**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System Theory Assignment #5**

**Submitted by:**

Sazjan Neupane

013BSCCSIT036

**Submitted to:**

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

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

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

**Entity- Relationship Diagram**

An entity–relationship model (ER model) is a [data model](https://en.wikipedia.org/wiki/Data_modeling) for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a [database](https://en.wikipedia.org/wiki/Database) such as a [relational database](https://en.wikipedia.org/wiki/Relational_database). The main components of ER models are [entities](https://en.wikipedia.org/wiki/Entities) (things) and the relationships that can exist among them.

Entity–relationship modeling was developed by [Peter Chen](https://en.wikipedia.org/wiki/Peter_Chen) and published in a 1976 paper. However, variants of the idea existed previously, and have been devised subsequently such as super type and subtype data entities and commonality relationships.

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a [flowchart](http://www.smartdraw.com/flowchart/). It is the specialized symbols, and the meanings of those symbols, that make it unique.

An entity relationship diagram is a means of visualizing how the information a system produces is related.

**2. Define entity and give an example.**

Entities are objects or concepts that can have data stored about them, the relationship between those entities, and the [cardinality](http://whatis.techtarget.com/definition/cardinality), which defines that relationship in terms of numbers. An entity is something that exists in itself, actually or potentially, concretely or abstractly, physically or not.

**Entities** are represented by rectangles. An entity is an object or concept about which you want to store information. A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



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

**Entity Class and 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 12345.

**4. Define attribute and its types.**

Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.   


A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.



**5. What is derived attributes?**

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary. If an attribute's value can be determined from the values of other attributes, then the attribute is derivable, and is said to be a derived attribute.

If an attribute's value can be derived from other attributes, you should consider dropping the attribute from the model. Perhaps you would keep it, if keeping it helped understandability (you can still decide to drop the attribute from the physical model). Derived attributes are shown with a dotted lined oval, see the figure below.



**6. 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. It is an association among entities.

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.  
  
The fundamental feature that differentiates relational databases from other database types (e.g., flat-files) is the ability to define relationships.

**7. Explain the difference between a relationship class and a relationship instance.**

**8. Define degree of relationship.**

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. The degree of relationship (also known as cardinality) is the number of occurrences in one entity which are associated (or linked) to the number of occurrences in another. 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.

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

There are three degrees of relationship, known as:

one-to-one (1:1)

one-to-many (1:M)

many-to-many (M:N)

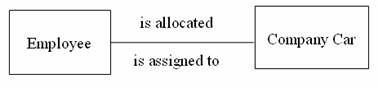
**One-to-one (1:1)**

This is where one occurrence of an entity relates to only one occurrence in another entity.

A one-to-one relationship rarely exists in practice, but it can. However, you may consider combining them into one entity.

For example, an employee is allocated a company car, which can only be driven by that employee.

Therefore, there is a one-to-one relationship between employee and company car.

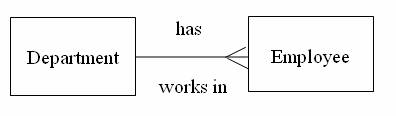


**One-to-Many (1:M)**

Is where one occurrence in an entity relates to many occurrences in another entity.

For example, taking the employee and department entities shown on the previous page, an employee works in one department but a department has many employees.

Therefore, there is a one-to-many relationship between department and employee.



**Many-to-Many (M:N)**

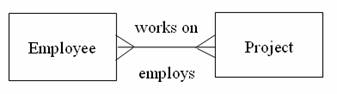
This is where many occurrences in an entity relate to many occurrences in another entity.

The normalisation process discussed earlier would prevent any such relationships but the definition is included here for completeness.

As with one-to-one relationships, many-to-many relationships rarely exist. Normally they occur because an entity has been missed.

For example, an employee may work on several projects at the same time and a project has a team of many employees.

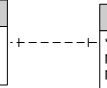
Therefore, there is a many-to-many relationship between employee and project.



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

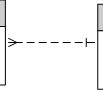
**Maximum Cardinality**

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.



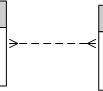
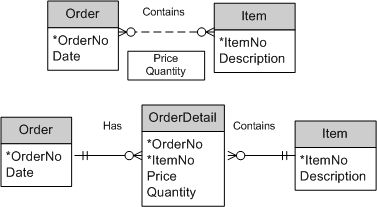
1:1 Relationship

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. 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.

M:N Relationship Equivalent conceptual designs, the composite identifier in OrderDetail uniquely identifies an items

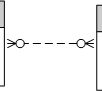
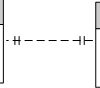
quatity and price for a specific order

 Note: A many can also be labeled as finite. Imagine a classroom with 36 seats. See below the example below.

finiteM.jpg

**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.

**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 Keys**

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.

