of the database on secondary storage. -- defines specific storage or access methods used by database
- What are the other requirements of the Organization and the Users

Database design

- Describes the storage structures and access methods used to achieve efficient access to the data.
- Tailored to a specific DBMS system -- Characteristics are function of DBMS and operating systems
- Includes estimate of storage space

Conceptual Database Design

- Conceptual design revolves around discovering and analyzing organizational and user data requirements
- The important activities are to identify
  - Entities
  - Attributes
  - Relationships
  - Constraints
- And based on these components develop the ER model using
  - ER diagrams

3.2 The Entity Relationship (E-R) Model

- Entity-Relationship modeling is used to represent conceptual view of the database
- The main components of ER Modeling are:

Entities
- Corresponds to entire table, not row
- Represented by Rectangle

Attributes
- Represents the property used to describe an entity or a relationship
- Represented by Oval

Relationships
- Represents the association that exist between entities
- Represented by Diamond

Constraints
- Represent the constraint in the data

Before working on the conceptual design of the database, one has to know and answer the following basic questions.

Entity Relationship Diagram (ERD)

- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* that hold? Constraints on each data with respect to update, retrieval and store.
- Represent this information pictorially in *ER diagrams*, then map ER diagram into a relational schema.

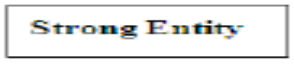
Developing an E-R Diagram

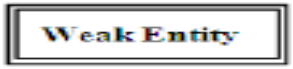
- Designing conceptual model for the database is not a one linear process but an iterative activity where the design is refined again and again.
  - To identify the entities, attributes, relationships, and constraints on the data, there are different set of methods used during the analysis phase.
  - These include information gathered by…
    - Interviewing end users individually and in a group
    - Questionnaire survey
    - Direct observation
    - Examining different documents
- The basic E-R model is graphically depicted and presented for review.
- The process is repeated until the end users and designers agree that the E-R diagram is a fair representation of the organization's activities and functions.
  - Checking for Redundant Relationships in the ER Diagram. Relationships between entities indicate access from one entity to another - it is therefore possible to access one entity occurrence from another entity occurrence even if there are other entities and relationships that separate them - this is often referred to as *Navigation'* of the ER diagram
  - The last phase in ER modeling is validating an ER Model against requirement of the user.

Graphical Representations in ER Diagramming

ERD

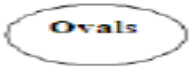
- Entity is represented by a **RECTANGLE** containing the name of the entity.





- Connected entities are called relationship participants

- Attributes are represented by **OVALS** and are connected to the entity by a line.



Attribute

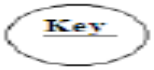
Multi-valued Attribute

Composite Attribute

- A derived attribute is indicated by a **DOTTED LINE**.

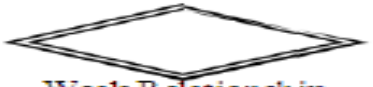
(.....)

- **PRIMARY KEYS** are underlined.



- Relationships are represented by **DIAMOND** shaped symbols
  - Weak Relationship is a relationship between Weak and Strong Entities
  - Strong Relationship is a relationship between two strong Entities

Strong Relationship

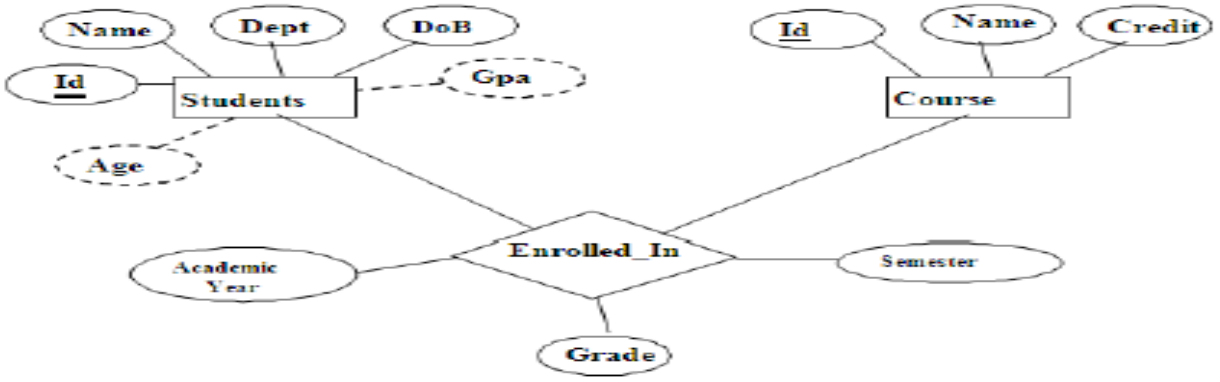


Weak Relationship

ERD Example

Example 1: Build an ER Diagram for the following information:

- A student record management system will have the following two basic data object categories with their own features or properties:
- Students will have an Id, Name, Dept, Age, GPA and
- Course will have an Id, Name, Credit Hours
- Whenever a student enroll in a course in a specific Academic Year and Semester, the Student will have a grade for the course



Example 2: Build an ER Diagram for the following information:

- A Personnel record management system will have the following two basic data object categories with their own features or properties:
  Employee will have an Id, Name, DoB, Age, Tel and Department will have an Id, Name, Location

ERD

- Whenever an Employee is assigned in one Department, the duration of his stay in the respective department should be registered.

