

LECTURE NOTES OF DATABASE ENGINEERING (BCS-204) CR.-04

Prepared by

Dr. Suvasini Panigrahi, Dr. Satyabrata Das, Dr. Rakesh Mohanty



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Veer Surendra Sai University of Technology (VSSUT)

Formerly University College Of Engineering, Burla

Sambalpur, Odisha, India

LECTURE PLAN OF DATABASE ENGINEERING (3-1-0) Cr.-04

MODULE – I (10 Lectures)

- Lecture 1: Introduction to Database Systems and its Applications
- Lecture 2: Database System Architecture
- Lecture 3: Data Abstraction, Data Independence
- Lecture 4: Data Models – Network, Hierarchical and Relational Data Models
- Lecture 5: Data Models – Relational and Object Oriented Data Models
- Lecture 6: Entity Relationship (ER) Model
- Lecture 7: Mapping ER Model to Relational Model
- Lecture 8: Data Definitions Languages
- Lecture 9: Data Manipulation Languages
- Lecture 10: Integrity Constraints and Data Manipulation Operations

MODULE – II (08 Lectures)

- Lecture 11: Relation Query Language, Fundamental Relation Algebra Operations
- Lecture 12: Extended Relation Algebra Operations
- Lecture 13: Tuple and Domain Relational Calculus
- Lecture 14: SQL
- Lecture 15: QBE, Domain and Data Dependency
- Lecture 16: Armstrong's Axioms, Dependency Preservation, Lossless Design
- Lecture 17: Normal Forms – 1NF, 2NF, 3NF, BCNF
- Lecture 18: Higher Normal Forms – 4NF, 5NF, Comparison of Oracle and DB2

MODULE – III (08 Lectures)

- Lecture 19: Introduction to Query Processing
- Lecture 20: Translating SQL Queries into Relational Algebra
- Lecture 21: Algorithms for External Sorting
- Lecture 22: Algorithms for SELECT and JOIN Operations
- Lecture 23: Algorithms for PROJECT and SET Operations
- Lecture 24: Query Equivalence, Join Strategies
- Lecture 25: Overview of Query Optimization
- Lecture 26: Query Optimization Algorithms

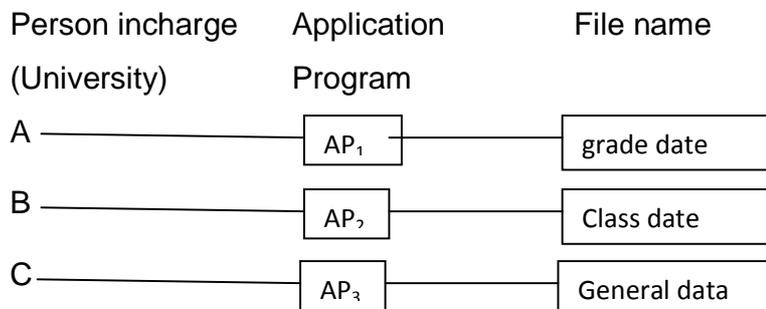
MODULE – IV (12 Lectures)

- Lecture 27: Storage Strategies – Indices, B-Trees
- Lecture 28: Storage Strategies – Hashing
- Lecture 29: Introduction to Transaction Processing, Transaction and System Concepts
- Lecture 30: Desirable Properties of Transactions
- Lecture 31: Schedules and Recoverability
- Lecture 32: Serializability of Schedules
- Lecture 33: Locking Techniques for Concurrency Control, Time-stamp based schedules
- Lecture 34: Multi-version and Optimistic Concurrency Control Schemes
- Lecture 35: Database Recovery Techniques
- Lecture 36: Advanced Topics – Object-Oriented and Object Relational Databases, Logical Database
- Lecture 37: Advanced Topics – Web Databases, Distributed Databases
- Lecture 38: Advanced Topics – Data Warehouse and Data Mining

Traditional File Oriented Systems

A time when there were no database system

Scenario: University administration



Assumptions:

- A file named “general data” may contain name, registration no, address and other information of the student.
- The file “grade data” includes the results of oral exam, written exams, seminars, projects etc.
- Class data additionally comprises the attendances of the students.

While generating the students progress report, the three files must be updated to give the consistent report.

Problems of traditional file systems: File oriented systems have many drawbacks of storing, using and maintaining data.

Redundancy: Repeated occurrences of the same data in different files.

Problems: Wastage of memory, increased management, increased processing time and cost.

Inconsistency: Logical mismatch of data in files Especially caused due to changes.

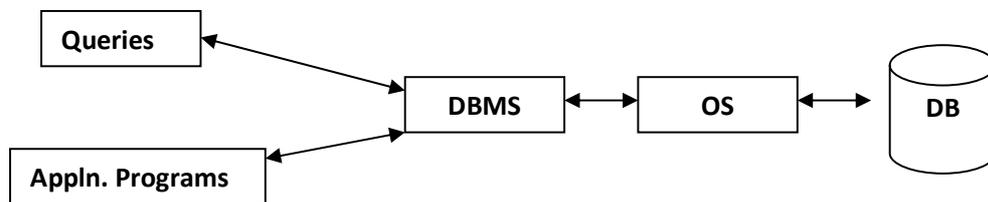
Atomicity: There may be partial updates in files.

Data Program dependence: Changes in program structure required to incorporate new data type. Changes in file structure lead to changes in the application program structure.

Data isolation: Multiple files and formats.

Inflexibility: Analysis of data as well as the realization of new application is problematic.

Data from several files can only be combined.



With high costs. (DBMS as an interface between user and the database)

No central control: Files are used in decentralized manner. Multiple file names and data names may be used by different departments.

Limited data sharing: Difficult to obtain data from several (Atomicity) in compatible files from separate systems.

Security problems: It is difficult to enforce security system.

Requirements of database systems:

Minimal data redundancy: Redundancy is carefully controlled by avoided duplicate copies of the same data by an integrated view on data.

- The controlled redundancy improves performance.

Data independence: Application programmes are independent of representation of data and data storage (Abstract view)

Efficient data Access: DBMS uses a variety of techniques to store and retrieve data efficiently.

- Application of index structures to have faster access.

Concurrent data access: System to allow simultaneous access to the same data by different users.

- Each user gets the impression of exclusively accessing data.
- Concept of transaction for the synchronization of concurrent data access.

Consistency of data: Caused by lack of redundancy. DBMS must ensure consistency of data after each transaction.

Integrity of data: correctness and completeness of data (Semantic aspects), formulation of integrity constraints and integrity rules.

- DBMS checks constraints for each insertion, change and deletion of data.

Data Security: Protection of the database against un authorized access (view on data)

- Access control with authentication and encoding as the possible protection mechanisms.

Database Concepts:

Database: An integrated collection of related files.

- The data must be organized in some logical manner so that it can be processed effectively and efficiently.
- Generally data can be organized in the form of file, records, fields, characters, bytes and site.

File: A group of related records is called a file- student.

Record: A set of related data items (fields) is called a record.

Roll	Name	Age	Sex	Class
102	Manas	17	M	XI

Field: An individual data item within a record is called a field. Ex. Roll, name, age, sex, class of a student.

Character: A single character, can be an alphabet, a digit or some other special character a, x, s, o etc. A database may be a collection of files such as student, course, teacher, fees etc.

DBMS: A set of related programs that controls the creation maintenance and utilization of database of an organization.

Type of database management systems: There are two types

- (i) **Corporate/Entreprise databse:** Designed primarily for applications in corporations, government and enterprises. Characterised by huge volume of data input and output Ex. SQL Server, Oracle, DBL etc.
- (ii) **Personal database:** Designed to be used as learning tools and in small business firms.
Ex. Personal oracle, Microsoft access, foxpro etc.
There are not highly scalable.

DBMS applications”

- Banking – All applications
- Air ticket reservations, schedules
- University registrations, grades, administrations

- Manufacturing, productions, inventory, orders.
- Human resources: records salaries, tax deduction.
- Database touches all aspects of our lives.

A DBMS – Contains information's about an enterprise

- A collection of interrelated data
- A set of programs to access data efficiently
- An environment that is convenient to users and effective to use.

Back up and recovery:

Protection against the consequences of system errors.

Backup: Copies an external storage media (e.g. tapes, CDS)

Recovery: Automatic reconstruction of the last up-to-date and consistent database status with the aid of tapes and a protocol listing the executed changes.

Posing queries: DBMS provides query language, which allows to pose ad hoc queries by using the key board and to get an immediate answer.

Diversified user Interfaces: DBMS provides query languages for occasional users.

- Programming interfaces for implementers/programmers.
- Menu driven interfaces for naïve users.
- Window based Graphic oriented interfaces for graphics users.
- Data definition languages, data manipulation languages.

Flexibility: Change of database structure without change of existing application programme (e.g. extension of a record by a new field, insertion of new collection of data in to database).

- Evaluation of data according to other criteria in an easy way.

Faster development of new application:

- Use of the powerful functions of a DBMS for new applications.

Limitations of DBMS:

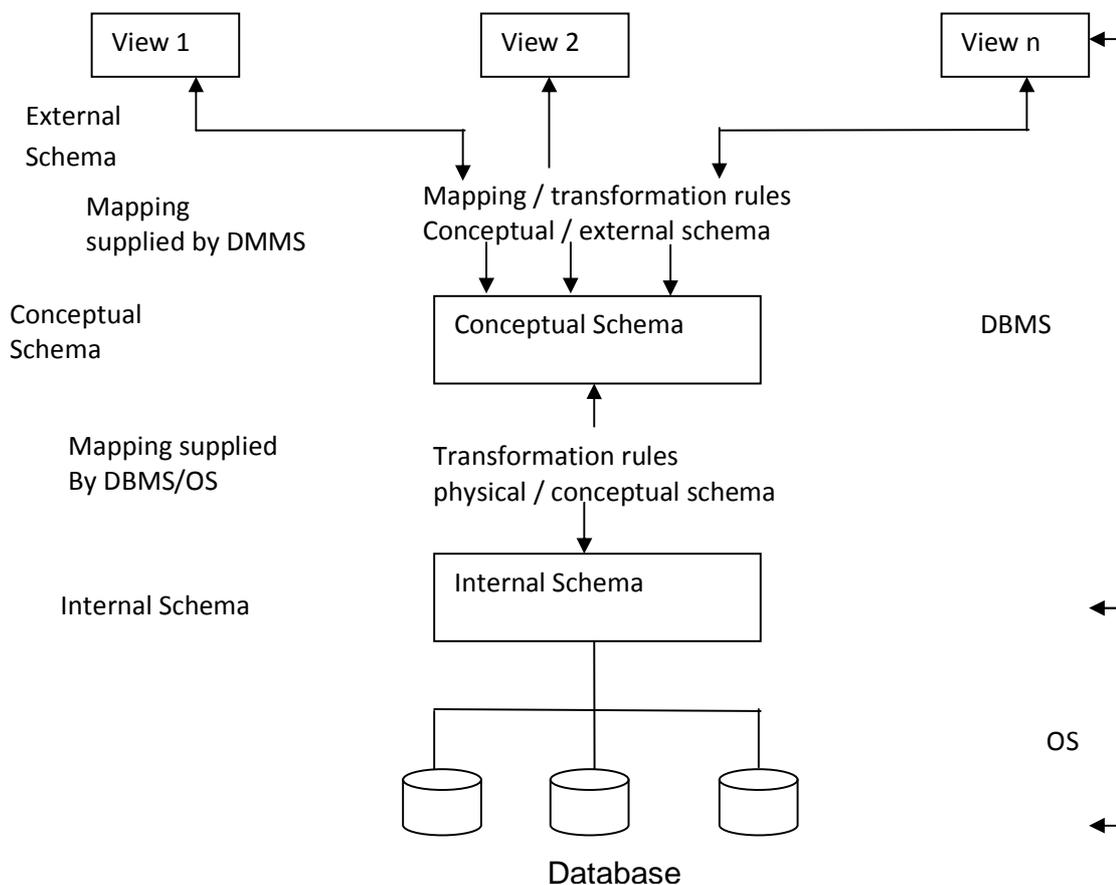
- **Increased complexity:** A multiuser DBMS is extremely complex piece of software.

- **Large storage requirement:** The large complexity and wide functionality makes DBMS complex and large size so it occupies a large storage space (few gigabytes)
- **Specialized manpower is required:** To use and maintain the DBMS
- **Required more installation and management cost.**
- **Requires additional hardware:** depending upon the environment and functionality.
- **Explicit backup and recovery system is required when damage occurs.**

Data Abstraction: The three tier DBMS architecture

DBMS has several abstraction levels such as

- External/view levels- External schema
- Conceptual/logical level- Conceptual schema
- Physical /internal level – Internal schema



- The view at each level is described by a schema
- The Schema describes the way in which the entities of one level of abstraction may be mapped in to the next level.
- The overall design of a database is called database schema.
- The database schema includes information such as
 - Characteristics of data items such as entities & attributes.
 - Logical structure and relationship among data items.
 - Format for storage representations.
 - Integrity parameters such as physically authorization and backup policies.

External Schema:

- The level closest to the user and highest level of abstraction.
- Concerned with the way in which data are required by the user / viewed by the user.
- Describes only part of the data base called view level.
- Each external schema is used by represented by a data model.
- Described the part of database, which is relevant to the user.

Conceptual Schema:

- It described the logical structure of the data base.
- It described the entities, data types, relationships, user operations and constraints.
- It defines a schema based on a data model.
- It gives information about existing data and relationships in database.

Internal Schema:

- The lowest level of abstraction. It describes the physical storage structure of the database.
- The internal schema uses physical data model.
- It describes complete details of the storage and access paths.
- It describes the detailed data structure of the data items.

Database schema and state:

- A schema describes the structural design of the DB.

- A State describes a concrete instances of the DB.

Data Abstraction

- Abstraction is an act of representing the essential features of something without including much details.
- The data abstraction refers to describing essential features of the data item without including the implementation details.
- Users can know only the essential features that they are interested but can not know the internal implementation.

Data Independence:

- Denotes the property that higher levels of abstraction are not influenced by changes in the lower levels.
- The ability to change the schema at one level of a database system without affecting a schema definition in the next higher level is called data independence.
- There are three levels of abstractions so there are two types of data independence.

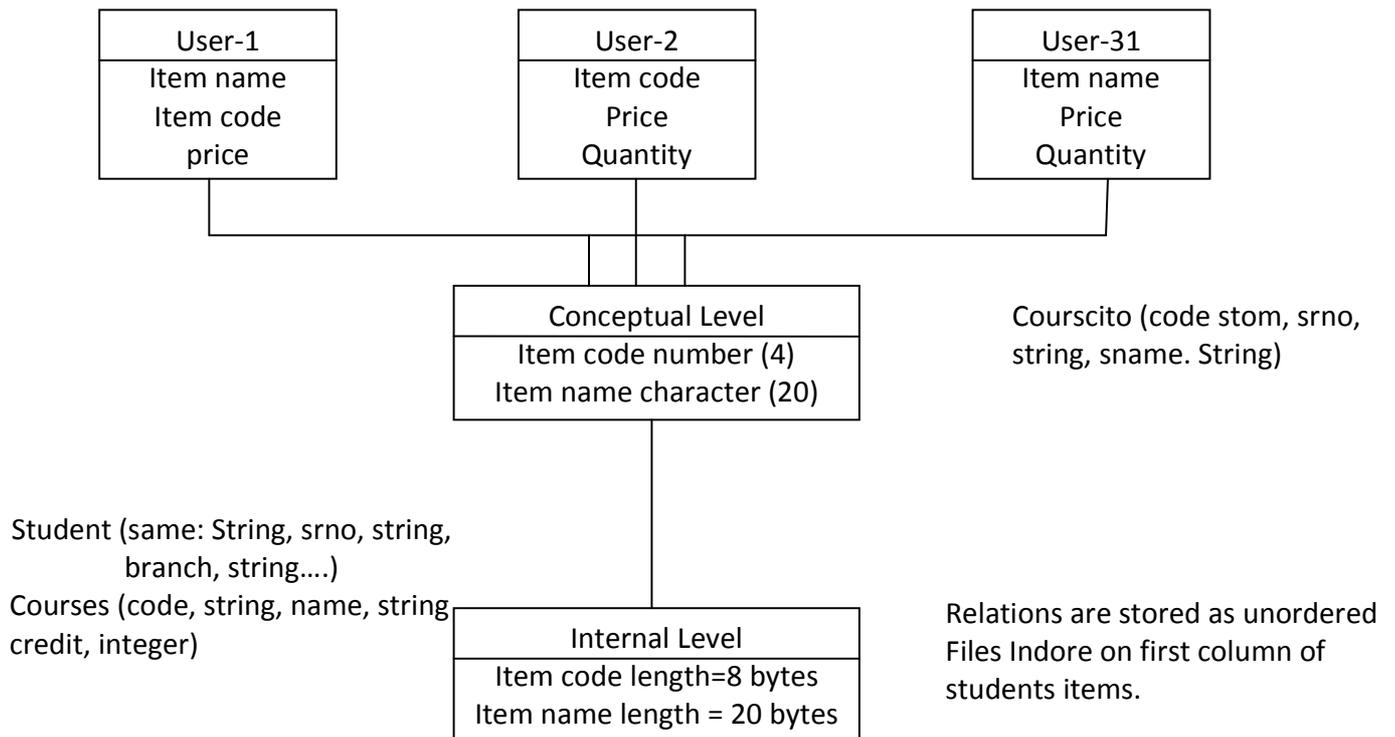
Logical data independence: The ability to change the conceptual schema without affecting the external schema.

