ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Assignment # 5**

**SUBMITTED BY:**

**Apil Neupane**

**013BSCCSIT008**

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Mr. Sanjay Kr. Yadav**  **Lecturer** |  |
| **Department of Computer Science** | |

Submission Date: 27thAugust 2015

Entity Relationship Model

**Questions**

1. What do you mean by Entity- Relationship Diagram? Explain.

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

* Entities
* Relationships
* Attributes

An entity is a real-world item or concept that exists on its own. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

An attribute of an entity is a particular property that describes the entity. A relationship is the association that describes the interaction between entities. Cardinality, in the context of ERD, is the number of instances of one entity that can, or must, be associated with each instance of another entity. In general, there may be one-to-one, one-to-many, or many-to-many relationships.

Steps involved in creating an ERD include:

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analyzing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD
5. Define entity and give an example.

In general, an entity (pronounced N-tih-tee) is an existing or real thing.

An entity is a grouping of things with rules or data in common. Entities are usually used to establish a mapping between an object and to a table in the database. Entities are also known as domain objects.

By defining entities, the same set of rules can be used for multiple instances of the same type, and rules can be written which relate to all of those instances.

For example, parent, children can be the entity.

1. Explain the different between an entity class and an entity instance.

An entity calss often represents a group of people (eg children, applicants, stakeholders) but it can also represent a group of objects (eg textbooks), activities (eg assignments) or concepts (eg school terms).

A member of the entity group is called an entity instance. For example, if a family had 2 children, Sarah and Peter, Sarah would be one instance of "the child" entity and Peter would be another instance of "the child" entity. By creating an entity to represent "the child", information such as the child's age can be collected for each child.

1. Define attribute and its types.

Attributes are, simply put, the characteristics of entities. Some entities can have many attributes while others may only have a couple. As well, there are five categories that attributes are classified in.

1. **Required or Optional Attributes:**

A required attribute is an attribute that must have a value in it, while an optional attribute may not have a value in it and can be left blank. The reasoning for making an attribute required is to put emphasis on what is important in that entity and what makes it stand out from other entities.

Example: Consider the entity Student above stud\_LastName and studFirstName would be required attributes as it uniquely defines that table and we assume all students have a first and last name. Optional attributes in the table Student could be stu\_MiddleName, stu\_Email, and stu\_Phone since some students may not have a middle name, a phone number, or an email address .

1. **Keys and non-keys Attributes:**

In every entity an attribute or grouped attributes uniquely identify that entity. These attributes are the key attributes and range from Primary key (single attribute identifier) to a Composite Key (Multi attribute Identifier). The rest of the attributes after the identifier are considered the non-key attributes or descriptors, which just describe the entity.

Example: Above in the table Student there is only one unique identifier, stu\_LastName, which is the primary key of the table. The rest of the attributes are descriptors.

1. **Single and Composite Attributes**

Attributes can be classified as having many parts to them or just a single unbreakable attribute. The composite attribute is an attribute that can be subdivided into other single attributes with meanings of their own. A single attribute is just an attribute that cannot be subdivided into parts.

Example: Imagine from the entity Student that instead of having the three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName it had one attribute called stu\_Name. The attribute stu\_Name would be considered a composite attribute since it can be subdivided into the other three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName. The rest of attributes would be consider single attributes since they can't be subdivided into parts.

1. **Single-valued and multi-valued Attributes**

Attributes can be classified as single or multi-value. The single-value attribute can only have one value, while the multi-valued attributes usually can store multiple data in them.

Example: In the entity Student, stu\_Address could be considered a multi-value attribute since a student could have multiple addresses where he lives at. An example of a single-value attribute would be stu\_LastNamesince a student usually has one last name that uniquely identifies him/her.

1. **Derived Attributes**

The last category that attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example: In the entity Student, stu\_Age would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.﻿﻿

Examples of derived attributes are: salary and age.

DOB is not an example of a derived attribute because it is inputted information and not calculated.

1. What is derived attributes?

The last category that attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example: In the entity Student, stu\_Age would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.﻿﻿

Examples of derived attributes are: salary and age.

DOB is not an example of a derived attribute because it is inputted information and not calculated.

1. Define relationship and give an example.

A relationship is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items.

For example, in a bank database a CUSTOMER\_MASTER table stores customer data with a primary key column named CUSTOMER\_ID; it also stores customer data in an ACCOUNTS\_MASTER table, which holds information about various bank accounts and associated customers. To link these two tables and determine customer and bank account information, a corresponding CUSTOMER\_ID column must be inserted in the ACCOUNTS\_MASTER table, referencing existing customer IDs from the CUSTOMER\_MASTER table. In this case, the ACCOUNTS\_MASTER table’s CUSTOMER\_ID column is a foreign key that references a column with the same name in the CUSTOMER\_MASTER table. This is an example of a relationship between the two tables.

1. Explain the difference between a relationship class and a relationship instance.
2. Define degree of relationship.

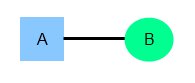
Degree of relationship refers to the number of participating entities in a relationship. The different degrees of relationship are:

1. 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".
2. 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.
3. List and give an example of the three types of binary relationships. Draw an E-R diagram for each.

