ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



# **DATABASE MANAGEMENT SYSYTEM**

Lab Assignment #5

**Submitted by:**

NARAYAN PANTHI

**Submitted to:**

Er. Sanjay Kumar Yadav

Lecturer  
Department of Computer Science

**Date of Submission: August 27, 2015**

# What do you mean by Entity- Relationship Diagram? Explain

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system.

An ERD is a data modeling technique that can help define business processes and can be used as the foundation for a relational database.

The three main cardinal relationships are:

* **One-to-one (1:1).** For example, if each customer in a database is associated with one mailing address.
* **One-to-many (1:M).** For example, a single customer might place an order for multiple products. The customer is associated with multiple entities, but all those entities have a single connection back to the same customer.
* **Many-to-many (M:N).**For example,at a company where all call center agents work with multiple customers, each agent is associated with multiple customers, and multiple customers might also be associated with multiple agents.

# Define entity and give an example.

An entity is something that exists in itself, actually or potentially, concretely or abstractly, physically or not. It need not be of material existence. In particular, abstractions and legal fictions are usually regarded as entities. Entities are objects or concepts that represent important data. They are typically nouns, e.g. customer, supervisor, location, or promotion.

* **Strong entities** exist independently from other entity types. They always possess one or more attributes that uniquely distinguish each occurrence of the entity.
* **Weak entities** depend on some other entity type. They don't possess unique attributes (also known as a primary key) and have no meaning in the diagram without depending on another entity. This other entity is known as the owner.
* **Associative entities** are entities that associate the instances of one or more entity types. They also contain attributes that are unique to the relationship between those entity instances.

**For example**, to develop a company's database for maintaining information on employees, the application should be able to store and provide data on employee such as when was the employee was hired; is the employee still with the company; if the employee has left the company when did he leave the company; which department does employee work for; who is his/her manager; what is his/her skill level etc. In this example, the entities are company, department, employee, manager.

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

* A **page** may describe a **class** or **instance** of an **entity** or **relationship**. Entities play specific **roles** in relationships.
  + In "A writer may write about a topic", "a writer" and "a topic" are classes of entities that may play particular roles in the "write about" relationship.
  + In "Rich Morin writes about Ontiki", "Rich Morin" and "Ontiki" are instances of these entities, plugged into the appropriate roles in an instance of the relationship.
* Classes define the nature of characteristics that instances may have.
  + The class "writes about", for example, is defined as having two roles: "a writer" and "a topic".
* Instances define the specific details of these characteristics. Each class or instance has a name.
  + Classes have descriptive names, as do instances of entities. Instances of relationships have mechanically-generated IDs.
* An instance may only have one class.
* A class may have zero or more instances.
* A class may have one or more super-classes (parents).
* A class may have zero or more of super-classes (children).
* An entity may play zero or more roles.
* A relationship may have one or more roles (few exceed four).

# Define attribute and its types.

An Attribute is a property that describes an entity. For example, to develop a company's database for maintaining information on employees, the application should be able to store and provide data on employee such as when was the employee was hired; is the employee still with the company; if the employee has left the company when did he leave the company; which department does employee work for; who is his/her manager; what is his/her skill level etc. In this example, the entities are company, department, employee, manager. In the above example, the employee is the entity and employee’s name, age, address, salary and job etc are the attribute.

* **Simple and composite Attributes**

Composite attributes can be divided into smaller subparts. These subparts represent basic attributes with independent meanings of their own. For example, take Name attributes. We can divide it into sub-parts like First\_name, Middle\_name, and Last\_name.

Attributes that can’t be divided into subparts are called Simple or Atomic attributes. For example, Employee Number is a simple attribute. Age of a person is a simple attribute.

* **Single-valued and multi-valued Attributes**

Attributes that can have single value at a particular instance of time are called singlevalued. A person can’t have more than one age value. Therefore, age of a person is a single-values attribute. A multi-valued attribute can have more than one value at one time. For example, degree of a person is a multi-valued attribute since a person can have more than one degree. Where appropriate, upper and lower bounds may be placed on the number of values in a multi-valued attribute. For example, a bank may limit the number of addresses recorded for a single customer to two.

* **Stored and derived Attributes**

There may be a case when two or more attributes values are related. Take the example of age. Age of a person can be can be calculated from person’s date of birth and present date. Difference between the two gives the value of age. In this case, age is the derived attribute.