- Changes of the conceptual schema (e.g. information about new types of entities, further information about existing entities) do not have impact on external schema (e.g. existing application programs.)
- Example: extension of the class data for an additional Boolean value expressing whether the merits required for a master thesis have been performed.

Physical data independence:

- The capacity to change internal schema without having to change the conceptual schema and thus also not to change the external schema.
- Ex. Creating additional access structure to (implement) improve performance of the retrieval and update.
- Ex. The class data, which are so far stored in an unsorted file, are to be reordered /reorganized in a B-tree to enable efficient access by the “registration number”.

- The three schema architecture can make it easier to achieve true data independence both physical and logical.
- In general the interfaces between various levels and components should be well defined so that changes in some part do not seriously influence others.



(Three level architecture of DBMS by example)

Data base languages: There are two main types of languages.

- (i) Data definition language (DDL)
- (ii) Data manipulation language (DML)

(i) Data Definition language (DDL)

- DBMS provides a facility known as data definition language (DDL)
- DDL is used by DBA to define conceptual and internal schema
- DDL describes the entities, attributes and their relationship.
- Ex. Create table account. (account no char (10), balance integer).
- DDL generates a set of tables stored in a data dictionary
- Data dictionary contains metadata (data about data) such as

- Data base schema
- Data storage and definition language (SDL) which describes storage structure and access methods used.
- Integrity constraints such as
- Domain constraints
- Referential integrity
- Assertions
- Authorization.
- DDL compiler processes the DDL statements in order to identify the description of schema objects / construct.
- DDL statements are used to create / define, update/alter and drop schema objects.

(ii) Data manipulation language (DML)

- The language for accessing and manipulating data
- Data access involves retrieval of a set of data items.
- Data manipulation involves user queries for insertion, deletion and update of records.
- Two types of DML statements.
- Procedural- uses specify what data is required and how to get those data
- Non procedural: user only specifies what data is required .
- DML is a part of the query language SQL
- The DML commands of SQL are
 - Insert: to insert a new record
 - Delete: to delete an existing record
 - Update: to modify an existing record

But in a true 3-tier schema architecture, the other languages that are used by the database are:

- (i) Storage Definition language (SDL):** Used to specify by the internal schema
 - It describes the storage structure and access methods of the database.
- (ii) Transaction control language (TCL)**

- A transaction is a collection of operations that performs a single logical function in a database application.
- TCL provides the transaction management commands such as commit and Roll back.

(iii) View Definition language:

- Used to specify user views and their mapping to conceptual schema.
- Ex. View point.

(iv) Data control languages (DCL)

- Allows a user to grant privileges or right to others to access the database.
- Ex. Grant revoke.

In most of the database, TCL, SDL, VDL and DCL are all part of the DBMS's DDL component. Where DDL is used to define both conceptual and external schema.

Data models:

- A collection of tools for describing
- Data
- Relationship among data
- Their semantics and
- Constraints

Types of data models:

1. Object based logical data models
2. Record based logical data models.
3. Physical data models.
4. Semi structured data models (XML)

1. Object based logical data models

- Object oriented logical data model
- Object relational data models Ex. E.R. model

2. Record based logical data models.

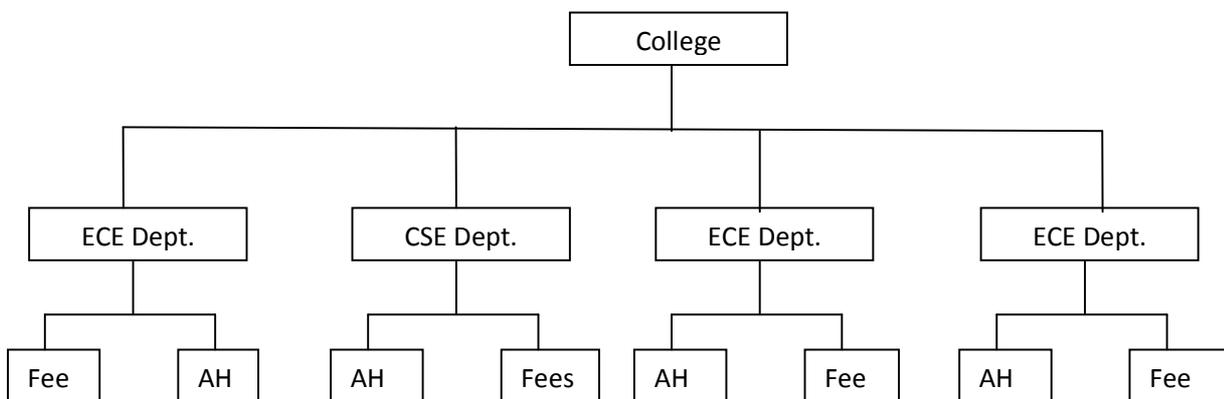
- Network data model
- Hierarchical data model
- Relational data model.

3. **Physical data models.:** Used for internal shcema definition.

Record based data models.

1. Hierarchical data models:

- Data are organized in a ordered tree manner.
- Relationship between records is parent child type.
- The data base in this model is a collection of disjoint trees.
- It is a simple and straight forward model & easy to understand.
- It is used when there is question of hierarchical relationship
- In order to represent link among records pointers are used.
- The relationship among records are physical
- Searching for a record is not easy.
- Represents one to many relationship
- Deletion of an internal node deletion of its leaves.



Advantages.

- Simple and easy to use
- Data with hierarchical relationship can be mapped on this model.
- Suitable for application such as: Employee Dept.

Disadvantages:

- Search for (an element or) a record is difficult.
- Insertion, deletion and updation is difficult.
- Many – many relationship can not be established.
- Data with non hierarchical relationship can not be mapped.
- The hierarchical relationship is maintained using points which requires extra storage.
- Changes in relationship requires changes in entire structure of the database.
- Processing is sequential among branches of the tree so access time is high.

Network data model:

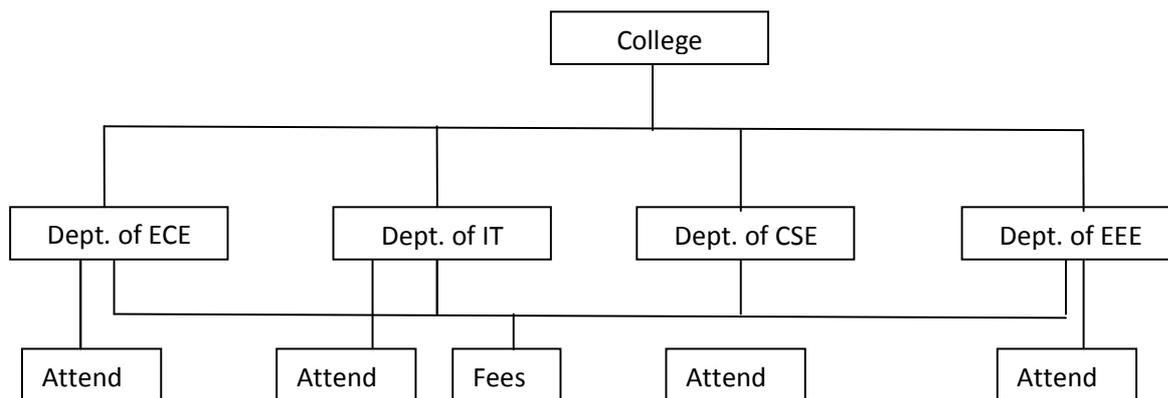
- It was formalized by conference on data system language (codasyl)
- It is an improvement over hierarchical data model.
- It represents many-to-many relationships.
- A child can have multiple parents.
- Relationship between records is maintained by pointers.
- Each record has a pointer field for the record with which it is associated.

Advantages:

- Many to many relationship among records can be implemented .
- Useful for representing such records which are represented in many to many relationships.
- Searching a record is easy since there are multiple access paths .
- No problem of consistency in case of addition deletion of records.

Disadvantages:

- Implementations of records and relationship is complex.
- Storage space requirement is high because of so many pointers.



(Network data model)

Relational data model:

1. In relational data model the relation or a named table is the only construct required to represent an association among the attributes of an entity set and the relationships among different entities.
2. Relationship between record is represented by a relation that contains a key for each record involved in the relation.
3. Many to many relationship can easily be implemented.
4. Relationship implementation is very easy through the use of key and composite key fields.
5. Relational model is useful for representing most of the real world entities and relationships among them.
6. Relational model does not maintain physical connection among records.
7. Data is organized logically in the form of rows and columns.
8. A unique indexed key field is used to search for data element.
9. Data integrity is maintained by the process like normalization .
10. Description of data in terms of this model is called a schema.
11. Schema for relation specifies, its name, name of each field. Ex. Student (sid: Integer, name: string, login: string etc.)

Advantages:

1. Tabular structure is easy to understand simple.

2. Data manipulation is easy.
3. We can apply mathematical operation on tables.
4. Built in query language support such as SQL.
5. Very flexible data organization.

Disadvantages:

1. Size of the data base becomes large.

Example: Oracle, Ingress, Sybase, Unity etc.

Customer table :

Customer ID	Customer Name	Customer Street
-------------	---------------	-----------------

Account No. :

Account No.	Balance
-------------	---------

Properties of a relation/table:

1. The data represented in this model is in the form of two dimensional table called relation.
2. An entity is represented by a tuple in a row.
3. The rows of a table are distinct.
4. Ordering of rows is immaterial.
5. Each column of the table is assigned distinct heading called name of the attribute.
6. In each column data item are of similar type.
7. The ordering of the columns is immaterial.
8. If there are M. columns, it is said to be of degree m.
9. If there are N rows, it is called a N tuple table or cardinality of the table is N.
10. Both the rows and columns can be viewed in any sequence at any time without affecting the information.

Properties of relational data base management system (RDBMS)

- A relational database management is represented by relational data models.
- It uses a collection of tables to represent the data and the relationship among them.
- Each table has multiple no of rows and columns
- Supports the concept of null values.

- Does not require the user to understand its physical implementation.
- Provides information about its contents and structure.

Domain and Integrity Constraints:

Domain Constraints:

- Limit the range of domain an attribute values on.
- Specify uniqueness and null ness of attributes.
- Specify default value of an attribute.

Integrity constraints:

Entity integrity: Every tuple is uniquely identified by a unique non null attribute, primary key i.e. the primary key values can not be null.

Referential integrity: Rows in different tables are correctly related by valid key values (Foreign keys refers to primary keys)

Basic terminologies of relational data model: (RDBMS)

1. **Entity:** A real world object with some properties which can be easily identified is called an entity.
2. **Attributes:** An attribute is a descriptive property or a characteristics of an entity. Ex name, roll marks, etc.
3. **Degree:** The no of columns or attributes associated with a table (or a relation among them) is called degree of the relation.
4. **Cardinality:** The no of rows in a table is called cordinality.
5. **Tuples:** A relation consisting of a number of records represented in row wise information called tuples.
6. **Domain:** The set of possible values that can allotted to an attribute is called domain of that attribute. Domain d is a set of atomic values of an attribute.
7. **Entity set:** A set of entities representing a real world object. Ex. Student, teacher, depositor, player etc.
8. **Weak entity:** An entity set which may not have sufficient attribute to form a primary key then it is called an weak entity.
9. **Strong entity:** An entity set which have a key attribute.
10. **Key:** An attribute that allows us to uniquely identity a record or an entity type.

11. **Super key:** A set of attributes that collectively allows us to identify an entity is an entity set.
12. **Candidate key:** A candidate key is the minimal super key. The super key for which no proper subset is a super key.
13. **Primary key:** Primary key is the one of the candidate key to identify the entity uniquely. The primary key is the principal means of identifying an entity.
14. **Entity type:** The occurrences of an entity set are called entities
15. **Composite key:** A composite key is a candidate key that consists of two or more attributes.
16. **Foreign key:** A foreign key is an attribute (a group of attributes) that is primary key to another relation. A foreign key represents a relationship between two tables.

Entity- Relationship Model:

- This model is introduced by the scientist peter chen in 76.
- E.R model is a generalization of hierarchical and network data model.
- It allows the representation of explicit constraints as well as relationships.
- The relationship between the entity set is represented by named E-R relationships such as 1:m, m:m, etc mapping from one entity set to another.

Terminologies: (Basic concepts)

Entity: The basic object that the E-R model represents is an entity which is a real world object.

Ex. A teacher , A student, A depositor, A borrower etc.

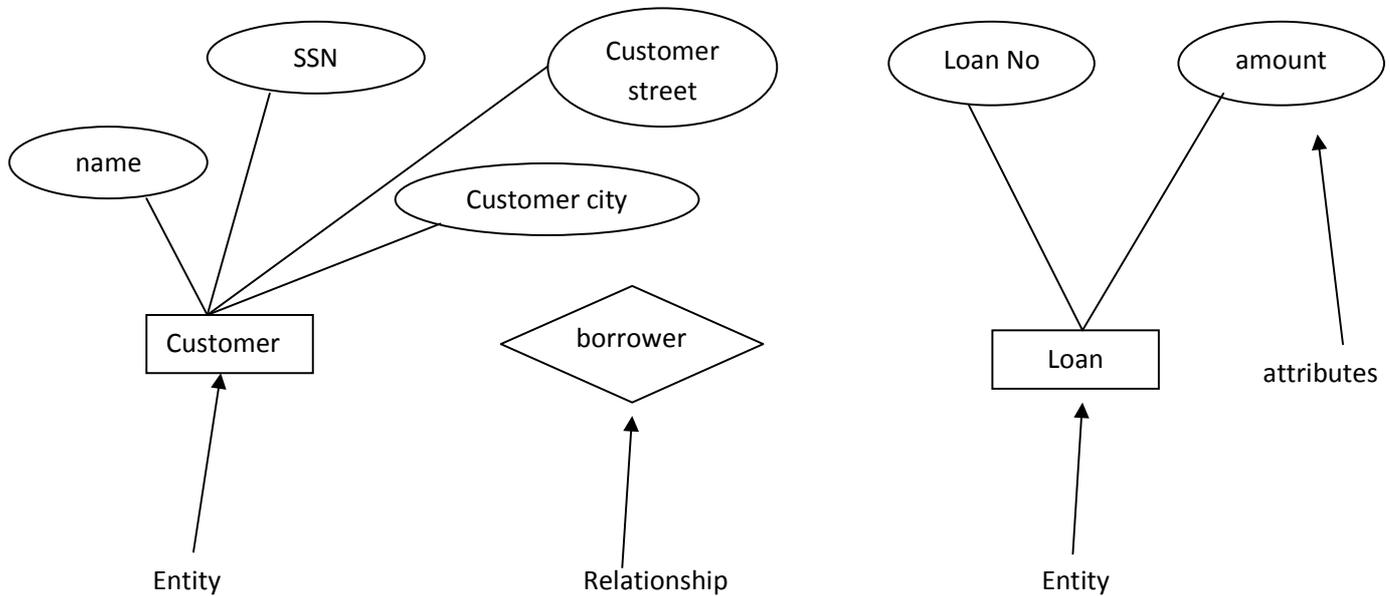
Entity set: A collection of entities is known as an entity set, Ex. Teacher, student etc.

Relationship set: A collection of relations between entities.

- The relations can be either.
- One-to-one
- One to many a many to one
- Many to many

Constraints: On the entities, relationships and attributes.

Ex.



Def.: In this model a database is considered as

A collection of entities and

Relationship among entities

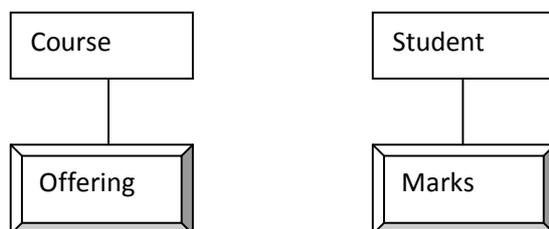
An entity : is an object that exists and distinguishable from other objects. A specific student, a teacher etc.

Types of entities: There are two types of entities

(i). Dependent entity (ii) Independent entity

(i). Dependent entity : An entity whose existence depends on the existence of another entity is called dependent entity.

Ex.



- The existence of the entity marks depends an existence of student.
- Dependent entities are also called weak entities.

(ii) Independent entity: Independent entities are called strong entities or regular or strong entities.

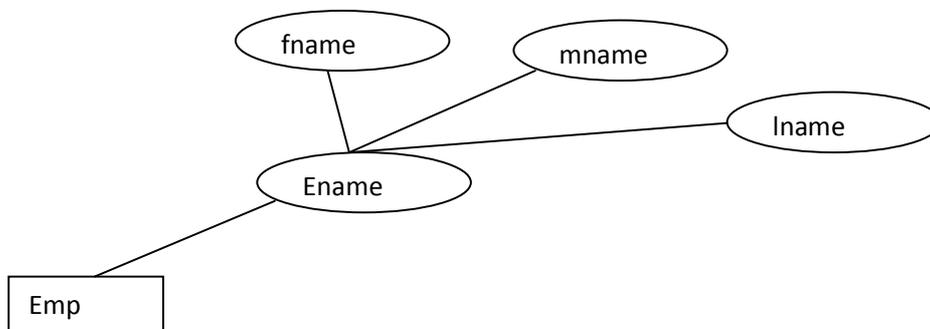
Weak entity types: Entity types that do not have key attributes of their own are called weak entities types.

