Data Modeling

Database Design

Department of Computer Engineering

Sharif University of Technology

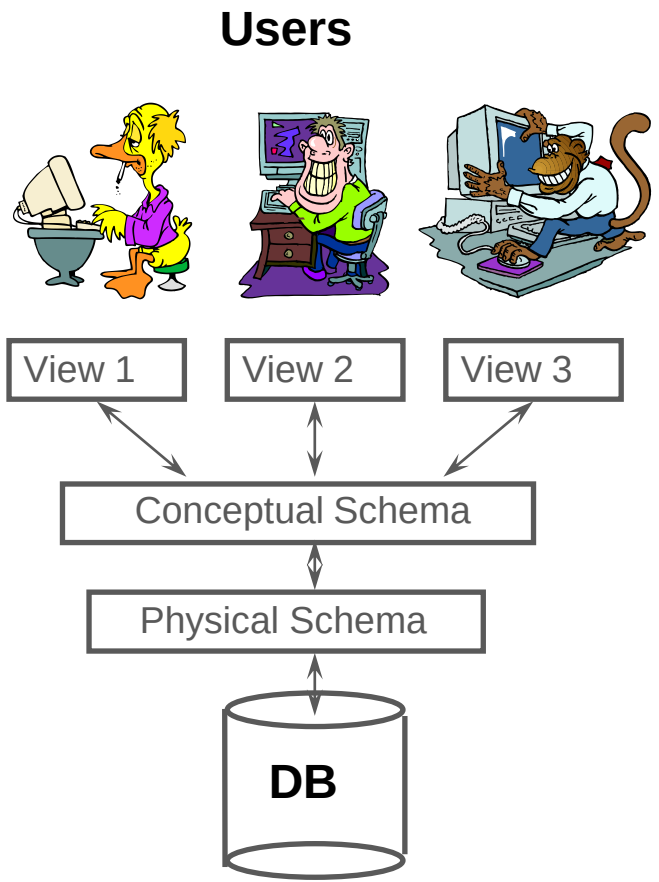
Maryam Ramezani maryam.ramezani@sharif.edu

The Big Picture

- Data Modelling
  - E-R
  - Relational
- Storing Data
  - File Indexes
  - Buffer Pool Management
- Query Languages
  - SQL
  - Relational Algebra
  - Relational Calculus
- Query Optimization
  - External Sorting
  - Join Algorithms
  - Query Plans, Cost Estimation

Levels of Abstraction

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.



Overview Database Design Methodology

Conceptual database design

- **Step 1 Build conceptual data model**
  - Step 1.1 Identify entity types
  - Step 1.2 Identify relationship types
  - Step 1.3 Identify and associate attributes with entity or relationship types
  - Step 1.4 Determine attribute domains
  - Step 1.5 Determine candidate, primary, and alternate key attributes
  - Step 1.6 Consider use of enhanced modeling concepts (optional step)
  - Step 1.7 Check model for redundancy
  - Step 1.8 Validate conceptual model against user transactions
  - Step 1.9 Review conceptual data model with user

Overview Database Design Methodology

Logical database design for the relational model

- **Step 2 Build and validate logical data model**
  - Step 2.1 Derive relations for logical data model
  - Step 2.2 Validate relations using normalization
  - Step 2.3 Validate relations against user transactions
  - Step 2.4 Define integrity constraints
  - Step 2.5 Review logical data model with user
  - Step 2.6 Merge logical data models into global model (optional step)
  - Step 2.7 Check for future growth

Overview Database Design Methodology

Physical database design for relational database

- **Step 3 Translate logical data model for target DBMS**
  - Step 3.1 Design base relations
  - Step 3.2 Design representation of derived data
  - Step 3.3 Design general constraints

- **Step 4 Design file organizations and indexes**
  - Step 4.1 Analyze transactions
  - Step 4.2 Choose file organization
  - Step 4.3 Choose indexes
  - Step 4.4 Estimate disk space requirements

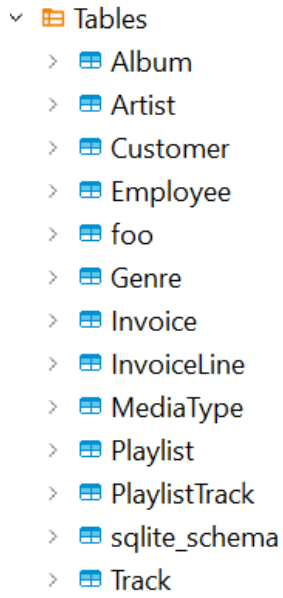
Overview Database Design Methodology

- Step 5 Design user views
- Step 6 Design security mechanisms
- Step 7 Consider the introduction of controlled redundancy
- Step 8 Monitor and tune the operational system

7

Data Modeling

- Components of Database Environment
  - Hardware
  - Software
  - User
  - Data
    - User
    - System



	type	name	tbl_name	rootpage	sql
1	view	test	test	0	CREATE VIEW test AS SELECT CURRENT_TIMESTAMP as ct
2	view	EmpView	EmpView	0	CREATE VIEW EmpView as ¶select * from Employee
3	table	Album	Album	2	CREATE TABLE [Album]¶(¶ [AlbumId] INTEGER NOT NULL,¶ [Title] NVARCHAR(160) NOT NULL,¶ [ArtistId] INTEC
4	table	Artist	Artist	3	CREATE TABLE [Artist]¶(¶ [ArtistId] INTEGER NOT NULL,¶ [Name] NVARCHAR(120),¶ CONSTRAINT [PK_Artist] PR
5	table	Customer	Customer	4	CREATE TABLE [Customer]¶(¶ [CustomerId] INTEGER NOT NULL,¶ [FirstName] NVARCHAR(40) NOT NULL,¶ [Las
6	table	Employee	Employee	7	CREATE TABLE [Employee]¶(¶ [EmployeeId] INTEGER NOT NULL,¶ [LastName] NVARCHAR(20) NOT NULL,¶ [Firs
7	table	Genre	Genre	9	CREATE TABLE [Genre]¶(¶ [GenreId] INTEGER NOT NULL,¶ [Name] NVARCHAR(120),¶ CONSTRAINT [PK_Genre] P
8	table	Invoice	Invoice	10	CREATE TABLE [Invoice]¶(¶ [InvoiceId] INTEGER NOT NULL,¶ [CustomerId] INTEGER NOT NULL,¶ [InvoiceDate] D
9	table	InvoiceLine	InvoiceLine	12	CREATE TABLE [InvoiceLine]¶(¶ [InvoiceLineId] INTEGER NOT NULL,¶ [InvoiceId] INTEGER NOT NULL,¶ [TrackId] I
10	table	MediaType	MediaType	14	CREATE TABLE [MediaType]¶(¶ [MediaTypeId] INTEGER NOT NULL,¶ [Name] NVARCHAR(120),¶ CONSTRAINT [P
11	table	Playlist	Playlist	15	CREATE TABLE [Playlist]¶(¶ [PlaylistId] INTEGER NOT NULL,¶ [Name] NVARCHAR(120),¶ CONSTRAINT [PK_Playlis
12	table	PlaylistTrack	PlaylistTrack	16	CREATE TABLE [PlaylistTrack]¶(¶ [PlaylistId] INTEGER NOT NULL,¶ [TrackId] INTEGER NOT NULL,¶ CONSTRAINT [
13	table	Track	Track	19	CREATE TABLE [Track]¶(¶ [TrackId] INTEGER NOT NULL,¶ [Name] NVARCHAR(200) NOT NULL,¶ [AlbumId] INTEC
14	table	foo	foo	1,067	CREATE TABLE foo¶ (¶ bar int,¶ baz varchar(20)¶)
15	index	sqlite_autoin	PlaylistTrack	17	[NULL]
16	index	IPK_Album	Album	21	CREATE UNIQUE INDEX [IPK_Album] ON [Album]([AlbumId])
17	index	IPK_Artist	Artist	22	CREATE UNIQUE INDEX [IPK_Artist] ON [Artist]([ArtistId])
18	index	IPK_Custome	Customer	23	CREATE UNIQUE INDEX [IPK_Customer] ON [Customer]([CustomerId])
19	index	IPK_Employe	Employee	24	CREATE UNIQUE INDEX [IPK_Employee] ON [Employee]([EmployeeId])
20	index	IPK_Genre	Genre	26	CREATE UNIQUE INDEX [IPK_Genre] ON [Genre]([GenreId])
21	index	IPK_Invoice	Invoice	27	CREATE UNIQUE INDEX [IPK_Invoice] ON [Invoice]([InvoiceId])
22	index	IPK_InvoiceLi	InvoiceLine	28	CREATE UNIQUE INDEX [IPK_InvoiceLine] ON [InvoiceLine]([InvoiceLineId])
23	index	IPK_MediaTy	MediaType	29	CREATE UNIQUE INDEX [IPK_MediaType] ON [MediaType]([MediaTypeId])
24	index	IPK_Playlist	Playlist	30	CREATE UNIQUE INDEX [IPK_Playlist] ON [Playlist]([PlaylistId])
25	index	IPK_PlaylistTr	PlaylistTrack	31	CREATE UNIQUE INDEX [IPK_PlaylistTrack] ON [PlaylistTrack]([PlaylistId], [TrackId])
26	index	IPK_Track	Track	32	CREATE UNIQUE INDEX [IPK_Track] ON [Track]([TrackId])
27	index	IFK_AlbumAr	Album	33	CREATE INDEX [IFK_AlbumArtistId] ON [Album] ([ArtistId])
28	index	IFK_Custome	Customer	34	CREATE INDEX [IFK_CustomerSupportRepId] ON [Customer] ([SupportRepId])
29	index	IFK_Employe	Employee	36	CREATE INDEX [IFK_EmployeeReportsTo] ON [Employee] ([ReportsTo])
30	index	IFK_InvoiceCu	Invoice	37	CREATE INDEX [IFK_InvoiceCustomerId] ON [Invoice] ([CustomerId])

