Chapter 13

DBMS concepts

Introduction to file system

An abstraction to store, retrieve and update a set of files is called, file system. A file system also includes the data structures specified by some of those abstractions. These data structures are designed to organize multiple files as a single stream of bytes. In a file system other abstractions also specified some network protocols, these are designed to allow files access on a remote machine.

The file system manages access to the data and the metadata of the files. A file system ensures the reliability and it is the major responsibility of the system.

File system disadvantages (problems):

• **Data redundancy**: Same information available in many files e.g. student address in different files for different purposes.

• **Data Access difficulty**: It requires new program when new request arrives that is, each time new program has to write to full fill coming request because program was not available for new request.

• **Data is isolated:** in different files which are also in different formats.

Multiple users can not access the same data simultaneously because there was difficulty in supervision for concurrent request.

• **Difficulties with security** enforcement.

Integrity issues: Constraints must be ensured on database.

Advantages of file system:

- Easy design with single-application.
- A single application based optimized organization.
- Efficient in term of performance.

HIERARCHY OF DATA

Data stored in computer systems can be view in following manner and we can define database management system after that.

Data are logically organized into

1. Bits (characters)
2. Fields
3. Records
4. Files

5. Databases
Bit-a bit is the smallest unit of data representation (value of a bit may be a 0 or 1)
Field - a field consists of a grouping of characters. A data field represents an attribute (a characteristic or quality) of some entity (object, person, place, or event).
Record - a record represents a collection of attributes that describe a real-world entity. A record consists of fields, with each field describing an attribute of the entity.
File - a group of related records. A **primary key** in a file is the field (or fields) whose value identifies a record among others in a data file.
Now we can define database management system. As the name suggests, the database management system is made of Database and Management System.
DATABASE: To find out what database is, we have to start from data, which is the basic building block of any DBMS.

Data: Facts, figures, statistics etc. having no particular meaning (e.g. 2, XYZ, 19 etc).
Record: Collection of related data items.
File - a group of related records
Database: Collection of interrelated data or data files or relation(for relational database) . This collection of data is interrelated so it can be a relevant information of an organization and this collection of information can be accessed using a set of application program.
Management system: A management system is a set of rules and procedures which help us to create, organize and manipulate the database. It also helps us to add, modify and delete data items in the database.

DBMS

A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data.

Goals of DBMS: Design of any DBMS system has following goals

1. Main goal of any DBMS system is to manage large bodies of information.
2. Provide convenient way to store information in database.
3. Efficient retrieval of information from databases.
4. Safety and security of information stored in database.
5. Avoiding the anomalies during simultaneous access of information by many

users.

Advantages of DBMS: Over conventional file system DBMS has many advantages. These advantages make DBMS more useful in many applications. Following are the advantages of DBMS.

1. **Removing the data redundancy(duplicacy):** If same information is stored in many places it will waste storage spaces and effort . This redundancy problem is handled in DBMS.
2. Sharing of Data: But in computerized DBMS, many users can share the same database.
3. Data Integrity: We can maintain data integrity by specifying integrity constrains, which are rules and restrictions about what kind of data may be entered or manipulated within the database. This increases the reliability of the database as it can be guaranteed that no wrong data can exist within the database at any point of time.
4. Data independence: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.
5. Efficient data access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.
6. **Data integrity and security**: Security is ensured by providing abstract view of data and integrity constraints. DBMS provides abstract view so that no need to see all kind of information by all kind of users that is , particular user can see only the part of database .
7. **Reduced application development time**: Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in Conjunction with the high-level interface to the data, facilitates quick development of applications.
8. **Recovery in DBMS:** During transaction failure your database will be restore in its original state

Applications of DBMS: In almost all area's DBMS has its applications. Some of these are

- Banking: all transactions of banking sector

- Airline: reservation, schedules, availability
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Example of DBMS: There are numbers of DBMS which is currently in use

Commercial DBMS:

Company	**Product**
Oracle	8i, 9i,10g
IBM	DB2, Universal Server
Microsoft	Access, SQL server
Sybase	Adaptive Server
Informix	Dynamic server

Along with commercial DBMS, one of the widely used Open source DBMS is MySQL.

Abstraction levels in a DBMS

As we have discussed one of the main goal of any DBMS is to simplify the user's interaction with the database that is, any kind of users(naive, programmers, sophisticated etc.) can easily and efficiently retrieve information from database. Abstract view of data helps to hide certain details of how data is stored and maintained.

Physical level: The physical schema specifies how the relations are actually stored in secondary storage devices. It also specifies auxiliary data structures (indexes) used to speed up the access to the relations.
Logical level: The conceptual schema describes the data stored in the database and relationships among those data that is, conceptual schema defines the logical structure of entire database. For example, in a relational database it describes all the relations stored in the database.
View level: The View level is a refinement of the conceptual level. It allows customized and authorized access to individual users or groups of users. Every database has one conceptual and one physical schema, but it can have many schemas at view level. A view(external schema) is conceptually a relation, but its records are not stored in the database instead, they are computed from other

relations. Figure 1 shows the relationship between these levels of abstraction.

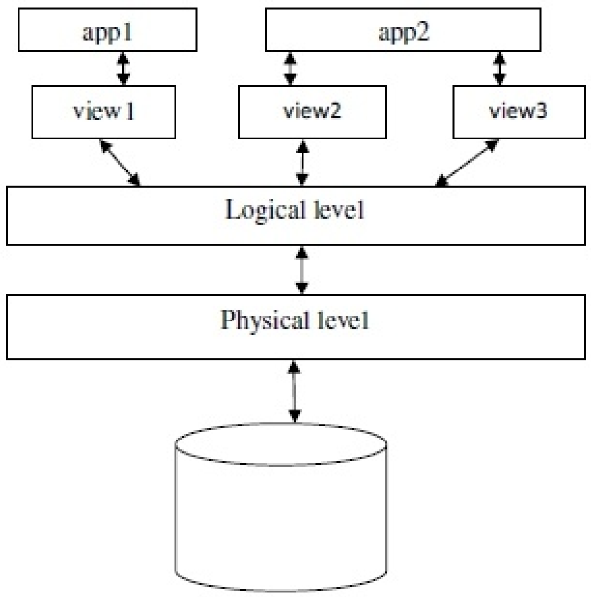


Figure 1. Data abstraction levels

Example school database: Above three schemas depicted in figure 1 can be understood by using school database example.