Attributes:

- The characteristics of an entity are called attributes. Ex. Roll no, name, marks, etc are the attributes of an entity student.
- An attribute is a descriptive property of an entity

Compound attribute: An attribute which can logically have some super attributes is called compound attribute.

Ex.



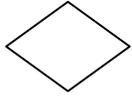
- A compound attribute is the one that actually consist of all other attributes.

Simple attribute: An attribute composed of a single component with an independent existence.

A simple attribute can not be decomposed in to smaller components.

E.R Model

Relationships: A relationship is an association among several entities.



In E.R diagram a relationship is represented by a diamond shape.

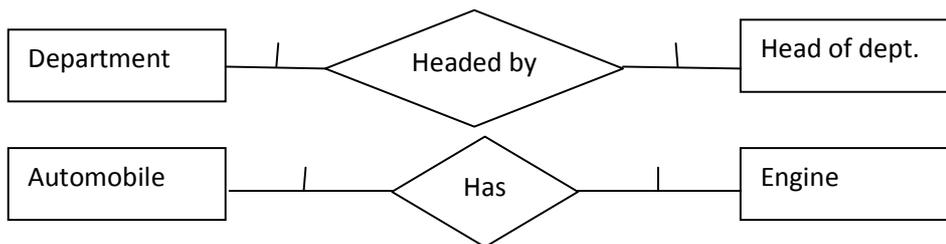
Types of relationships: There are three type of relationships

(Binary)

- (i) 1 : 1 (one-one)
- (ii) 1 : M (one-to many)
- (iii) M : 1 (many to one)
- (iv) M : M (many to many)

(i) **one to one relationship:** One to one (1:1) relationship is an association only between two entities.

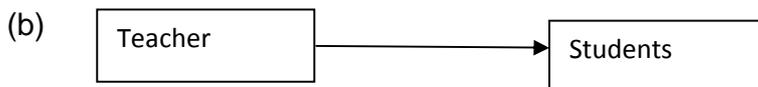
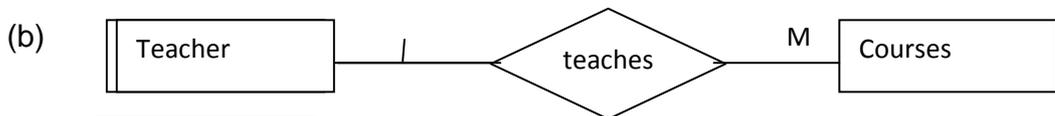
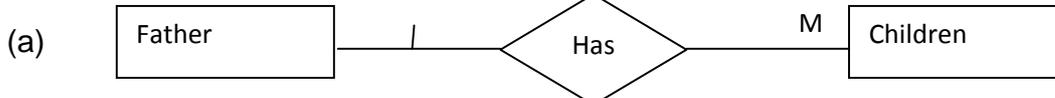
Ex.



- One instance of an entity is associated with only one instance of the other entity set.

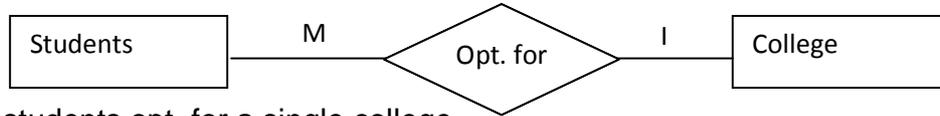
(ii) **One to many relationship:** The relationship is said to be one to many, when one instance of an entity set is related to more than one instance of another entity set.

Ex.



(iii) Many to one relationship: When many instances of an entity set are associated with at most one instance of another entity set.

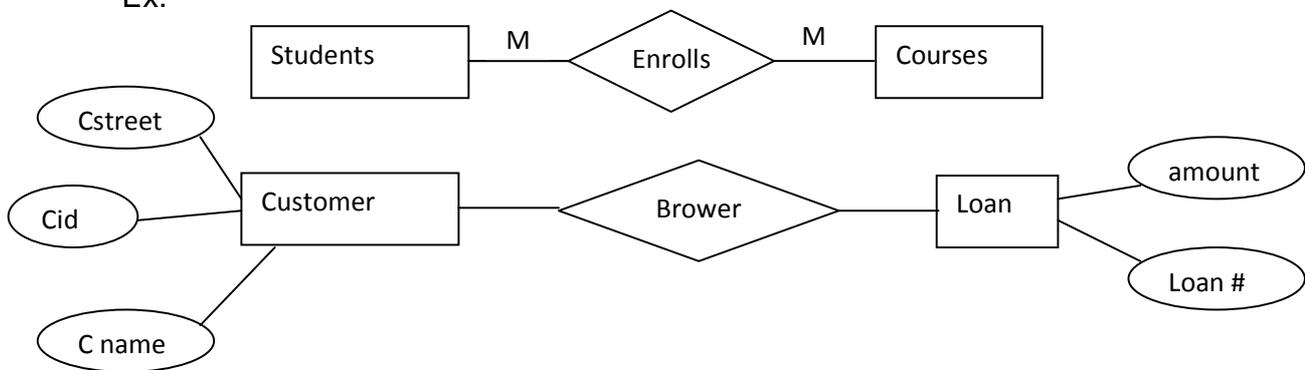
Ex.



Many students opt. for a single college.

(iv) Many to many relationship: A many to many relationship represents association of one entity with many entities of other or many entity of one entity set are associated with one entity of other set.

Ex.



Participation of an entity set in a relationship set: There can be total participation or partial participation.

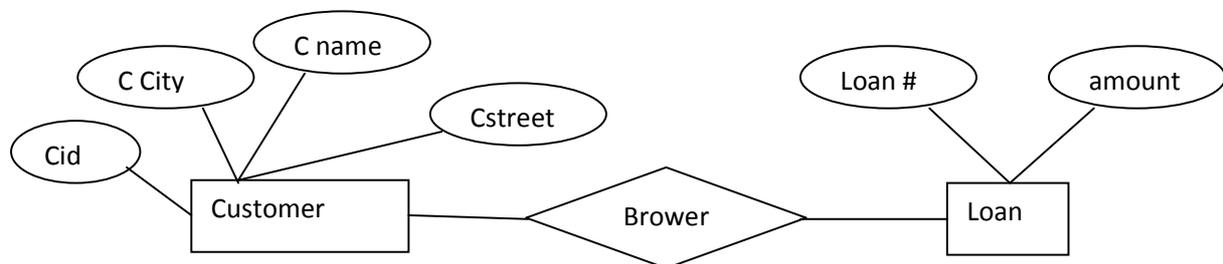
(i) Total participation: Every entity in the entity set participates in at least one relationship in the relationship set.

Eg. Participation of loan in borrower is total.

i.e. Every loan must have a customer associated with it.

(ii) Partial participation: Some entities entities many not participate in any relationship in the relationship set.

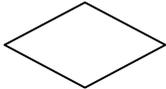
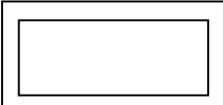
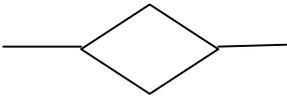
Ex. Participation of customer in borrower is partial.

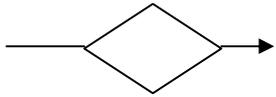
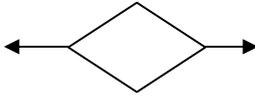
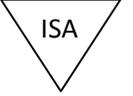
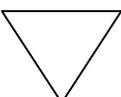
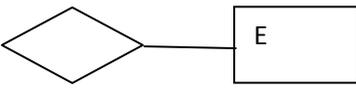
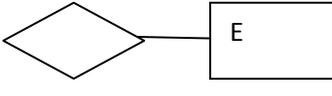
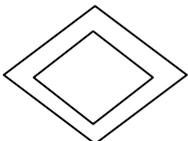


----- represents total participation.

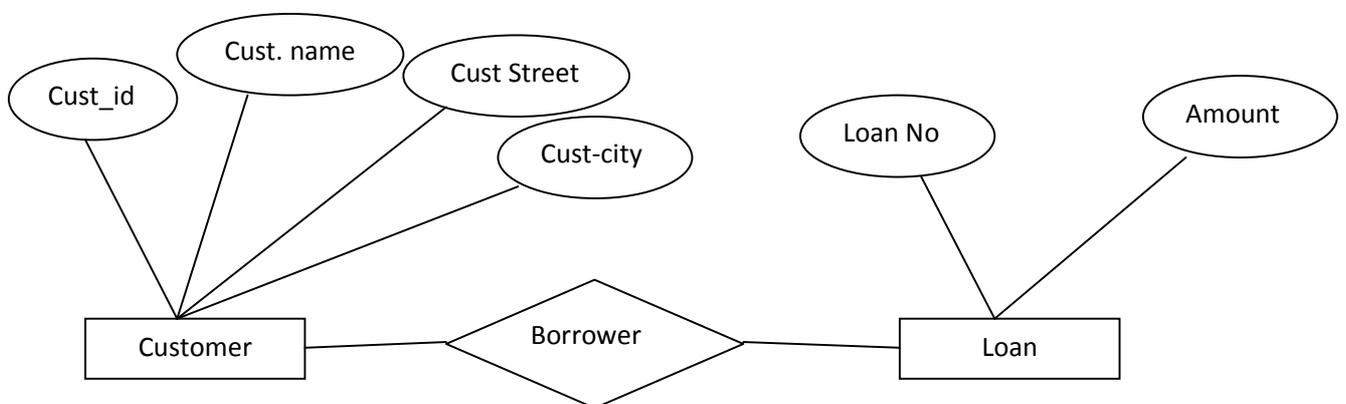
----- represents partial participation.

E.R. Diagrams: Represents the conceptual E-R data model

	Symbols	Description
1	Rectangle 	Represents entity set
2.	Diamond 	Represents relationship set
3.	Ellipses 	Represents attributes.
4.	Double ellipse 	Represents multivalued attributes
5.	Dashed ellipse 	Represents derived attributes
6.	Underlined attribute Ex. 	Represents primary key attributes.
7.	Lines 	Link attributes to relate with entity sets and entity sets to entity sets.
8.	Double rectangle 	Represents weak entity set.
9.	Dashed underline 	Discriminating attribute of weak entity set.
10.		Many to many relationship

11.		Many to one relationship
12.		One to one relationship
13.		Specialization or generalization
14.		Total Generalization
15.		Disjoint generalization
16		Total participation of entity set in relationship
17		Role indicator
18		Cardinality limits
19.		Identifying relationship set for weak entity set.

1. A simple E-R diagram:



2. E-R diagram with composite, multivalued and denved attributes.

3. E-R diagram with relationship set having attribute.

4. E-R diagram with roles.

- Roles : Manager, Worker
- Role labels are optional and used to clarify semantics of the relationship .
- Here manager and worker belong to same entity set.

5. E-R diagram representing cardinality limit.

6. E-R diagram to represent weak entity sets.

- An entity set that does not have a primary key is referred to as a weak entity set.

- The existence of a weak entity set depends on the existence of identifying entity set.
- It must relate to the identifying entity set via a total one to many relationship set from the identifying to weak entity set.
- Identifying relationship depicted from using a double diagram.
- The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existing plus the weak entity sets discriminator .
- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line
- Payment number- discriminator of the payment entity set.
- Primary key of payment- (loan no, payment number)

Note: Primary key of the strong entity set is not explicitly stored with the weak entity set since it is implicit in the identifying relationship.

- **Single valued attribute:** An attribute that holds a single value for each occurrence of an entity type Ex. Empid, emphasis ok.
- **Multivalued attribute:** A multivalued attribute is an attribute that holds multiple values for each occurrence of an entity type Ex. Phone numbers, skill of an employee.
- **Derived attribute:** Can be computed from other attributes . Ex. Age, for given date of birth, year of service, given a date of joining.
- **Identifier attribute:** An attribute that uniquely identifies a particular entity from an entity set.
Ex. Empid, personid , regn. No etc.

Super type and sub types.

Here employee is the super type, it can be of two types salaried employee or hourly employees. So hourly employees and salaried employee are sub types.

Primary key of the super type is the foreign key of sub types.

Ternary relationships.

E. R diagram with ternary relationship.

E-R diagram with ternary relationship.

N. ary relationship: In case of n-ary relationship along a single relationship diagram with n connections, one to each entity represents some associations among n entities.

Aggregation

- Aggregation is the process of compiling information on an object thereby abstracting a higher level object.
- Aggregation is the part whole of a part of relationship in which objects representing the components of something are associated with an object representing the entire assembly.

Ex.

- Aggregation is an abstraction (such as the one just through which relationships are treated as higher level entities).
- To eliminate redundancy via aggregation.
 - Treat relationship as an abstract entity
 - Allows relationship between relationships.
 - Abstraction of relationship into new entity.
- Without introducing redundancy, the following diagram represents
 - an employee works on a particular job on a branch
 - an employee, branch, job combination may have an associated manager.

(E-R diagram with aggregation)

Categorization:

- Categorization is the process of modeling of a single sub type (sub class) with a relationship that involves more than one distinct super type (or super class)

Aggregation – contd. Another form of aggregation is abstracting a relationship between objects and viewing the relationship as an object. Ex. Enrollment relationship student and course could be viewed as registration.

Generalization:

- A bottom up design process – combines a number of lower level entity sets that share the common features into a higher level entity set.
- Generalization is used to emphasize the similarities among lower level entity sets and to hide the differences.
- It is called as “a kind of” or is a relationship.
- In terms of ER diagram, generalization is pictured through a triangle component labeled as a relationship.
- Generalization and specialization are inverse of each other .
- Generalization and specialization are interchangeably used.
- Generalization and specialization are used to represent in the E-R diagram in the same way.
- The ISA relationship is referred to as super class sub class relationship.

Extended E-R features:

- The EE-R model consist of all the concepts of E-R model together with the following additional concepts.
- Specialization.
- Generalization
- Aggregation
- Categorization

Specialization: Specialization is the process of identifying subsets of an entity set (the super class or super type) that share some distinguishing characteristics.

- The higher level entity set is called the super class.
- The lower level entity set as a result of specialization is called sub class.
- The supper class includes the properties of the sub classes.
- The sub classes are defined on the basis of some distinguishing characteristics.

Attribute in heritage: a lower level entity set inherits all the attributes and relationship participation of the higher level entity set to which it is linked.

Specialization: A top down design process. Depicted by a triangle labeled by a ISA relationship.

E-R Model

Super type and sub type: The sub type / super type relationship is used when either or both of the following conditions are satisfied.

(i) There are attributes

That are common to some (but not all) of the instances of the entity type.

(ii) The instances of a sub type participate in a relationship unique to that sub type

Advantages of using super type / sub type / class.

The concept of introducing super class and sub class in to ER model provides the following enhanced feature.

- It avoids the need to described similar concepts more than once, thus saving time for the data modeling person.
- It results in more reliable and better looking E-R diagram.
- Super class and sub class relationships and more semantic content and information to the design in a concise form.

Specifying constraints on Specialization and Generalization.

There are mainly two types of constraints.

- Participation constraints.
- Disjoint constraints.

Participation constraints. : Two types of participations constraints (a) total participation (b) partial participation.

- (a) **Total participation constraint:** Every number in the super type must participate as a member of some sub class, it is represented by double line.
- (b) **Partial participation Constraint:** A member of super type need not belong to any of its sub classes of a specialization or generalization.

Ex. An employee may not be a union of club member.

(b) Partial or optional constraints.

Disjoint constraints.: The constraint is only applied only when a super type has more than one sub type. The sub types are disjoint so entity occurrence can be at most one place of sub types.

CODD'S Rules: There are twelve rules formulated by Ted codd for RDBMS in 1970.

A database is said to be an RDBMS if it satisfies all the twelve rules.

1. Information presentation: All information should be explicitly and logically represented by entering the data value in the form of tables.

2. Guaranteed access rule: Every item of data must be logically addressable by resorting to a combination of table name, primary key value and column name.

3. Systematic treatment of null values: Null values can be supplied if the information is missing a inapplicable.

4. Database description rule: An RDBMS must have a data dictionary which describes the database and consist of tables and views.

5. The comprehensive sublanguage rules: There must be at least one language supporting statements in character strings for

- Data definition (Language)
- Integrity constraints
- View definition (Language)
- Authorization
- Data manipulation
- Transaction boundaries

6. The view updating rule: All views that can be updated in theory can also be updated by the system.

7. The insert and update rule: Apart from data retrieval an RDBMS must be capable of updating and deleting data as a relation set.

8. The physical independence rule: The changes in storage structure and access methods should not affect the application programs through which users access the database.

9. The logical data independence rules: Application programs must be independent of changes made to the base table.

10. Integrity rules: There are two types of integrity rules.

(a) Integrity rule 1 (Entity integrity): If the attribute A of a relation R is a prime attribute of R, then A can not accept null values.

(b) Integrity rule 2 (Referential Integrity): Attributes of a relation having domains that are those of primary key of another relation.

If relation (A) may contain references to another relation (s) then the relations R and S may not be distinct.

11. Distribution Rule: An RDBMS must have distribution independence. It should allow the database to be distributed across multiple computers even though they are having heterogeneous platforms.

