Chapter 3: Data Modeling Using the Entity-Relationship Model

DATA MODELING USING THE ENTITY-RELATIONSHIP MODEL

HIGH-LEVEL CONCEPTUAL DATA MODELS FOR DATABASE DESIGN

Above figure shows a simplified description of the database design process. The first step shown is requirements collection and analysis. During this step, the database designers interview prospective database users to understand and document their data requirements. The result of this step is a concisely written set of users' requirements.
In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of the user-defined operations (or transactions) that will be applied to the database, including both retrievals and updates.

Once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model. This step is called conceptual design. The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints.

The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model such as the relational or the object-relational database model so the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called logical design or data model mapping, and its result is a database schema in the implementation data model of the DBMS.

The last step is the physical design phase, during which the internal storage structures, indexes, access paths, and file organizations for the database files are specified. In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

**AN EXAMPLE DATABASE APPLICATION**

We describe an example database application, called COMPANY that serves to illustrate the basic ER model concepts and their use in schema design. The COMPANY database keeps track of a company's employees, departments, and projects.

1. 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.

2. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

3. 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.

4. 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.
ENTITY TYPES, ENTITY SETS, ATTRIBUTES, AND KEYS
The ER model describes data as entities, relationships, and attributes.

Entities and Attributes
The basic object that the ER model represents is an entity, which is a "thing" in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for example, a company, a job, or a university course). Each entity has attributes—the particular properties that describe it. For example, an employee entity may be described by the employee’s name, age, address, salary, and job.

Several types of attributes occur in the ER model: simple versus composite, single-valued versus multivalued, and stored versus derived.

Composite versus Simple (Atomic) Attributes: Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. For example, the Address attribute of the employee entity can be subdivided into StreetAddress, City, State, and Zip, with the values "Gss BCA," "Tilakwadi," "Belgaum," and "590001." Attributes that are not divisible are called simple or atomic attributes.

Single-Valued versus Multivalued Attributes: Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity—for example, a Colors 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. A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

Stored versus Derived Attributes: In some cases, two (or more) attribute values are related—for example, the Age and BirthDate attributes of a person. For a particular person entity, 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 derivable from the BirthDate attribute, which is called a stored attribute.

Null Values: In some cases a particular entity may not have an applicable value for an attribute. For example, the ApartmentNumber attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes. For such situations, a special value called null is created. An address of a single-family home would have null for its ApartmentNumber attribute. Null can also be used if we do not know the value of an attribute for a particular entity—for example, if we do not know the home phone of "John Smith".

Complex Attributes: We 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. For example, 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 shown in Figure below.

\[
\{\text{AddressPhone}(\{\text{Phone}(\text{AreaCode},\text{PhoneNumber})\}), \text{Address}(\text{StreetAddress}(\text{Number},\text{Street},\text{ApartmentNumber}), \text{City},\text{State},\text{Zip})\})\}
\]

**Entity Types, Entity Sets, Keys, and Value Sets**

**Entity Types and Entity Sets:** A database usually contains groups of entities that are similar. For example, 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. An entity type defines a collection (or set) of entities that have the same attributes, for example two entity types in previous example are EMPLOYEE and COMPANY. The collection of all entities of a particular entity type in the database at any point in time is called an entity set.

An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name. Attribute names are enclosed in ovals and are attached to their entity type by straight lines. Composite attributes are attached to their component attributes by straight lines. Multivalued attributes are displayed in double ovals.

**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. For example, the Name attribute is a key of the COMPANY entity type, because no two companies are allowed to have the same name. 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. If a set of attributes possesses this property we call it as composite attribute and designate it as a key attribute of the entity type. In ER diagrammatic notation, each key attribute has its name underlined inside the oval.

**Value Sets (Domains) of Attributes:** Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity. Example, if the range of ages allowed for employees is between 16 and 70, we can specify the value set of the Age attribute of EMPLOYEE to be the set of integer numbers between 16 and 70. Value sets are not displayed in ER diagrams. Value sets are typically specified using the basic data types available in most programming languages, such as integer, string, boolean, float, enumerated type, subrange, and so on.
Chapter 3: Data Modeling Using the Entity-Relationship Model

Initial Conceptual Design of the COMPANY Database

According to the requirements listed, we can identify four entity types—one corresponding to each of the four items in the specification:

- An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and ManagerStartDate. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes, because each was specified to be unique.
- An entity type PROJECT with attributes Name, Number, Location, and ControllingDepartment. Both Name and Number are (separate) key attributes.
- An entity type EMPLOYEE with attributes Name, SSN (for social security number), Sex, Address, Salary, BirthDate, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name-FirstName, MiddleInitial, LastName—or of Address.
- An entity type DEPENDENT with attributes Employee, DependentName, Sex, BirthDate, and Relationship (to the employee).

Relationship Types, Relationship Sets, Roles and Structural Constraints

A relationship type $R$ among $n$ entity types $E_1, E_2, \ldots, E_n$ defines a set of associations or a relationship set—among entities from these entity types.

Degree of a Relationship Type: The degree of a relationship type is the number of participating entity types. For Example, the WORKSFOR relationship is of degree two. A relationship type of degree two is called binary, and one of degree three is called ternary. Relationships can generally be of any degree, but the ones most common are binary relationships.

Relationships as Attributes: It is sometimes convenient to think of a relationship type in terms of attributes. Consider the WORKS_FOR relationship type. One can think of an attribute called Department of the EMPLOYEE entity type whose value for each employee entity is (a reference to) the department entity that the employee works for. Hence, the value set for this Department attribute is the set of all DEPARTMENT entities, which is the DEPARTMENT entity set.