Example physical schema: Relations stored as unordered files and index on first column of Students.

Example conceptual schema:

Students(Roll_no: int, name: varchar, address: varchar, age: integer, class: char)

Subjects(subjectid: char, Sname: char,)

admission(Roll_no: int, ClassName:char, AdmissionDate: date)

Example external Schema (View):

Class_info(ClassName:char, Strength:integer)

Schemas and instances

A schema is a description (over all design) of the data in terms of the data model. We can say about a schema (metadata) that, a schema is a specification of how data is to be structured logically and rarely changes.

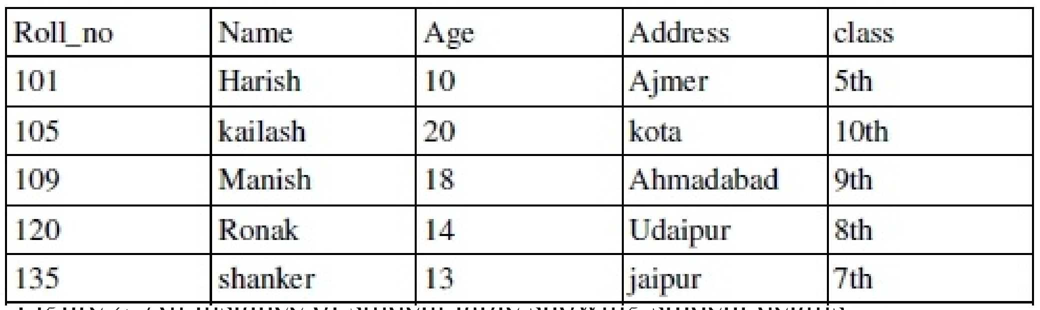
In the relational model the schema looks like:

RelationName(field1 : type1, ..., fieldn : typen)

Students(Roll_no : int, name : char, age : integer, class : char)

An instance on the other hand represents the contents of schema at any particular moment of time which changes rapidly, but always conforms to the schema. we can compare schema and instances with type and objects of type in a programming language. An instance of the student relation, can be represented as depicted in figure 2.

Student table



Roll_no	Name	Age	Address	class
101	Harish	10	Ajmer	5th
105	kailash	20	kota	10th
109	Manish	18	Ahmadabad	9th
120	Ronak	14	Udaipur	8th
135	shanker	13	jaipur	7th

Figure 2: An instance of student table showing student details

Database languages: A database system has, Data Definition Language (DDL) to specifies the data schemas and data Manipulation Language (DML) to facilitate the retrieval and update of data in the database. DML are basically of two types.

1. Procedural DML: It requires that the data and the procedure to obtained those data must be specified by the user.

2. **Non-procedural DML**: In non-procedural DML only required data is specified by the user without specifying the procedure to obtain those data. A DBMS provides a specialized language, called **query** language to ask questions to the DBMS. For retrieval, we query the database with the query language, which is part of the DML. Term query language and DML are synonymous.

Classification of DBMS: A DBMS system can be of several types based on following criteria.

1. **Based on users:** A first criteria is the number of users supported by the system. **Single-user systems** support only one user at a time and are mostly used with personal computers. **Multiuser systems**, which include the majority of DBMS, support multiple users concurrently.

2. **Based on Architecture:** A second criteria is the number of computer system on which database system runs. A **centralized or client-server** DBMS can support multiple users, but the DBMS and the database themselves reside and runs totally at a single computer site(server machine). A **distributed**

DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. Homogeneous DDBMSs use the same DBMS software at multiple sites.

3**. Based on types of data models:** Third criteria is the types of data models on which the DBMS is based. These DBMS are can be of following types.

- Hierarchical databases.
- Network databases.
- Relational databases.
- Object-oriented databases

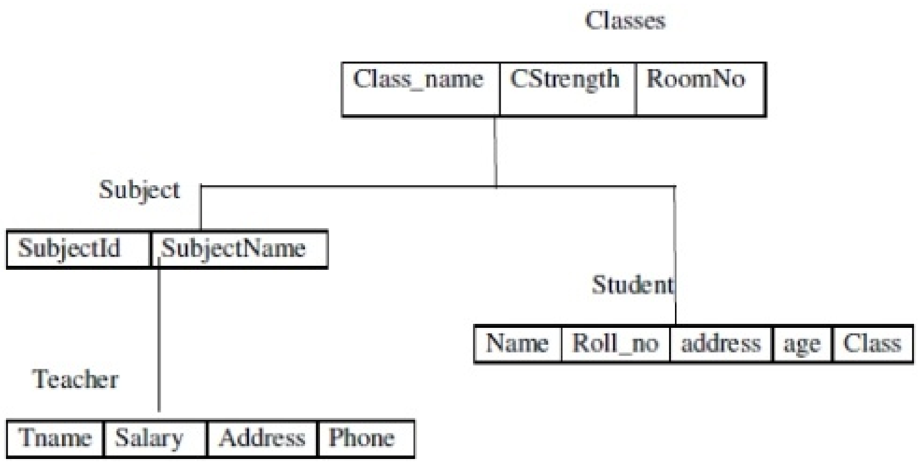
Data model:

A data model is a collection of concepts for describing data, the meaning of data, data constraints and data relationships. Some of the data models are Hierarchical, Network, Relational and Object-oriented.

