DATA MODELS

Introduction to Data model:

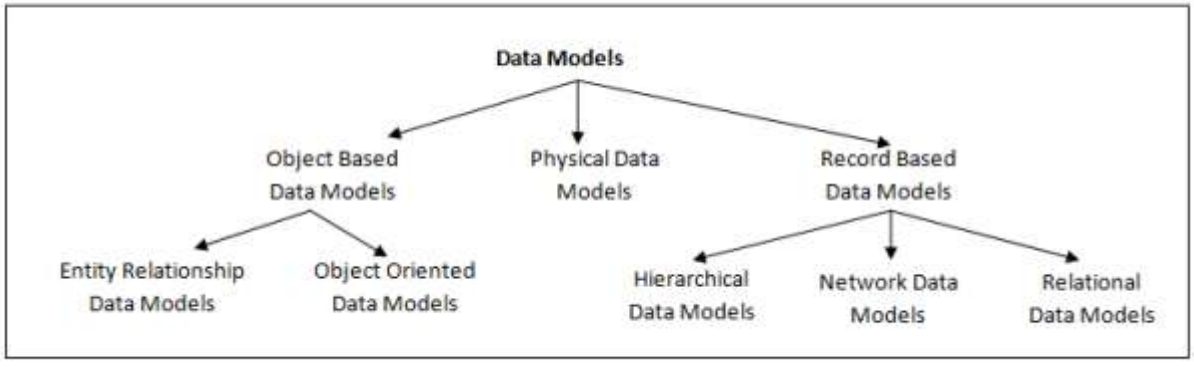
When we construct a building, we first plan what do need in the building, and then we put it in the form design on the paper. Once rough copy of design is done, we review it and modify wherever is necessary. Once the design is complete, we gather all the materials to construct the building, then we place, roof, wall, rooms, doors, windows etc at the right place as per the design. Once the building is complete, we design the interior for each room. Once interior design is complete, we start building it on each room. Once it is complete, we plan furniture etc and the process goes on. One complete furnished building takes different steps of well designing and effort of giving them life.

Similarly, when we say database for particular requirement is needed, then we look into what are components of requirement. Then we plan how it can be structured in the database. Planning the structure of database is called data models. It involves planning about tables, their columns, mapping between the tables, how they are structured in the physical memory etc. A data model helps to put the real world requirement into a design. This makes the developer to understand the relationship between various objects in the database. It helps to highlight any drawbacks of the plan and correct it at the design stage itself.

Depending on the levels of data we are modeling, we have divided data models into 3 categories – Object Based, Physical and Record based Data models.

Physical data model represent the model where it describes how data are stored in computer memory, how they are scattered and ordered in the memory, and how they would be retrieved from memory. Basically physical data model represents the data at data layer or internal layer.

Object and Record based data models are modeled based on the data at the application and user level. They are basically responsible for designing various objects of the database, and their mappings. They are further divided into different categories as shown in below diagram.

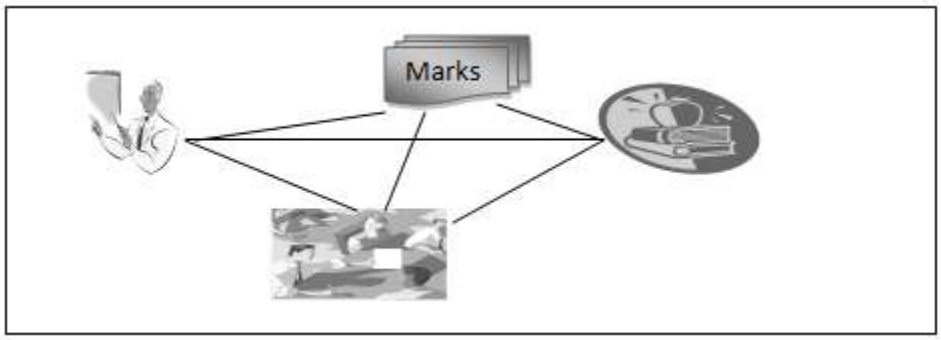


Object based Data Model:

Imagine we have to design database for college. What is the real world entities involved with college? They are college, Students, Lecturer, Course, Subject, Marks etc. Once all the entities

are listed, we find out the relationship between them and try to map all of them. Also we list what are the attributes related to each entity like student id, name, lecturer name, course that he is teaching, different subjects, pass mark, grade levels etc. Here we are not bothered about what data value is stored, what is the size of each data etc. We know only entities involved, their attributes and mapping at this stage.

