**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**LAB ASSIGNMENT#5**

**Submitted by:**

Shreesha Pokharel

013BSCCSIT038

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav** |  |

Lecturer

Department of Computer Science

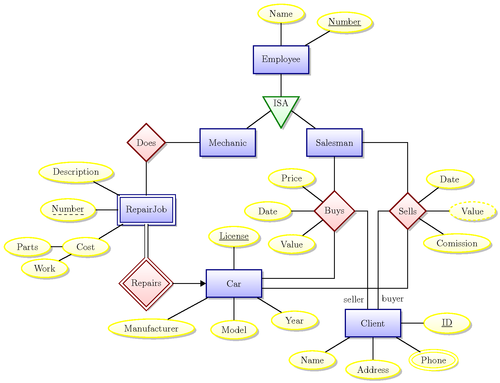
Date of submission: 27th August, 2015

1. 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.

While useful for organizing [data](http://searchdatamanagement.techtarget.com/definition/data) that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or [unstructured data](http://searchbusinessanalytics.techtarget.com/definition/unstructured-data), and an ERD is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Three main components of an ERD are the [entities](http://whatis.techtarget.com/definition/entity), which 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.



1. Define entity and give an example.

An entity is a singular, identifiable object. An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values.

For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

1. Difference between entity class and an entity instance.
2. Define attribute and its types.

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

* **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
* **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
* **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.
* **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.
* **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

1. What is derived attribute?

Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.

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

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

The number of entities involved in the relationship is referred to as degree of relationship. It is usually 2 (binary relationship) however Unary and higher degree relationships can exist.

1. List and give examples of three types of binary relationships. Draw an E-R diagram for each.
2. 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, you should think about what is actually taking place. 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.

**Primary key:** A **primary key**, also called a **primary** keyword, is a **key** in a relational **database** that is unique for each record. It is a unique identifier, such as a driver license number, telephone number (including area code), or vehicle identification number (VIN). A relational **database** must always have one and only one **primary key**.

**Candidate key:** A **Candidate Key** can be any column or a combination of columns that can qualify as unique **key in database**. There can be multiple **Candidate Keys** in one table. Each **Candidate Key** can qualify as Primary **Key**.

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

1. What are the main building modules of the entity relationship model? Discuss each one.
2. What is composite attributes, when it is used?

A **composite attribute** has multiple components, each of which is atomic or **composite**.