Data Modeling Overview

Miss. Brenda Tugume

Introduction

This unit is about one of the most critical stages in the development of a computerized information system – the design of data structures and the documentation of that design in a set of data model.

Objectives

By the end of this unit, you should be able to:

1. Know what data modeling and Entity Relationship is all about
2. Understand the E-R modeling constructs
3. Identify an entity in an E-R relation
4. Know what relationship is in E-R relationship model
5. Draw graph of relations in E-R relationship model
6. Know the advantages of using database management system

What is Data Modeling?

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern the operations on the objects. To use common analogy, the data model is equivalent to an architect's building plans.

There are two major methodologies used to create a data model: the **Entity-Relationship (ER) approach** and the **Object Model**. In this unit, we shall focus on the Entity Relationship approach.

Data Modeling in the Context of Database Design

Database design is defined as: “designing the logical and physical structure of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications”. The design process roughly follows five steps:

1. planning and analysis
2. conceptual design
3. logical design
4. physical design
5. implementation

Components of a Data Model

The data model gets its inputs from the planning and analysis stage. Here the modeler, along with system analysts, collects information about the requirements of the database by reviewing the existing documentation and interviewing end-users.

The data model has two outputs. The first is an entity-relationship diagram which represents the data structure in a pictorial form. Because the diagram is easily learned, it is valuable tool to communicate the model to the end-user. The second component is a data document. This is a document that describes in detail the data objects, relationships, and rules required by the database.

Why is data Modeling Important

The goal of the data model is to make sure that all the data objects required by the database are completely and accurately represented. Because the data model uses easily understood notations and natural language, it can be reviewed and verified as correct by the end-users.

The data model is also detailed enough to be used by the database developers as a “blueprint” for building the physical database.

The information contained in a data model will be used to define the relational tables, the primary and the foreign keys, stored procedures, and triggers.

A poorly designed database will require more time in the long-run. Without a careful planning you may create a database that omits data required to create critical reports, produces results that are incorrect or inconsistent, and is unable to accommodate changes in user’s requirements.

What Makes a Good Data Model?

The following are the characteristics of a good Data Model:

a. **Completeness:** Does the model support all necessary data?

b. **Non redundancy:** Does the model specify a database in which the same fact could be recorded more than once?

c. **Enforcement of Business Rules:** How accurately does the model reflect and enforce the rules that apply to the business data?

d. **Data Reusability:** Will the data stored in the database be reusable for the purposes beyond those anticipated in the process model?

e. **Stability and Flexibility:** How well will the model cope with possible changes to the business requirements?

f. **Elegance:** Does the data model provide a reasonable neat and simple classification of the data?

g. **Communication:** How effective is the model in supporting communication among the various stakeholders in the design of the system?

h. **Integration:** How will the proposed database fit with the organization’s existing and future database?

Activity A

1. What is Data Modeling?
2. Why is Data Modeling important?
3. What are the characteristics of a good data model?

The Entity-Relationship Model

The Entity-Relationship (ER) model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Today, ER model is commonly used for database design. For the database designer, the utility of the ER model is:

a. It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

b. 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.

c. In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

Basic Constructs of E-R Modeling

The ER model views the real world as a construct of entities and association between entities. E-R Modeling Constructs are: Entity, Relationship, Attributes, and Identifiers

It is important to get used to this terminology and to be able to use it at the appropriate time. For example, in the ER Model, we do not refer to tables. Here we call them 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:

I. EMPLOYEES
II. PROJECTS
III. CUSTOMER
IV. ORGANIZATION

V. PART
VI. INGREDIENT
VII. PURCHASE ORDER
VIII. CUSTOMER ORDER PRODUCT
IX. 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. The following terms are used with entity:

a. **Entity Occurrence:** 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.

An instance of an entity is like a specific example:

Bill Gates is an Employee of Microsoft

SPAM is a Product

Greenpeace is an Organization

Flour is an ingredient

b. **Associative entities (**also known as intersection entities) are entities used to associate two or more entities in order to reconcile a many-to-many relationship.

c. **Subtypes entities** are used in generalization hierarchies to represent a subset of instances of their parent entity, called the supertype, but which have attributes or relationships that apply only to the subset.

