E-R Modeling

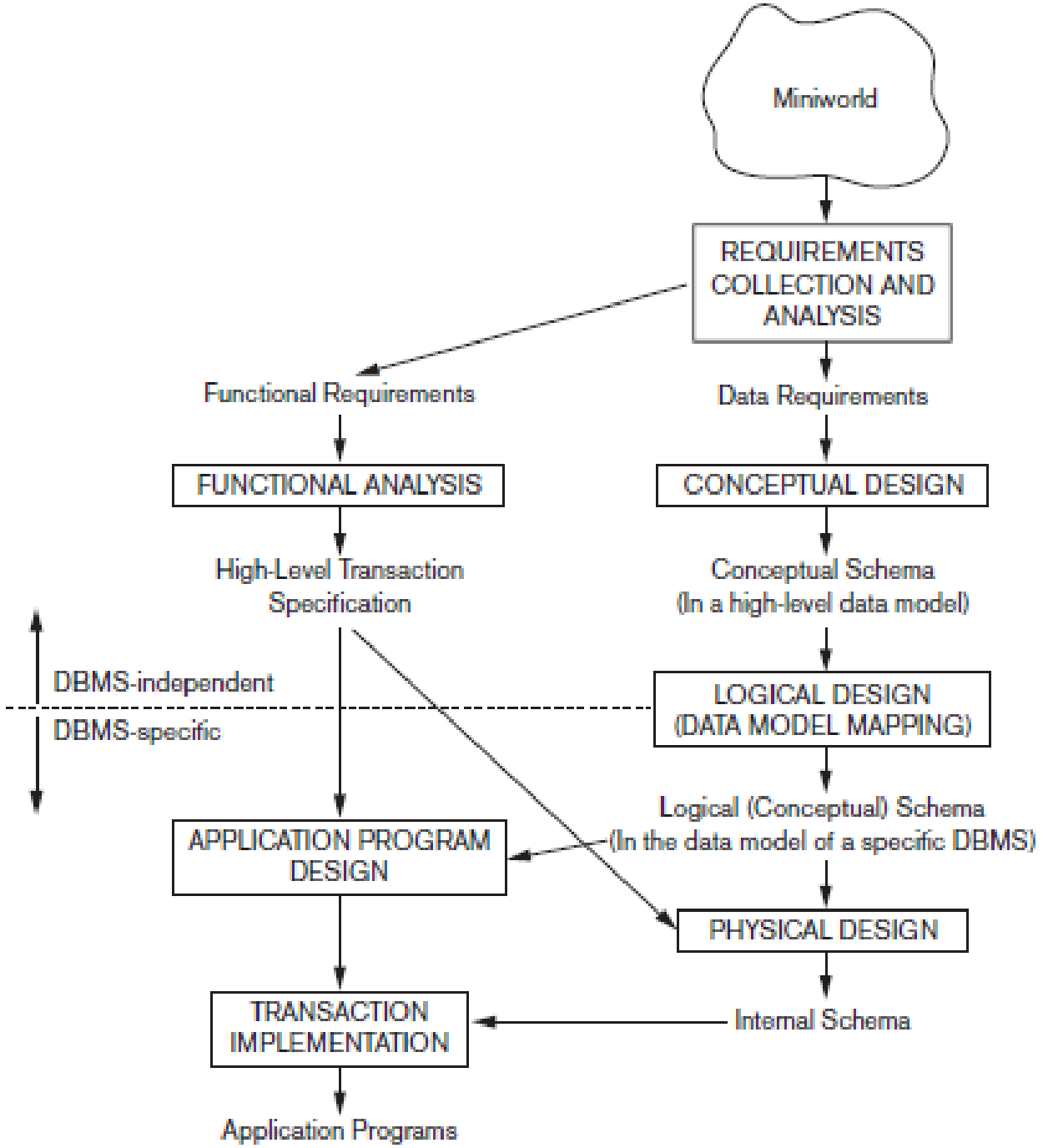
Goonjan Jain
Department of Applied Mathematics
Delhi Technological University

Database Design

- Requirement Analysis
- Conceptual Design
- Logical Design
- Schema Refinement
- Physical Design
- Security Design

Database Design

• Requirement Analysis	user’s needs
• Conceptual Design	high level (ER)
• Logical Design	Tables
• Schema Refinement	Normalization
• Physical Design	Indices etc.
• Security Design	Access control