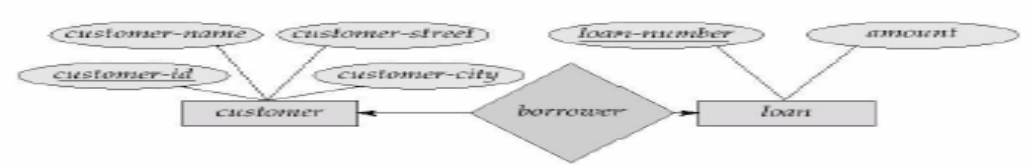
3.3 Structural Constraints on Relationship

1. Constraints on Relationship / Multiplicity/ Cardinality Constraints

- Multiplicity constraint is the number or range of possible occurrence of an entity type/relation that may relate to a single occurrence/tuple of an entity type/relation through a particular relationship.
- Mostly used to insure appropriate enterprise 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*



E.g.: Relationship *Manages* between *STAFF* and *BRANCH*

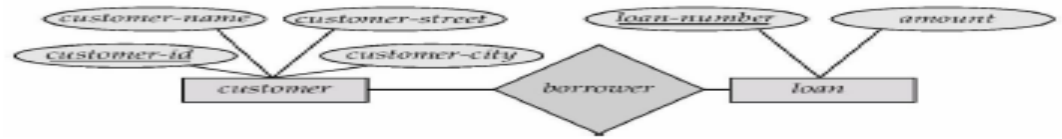
The multiplicity of the relationship is:

- *One branch can only have one manager*
- *One employee could manage either one or no branches*

One-To-Many Relationships

- 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*

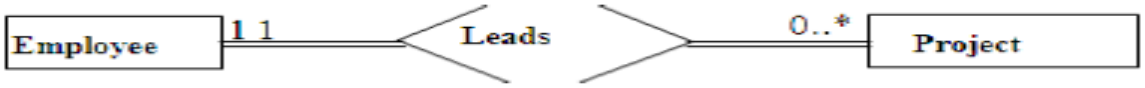
ERD



E.g.: Relationship *Leads* between *STAFF* and *PROJECT*

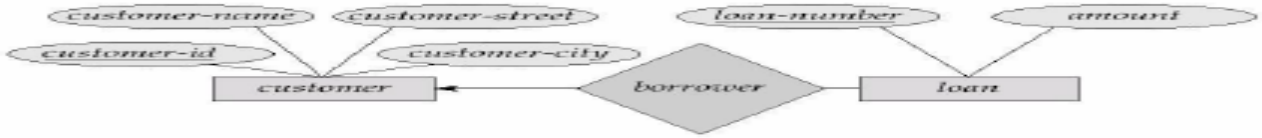
The multiplicity of the relationship

- *One staff may Lead one or more project(s)*
- *One project is Lead by one staff*



Many-To-Many Relationship

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



E.g.: Relationship *Teaches* between *INSTRUCTOR* and *COURSE*

The multiplicity of the relationship

One Instructor Teaches one or more Course(s)

One Course Thought by Zero or more Instructor(s)

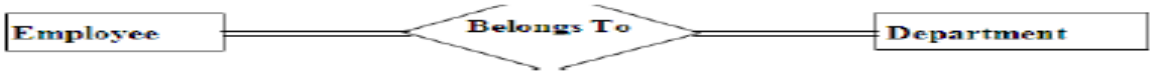
ERD

3.4 Participation of an Entity Set in a Relationship Set

Participation constraint of a relationship is involved in identifying and setting the mandatory or optional feature of an entity occurrence to take a role in a relationship. There are two distinct participation constraints with this respect, namely: *Total Participation* and *Partial Participation*

- **Total participation**: every tuple in the entity or relation participates in at least one relationship by taking a role. This means, every tuple in a relation will be attached with at least one other tuple. The entity with total participation in a relationship will be connected to the relationship using a double line.
- **Partial participation**: some tuple in the entity or relation may not participate in the relationship. This means, there is at least one tuple from that Relation not taking any role in that specific relationship. The entity with partial participation in a relationship will be connected to the relationship using a single line.

