**Database Systems: CS-329**

**Week 03 : Data Models**

The entity relationship (ER) data model has existed for over 35 years. It is well suited to data modeling for use with databases because it is fairly abstract and is easy to discuss and explain. ER models are readily translated to relations. ER models, also called an ER schema, are represented by ER diagrams.

ER modelling is based on two concepts:

* Entities, defined as tables that hold specific information (data)
* Relationships, defined as the associations or interactions between entities

For the rest of this chapter, we will use a sample database called the COMPANY database to illustrate the concepts of the ER model. This database contains information about employees, departments and projects. Important points to note include:

* There are several departments in the company. Each department has a unique identification, a name, location of the office and a particular employee who manages the department.
* A department controls a number of projects, each of which has a unique name, a unique number and a budget.
* Each employee has a name, identification number, address, salary and birthdate. An employee is assigned to one department but can join in several projects. We need to record the start date of the employee in each project. We also need to know the direct supervisor of each employee.
* We want to keep track of the dependents for each employee. Each dependent has a name, birthdate and relationship with the employee.

## Entity, Entity Set and Entity Type

An entity is an object in the real world with an independent existence that can be differentiated from other objects. An entity might be

* An object with physical existence (e.g., a lecturer, a student, a car)
* An object with conceptual existence (e.g., a course, a job, a position)

Entities can be classified based on their strength. An entity is considered weak if its tables are existence dependent.

* That is, it cannot exist without a relationship with another entity
* Its primary key is derived from the primary key of the parent entity
  + The Spouse table, in the COMPANY database, is a weak entity because its primary key is dependent on the Employee table. Without a corresponding employee record, the spouse record would not exist.

An entity is considered strong if it can exist apart from all of its related entities.

* Kernels are strong entities.
* A table without a foreign key or a table that contains a foreign key that can contain nulls is a strong entity

Another term to know is entity type which defines a collection of similar entities.

An entity set is a collection of entities of an entity type at a particular point of time. In an entity relationship diagram (ERD), an entity type is represented by a name in a box.

## Attributes

Each entity is described by a set of attributes (e.g., Employee = (Name, Address, Birthdate (Age), Salary).

Each attribute has a name, and is associated with an entity and a domain of legal values. However, the information about attribute domain is not presented on the ERD.

## Types of Attributes

There are a few types of attributes you need to be familiar with. Some of these are to be left as is, but some need to be adjusted to facilitate representation in the relational model. This first section will discuss the types of attributes. Later on we will discuss fixing the attributes to fit correctly into the relational model.

## Simple attributes

Simple attributes are those drawn from the atomic value domains; they are also called single-valued attributes. In the COMPANY database, an example of this would be: Name = {John} ; Age = {23}

## Composite attributes

Composite attributes are those that consist of a hierarchy of attributes. Using our database example, and shown in Figure 8.3, Address may consist of Number, Street and Suburb. So this would be written as → Address = {59 + ‘Meek Street’ + ‘Kingsford’}

.

## Multivalued attributes

Multivalued attributes are attributes that have a set of values for each entity. An example of a multivalued attribute from the COMPANY database, as seen in Figure 8.4, are the degrees of an employee: BSc, MIT, PhD.

### Derived attributes

Derived attributes are attributes that contain values calculated from other attributes. An example of this can be seen in Figure 8.5.  Age can be derived from the attribute Birthdate. In this situation, Birthdate is called a stored attribute, which is physically saved to the database.

An important constraint on an entity is the key. The key is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set.

## Types of Relationships

Below are descriptions of the various types of relationships.

## One to many (1:M) relationship

A one to many (1:M) relationship should be the norm in any relational database design and is found in all relational database environments. For example, one department has many employees

## One to one (1:1) relationship

A one to one (1:1) relationship is the relationship of one entity to only one other entity, and vice versa. It should be rare in any relational database design. In fact, it could indicate that two entities actually belong in the same table.

## Many to many (M:N) relationships

For a many to many relationship, consider the following points:

* It cannot be implemented as such in the relational model.
* It can be changed into two 1:M relationships.
* It can be implemented by breaking up to produce a set of 1:M relationships.
* It involves the implementation of a composite entity.
* Creates two or more 1:M relationships.
* The composite entity table must contain at least the primary keys of the original tables.
* The linking table contains multiple occurrences of the foreign key values.
* Additional attributes may be assigned as needed.
* It can avoid problems inherent in an M:N relationship by creating a composite entity or bridge entity. For example, an employee can work on many projects OR a project can have many employees working on it,

**Ternary Relationships**

A ternary relationship is a relationship type that involves many to many relationships between three tables.

* For each n-ary (> 2) relationship, create a new relation to represent the relationship.
* The primary key of the new relation is a combination of the primary keys of the participating entities that hold the N (many) side.
* In most cases of an n-ary relationship, all the participating entities hold a **many** side.