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

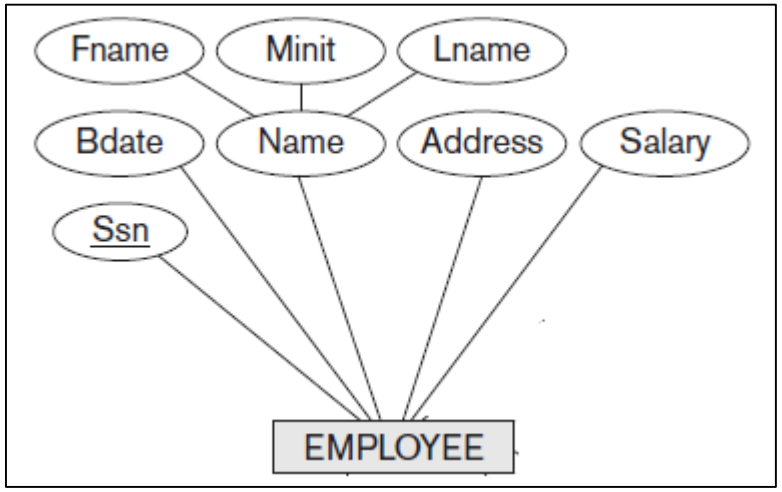
Overview

- Concepts
  - Entities
  - Relationships
  - Attributes
  - Specialization/Generalization
  - Aggregation
  - ER modeling questions

Entity

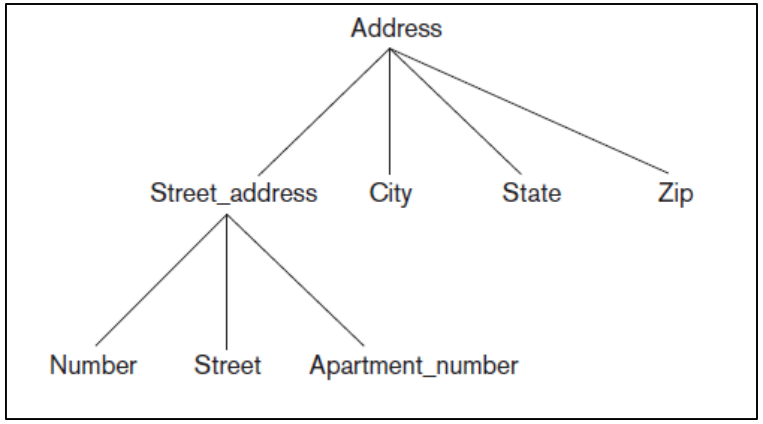
- an object in the real world that is distinguishable from other objects.
- May be physical like car, house, etc or conceptual like job, university etc.
- Examples:
  - Green toy,
  - toy department,
  - manager of the toy department,
  - home address of the manager of the toy department.
- Collection of similar entities is called an **entity set.**
- Entity sets need not be disjoint.
- Collection of toy department employees and the collection of appliance department employees may both contain employee John Doe (who happens to work in both departments).
- An entity set, Employees, may contain both the toy and appliance department employee sets.
- Nouns -> entity sets

Attributes

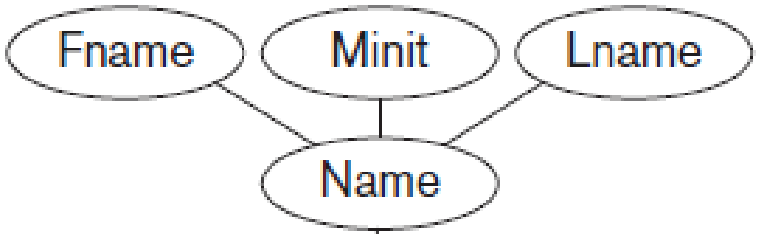


- An entity is described using a set of attributes.
- All entities in a given entity set have same attributes
- E.g. Employees entity set has name, social security number (ssn), and salary as attributes
- We must identify a domain of possible values for each attribute
- For example, the domain associated with the attribute *name* of Employees might be the set of 20-character strings.

Composite vs Simple Attributes

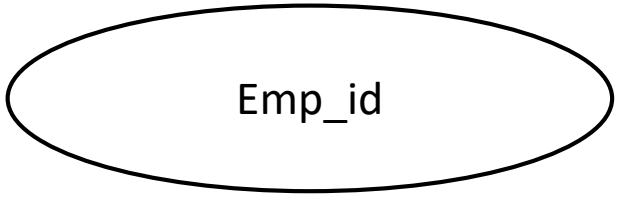


- **Composite attributes** can be divided into smaller subparts
- Each subpart represents more basic attributes with independent meanings.
- E.g. address attribute can be subdivided into Street_address, City, State, and Zip
- If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.
- Attributes that are not divisible are called **simple** or **atomic attributes**.

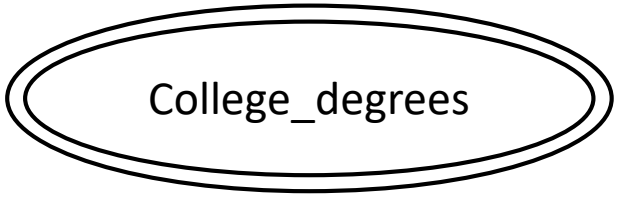


Single-Valued vs Multivalued Attributes

- **Single-valued attributes:** Attributes that have a single value for a particular entity.
- For example, Age is a single-valued attribute of a person.
- **Multivalued Attributes**: can have a set of values for the same entity
- for instance, a Colors attribute for a car, or a College_degrees attribute for a person.
- A multivalued attribute may have lower and upper bounds to constrain the *number of values* allowed for each entity.



