8

CHAPTER 8 ENTITY RELATIONSHIP MODEL

ADRIENNE WATT

The Entity Relationship Data Model (ER) has existed for over 35 years.

The ER model is well-suited to data modelling for use with databases because it is fairly abstract and is easy to discuss and explain. ER models are readily translated to relations.

ER modelling is based on two concepts:

- Entities, that is, things. E.g. Prof. Ba, Course Database System.
- Relationships, that is, associations or interactions between entities.

E.g. Prof. Ba teaches course Database Systems

ER models (or ER schemas) are represented by ER diagrams.

The example database

For the next section we will look at a sample database called the COMPANY database to illustrate the concepts of Entity Relationship Model.

This database contains the information about employees, departments and projects:

- 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 unique name, a unique number and the budget.
- Each employee has name, an identification number, address, salary, 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 of the employees. Each dependent has 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 and can be differentiated from other objects

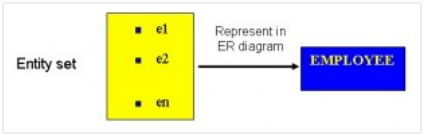
An entity might be

- 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

An Entity Type defines a collection of similar entities.

An Entity Set is a collection of entities of an entity type at a point of time. In ER diagrams, an entity type is represented by a name in a box.



Source: http://cnx.org/content/m28250/latest/

TYPES OF ENTITIES

Independent entities, also referred to as Kernels, are the backbone of the database. It is what other tables are based on. Kernels have the following characteristics:

- they are the 'building blocks' of a database
- the primary key may be simple or composite
- the primary key is not a foreign key
- they do not depend on another entity for their existence

Example: Customer table, Employee table, Product table

Dependent Entities, also referred to as derived, depend on other tables for their meaning.

- Dependent entities are used to connect two kernels together.
- They are said to be existent dependent on two or more tables.
- Many to many relationships become associative tables with at least two foreign keys.
- They may contain other attributes.
- The foreign key identifies each associated table.
- There are three options for the primary key:
  - use a composite of foreign keys of associated tables if unique
  - use a composite of foreign keys and qualifying column
  - create a new simple primary key

Characteristic entities provide more information about another table. These entities have the following characteristic.

- They represent multi-valued attributes.
- They describe other entities.
- They typically have a one to many relationship.
- The foreign key is used to further identify the characterized table.
- Options for primary key are as follows:
  - foreign key plus a qualifying column
  - or create a new simple primary key
    - Employee(EID, Name, Address, Age, Salary)
    - EmployeePhone(EID, Phone)

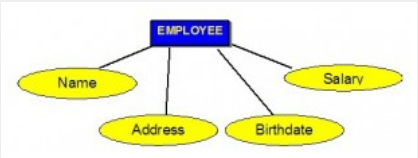
ATTRIBUTES

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

Each attribute has a name, associated with an entity and is associated with a domain of legal values. However the information

about attribute domain is not presented on the ER diagram.

In the diagram, each attribute is represented by an oval with a name inside.



Source: http://cnx.org/content/m28250/latest/

TYPES OF ATTRIBUTES

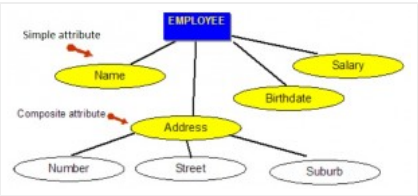
There are a few types of attributes you need to be familiar with. Some of these are to be left as is, but some need to be adjusted to facilitate representation in the relational model. This first section will discuss the types of attributes. Later on we will discuss fixing the attributes to fit correctly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

Simple attributes are attributes that are drawn from the atomic value domains

e.g. Name = {John} ; Age = {23} , also called Single valued.

Composite attributes: Attributes that consist of a hierarchy of attributes

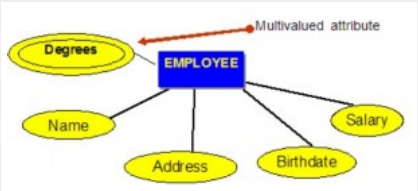
e.g. Address may consists of "Number", "Street" and "Suburb" → Address = {59 + 'Meek Street' + 'Kingsford'}



Source: http://cnx.org/content/m28250/latest/

Multivalued attributes: Attributes that have a set of values for each entity

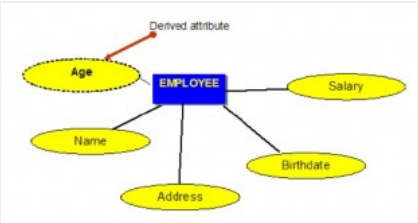
e.g. Degrees of a person: ' BSc' , 'MIT', 'PhD'



Source: http://cnx.org/content/m28250/latest/

Derived attributes: Attributes contain values that are calculated from other attributes

e.g. Age can be derived from attribute DateOfBirth. In this situation, DateOfBirth might be called Stored Attribute.



