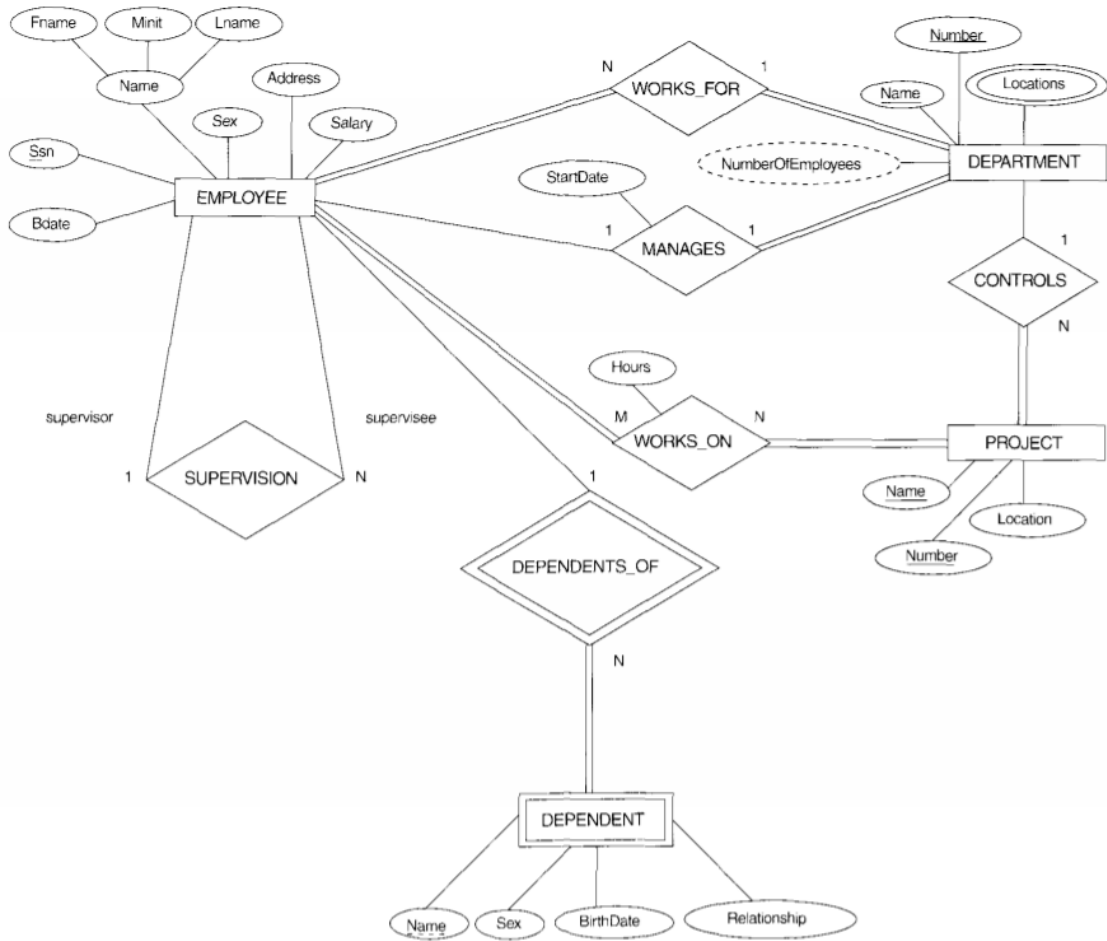


**Database Management System**  
**Lecture: 6**  
**Data Modeling Using the Entity-Relationship (ER) Model**

**Example:** Consider the following scenario to design a database for a company:

- 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.
- We store each employee's name, social security number, address, salary, sex, 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. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

**We will design ER model for the above said database.**



**Fig: An ER Schema diagram for above said COMPANY database.**

## **The Entity-Relationship Model:**

ER Model is a conceptual data model that views the real world as entities and relationships. For the database designer, the utility of the ER model is:

- It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
- The model can be used as a design plan by the database developer to implement a data model in a specific database management software.

## **Basic Constructs of E-R Modeling:**

The ER model view the real world as a construct of entities and association between entities.

### **Entities:**

Entities are the principal data object about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database. Some specific examples of entities are EMPLOYEES, PROJECTS, INVOICES. An entity is analogous to a table in the relational model.

Entities are classified as **independent or dependent** (in some methodologies, the terms used are **strong and weak**, respectively).

An independent entity is one that does not rely on another for identification. A dependent entity is one that relies on another for identification.

An entity occurrence (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table.

**Entity Types and Entity Sets:** An entity type defines a collection (or set) of entities that have same attributes. Each entity type in the database is described by its name and attributes.

A database usually contains group of entries that are similar. E.g. a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own value(s) for each attribute.

**Key Attributes of an Entity Type:** An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes. An entity type usually has an attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.

E.g. for the PERSON entity type, a typical key attribute is SocialSecurityNumber. Sometimes, several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity.

**Weak and Strong Entity Types:** Entity types that have a key attribute are called strong entity types.

Entity types that do not have key attributes of their own are called weak entity types. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values. This other entity type is called the identifying or owner entity type and such type of relationship that relates a weak entity type to its owner is called the identifying relationship of the weak entity type.

### **Attributes:**

Attributes describe the entity of which they are associated. A particular instance of an attribute is a value. E.g. "Rohit Sharma" is one value of the attribute Name.