**Composite Key**

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.

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

**Entity- Relationship Building Blocks**

At first glance an entity relationship diagram looks very much like a [flowchart](http://www.smartdraw.com/flowchart/). It is the specialized symbols, and the meanings of those symbols, that make it unique.

An entity relationship diagram is a means of visualizing how the information a system produces is related.

Entity

Entities are objects or concepts that can have data stored about them, the relationship between those entities, and the [cardinality](http://whatis.techtarget.com/definition/cardinality), which defines that relationship in terms of numbers. An entity is something that exists in itself, actually or potentially, concretely or abstractly, physically or not.

**Entities** are represented by rectangles. An entity is an object or concept about which you want to store information. A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



Entity Class and 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 12345.

Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.   


A multi valued attribute can have more than one value. For example, an employee entity can have multiple skill values.



A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



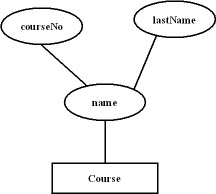
**13. What is composite attributes, when it is used?**

**Composite Attribute**

An attribute is considered composite if it comprises two or more other attributes.

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.**

**Single Valued and Simple Attributes**

An attribute is considered single-valued if there is at most one value associated with it at any one point in time.

For example, suppose "gender" is an attribute in our design. For most applications we would say that gender is single-valued; at any given point in time, there is just one value (male, female) recorded for gender for a person.

We characterize an attribute as being single-valued if, for any instance of the pertinent entity set, there is only one value at a given time for the attribute.

Single-valued attributes are shown with a simple oval, one with a single line for a border (as opposed to a double-lined border). In all of our examples so far, we have assumed that each attribute was single-valued.

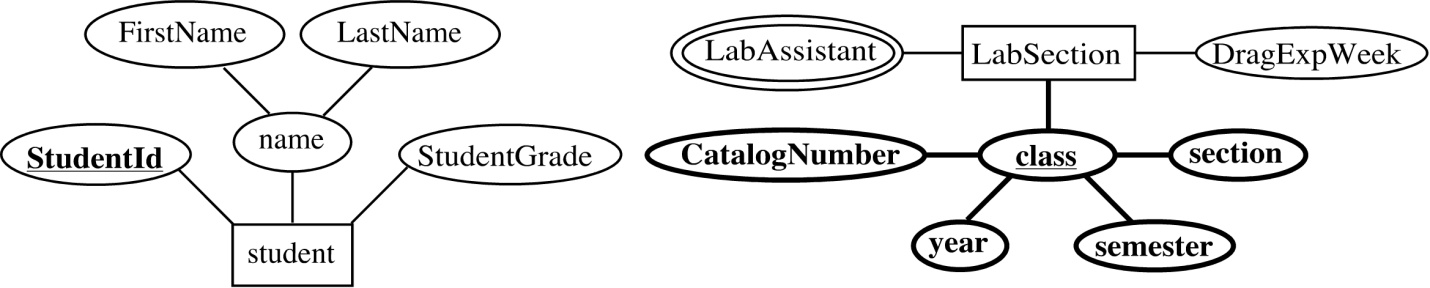
A single-valued attribute is one that can have only one value. For example, a person has only one first name and only one social security number. A simple attribute is one that cannot be decomposed into its component pieces. For example, a person's sex is classified as either M or F and there is no reasonable 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. For instance, the product serial number TNP5S2M231109154321 might be decomposed this way:  
  
TN = state = Tennessee  
P5 = plant number 5   
S2 = shift 2  
M23 = machine 23  
11 = month, i.e., November  
09 = day  
154321 = time on a 24-hour clock, i.e., 15:43:21, or 3:43 p.m. plus 21 seconds

**15. Discuss the difference between a composite key and a composite attribute. How would**

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

**Composite Key and Attribute**

Attribute is the property of entity.The composite attribute is like address(where street no,house no,town name all include).Composite key is also an attribute,but only which attribute are work as a unique identifier. Example:> In an ERD if vendor placed with orders then order(order day, order number) vendor(vendor code,vendor address) Here, order and vendor both are entity and order number, vendor code both are Composite key(because those are unique)but vendor address is a Composite attribute and order day(may be not unique)is an attribute only. So, we can conclude that all attribute not Composite key.