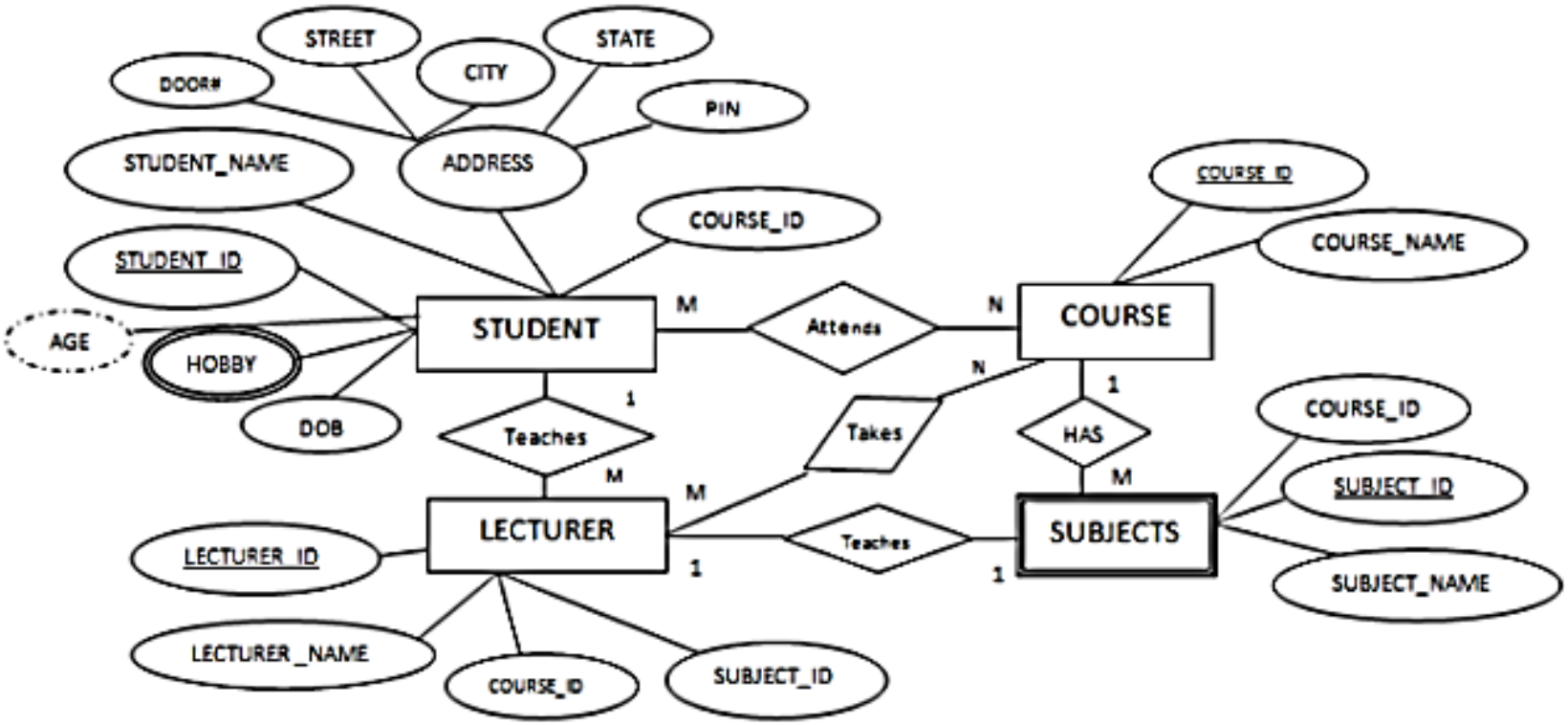
- E.g. 1: Participation of EMPLOYEE in "belongs to" relationship with DEPARTMENT is total since every employee should belong to a department.Participation of DEPARTMENT in "belongs to" relationship with EMPLOYEE is total since every department should have more than one employee.



E.g. 2: Participation of EMPLOYEE in "manages" relationship with DEPARTMENT, is partial participation since not all employees are managers. Participation of DEPARTMENT in "Manages" relationship with EMPLOYEE is total since every department should have a manager.

Mapping ERD into Relational model

- converting the conceptual design to a form suitable for relational logical model, which is in a form of tables.



Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into tables

- 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 diagram, STUDENT, COURSE, LECTURER and SUBJECTS forms individual tables

Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into 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

Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into tables

- 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.

Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into tables

- 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
  - STUD_HOBBY(**student_id**, **hobby**,…)

Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into tables

- 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.

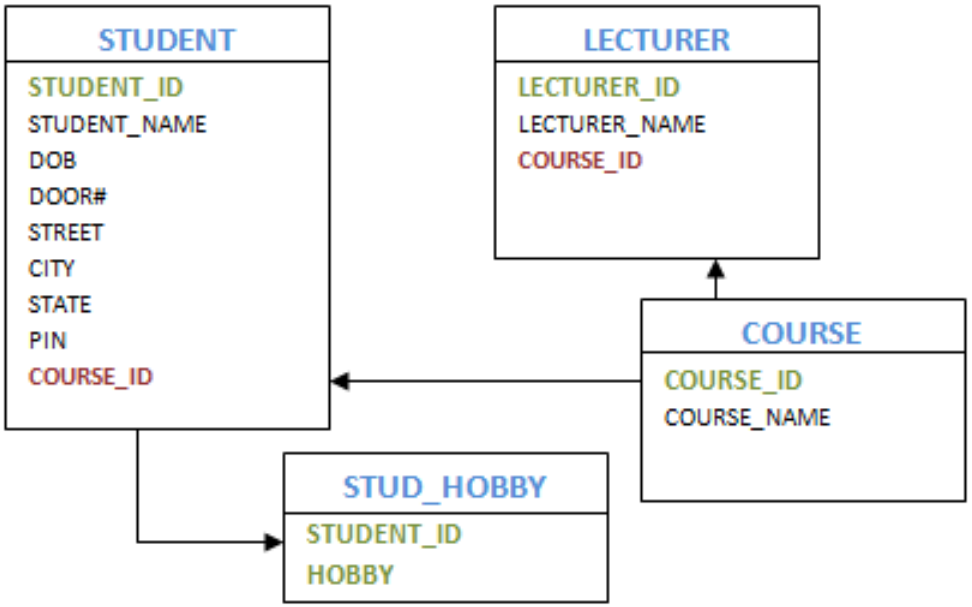
Mapping ERD into Relational model

1. The basic rule for converting the ER diagrams into tables

- 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 yearOfBirth and current year.
  - Hence we need not create a column for this attribute.
  - It reduces the duplicity in the database.