The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.

Attributes can be classified as identifiers or descriptors.

- Identifiers (also called keys) uniquely identify an instance of an entity.
- A descriptor describes a non unique characteristic of an entity instance.

**Composite versus Simple (Atomic Attributes):** Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.

e.g. the Address attribute of the employee entity can be subdivided into street address, City, state and Zip.

Attributes that are not divisible are called simple or atomic attributes.

**Single-Valued versus Multi-valued Attributes:** Most attributes have a single value for a particular entity such attributes are called single-valued. E.g. Age is a single-valued attribute of a person.

In some cases an attribute can have a set of values for the same entity e.g. a color attribute for a car. Cars with one color have a single value whereas two-tone cars have two values for Colors. Such attributes are called multivalued.

**Stored versus Derived Attributes:** In some cases, two (or more) attribute values are related –e.g. the Age and Birth Date attributes of a person. For a particular person, the value of Age can be determined from the current (today's) date and the value of that person's BirthDate. The Age attribute is hence called a derived attribute and is said to be derived from the BirthDate attribute which is called a stored attribute.

**Complex Attributes:** Composite and multivalued attributes can be nested in an arbitrary way.

We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas and by displaying multivalued attributes between braces {}. Such attributes are called complex attributes. E.g. if a person can have more than one residence and each residence can have

multiple phones, an attribute AddressPhone for a person can be specified as:

```
{AddressPhone( {Phone(AreaCode,PhoneNumber)},  
Address(StreetAddress(Number,Street,ApartmentNumber),  
City,State,Zip) ) }
```

### **Relationships:**

A relationship represents an association between two or more entities. An example of a relationship would be:

- Employees are assigned to projects.
- Projects have subtasks.
- Departments manage one or more projects

Relationships are classified in terms of degree, connectivity, cardinality and existence.

### **Degree of a Relationship:**

The degree of a relationship is the number of entities associated with the relationship.

The n-ary relationship is the general form for degree n e.g. binary and ternary where degree is 2 and 3 respectively.

Binary relationships, the association between two entities is the most common type in the real world.

A recursive binary relationship occurs when an entity is related to itself. E.g. "some employees are married to other employees".

A ternary relationship involves three entities and is used when a binary relationship is inadequate.

Many modeling approaches recognize only binary relationships. Ternary or n-ary relationships are decomposed into two or more binary relationships.

## Connectivity and Cardinality:

The connectivity of a relationship describes the mapping of associated entity instances in the relationship.

The values of connectivity are “one” or “many”. The cardinality of a relationship is the actual number of related occurrences for each of the two entities.

The basic types of connectivity for relations are:

- **One-to-One (1:1):** A one-to-one (1:1) relationship is when at most one instance of a entity A is associated with one instance of entity B. e.g. employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee.
- **One-to-Many (1:N) :** A 1:N relationship is when for one instance of entity A, there are zero, one or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example is: A department has many employees. Each employee is assigned to one department.
- **Many-to-Many (M:N):** A many-to-many (M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. An example is: employees can be assigned to no more than two projects at the same time; projects must have assigned at least three employees A single employee can be assigned to many projects; conversely, a single project can have assigned to it many employee. Here the cardinality for the relationship between employees and projects is two and the cardinality between project and employee is three. Many-to-many relationships cannot be directly translated to relational tables but instead must be transformed into two or more one-to-many relationships using associative entities.

**Direction:** The direction of a relationship indicates the originating entity of a binary relationship. The entity from which a relationship originates is the

parent entity; the entity where the relationship terminates is the child entity.

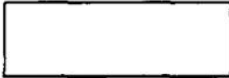
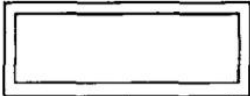
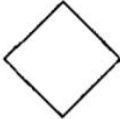
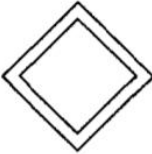


The direction of a relationship is determined by its connectivity.

In a one-to-one relationship, the direction is from the independent entity to a dependent entity. If both entities are independent, the direction is arbitrary.

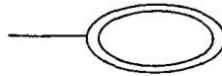
With one-to-many relationships, the entity occurring once is the parent.

The direction of many-to-many relationships is arbitrary.

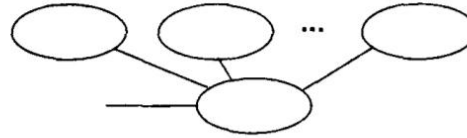
### **Notations for ER Diagram:**

<u>Symbol</u>	<u>Meaning</u>
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE

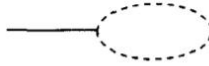




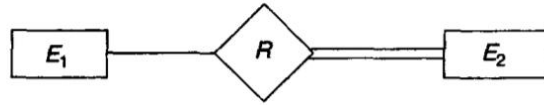
MULTIVALUED ATTRIBUTE



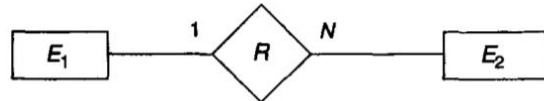
COMPOSITE ATTRIBUTE



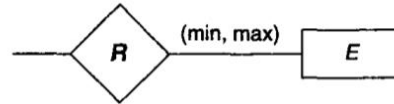
DERIVED ATTRIBUTE



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  $R$