Hierarchical data model :

We can find older systems that are based on a hierarchical model .The first hierarchical DBMS is "IMS" and it was released in 1968. The hierarchical DBMS is used to model one-to-many relationships, presenting data to users in a treelike structure. Within each record, data elements are organized into pieces of records called segments. To the user, each record looks like an organizational chart with one toplevel segment called the root. An upper segment is connected logically to a lower segment in a parent–child relationship. A parent segment can have more than one child, but a child can have only one parent. Figure 1 shows a hierarchical structure that might be used for school management system.

The root segment is classes, which contains basic classe's information such as name, strength, and room number. Immediately below it are two child segments: subjects (containing subject id and subject name data), student containing (name , address, rollno , age and class data). The subject segment has one child below it: teacher (containing data about teacher name , salary, address, phone and results evaluations) .



It is found that in large legacy systems where high volume transaction processing is required, hierarchical DBMS can still be used. Banks, insurance companies, and other high-volume users continue to use reliable hierarchical databases,

Network data model: **A** network DBMS depicts data logically as many-to-many relationships. In other words, parents can have multiple children, and a child can have more than one parent. A typical many-to-many relationship for a network DBMS is the student–subject relationship (see Figure 4 below). There are many subjects and many students in a classs. A student takes many subjects, and a subject has many students.

Hierarchical and network DBMS are considered outdated and are no longer used for building new database applications.

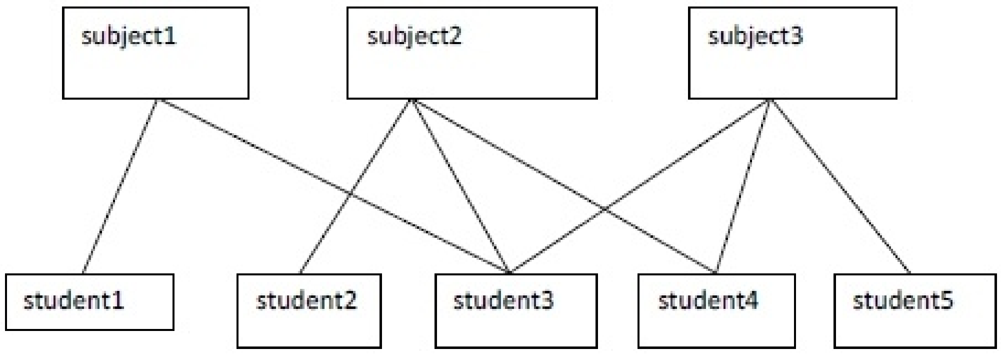


Figure 4 Network model for student subject relationship

Relational data model:

This is a record based data model. It uses a collection of relations (or tables) to represent data and relationship between data. Each relation has a list of attributes (or columns) which has a unique name. Each attribute has a domain (or type). Each relation contains a set of tuples (or rows). Each tuple has a value for each attribute of the relation. Duplicate tuples are not allowed. It is the data model used by mostly current database systems. Given below is an example of student table showing students details.

Example

Student table

Roll_no	Name	Age	Address	Class
101	Harish	10	Ajmer	5th
105	Kailash	20	Kota	10th
109	Manish	18	Ahmadabad	9th
120	Ronak	14	Udaipur	8th
135	Shanker	13	Jaipur	7th

Figure 5: Relational data model for student database

Database design steps or phase

Designing a DBMS application for any enterprise should follow some phases.

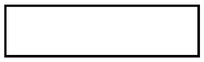
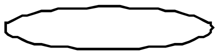
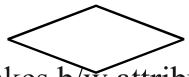
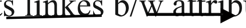
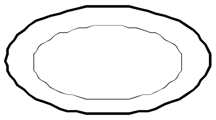
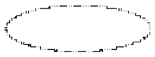
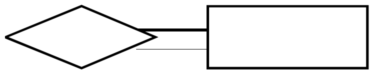
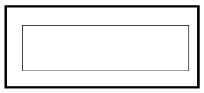
- **Requirements analysis: In this phase data needs of an organization is identified.**
- **Conceptual Database design): Mostly done using the ER model. Concept of chosen data model is applied on identified data to construct the conceptual schema.**
- **Logical Database design: In this phase high level conceptual schema maps onto the implementation data model of the database system that will be used e.g. for RDBMS it is relational model.**
- **Schema refinement: Normalization is applied to refine schema into smaller schema**
- **Physical Database Design: This phase specified the physical features of the database e.g. internal storage structure, form of file organization etc.**

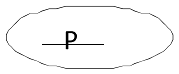
E-R model and E-R diagram

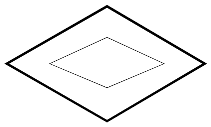
E-R model:

Entity-Relationship (ER) model is a popular conceptual data model. The model describes data to be stored and the constraints over the data. E-R model views the real world as a collection of **entities** and **relationships** among entities. An entity is an "object" in real world that can be differentiable from other objects.

E-R diagram: It is basically graphical representation of overall logical structure of a database. The main components in this diagram are as follows.

Symbol name	Symbol	Purpose
Rectangles		Represent entity set
Ellipses		Represents attributes
Diamonds		Represents relationships
Lines		Represents linkes b/w attributes to entity set and entity sets to relationship set
Double ellipses		Multivalued attributes
Dashed ellipses		Represents derived attributes
Double lines		total participation
Double rectangles		Represents weak entity sets

Ellipses with line  Primary key

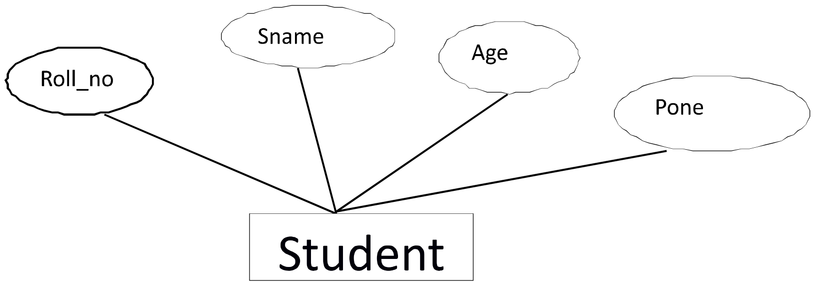
Double diamonds  Identifying relationship for weak entity set

Entity: is an object in the real world that is distinguishable from all other objects. E.g., A class, A teacher, The address of the teacher, a student, a subject. An entity is described using a set of attributes. Each attribute has a domain of possible values.