Role Names and Recursive Relationships: Each entity type that participates in a relationship type plays a particular role in the relationship. The role name signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means. For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer. However, in some cases the same entity type participates more than once in a relationship type in different roles. In such cases the role name becomes essential for distinguishing the meaning of each participation. Such relationship types are called recursive relationships. For Example, the SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity type.
Constraints on Relationship Types

Cardinality Ratios for Binary Relationships: The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in. For example, in the WORKS_FOR binary relationship type, DEPARTMENT: EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to any number of employees, but an employee can be related to only one department. The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

Participation Constraints and Existence Dependencies: The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints—total and partial—which we illustrate by example. 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. Thus, the participation of EMPLOYEE in WORKS_FOR is called total participation. Total participation is also called existence dependency. If we do not expect every employee to manage a department, the participation of EMPLOYEE in the MANAGES relationship type is partial, meaning that some or "part of the set of" employee entities are related to some department entity via MANAGES, but not necessarily all.

Weak Entity Types

Entity types that do not have key attributes of their own are called weak entity types. In contrast, regular entity types that do have a key attribute are called strong 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. We call this other entity type the identifying or owner entity type, and we call the relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type. A weak entity type always has a total participation constraint with respect to its identifying relationship, because a weak entity cannot be identified without an owner entity.

Refraining the ER Design for the Company Database

In our example, we specify the following relationship types:

1. MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation. The attribute StartDate is assigned to this relationship type.
2. WORKSFOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
3. CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.
4. SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.
5. **WORKS_ON**, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.

6. **DEPENDENTS_OF**, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

**ER DIAGRAMS, NAMING CONVENTIONS, AND DESIGN ISSUES**

**Summary of Notation for ER Diagrams**

In ER diagrams the emphasis is on representing the schemas rather than the instances. This is more useful in database design because a database schema changes rarely, whereas the contents of the entity sets change frequently. In addition, the schema is usually easier to display than the extension of a database, because it is much smaller.
Chapter 3: Data Modeling Using the Entity-Relationship Model

Figure above displays the COMPANY ER database schema as an ER diagram. We now review the full ER diagram notation. Entity types such as EMPLOYEE, DEPARTMENT, and PROJECT are shown in rectangular boxes. Relationship types such as WORKS JOR, MANAGES, CONTROLS, and WORKS ON are shown in diamond-shaped boxes attached to the participating entity types with straight lines. Attributes are shown in ovals, and each attribute is attached by a straight line to its entity type or relationship type. Component attributes of a composite attribute are attached to the oval representing the composite attribute, as illustrated by the Name attribute of EMPLOYEE. Multivalued attributes are shown in double ovals, as illustrated by the Locations attribute of DEPARTMENT. Key attributes have their names underlined. Derived attributes are shown in dotted ovals, as illustrated by the NumberOfEmployees attribute of DEPARTMENT.

Weak entity types are distinguished by being placed in double rectangles and by having their identifying relationship placed in double diamonds, as illustrated by the DEPENDENT entity type and the DEPENDENTS_OF identifying relationship type. The partial key of the weak entity type is underlined with a dotted line.

The cardinality ratio of each binary relationship type is specified by attaching a 1, M, or N on each participating edge. The cardinality ratio of DEPARTMENT:EMPLOYEE in MANAGES is 1:1, whereas it is 1:N for DEPARTMENT:EMPLOYEE in WORKS_FOR and M:N for WORKS ON. The participation constraint is specified by a single line for partial participation and by double lines for total participation (existence dependency).

We show the role names for the SUPERVISION relationship type because the EMPLOYEE entity type plays both roles in that relationship. Notice that the cardinality is 1:N from supervisor to supervisee because each employee in the role of supervisee has at most one direct supervisor, whereas an employee in the role of supervisor can supervise zero or more employees.

Figure below summarizes the conventions for ER diagrams.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ENTITY</td>
</tr>
<tr>
<td></td>
<td>WEAK ENTITY</td>
</tr>
<tr>
<td></td>
<td>RELATIONSHIP</td>
</tr>
<tr>
<td></td>
<td>IDENTIFYING RELATIONSHIP</td>
</tr>
<tr>
<td></td>
<td>ATTRIBUTE</td>
</tr>
</tbody>
</table>

Prof. Sushant. S. Sundikar

Introduction to Database Management Systems
Proper Naming of Schema Constructs

When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straightforward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema. We choose to use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type. In our ER diagrams, we will use the convention that entity type and relationship type names are in uppercase letters, attribute names are capitalized, and role names are in lowercase letters.

As a general practice, given a narrative description of the database requirements, the nouns appearing in the narrative tend to give rise to entity type names, and the verbs tend to indicate names of relationship types. Attribute names generally arise from additional nouns that describe the nouns corresponding to entity types. Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

SUMMARY

We started by discussing the role that a high level data model plays in the database design process, and then we presented an example set of database requirements for the COMPANY database. We then defined the basic ER model concepts of entities and their attributes. We discussed null values and presented the various types of attributes, which can be nested arbitrarily to produce complex attributes:

- Simple or atomic
- Composite
• Multivalued

We also briefly discussed stored versus derived attributes. We then discussed the ER model concepts at the schema or "intension" level:

• Entity types and their corresponding entity sets
• Key attributes of entity types
• Value sets (domains) of attributes
• Relationship types and their corresponding relationship sets
• Participation roles of entity types in relationship types

We presented two methods for specifying the structural constraints on relationship types. The first method distinguished two types of structural constraints:

• Cardinality ratios (1:1, 1:N, M:N for binary relationships)
• Participation constraints (total, partial)

Entity-Relationship schemas can be represented diagrammatically as ER diagrams. We showed how to design an ER schema for the COMPANY database by first defining the entity types and their attributes and then refining the design to include relationship types. Finally we displayed the ER diagram for the COMPANY database schema.