**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Lab Assignment #5**

**Submitted by:**

BIRAJ POUDYAL

013BSCCSIT015

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav**  Lecturer  St. Xavier’s College |  |

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

**Entity Relationship model**

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

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.**

Entity refers to individual things, including people, concepts or objects with data that is first stored in a database management system (DBMS) and has attributes and relationships to other entities.

An entity is something of importance to the user. Something the user wants to track. VEHICLE.

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

An entity class is a group of entities of the same type, i.e. VEHICLE. An entity instance is a particular entity, i.e. VEHICLE 4072.

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. This simple table will be used to explain how each attribute can be a different type of attribute:

Student (*stu\_LastName, stu\_MiddleName, stu\_FirstName, stu\_Age, stu\_Phone, stu\_Email*).

## 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.

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\_LastName* since a student usually has one last name that uniquely identifies him/her.

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, in the context of databases, 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.**

A relationship class is an association among entity classes; a relationship instance is an association among entity instances.

1. **Define degree of relationship.**

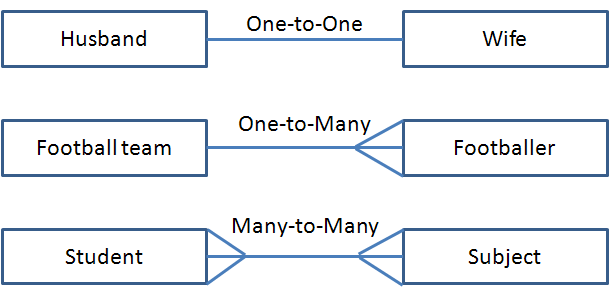
Degree is the number of entities that participate in a relationship. The relationship ASSIGNMENT associates a CLIENT with an ATTORNEY with a TASK.

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

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

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

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



**10. Define the terms maximum cardinality and minimum cardinality.**

### Maximum 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. You would denote this by having a perpendicular dash next to each entity in the 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.

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. In the second solution in the previous page, you see that Employees can take many Courses but a Course also contains many Employees. This relationship also had an attribute Attendance, M:N relationships may have additional information that is stored as part of the instance of the relationship rather than with either entity. If you imagine an order and an item, the order contains many items and an item can be sold on many different orders. For and instance of Order-Item, there can be a price or quantity that corresponds to a specific instance of the relationship so quantity and price belong to the relationship rather than either item (see the equivalent pictures below). To avoid that in the first solution, an additional table was added. I will accept either approach a M:N with attributes or an additional entity (called an associative entity) unless otherwise noted.

### 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.

  Strong M:N, optional-to-optional Weak1:M, mandatory-to-optional Strong 1:1, mandatory-to-manadatory (a very unlikely combination! :)

**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.  
  
**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.  
**Primary Key**  
It is a candidate key that is chosen by the database designer to identify entities with in an entity set. Primary key is the minimal super keys. In the ER diagram primary key is represented by underlining the primary key attribute. 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.**

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?**

Composite key consists of more than one attributes.  
  
Example: Consider a Relation or Table R1. Let A,B,C,D,E are the attributes of this relation.  
  
R(A,B,C,D,E)  
A→BCDE This means the attribute 'A' uniquely determines the other attributes B,C,D,E.  
BC→ADE This means the attributes 'BC' jointly determines all the other attributes A,D,E in the relation.  
  
Primary Key :A  
Candidate Keys :A, BC  
Super Keys : A,BC,ABC,AD  
  
ABC,AD are not Candidate Keys since both are not minimal super keys.

**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**

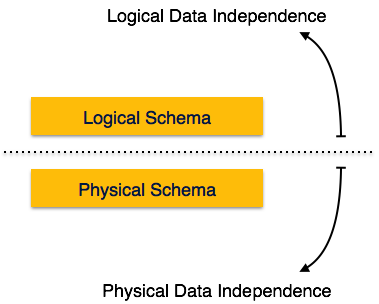
**each indicated in an E-R diagram?**

**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?**

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

A database system normally contains a lot of data in addition to users’ data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation. Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data. For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

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

**20. What do you mean by mapping cardinalities ? explain various type of cardinalities.**

**21. What is the difference between single-value and multivalued 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:**

**· Specialization**

**· Generalization**

**1 · Aggregation**