# Entity-Relationship model

Entity-relationship model is a conceptual data model that describes data using concepts of entity and relationship. ER model facilitates creation of a conceptual design of a database and this design is called ER schema. ER schema is a conceptual design of a database and the design is called conceptual because this design cannot directly be mapped to computer storage media. A pictorial representation of ER schema is called ER diagram. ER schema is meant for human comprehension and is understandable to end users.

An entity is an object or a thing that has independent existence. The word independent existence is important. The social security number, as for example, cannot be an entity because it has no existence without being associated with an employee, similarly, a registration number of a student cannot be an entity because it does not have independent existence it cannot exist without belonging to a student. An entity has value for its attributes. <'Product A', 4, 'Bellaire'> is a PROJECT entity, <'Production', 4, '943775543', '2007-10-01'> is a DEPARTMENT entity. A major part of data stored in a database is values of attributes.

A relationship is an association of an entity with some other entity. A particular association is called a relationship instance. The structure of an entity is called entity type. The characteristics of an entity are called its attributes. An entity type has a name and a list of attributes.

An attribute that cannot be decomposed into simpler attributes is called a simple attribute. The registration number of a student, social security number of an employee and the likes are examples of simple attributes. An attribute that can be decomposed into simpler attributes is called a composite attribute. The various parts of a composite attribute are called component attributes. A composite attribute Name with component attributes Fname, Minit, Lname is represented as Name(Fname, Minit, Lname) in ER schema. The DB designer after interaction with the prospective end users of a DB decides whether an attribute should be considered as a composite attribute. The Address attribute of EMPLOYEE entity, for example, could be treated as a composite attribute with various components such as house number, name of street, name of city and the likes, but after interaction with the prospective end users the DB designer found that end users are not interested in different parts of address and so decided to treat it as a simple attribute. In some circumstances decomposition of an attribute into various parts may result in loss of semantic of the attribute and so the attribute should be treated as a simple attribute.

An attribute is called single valued attribute if it may assume at the most one value for an entity. e.g. Ssn, Salary, Regno are single valued attributes. An attribute is called multi valued attribute if it can assume any number of values for an entity. e.g. color of a car written as {Color} in ER schema, phone number {Phone_number} may be multi valued attributes.

An attribute which is simple and single-valued is called an atomic attribute.

An attribute which is composite and multi-valued is called a complex attribute. e.g. {Address(HNo, Street, City, Country)} is a composite and multi-valued attribute (indicated in ER schema as shown here).

When the value of an attribute (say A) may be computed from the value of some other attribute (say B) then the later is called a stored attribute and the former is called a derived attribute.

The two attributes (A & B) may belong to two different entity type (e.g. DEPARTMENT & EMPLOYEE)

Bdate is a stored attribute & Age is a derived attribute, Basic_ salary is a stored attribute and Gross_pay is a

derived attribute. A database does not store the value of a derived attribute.

In ER schema an entity type is described by a name and a list of attributes and is denoted by ENTITYNAME(Attribute_1, Attribute_2, Attribute_3, …, Attribute_n). To give you a concrete example the employee entity type with Ssn, name as a composite attribute with its components as Fname, Minit, and Lname, and other attributes such as Address, Salary, etc. can be denoted by EMPLOYEE(Ssn, Name(Fname, Minit, Lname), Address, Salary, …). As a matter of convention, the name of entity type is written in uppercase and those of attributes in title case.

The set all possible values that an attribute may assume is called value set of the attribute. The value set of Ssn is a set of all possible strings of exactly nine numeric characters.

V(Ssn) = { string of exactly nine numeric characters }

V(Fname) = { string of alphabetic characters of varying length *without space*}

The value sets are not displayed in ER diagram. A value set defines all possible values of an attribute.

Usually, a subset of this set is found in a database state. V(Dnumber) = {single digit number} but you found only the three values 1, 5 and 4 in the company database state. The value set of a multi valued attribute is the power set of the value set of the corresponding single valued attribute and the value set of a composite attribute is the Cartesian product of the value set of the component attributes. See the examples below.