Object based Data Models are based on above concept. It is designed using the entities in the real world, attributes of each entity and their relationship. It picks up each thing/object in the real world which is involved in the requirement.



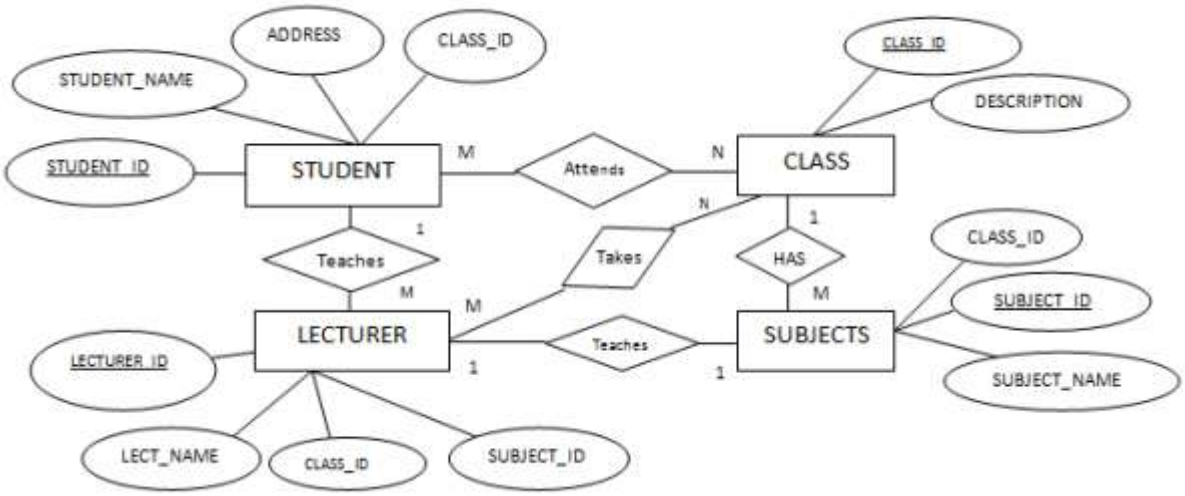
There are two types of object based data Models – Entity Relationship Model and Object oriented data model. ER data model is one of the important data model which forms the basis for the all the designs in the database world. It defines the mapping between the entities in the database. Object oriented data model, along with the mapping between the entities, describes the state of each entity and the tasks performed by them.

Entity Relationship Data Model:

Consider the example above. It maps entities like Student, Lecturer, Subjects, and Marks with each other to form the relation among them. It also list attributes of each objects. ER model represents the all these entities, attributes and their relationship in the form of picture to make the developer understand the system better. A simple ER diagram for above example can be drawn as below. Are you able to understand what are the entities involved, what are its attributes and their relations that we were discussing better here? Yes, it is clean and clear what a STUDENT database look like. It gives the clear understanding of how they are scattered and mapped. If we have missed any entities or attribute or the mapping, we can easily identify here. If we represent it in some tables, it would be difficult to identify this gap.

In the below diagram, Entities or real world objects are represented in a rectangular box. Their attributes are represented in ovals. Primary keys of entities are underlined. All the entities are mapped using diamonds. This is one of the methods of representing ER model. There are many different forms of representation. More details of this model are described in ER data model article.

DATA MODELS



Basically, ER model is a graphical representation of real world objects with their attributes and relationship. It makes the system easily understandable. This model is considered as a top down approach of designing a requirement.