How to Build a DB Application

- Pick an application
- Figure out what to model (ER model)
  - Output: ER diagram
- Transform the ER diagram to a relational schema
- Refine the relational schema (normalization)
- Now ready to implement the schema and load the data!

Conceptual Design

- 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 or business rules* that hold?
- A database `schema' in the ER Model can be represented pictorially (*ER diagrams*).
- Can map an ER diagram into a relational schema.
  - Traditional ER models
  - Extended or Enhanced ER models

ER Model

- Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design.
- An ERD contains different symbols and connectors that visualize two important information: **The major entities within the system scope**, and the **inter-relationships among these entities**.

ER Model Basics

Entity

- Entity: Thing or object in the real world with an independent existence. Real-world object distinguishable from other objects.
  - "Thing" encompasses a wide range, including abstract and physical concepts, while "Object" usually refers to something tangible and physical.
    - physical existence (e.g: a particular person, car, house, or employee)
    - an object with a conceptual existence (e.g: a company, a job, or a university course).
  - Example:
    - a person/role (e.g. Student)
    - object (e.g. Invoice)
    - concept (e.g. Profile)
    - event (e.g. Transaction)

Attributes of Entity

- Attributes: An entity is described (in DB) using a set of attributes.
  - The particular properties that describe it.
  - Example
    - A student with a particular student number is an entity.
    - A company with a particular registration number is an entity.
  - Types:
    - Composite
    - Simple (Atomic)
    - Single-Valued
    - Multivalued
    - Stored
    - Derived
    - Null
    - Complex

Types of Attributes of Entity in ER Model

Composite vs Simple (Atomic)

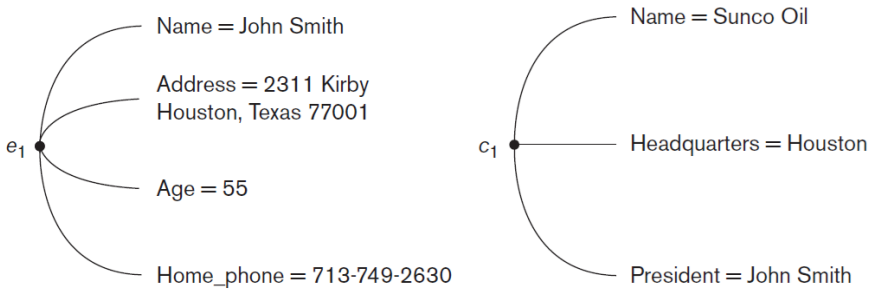
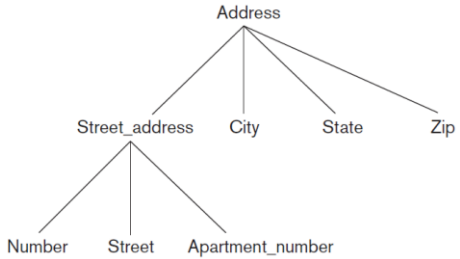


Figure 3.3
Two entities, EMPLOYEE e_1, and COMPANY c_1, and their attributes.

- Composite attributes can form a hierarchy.



- Note: If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes

Types of Attributes of Entity in ER Model

- **Single-Valued vs Multivalued Attributes**
  - Example:
    - Age is a single-valued attribute of a person.
    - People can have different numbers of values for the College_degrees attribute.
    - Two-tone cars have two color values
  - A multivalued attribute **may have lower and upper bounds** to constrain the number of values allowed for each individual entity.
    - The Colors attribute of a car may be restricted to have between one and two values, if we assume that a car can have two colors at most.

Types of Attributes of Entity in ER Model

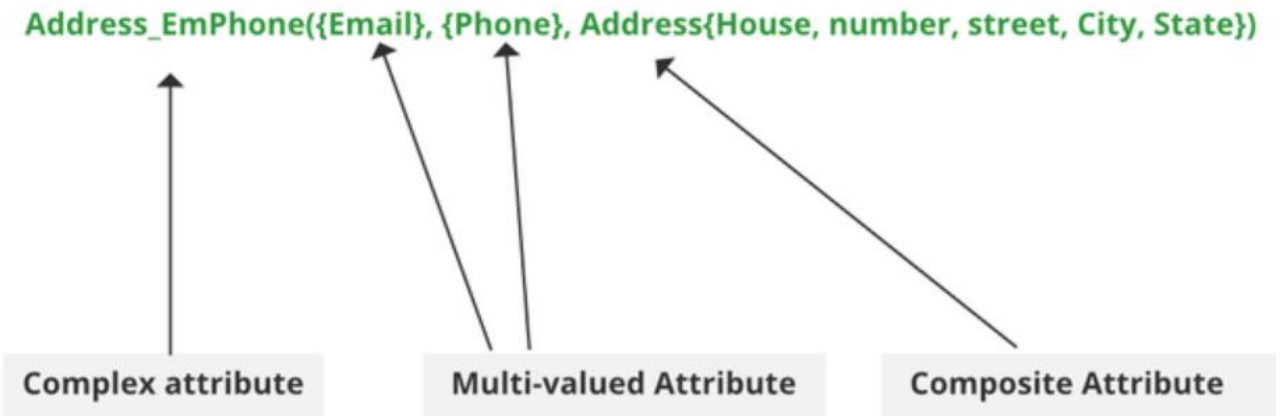
- **Stored vs Derived Attributes:** The stored attribute are those attribute which doesn't require any type of further update since they are stored in the database. In some cases, two (or more) attribute values are related.
  - Example: The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute.
  - **Note: Some attribute values can be derived from related entities:**
    - Example
      - Total and average marks of a student
      - Number_of_employees of a DEPARTMENT entity can be derived by counting the number of employees related to (working for) that department.