V(Color) = {red, green, blue}

V({Color}) = { {red}, {green}, {blue}, {red, green}, {red, blue}, {blue, green}, {red, green, blue} } = Power set of V(Color)

Name(Fname, Minit, Lname) – a composite attribute

V(Name(Fname, Minit, Lname)) = V(Fname) × V(Minit) × V(Lname)

V(Composite attribute) = Cartesian product of value set of component attributes

A **NULL value** is a special value

- value NOT APPLICABLE
- value UNKNOWN
    - exists but could not be found [**UNAVAILABLE**]
    - not sure whether value exists [**MISSING**]

e.g. Height (exits but not available) [UNAVAILABLE], Phone_number (not sure whether he has) [MISSING].

A collection of entities of the same type is called an entity set. An entity type defines the structure of entity. A collection of entities of the same type is called entity set. An entity type is called an intension and an entity set is called extension of entity type. An entity type and entity set are called by the same name. A particular association of an entity with some other entity is called relationship instance. The structure of relationship instance is called relationship type. A set of relationship instances of the same type is called relationship set. A relationship set and relationship type are called by the same name. Every relationship type involves one or more entity types. The number of distinct entity types involved in a relationship type is called degree of relationship type. Relationship type involving two entity types is called binary relationship type (relationship is called binary relationship). Relationship type involving three distinct entity types is called a ternary relationship type, an N-ary relationship type involves n distinct entity types. A unary relationship type defines an association of an entity type with itself (and this type of relationship type is also known as a *Recursive relationship type*). An entity type involved in a relationship type is called a participating entity type. Each relationship instance involves exactly one entity from participating entity type(s).

A set of attributes that can uniquely identify the entities of an entity type is called a key of the entity type. A key with multiple attributes is called a composite key and the attributes that are part of a key are called key attributes and the other attributes are called non-key attributes of the entity type. A key is a feature of an entity type and so it holds on any entity set of the entity type. Defining key is defining a constraint on the entity type and this constraint is known as uniqueness constraint. The value of a key is distinct for each entity of any entity set of an entity type. The uniqueness constraint is derived from the universe of discourse. An entity type may have multiple keys. But there is no concept of primary key in ER model.

Cardinality ratio of a binary relationship specifies the maximum number of entities that may be associated with some other entity. The cardinality ratio may be N:1 (many to one), 1:N (one to many), M:N (many to many) and 1:1 (one to one). If there are two entity types $E_1$ and $E_2$ associated with each other by the relationship type R then if any number of entities of $E_1$ may be associated with an entity of $E_2$ but at the most one entity of $E_2$ may be associated with an entity of $E_1$ then the relationship type between $E_1$ and $E_2$ has a cardinality ratio of N:1 (many to one), if at the most one entity of $E_1$ may be associated with an entity of $E_2$ but any number of entities of $E_2$ may be associated with an entity of $E_1$ then the relationship type between $E_1$ and $E_2$ has a cardinality ratio of 1:N (one to many), if any number of entities of $E_1$ may be associated with any number of entities of $E_2$ then the relationship type between $E_1$ and $E_2$ is called M:N (many to many), if at the most one entity of $E_1$ may be associated with at the most one entity of $E_2$ then the relationship type between $E_1$ and $E_2$ is called 1:1 (one to one).

A relationship type may have its own attributes. If the cardinality ratio is 1:1 or N:1 or 1:N, then these attributes is merged with one of the participating entity types, for 1:N or N:1, the attribute is preferably merged with the entity type lying on the many side, but if the cardinality ratio is M:N then the attribute of relationship type cannot be merged with participating entity type.

The nature of association of a participating entity type in a relationship type defines the **role** of the entity type in the relationship type. If the participating entities are distinct then role name (attached to participating edge) is not necessary, otherwise it is necessary to mention the role name using role name as a label of the participating edge.

Participation constraint specifies the minimum number of entities that may participate in a relationship instance with some other entity. If every entity in any entity set of a participating entity type is involved in a relationship instance then the participation of the entity type in the relationship type is said to be total. If every entity of a participating entity type is not involved in a relationship instance then the participation of the participating entity type in the relationship type is called partial. If minimum number of entities that may participate in any relationship instance is zero then the participation is partial and if this minimum is one then participation is total. Total participation is also called existence dependency.

The participation and cardinality constraints together are called structural constraints.