Attributes

Attributes describe the entity of which they are associated. i.e., properties used to distinguish one entity instance from another.

Attributes of entity EMPLOYEE might include:

I. EmployeeID
II. First Name
III. Last Name
IV. Street Address

v. City

vi. Local Government Area

vii. State

viii. Date of First Appointment

ix. Current Status

x. Date of Birth

Attributes of entity PRODUCT might include:

i. ProductID

ii. Product_Description

iii. Weight

iv. Size

v. Cost

A particular instance of an attribute is a value. For example, "Chukwudi R. Nnanna" 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, more commonly called keys, uniquely identify an instance of an entity. A descriptor describes a non unique characteristic of an entity instance.

Identifier

Identifier is a special attribute used to identify a specific instance of an entity.

o Typically we look for unique identifiers:

o Personal File Number uniquely identifies an EMPLOYEE

o CustomerID uniquely identifies a CUSTOMER

o We can also use two attributes to indicate an identifier: ORDER_NUMBER and LINE_ITEM uniquely identify an item on an order.

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 by their degree, connectivity, cardinality, direction, type, and existence. Not all modeling methodologies use all these classifications.

(a) 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. Special cases are the binary, and ternary, where the degree is 2, and 3, respectively.

Binary relationships, the association between two entities, are the most common type in the real world. A recursive binary relationship occurs when an entity is related to itself. An example might be "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.

(b) 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, one-to-many, and many-to many.

i. 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. For example, "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.

ii. A one-to-many (1:N) relationships 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 of a 1:N relationships is

a department has many employees

each employee is assigned to one department

iii. 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.

(c) 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.

(d) Type

An identifying relationship is one in which one of the child entities is also a dependent entity. A non-identifying relationship is one in which both entities are independent.

(e) Existence

Existence denotes whether the existence of an entity instance is dependent upon the existence of another, related, entity instance. The existence of an entity in a relationship is defined as either mandatory or optional. If an instance of an entity must always occur for an entity to be included in a relationship, then it is mandatory. An example of mandatory existence is the statement "every project must be managed by a single department". If the instance of the entity is not required, it is optional. An example of optional existence is the statement, "employees may be assigned to work on projects".

Generalization Hierarchies

A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into a higher level entity type called a *supertype or generic entity*. The lower-level of entities become the **subtype**, or categories, to the supertype. Subtypes are dependent entities.

Generalization occurs when two or more entities represent categories of the same real world object. For example, Wages_Employees and Classified_Employees represent categories of the same entity, Employees. In this example, Employees would be the supertype; Wages_Employees and Classified_Employees would be the subtypes.

Subtypes can be either mutually exclusive (disjoint) or overlapping (inclusive). A mutually exclusive category is when an entity instance can be in only one category. The above example is a mutually exclusive category. An employee can either be wages or classified but not both. An overlapping category is when an entity instance may be in two or more subtypes. An example would be a person who works for a university could also be a student at that same university. The completeness constraint requires that all instances of the subtype be represented in the supertype.

Generalization hierarchies can be nested. That is, a subtype of one hierarchy can be a supertype of another. The level of nesting is limited only by the constraint of simplicity. Subtype entities may be the parent entity in a relationship but not the child.

ER Notation

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. Today, there are a number of notations used; among the more common are Bachman, crow's foot, and IDEFIX.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The symbols used for the basic ER constructs are:

i. Entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.

ii. Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.

iii. Attributes, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.

iv. Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

v. Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.

Conclusion

The data model is relatively small part of the total systems specification but has a high impact on the quality and useful life of the system. Time spent producing the best possible design is very likely to be repaid many times over in the future.

Summary

In this unit, we have learnt that:

i. A data model is a plan for building a database. To be effective, it must be simple enough to communicate to the end user the data structure required by the database yet detailed enough for the database designer to use to create the physical structure.

ii. The Entity-Relationship Model is a conceptual data model that views the real world as consisting of entities and relationships. The model visually represents these concepts by the Entity-Relationship diagram.

iii. The basic constructs of the ER model are entities, relationships, and attributes.

iv. Entities are concepts, real or abstract, about which information is collected.

v. Relationships are associations between the entities.

vi. Attributes are properties which describe the entities.