12. No sub version rule: If an RDBMS supports lower (version) level language that permits for example a row at a time processing. Then this language must not be able to by pass any integrity rules or constraints of the relational language.

Thus not only an RDBMS be governed by a set of relational rules but those rules (or constraints of the relational language must be its primary laws.

Note: An RDBMS package should satisfy at least six of the 12 rules to be accepted as an RDBMS software package.

The is no RDBMS package commercially available that satisfies all the 12 rules given by ted codd.

Introduction to object oriented database system.

Research to model and process complex data has gone in two way

- (a) Extending the functionality of RDBMS
- (b) Developing and implementing OODBMS that is based on object oriented programming paradigm.

Applications of OODBMS:

Multimedia

CAD

Office automation etc.

The next generation database systems

Object orient DBMS (OODBMS)

Knowledge database Management system (KDBMS)

New database applications:

1. **Design data bases:** Designing CAD/CAM/CASE Systems.
2. **Multimedia databases:** Designing systems to support data storage and access which are in the form of text, image, video, voice, graphs, figures etc.
3. **Knowledge bases:** AI and Expert systems represent information as facts and rules that can be collectively viewed as knowledge base.

Object oriented Database Management System.

- Object oriented technology includes object oriented programming languages c++, small talk, object oriented data base system, object oriented user interfaces (Macintosh & MS windows systems and so on.)
- An object oriented technology makes available to the users facilities that are based on object oriented concepts

Relational Query Language.

Relational query languages are of two types.

1. Procedural query language e.g. Relational algebra.
2. Non- procedural language e.g. Relational calculus
3. Pure Languages e.g. (i) Relational Algebra (ii) Tuple relational

Relational Algebra: Calculus (iii) Domain relational calculus.

- Relational databases are usually defined by relational algebra.
- It is a procedural language.
- It consists of set of operations that take one or two relations as input and produces a new relation as output / result.

Relational Algebra is:

- The formal description of how a relational database operates.
- An interface to the data stored in the database it self.
- The mathematics which underpin SQL operations.

The fundamental relational algebra operations are.

(i) Simple operations

- (i) Union – \cup (cup) (ii) Intersection – \cap (Cap)
(iii) Difference – $-$ (minus) (iv) Cartesian product – \times (times)

(ii) Special relational operations

- (i) Selection – σ (sigma) (ii) Projection – π (pie)
(iii) Division () (iv) Join – \bowtie (bow-tie)
(v) Rename – ρ (Rho)

(i) Simple operations

(i) Union: Union of two relations R_1 and R_2 is a relation R_3 that includes all the tuples that are either in R_1 or in R_2 or in both R_1 and R_2 and duplicate tuples are eliminated.

Requirements for set operations (Union, difference, intersection)

Two relations are said to be union compatible

If (a) They have same no of attributes

(b) the domain of each attribute in column order is the same as in both R_1 and R_2 .

Mathematically, two relations $R_1 (A_1 A_2 \dots A_n)$ and $R_2 (B_1, B_2 \dots B_m)$ are union compatible if.

(a) $a=n$

(b) Domain $(A_i) = \text{domain } (B_i)$ for $1 \leq i \leq n$

Notation: $R_1 \cup R_2$

Definition: $R_1 \cup R_2 = \{ t \mid t \in R_1 \text{ or } t \in R_2 \}$

Ex.

A	B
α	1
β	2
γ	3
δ	2

The general form of Union operation is given by

Union Tablename1, table name2
 In to table name3, (out put relation)

In SQL the relational Union operation is expressed as

Select from relation1
 Union
 Select from relation 2

Relational calculus form of union is given by

π Customer name (depositor) \cup π customer name (borrower)

(ii) Intersection: Intersection of two relations R_1 and R_2 is a relation that includes all tuples that are both in R_1 and R_2 . The resulting relation might have the same attribute names as the first or the second relation.

Relation R_1, R_2 .

Notation $R_1 \cap R_2$.

Definition:

$$R_1 \cap R_2 = \{ R | t \in R_1 \text{ and } t \in R_2 \}$$

- The general form of INTERSECTION operation is given as

INTERSECTION table-name 1 table-name 2
Into Result (output relation 1 table)

- In SQL the relational INTERSECTION OPERATION is expressed as

```
SELECT From relation 1  
INTERSECTION  
SELECT from relation 2
```

- In relational calculus the intersection can be used as

Ex. Find all customers with an account as well as loan

Π Customer name (Depositor) \cap Π customer name (borrower)

(iii) Difference operation: If two relations R_1 and R_2 are union compatible then they can participate in difference operation.

The resultant relation R_3 is having the same attributes as that of R_1 and R_2 .

The tuples of R_3 are belonging to R_1 which do not belong to R_2

i.e. $R_3 = R_1 - R_2 = \{ R | t \in R_1 \text{ and } t \notin R_2 \}$

- The general form of DIFFERENCE operation is given as

DIFFERENCE table-name 1 table-name 2
Into Result (output relation 1 table)

- In SQL the relational difference operation is expressed as

```
SELECT From relation 1  
MINUS  
SELECT from relation 2
```

(iv) Cartesian product: (cross product)

- If two relations R_1 and R_2 do not have common attributes names then the relations R_1 and R_2 are called product compatible.
- If the two relations are product compatible then they can participate in Cartesian product.
- The resultant relation is a concatenation of each tuple of R_1 with each tuple of R_2 .
- So the resultant relation R_3 will have the attributes which are attributes of both R_1 and each tuple of R_1 appear with all tuples in R_2 .

Relation R_1, R_2 .

Notation $R_1 \times R_2$.

Definition: $R_1 \times R_2$

$\{ (x,y) \mid x \in R_1 \text{ and } Y \in R_2 \}$

$R_1 \times R_2$

- The Cartesian product of two relations $R_1 (A_1 A_2 \dots A_n)$ with cardinality i and $R_2 (B_1 B_2 B_3 \dots B_m)$ with cardinality j is a relation R_3 with degree $k=n+m$, cardinality $i \times j$ and attributes $(A_1, A_2 \dots \dots \dots A_n, B_1, B_2 \dots \dots B_m)$
- Cartesian product operation is costly and its practical use is limited.
- The general form of Cartesian product is given by `CARTESIAN PRODUCT table name 1, table name 2 into Result (opp. relation)`
- In SQL the Cartesian product is expressed as.
`CARTESIAN PRODUCT Relation 1, Relation 2`
`Into Relation 3`
 Or

$R_3 = \text{CARTESIAN PRODUCT } R_1, R_2$

Note:

1. Cartesian product is possible if attributes of R_1 and R_2 are disjoint ($R_1 \cap R_2 = \emptyset$)
2. If the attributes of R_1 and R_2 are not disjoint then renaming must be used.

Composition of operations:

We can build relations using multiple relational operations.

Ex. $\sigma_A = C (R_1 \times R_2)$

A	B	C	D	E
α	1	α	2	A
β	2	β	3	b

(v) Rename operation:

- Allows us to refer a relation in more than one name.
- The rename operator returns an existing relation under a new name
- Example $f_X (E)$ returns the expression E under the new name X.
- If the relation algebra expression E has arity n then Example $f_{X A_1, A_2 \dots A_n} (E)$ returns the results of expression E under the name X and with the attributes renamed to A_1, A_2, \dots, A_n .

Equivalences:

The same relational expression can be written in many different ways.

The order in which the tuples appear in relations is never significant.

$$A \times B \cong B \times A$$

$$A \cap B \cong B \cap A$$

$$A \cup B \cong B \cup A$$

$$A - B \neq B - A$$

$$\sigma_{C1} (\sigma_{C2} (A)) \cong \sigma_{C2} (\sigma_{C1} (A))$$

$$\Pi_{a_1} (A) \cong \Pi_{a_1} (\Pi_{a_1, \dots, a_n} (A))$$

- The equivalent expression always give same result, but when the query is submitted to the DBMS, its query optimizer tries to find the most efficient equivalent expression before evaluating it.

Division operation :

The division of two relations $R_1 (A_1, A_2, \dots, A_n)$ with cardinality i and $R_2 (B_1, B_2, \dots, B_m)$ with cardinality j is a relation R_3 with degree $K=n-m$, cardinality $i \div j$ and attributes that satisfy division condition.

Let $A \leftarrow$ set of attributes $A_1, A_2, A_3, \dots, A_n$.
 $B \leftarrow$ set of attributes $B_1, B_2, B_3, \dots, B_m$.
 $C \leftarrow$ set of attributes $C_1, C_2, C_3, \dots, C_{n-m}$.
 $C = A - B, B \leq A$

C is the set of attributes of R_1 that are not in R_2

The division operator can be defined as a relation over the attributes C that consist of the tuples from first relation R_1 that match the combination of every tuple in another relation R_2 .

- The general form of division operation is
 Division relation name2, relation name 1
 Into Result (output relation)
- In SQL the division operation may be expressed as:
 Select from relation1.
 Division
 Select from relation2

Ex.

Summary of operations:

1. Select
2. Project
3. Union
4. Intersection
5. Difference
6. Division
7. Join

Special operations

Selection: The select operator is used to extract (select) tuples from a specified relation.

- Selection condition can be applied on attributes of a relation.
- The selection condition may contain the comparison operators =, <, >, < >, <=, >=
- The selection condition may contain Boolean operators AND, OR, NOT, etc.

The general form of selection operation is

< selection condition > (< relation name >)

Select table (or relation) name

< where predicates >

In to results (out put relation)

- The result of selection has same attributes as the relation specified in relation name

- Select is used to obtain a subset of the tuples of a relation that satisfy a select condition.

Relation:

A	B	C	D
α	α	1	7
α	β	5	7
β	β	5	7
β	β	12	3
β	β	13	10

A	B	C	D
α	α	1	7
β	β	23	10

Notation $\sigma_P(r)$ $P \leftarrow$ selection predicate

Defined as $\sigma_P(r) = \{ t \mid t \in r \text{ and } P(t) \}$

Where p is a formula in propositional calculus consisting of terms connected by \wedge, \vee, \neg etc.

Each term is one of

$\langle \text{attribute} \rangle$ or $\langle \text{constant} \rangle$ where op is one of $=, <, >, \leq, \geq, =$ etc.

Example: $\sigma_{(\text{salary} > 2000)}(\text{EMP})$

In SQL, the query will be represented as:

Select From EMP

Where Salary > 20000,

If this query is applied to the relation EMP as below.

EMP

EMP No,	ENAME	JOB	MGR	SALARY

7782	CLERK	MANAGER	7839	24,500
7839	KING	PRESIDENT		50,000
7934	MILLER	CLERK	7782	13,000

The resultant relation will be

EMP No,	ENAME	JOB	MGR	SALARY
7782	CLERK	MANAGER	7839	24,500
7839	KING	PRESIDENT		50,000

PROJECTION :

- Projection operation is used to extract entire columns or attributes.
- It constructs a new relation by selecting only specified attributes of the existing relation and eliminating duplicate tuples in the newly formed relation.
- Degree of the new relation is same as the no of attributes. The general format of representation of Projection operation

Π <attribute list> (<relation name>)

- The project operation is used to select a subset of the attributes of a relation by specifying the names of the required attributes.

Ex. EMP

EMP No,	ENAME	JOB	MGR
7369	SMITH	CLERK	7902
7521	JOHNS	SALESMAN	7698
7698	BLACKE	MANAGER	7839
7782	CLERK	MANAGER	7839

After using the relational expression π (job, mgr) EMP on the above relation it would produce.

JOB	MGR
-----	-----

CLERK	7905
SALESMAN	7698
MANAGER	7839

The general form of projection operation is given by.

Project table (relation) name on (or over) column names

Into Result (output relation)

In SQL, the project operation is expressed as.

Select distinct attribute data.

From relation (or table)

The composition of select and project operations:

Select and project can be combined together. For example if the relational expression.

Π (job, mgr) (σ (salary > 20000) EMP) is applied on the relation.

EMP No,	ENAME	JOB	MGR	SALARY
7782	CLERK	MANAGER	7839	24,500
7839	KING	PRESIDENT		50,000
7934	MILLER	CLERK	7782	13,000

It will result the following information.

JOB	MGR
MANAGER	7839
PRESIDENT	

JOIN OPERATION

Notation $R_1 \& R_2$

Let t and s are the pair of tuples from R_1 and R_2 . If t and s have the same value on each of the attributes in $R_1 \cap R_2$ and tuple t to the result, where

t^1 has the same value as t and

t^1 has the same value as s

- Join is a method of combining two or more relations in a single relation.
- It brings together rows (tuples) from different relations (table) based on the truth of some specified condition.
- It requires choosing attributes to match tuples in the relations .
- Tuples in different relations but with the same value of matching attributes are combined into a single tuple in the output (result) relation.
- Joining is the most useful of all the relational algebra operations.
- The general form of joining operation is

Join relation name 1

With relation name 2

On (or over) domain name

Into result (output relation)

- In SQL the join operation is expressed as

Select attributes data

From outer table (relation), inner table (relation) name <where predicate(s) >

Ex. Join Warehouse

With items

On whid

Into R_6

Or $R_6 = \text{Join Warehouse, Items over WHID}$

R_1

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

R_2

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

$R_1 \ \& \ R_2$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
α	2	β	b	δ

Example:

$R_1 = (A,B,C,D)$

$R_2 = (E,B,D)$

$R_1 \ \& \ R_2 = (A,B,C,D, E,B,D)$

Def: $R_1 \ \& \ R_2 = \pi \ r.A, r.B, r.C, r.D, s.E \ (6r.B = S.B \wedge r.D = S.D \ (r \times s))$

Example Query

Find the name of all customers who have a loan at the bank and the loan amount.

$\Pi \ c \ name, \ Loanno, \ amount \ (borrower \ \& \ loan)$

In SQL:

Select district cname, loanno, amount

From borrower natural join loan

Select b.c name l.loanno, l. amount

From borrower e join loan L using l.loanno.

Types of join operation

- (i) Equi join
- (ii) Theta join
- (iii) Natural join
- (iv) Outer join

(i) Equijoin: When two relations are joined together using equality of values in one or more attributes they make an equijoin.

In equijoin the comparison operation (=) is only used. The result of an equijoin will always have one or more pairs of attributes that have identical values in every tuple.

Write a query to retrieve empno, empname, salary, mgrno and corresponding manager name.

```
Select e.name, e empno, e.sal, emgr, m.ename
```

```
From emp e, emp m
```

```
Where e.mgr = m.empno;
```

(ii) Theta Join (Q) : It is denoted by $R_1 \bowtie_Q R_2$, $i Q j$

- Where i and j are columns of R and S respectively and Q represents any of the comparison operators ($=, <, \leq, \geq, \neq$).
- The general join condition is of the form $\langle \text{condition} \rangle \text{ AND } \langle \text{condition} \rangle \dots$. Here each condition is of the form $A_i Q B_j$.
- Where A_i is an attribute of relation R_1 and B_j is an attribute of relation R_2 .
- A join operation with such a general join condition is called theta join.

(iii) ...

(iv) Outer Join: An extension of join operation that avoids loss of information.

- It computes the join operation and adds tuples from one relation that do not match tuples in the other relation to the result of the join.
- An outer join retains the information that would have been lost from the relation (table), replacing missing data with nulls.

There are three forms of outer join

- (i) Left outer join
- (ii) Right outer join
- (iii) Full outer join

(i) Left outer join: It is represented by $R_1 \ltimes R_2$

It retains data from the left hand table

(ii) Right outer join: It is represented by R1 & R2

It retains data from right hand table

(iii) Full outer join: It is represented by R1 & R2

It retains data from both the tables.

Example:

R₁

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
β	2	β	b

R₂

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	β
3	b	ε

The left outer join retains the tuple from the left hand table even when there is not matching value in other relation.

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
β	2	β	b	β
β	2	γ	a	null
	4		b	

γ		β		null
----------	--	---------	--	------

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
β	2	β	b	β
null	3	null	a	β
null	3	null	b	ϵ

(i) Left outer Join

(ii) (Right outer join)

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
β	2	β	b	β
null	3	null	a	β
null	3	null	b	ϵ

(iii) (Full outer join)

General form of outer join:

Outer join outertable name, innertablename

On (or over) domain name

Into result (output relation)

In SQL, the join operation is expressed as

SELECT attribute (s) data

From outer table (relation), inner table (relation) name

< Where predicate (s) >

Example of Queries: With Banking operation

The Banking operation system consists of following entities.

Branch (bname, bcity, assets)

Customer (cname, cstreet, ccity)

Account (accountno, bname, balance)

Loan (loanno, bname, amount)

Depositor (cname, accountno)

Borrower (cname, loanno)

1. Find loans over Rs.12,000
 - (i) Б amount > 12,000 (loan)
 - (ii) SELECT * from loan
Where amount > 12000;
2. Find the loan number for each loan of an amount greater than Rs.12,000.
 - (i) П loanno (Б amount > 12,000 (loan))
 - (ii) SELECT Distinct loanno from loan
Where amount > 12,000;
3. Find the names of all customers who have a loan, an account , or both from the bank.
 - (i) П customer name (borrower) U П customer – name (depositor)
П iname (borrower) U П name (depositor)
 - (ii) SELECT Distinct name from borrower UNION.
SELECT Distinct name from depositor;
4. Find the names of the all customers who have loan at chandaka branch.
 - (i) П c name (Бb name = "CHANDAKA"
(Б borrower. Loanno =loan.loan.no (borrower X loan)
5. Find the names of all customers who have a loan at the CHANDAKA branch but do not have an account at any branch of the bank.
 - (ii) П name (Бb name = "CHANDAKA"
(Б borrower. Loanno =loan.loan.no (borrower X loan)- П c name (depositor)
6. Find names of the customer who have a loan at "CHANDAKA" branch.