The attribute from which another attribute value is derived is called stored attribute.

# What is derived attributes?

The attributes 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. There may be a case when two or more attributes values are related. Take the example of age. Age of a person can be calculated from person’s date of birth and present date. Difference between the two gives the value of age. In this case, age is the derived attribute. Derived attributes are usually created by a formula or by a summary operation on other attributes.

# Define relationship and give an example.

Relationships are meaningful associations between or among entities. They are usually verbs, e.g. assign, associate, or track. A relationship provides useful information that could not be discerned with just the entity types. Weak relationships, or identifying relationships, are connections that exist between a weak entity type and its owner. After two or more entities are identified and defined with attributes, the participants determine if a relationship exists between the entities. A relationship is any association, linkage, or connection between the entities of interest to the business; it is a two-directional, significant association between two entities, or between an entity and itself. Each relationship has a name, an optionality (optional or mandatory), and a degree (how many). A relationship is described in real terms.

An example of a relationship would be:·

* Employees are assigned to projects·
* Projects have subtasks·
* Departments manage one or more projects

# Explain the difference between a relationship class and a relationship instance.

A page may describe a class or instance of an entity or relationship. Entities play specific roles in relationships. In "A writer may write about a topic", "a writer" and "a topic" are classes of entities that may play particular roles in the "write about" relationship. In "Rich Morin writes about Ontiki", "Rich Morin" and "Ontiki" are instances of these entities, plugged into the appropriate roles in an instance of the relationship.

Classes define the nature of characteristics that instances may have. The class "writes about", for example, is defined as having two roles: "a writer" and "a topic".

Instances define the specific details of these characteristics. An instance of "writes about" would define the entities (i.e., "Rich Morin" and "Ontiki") that fill these roles. Each class or instance has a name. Classes have descriptive names, as do instances of entities. Instances of relationships have mechanically-generated IDs.

# Define degree of relationship.

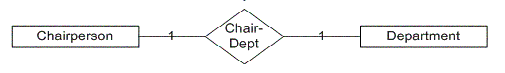
Degree of relationship refers to the number of participating entities in a relationship. If there are two entities involved in relationship then it is referred to as binary relationship. If there are three entities involved then it is called as ternary relationship and so on.

On the other hand, it is the cardinality of relationship that defines the number of instances of one entity as it relates to the number of instances of the other entity. Based on the different combinations between two entities we can have either one-to-one, one-to-many or many-to-many relationship.

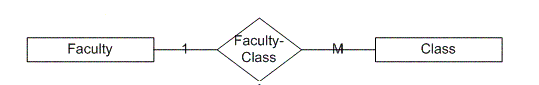
# List and give an example of the three types of binary relationships. Draw an E-R diagram for each.

Answer: The three different types of binary relationships are:

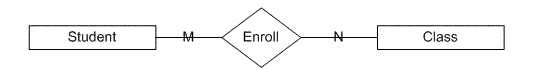
1. 1:1 - a single entity instance of one type is related to a single-entity instance of another type.



1. 1: M - a single entity instance of one type is related to many-entity instances of another type.



1. M: N - many-entity instances of one type relate to many-entity instances of another type.



# Define the terms maximum cardinality and minimum cardinality.

Maximum cardinality indicates how many instances are participating in a relationship. 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.

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, we 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.

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

Super Keys: Super key stands for superset of a key. A Super Key is a set of one or more attributes that are taken collectively and can identify all other attributes uniquely. Each super key is able to uniquely identify each tuple (record).

Candidate Keys: Candidate Keys are super keys for which no proper subset is a super key. In other words candidate keys are minimal super keys. Candidate keys are a super key which are not having any redundant attributes.

Primary Key: It is a candidate key that is chosen by the database designer to identify entities with in an entity set or a key which is used to uniquely identify each record is known as primary key. Primary key is the minimal super keys.

Ideally a primary key is composed of only a single attribute. But it is possible to have a primary key composed of more than one attribute.

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

Answer:

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:

a. Entities

b. Relationships

c. Attributes

* 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.

* 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 is a relationship between the two entities, student and class.

* 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?