Advantages

- It makes the requirement simple and easily understandable by representing simple diagrams.
- One can covert ER diagrams into record based data model easily.
- Easy to understand ER diagrams

Disadvantages

- No standard notations are available for ER diagram. There is great flexibility in the notation. It's all depends upon the designer, how he draws it.
- It is meant for high level designs. We cannot simplify for low level design like coding.

Object Oriented Data Model:

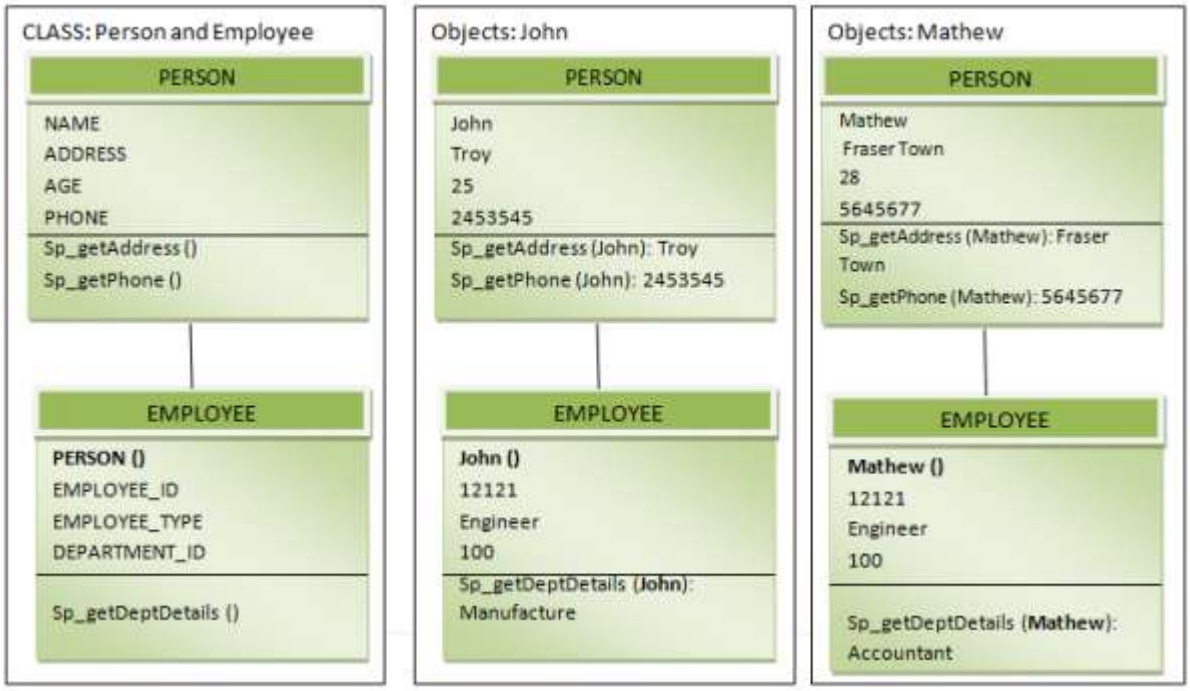
This data model is another method of representing real world objects. It considers each object in the world as objects and isolates it from each other. It groups its related functionalities together and allows inheriting its functionality to other related sub-groups.

Let us consider an Employee database to understand this model better. In this database we have different types of employees – Engineer, Accountant, Manager, Clark. But all these employees belong to Person group. Person can have different attributes like name, address, age and phone. What do we do if we want to get a person's address and phone number? We write two separate procedure sp_getAddress and sp_getPhone.

What about all the employees above? They too have all the attributes what a person has. In addition, they have their EMPLOYEE_ID, EMPLOYEE_TYPE and DEPARTMENT_ID attributes to identify them in the organization and their department. We have to retrieve their department details, and hence we sp_getDeptDetails procedure. Currently, say we need to have only these attributes and functionality.