An entity type without a key is called a weak entity type. An entity type with a key is called a strong entity type. The relationship type between a strong entity type and a weak entity type is called an identifying relationship type. The strong entity type in this case is called identifying (owner) entity type. A set of attributes of a weak entity type that can uniquely identify weak entities that are related to the same owner entity is called a partial key (discriminator) of the weak entity type. Every weak entity type has total participation in the relationship type. Every weak entity type is existence dependent on its owner entity type (weak entity type → existence dependency). But existence dependency does not necessarily imply that the entity type is weak. An owner entity type may itself be a weak entity type. Any number of levels of weak entity types may exist. A weak entity type may have more than one identifying entity type. A weak entity type may be found to be involved in a relationship type of degree greater than two. A weak entity type may be represented as a complex attribute of the identifying entity type. The choice is made by the database designer. A weak entity type may be represented as a multi valued composite attribute of its owner. If there are many attributes in the weak entity type, then retain the weak entity type, do not represent weak entity type as an attribute. If weak entity type participates in some other relationship independently then retain the weak entity type, do not represent weak entity type as an attribute.

The participation constraint is also known as minimum cardinality constraint. The participation constraint specifies the minimum number of entities that may be associated with some other entity, whereas, cardinality constraint specifies the maximum number entities that may be associated with some other entity and this is why the former is called a min constraint and the later is called a max constraint and these two constraints together my be used as (min, max) constraint on an ER diagram replacing the notation of cardinality constraint (1:1, 1:N, N:1 and M:N) and participation constraint (double line for total and single line for partial). So an ER diagram may be displayed with (min, max) notation instead of the notation for cardinality and participation.

**Database design using ER model**

Database schema design should be considered as an iterative process.

To design a schema

```
1) Create an initial design
2) Loop
       Refine the design
   Until all data requirements are satisfied
```

It is important to maintain the least possible redundancy in a conceptual schema.

**A case study**

Here are the data requirements for design of a database for a company.

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, 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. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).

We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's name, gender, birth date, and relationship of the dependent to the employee.

An initial ER schema and ER diagram that show only the entity types and the attributes is prepared as shown below and this design is refined to show association among entity types.

**ER schema (Initial design)**

DEPARTMENT (<u>Name</u>, <u>Number</u>, {Locations}, Manager, Manager_start_date)

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

EMPLOYEE (Name, Ssn, Gender, Bdate, Address, Salary, Department, Supervisor)

DEPENDENT (Employee, Dependent_Name, Gender, Bdate, Relationship)

**ER diagram (initial design)**

See slide 4

**Refinement of initial design**

To refine the initial design the following steps are used.

1) Create a relationship type if an attribute points to some other entity type and remove the attribute.

2) Attribute present in several entity types may be removed from all entity types and represented as an entity type and assign an attribute to it and associate it with other entity types, creating relationship type.

3) An entity type E1 with one attribute and associated with only one other entity type E2, may be declared as an attribute of other entity type (E2)

*The relationship types obtained through this refinement are shown below.*

MANAGES, 1:1 between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial and DEPARTMENT participation total. The attribute Start_date is assigned to this relationship type.

WORKS_FOR, 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.

CONTROLS, 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, where as that of DEPARTMENT is partial.

SUPERVISION, 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are partial.

WORKS_ON, determined to be an M:N relationship type with attribute Hours, Both participations are total.

DEPENDENTS_OF, 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity.

***The complete ER schema of the company database is shown below.***

*The following are the entity types of the ER schema.*

DEPARTMENT (Name, Number, {Locations})

PROJECT(Name, Number, Location)

EMPLOYEE (Name(Fname, Minit, Lname), Ssn, Gender, Bdate, Address, Salary)

DEPENDENT (Dependent_Name, Gender, Bdate, Relationship)

*The following are the relationship types.*

MANAGES, 1:1 between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial and DEPARTMENT participation total. The attribute Start_date is assigned to this relationship type.

WORKS_FOR, 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.

CONTROLS, 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, where as that of DEPARTMENT is partial.

SUPERVISION, 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are partial.

WORKS_ON, determined to be an M:N relationship type with attribute Hours, Both participations are total.

DEPENDENTS_OF, 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity.

*For an ER diagram see the slide or the text book.*