Entity Set: An entity set is a collection of entities of similar objects(same type). In an entity set values of attributes are used to distinguish one entity from another of same type. For each entity set, we should identify a key. A key is a minimal set of attributes that uniquely identify an entity in a set. All entries in a given entity set have the same attributes (the values may be different).

Representation of entity set and attributes in E-R diagram: An entity set can be represented as a rectangle in E-R diagram.

Graphical representation in E-R diagram: Example of student entity

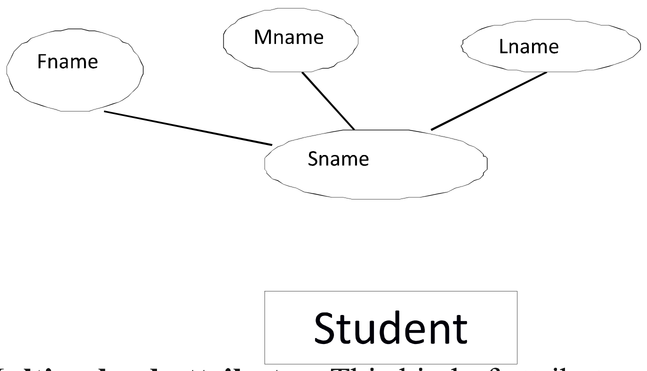


Different attribute types:

Attributes are properties of entities and relationships like, attributes of tuples or objects. Attributes can be of following types and represented as ovals in E-R diagram.

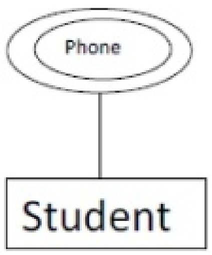
- **Simple attribute:** This kind of attributes contains a single value e.g. in above student entity Roll_no and age are simple attributes.
- **Composite attribute:** This kind of attributes contains several components e.g. Sname attributes contains first name, middle name, last name components.

Graphical representation in E-R diagram: Example of student entity and its attribute Sname.



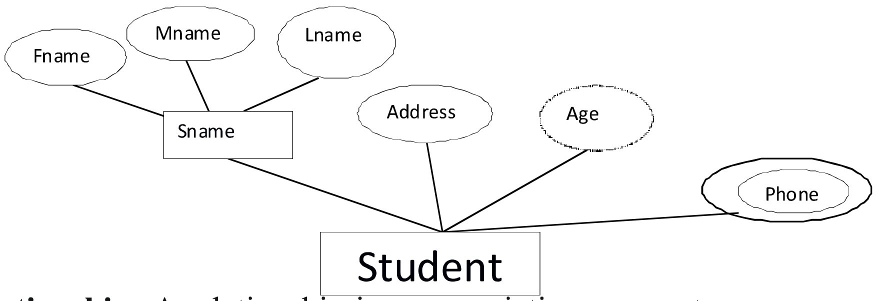
- **Multi-valued attribute:** This kind of attributes contains more than one value e.g. phone attributes contains many mobiles numbers, land line number, office number.

Graphical representation in E-R diagram: Example of student entity and its phone attribute.



- **Derived attribute:** Value of this kind of attributes can be computed from other attributes e.g., age can be computed from data of birth and the current date.

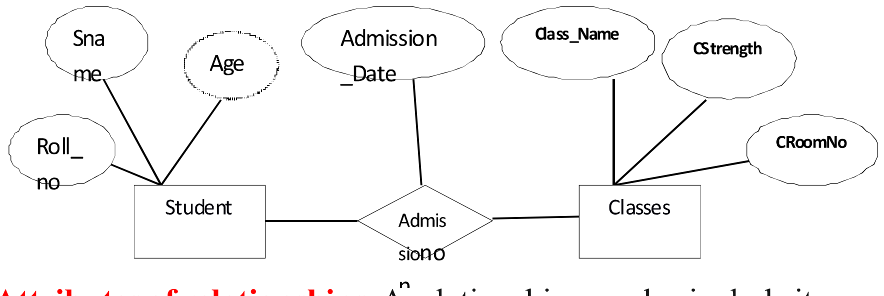
Graphical representation in E-R diagram: Example of student entity and its age attribute along with multi valued and composite attribute.



Relationship: A relationship is an association among two or more entities. Relationship that involve two entity sets are called binary (or degree two) relationship .

Relationship set: A set of relationships of the same type(among same entity sets) e.g. student admission in classes. Here admission is a relationship between student and classes entity sets. It can be represented as a diamond in E-R diagram.

Graphical representation in E-R diagram:



Attributes of relationships: A relationship can also include its own attributes (called descriptive attributes). For example assume that students take admission in particular class on a particular date.

So where do the admission date go? Following are two possibilities here

With *Student* ?

1. But a student can have different admission dates for different classes.

With classes ?

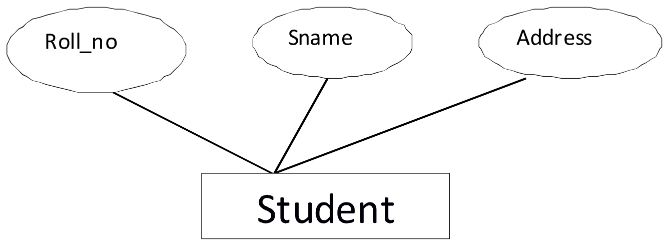
2. But a class can assign different admission date for multiple students. So it will go with admission as shown in above relationship example of E-R diagram..