Since all employees inherit the attributes and functionalities of Person, we can re-use those features in Employee. But do we do that? We group the features of person together into class. Hence a class has all the attributes and functionalities. For example, we would create a person class and it will have name, address, age and phone as its attribute, and sp_getAddress and sp_getPhone as procedures in it. The values for these attributes at any instance of time are object. i.e. ; {John, Troy, 25, 2453545 : sp_getAddress (John), sp_getPhone (John)} forms on person object. {Mathew, Fraser Town, 28, 5645677: sp_getAddress (Mathew), sp_getPhone (Mathew} forms another person object.

Now, we will create another class called Employee which will inherit all the functionalities of Person class. In addition it will have attributes EMPLOYEE_ID, EMPLOYEE_TYPE and DEPARTMENT_ID, and sp_getDeptDetails procedure. Different objects of Employee class are Engineer, Accountant, Manager and Clerk.



Here we can observe that the features of Person are available only if other class is inherited from it. It would be a black box to any other classes. This feature of this model is called encapsulation. It binds the features in one class and hides it from other classes. It is only visible to its objects and any inherited classes.

Advantages

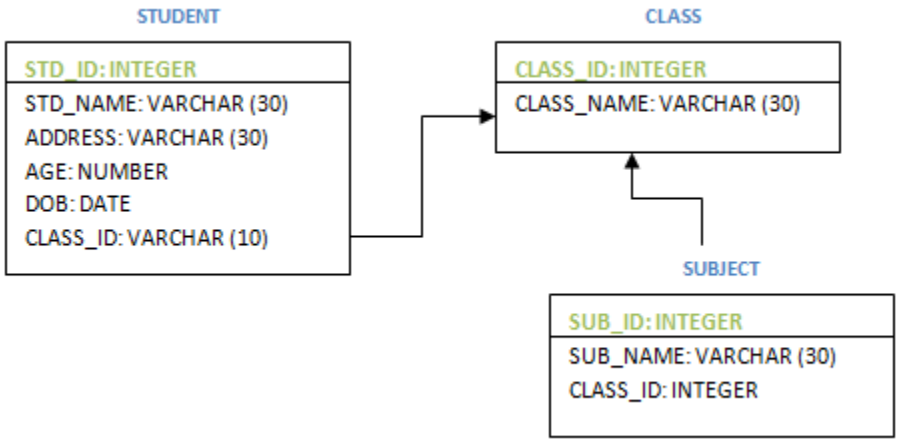
- Because of its inheritance property, we can re-use the attributes and functionalities. It reduces the cost of maintaining the same data multiple times. Also, these informations are encapsulated and, there is no fear being misused by other objects. If we need any new feature we can easily add new class inherited from parent class and adds new features. Hence it reduces the overhead and maintenance costs.
- Because of the above feature, it becomes more flexible in the case of any changes.
- Codes are re-used because of inheritance.
- Since each class binds its attributes and its functionality, it is same as representing the real world object. We can see each object as a real entity. Hence it is more understandable.

Disadvantages

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.
- It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.

Physical Data Model:

Physical data model represent the model where it describes how data are stored in computer memory, how they are scattered and ordered in the memory, and how they would be retrieved from memory. Basically physical data model represents the data at data layer or internal layer. It represents each table, their columns and specifications, constraints like primary key, foreign key etc. It basically represents how each tables are built and related to each other in DB.



