

(A Constituent College of Somaiya VidyaviharUniversity)

# Batch: B-1 Roll No.: 16010422234 Name: Chandana Ramesh Galgali

# Experiment No.: 02

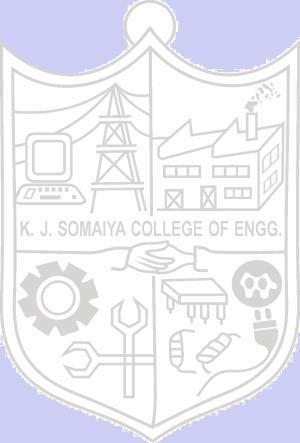
**Aim:** To Map EER diagram drawn in experiment no.1 to relational model.



**Resources needed:** MS-office



**Theory:**

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns and each column has a unique name. The relational model is an example of a record-based model. Each table contains records of a particular type. The columns of the table correspond to the attributes of the record type. The relational model is the most widely used data model.

**Procedure / Approach /Algorithm / Activity Diagram:**

**Steps for Reducing EER model into relational model**

1. Any strong entity set E having attributes a1, a2,…,an is reduced into a relation schema called E with n distinct attributes i.e. a separate relation with name E and n distinct columns.
2. Any weak entity set A having attributes a1, a2,..n and a strong entity set B on which A depends, having primary key attributes as b1, b2, …, bn is reduced into a relation schema called A with one attribute for each member of set

{ a1, a2,…, an} U {b1, b2, ……. , bn}

1. Any relationship set R having a1,a2,…,an as a set of attributes formed by union of the primary keys of each of the entity sets participating in R and b1, b2,….,bn as set of descriptive attributes is reduced into a relation schema called R with one attribute for each member of the set {a1, a2, …. ,an} U {b1, b2, …., bn}

# Primary key of relationship set is decided as follows

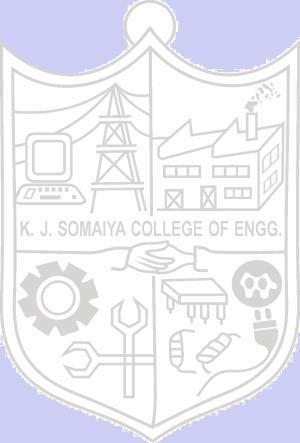
For **binary many to many relationships** the union of primary key attributes from the participating entity sets is primary key.

For **binary one to one relationship set** the primary key of either of the participating entity sets can be chosen as the primary key.

For **binary many to one or one to many relationship set** the primary key of the entity set on the many side of the relationship set serves as the primary key.

For **n-ary relationship sets without any arrows on its edges**, union of the primary key attributes of participating entity sets is a primary key.

For **n-ary relationship sets with an arrows on one of its edges**, union of the primary key attributes of participating entity sets is a primary key.

To remove redundancy we generally make separate relation schema for many to many relationship set with primary key and other attributes as mentioned above.

For one to one we combine relation schema of relationship set with relation schema of either side of entity sets relation schema.

For one to many and many to one we combine relation schema of relationship set with relation schema of entity set on many side entity set.

We don’t make separate relation schema for identifying relationship set. Every composite attribute A having subparts a1, a2,…,an is represented by a separate column for each subpart in relation schema of the associated entity set.

For **multivalued attribute** separate schema is form having columns as attributes of primary key of associated entity set and a column for multivalued attribute

For **overlapping generalization/specialization** create separate relation schemas for higher level as well as lower level entity sets.