Constraints:

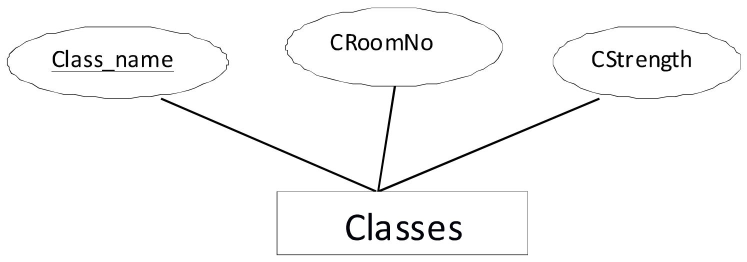
E-R model describes data to be stored and also the constraints over the data. These constraints are mapping cardinality, key constraints and participation constraints.

Key constraints: A key is a set of attributes whose values can belong to at most one entity in an entity set that is, a set of attributes that can uniquely identity an entity e.g. Roll_no is a key of student entity set. A key of an entity set is represented by underlining all attributes in the key.

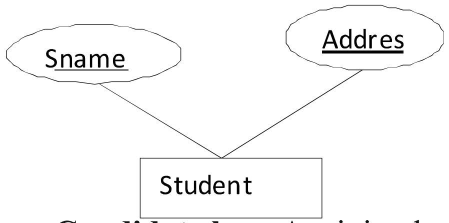
Graphical representation in E-R diagram: Example of Roll_no attribute in student entity.



E.g2. In classes entity set Class_Name is the primary key which is unique because we are assuming that 12 different classes from 1st to 12th exit in a school without any sections.



- Composite Key: Two or more attributes are used to serve as a key e.g. Name or Address alone cannot uniquely identify a student, but together they can identify a student.



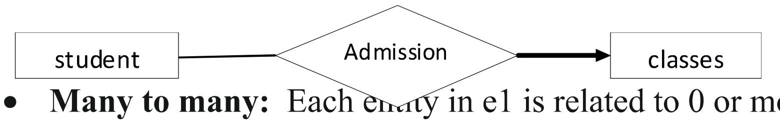
- **Candidate key:** A minimal set of attributes that uniquely identifies an entity is called a candidate key. e.g.{Roll_no) and {Sname, Address}both are two candidate keys but {Roll_no, Sname}is not a candidate key. If there are many candidate keys, we should choose one candidate key as the primary key.

Mapping cardinality constraints: Mapping cardinality between two entity sets e1 and e2 for binary relationship R can be of following types.

- **Many to one:** Each entity in e1 is related to 0 or 1 entity in e2, but each entity in e2 is related to 0 or more in e1.

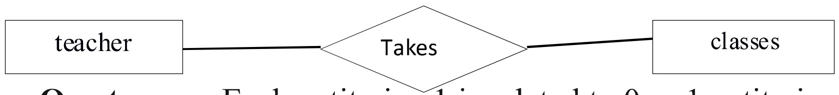
Example: One student can be in one class at a time but one class can have many students.

Graphical representation in E-R diagram: Using lines.



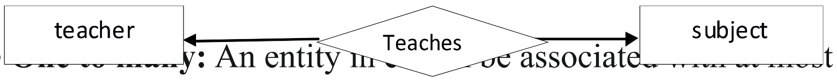
- **Many to many:** Each entity in e1 is related to 0 or more entities in e2 and vice versa.

Example: Many teachers can teach one class or many classes can be taught by a single teacher.



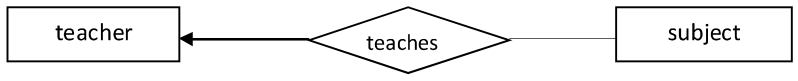
- **One to one:** Each entity in e1 is related to 0 or 1 entity in e2 and vice versa.

Example: A teacher can only teach one subject e.g. Hindi , English ,Maths . It is assuming that in school particular subject teacher is fixed.



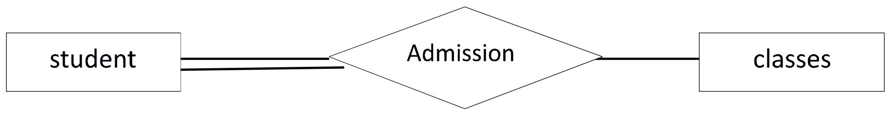
- **One to many:** An entity in e1 can be associated with at most one entity in e1.

Example: A teacher can also teach many subject in special case if teacher of particular subject is not available in school.



Participation constraints: Diagram below **s**hows the participation of an entity set's all entities in a relationship. There are two types of participation constraints for an entity in a relationship.

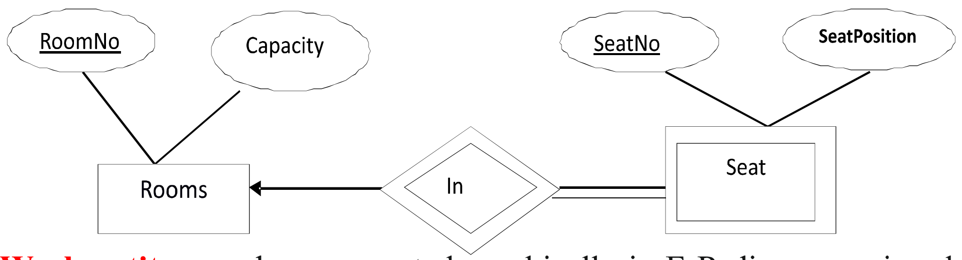
- **Total:** Every instance of the entity is present in the relationship (represent it by a thick line).
  Example: In this example participation of student entity is total because every student must take admission in one class.
  Graphical representation in E-R diagram: Using double lines.



- **Partial:** Not every instance of the entity is present in the relationship.

Example: Participation of classes is **partial** if some classes do not have student admission.