Mapping ERD into Relational model

- The above ERD could be converted into Relationship Diagram as follows



Mapping ERD into Relational model

2. Let us see some of the special cases

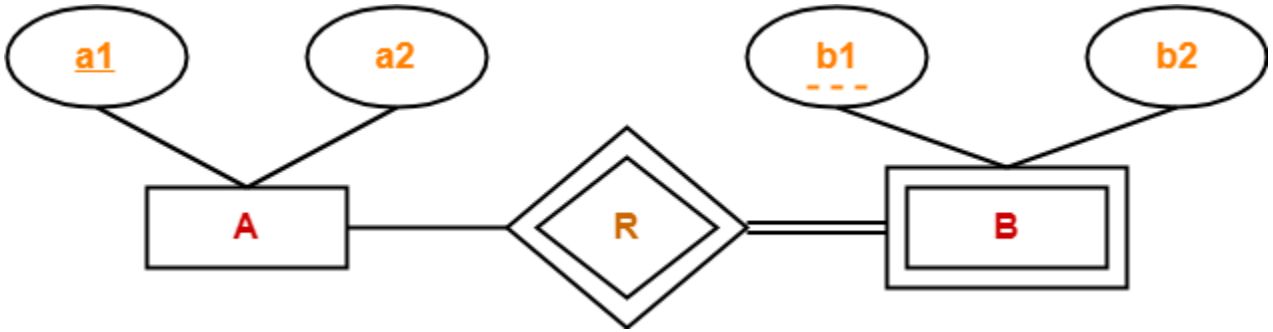
❑ **Converting Weak Entity**

- Weak entity is also represented as table.
- All the attributes of the weak entity forms the column of the table.
- But the key attribute represented in the diagram cannot form the primary key of this table.
- We have to add a foreign key column, which would be the primary key column of its strong entity.
  - This foreign key column along with its key attribute column forms the primary key of the table.

Mapping ERD into Relational model

2. Let us see some of the special cases

- Weak entity set always appears in association with identifying relationship with total participation



- In this case two tables are required

1. A(a1, a2)
2. BR(a1, b1, b2)

Mapping ERD into Relational model

2. Let us see some of the special cases

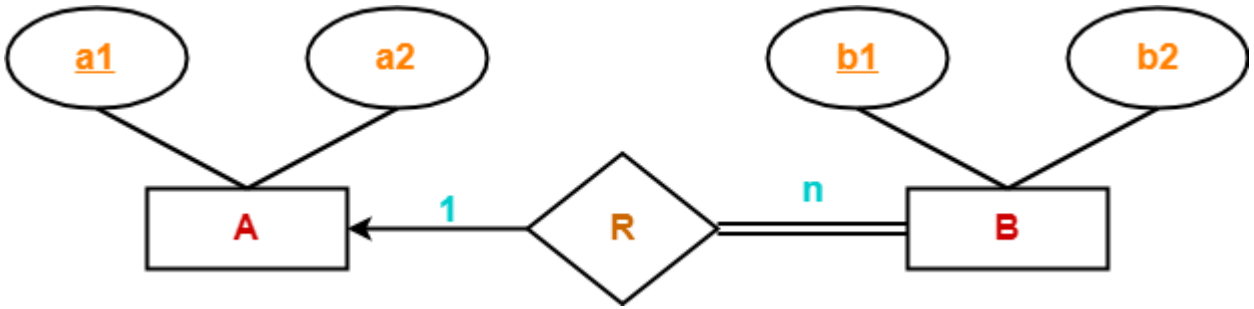
- In our example above, SUBJECTS is the weak entity.
- Although, SUBJECT_ID is represented as key attribute in the diagram, it cannot be considered as primary key.
- In order to add primary key to the column, we have to find the foreign key first.
- COURSE is the strong entity related to SUBJECT.
  - Hence the primary key COURSE_ID of COURSE is added to SUBJECT table as foreign key.
  - Now we can create a composite primary key out of COURSE_ID and SUBJECT_ID.
  - Course (COURSE_ID, course_name)
  - Subject(key{**COURSE_ID(FK),** subject_id**(partial key)},** subject_name)

Mapping ERD into Relational model

2. Let us see some of the special cases

- **Binary relationship with both cardinality constraints and participation constraints**

Case-1:Binary relationship with cardinality and total participation constraint from one side



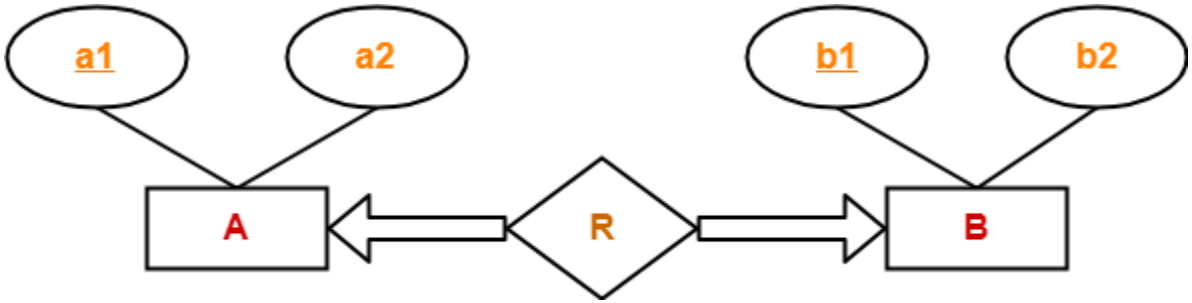
Combine the entity set B and relationship set R then two tables will be required:

1. A(a1, a2)
2. BR(a1, b1, b2)

Mapping ERD into Relational model

2. Let us see some of the special cases

- **Case :Binary relationship with cardinality constraint and total participation from both sides**

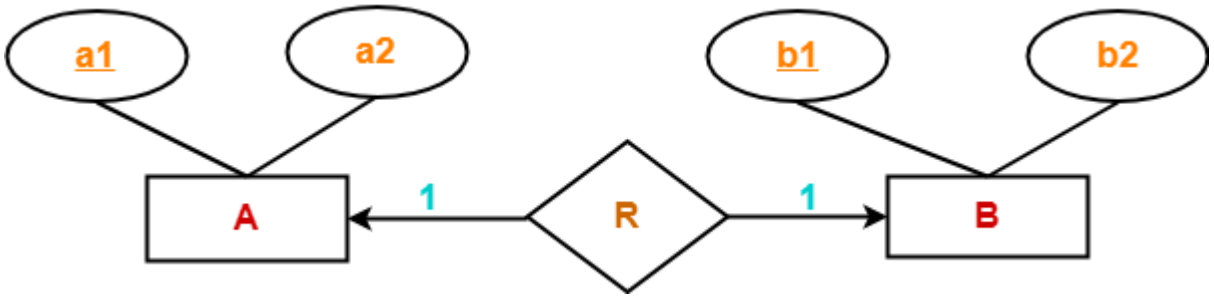


- Here we need a single table ARB(a1,a2,b1,b2)
- Both a1 and b1 together create primary key

Mapping ERD into Relational model

2. Let us see some of the special cases

❑Representing 1:1 relationship



Either of the Two tables are needed

Case 1:
1. AR(a1, a2, b1)
2. B(b1, b2)

Case 2:
1. A(a1, a2)
2. BR(a1, b1, b2)

Mapping ERD into Relational model

2. Let us see some of the special cases

- Imagine SUBJECT is not a weak entity, and we have LECTURER teaches SUBJECT relation.
  - It is a 1:1 relation. i.e.; one lecturer teaches only one subject.

❑ We can represent this case in two ways

Case 1:Create table for both LECTURER and SUBJECT.

- Add the primary key of LECTURER in SUBJECT table as foreign key.
  - It implies the lecturer name for that particular subject.

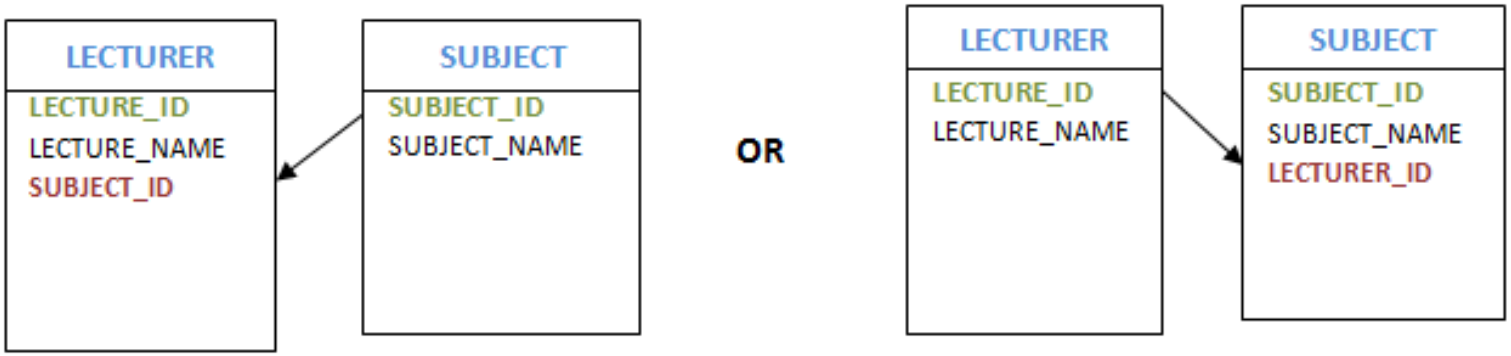
Case 2: Create table for both LECTURER and SUBJECT.

- Add the primary key of SUBJECT in LECTURER table as foreign key.
  - It implies the subject taught by the lecturer.

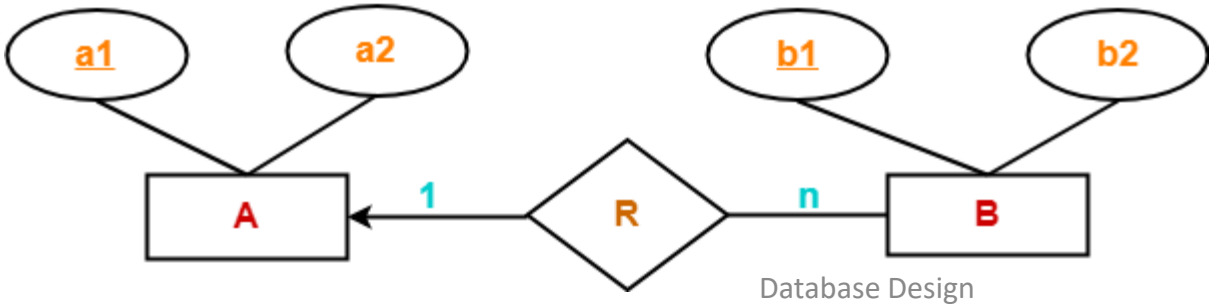
Mapping ERD into Relational model

2. Let us see some of the special cases

- In both the case, meaning is same.
  - Foreign key column can be added in either of the table, depending on the developer's choice.