A composite attribute is an attribute that can be further subdivided. For example the attribute ADDRESS can be subdivided into street, city, state and zip code. Consider an attribute such as name that comprises first and last names. For example, suppose an employee's name is John McKenzie. The first name is John and the last name is McKenzie. It is easy to appreciate that one application may only want the last name, another may display the first name followed by the last name, and yet another application may display the last name, a comma, and then the first name.

Since it is meaningful to decompose empName into two attributes for first name, firstName, and last name, lastName, we consider the name attribute to be a composite attribute (firstName and lastName are non-composite; they are atomic attributes). A composite attribute is an attribute that is shown as comprising two or more simpler attributes; we show a composite attribute below.

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

Answer: A single-valued attribute is one that can have only one value. For example, a person has only one first name and only one social security number. A simple attribute is one that cannot be decomposed into its component pieces. For example, a person's sex is classified as either M or F and there is no reasonable way to decompose M or F. Similarly, a person's first name cannot be decomposed into meaningful components. (In contrast, if a phone number includes the area code, it can be decomposed into the area code and the phone number itself. And a person's name may be decomposed into a first name, an initial, and a last name.)  
  
Single-valued attributes are not necessarily simple. For example, an inventory code HWPRIJ23145 may refer to a classification scheme in which HW indicates HardWare, PR indicates Printer, IJ indicates InkJet, and 23145 indicates an inventory control number. Therefore, HWPRIJ23145 may be decomposed into its component parts... even though it is single-valued. To facilitate product tracking, manufacturing serial codes must be single-valued, but they may not be simple.

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

A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table. When a primary key is created from a combination of 2 or more columns, the primary key is called a composite key.

Composite attributes are made of more than one simple attribute. Simple attributes are atomic values, which cannot be divided further. For example, a student's complete name may have first\_name and last\_name.

# 16. What two courses of action are available to a designer when a multivalued attribute is encountered ?

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

The various terms of an E-R model are:

**Rectangles**: representing entity sets.

**Ellipses**: representing attributes.

**Diamonds**: representing relationship sets.

**Lines**: linking attributes to entity sets and entity sets to relationship sets.

**Double ellipses**: represents multi-valued attributes.

**Dashed ellipses**: represents derived attributes

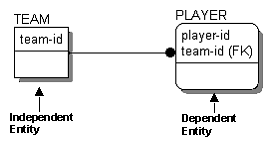
**Double lines**: represents total participation of entity in relationship sets 

**Double rectangles**: represents weak entity sets (discuss in later section)

**Underline**: indicates primary key attributes

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

In relational terms, a child entity that depends on the foreign key attribute for uniqueness is called a dependent entity.



Dependent entities are further classified as existence dependent, which means the dependent entity cannot exist unless its parent does, and identification dependent, which means that the dependent entity cannot be identified without using the key of the parent. The PLAYER entity is identification dependent but not existence dependent, since PLAYERs can exist if they are not on a TEAM.

# 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

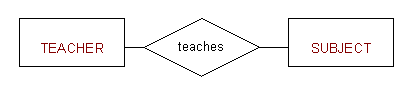
22. Explain the concept of participation constraints.

# 23. Difference the binary relationship with ternary relationship with example.

The degree of a relationship is the number of entity types that participate in the relationship. The three most common relationships in ER models are Binary, Unary and Ternary.

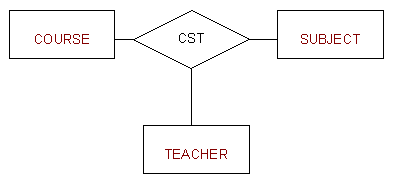
A **binary relationship**is when two entities participate and is the most common relationship degree.

**For Example:**

****

A **ternary relationship**is when three or (n) relations have relationship between themselves, and providing all relationship between them makes the database complex, so here the relationships will turned into a relation which has one-to-many/one-to-one with base relations.

**For Example:**  
The University might need to record which teachers taught which subjects in which courses.



# 24. Explain the difference between weak and strong entity set.

The difference between a weak and a strong entity is

**Strong entity set:**

* The relationship between two strong entity set is represented by a diamond symbol
* The line connecting strong entity set with the relatiionship is single\
* Member of a strong entity set is a dominant entity
* It has its own primary key and it is represented by a rectangle

**Weak entity set:**

* The relationship between one strong entity set is represented by a double diamond sign
* The line connecting weak entity set with the relationship is double
* Member of a weak entity set is a subordinate entity.
* It is represented by a double rectangle
* The primary key of a weak entity is found by taking the primary key of the strong entity on which it is existence-dependent, plus the discriminator of the weak entity set.