Types of Attributes of Entity in ER Model

- **Null Values**
  - Not applicable
    - Person's middle name. Not everyone has a middle name, so in that case, they can be made
  - The attribute value exists but is missing or we don't know it.
    - Data cleaning in data science projects.
  - Can not be known until a certain time
    - "date of death" in a people database

Types of Attributes of Entity in ER Model

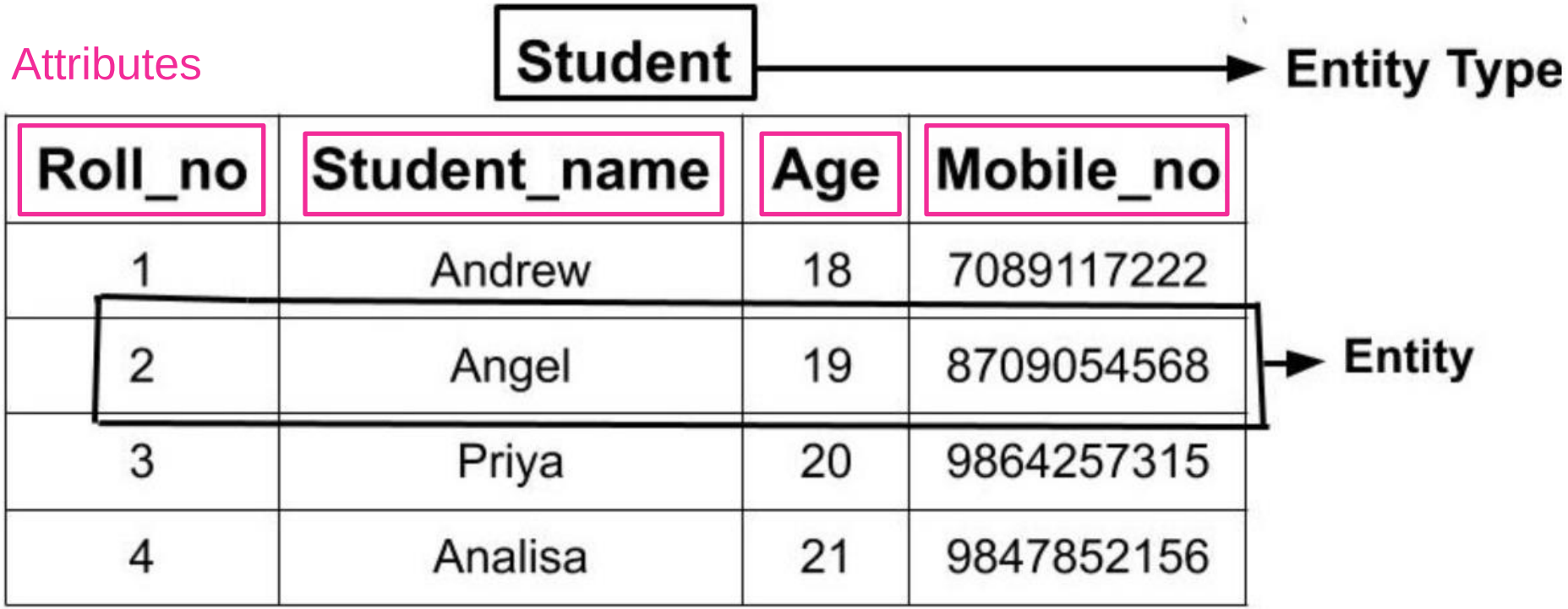
- **Complex Attributes:** composite and multivalued attributes can be nested arbitrarily. These components are grouped between parentheses '( )' and multi-valued attributes between curly braces '{ }', Components are separated by commas ', '.
  - Note: Rarely used in DBMS(DataBase Management System). That's why they are not so popular.



Entity Type

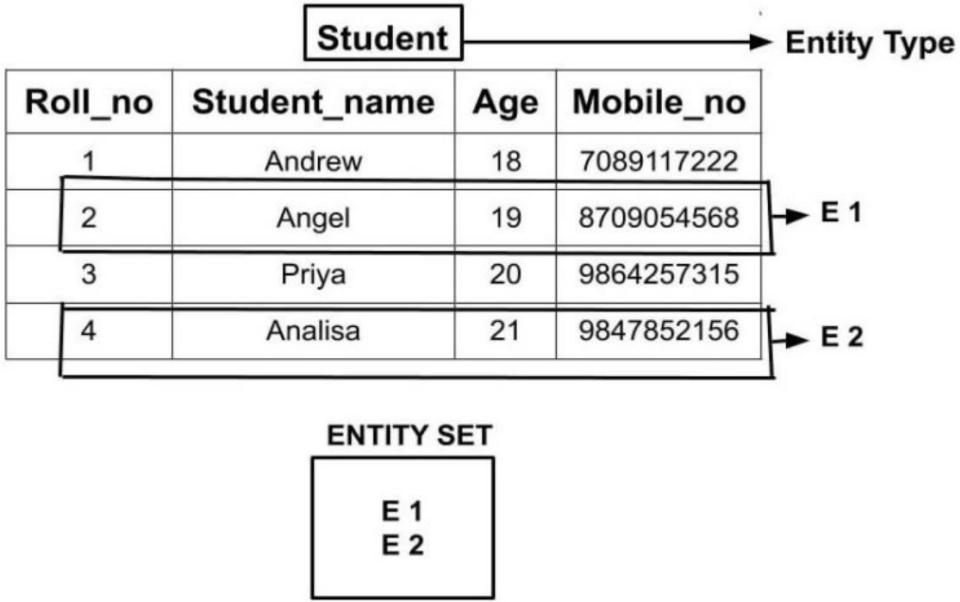
- Entity Type : It refers to the category that a particular entity belongs to. A collection of the entity having similar attributes
  - Example :
    - A table named student in a university database.
    - A table named employee in a company database.

Entity, Attribute, Entity Type



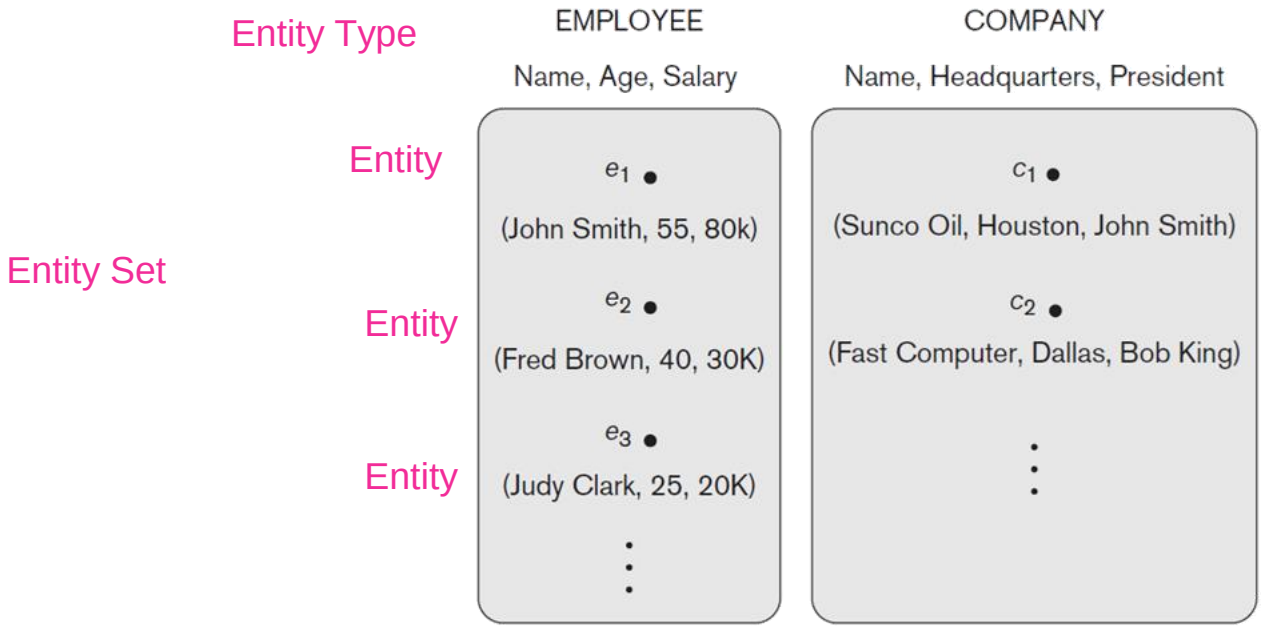
Entity Set

- **Entity Set:** **A collection of entities of the same entity type**. An entity set is a collection or set of all entities of a particular entity type at any point in time. The type of all the entities should be the same.