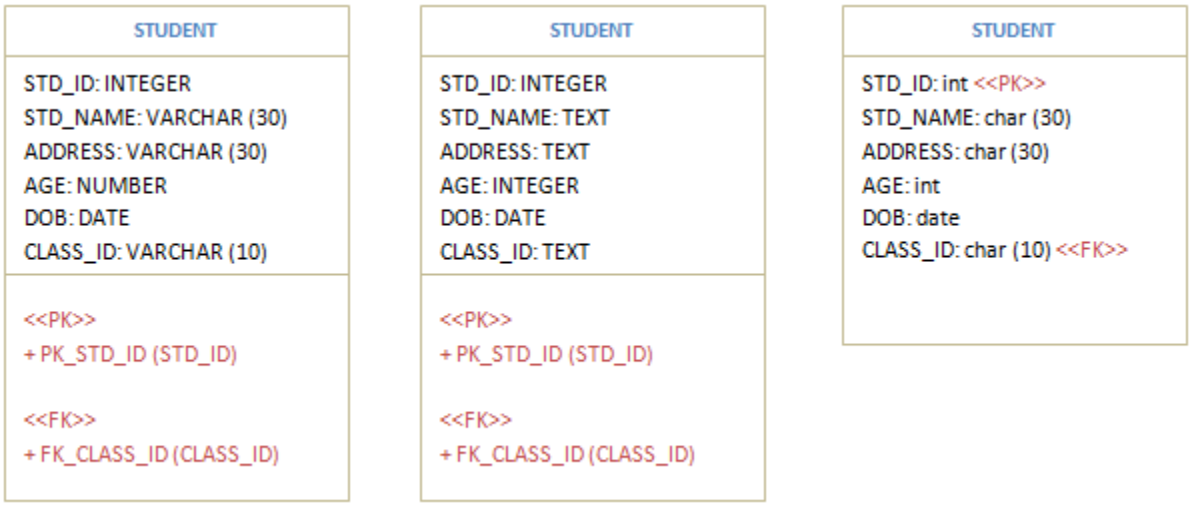
Above diagram shows how physical data model is designed. It is represented as UML diagram along with table and its columns. Primary key is represented at the top. The relationship between the tables is represented by interconnected arrows from table to table. Above STUDENT table is related to CLASS and SUBJECT is related to CLASS. The above diagram depicts CLASS as the parent table and it has 2 child tables – STUDENT and SUBJECT.

DATA MODELS

In short we can say a physical data model has

- Tables and its specifications – table names and their columns. Columns are represented along with their datatypes and size. In addition primary key of each table is shown at the top of the column list.
- Foreign keys are used to represent the relationship between the tables. Mapping between the tables are represented using arrows between them.
- Physical data model can have denormalized structure based on the user requirement. The tables might not be in normalized forms.

Physical data model is dependent on the RDBMS i.e.; it varies based on the RDBMS used. This means datatype notation varies depending on the RDBMS. For example, we have different datatypes in SQL server and oracle server. In addition, the representation of physical data model diagram may be different, though it contains same information as described above – some may represent primary key and foreign keys separately at the end of the column list. This data model depends on the user / designer how he specifies the diagram and the RDBMS servers. Below diagram shows different ways of representing a table.



Hence object based data model is based on the real requirement from the user, whereas record based data model is based on the actual relationships and data in DB. The Physical data model is based on the table structure in the DB.

Record based Data Models:

These data models are based on application and user levels of data. They are modeled considering the logical structure of the objects in the database. This data models defines the actual relationship between the data in the entities.

For example, employee and department entities are related to each other by means of department. This minute level of relationship is defined in the record based data models. We will not be able

get the mapping at this level in the object based data models. There it simply defines two entities are related. But the actual relationship between any two entities can be observed in record based data models.

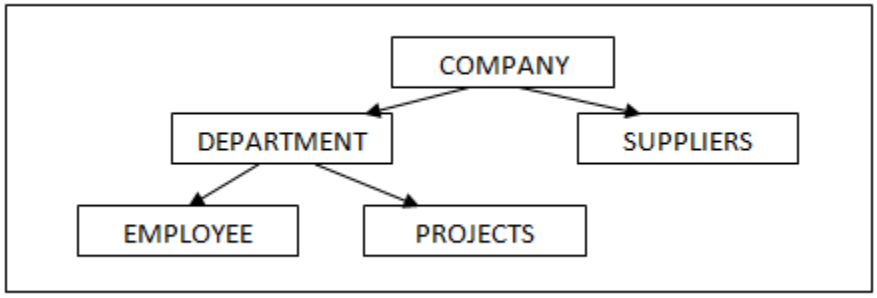
There are 3 types of record based data models defined so far- Hierarchical, Network and Relational data models. Most widely used record based data model is relational data model. Other two are not widely used. Let us understand how they are different from each other.