- **Representing 1:N relationship**

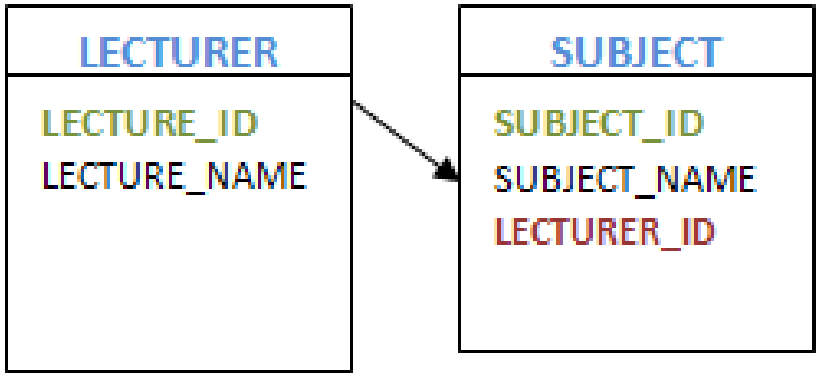


Two tables required
1. A(a1, a2)
2. BR(a1, b1, b2)

Mapping ERD into Relational model

2. Let us see some of the special cases

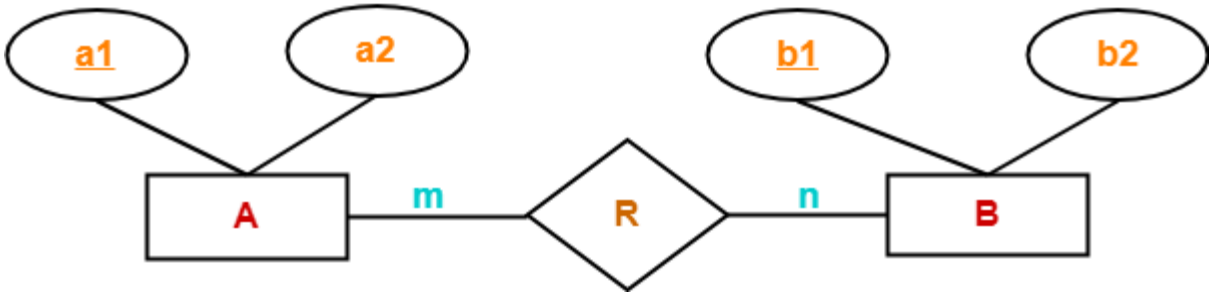
- Consider SUBJECT and LECTURER relation, where each Lecturer teaches multiple subjects.
- This is a 1: N relation.
- In this case, primary key of LECTURER table is added to the SUBJECT table.
  - i.e.: the primary key at 1 cardinality entity is added as foreign key to N



Mapping ERD into Relational model

2. Let us see some of the special cases

❑Representing M:N relationship



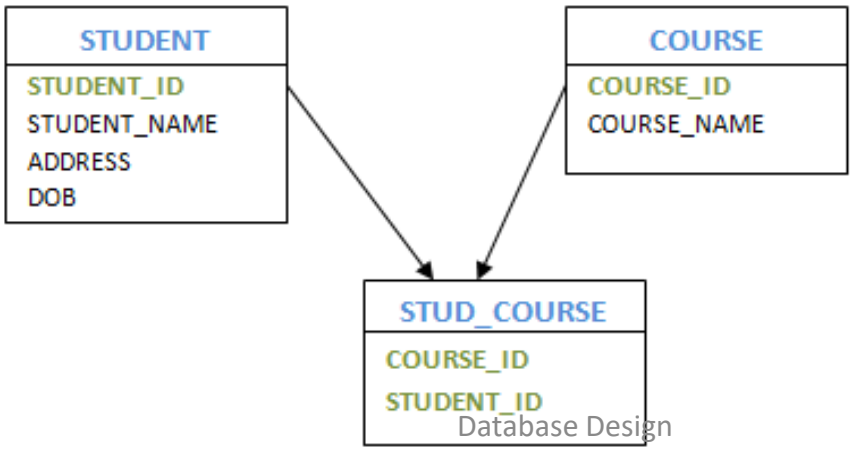
Three tables will be needed

1. A(a1, a2)
2. R(a1, b1)
3. B(b1,b2)

Mapping ERD into Relational model

2. Let us see some of the special cases

- Consider the example, multiple students enrolled for multiple courses,
  - which is M:N relation.
- In this case,
  - we create STUDENT and COURSE tables for the entities.
  - Create one more table for the relation between them and name it as STUD_COURSE.
  - Add the primary keys of COURSE and STUDENT which will become foreign key into it along with some additional attributes (if applicable) ,
    - which forms the composite primary key of the new table.
      - Student (student_id, student_name, …)
      - Course (course_id, course_name,…)