Entity Set

- All entities in an entity set have the same set of attributes



Entity, Entity Type, Entity Set

Entity	Entity Type	Entity Set
A thing in the real world with independent existence	A category of a particular entity	Set of all entities of a particular entity type.
Any particular row (a record) in a relation(table) is known as an entity.	The name of a relation (table) in RDBMS is an entity type	All rows of a relation (table) in RDBMS is entity set

Attributes of Entity Type

- Each attribute has a domain (Value Sets).
- Are similar to the basic data types available in most programming languages, such as integer, string, Boolean, float, enumerated type, subrange, and so on
- Additional data types to represent common database types, such as date, time, and other concepts, are also employed
- Note: Value sets are not typically displayed in basic ER diagrams. Specified in UML class diagrams.

Composite Attributes

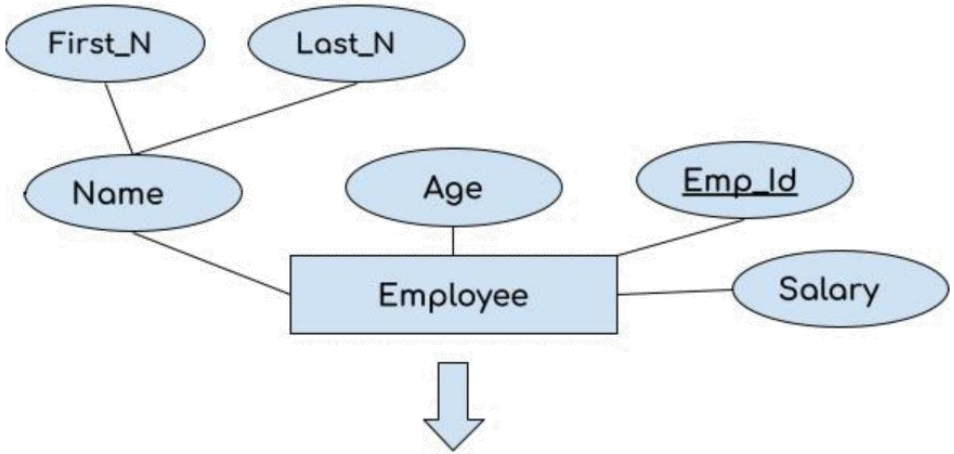
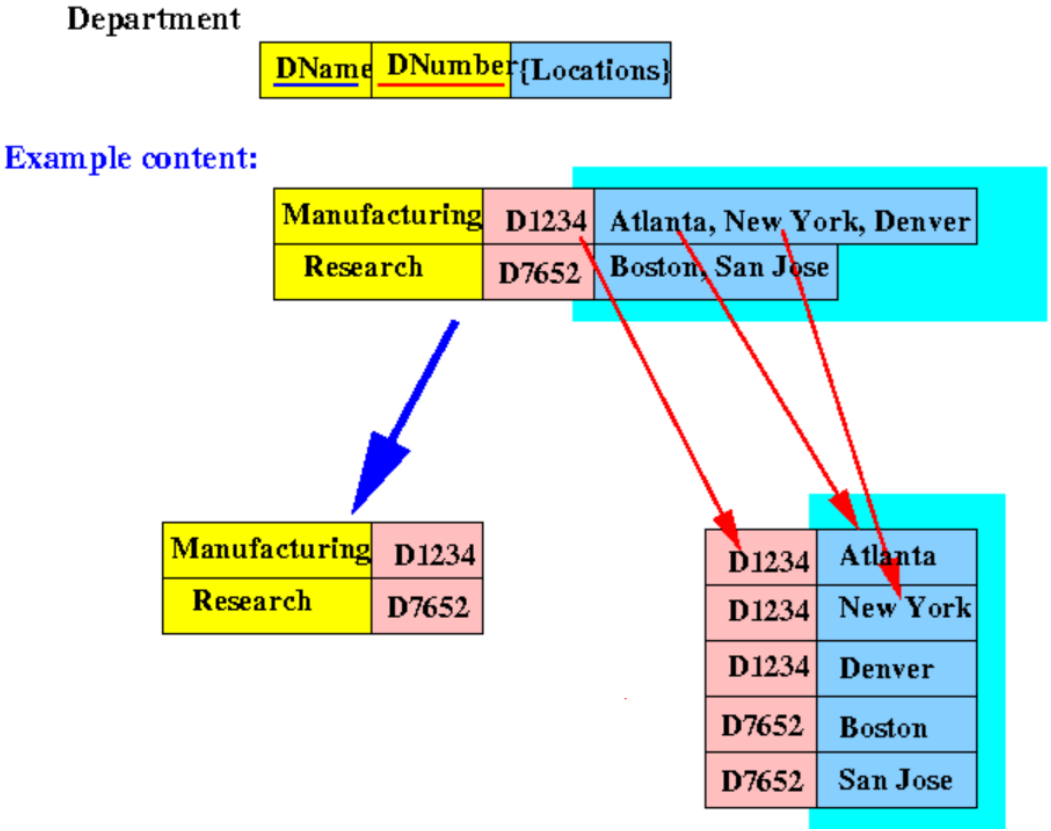


Table Schema: (Emp_id, First_N, Last_N, Age, Salary)

Multi-Valued Attributes



Key Attributes of Entity Type

Key Attributes of an Entity Type:

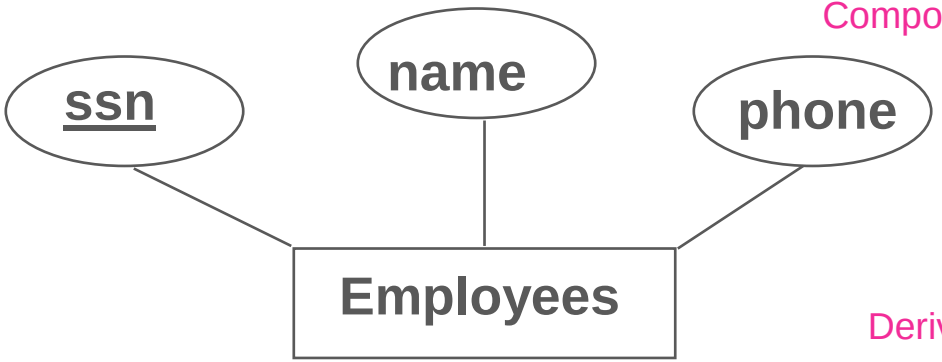
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set.
- Its values can be used to identify each entity uniquely.
- Composite key: Sometimes several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity.
  - Such a composite key must be minimal;