ER diagram notation for key *StudentId* for *student* and composite key (*class*, consistent of (*CatalogNumber, section, year,*and*semester*) for *LabSection*

**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?**

**ER Model for Conceptual Design**

In 1976, Chen developed the Entity-Relationship (ER) model, a high-level data model that is useful in developing a conceptual design for a database. Creation of an ER diagram, which is one of the first steps in designing a database, helps the designer(s) to understand and to specify the desired components of the database and the relationships among those components. An ER model is a diagram containing entities or "items", relationships among them, and attributes of the entities and the relationships.

|  |  |
| --- | --- |
| Definition | The Entity-Relationship (ER) model, a high-level data model that is useful in developing a conceptual design for a database. |

To make the description of the model more complete, we consider the example of a physics department at a college that maintains a database of experimental results. Throughout a laboratory, students collaborate and share their results and access data sets from other semesters on a computer system. For example, in the laboratory session on "Freely Falling Objects with Significant Drag," students determine the drag coefficient by dropping dust balls from different heights and measuring the times they take to fall. Each team enters its results into the distributed database, and the class analyzes the data. After a team enters data into the web-accessed database, all students can obtain the measurements simultaneously. To simplify the analysis, we assume that the database only stores results related to this experiment over a period of several years. 

Entity

An entity is a real-world item or concept that exists on its own. In our example, a particular student (such as, "Emanuel Vagas"), team, lab section, or experiment is an entity. The set of all possible values for an entity, such as all possible students, is the entity type. In an ER model, we diagram an entity type as a rectangle containing the type name, such as *student* (see [Figure 2](http://wofford-ecs.org/dataandvisualization/ermodel/material.htm#Figure 2)).

|  |  |
| --- | --- |
| Definition | An entity is a real-world item or concept that exists on its own. The set of all possible values for an entity is the entity type. |

Figure 2.  ER diagram notation for entity *student*



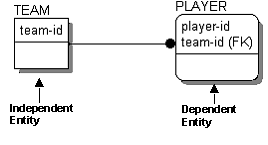
Attribute

Each entity has attributes, or particular properties that describe the entity. For example, student Emanuel Vagas has properties of his own Student Identification number, name, and grade. A particular value of an attribute, such as 93 for the grade, is a value of the attribute. Most of the data in a database consists of values of attributes. The set of all possible values of an attribute, such as integers from 0 to 100 for a grade, is the attribute domain. In an ER model, an attribute name appears in an oval that has a line to the corresponding entity.

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

**Dependent Entities**

In relational terms, a child entity that depends on the foreign key attribute for uniqueness is called a dependent entity. In IDEF1X notation, dependent entities are represented as round-cornered boxes.



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.

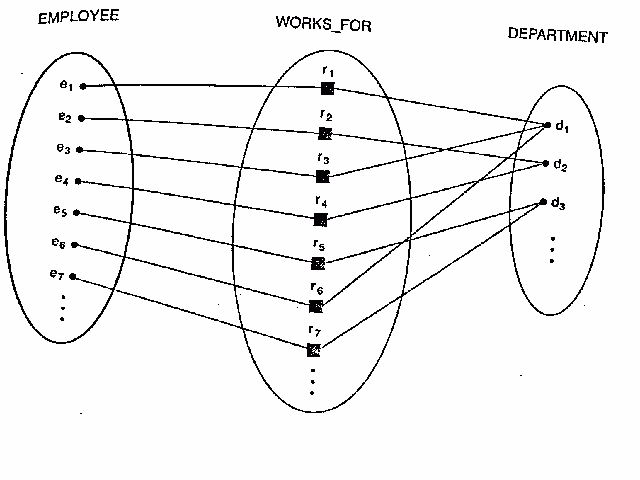
**19. What is the difference total and partial participation? Explain.**

CONSTRAINTS: We use CARDINALITY RATIO to express a constraint on a relationship type, such as  
1:1  
1:N  
M:N 

 PARTICIPATION CONSTRAINTS: partial or total PARTICIPATION CONSTRAINT   
An employee MUST work for a department   
An employee entity can exist only if it participates in a WORKS\_FOR relationship instance   
Thus its participation is TOTAL   
  
  
Only some employees manage departments   
The participation is PARTIAL   
  
  
A formal constraint: (m, n) where   
m, n are number of times

An employee MUST work for a department   
An employee entity can exist only if it participates in a WORKS\_FOR relationship instance   
Thus its participation is TOTAL 

Only some employees manage departments   
The participation is PARTIAL   
  
A formal constraint: (min,max) where m, n are min and max number of times an entity participates in a relationship instance. For example, (0,10) means partial participation, and (1,max) means total participation.

Some instances of the WORK\_FOR relationship:   
  
  
The ternary relationship SUPPLY:   