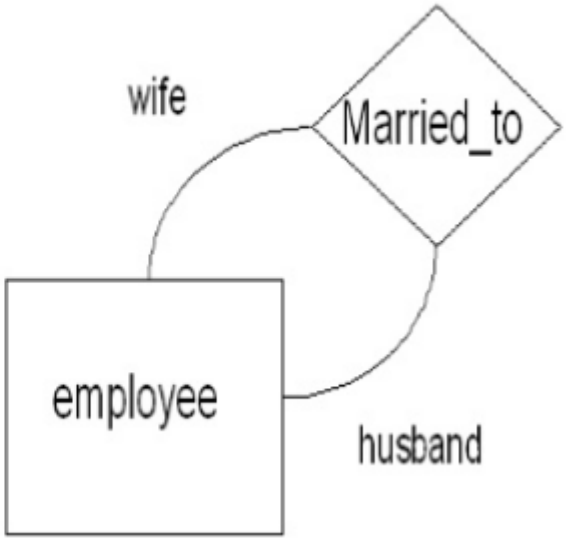
Mapping ERD into Relational model

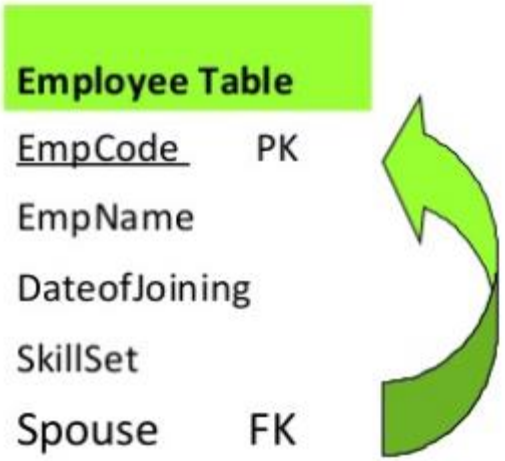
2. Let us see some of the special cases

❑Self referencing 1:1

- Consider employees who are also a couple
- The primary key field itself will become foreign key in the same table

Employee(E#, Name,... **Spouse**)



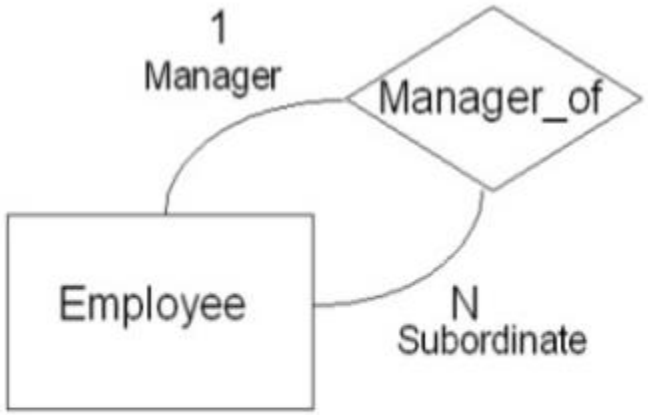


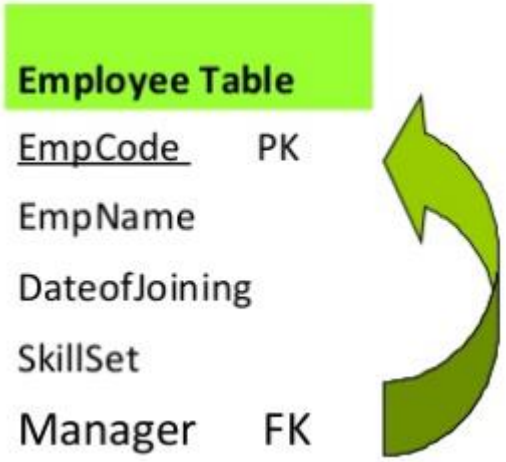
Mapping ERD into Relational model

2. Let us see some of the special cases

❑Self referencing 1:N

- The primary key field itself will become foreign key in the same table
- Same as unary 1:1



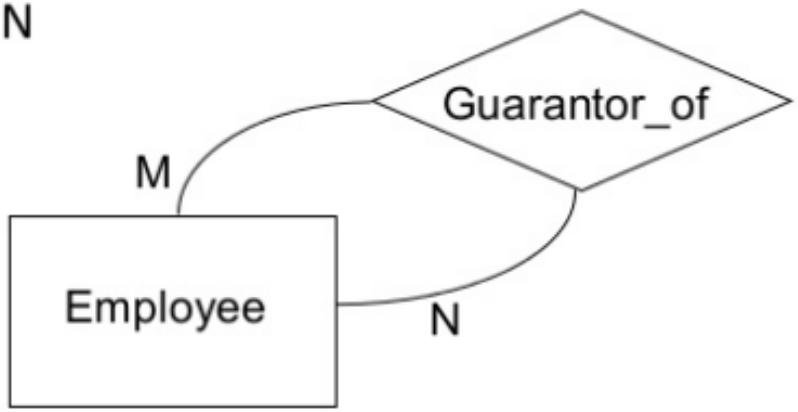


Employee(E#, Name,…,Manager)