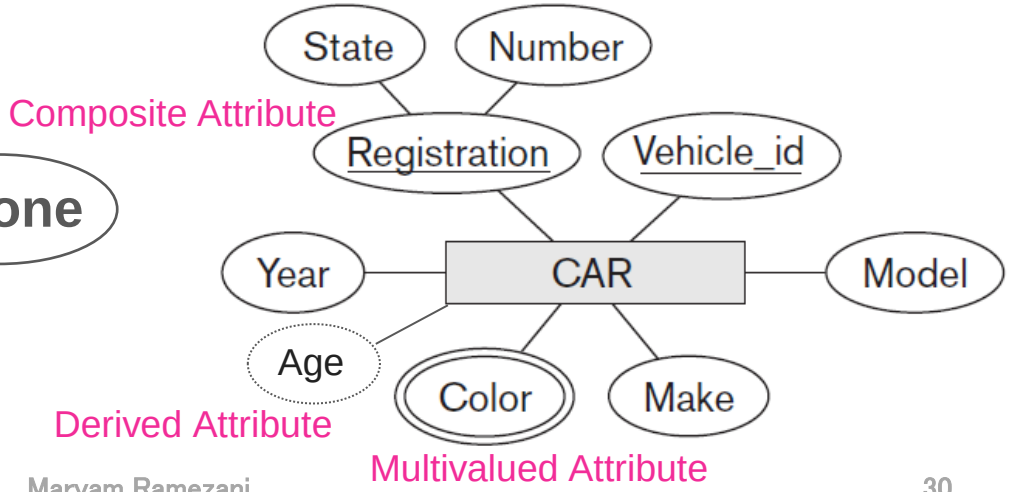
Key Attributes of Entity Type

- Criteria for selecting Identifiers
  - Will not change value
  - Will not be null
  - No intelligent identifiers (containing e.g. locations or people that might change)
  - Substitute new, simple keys for long, composite key.

Entity Type in ER Diagram

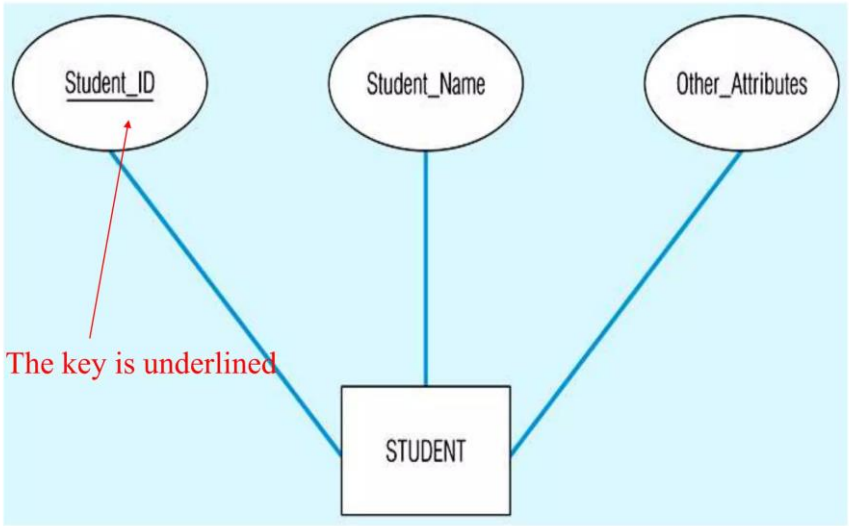
- Entity Type is represented by a **rectangle**.
  - Note: an entity type in the E–R diagram, not entity.
- Attributes of entity type is represented by **oval**.
- Each key attribute has its name **underlined** inside the oval.
- If two attributes are underlined separately, then each is a key on its own.



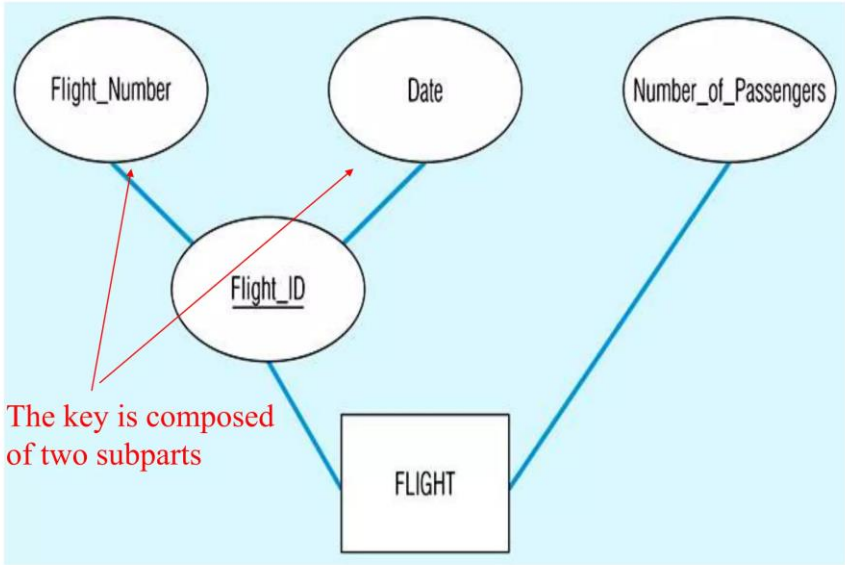


Key Attribute in ER Diagram

Simple key attribute



Composite key attribute



The company is organized into departments:

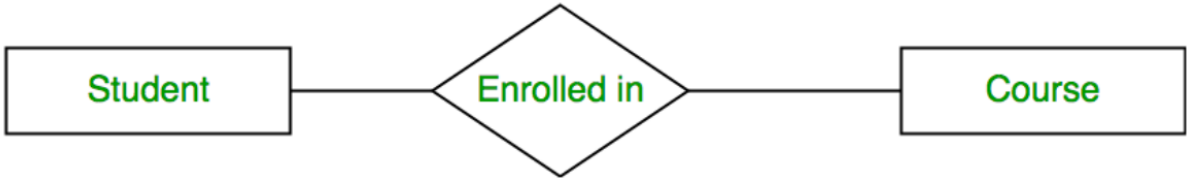
- Each **department** has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number, 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. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

Relationship

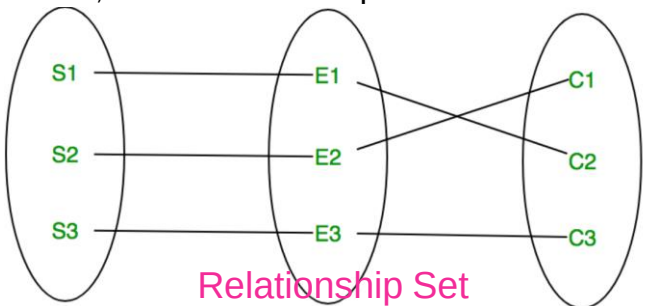
- A relationship indicated how one or more entity classes interact with one and another.
- Each entity plays a role in a relationship.

Relationship

- Relationship Type: Association among two or more entities.
  - In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines

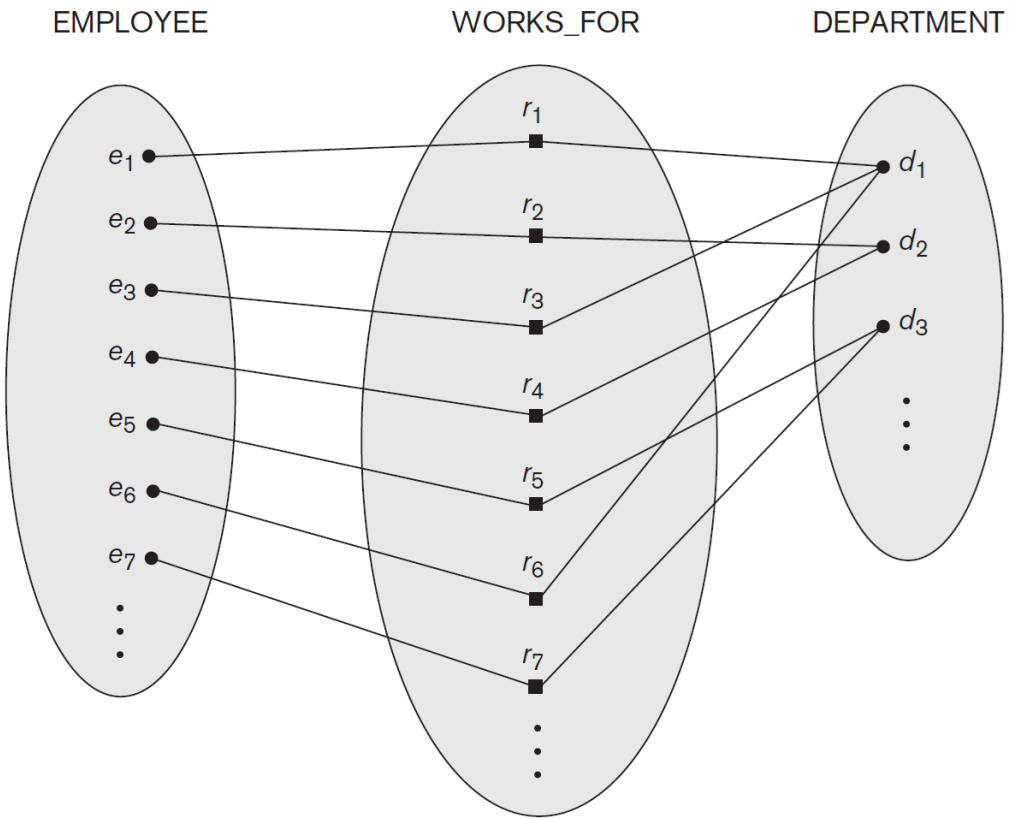


- Relationship Set: A set of relationships of the same type is known as a relationship set.
  - An n-ary relationship set R relates n entity sets E1 … En; each relationship in R involves entities e1, …, en.



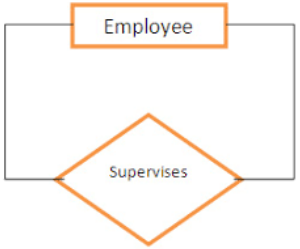
Relationship Set

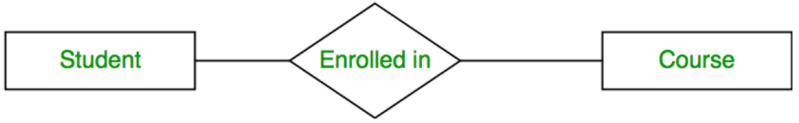
Relationship Set

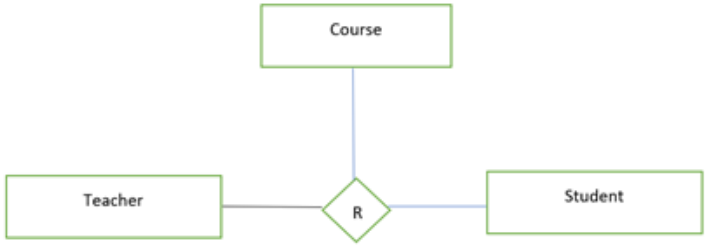


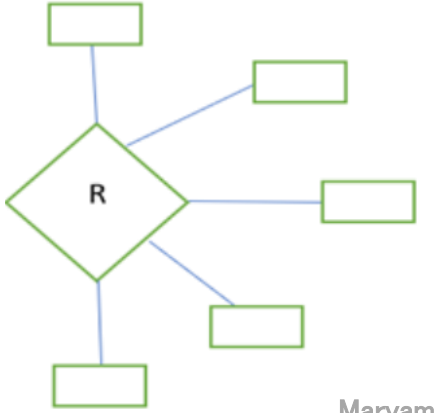
Relationship

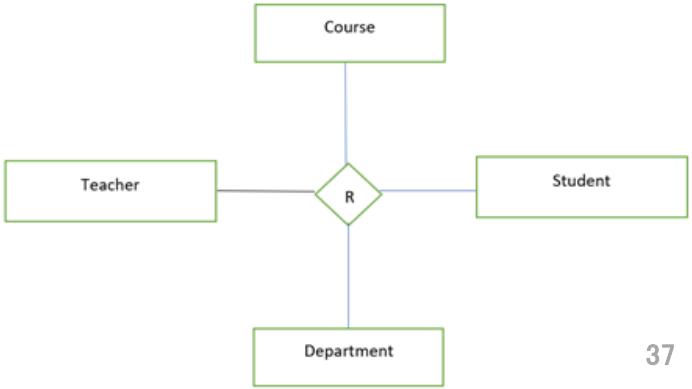
- Degree of Relationship Type: The number of participating entity types
  - Unary: degree one
  - Binary: degree two
  - Ternary: degree three
  - Quaternary: degree four
  - N-ary







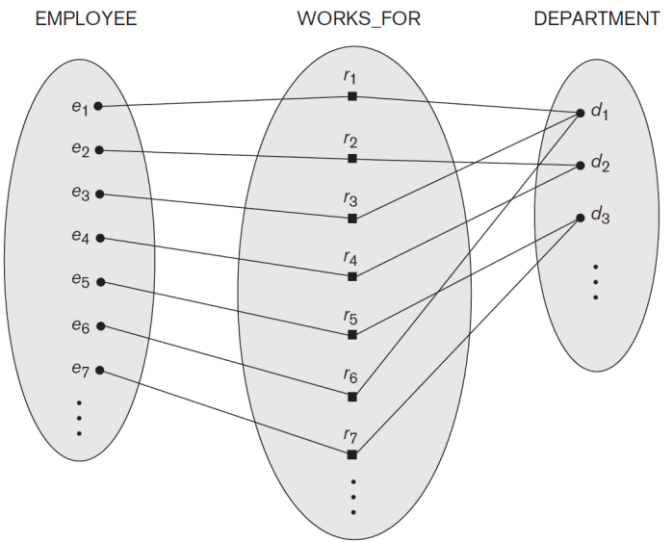




Constraints on Binary Relationship Types

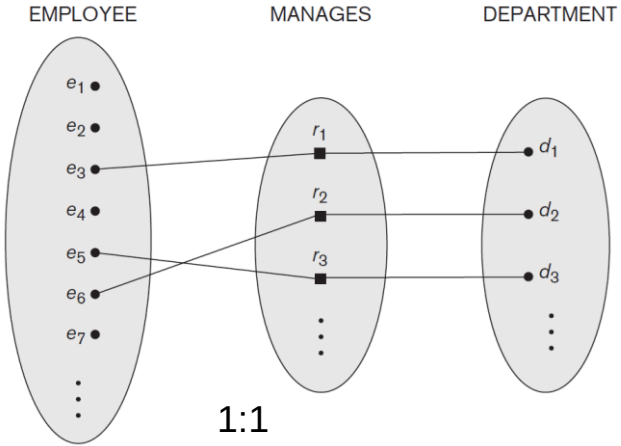
- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. These constraints are determined from the miniworld situation that the relationships represent.

- We can distinguish two main types of binary relationship constraints:
  - Cardinality ratio
  - Participation

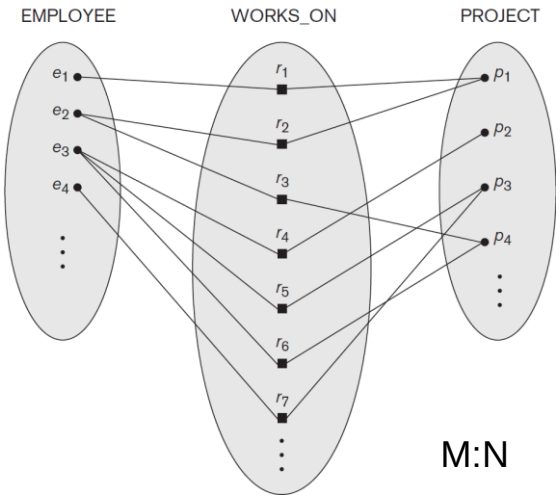


Cardinality Ratios

- Maximum number of relationship instances that an entity can participate in.
- The possible cardinality ratios for binary relationship types are:
  - 1:1
  - M:N
  - 1:N
  - N:1