Hierarchical Data Models

Imagine we have to create a database for a company. What are the entities involved in it? Company, its department, its supplier, its employees, different projects of the company etc are the different entities we need to take care of. If we observe each of the entity they have parent – child relationship. We can design them like we do ancestral hierarchy. In our case, Company is the parent and rests of them are its children. Department has employees and project as its children and so on. This type of data modeling is called hierarchical data model.

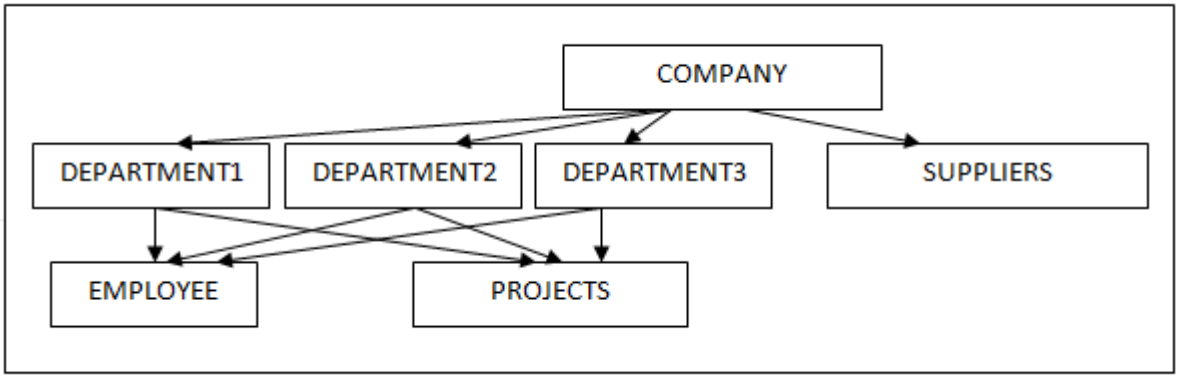
In this data model, the entities are represented in a hierarchical fashion. Here we identify a parent entity, and its child entity. Again we drill down to identify next level of child entity and so on. This model can be imagined as folders inside a folder!

In our example above, it is diagrammatically represented as below:



It can also be imagined as root like structure. This model will have only one main root. It then branches into sub-roots, each of which will branch again. This type of relationship is best defined for 1:N type of relationships. E.g.; One company has multiple departments (1:N), one company has multiple suppliers (1:N),one department has multiple employees (1:N), each department has multiple projects(1:N) . If we have M:N relationships, then we have to duplicate the entities and show it in the diagram. For example, if a project in the company involves multiple departments, then our hierarchical representation changes as below:

DATA MODELS



Advantages

It helps to address the issues of flat file data storage. In flat files, data will be scattered and there will not be any proper structuring of the data. This model groups the related data into tables and defines the relationship between the tables, which is not addressed in flat files.

Disadvantages

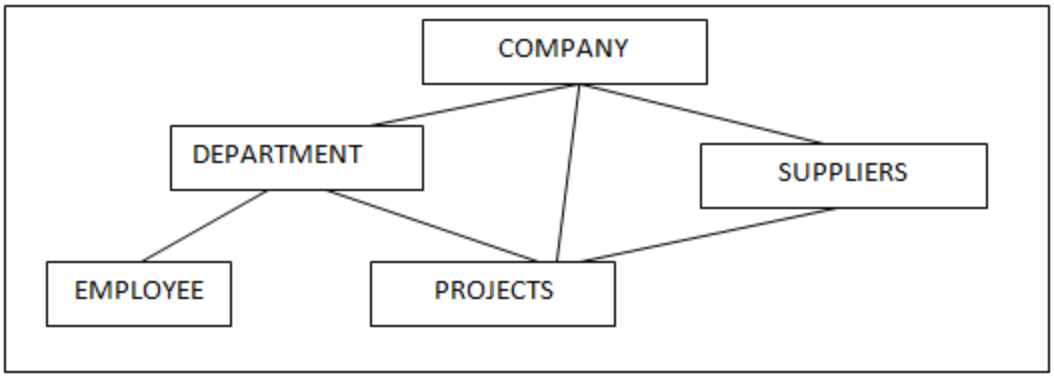
- Redundancy: - When data is stored in a flat file, there might be repetition of same data multiple times and any changes required for the data will need to change in all the places in the flat file. Missing to update at any one place will cause incorrect data. This kind redundancy is solved by hierarchical model to some extent. Since records are grouped under related table, it solves the flat file redundancy issue. But look at the many to many relationship examples given above. In such case, we have to store same project information for more than one department. This is duplication of data and hence a redundancy. So, this model does not reduce the redundancy issue to a significant level.
- As we have seen above, it fails to handle many to many relationships efficiently. It results in redundancy and confusion. It can handle only parent-child kind of relationship.
- If we need to fetch any data in this model, we have to start from the root of the model and traverse through its child till we get the result. In order to perform the traversing, either we should know well in advance the layout of model or we should be very good programmer. Hence fetching through this model becomes bit difficult.
- Imagine company has got some new project details, but it did not assign it to any department yet. In this case, we cannot store project information in the PROJECT table, till company assigns it to some department. That means, in order to enter any child information, its parent information should be already known / entered.