There are three types of binary relationships between entities they are:

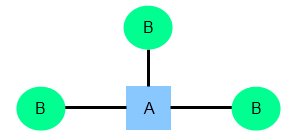
**1. One-to-One**

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



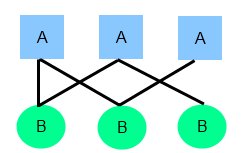
**2. One-to-Many**

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



## 3. Many-to-Many

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



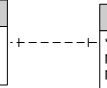
1. Define the terms maximum cardinality and minimum cardinality.

### Maximum Cardinality

Maximum cardinality indicates how many instances are participating in a relationship. It is always 1 or many.

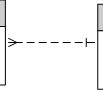
The possibilities include one-to-one (1:1), one-to-many (1:M), or many-to-many (M:N).

A 1:1 relationship can be thought of as the relationship between a football stadium and the home team. There can be only one team per stadium. You would denote this by having a perpendicular dash next to each entity in the relationship.



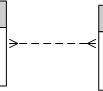
1:1 Relationship

A 1:M or M:1 relationship, involves one instance (the parent) of an entity in a relationship that connects to many instances (the children) in the other entity. This is the most common relationship seen in the relational database. Logically it should make sense, for example a single Course has many Classes/Sections, one Employee takes/instructs many Classes, or one Building has many Rooms. You illustrate the many by using a crow's foot.



M:1 Relationship (read left to right)

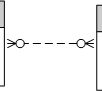
A M:N relationship can be used in a conceptual model to illustrate a situation where many instances of one entity in a relationship with many instances of the other. M:N relationships may have additional information that is stored as part of the instance of the relationship rather than with either entity.



M:N Relationship

### Minimum Cardinality

The minimum cardinality indicates the smallest number of participants in a relationship, which can be 0 or 1 (optional or mandatory). When evaluating minimum cardinality, you should think about what is actually taking place. Rarely is there a situation that is mandatory-to-mandatory (difficult to implement because you are stating the instances must both exist simultaneously) or optional-to-optional (an "open design," usually shown with a M:N), rather it is some form of optional-to-mandatory or mandatory-to-optional. For example, you can read that a building must conceptually be mandatory for a room to exist, but the building can exist without rooms. Ultimately, you are defining the order of adding data to your database. The building instance must be in the database before any room instances. The inner marks indicate minimum cardinality below.



1. Explain the distinctions among the terms primary key, candidate key and super key.

**Key** A key is a single or combination of multiple fields. Its purpose is to access or retrieve data rows from table according to the requirement. The keys are defined in tables to access or sequence the stored data quickly and smoothly. They are also used to create links between different tables.

**Types of Key**

* 1. **Primary Key:**

A Primary Key is a minimal Candidate Key, which is to say all constituent columns are strictly required in order to ensure uniqueness.

In other words, the attribute or combination of attributes that uniquely identifies a row or record in a relation is known as primary key.

* 1. **Candidate Key or Alternate key** A relation can have only one primary key. It may contain many fields or combination of fields that can be used as primary key. One field or combination of fields is used as primary key. The fields or combination of fields that are not used as primary key are known as candidate key or alternate key.
  2. **Super Key:**

A Super Key is simply a non-minimal Candidate Key, that is to say one with additional columns not strictly required to ensure uniqueness of the row.

In other words, a **superkey** is a combination of attributes that can be uniquely used to identify a database record. A table might have many superkeys. Candidate keys are a special subset of superkeys that do not have any extraneous information in them.

1. What are the main building modules of the entity relationship model? Discuss each one.

The Entity-Relationship model is a top-down approach to design database that is based on uniquely identifiable object. If begins by identifying things that are uniquely distinguishable called entities and relationships among these entities.  
The main building modules of the Entity-Relationship model are:

1. Entities: An Entity is a basic object of ER-model which is an object in real world that can be distinguishable and can exists independently.
2. Relationships: Relationship defines the association among two entities. Suppose, consider student and a class are the two entities. These entities are associated as “student studies in class”. Hence studies are a relationship between the two entities, student and class.
3. Attributes: The properties of the entities are called attributes.  
   For example if we consider a mobile phone as an entity then each mobile well have its own color, design, model company. All these are the attributes of the mobile entity.

13. What is composite attributes, when it is used?

14. Explain the difference between single-value attributes and simple attributes.

15. Discuss the difference between a composite key and a composite attribute. How would each indicate in an E-R diagram?

16. What two courses of action are available to a designer when a multi-valued attribute is encountered?

17. Explain the various terms of an E-R model and how are they represented in an E-R model?

18. Explain the concept of dependent entities? Give example.

19. What is the difference total and partial participation? Explain.

20. What do you mean by mapping cardinalities? Explain various types of cardinalities. 21. What is the difference between single-value and multi-valued attributes? Explain 22. Explain the concept of participation constraints.

23. Difference the binary relationship with ternary relationship with example. 24. Explain the difference between weak and strong entity set.

25. Define the components of extended E-R features.

26. Define the concept of aggregation. Give two examples of where this concept is useful. 27. Explain the distinction between disjoint and overlapping constraints.

28. Explain the distinction between total and partial constraints.

29. Write short notes on:

1. Specialization
2. Generalization
3. Aggregation