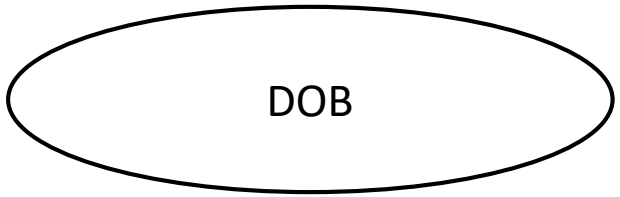
single-values attribute



multivalued attribute

Stored vs Derived Attributes

- If two (or more) attribute values are related:
- for example, the Age and Birth_date attributes of a person.
- Value of Age can be determined from the current (today's) date and the value of Birth_date.
- 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**.
- Some attribute values can be derived from *related entities*;
- for example, an attribute Number_of_employees of a DEPARTMENT entity can be derived by counting the number of employees related to (working for) that department.

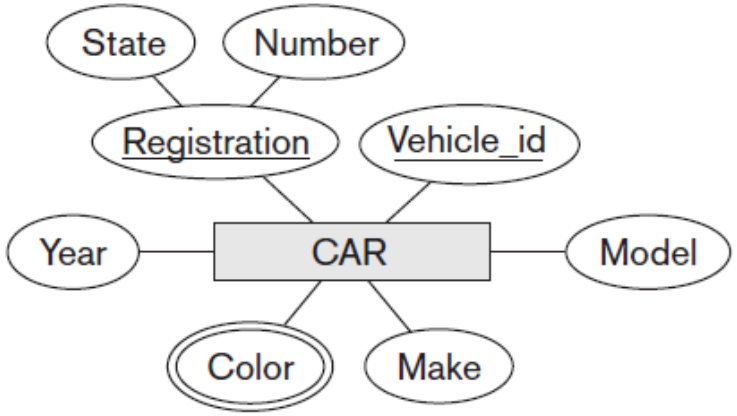


stored attribute

derived attribute

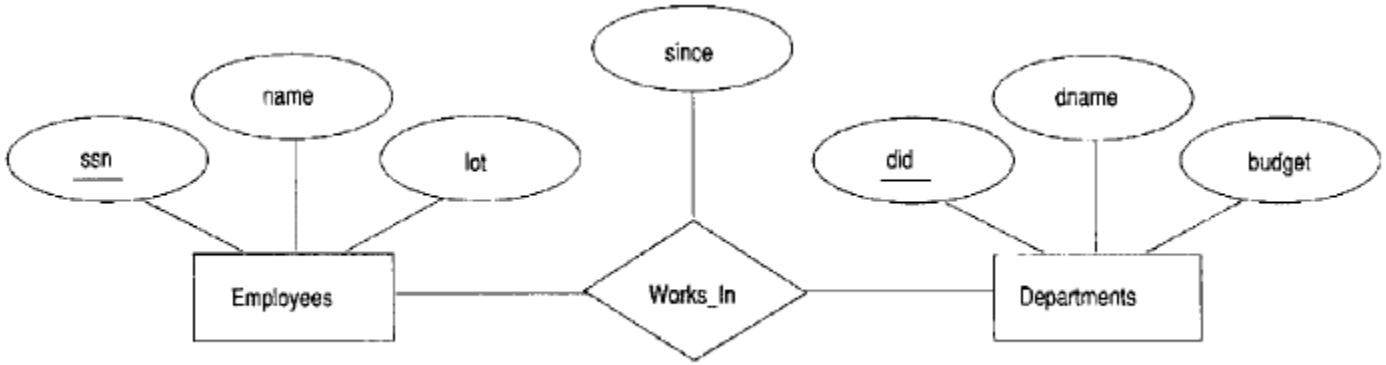
Keys

- the values of attributes values of an entity must be such that they can *uniquely identify* the entity.
- no two entities in an entity set are allowed to have exactly the same value for all attributes.
- A key is a minimal set of attributes whose values uniquely identify an entity in the set.
- each key attribute has its name underlined inside the oval



Relationships

- an association among two or more entities.
- For example, we may have the relationship that Zulu works in the pharmacy department.
- a set of similar relationships into a **relationship** set
- Verbs -> relationship sets

