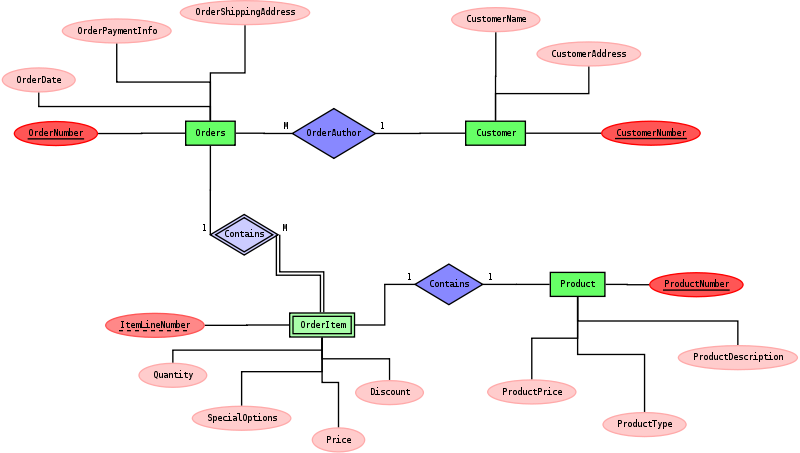
1. **What do you mean by Entity- Relationship Diagram? Explain**

An entity-relationship diagram, or ERD, is a chart that visually represents the relationship between database entities. ERDs model an organization’s data storage requirements with three main components: entities, attributes, and relationships. An entity-relationship (ER) diagram is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored. there are 3 ingredients in a standard entity-relationship diagram:

* **Entities**, which represent people, places, items, events, or concepts.
* **Attributes**, which represent properties or descriptive qualities of an entity. These are also known as data elements.
* **Relationships**, which represent the link between different entities.

Entities, attributes, and relationships can be represented in one of three ways: with a conceptual model, logical model, or physical model. These models increase in complexity as you move from conceptual to logical to physical. It's usually best to start with a conceptual ERD model, so you can understand—at the highest level—the entities in your data and how they relate to each other. As you transform a conceptual ERD to a physical model, you'll learn exactly how to implement modeled information into the database of your choice.



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

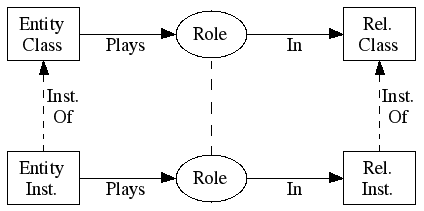
An entity is something that exists in itself, actually or potentially, concretely or abstractly, physically or not. It need not be of material existence. In particular, abstractions and legal fictions are usually regarded as entities. Entities are objects or concepts that represent important data. They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

1. 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).



1. **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.

1. **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.

1. **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

1. **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.

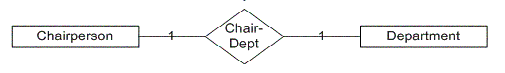
1. **Define degree of relationship.**

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.

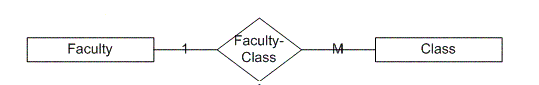
1. **List and give an example of the three types of binary relationships. Draw an E-R diagram for each.**

The three different types of binary relationships are:

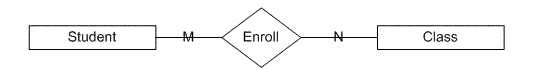
1. 1:1 - a single entity instance of one type is related to a single-entity instance of another type.



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.



1. **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.
* **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.

1. **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.

1. **What are the main building modules of the entity relationship model? Discuss each one.**
2. **What is composite attributes, when it is used?**
3. **Explain the difference between single-value attributes and simple attributes.**
4. **Discuss the difference between a composite key and a composite attribute. How would each indicated in an E-R diagram?**
5. **What two courses of action are available to a designer when a multivalued attribute is encountered ?**
6. **Explain the various terms of an E-R model and how are they represented in an E-R model?**
7. **Explain the concept of dependent entities? Give example.**
8. **What is the difference total and partial participation? Explain.**
9. **What do you mean by mapping cardinalities ? explain various type of cardinalities.**
10. **What is the difference between single-value and multivalued attributes? Explain**
11. **Explain the concept of participation constraints.**
12. **Difference the binary relationship with ternary relationship with example.**
13. **Explain the difference between weak and strong entity set.**
14. **Define the components of extended E-R features.**
15. **Define the concept of aggregation. Give two examples of where this concept is useful.**
16. **Explain the distinction between disjoint and overlapping constraints.**
17. **Explain the distinction between total and partial constraints.**
18. **Write short notes on:**

* **Specialization**
* **Generalization**
* **Aggregation**