Weak Entities: Sometimes, the key of an entity set E can not be completely formed by its own attributes, but from the keys of other (one or more) entity sets to which E is linked by many to one (or one to one) relationship sets E can be completely formed that is, An entity cannot be uniquely identified by all attributes related to this entity. Such entity is called weak entity.
Example: Suppose each room in school can have seat numbers for each student. Attributes of seats are SeatNo and SeatPosition. These attributes can not uniquely identified a seat entity. So seat is an weak entity.



Weak entity: can be represented graphically in E-R diagram using double diamond. A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity (called **identifying or owner entity**) and relating relationship is called **identifying relationship** set.

Database design for a school management system:

Firstly, we assume some of the characteristics of a school.
1. A school have many classes from 1st to 12th.
2. Each class study number of subjects.
3. Many teachers work in a school.
4. A teacher can teach only one subject.
5. Each teacher can take any number of classes with same subject.
6. A student can take dmission in any class.
7. Each class can have any number of students.
8. Each class has its own time table.

How we can start E-R modeling?

Designing of an E-R model can be done by identifying**.**

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key

Step 1: Identify the Entities:

- CLASSES
- STUDENT
- SUBJECT
- TEACHER

Step 2: Finding the relationships:

- Many classes contains many subjects and class time table contains subjects , hence the cardinality between subjects and classes is many to many .

- One class has multiple teachers and one teacher belongs to many classes for same subject , hence the cardinality between classes and teacher is many to Many.

- One class admits multiple students and one student admits in one and only one class.
  Hence the cardinality between class and student is one to Many.

- One subject is taught by only one teacher, hence the cardinality between subject and teacher is one to one.

- Many subjects read by multiple students and one student reads multiple subjects in a class, hence the cardinality between subjects and student is Many to Many.

Step 3: Identify the key attributes and other relevant attributes:

- SubjectId is the key attribute and SubjectName is the other attribute for "subject" Entity.
- Roll_no is the key attribute and Sname, Age, Address, phone are other attributes for "student" Entity
- Tname is the key attribute and address, phone, salary are other attributes for "Teacher" Entity.
- Class_name is the key attribute and CStrength, CRoomNo are other attributes for the Entity "classes".
- CRoomNo is the key attribute and PeriodNo is other attributes for the Entity "timetable".

Complete E-R diagram of the school database

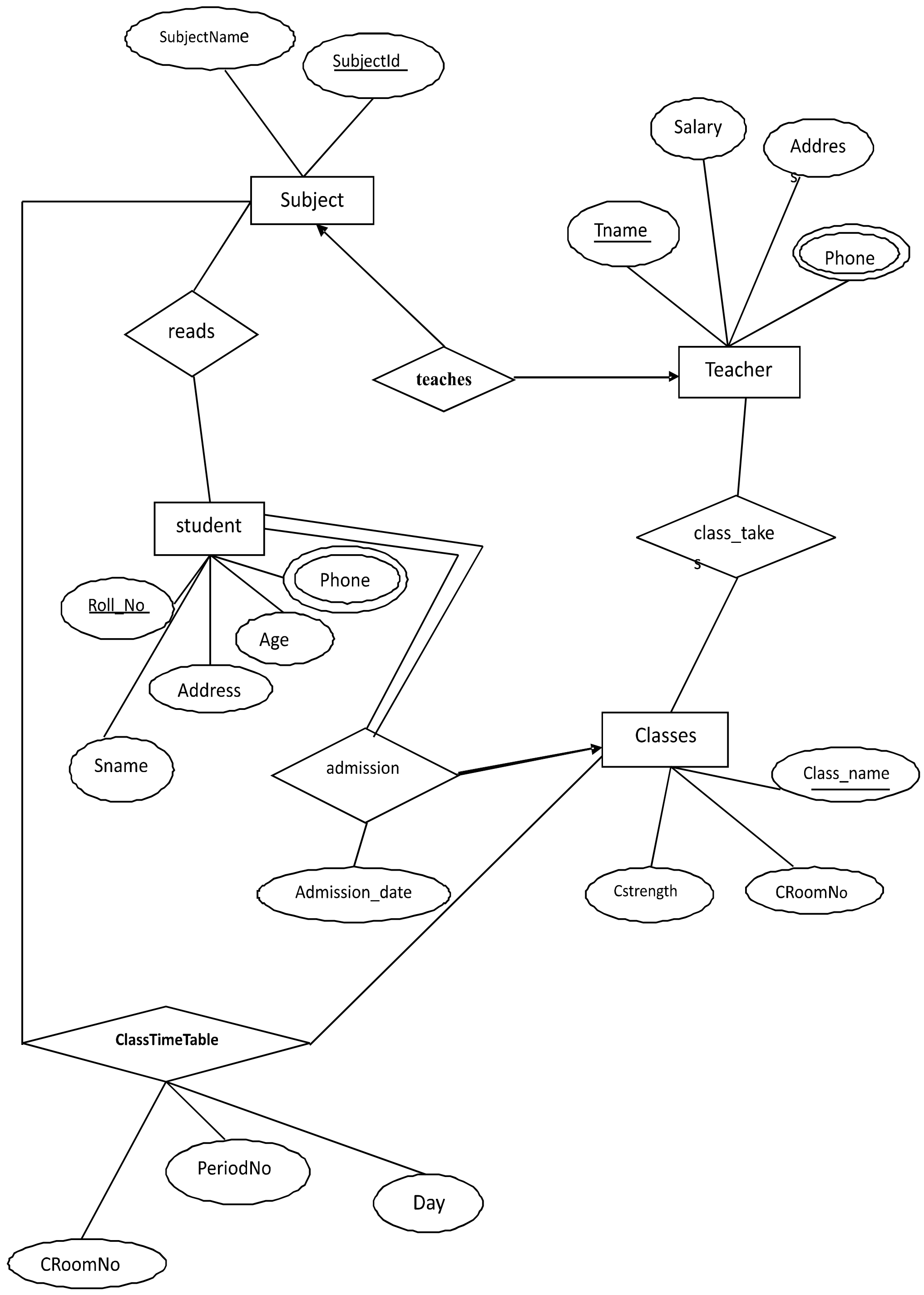


Figure 6. E-R diagram of a school from 1st to 12th

Translating E-R into Relational schema: Given E-R diagram in Fig 6 can be converted into relational design. This mapping can be done by some steps.