Source: http://cnx.org/content/m28250/latest/

Source: http://cnx.org/content/m28250/latest/

KEYS

An important constraint on the entities 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.

CANDIDATE KEY

- a simple or composite key that is unique and minimal
- unique – no two rows in a table may have the same value at any time
- minimal – every column must be necessary for uniqueness
- For example, for the entity
  Employee(EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID)

  Possible candidate keys are EID, SIN

FirstName and Last Name – assuming there is no one else in the company with the same name, Last Name and DepartmentID – assuming two people with the same last name don't work in the same department.

EID, SIN are also candidate keys

COMPOSITE KEY

- Composed of more than one attribute
- For example, as discussed earlier
  First Name and Last Name – assuming there is no one else in the company with the same name, Last Name and Department ID – assuming two people with the same last name don't work in the same department.
- A composite key can have two or more attributes, but it must be minimal.

PRIMARY KEY

- A candidate key is selected by the designer to uniquely identify tuples in a table. It must not be null.
- A key is chosen by the database designer to be used as an identifying mechanism for the whole entity set. This is referred to as the primary key. This key is indicated by underlining the attribute in the ER model.
- For example **Employee**(EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID) – EID is the Primary key.

SECONDARY KEY

- an attribute used strictly for retrieval purposes (can be composite), for example: Phone number, Last Name and Phone number, etc.
- all other candidate keys not chosen as the primary key
- An attribute in one table that references the primary key of another table OR it can be null.
- Both foreign and primary keys must be of the same data type
- For example: **Employee**(EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID) – DepartmentID is the Foreign key.

ALTERNATE KEY

- all other candidate keys not chosen as the primary key

FOREIGN KEY

- An attribute in one table that references the primary key of another table OR it can be null.

- Both foreign and primary keys must be of the same data type
- For example: **Employee**(EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID) – DepartmentID is the Foreign key.

NULLS

This is a special symbol, independent of data type, which means either unknown or inapplicable. (*it does not mean zero or blank*)

- No data entry
- Not permitted in primary key
- Should be avoided in other attributes
- Can represent
  - An unknown attribute value
  - A known, but missing, attribute value
  - A "not applicable" condition
  - Can create problems when functions such as COUNT, AVERAGE, and SUM are used
  - Can create logical problems when relational tables are linked

NOTE: the result of a comparison operation is null when either argument is null. The result of an arithmetic operation is null when either argument is null (except functions which ignore nulls)

Salary_tbl

emp#	jobName	salary	commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

PROBLEM: Find all employees (EMP#) in Sales whose salary plus commission is greater than 30,000.

SELECT emp# FROM salary_tbl

WHERE jobname = 'Sales' AND

(commission + salary) > 30000 –> E10 and E12

This result does not include E13 because of the Null value in Commission. To ensure the row with the null value is included, we need to look at the individual fields. By adding commission and Salary for employee E13, the result will be a null value. The solution is shown below.

SELECT emp# FROM salary_tbl

WHERE jobName = 'Sales' AND

(commission > 30000 OR salary > 30000 OR

(commission + salary) > 30000) –>E10 and E12 and E13

ENTITY TYPES

EXISTENCE DEPENDENCY

- An entity's existence is dependent on the existence of the related entity.
- It's existence-dependent if it has a mandatory foreign key. That is a foreign key attribute that cannot be null.