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

The extended entity-relationship (EER) model is a language for definition of structuring (and functionality) of database or information systems.

* Created to design more accurate database schemas.
* Reflect the data properties and constraints more precisely.
* More complex requirements than traditional applications

**Components:**

* Stability - will the diagram support business needs that change over time?
* Breadth - can this diagram accommodate all of the data we need to store?
* Flexibility - can data in this model be re-allocated to support additional information requirements?
* Efficiency - does this model represent the simplest solution? Is data modeled with the appropriate symbols?
* Accessibility - can both creators and end users of the ERD easily understand it?
* Conformity - will the design integrate seamlessly with any existing database structure?

# 26. Define the concept of aggregation. Give two examples of where this concept is useful.

Aggregation is an abstraction through which relationships are treated as higher-level entities. Thus the relationship between entities A and B is treated as if it were an entity C.

It is a way of composing different abstractions together in defining a class.

Some examples of this are:

* Employees work for projects. An employee working for a particular project uses various machinery.
* Manufacturers have tie-ups with distributors to distribute products. Each tie-up has specified for it the set of products which are to be distributed.
* A car object is an aggregation of engine, seat, wheels and other objects.

# 27. Explain the distinction between disjoint and overlapping constraints.

# In a disjoint design constraint, an entity can be at most one of the subclasses of the specialization. i.e. Employee is disjoint to secretary, technician, engineer etc which means that an employee can be anyone of the above mentioned designation On the other hand, overlapping gen are revert in the sense that the super class can belong to more than one subclass. for ex: person as a super class and passport, driving license as its overlapping subclass

# 28. Explain the distinction between total and partial constraints.

# Total constraints:

# Total constraints are those in which a table's existence requires the existence of an associated table in a particular defined relation between them.

# In a total design constraint, each higher-level entity must belong to a lower-level entity set. The same need not be true in a partial design constraint.

# For instance, some employees may belong to no work-team.

**Partial constraints:**

 Partial constraints are involved with the tables in which presence of one table is partial for the associated table.

# 29. Write short notes on:

# Specialization

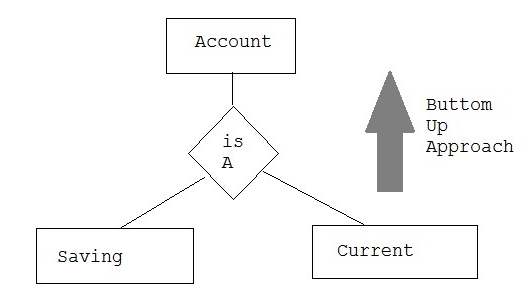
# Generalization

# Aggregation

# Generalization:

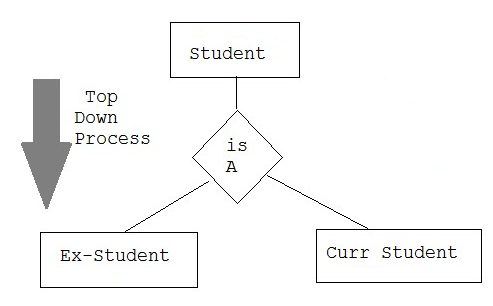
Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.

The common characteristics can be attributes or methods. Generalization is represented by a triangle followed by a line.



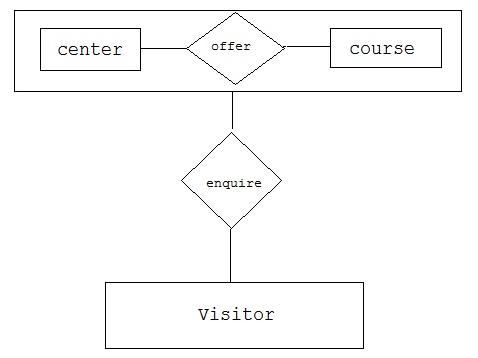
**Specialization:**

Specialization is the reverse process of Generalization means creating new sub classes from an existing class. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.



# Aggregation:

Aggregation is a process when relation between two entities is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.

****