1:1



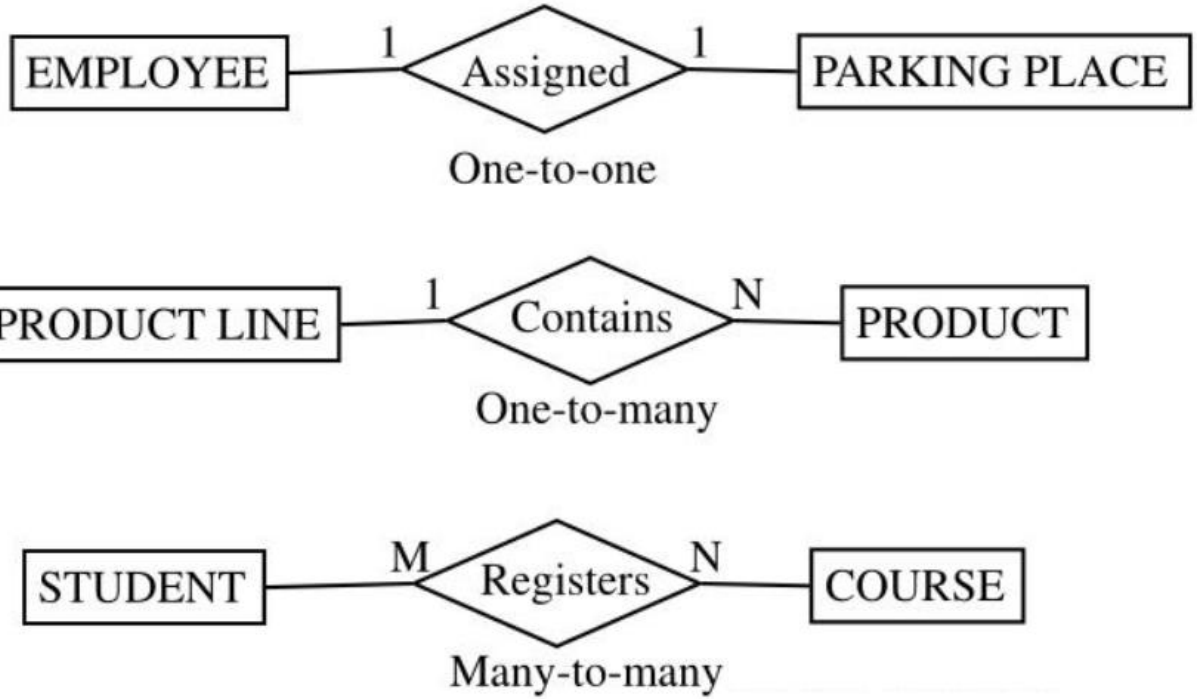
M:N

- One in ER means: zero or one
- Many in ER means: zero or more

Participation Constraints

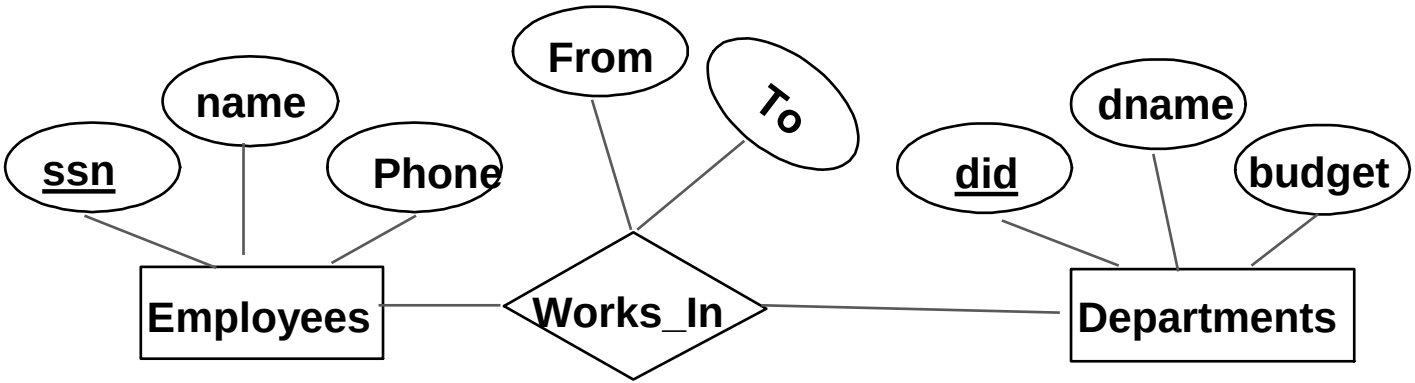
- Whether the existence of an entity depends on its being related to another entity via the relationship type.
  - Total (existence dependency):
    - If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.
    - Meaning that every entity in the total set of employee entities must be related to a department entity via WORKS_FOR.
    - ER: double line connecting the participating entity type to the relationship. A minimum of one.
  - Partial
    - We do not expect every employee to manage a department, so the participation of EMPLOYEE in the most one department and a department can have at most one manager.
    - Meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.
    - ER: single line connecting the participating entity type to the relationship. No minimum.

Relationship



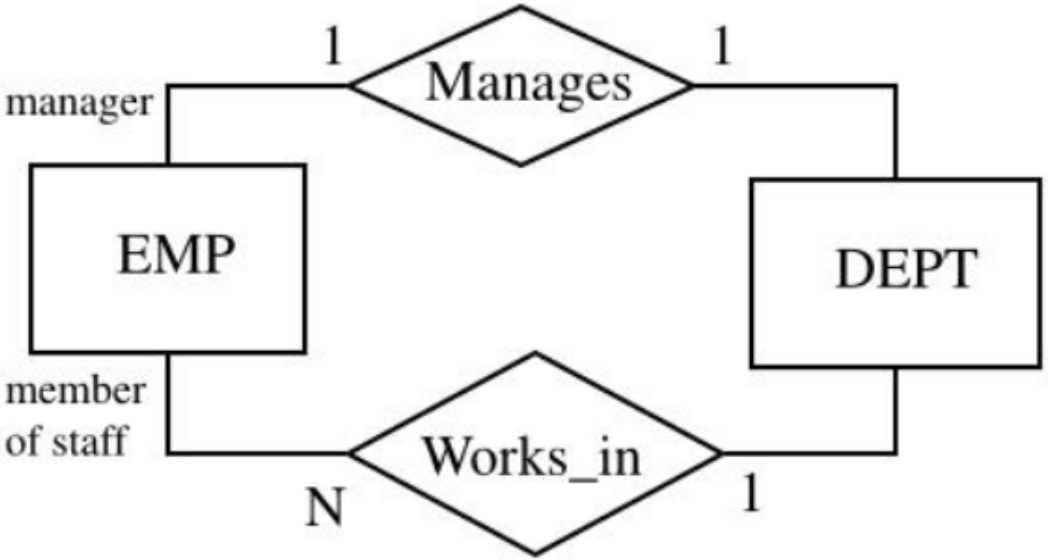
Attributes of Relationship Types

- Relationships can also have **attributes** associated to them. Generally it is not recommended to give attributes to the relationships if not required because while converting the ER model into Relational model, things may get complex and we may require to create a separate table for representing the relationship.