1. **Translating entity sets:** For entity set all its attributes will form columns of table and its key attribute will be key column that is,

- Attributes → columns
- Key attributes → key columns

So converted schema's of school E-R Diagram are.

Student(Roll_No , Sname, age, Address, Phone)
Classes(Class_name, Cstrength, CRoomNo)
Teacher(Tname, Salary, Address, Phone)
Subject(SubjectId, SubjecrName)

2. Translating relationship sets:

A relationship set will also be translated into a table as.

- Keys of connected entity sets will be the columns of table.
- Attributes of the relationship set (if any) will be columns of table.
- Foreign key in relationship set will be primary key of participating entity sets.
- Multiplicity of the relationship set determines the key of the table and will be as follows.

For (one-to-one Relationship): Primary key of either entity set can be primary key of relationship.

For (one-to-many Relationship) or many to one: Primary key of the entity set on the "many " side of the relationship set will be primary key of relationship set.

For (Many-to-many Relationship): Combination of primary keys of participating entity sets will form the primary key of relationship.

So converted schema's of school diagram for relationships are.

Class_takes(Class_name, Tname)
Admission(Roll_No, Class_name, Admission_date)
Teaches(SubjectId, Tname)
Reads(Roll_No, SubjectId)
ClassTimeTable(Class_name,SubjectId, PeriodNo, CRoomNo, PeriodNo)

After merging resultant schema's of school database:

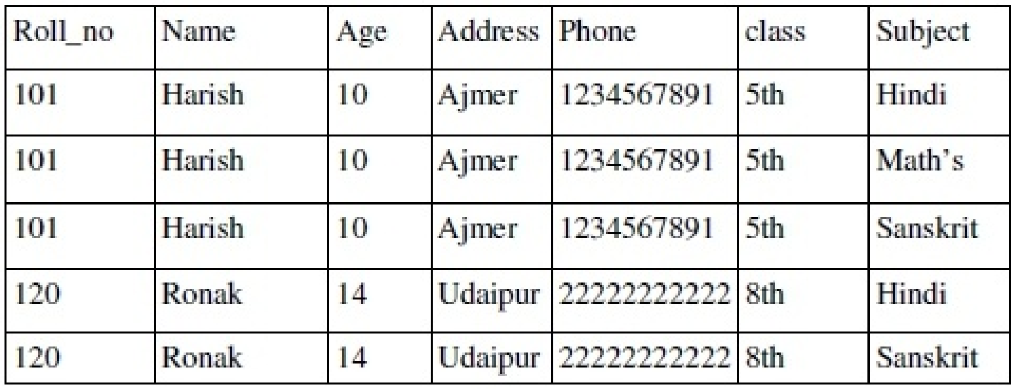
Student(Roll_No , Sname, age, Address, Phone, Class_name, admission_date)
Classes(Class_name, Cstrength, CRoomNo)
Teacher(Tname, Salary, Address, Phone)
Subject(SubjectId, SubjecrName)
Class_takes(Class_name, Tname)
Teaches(SubjectId, Tname)
Reads(Roll_No, SubjectId)
ClassTimeTable(Class_name, SubjectId, PeriodNo, CRoomNo, PeriodNo)

Introduction to normalization

One more method for designing a relational database is to use a process commonly named as normalization. The main purpose is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, along with that also allows us to retrieve information easily. So to do all design, design the schemas in appropriate normal forms using the concepts of functional dependencies.

Bad database and purpose of normalization: To understand the concept of bad database and purpose of normalization we consider following student table.

Student



Roll_no	Name	Age	Address	Phone	class	Subject
101	Harish	10	Ajmer	1234567891	5th	Hindi
101	Harish	10	Ajmer	1234567891	5th	Math's
101	Harish	10	Ajmer	1234567891	5th	Sanskrit
120	Ronak	14	Udaipur	22222222222	8th	Hindi
120	Ronak	14	Udaipur	22222222222	8th	Sanskrit

By analyzing the above schema we can easily find that this design is not a good database design.

Why this design is bad?

For student reads subjects(Roll_no, name, age, address, class, subject) This design has redundancy, because the name of a student, age, address, class is recorded multiple times, once for each subjects the student is reading. This redundancy caused serious problems in design that is, it wastes storage space and introduce potential inconsistency in database. That is why this database design is bad.

A bad database design also caused many problems(Anomalies) during operation on database.

Update anomalies: all repeated data needs to be updated when one copy is updated. For example if we want to update the address of a particular student we have to update all tuples of that student.

Insertion anomalies: It might not be possible to store certain data unless some other, unrelated information is also stored. We need to know the phone in order to insert a tuple .This could be fixed with a NULL value but nulls also caused problems or hard to handle.

Deletion anomalies: It may not be possible to delete certain information without losing some other, unrelated information that is if we delete all the tuples with a given (class, Roll_no) we might lose that association. So if we want to detect and remove redundancy in designs we need a systematic approach. In another word, if we want to design a good database then we have to normalized it by using Dependencies, decompositions and normal forms.

Functional dependencies:

A functional dependency (FD) is a kind of IC that generalizes the concept of a key. Let R be a relation schema, with X and Y be nonempty sets of attributes in R. For an instance r of R, we say that the FD

$X \rightarrow Y$ (X functionally determines Y)

is satisfied if: $\forall$ t1, t2 $\in$ r, t1.X = t2.X =) t1.Y = t2.Y

$X \rightarrow Y$ means that whenever two tuples in R agree on all the attributes in X, they must also agree on all attributes in Y

Example of a Functional Dependency:

The functional dependency AB $\rightarrow$ C: satisfies for following instance

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

- A primary key is a special case of a FD: If X → Y (holds, where Y is the set of all attributes, then X is a superkey
- FDs should hold for any instance of a relation.
- Given a set of FDs we can usually find additional FDs that also hold.