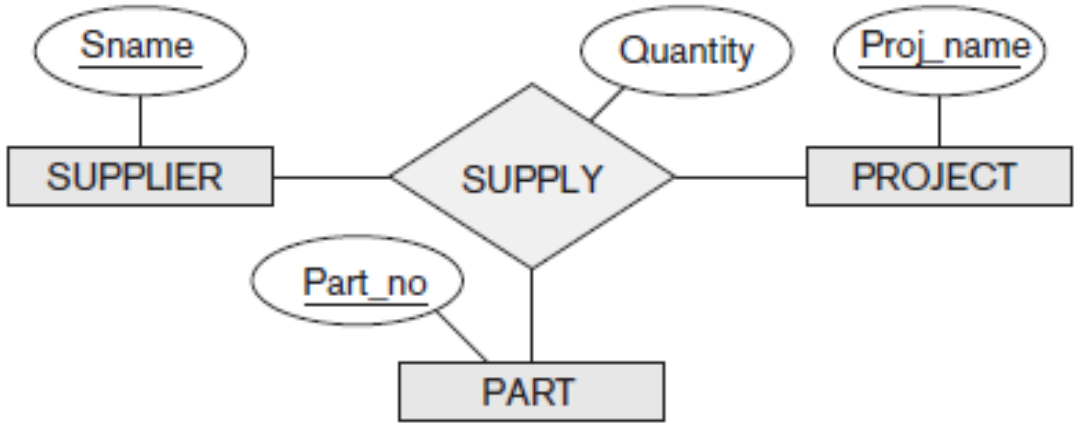


Descriptive Attributes

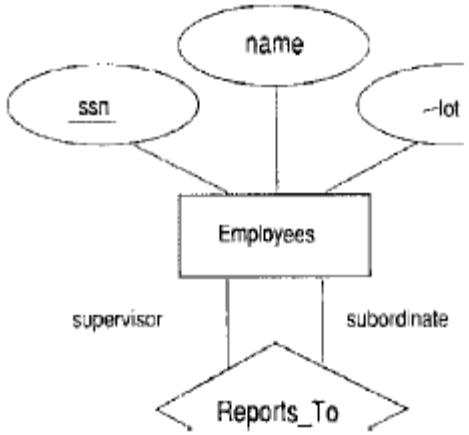
- used to record information about the relationship, rather than about any one of the participating entities
- for example, we may wish to record that Zulu works in the pharmacy department since January 1991.
- A relationship must be uniquely identified by the participating entities, without reference to the descriptive attributes.
- For a given employee-department pair, we cannot have more than one associated *since* value.

Ternary Relationship

- Suppose each supplier supplies different parts to projects.
- We want to record the projects in which each part is supplied.
- This relationship is **ternary** because we must record an association between a supplier, parts and project



Recursive Relationship

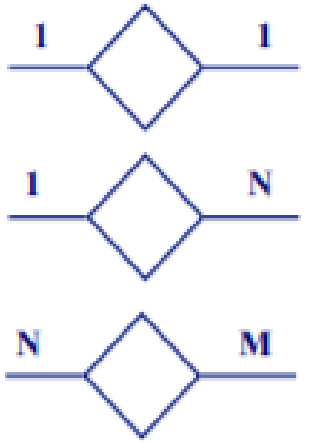


- Employees report to other employees,
- every relationship in Reports_To is of the form (emp1 , *emp2)* , where both *emp1* and *emp2* are entities in Employees.
- However, they play different roles: *emp1* reports to the managing employee *emp2*,
- reflected in the role indicators *supervisor* and *subordinate*
- If an entity set plays more than one role, the role indicator concatenated with an attribute name from the entity set gives us a unique name for each attribute in the relationship set.

Mapping Cardinalities

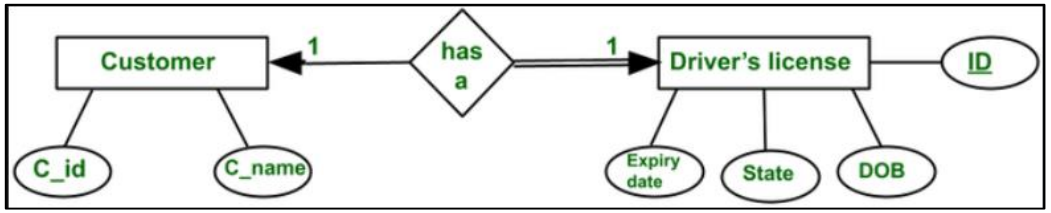
- Expresses the no. of entities to which another entity can be associated via a relationship set
- Useful in describing binary relationship sets