- Spouse entity is existence dependent on the employee

WEAK ENTITIES

- These tables are existence dependent.
- They cannot exist without entity with which it has a relationship.
- Primary key is derived from the primary key of the parent entity
  - The spouse table is a weak entity because its PK is dependent on the employee table. Without a corresponding employee record, the spouse record could not exist

STRONG ENTITIES

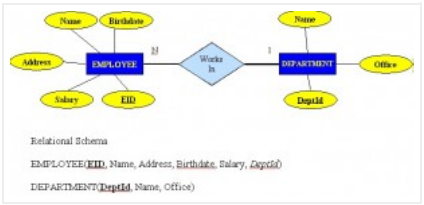
- If an entity can exist apart from all of its related entities it's considered a strong entity.
- Kernels are strong entities.
- A table without a foreign key is or a table that contains a foreign key which can contain NULLS is a strong entity.

RELATIONSHIPS

These are the glue that holds the tables together. They are used to connect together related information in other tables.

1:M RELATIONSHIP

- Should be the norm in any relational database design
- Relational database norm
- Found in any database environment
  e.g. One department has many employees



Source: http://cnx.org/content/m28250/latest/

1:1 RELATIONSHIP

- Should be rare in any relational database design
- One entity can be related to only one other entity, and vice versa
- Could indicate that two entities actually belong in the same table

e.g. A professor chairs one department and a department has only one chair.

1:1 EXAMPLE:

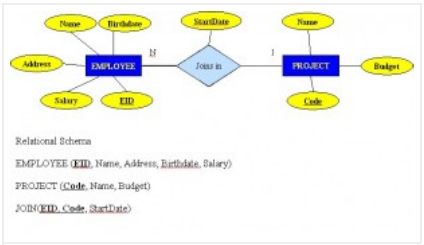
- An employee is associated with one spouse, and one spouse is associated with one employee.
- Cannot be implemented as such in the relational model
- M:N relationships can be changed into two 1:M relationships
- Can be implemented by breaking it up to produce a set of 1:M relationships
- Can avoid problems inherent to M:N relationship by creating a composite entity or bridge entity
- A student has many classes and a class can hold many students.
- Implementation of a composite entity
- Yields required M:N to 1:M conversion
- Composite entity table must contain at least the primary keys of original tables

- Linking table contains multiple occurrences of the foreign key values
- Additional attributes may be assigned as needed

M:N RELATIONSHIPS

Mapping M-N Binary Relationship Type: For each M-N Binary Relationship, identify two relations A, B represent two entity type participating in R. Create a new relation S to represent R. Include in S as foreign keys the primary keys of A and B and all the simple attributes of R. The combination of primary keys of A and B will make the primary key of S.

M:N RELATIONSHIPS



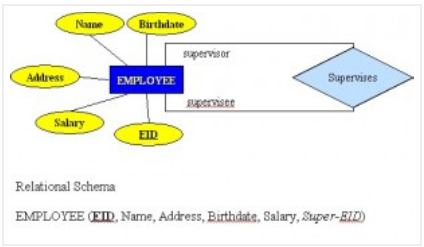
Source: http://cnx.org/content/m28250/latest/

UNARY RELATIONSHIPS (RECURSIVE)

One in which a relationship exists between occurrences of the same entity set

The two keys (primary key and foreign key) are the same but they represents two entities of different roles relate to this relationship.

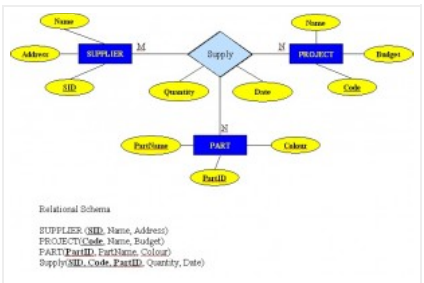
In some entities, a separate column can be created that refers to the primary key of the same entity set.



Source: http://cnx.org/content/m28250/latest/

TERNARY RELATIONSHIPS

Mapping Ternary Relationship Type: For each n-ary (> 2) Relationships create a new relation to represent the relationship. The primary key of the new relation is the 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. Source: http://cnx.org /content/m28250/latest/



Source: http://cnx.org/content/m28250/latest/

RELATIONSHIP STRENGTH

Relationship strength is based on how the primary key of a related entity is defined.

Weak Relationships (non-identifying relationship)

- Exists if the PK of the related entity does not contain a PK component of the parent entity.
- Customer(**custid**, custname)
- Order(**orderID**, custid, date)

Strong Relationship (identifying)

- Exists when the PK of the related entity contains the PK component of the parent entity.
- Course(**CrsCode**, DeptCode, Description)
- Class(**CrsCode, Section**, ClassTime…)