- (i) Π c name (Bb name = "CHANDAKA")
 (B borrower. Loanno =loan.loan.no (borrower X loan)
 Or
 Π c name (Bloan. Loan no = borrower. loanno
 (B bname= CHANDAKA (Loan) X borrower)

RELATIONAL DATABASE DESIGN (RDD)

- A database must be designed before its creation and use.
- The database design must confirm to common standards
 Such as : Memory saving, faster access, easy maintenance portability, facilities for future enhancements, performance and cost efficiency etc.
- The final logical design of a database needs to give the optimal performance along with a database integrity.
- This can be achieved by a procedure called "Normalization"
- The data base must be in a state of fully normalized form.

NORMALIZATION:

Def: Normalization is a body of rules addressing analysis and conversion of data structures into relations that exhibit more desirable properties of internal consistency, minimal redundancy and maximum stability.

Def: Normalization is the process of decomposing a set of relations with anomalies to produce smaller and well structured relations that contain minimum or no redundancy.

Def: Normalization is the process of reducing /decomposing a relation into a set of small relations free from data redundancy and ensuring data integrity.

Def: Normalization is a procedure of successive reduction of a given collection of relational schemas based on their FDS and primary keys to achieve desirable form of minimized redundancy, minimized insertion, deletion and minimized update anomalies.

OBJECTIVES

- The basic objective of normalization is to reduce the data redundancy and prevent loss of information or avoid inconsistencies when the data base is altered.
- The types of alterations needed for relations are:

1. Insertion of new data values to a relation.
 2. Deletion of tuple or a row in a relation.
 3. Updating or changing the value of an attribute in a tuple.
- The normalized relation does not include spurious information when the original schema is reconstructed.
 - Preserves dependencies present in the original database /shed

About Normalization:

- The process of normalization was proposed by EF Codd.
- Normalization is a bottom-up design technique for relational database.
- This technique is useful in the circumstances such as .
 - A different method of checking the properties of design arrived at through EER modeling.
 - A technique of reverse engineering a design from an existing undocumented implementation.

Key terms:

Data redundancy: If data in the database exist in two or more places / locations i.e. direct redundancy or if data can be calculated from other data items i.e. indirect data redundancy. Then the data base is said to be redundant.

- Data should only be stored once and avoid storing data that can be calculated from other data.
- During normalization redundancy is removed of course not at the cost of integrity (rules) of the database.

Data Integrity: All the data in the database are consistent and satisfies all integrity constraints.

INTEGRITY CONSTRAINTS = An integrity constraint is a rule that restricts the values that may be present in the database.

- The relational data model includes constraints that are used to verify the validity or the data as well as adding meaningful structure to it. Value in each row.

Entity Integrity: The primary key must have unique not null.

Referential Integrity: Every foreign key must either be null or its values must be the actual value of a key in another relation.

Successive normal forms of a relation:

- Converting the relation to 1NF is the essential step.
- There are successive higher normal forms such as 2NF, 3NF, BCNF, 4NF, 5NF.
- Each normal form is an improvement over the earlier form.
- A higher normal form relation is a subset of lower normal form.

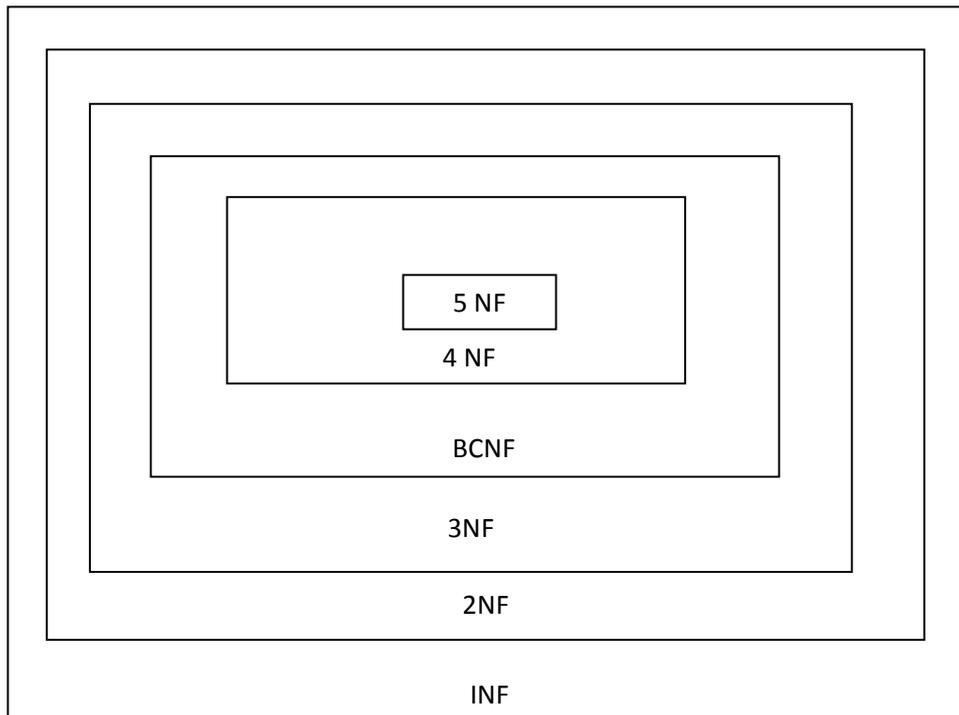


Fig: Successive normal forms

- The higher normalization steps are based on three important steps.
 - (i) Dependence among attributes in a relation.
 - (ii) Identification of an attribute or a set of attributes as the key of a relation.
 - (iii) Multivalued dependency between attributes.

Normalization of data or relations is the process of analyzing the given relation schema based on their FDS and primary keys to achieve the desirable properties or

- (i) Minimizing redundancy
- (ii) Minimizing insertion, deletion and updation anomalies.

Normalization procedure provides the database designers.

- A formal framework for analyzing the relation based on their keys and functional dependencies among their attributes.

- A series of normal form tests that can be carried out on individual relations so that the relational database can be normalized to any desired degree.

The normal form of a relation refers to the highest normal form condition that it meets and hence indicates the degree to which it has been normalized.

- To reach the highest normal form the designers decompose the relation in top-down fashion.
- The process of normalization through decomposition must conform to the following properties while satisfying the criteria of normal forms.
 - (i) Lossless join /decomposition
 - (ii) Functional dependency preservation
 - (iii) Attribute preservation.

FIRST NORMAL FORM:

- First Normal Form (1NF) deals with the shape of the record type.
- The relation is said to be in 1NF if and only if, it contains no repeating attributes or groups of attributes.
- The relation/table is said to be in 1NF when each cell of the table/relation contains precisely one value (atomic)
- The relation is said to be in 1NF if each cell value is atomic.

Let us consider a relation student (Roll no, name, dob, subject, grade)

Roll No	Name	DOB	Subject	Grade	Teacher
101	Rakesh	14.11.1985	RDBMS	E	Smith
			DEC	A	Clark
			MATH-II	O	Allen
			PHY-II	E	Johnes
			NT	A	Martin
102	Tapan	18.07.1986	DEC	A	Clark
			MATH-II	E	Allen
			PHY-II	O	Johnes
			RDBMS	E	Smith

			NT	B	Martin
104	Mohan	19.07.1985	RDBMS	O	Smith
			MATH-II	A	Allen
			PHY-II	E	Johnes
			DEC	B	Clark
			NT	A	Martin

Table: An un normalized relation with multiple values in a cell.

- Here the table students is not normalized because the cells in subjects and grade are not atomic.
- By the definition of INF, each cell value must be atomic so, to normalize the relation we need to fill all the columns of rollno, name, dob by repeating values for each rows corresponding to subject and grade or each student.

So, the normalized table would be

Roll No	Name	DOB	Subject	Grade	Teacher
101	Rakesh	14.11.1985	RDBMS	E	Smith
101	Rakesh	14.11.1985	DEC	A	Clark
101	Rakesh	14.11.1985	MATH-II	O	Allen
101	Rakesh	14.11.1985	PHY-II	E	Johnes
101	Rakesh	14.11.1985	NT	A	Martin
102	Tapan	18.07.1986	DEC	A	Clark
102	Tapan	18.07.1986	MATH-II	E	Allen
102	Tapan	18.07.1986	PHY-II	O	Johnes
102	Tapan	18.07.1986	RDBMS	E	Smith
102	Tapan	18.07.1986	NT	B	Martin
103	Mohan	19.07.1985	RDBMS	O	Smith
103	Mohan	19.07.1985	MATH-II	A	Allen
103	Mohan	19.07.1985	PHY-II	E	Johnes
103	Mohan	19.07.1985	DEC	B	Clark
103	Mohan	19.07.1985	NT	A	Martin

Table-2: Student data representation in relation students.

The table is normalized to 1NF but it can lead to several undesirable problems

- Redundancy exists which leads to wastage of memory also.
- The multiple copies of same fact leads to update anomalies.
Ex: Change in date of birth leads to change in all tuples.
- If the teacher teaching a subject, it can be entered only if a student enrolls to the subject.
So it leads to insertion anomalies.
- If the only student in a given course discontinues the information as to which course a professor is teaching will be lost if the student information is deleted.

Properties of normalized relation

An ideal relation after normalization should have the following properties so that the problems of redundancy and inconsistency is avoided.

1. No data value should be duplicated in different rows unnecessarily.
2. A value must be specified (and required) for every attribute in a row.
3. Each relation should be self contained. In other words if a row from a relation is deleted, important information should not be (Deleted) accidentally lost.
4. When a row is added to a relation, other relations in the database should not be affected.
5. A value of an attribute in a tuple may be changed independent of other tuples in the relation and other relations.

The idea of normalizing relations to higher and higher normal forms is to attain the goals of having a set of ideal relations meeting the above criteria.

Decomposing the relation

The decomposition of a relation scheme $R = (A_1, A_2, A_3, \dots, A_n)$ is its replacement by a set of relation schemes $\{ R_1, R_2, \dots, R_m \}$ such that for $R_i \subseteq R$ for $1 \leq i \leq m$ and $R_1 \cup R_2 \cup R_3 \dots \cup R_m = R$.

A relation scheme R can be decomposed into a collection of relation schemes $\{ R_1, R_2, \dots, R_m \}$ to eliminate some of the anomalies contained in the original relation R .

Here (i) the relation scheme $R_i \leq R$. or $R_i \leq R, 1 \leq i \leq m$.

(ii) $R_i \cap R_j$ for $i \neq j$, need not be empty

(iii) $R_1 \cup R_2 \cup R_3 \dots \cup R_m = R$

Decomposition Criteria: The process of normalization through decomposition must confirm to the following properties while satisfying the criteria of normal forms.

- (i) Loss less join / decomposition
- (ii) Functional dependency preservation
- (iii) Attribute preservation.

Armstrong Rules:

The well known rules called inference rules are defined in functional dependencies.

IR1 : Reflexive rule:

If $X \supseteq Y$ then $X \rightarrow Y$

If $Y \subseteq X$ then $X \rightarrow Y$

Any set of attributes functionally determines itself $X \rightarrow X$

IR2 : Augmentation Rule:

If $X \rightarrow Y$ then $XZ \rightarrow YZ$

The augmentation rule may also be stated as

If $X \rightarrow Y$ then $XZ \rightarrow Y$

Augmenting the left hand side attribute of a functional dependency produces another functional dependency.

IR3: Decomposition / projective rule:

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

IR4: Decomposition / projective rule:

If $X \rightarrow YZ$ and $Z \rightarrow Y$ then $X \rightarrow Z$

IR5: Union or Additive rule:

If $\{ X \rightarrow Y \text{ and } X \rightarrow Z \}$ then $X \rightarrow YZ$

IR6: Pseudo Transitive rule:

If $\{ X \rightarrow Y, WY \rightarrow Z \}$ then $WX \rightarrow Z$

Of these the first three are known as Armstrong rules. These are sound because it is enough if a set of FDS satisfy these three.

They are complete because using these three rules we can generate the rest of the inference rules.

Full Functional dependency (FFD)

- The term full functional dependency (FFD) is used to indicate the minimum set of attributes in a determinant of a FD.
- In other words, the set of attributes X will be fully functionally dependent on the set of attributes Y if.
- X is functionally dependent on Y and
- X is not functionally dependent on any subset of Y

Eg.

Note: Like FD, FFD is a property of the information presented by the relation.

- It is not an indication of the way that attributes are formed in to relations or current contents of the relations.

IR7: Composition:

If $X \rightarrow Y$, and $Z \rightarrow W$ then $XZ \rightarrow YW$

IR8: Self Accumulation:

If $X \rightarrow YZ$, and $Z \rightarrow W$ then $X \rightarrow YZW$

FUNCTIONAL DEPENDENCY

- Normalization beyond 1NF relies on functional dependency.

- The value of an attribute in a row will uniquely determine the value of another attribute.
- Let X and Y be two attributes of a relation. If there is only one attribute value of Y corresponding to it for a given value of X, then Y is said to be functionally dependent on X.
- This is indicated by a relation $X \rightarrow Y$.
- The arrow notation should be read as “Functionally determines”.
- So X functionally determines Y and X is functionally dependent on X.
- If $X \rightarrow Y$, for any two tuples t_1 and t_2 , if $t_1X = t_2X \Rightarrow t_1Y = t_2Y$. Trivial functional dependency.
- The functional dependency $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$.
- If $X \rightarrow Y$ in R, this does not mean whether or not $Y \rightarrow X$ in R.

Ex.: $\{\text{Roll No}\} \rightarrow \{\text{Name}\} \neq \{\text{Name}\} \rightarrow \{\text{Roll No}\}$

Proof of IR1: suppose that $X \supseteq Y$ and there exist two tuples in some instance of r of R such that $t_1[X] = t_2[X]$

Then $t_1[Y] = t_2[Y]$ because $X \supseteq Y$

Hence $X \rightarrow Y$ must hold in R

Proof of IR2 : if $X \rightarrow Y$ then $XZ \rightarrow YZ$.

Let $X \rightarrow Y$ holds in a relation instance r of R but $XZ \rightarrow YZ$ does not hold.

Then there must be two tuples t_1 and t_2 in r such that

$$(1) t_1[X] = t_2[X]$$

$$(2) t_1[Y] = t_2[Y]$$

$$(3) t_1[XZ] = t_2[XZ]$$

$$(4) t_1[YZ] = t_2[XZ] \text{ which is not possible because from (1) \& (3) we deduce (5)}$$

$$(5) t_1[Z] = t_2[Z] \text{ and from (2) and (5) we deduce (6)}$$

$$(6) t_1[YZ] = t_2[YZ] \text{ contradicting (4) Hence the proof}$$

CLOSURE

The set of all functional dependencies that include F as well as all dependencies that can be inferred from F is called the closure of F and denoted by F^+ .

The closure F^+ of F is the set of all functional dependencies that can be inferred from F.

Eg.: $F = \{ \{A\} \rightarrow \{B\} \}$

$\{A\} \rightarrow \{B\} \}$

$A^+ = \{A, B, C\}$

i.e. $F^+ = A^+ = \{A, B, C\}$

Algorithm: Determining X^+ , the closure of X under F

$X^+ = X$ repeat

Old $X^+ = X^+$

For each functional dependency, $Y \rightarrow Z$ in F do

If $X^+ \supseteq Y$ then If $X^+ = X^+ \cup Z$

Until ($X^+ = \text{old } X^+$),

Proof of 1R3 " $\{X \rightarrow Y, Y \rightarrow Z\} \Rightarrow X \rightarrow Z$ "

Assume that (1) $X \rightarrow Y$

(2) $Y \rightarrow Z$

For any two tuple T_1 and T_2 in r such that

$T_1[X] = T_2[X]$ we must have (By assumptn (1))

(3) $T_1[Y] = T_2[Y]$ Hence we must that

(4) $T_1[Z] = T_2[Z]$ from (3) and assumptn (2) Hence $X \rightarrow Z$ must hold in r from (1) & (4)

Keys of relational schema:

A super key is a set of attributes S of a relation schema R i.e. SCR , with the property that no two tuples t_1 and t_2 will have $t_1[S] = t_2[S]$

The difference between any key and a super key is that a key is minimal.

Ex. $\{EMPNO\}$ is a key for the relation EMP whereas $\{EMPNO, ENAME\}$, $\{EMPNO, ENAME, JOB\}$ and any set of attributes is said to be a super key if it includes the key $\{EMPNO\}$

Candidate key: The minimal super key is called the candidate key.

Primary Key: If a relation schema has more than one key, each is called a candidate key.

- One of the candidate key is chosen as primary key and others are secondary keys.

Prime attribute: An attribute of a relation schema R is called a prime attribute of R if it is a member of some candidate key of R.

Non prime attribute: An attribute is called a non prime attribute if it is not a prime attribute i.e. an attribute is said to be a non-prime attribute if it is not a part of the candidate keys.