Network Data Models

This is the enhanced version of hierarchical data model. It is designed to address the drawbacks of the hierarchical model. It helps to address M:N relationship. This data model is also represented as hierarchical, but this model will not have single parent concept. Any child in the tree can have multiple parents here.

DATA MODELS

Let us revisit our company example. A company has different projects and departments in the company own those projects. Even suppliers of the company give input for the project. Here Project has multiple parents and each department and supplier have multiple projects. This is represented as shown below. Basically, it forms a network like structure between the entities, hence the name.



Advantages

- Accessing the records in the tables is easy since it addresses many to many relationships. Because of this kind of relationship, any records can be easily pulled by using any tables. For example, if we want to know the department of project X and if we know SUPPLIER table, we can pull this information. i.e.; SUPPLIER has the information about project X which includes the departments involved in the projects too. Hence makes the accessibility to any data easier, and even any complex data can be retrieved easily and quickly.
- Because of the same feature above, one can easily navigate among the tables and get any data.
- It is designed based on database standards – ANSI/SP ARC.

Disadvantages

- If there is any requirement for the changes to the entities, it requires entire changes to the database. There is no independence between any objects. Hence any changes to the any of the object will need changes to the whole model. Hence difficult to manage.
- It would be little difficult to design the relationship between the entities, since all the entities are related in some way. It requires thorough practice and knowledge about the designing.

Relational Data Models

This model is designed to overcome the drawbacks of hierarchical and network models. It is designed completely different from those two models. Those models define how they are structured in the database physically and how they are inter-related. But in the relational model,

we are least bothered about how they are structured. It purely based on how the records in each table are related. It purely isolates physical structure from the logical structure. Logical structure is defines records are grouped and distributed.

Let us try to understand it by an example. Let us consider department and employee from our previous examples above. In this model we look at employee with its data. When we say an employee what all comes into our mind? His employee id, name, address, age, salary, department that he is working etc. are attributes of employee. That means these details about the employee forms columns in employee table and value set of each employee for these attribute forms a row/record for an employee. Similarly, department has its id, name.

Now, in the employee table, we have column which uniquely identifies each employee – that is employee Id column. This column has unique value and we are able to differentiate each employee from each other by using this column. Such column is called as primary key of the table. Similarly department table has DEPT_ID as primary key. In the employee table, instead of storing whole information about his department, we have DEPT_ID from department table stored. i.e.; by using the data from the department table, we have established the relation between employee and department tables.

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Observe the table structures above. They are very simple to understand. There is no redundant data as well. It addressed major drawback of earlier data models. This type of data model is called relational data model.

This model is based on the mathematical concepts of set theory. It considers the tables as a two dimensional table with rows and columns. It is least bothered about the physical storage of structure and data in the memory. It considers only the data and how it can be represented in the form of rows and columns, and the way it can establish the relation between other tables.