- 1 to 1 (example?)
- 1 to N
- N to M

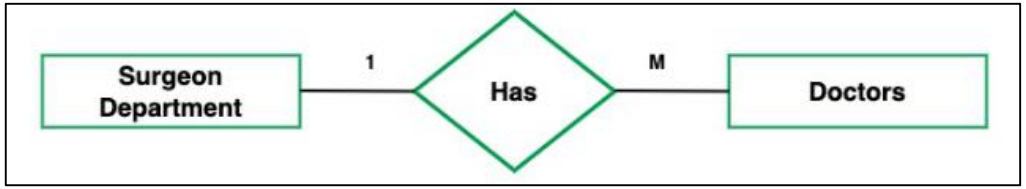


Mapping Cardinalities

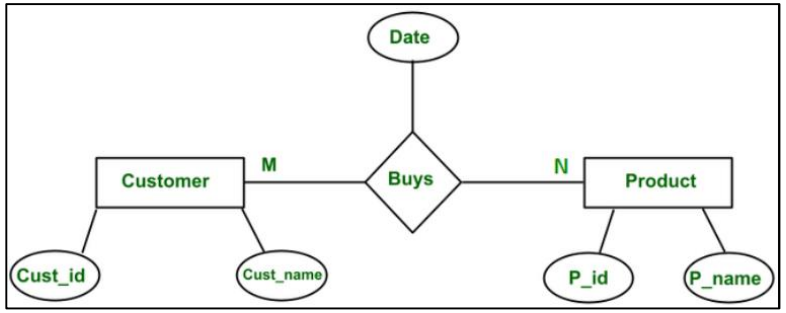
- 1 – to – 1



- 1 – to – many (many – to - 1)



- Many – to – many

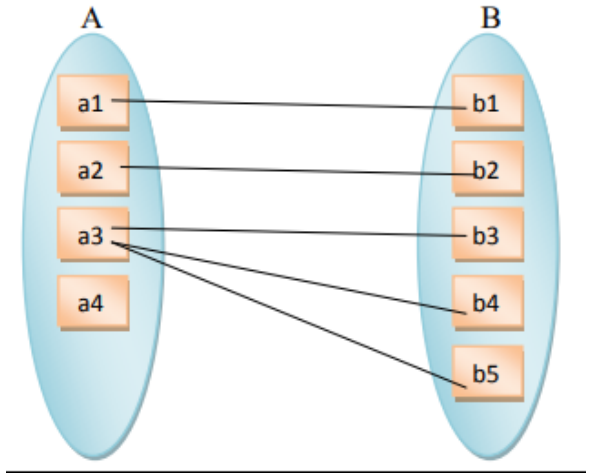


Participation Constraints

- The participation constraint specifies the number of instances of an entity can participate in a relationship set.
  - Total Participation
  - Partial Participation

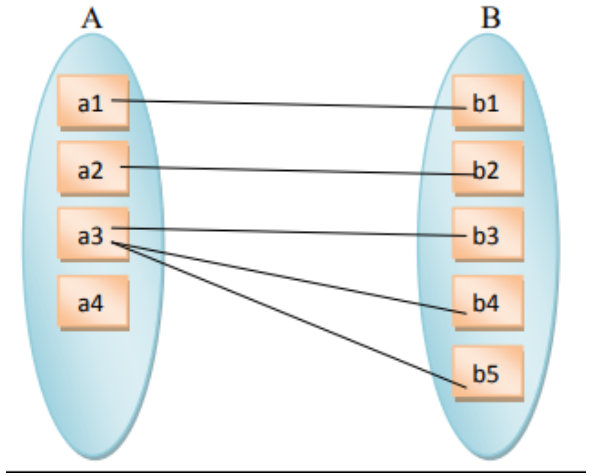
Total Participation

- The Participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.
- Participation of B is Total



Partial Participation

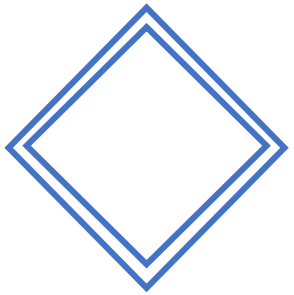
- The participation of an entity set E in relationship set R is said to be partial if only some entities in E participate in relationships in R.
- Participation of A is partial



Weak Entity set

- An entity set that does not have sufficient attributes to form a primary key
- An entity set that has a primary key - **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

Weak entity set



Identifying relationship

Partial Key

- A weak entity type normally has a **partial key**, the attribute that can uniquely identify weak entities related to the same owner entity.
- The partial key attribute is underlined with a dashed or dotted line.
- In the worst case, a composite attribute of *all the weak entity's attributes* will be the partial key.

- The primary key of a weak entity set is formed by the primary key of the identifying entity set plus the weak entity set's discriminator (or partial key) .