Examples of these symbols are shown in Figure 2.1:

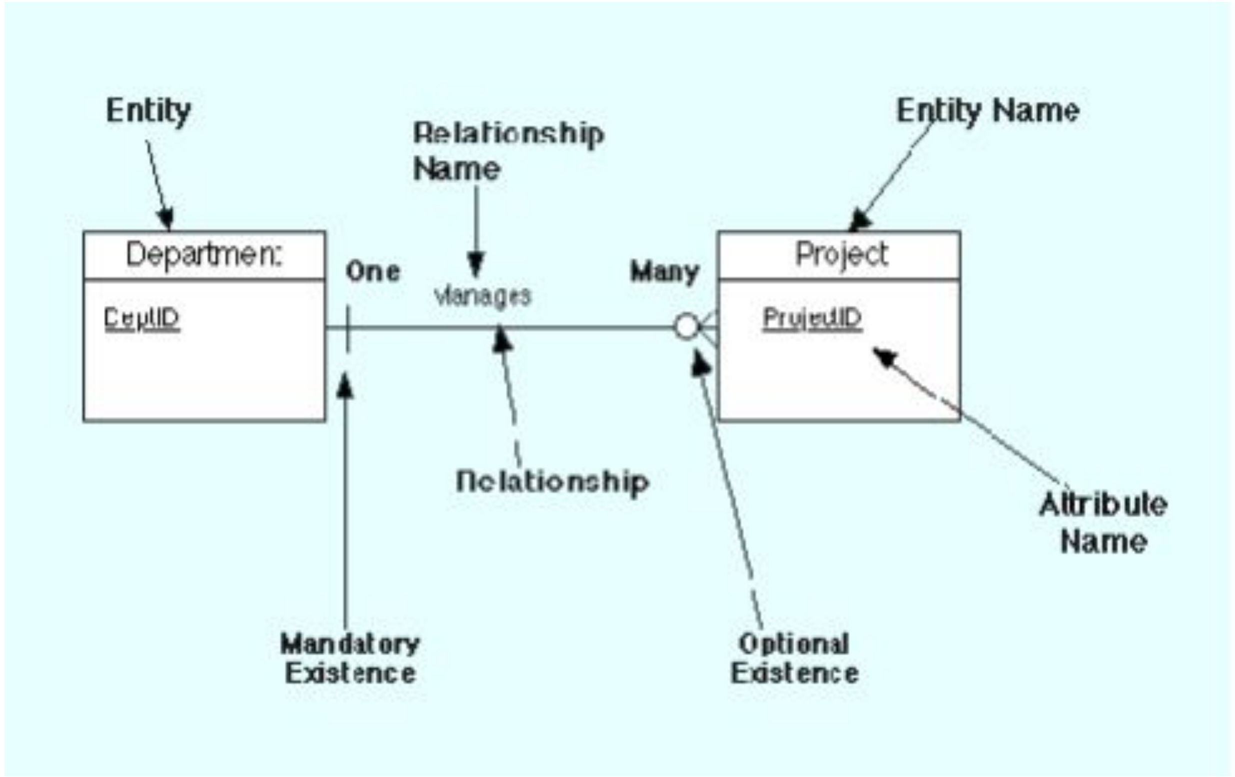

Figure 2.1: ER Notation

Source: http://www.utexas.edu/

Activity B

1. Come up with a list of attributes for each of the entities in section 3.3.1

2. Choose one of your attributes as the identifier for each of the entities.

Tutor Marked Assignment

1. Explain the following E-R Modeling Constructs with examples:

i. Entity

ii. Relationship

iii. Attributes

iv. Identifiers

2. What do you understand by the term Generalization Hierarchies?

Further Reading and other Resources

David M. Kroenke, David J. Auer (2008). Database Concepts. New Jersey . Prentice Hall

Elmasri Navathe (2003). Fundamentals of Database Systems. England. Addison Wesley.

Fred R. McFadden, Jeffrey A. Hoffer (1994). Modern Database management. England. Addison Wesley Longman

Graeme C. Simsion, Graham C. Witt (2004). Data Modeling Essentials. San Francisco. Morgan Kaufmann

Pratt Adamski, Philip J. Pratt (2007). Concepts of Database Management. United States. Course Technology