Prove IR4 through IR8 by using IR1 through IR3

Proof of IR4: $X \rightarrow YZ \Rightarrow X \rightarrow Y, X \rightarrow Z$

1. $X \rightarrow YZ$ (Given)
2. $YZ \rightarrow Y$ (using IR1 as $YZ \supseteq Y$)
3. $X \rightarrow Y$ (Using IR3 on 1 and 2)

Similarly, we can prove $X \rightarrow Z$

Proof of IR5: $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$

1. $X \rightarrow Y$ (Given)
2. $X \rightarrow Z$ (Given)
3. $X \rightarrow ZY$ (By augmenting with X on IR2)
4. $XY \rightarrow YZ$ (Using IR2 on 2 by Augmenting with Y)
5. $X \rightarrow YZ$ (Using IR3 on 3 and 4)

Proof of IR6: $\{X \rightarrow Y, WY \rightarrow Z\} \Rightarrow WX \rightarrow Z$

1. $X \rightarrow Y$
2. $WY \rightarrow Z$
3. $WX \rightarrow WY$ (Using IR2 on 1 by augmenting with W)
4. $WX \rightarrow Z$ (Using IR3 on 3 and 2)

Proof of IR7: $\{X \rightarrow Y, Z \rightarrow W\} \Rightarrow XZ \rightarrow YW$.

1. $X \rightarrow Y$ (Given)
2. $XZ \rightarrow YZ$ (Augmenting with Z on 1) –IR2
3. $Z \rightarrow W$ (Given)
4. $YZ \rightarrow YW$ (Augmenting with Y on 1) –IR2

5. $XZ \rightarrow YW$ (Using IR3, transitive on 2 and 5)

Proof of IR8: $\{X \rightarrow YZ \text{ and } Z \rightarrow W\} \Rightarrow X \rightarrow YZW$.

1. $X \rightarrow YZ$ (Given)
2. $YZ \rightarrow Z$ (IR1, on 1)
3. $X \rightarrow Z$ (IR3 on 1 and 2)
4. $Z \rightarrow W$ (Given)
5. $X \rightarrow W$ (IR3 on 3 and 4)
6. $X \rightarrow ZYZ$ (Augmenting with X on (1))- IR2
7. $XYZ \rightarrow YZW$ (Augmenting with YZ on 5)
8. $X \rightarrow YZ W$ (Using IR3 on 6 & 7) IR2.

Q. State Armstrong's axioms. Show that Armstrong's axioms are complete.

Ans. Armstrong's axioms: Let R (X, Y,Z) is a relation schema with attributes X, Y,Z for which Armstrong's axioms are stated as follows.

Inf. Rule-1 : Reflexivity, If $X \supseteq Y$, the $X \rightarrow Y$

Inf. Rule-2 : Augmentation, If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Inf. Rule-3 : Transitivity, If $X \rightarrow Y$, $Y \rightarrow Z$ then $X \rightarrow Z$.

Completeness of Armstrong's axioms:

Armstrong's axioms are said to be complete because for a given set F of functional dependencies we can determine the closure F^+ from the FD's of F by using the Armstrong axioms i.e. inference rules 1 through inference rule 3.

Proof: Let us consider a relation scheme R (A,B,C,D,E,F)

With set of FD'S, $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow E, CD \rightarrow DF, B \rightarrow E\}$.

$A^+ = \{A,B,C,E\}$

$\{CD\}^+ = \{C,D,E,F\}$

$F^+ = (ACD)^+ = \{A,B,C,D,E,F\}$

Here, several FDS which are inferred in F^+ are:

- (i) $A \rightarrow E$
- (ii) $CD \rightarrow EF$
- (iii) $AD \rightarrow F$

IF we are able to prove the above using Armstrong's axioms then we can say that the Armstrong's axioms are complex.

(i) $A \rightarrow E$

Since $A \rightarrow B$, and $B \rightarrow E$ then $A \rightarrow E$ by IR3: Transitive rule

(ii) $CD \rightarrow EF$

We have $CD \rightarrow E$ (i)

And $CD \rightarrow F$ (ii)

$CD \rightarrow CDE$ (iii) using IR2, Augmenting

$CDE \rightarrow EF$ (iv) using IR2, Augmenting with E on (ii)

(iii), (iv) = $CD \rightarrow EF$, using IR3 on (iii) and (iv)

(iii) $AD \rightarrow F$

We have the FD'S $A \rightarrow C$ (i) and $CD \rightarrow F$ (ii)

$AD \rightarrow CD$ (iii) using IR2, augmenting with D on (1)

$AD \rightarrow F$, Using IR3, transitive rule on (2) and (3)

Similarly all the FD'S in F^+ can be derived by using Armstrong's axioms, i.e. IR1 through IRS.

Hence, Armstrong's axioms are complete.

SECOND NORMAL FORM:

2NF is more stringent normal form than 1 NF.

Def1: A relation is in 2NF if and only if, it is in 1NF and every non-key attribute is fully functionally dependent on the whole key.

Def2: A relation schema R is in 2NF if every non prime attribute A in R is fully functionally dependent on the primary key of R.

Def3: A relation R is in 2NF provided it is in 1NF and every non prime attribute to be fully functionally dependent on candidate key of R.

Def4: A relation schema R is in 2NF if every non prime attribute A in R is not partially dependent on any key of R. or

A relation schema R is in 2NF if every non prime attribute A in R is fully functionally, dependent on every key of R.

- 2 NF is really a test of primary key.

- 2 NF is concerned with the concept of full functional dependency.
- If a relation is in 1 NF and has a single attribute key it must automatically be in 2NF.

Decomposing the relation:

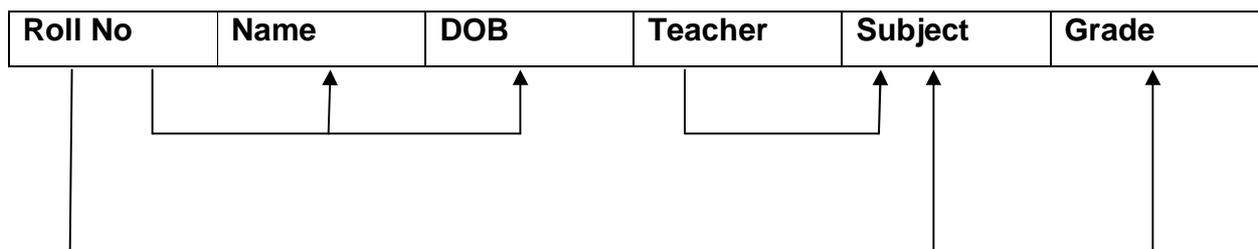
As the table-2 for student relation involved redundancy and inconsistencies due to insertion, deletion & updation. It can be decomposed in the three simple relations to eliminate the redundancy and inconsistency.

Stdinf (Roll no, name., dob)

Courseinf (subject, grade, rollno)

Teacher inf (subject, teacher)

Since, the dependencies in the student table are:



Stdinf

Roll No	Name	DOB
101	Rakesh	14.11.1985
102	Tapan	18.07.1986
103	Mohan	19.07.1985

Teacherinf

Subject	Teacher
RDBMS	Smith
DEC	Johnes
MATH-II	Allen
PHY-II	Clark
NT	Martin

Courseinf

Roll No	Subject	Grade
101	RDBMS	E
101	DEC	A
101	MATH-II	O
101	PHY-II	E
101	NT	A

Second Normal Form (2NF)

Let us consider a relation schema R (A,B,C,D,E,F) with a set of functional dependencies.

F=

- $\{A\} \rightarrow \{B\}$
- $\{A\} \rightarrow \{C\}$
- $\{A\} \rightarrow \{F\}$
- $\{A\} \rightarrow \{B\}$
- $\{A,D\} \rightarrow \{F\}$
- $\{D\} \rightarrow \{E\}$

- (i) Find the scooper key
- (ii) Closure
- (iii) Identify the candidate key
- (iv) IS it in 2NF ?
- (v) If not decompose it in to 2 NF form.

Solution: The given functional Dependencies are:

$\{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{D\} \rightarrow \{E\}, \{AD\} \rightarrow \{F\}$

Super key of R is $\{AD\}$

Closure $A^+ = \{A,B,C\}$

$D^+ = \{D,E\}$

$\{A,D\}^+ = \{A,B,C,D,E,F\}$

- There is no such single key that can derive /determine the other attribute.
- Here we take two keys together i.e. $\{AD\}$ as the minimum super key.

Hence the candidate key is $\{AD\}$.

The relation is not is 2 NF since partial dependency exists.

$\{A, D\} \rightarrow \{B\}$ partially dependency

$\{A, D\} \rightarrow \{C\}$

$\{A,D\} \rightarrow \{F\}$ Full functional dependency

$\{A,D\} \rightarrow \{E\}$ partial dependency

So we decompose our original relations

$R_1 (A,D,F)$ Since $\{A,D\} \rightarrow \{F\}$

$R_2 (A,B,C)$ Since $\{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}$

$R_3 (D,E)$ Since $\{D\} \rightarrow \{E\}$

Example:

Let us consider the relation student (Roll No, Name, Address , courseno, couasename, grade)

Roll No	Name	Address	Courseno	Coursename	Grade
---------	------	---------	----------	------------	-------

$\{\text{RollNo}\} \rightarrow \{\text{Name}\}$

$\{\text{Roll No}\} \rightarrow \{\text{Address}\}$

$\{\text{Course no}\} \rightarrow \{\text{Course name}\}$

$\{\text{Roll No, Course no}\} \rightarrow \{\text{Grade}\}$

Problems

- (1) IF we want to insert a new course then it can't be inserted unless a new student roll no, is created.
- (2) If we delete a student roll no, the subject or course information is deleted if the course is taken by only student.
- (3) If we update a name, address it has to be updated at many places.
- (4) redundancy exist at many places.

So, we decompose the relation student into subrelators as:

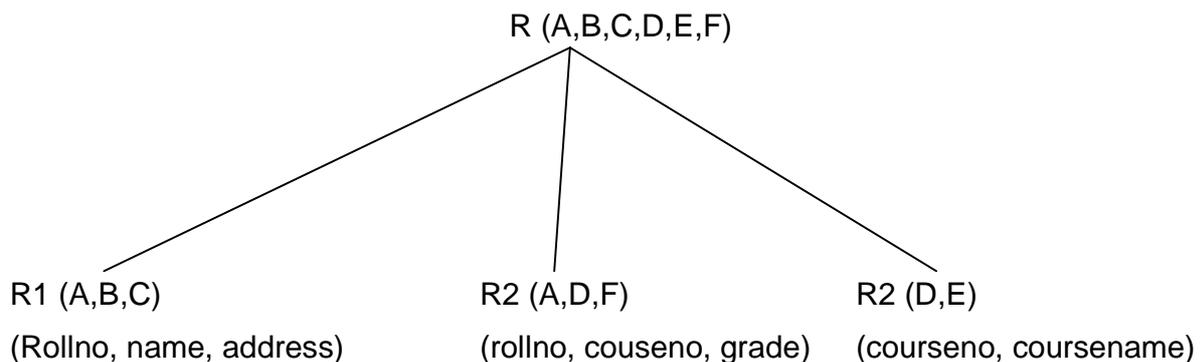
Student (roll no, name, address, courseno, coursename, grade)

Stdinfor (Roll no, name, address) = $R_1 (A,B,C)$

Grade course infor (Rollno, courseno, grade) = $R_2 (A,D,F)$

Course infor (Courseno, coursename) = $R_3 (D,E)$

(Student (rollno, name, address, courseno, coursename, grade)



THIRD NORMAL FORM (3NF)

- 3NF is more stricter normal form than the 2NF
- 3NF is concerned with transitive functional dependency.
- Transitive functional dependency may exist, only if there are more than one non key field.
- We may say that if a relation is in 2NF and has zero or one non key field it must automatically be in 3NF

Definition: A relation is in 3NF if and only if it is in 2NF and there are no transitive functional dependency.

- Transitive functional dependency arises when one non key attribute is functionally dependent on another non key attribute .
- In other words, the value of one non key attribute is determined by the value of another non key attribute.
- This means there is redundancy in the database.
- Let us consider a relation employee (proj.no, manager, address)

Proj.no	Manager	Address
P1	Black, B	32, High Street

P2	Smith, J	11, New street
P3	Black, B	32 High street
P4	Black,B	32 High street
P5	Thomas,J	69 Black street

- Here transitive dependency exists, as proj. no determines manager and manager determines address.
- If a manager is associated with more than one project then his address is repeated leading to data redundancy.
- If we need to change address of a manager we need to change at many places leading to updation anomalies.
- So, we split the table in to two relations.
- The determinant attributes become the primary key in the new relation
- The attribute manager becomes the link between two tables/new relations.

Project

Proj.no	Manager
P1	Black, B
P2	Smith, J
P3	Black, B
P4	Black,B
P5	Thomas,J

Manager

Manager	Address
Black, B	32, High Street
Smith, J	11, New street
Thomas,J	69, Back street

- Here in project relation , proj.no serves as the primary key and manager can be determined by project no.
- In manager relation, address can be determined by knowing the manager name, so, manager becomes the primary key.
- So, in the above relation the non prime attributes depend upon the primary key.
- Hence we can define the third normal form as below

Alternative

Definition: A relation is said to be in 3NF when it is in 2NF and every non key attributes is functionally dependent only on the primary key.

Guidelines for converting a table to 3NF:

- Find the non. key attributes and remove them that are functionally dependent on attributes that are not the primary key.
- Place them in a different relation.

Third Normal Form (3NF) (Summary)

A relation R is said to be in third normal form (3NF) if the relation R is in 2NF and non-prime attributes are:

- Mutually independent
- Functionally dependent on the primary key
- In other words, no attributes of the relation should be transitively dependent on the primary key.
- Thus in 3NF, no non prime attribute is functionally dependent on another non-prime attribute.
- This means that a relation in 3NF consists of the primary key and a set of independent non prime attributes.

Summary of Normal Forms

1NF	Relation should have no non atomic attributes	Forms new relations for each non atomic attribute or rested relations.
2NF	For relations where primary key	Decompose and set up a new relation for

	contains multiple attribute s no non key attribute should be functionally dependent on a primary key.	each partial key with its dependent attribute(s) make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
3NF	A relation should have a nonkey attribute functionally determined by another nonkey attribute (or by a set of non key attributes) .That is , there should be no transitive dependency of non key attribute on the primary key.	Decompose and set up a relation that includes the non key attribute(s) that functionally determine (s) other non key attributes.

Third normal form (3Nf) contd.

Definition: A Relation is in 3NF if and only if it is in 2NF and for any non trivial functional dependency.

$X \rightarrow Y$

- (i) Either X is the super key or
- (ii) Y to be a prime attribute

Example: Let us consider a relation R (A,B,C,D)

With functional dependencies $\{A\} \rightarrow \{B\}$, $\{A\} \rightarrow \{C\}$

And $\{C\} \rightarrow \{D\}$.Determine the candidate key and normalize upto 3NF.

Solution: Functional Dependencies are:

Here $\{A\} \rightarrow \{B\}$

$\{A\} \rightarrow \{C\}$

$\{C\} \rightarrow \{D\}$

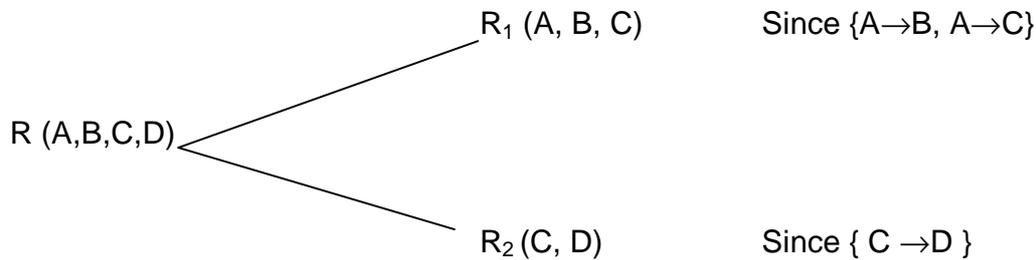
The closure of C = $C^+ = \{C, D\}$

The closure of A = $A^+ = \{A, B, C, D\}$

So, A is the candidate key. Since the key determine the non key attributes it is in 2NF.

Since there exist transitive dependencies $\{A\} \rightarrow \{C\}$ and $\{C\} \rightarrow \{D\}$.

So, we decompose, the relation into two relations



Lossless join and Dependency preserving Decomposition.

A relation schema R can be decomposed in to collection of relation schemes to eliminate some of the anomalies contained in the original relation scheme R .

However the decomposition should be maintained with

- (i) No loss of information
- (ii) Dependency preserved
- (iii) Attribute preserved.

To main ten the above properties we check the following.

- (i) Lossless join decomposition
- (ii) Dependency preserving
- (iii) Attribute preserving

Lossless join decomposition

A decomposition of a relation scheme $R \langle S,F \rangle$ in to the relation schemes, $R_i (1 \leq i \leq n)$ is said to be lossless join decomposition or simply lossless if for every relation $R(R)$ that satisfies the FDS in F , the natural join of the projections of R gives the original relation R . is

$$R = \pi_{R_1}(R) \times \pi_{R_2}(R) \times \dots \times \pi_{R_n}(R)$$

If $R_C = \pi_{R_1}(R) \times \pi_{R_2}(R) \times \dots \times \pi_{R_n}(R)$ then the decomposition is called lossy.