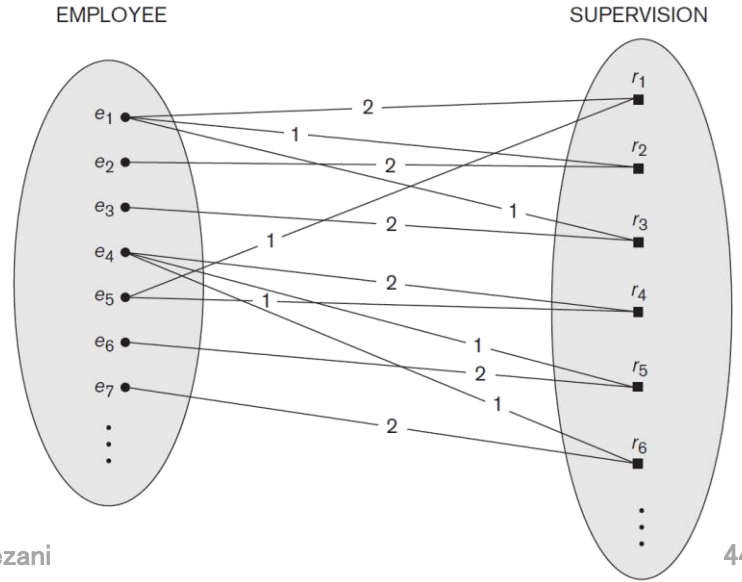
Role Name

- Each entity type that participates in a relationship type plays a particular role in the relationship. Role names may also be used when two entity classes are associated through more than one relationships



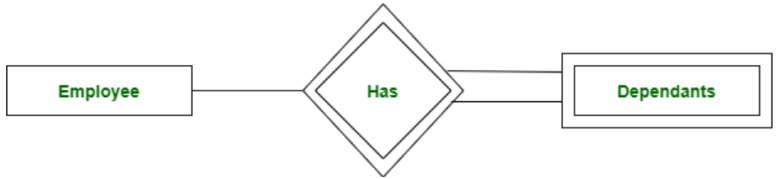
Recursive Relationships (Self-Referencing)

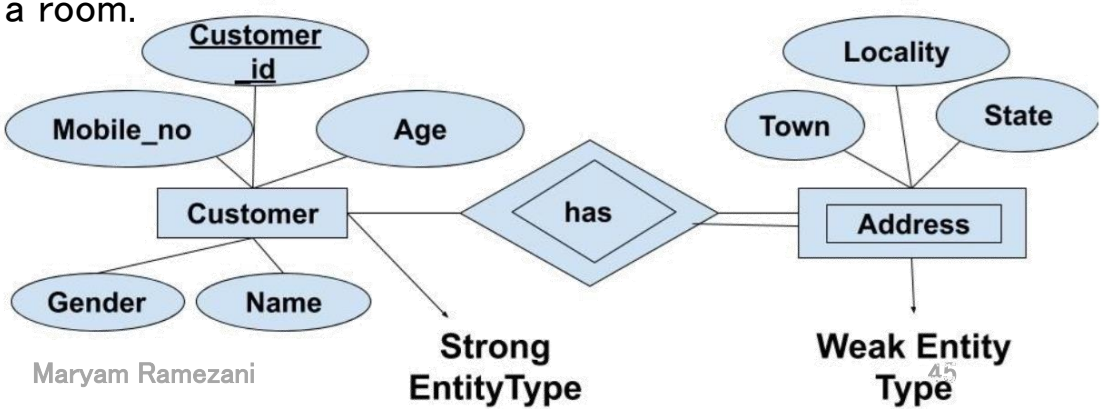
- In some cases the same entity type participates more than once in a relationship type in different roles.
  - The SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity set.
  - A recursive relationship SUPERVISION between EMPLOYEE in
    - (1) supervisor role
    - (2) supervisee role



Types of Entity Type

- **Strong Entity Type**: Has a key attribute which helps in identifying each entity uniquely. It is represented by a rectangle in ER model.
- **Weak Entity Type**: Doesn't have a key attribute. Weak entity type can't be identified on its own. It depends upon some other strong entity for its distinct identity. It is represented by a double outlined rectangle in ER model.
  - Relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.
    - Example: There can be a room only if building exits. There can be no independent existence of a room.



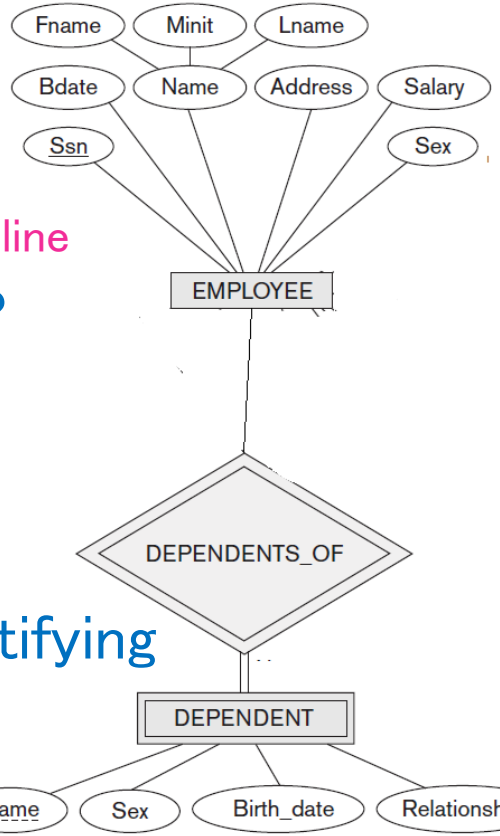


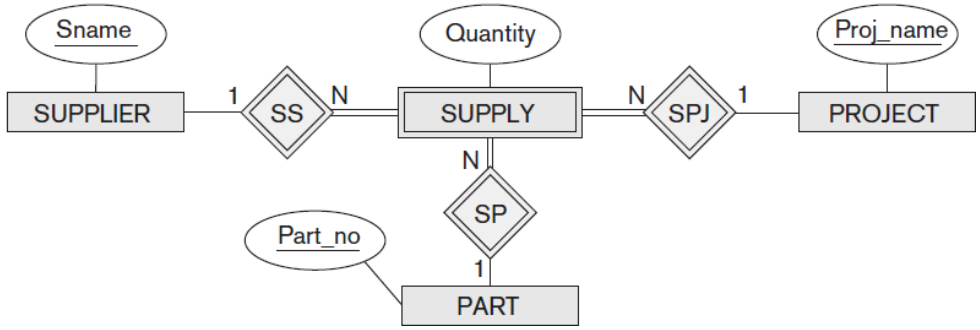
Weak Entity Type

- Not every existence dependency results in a weak entity type.
  - Example: DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity
- Partial key or Discriminator:
  - The attribute that can uniquely identify weak entities that are related to the same owner entity.
  - In the worst case, a composite attribute of all the weak entity's attributes will be the partial key.

Weak Entity Type in ER Model

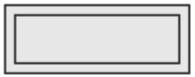
- Weak entity type and its identifying relationship are represented by surrounding their boxes and diamonds with double lines
- Partial key attribute is underlined with a dashed or dotted line
- When use weak entity or complex attribute?
  - Data base designer choice
  - If the weak entity type participates independently in relationship types other than its identifying relationship type.
- Ca … identifying … er





ER Notations

Entity



Weak Entity

Relationship

Indentifying Relationship

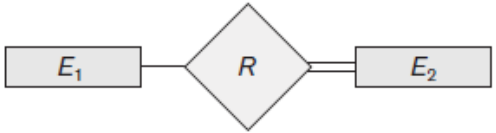
Attribute

Key Attribute

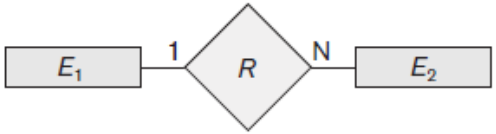
Multivalued Attribute

Composite Attribute

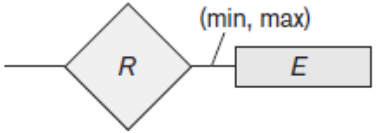
Derived Attribute



Total Participation of E_2 in R



Cardinality Ratio 1: N for $E_1 : E_2$ in R



Structural Constraint (min, max) on Participation of E in R

References

- Chapter 3 of FUNDAMENTALS OF Database Systems, SEVENTH EDITION
- Chapter 12 of Database Systems A Practical Approach to Design, Implementation, and Management, SIXth edition

- https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/
- https://afteracademy.com/blog/what-is-an-entity-entity-type-and-entity-set/