Mapping ERD into Relational model

2. Let us see some of the special cases

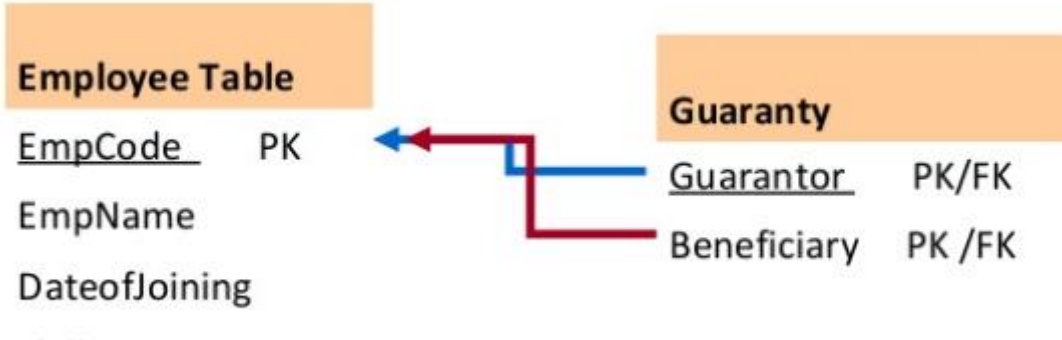
Self referencing M:N



- There will be two resulting tables. One to represent the entity and another to represent the M:N relationship as follows

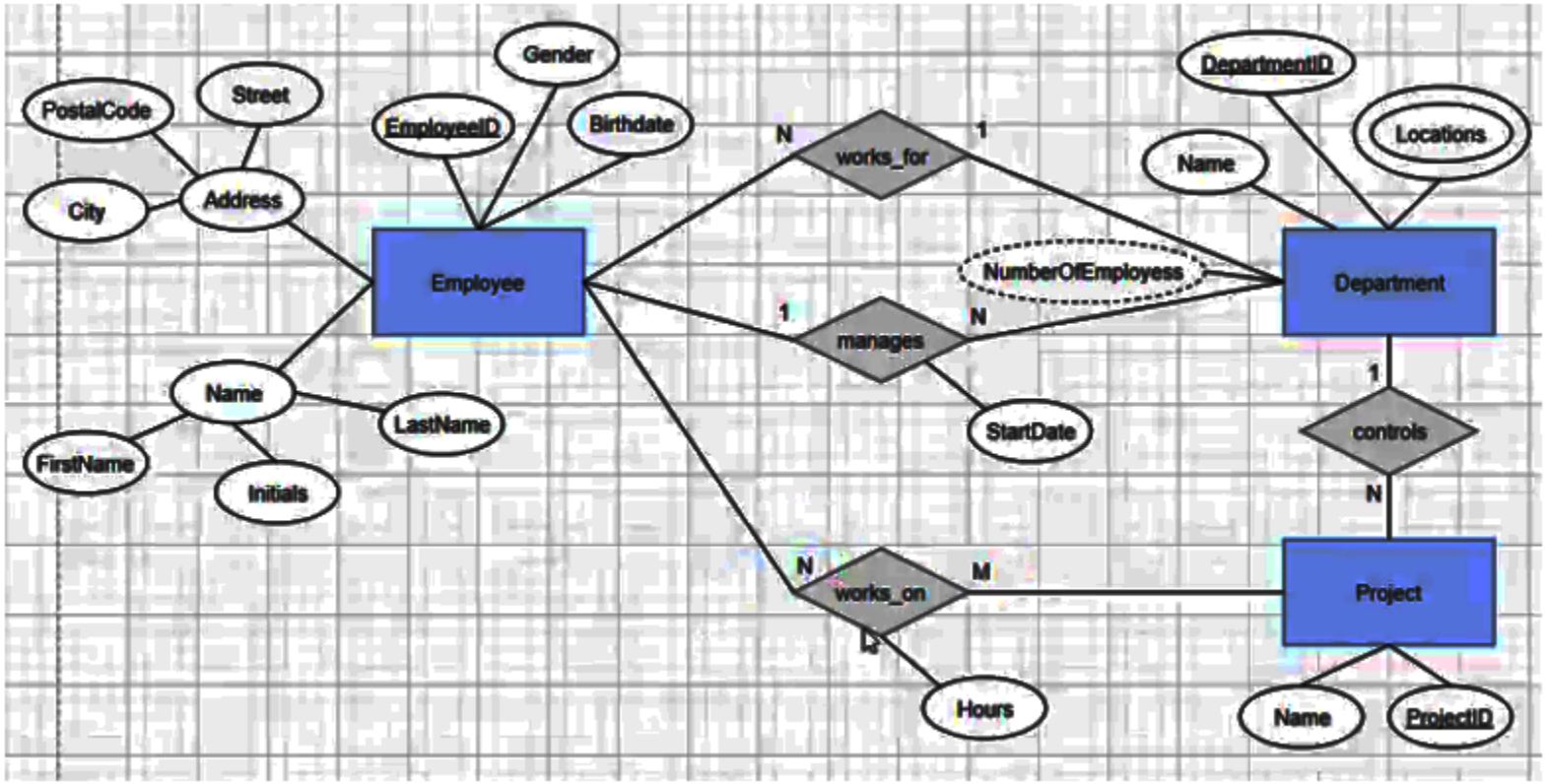
Employee(E#, Name,…)

Guaranty(Guarantor, beneficiary)



Exercise

- Convert the following ERD into Relationship diagram



- Possible Relationship diagram for the ERD on the previous slide