Example: Given a key we can always find a superkey.

Redefining keys using FDs:

A set of attributes K is a key for a relation R if K → all (other) attributes of R that is, K is a "super key". No proper subset of K satisfies the above condition that is, K is minimal.

Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. In this process we check a given relation schema against certain normal form to check whether or not it satisfies certain normal form. If a relation schema does not satisfies certain normal form then we decompose it into smaller schemas.

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Relational database design: To design a relational database ,we need to know for a given schema that it is a good design. If design is not good then we decompose it into smaller schemas but the decomposition should be good. After that we check each decomposed schema for certain normal forms. If a given relation schema is in a normal form then we know some problems cannot arise.

Normal forms: Various normal forms are

1. First Normal Form(1NF)
2. Second Normal Form(2NF)
3. Third Normal Form(3NF)
4. BCNF

First Normal Form(1NF): A relation is in first normal form if every field contains only atomic values (no lists nor sets).

Example:

Student table given below in figure 7 is not in 1NF but in figure 8 is in 1NF

Student

Roll_no	Name	Age	Subject
101	Harish	10	hindi,maths
120	Ronak	14	maths

Figure 7. Student table showing subjects of students not in 1NF

Student

Roll_no	Name	Age	Subject
101	Harish	10	Hindi
101	Harish	10	Maths
101	Harish	10	Sanskrit
120	Ronak	14	Hindi
120	Ronak	14	Maths

Figure 8. Student table showing subject of student in 1NF

Second normal form

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has primary key, every non prime attribute in the table should be fully functionally dependent on primary key attribute . If any column depends only on one part of primary key, then the table fails in Second normal form.

Student Reads Subject (Roll_no, subject Id, Sname, address, Subject Name)

In this student subject relation primary key attribute are Roll_no, and subjecteId. According to the rule, non-key attributes, i.e. Sname and SubjectName must be dependent upon both and not on any of the prime key attribute individually. But we find that Sname can be identified by Roll_no and SubjectName can be identified by SubjectId independently. This is called partial dependency, which is not allowed in Second Normal Form.

We broke the relation in two as depicted in the below picture. So there exists no partial dependency.

Student(Roll_no, Sname, address)

Subject(SubjectId, SubjectName)

Third Normal Form

If R is a relation schema, F is a set of functional dependencies on R and A is a single attribute in R then to check whether this schema is in 3NF or not, for every FD we check the following conditions. If for any FD, these conditions fails then given schema R will not be in 3NF.

Conditions:

- If any FD is trivial that is, in β A (A β), *or*
- IF left side attribute in FD is a key for schema *or*
- If right side attribute is *part of* some key(s) for R

BCNF: To check for a relation schema that, whether this schema is in BCNF or not, for every FD we check the following conditions. If for any FD, these conditions fails then given schema R will not be in BCNF.

Conditions:

- If any FD is trivial that is, in β A (A β), *or*
- IF left side attribute in FD is a key for schema *or*

If any relation is in BCNF then it will be also in 3NF. That is BCNF implies 3NF but 3NF can not implies BCNF.

Design goals

So for a good relational database design, a relational schema should be in BCNF with Lossless- join and dependency preservation. If we cannot achieve this, we accept 3NF with Lossless-join and dependency preservation.

Important Points:

- DBMS is used to maintain and query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity, security and sharing of Data.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- Functional components of DBMS are file manager, Buffer manager, Query processor, Data files, Data dictionary and Indices.
- A DBMS system can be of several types based on criteria, based on users, architecture and types of data models
- Entity: is an object in the real world that is distinguishable from all other objects.

- A minimal set of attributes that uniquely identifies an entity is called a candidate key.
- A bad database design caused many anomalies during operation on database such as update anomalies, insertion anomalies and deletion anomalies.
- A functional dependency (FD) is a kind of IC that generalizes the concept of a key
- If any relation is in BCNF then it will be also in 3NF. That is BCNF implies 3NF but 3NF can not implies BCNF.

Practice Questions

Objective type questions:

Q1. Which one of the following is not a goal of DBMS.

a) Managing large information   b) Efficient retrieval

c) Preventing concurrent access   d) Safety of data

Q2. Which one of the following is an example of a commercial DBMS.

a) Oracle   b) IBM

c) Sybase   d) all

Q3. Which one of the following is the simplest level of data abstraction.

a) Physical   b) Logical

c) View   d) none of these

Q4. Normalization means

a) Joining relations   b) Decomposition of relation

c) both   d) none of these

Q5. Which normal form is more restricted?

a) 1NF   b) 2NF

c) BCNF   d) 3NF

Very short answer type questions:

Q1 What is DBMS?.

Q2. Define a record.

Q3. Give names of different data redundancy.

Q4. Define database schema.

Q5. What is the role of indexes in DBMS?

Q6. What is a query language?

Q7. Differentiate between procedural and non -procedural DML.

Q8. Differentiate between schema and instances.

Q9. What are the steps of designing a database?

Q10. What is an entity?

Short answer type questions:

Q1. What do you understand by atomicity?

Q2. Differentiate between logical and physical data independence.

Q3. What is a weak entity? Draw it in E-R diagram?.

Q4. Differentiate between primary and composite key.

Q5. What is a bad database?.

Essay type questions:

Q1.What are the various components of DBMS? Explain with suitable diagram.

Q2. What is a data model.? Explain hierarchical data model? How it is differ from network data model.

Q3. Explain various types of attributes and relationships in E-R diagram and also give graphical representation for them.

Q4. Design an E-R diagram for a school consisting of different classes from 1st to 10th.

Q5. What is normalization? Give various forms of normalization.

Answers key for objective questions

Q1: c	Q2: d	Q3: c	Q4: b	Q5: c