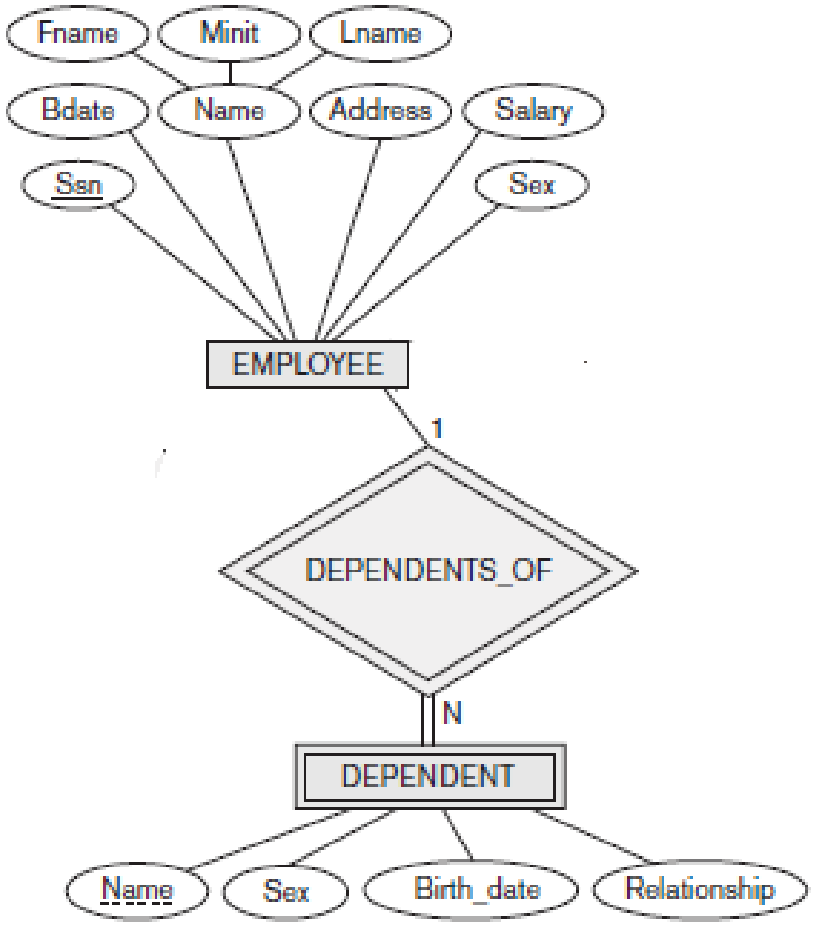
Relationships

- The owner entity set and the weak entity set must participate in a one-to-many relationship set (one owner entity is associated with one or more weak entities, but each weak entity has a single owner).
- The weak entity set must have total participation in the identifying relationship set.
- The identifying relationship set should not have any descriptive attributes since any such attributes can instead be associated with the weak entity set.
- Possible to have a weak entity set with more than one identifying entity set. The primary key of the weak entity set would consist of the union of the primary keys of the identifying entity sets plus the discriminator of the weak entity set.
- A weak entity set can participate in relationships other than the identifying relationship.
- A weak entity set may participate as an owner in an identifying relationship with another weak entity set.

Example

- Consider the entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee via a 1:N relationship.
- Attributes of DEPENDENT are Name (the first name of the dependent), Birth_date, Sex, and Relationship (to the employee).
- Two dependents of *two distinct employees* may have the same values for Name, Birth_date, Sex, and Relationship, but they are still distinct entities.
- They are identified as distinct entities only after determining the *particular employee entity* to which each dependent is related.
- Each employee entity is said to *own* the dependent entities that are related to it.
- In our example, if we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the partial key.

Example


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
EMPLOYEE
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Weak Entity Set – Alternate representation

- Weak entity types can sometimes be represented as complex (composite, multivalued) attributes.
- we could specify a multivalued attribute Dependents for EMPLOYEE, a multivalued composite attribute with the component attributes Name, Birth_date, Sex, and Relationship.
- A weak entity set may be more appropriately modeled as an attribute if it participates in only the identifying relationship and if it has few attributes.
- Conversely, a weak entity set representation more aptly models a situation where the set participates in relationships other than the identifying relationship and where the weak entity set has several attributes.

Example

- A country bus company owns a number of buses.
- Each bus is allocated to a particular route, although some routes may have several buses.
- Each route passes through a number of towns.
- One or more drivers are allocated to each stage of a route, which corresponds to a journey through some or all of the towns on a route.
- Some of the towns have a garage where buses are kept and each of the buses are identified by the registration number and can carry different number of passengers, since the vehicles vary in size and can be single or double-decked.
- Each route is identified by a route number and information is available on the average number of passengers carried per day for each route.
- Drivers have an employee number, name, address, and sometimes a telephone number.

Example - Entities

- Bus – Company owns buses and will hold information about them.
- Route – Buses travel on routes and will need described.
- Town – Buses pass through towns and need to know about them.
- Driver – Company employs drivers, personnel will hold their data.
- Stage – Routes are made up of Stages. Garage –
- Garage houses buses, and need to know where they are.

Example - Relationships

- A bus is allocated to a route and a route may have several buses.
- Bus-route (m:1) is serviced by a route comprises of one or more stages.
- Route-Stage (1:m) Comprises One or more drivers are allocated to each stage.
- Driver-stage (m:1) is allocated A Stage passes through some or all of the towns on a route.

Example - Relationships

- Stage-town (min) passes-through. A route passes through some or all of the towns.
- Route-town (min) passes-through Some of the towns have a garage.
- Garage-town (1:1) is situated Agarage keeps buses and each bus has one home garage.
- Garage-bus (m:1) is garaged.