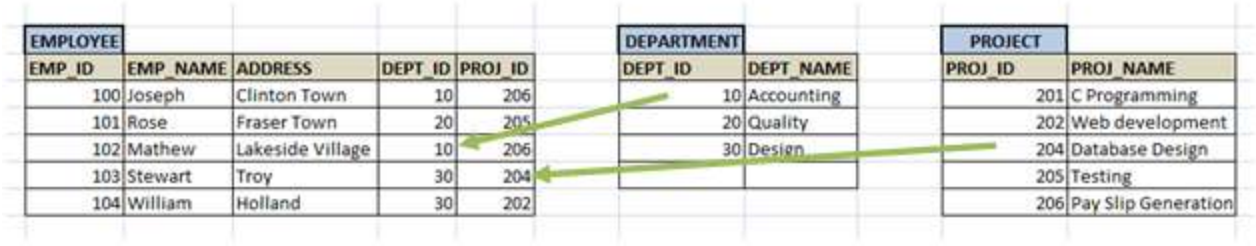
A relational data model revolves around 5 important rules.

1. Order of rows / records in the table is not important. For example, displaying the records for Joseph is independent of displaying the records for Rose or Mathew in Employee table. It does not change the meaning or level of them. Each record in the table is independent of other. Similarly, order of columns in the table is not important. That means, the value in each column for a record is independent of other. For example,

representing DEPT_ID at the end or at the beginning in the employee table does not have any affect.

2. Each record in the table is unique. That is there is no duplicate record exists in the table. This is achieved by the use of primary key or unique constraint.
3. Each column/attribute will have single value in a row. For example, in Department table, DEPT_NAME column cannot have 'Accounting' and 'Quality' together in a single cell. Both has to be in two different rows as shown above.
4. All attributes should be from same domain. That means each column should have meaningful value. For example, Age column cannot have dates in it. It should contain only valid numbers to represent individual's age. Similarly, name columns should have valid names, Date columns should have proper dates.
5. Table names in the database should be unique. In the database, same schema cannot contain two or more tables with same name. But two tables with different names can have same column names. But same column name is not allowed in the same table.

Examine below table structure for Employee, Department and Project and see if it satifies relational data model rules.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID	PROJ_ID
100	Joseph	Clinton Town	10	206
101	Rose	Fraser Town	20	205
102	Mathew	Lakeside Village	10	206
103	Stewart	Troy	30	204
104	William	Holland	30	202

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

PROJECT

PROJ_ID	PROJ_NAME
201	C Programming
202	Web development
204	Database Design
205	Testing
206	Pay Slip Generation

Advantages

- Structural independence:- Any changes to the database structure, does not the way we are accessing the data. For example, Age is added to Employee table. But it does not change the relationship between the other tables nor changes the existing data. Hence it provides the total independence from its structure.
- Simplicity:- This model is designed based on the logical data. It does not consider how data are stored physically in the memory. Hence when the designer designs the database, he concentrates on how he sees the data. This reduces the burden on the designer.
- Because of simplicity and data independence, this kind of data model is easy to maintain and access.
- This model supports structured query language – SQL. Hence it helps the user to retrieve and modify the data in the database. By the use of SQL, user can get any specific information from the database.

Disadvantages

Compared to the advantages above, the disadvantages of this model can be ignored.

- High hardware cost:- In order to separate the physical data information from the logical data, more powerful system hardwares – memory is required. This makes the cost of database high.
- Sometimes, design will be designed till the minute level, which will lead to complexity in the database.

Below table provides the comparison among the three models

Hierarchical Data Model	**Network Data Models**	**Relational Data Models**
Supports One-Many Relationship	Supports both one to many and Many to Many relationship	Supports both one to many and Many to Many relationship
Because of single parent-child relationship, difficult to navigate through the child	It establishes the relationship between most of the objects, hence easy to access compared to hierarchical model	It provides SQL, which makes the access to the data simpler and quicker.
Flexibility among the different object is restricted to the child.	Because of the mapping among the sub level tables, flexibility is more	Primary and foreign key constraint makes the flexibility much simpler than other models.
Based on the physical storage details	Based on the physical storage details	Based on the logical data view