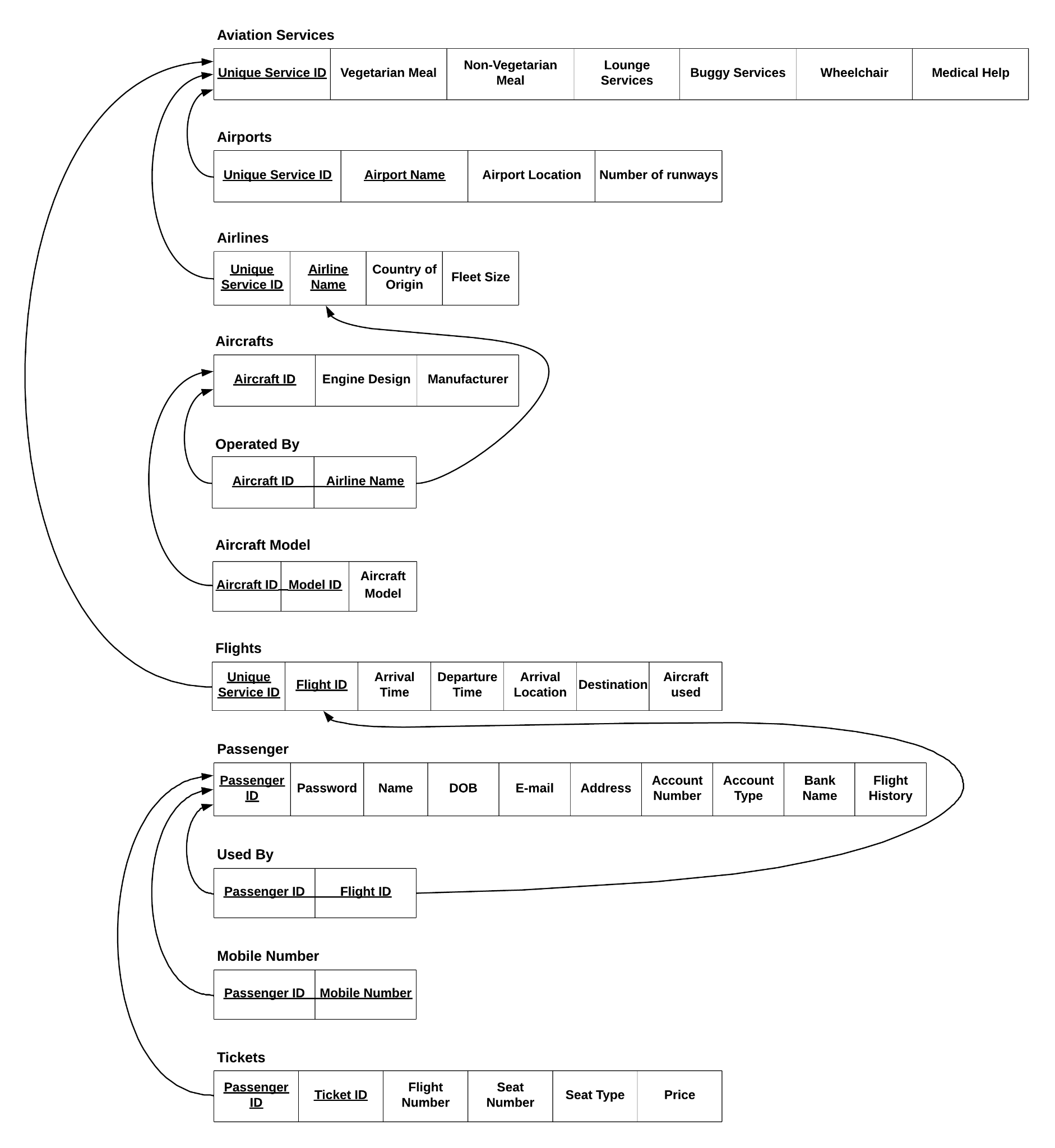
Also include the foreign key constraint in lower level entityset for the primary key attributes of higher level entity set.

For **disjoint generalization/specialization, create** separate relation schemas only for every lower level entity set(higher level entity set’s attributes are inherited so add columns for the same) and not for higher level entity set.

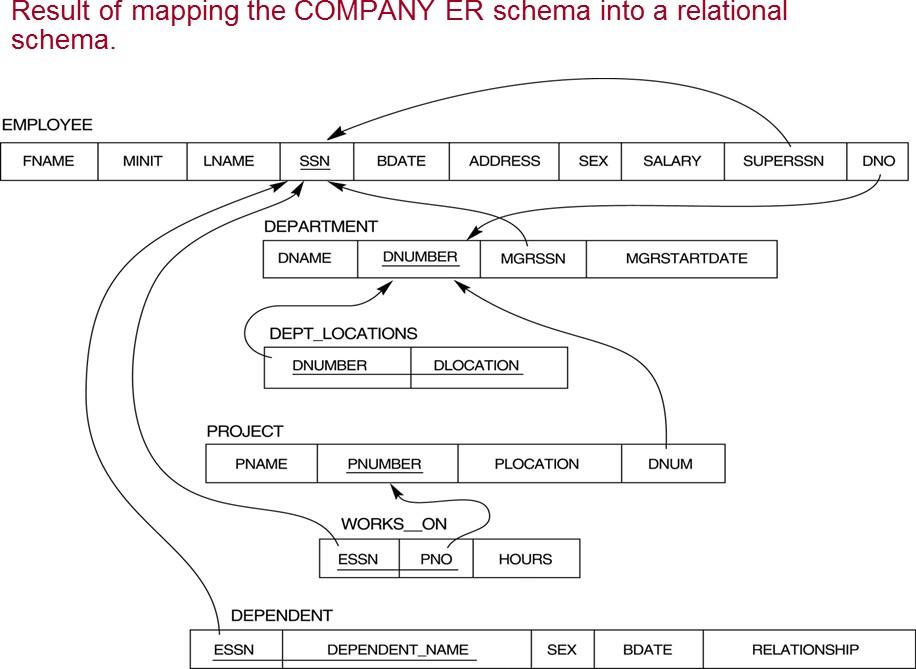
No separate relation is required to represent the **aggregation** the relation created from the defining relationship is used instead (design schema for relationship set treated as entity set carefully)