Example - Attributes

- Bus (reg-no, make, size, deck, no-pass)
  - Route (route-no avg-pass)
- Driver (emp-no, name, address, tel-no)
  - Town name)
  - Stage (stage-no)
  - Garage (name, address)

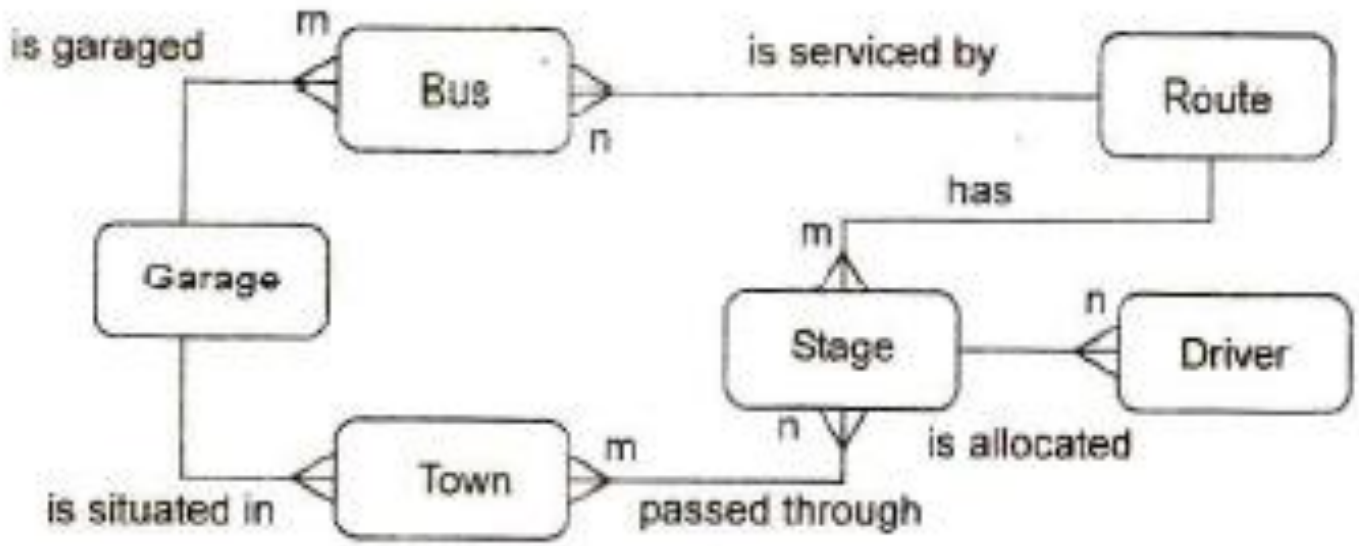
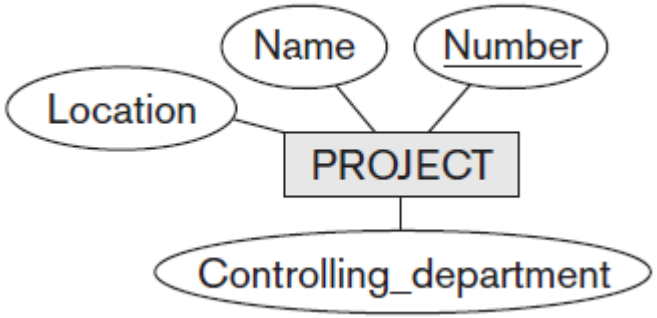


Fig. Bus Company

Reduction of ER Model to Relational Model

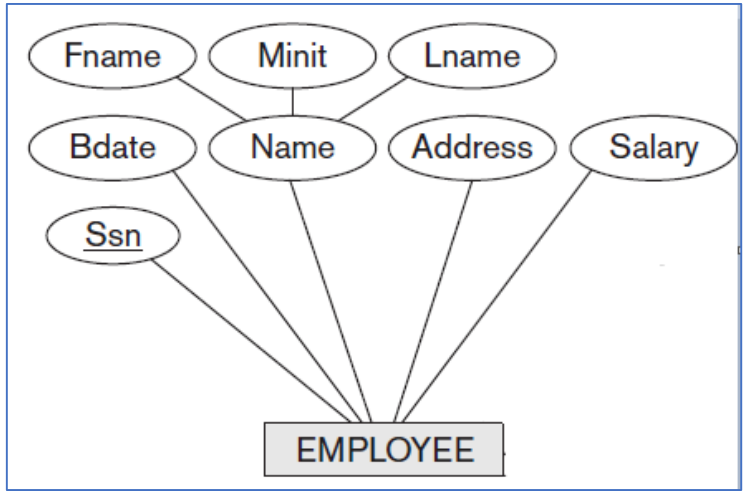
- Strong Entity Sets with simple attributes
  - Strong Entity Set with *n* attributes -> Table with *n* distinct columns
  - Key of entity set -> Primary Key of table



Project (<u>Number</u>, Name, Location, Controlling_department)