Dependency Preserving:

The given relation scheme $R(S, F)$ where F is the set of functional dependencies on the attribute in S , R is decomposed into the relation schemes R_1, R_2, \dots, R_n with the functional dependencies F_1, F_2, \dots, F_n . Then this decomposition of R is dependency preserving if the closure of F

Where, $F^1 = F_1 \cup F_2 \dots \cup F_n$ is identical to F^+

(i.e. $F^+ = F$) i.e. $F_1(R_1) \cup F_2(R_2) \cup F_3(R_3) \dots \cup F_n(R_n) = F^+$

Thm: A decomposition of relation scheme $R(C, X, Y, Z, F)$ into $R_1(C, Y, F_1)$ and $R_2(X, Z, F_2)$ is

- (a) Dependency preserving if every functional dependency in R can be logically derived from the functional dependencies of R_1 and R_2 i.e. $(F_1, F_2)^+ = F^+$ and
- (b) Is lossless if the common attributes X of R_1 and R_2 form a super key of at least one of these i.e. $X \rightarrow Y$ or $X \rightarrow Z$.

Example: Check for lossless decomposition:

Let the given relation $R(A, B, C, D)$ with the functional dependencies $FD = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$ let the relation has been decomposed preserving the dependencies as $R_1(A, B, C)$ and $R_2(C, D)$ let us verify whether it is lossless or not

A	B	C	D
a1	a2	a3	a4
b2	b2	a3	a4

A	B	C	D
a1	a2	a3	a4
b2	b2	a3	a4

Here, we considered the FD: $C \rightarrow D$, Since the symbols in column C are same and hence we made the symbol in column C are same and hence we made the symbols in column D same as one of the value in D column is a . For other dependencies $A \rightarrow B$, $A \rightarrow C$ we do not find two rows with identical entries for columns of the determinant. Here we get one row with all a values lossless.

Problem: Given $R(A, B, C, D, E)$ with FDS, $F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}$, the decomposition of R into $R_1(A, B, C)$, $R_2(B, C, D)$ and $R_3(C, D, E)$ is lossy.

A	B	C	D	E
a1	a2	a3		
	a2	a3	a4	

		a3	a4	a5
--	--	----	----	----

When we consider the FD: $C \rightarrow D$, we find all rows of the column C, the determinant of the FD, are identical this allows us to change the entries in the column D to a4.

No further changes are possible.

Finally, we find no rows in the table with all as and conclude that the decomposition is lossy.

Problem: The relation R (A,B,C) with FD = $\{B \rightarrow C\}$ exists.

Does the relation have potential candidate key

If it does what is it ? if it does not why not.

Solution: R (A,B,C) So, super key = $\{A,B,C\}$

$$F = \{B \rightarrow C\}$$

$$A^+ = \{A\}$$

$$B^+ = \{B,C\}$$

$$AB^+ = \{A,B,C\}$$

The potential candidate key = $\{A,B\}$

Problem: Let us consider a relation R (A,B,C,D,E).

With following dependencies $\{A,B\} \rightarrow \{C\}$

$\{C,D\} \rightarrow \{E\}$, $\{D,E\} \rightarrow \{B\}$

Find the candidate key of the given relation.

Solution: FD^s are: $\{A,B\} \rightarrow \{C\}$

$$\{C,D\} \rightarrow \{E\}$$

$$\{D,E\} \rightarrow \{B\}$$

$$\{A,B\}^+ = \{A,B,C\}$$

$$\{A,B,D\}^+ = \{A,B,C,D,E\}$$

So, $\{A,B,D\}$ is the candidate key

Problem: Let the relation R (A,B,C,D,E,F,G,H,I,J) has functional dependencies:

$$F = \{A,B\} \rightarrow \{C\}$$

$$\{B,D\} \rightarrow \{E,F\}$$

$$\{A,D\} \rightarrow \{G,H\}$$

$$\{A\} \rightarrow \{I\}$$

$$\{H\} \rightarrow \{J\}$$

Solution: $\{AB\}^+ = \{A,B,C,I\}$

$$\{BD\}^+ = \{B,D,E,F\}$$

$$\{AD\}^+ = \{A,D,G,H,I,J\}$$

$$\{A\}^+ = \{A,I\}$$

$$\{H\}^+ = \{H,J\}$$

$$\{A,B,D\}^+ = \{A,B,C,D,E,F,G,H,I,J\}$$

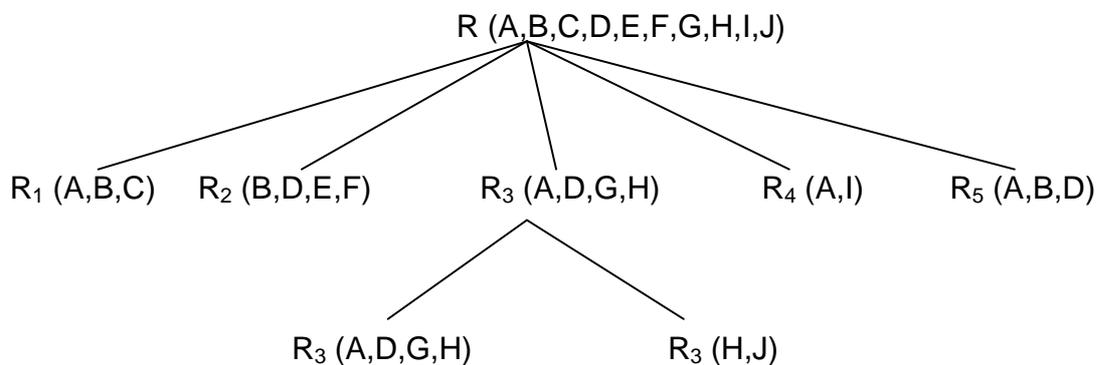
Training / Seminar

Roll No.	Date	Topic	Guide	Branch	Grade	Name	Dept.	Cod.
101	3.4.08	DM	CKM					

Roll No.	Name	Branch	Sub	Teacher	Grade	Rem.
----------	------	--------	-----	---------	-------	------

(i) Here, single dependency, exists so it is in INF.

(ii) Partial dependency exists, so it is not in 2NF Decomposition for 2NF.



BCNF:

A relation schema R is in BCNF with respect to a set for functional dependencies, if for all FDS in .

F+ $X \rightarrow Y$ atleast one of the following is true.

(i) $X \rightarrow Y$ is trivial or

(ii) X is a super key

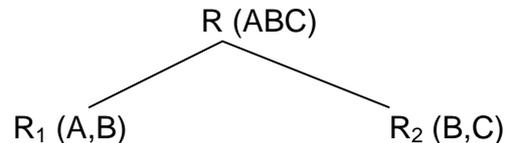
EX: R (A,B,C) is the given relation schema and the given Functional Dependency is

$$F = \{A \rightarrow B, B \rightarrow C\}$$

Solution: Key = {A} since $A^+ = \{A, B, C\}$

R is not in BCNF as B is not a super key.

Decomposition:



Now R_1 and R_2 in BCNF

The decomposition is lossless and dependency preserving.

Boyce Code Normal Form (BCNF)

- To eliminate the problems and redundancy of 3NF. Boyce proposed a normal form known as Boyce code Normal form (BCNF).
- When the relation has more than one candidate key anomalies may arise even though the relation is in 3NF.
- 3NF does not deal with the case of a relation with overlapping candidate keys.
- BCNF is based on the concept of a determinant
- A determinant is any attribute(s) on which some other attributes is fully functionally dependent.

Def: A relation is in BCNF if and only if every determinant so candidate key.

Let us consider a relation R (A,B,C,D) with the set of functional dependencies.

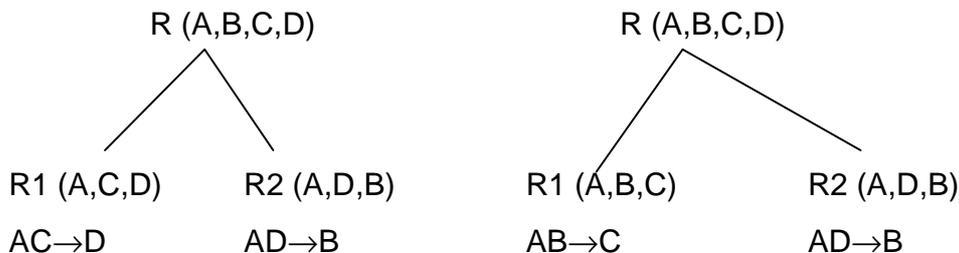
$$\{A, C\} \rightarrow \{B, D\}$$

$$\{A, D\} \rightarrow \{B\}$$

- Here the first determinant suggests that the primary key of R could be changed from {A,B} to {A,C}.
- If {A,C} would be the key then it could also determine all of the non key attributes present in R.
- However the second determinant indicates that {A,D} determines B. But {A,D} can't be the key of R since it does not determine all the non key attributes of R (Eg.C).
- Here, first determinant is a candidate key $\rightarrow \{A, C\}$.
- But, the second determinant {A,D} is not a candidate key.

- Thus the relation $R(A,B,C,D)$ is not in BCNF, since there exist overlapping keys candidate keys.

So the decomposition of the relation for BCNF is



Comparison of 3NF and BCNF:

- BCNF is stronger than 3NF
- The relations there in 3NF are not necessarily in BCNF
- BCNF is needed in certain situations to obtain full understanding of the data model.
- There are several routes to take to arrive at the same set of relations in BCNF.

Difference: The difference between 3NF and BCNF is that for a functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a primary key attribute and A is not a candidate key.

Whereas, BCNF insists that for this dependency to remain in this relation, A must be candidate key.

Therefore, BCNF is a stronger form of 3NF, such that every relation in BCNF is also in 3NF.

Q. Prove that any binary relation is in BCNF:

A relation R is in BCNF if for every non trivial functional dependency $X \rightarrow Y$, X to be the super key.

Let us consider a binary relation $R(A,B)$.

The possible functional dependencies are

$\{A,B\} \rightarrow Q$

$\{A,B\} \rightarrow \{A\}$

$\{A,B\} \rightarrow \{B\}$

$\{A,B\} \rightarrow \{A,B\}$

$\{A\} \rightarrow \{B\}$

The non trivial functional dependencies are

(i) $\{A\} \rightarrow \{B\}$, (ii) $\{B\} \rightarrow \{A\}$, (iii) $\{A\} \rightarrow \{B\}$, $\{B\} \rightarrow \{A\}$

So in any case for $X \rightarrow Y$, X is the super key.

Hence, the relation R (A,B) is in BCNF.

BCNF Contd.

Def: A relation R is Said to be in BCNF if for every non trivial FD : $X \rightarrow Y$ between attributes X and Y holds in R.

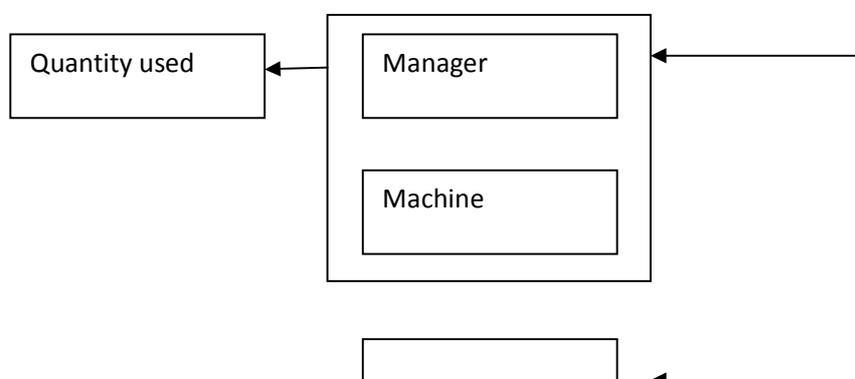
i.e. X is a super key of R.

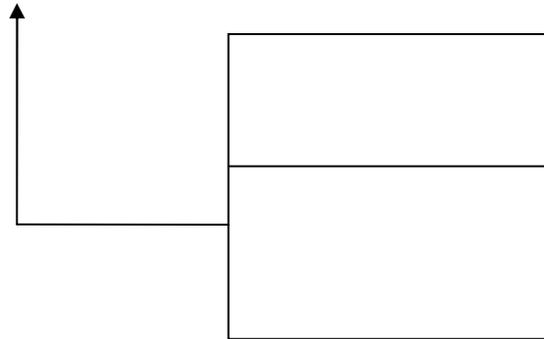
- $X \rightarrow Y$ is a trivial functional dependency i.e. $Y \rightarrow X$.
- In other words, a relation must have candidate keys as determinants.
- To find whether a relation is in BCNF or not, FDS within each relation is examined. If all non key attributes depend only on the complete key than the relation is in BCNF.
- Let us consider a relation project (Project code, manager machine, Quantityused).

Project

Project Code	Manager	Machine	Quantity used
P1	Thomas	Excavator	5
P3	John	Shovel	2
P2	Abhisek	Drilling	5
P4	Avinash	Dumper	10
P3	John	Welding	3
P1	Thomas	Drilling	4

The Functional Dependency diagram is given by:





Problems

Here, there are two overlapping candidate keys {Manager, Machine} and {Project code, Machine}

In the functional dependencies

$\{\text{Manager}\} \rightarrow \{\text{Project Code}\}$

$\{\text{Project code}\} \rightarrow \{\text{Manager}\}$

Neither, manager, nor project code is a super key.

Decomposition for BCNF:

(1) PROJECT (Project code, Machine, Quantity used.

(2) MANAGER (Project code, Manager)

Boyce Code- Normal form (BCNF) contd.

Def: A relation is said to be in BCNF if and only if every determinant is a candidate key.

Def: A relation is said to be in BCNF if and only if it is in 3NF or 2NF and for every non trivial functional dependency $X \rightarrow Y$.

- (i) X is a super key
- (ii) Y to be the prime attribute X to be super key.

Example: Let us consider a relation R (A,B,C) with the set of functional dependencies.

$F = \{A,B\} \rightarrow \{C\}$ or $\{A\} \rightarrow \{B\}$, $\{B\} \rightarrow \{C\}$

$\{C\} \rightarrow \{B\}$.

Solution:

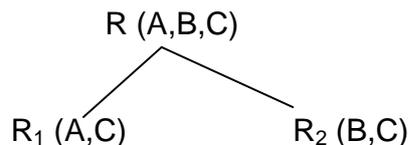
$\{A,B\}^+ = \{A,B,C\}$

Here, $\{A,B\}$ is the candidate key, since there is no such single key for deriving all the non key attributes. So we take the minimal super key $\{A,B\}$ as the candidate key.

The relation is not in BCNF since $\{C\}$ is not the .

Decomposition for BCNF:

The relation $R(A,B,C)$ can be decomposed into $R_1(A,C)$ and $R_2(B,C)$



Design goals of relational database design.

There are three important design goals

- (i) BCNF
- (ii) Loss less join
- (iii) Dependency preservation

If we can't achieve all these three, we accept

- (i) 3NF
- (ii) Loss less join
- (iii) Dependency preservation

Multivalued Dependencies:

A multivalued dependency corresponds to a many many relationship or a many one or one many relationship.

Def: The (Functional) Dependency represented by $X \twoheadrightarrow Y$ is said to be multivalued dependency from X to Y if and only if each X value exactly determined a set of Y values, independently of other attributes.

Def: The dependency $X \twoheadrightarrow Y, Z$ is said to be a multivalued dependency if.

- (i) For a single value of X there exist a set of values of Y
- (ii) For a single value of X there exist a set of values of Z

Where Y and Z are independent of each other.

Explanation:

Let R be a relation and X and Y are any arbitrary set of attributes of R, if for any two tuples, t_1 and t_2 with $t_1[X] = t_2[X]$, there exists two tuples t_3 and t_4 such that $(Z=R=(XUY))$.

$$t_1[X] = t_3[X] = t_4[X]$$

$$t_1[Y] = t_3[Y] \ \& \ t_2[Y] = t_4[Y]$$

$$t_1[Z] = t_3[Z] \ \& \ t_2[Z] = t_4[Z]$$

then $X \twoheadrightarrow Y$

Example:

R (A,B,C)

(i) $A \twoheadrightarrow B$

(ii) $A \twoheadrightarrow C$

A	B	C
10	a	w
10	b	t
10	a	t
10	b	w
20	p	u
20	p	v
20	q	u

Trivial Multivalued Dependencies : Trivial MVDS Let R is a relation with set of attributes X and Y then $X \twoheadrightarrow Y$ is said to be trivial iff.

(a) $Y \subseteq X$ or

(b) $X \cup Y = R$

(c) A trivial MVD does not represent a constraint.

(d) MVDs are generalization of FDS

(e) The existence of non-trivial MVDs that are not FDS causes problems.

Non Trivial MVD : An MVD that satisfies neither (a) nor (b) conditions above is called non trivial MVD.

Types: Like trivial FDS, there are two kinds of trivial MVDS.

(a) $X \twoheadrightarrow Q$, where Q is an empty set of attributes.

(b) $X \twoheadrightarrow A-X$, where A comprises all the attributes in a relation.

Both these types of trivial MVDs hold for any set of attributes of R and therefore can serve no purpose as design criteria.