# Results: Relational model



# Example:





**Outcomes:**

# Apply data models to real world scenarios.

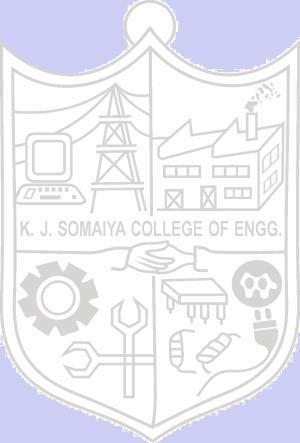


**Questions:**

**Q1 Explain generalization and specialization with examples.**

**Ans:**  Generalization is the process of combining similar entities into a more general entity. It allows for the creation of a higher-level entity that encompasses common characteristics of multiple lower-level entities. This higher-level entity is called a superclass or a generalization. Generalization is used to abstract common attributes and relationships from multiple entities, reducing redundancy and improving data organization.

For *example*, consider a database for a university. The entities "Student," "Faculty," and "Staff" may have common attributes such as "Name," "Address," and "Phone Number." Instead of duplicating these attributes in each entity, we can create a superclass called "Person" and generalize the common attributes into it. The "Student," "Faculty," and "Staff" entities then become subclasses of the "Person" superclass.

Specialization, on the other hand, is the process of creating new entities from an existing entity by adding specific attributes or relationships. It allows for the creation of lower-level entities that inherit attributes and relationships from a higher-level entity. This lower-level entity is called a subclass or a specialization. Specialization is used to represent entities that have unique characteristics or relationships.

Continuing with the university *example*, we can specialize the "Person" superclass into subclasses such as "Undergraduate Student," "Graduate Student," "Professor," and "Administrative Staff." Each subclass will have its own specific attributes and relationships in addition to the attributes inherited from the "Person" superclass.

In summary, generalization and specialization are techniques used in DBMS to organize data models hierarchically. Generalization combines similar entities into a more general entity, while specialization creates new entities with specific attributes and relationships. These concepts help in reducing redundancy, improving data organization, and representing complex relationships within a database.

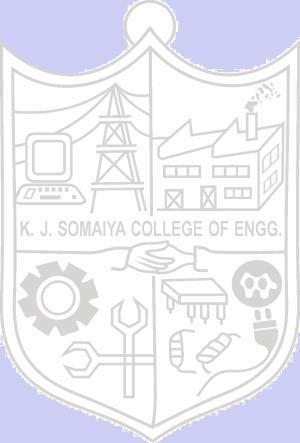
**Q2 What is physical and logical data independence in DBMS.**

**Ans:** Physical Data Independence: Physical data independence refers to the ability to modify the physical storage structures or devices used to store data without affecting the conceptual or logical schema of the database. It allows for changes in the physical organization of data, such as adding or removing storage devices, changing file structures, or reorganizing data storage, without requiring modifications to the application programs or queries that access the data.

For *example*, let's say a DBMS initially stores data in a flat file format on a hard disk. If the system later decides to switch to a relational database management system (RDBMS) and store the data in tables, the application programs that interact with the database should not be affected. The physical data independence ensures that the changes in the storage structure do not require modifications to the application programs.

Logical Data Independence: Logical data independence refers to the ability to modify the logical schema or structure of the database without affecting the external schema or the application programs that use the database. It allows for changes in the organization of data, such as adding or modifying tables, changing relationships between tables, or altering attributes, without requiring modifications to the application programs or queries.

For *example*, let's consider a database that stores information about employees, including their names, addresses, and salaries. If the database needs to be modified to include additional information, such as employee performance ratings, the logical data independence ensures that the application programs accessing the existing employee information do not need to be modified. The changes can be made to the logical schema without affecting the external view or the programs that rely on it.

In summary, physical data independence allows for changes in the physical storage structures without affecting the application programs, while logical data independence allows for changes in the logical schema without affecting the external view or the programs that use the database. These forms of data independence provide flexibility and ease of maintenance in a DBMS, allowing for modifications to the system without disrupting the applications that rely on it.



**Conclusion:** The experiment demonstrated the process of mapping an EER diagram to a relational model, highlighting the importance of understanding the conceptual and logical aspects of database design. The successful mapping of the EER diagram to the relational model provides a foundation for building a robust and efficient database system.

**Reference books:**

1. Elmasri and Navathe, “Fundamentals of Database Systems”, 6th Edition, Pearson Education
2. Korth, Slberchatz,Sudarshan, :”Database System Concepts”, 6th Edition, McGraw – Hill