Representation of Strong Entity Sets with Complex Attributes

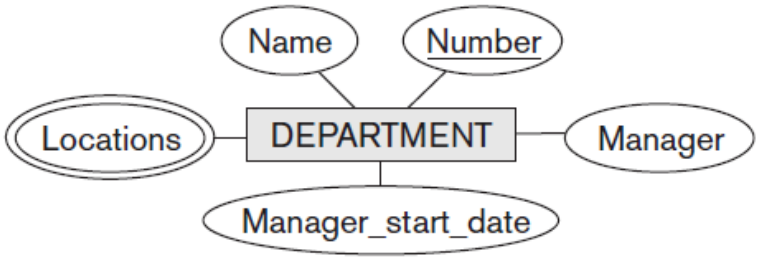
- Composite attributes
  - create separate columns for each component of a composite attribute.
  - do not create a separate attribute for the composite attribute itself



Employee(<u>Ssn</u>, Bdate, Fname, Minit, Lname, Address, Salary)

Representation of Strong Entity Sets with Complex Attributes

- Derived attribute
  - Not stored explicitly in the relational model
  - E.g. age (a derived attribute in ER model) is not stored in relational schema
- Multivalued attribute
  - New relational schema (table) is created for the attribute



Department (Number, Name, Manager, Manager_start_date)

Foreign Key

Department_Loc (Number, Location)

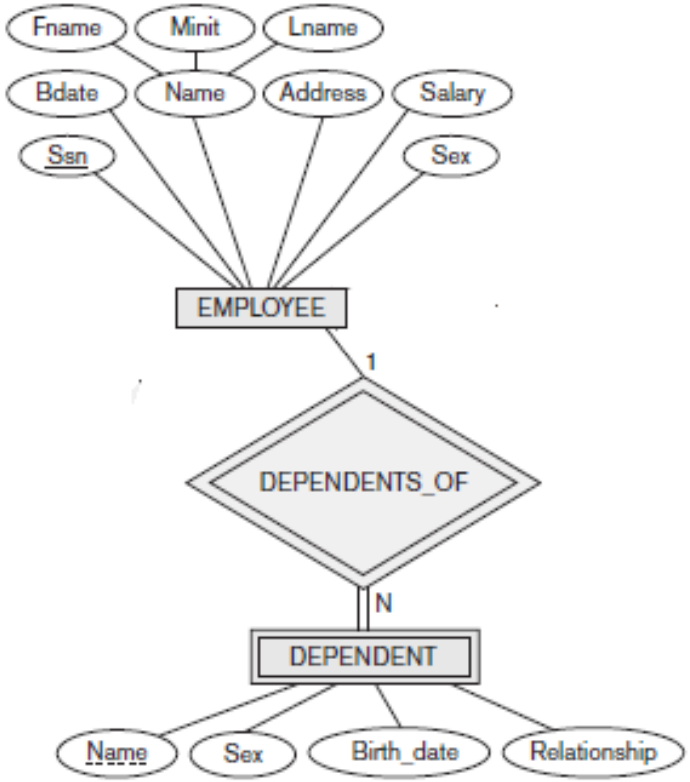
Representation of Weak Entity Sets

- Let A be a weak entity set with attributes $a_1, a_2, \ldots, a_m$.
- Let B be the strong entity set on which A depends.
- Let the primary key of B consist of attributes $b_1, b_2, \ldots, b_n$.
- We represent the entity set A by a relation schema called A with one attribute for each member of the set:

$$\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$$

- For schemas derived from a weak entity set, the combination of the primary key of the strong entity set and the discriminator of the weak entity set serves as the primary key of the schema.

Representation of Weak Entity Sets



Employee (Ssn, Bdate, Fname, Minit, Lname, Address, Salary, Sex)

Foreign Key

Dependent (Ssn, Name, Sex, Birth_Date, Relationship)

Representation of Relationship Sets

- Let R be a relationship set, let $a_1, a_2, \ldots, a_m$ be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R, and let the descriptive attributes (if any) of R be $b_1, b_2, \ldots, b_n$.
- We represent this relationship set by a relation schema called R with one attribute for each member of the set:

$$\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$$

- *No schema is created for identifying relationship sets.*

Primary Key of Relational Schema

- For a binary many-to-many relationship, the union of the primary-key attributes from the participating entity sets becomes the primary key.
- For a binary one-to-one relationship set, the primary key of either entity set can be chosen as the primary key. The choice can be made arbitrarily.
- For a binary many-to-one or one-to-many relationship set, the primary key of the entity set on the “many” side of the relationship set serves as the primary key.

Combination of schemas

- Consider a many-to-one relationship set *AB* from entity set *A* to entity set *B*.
- We get three schemas: *A*, *B*, and *AB*.
- participation of *A* in the relationship is total
- Then we can combine the schemas *A* and *AB* to form a single schema consisting of the union of attributes of both schemas.
- The primary key of the combined schema is the primary key of the entity set into whose schema the relationship set schema was merged.
- In the case of one-to-one relationships, the relation schema for the relationship set can be combined with the schemas for either of the entity sets.
- We can combine schemas even if the participation is partial by using null values.