Properties of MVDS: Berry described the relevant rules to derive MVD

Rule 1: Reflexive (inclusive) : If $Y \subset X$ then $X \twoheadrightarrow Y$

Rule 2: Augmentation rule : If $X \twoheadrightarrow Y$ and $W \subset V$ and $V \subset W$ then $WX \twoheadrightarrow VY$.

Rule 3 Transitive rule: If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$ then $X \twoheadrightarrow Z$.

Rule 4 Complementation : if $X \twoheadrightarrow Y$ then $X \twoheadrightarrow U-X-Y$ holds.

Additional rules to derive closure of set of FDs and MVDs.

Rule 5 Intersection: If $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow Y \cap Z$

Rule 6 Prado transitivity: If $X \twoheadrightarrow Y$ and $WY \twoheadrightarrow Z$ then $XW \twoheadrightarrow Z-W$

Rule 7 Union: If $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$ then $X \twoheadrightarrow YZ$

Rule 8 Difference: If $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$ then $X \twoheadrightarrow Y-Z$ and $X \twoheadrightarrow Z-Y$.

Rule 9 Replication : if $X \rightarrow Y$, then $X \twoheadrightarrow Y$.

Rule 10 coalescence: If $X \twoheadrightarrow Y$ and $Z \subset Y$ and there is a W such that $W \subset U$ and $W \cap Y = \text{NULL}$ and $W \rightarrow Z$, then $X \rightarrow Z$,

Note: U is the set of all attributes of R.

Fourth Normal Form:

Def: A relation R is said to be in fourth normal form (4NF) if it is in BCNF and for every non trivial MVD $X \twoheadrightarrow Y$ in F^+ , X is a super key of R.

- The 4NF is concerned with dependencies between the elements of compound keys composed of three or more attributes.
- The 4NF eliminates the problems of 3NF.
- The 4NF is violated if the relation has undesirable MVDS and hence can be used to identify and decompose such relations.

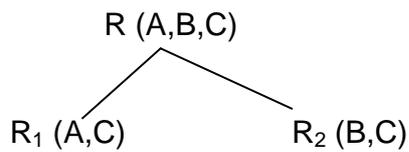
•

Example:

R (A,B,C) with MVDS

(i) $A \twoheadrightarrow B$

(ii) $A \twoheadrightarrow C$



A	B	C
10	a	w
10	b	t
10	a	t
10	b	w
20	p	u
20	p	v
20	q	u

A	B
10	a
10	b
20	P
20	q

A	C
10	w
10	t
20	u
20	v

Here, $R = R_1 \times R_2$

So, it is a lossless decomposition

Problems: Consider a relation R (A,B,C,D,E,F) with FDS and with MVDS $A \twoheadrightarrow B$ and $CD \twoheadrightarrow EF$.

Solution: R (A,B,C,D,E,F), the given relation

Here , the MVDS are $A \twoheadrightarrow B$

By complement rule $A \twoheadrightarrow CDEF$

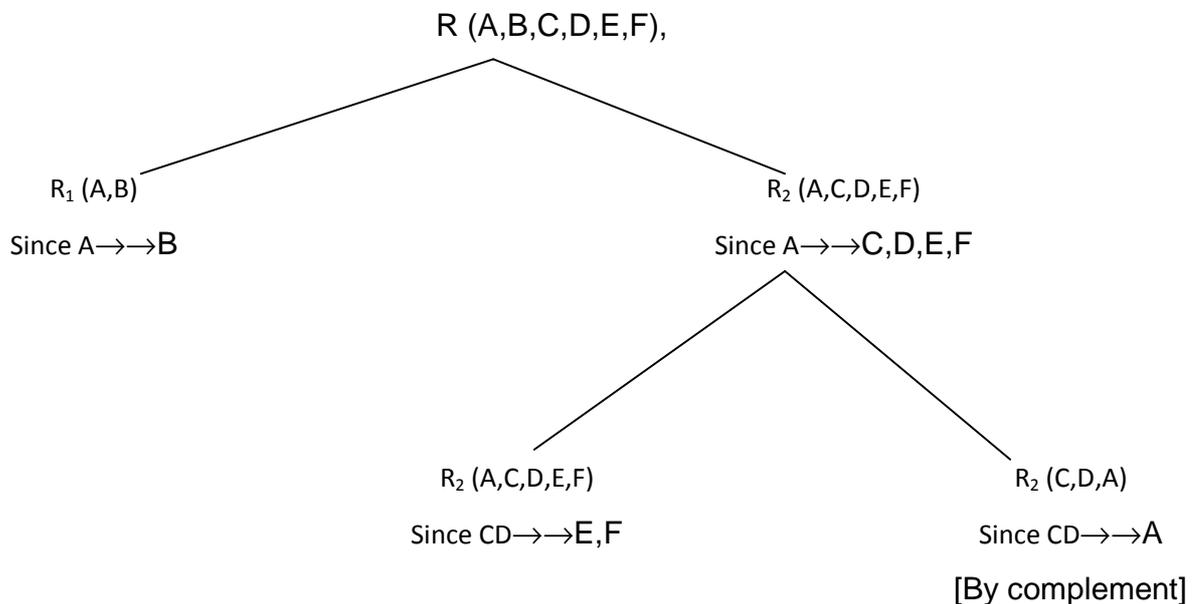
To decompose the relation R in to two relation for 4NF, such that

$R_1 (A,B)$ and $R_2 (A,C,D,E,F)$

As $A \twoheadrightarrow B$ and $A \twoheadrightarrow C,D,E,F$

- Now $R_1 (A,B)$ is in 4NF because $A \twoheadrightarrow B$ is trivial and is the only MVD in the relation.
- The relation $R_2 (A,C,D,E,F)$ is still in BCNF because there exist non trivial MVD $CD \twoheadrightarrow EF$.
- Now $R_2 (A,C,D,E,F)$ is decomposed in to relations $R_2 (C,D,E,F)$ and $R_2 (C,F,A)$ by applying MVDS, $CD \twoheadrightarrow EF$ and $CD \twoheadrightarrow A$ [By complement rule]

Now both $R_2 (C,D,E,F)$ and $R_2 (C,F,A)$ are in 4NF.



Join Dependency:

So for every relation was non loss decomposable in to two projections.

In case of n-decomposable relations we think about join dependency.

A join dependency is a further generalization of MVDS

Def: A join dependency (JD) $X \{R_1, R_2, \dots, R_n\}$ is said to be hold over a relation R if R_1, \dots, R_n is a lossless join decomposition of R.

An MVD $X \twoheadrightarrow Y$ over a relation R can be expressed as the join dependency $\{XY, X(R-Y)\}$.

Let us consider the relation, CTB (Course, teacher, book)

Course	Teacher	Book
Phy101	S.K. Biswal	Mechanics
Phy101	S.K. Biswal	Optics
Phy101	R.K. Pati	Mechanics
Phy101	R.K. Pati	Optics
Math 301	R.Das	Algebra
Math 301	R.Das	Vectors
Math 301	R.Das	Geometry

Here in the CTB relation the MVD, $C \twoheadrightarrow T$ can be expressed as the join dependency $\{CT, CB\}$

Join Dependency and multivalued dependency : (JDS Vs MVDS)

- Join Dependency R (A,B,C) satisfies multivalued dependency (AB, AC) if and only if it satisfies the MVDs $A \twoheadrightarrow BC$.
- JD is the most general form of dependency (read as determination) possible between the attributes of a relation (in the relational model)

Fifth Normal Form: (5 NF)

Def: A relation schema R is said to be in fifth normal form (5NF) if, for every JD $X [R_1, R_2 \dots R_n]$ that holds over R, one of the following statements is true.

- $R_1 = R$ for some I or
- The JD is implied by the set of those FDS over R in which the left side is a key for R.
- The decomposition of R into $[R_1, R_2, \dots, R_n]$ is lossless join whenever the key dependencies (Functional dependency in which the left side is a key for R) hold.
- JD $X [R_1, \dots, R_n]$ is a trivial FD if $R_1 = R$ for some I such a JD always holds.

- If a relation schema is in 3NF and each of its keys consists of single attribute it is also in 5NF.
 - (i) This is sufficient for a relation to be in 5NF but not necessary.
 - (ii) We conclude that a relation is in 5NF without ever identifying the MVDS and JDS that may hold over the relation.

NORMALIZATION

SAMPLE QUESTIONS

1. Define the term functional dependency
2. Why some functional dependencies are called trivial ?
3. Describe the concept of full functional dependency
4. Draw a functional dependency diagram with an example.
5. What is the closure of a functional dependency ?
6. What do you understand by the form normalization ?
7. What is the purpose of normalizing the data ?
8. Define 1NF, 2NF, 3NF and BCNF
9. Describe the properties of a normalized relation
10. How BCNF is different from 3NF
11. Why 4NF is preferred to BCNF
12. Define MVD and 4NF
13. Define JD and 5NF.
14. When an MVD is said to be trivial ?
15. Write four important armstrongs inference rule of MVD
16. Mention three important armstrongs inferences rules of FDS.
17. Discuss the problems of 2NF. How it is over come ?
18. Discuss the problems of 3NF. How it is over come.
19. Discuss the problems of 4NF ? How is over come ?
20. Explain the lossless join decomposition with example.
21. How do you check for the lossless join decomposition ?
22. How do you test for dependency preserving in a decomposition ?
23. How do you check for existence of redundancy in the relation.
24. Prove that BCNF \Rightarrow 3NF but the converse is not true.

25. Prove that every binary relation is in BCNF.
26. Describe Armstrong's Axioms. What are the derived rules ?
27. Describe how database designers typically identify the set of FDS associated with a relation.
28. Describe the concept of full functional dependency.
29. What is dependency preservation property of for decomposition why it is important ?
30. What is lossless or non additive join property of decomposition why it is important?

Assignment Problems

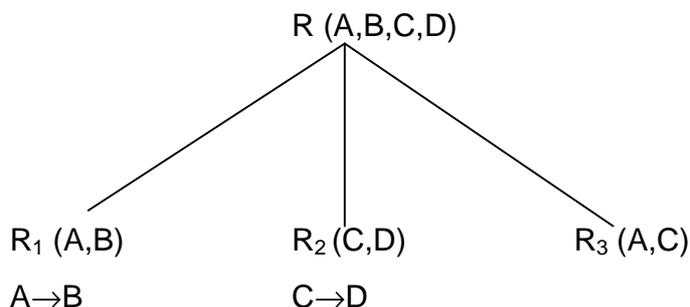
1. Give a set of FDS for the relation schema R (A,B,C,D) with primary key AB under which R is in 1NF but not in 2NF.
2. Consider a relation R with five attributes R (A,B,C,D,E) with the set of functional dependencies.
 $F = \{A \rightarrow B, BC \rightarrow E \text{ and } ED \rightarrow A\}$
 - (i) Find the candidate key
 - (ii) Is R in 3NF
 - (iii) Is R in BCNF.
3. Compute the closure of the following set F of functional dependencies for relation schema R = (A,B,C,D,E)
 - $A \rightarrow BC$
 - $CD \rightarrow E$
 - $B \rightarrow D$
 - $E \rightarrow A$
 - (a) List the candidate key
 - (b) Compute the closure B^+
4. Give a set of FDs for the relation schema R (A,B,C,D) with primary key AB under which R is in 2NF but not in 3NF.

5. Consider the relation schema $R(ABC)$ which has the FDS $B \rightarrow C$. if A is the candidate key of R , is it possible for R to be in BCNF ? If so under what conditions ? If not why not ? Explain.
6. Consider the relation $r(ABCDE)$ with the dependencies $A \rightarrow B$, $BE \rightarrow E$ and $ED \rightarrow A$.
 - (i) List all the keys
 - (ii) Is R in 3NF
 - (iii) Is R in BCNF
 - (iv) If R is not in BCNF, decompose to obtain BCNF
7. A relation $r(A,B,C,D)$ has FD $AB \rightarrow C$ and $C \rightarrow A$, is R in 3NF or in BCNF ? justify your answer
8. A relation $R(A,B,C,D)$ has FD $C \rightarrow B$ is R in 3NF ? justify your answer.
9. A relation $R(A,B,C)$ has FDS $A \rightarrow C$ is R in 3NF ? Does $AB \rightarrow C$?
10. What undesirable dependencies are avoided in 3NF?

Problems: Consider the relation scheme $R(A,B,C,D)$ and the functional dependency $A \rightarrow B$ and $C \rightarrow D$ then the decomposition of R into $R_1(A,B)$ and $R_2(C,D)$ and $R_3(A,C)$

- (a) Dependency preserving and lossless join
- (b) Lossless join but not dependency preserving
- (c) Dependency preserving and not lossless join
- (d) Not dependency preserving and not lossless join

Solution



- (i) From the above decomposition it is clear that it is dependency preserving
- (ii) Lossless check

A	B	C	D
---	---	---	---

R ₁	a ₁	a ₂		
R ₂			a ₃	a ₄
R ₃	a ₁		a ₃	a ₄

Since there is no single row with all a values . Hence it is a lossy decomposition.

Solution

$D3 = \{R_1, R_2, R_3, R_4, R_5\}$

$R_1 = \{A, B, C, D\}$

$R_2 = \{D, E\}$

$R_3 = \{B, F\}$

$R_4 = \{F, G, H\}$

$R_5 = \{D, I, J\}$

(i) Dependency Preserving

In $R_1 = \{A, B, C, D\}$, $\{A, B\} \rightarrow \{C\}$, $\{A\} \rightarrow \{D, E\} \Rightarrow \{A\} \rightarrow \{D\}$

In $R_2 = \{D, E\}$, $\{D\}$ the candidate key

In $R_3 = \{B, F\}$, $\{B\} \rightarrow \{F\}$

In $R_4 = \{F, G, H\}$, $\{F\} \rightarrow \{G, H\}$

In $R_5 = \{D, I, J\}$, $\{D\} \rightarrow \{I, J\}$

So, the decompositions is dependency preserving

(ii) Check for Lossess

	A	B	C	D	E	F	G	H	I	J
R ₁	a ₁	a ₂	a ₃	a ₄		a ₆	a ₇	a ₈	a ₉	a ₁₀
R ₂				a ₄	a ₅					
R ₃		a ₂				a ₆		a ₈		
R ₄				a ₄		a ₆	a ₇		a ₉	a ₁₀
R ₅										

Since there is not row with all a values we conclude that the decomposition is lossy.

(iii) Determining the normal form

$R_1 = \{A, B, C, D\}$, $\{A, B\} \rightarrow \{C\}$, $\{A\} \rightarrow \{D\} \Rightarrow$ partial dependency \Rightarrow not in 2NF \Rightarrow in 1 NF.

$R_2 = \{D, E\}$, Binary relation \Rightarrow BCNF

$R_3=\{B,F\}$, Binary relation \Rightarrow BCNF

$R_4=\{F,G,H\}$, $\{F\} \Rightarrow \{G,H\}$, $\{F\} \rightarrow$ super key \Rightarrow BCNF

$R_5=\{D,I,J\}$, $\{D\} \Rightarrow \{I,J\}$, $\{D\} \rightarrow$ super key \Rightarrow BCNF

Problems: Consider the universal relation R (A,B,C,D,E,F,G,H,I,J) and the set of functional dependencies.

$F = \{A,B\} \rightarrow \{C\}, \{A\} \rightarrow \{D,E\}, \{B\} \rightarrow \{F\}, \{F\} \rightarrow \{G,H\}, \{D\} \rightarrow \{I,J\}$

Determine whether each decomposition has satisfied.

- (i) The dependency preserving property
- (ii) Lossless join property with respect to the set of FDS, F.
- (iii) Determine in which normal form the decomposed relations exist.

(a) $D_1 = \{R_1, R_2, R_3, R_4, R_5\}$

$R_1=\{A,B,C,D\}$, $R_2=\{D,E\}$, $R_3=\{B,F\}$, $R_4=\{F,G,H\}$, $R_5=\{D,I,J\}$

(b) $D_2 = \{R_1, R_2, R_3\}$

$R_1=\{A,B,C,D\}$, $R_2=\{D,E\}$, $R_3=\{B,F\}$,

(c) $D_3 = \{R_1, R_2, R_3, R_4, R_5\}$

$R_1=\{A,B,C,D\}$, $R_2=\{D,E\}$, $R_3=\{B,F\}$, $R_4=\{F,G,H\}$, $R_5=\{D,I,J\}$

Solution

$\{A,B\}^+ = \{A,B,C\}$

$\{A\}^+ = \{A,D,E\}$

$\{B\}^+ = \{B,F\}$

$\{F\}^+ = \{F,G,H\}$

$\{D\}^+ = \{D,I,J\}$

$F^+ = \{A,B,C,D\}^+ = \{A,B,C,D,E,F,G,H,I,J\}$

$FDS = F_1(R_1) \cup F_2(R_2) \cup F_3(R_3) \cup F_4(R_4) \cup F_5(R_5)$

$= \{A,B,C\} \cup \{A,D,E\} \cup \{B,F\} \cup \{F,G,H\} \cup \{D,I,J\}$

$= \{A,B,C,D,E,F,G,H,I,J\} = F^+$

So, the decomposition D_1 is dependency preserving .

- (i) Check for Lossess

A	B	C	D	E	F	G	H	I	J
---	---	---	---	---	---	---	---	---	---

Since all the values in a row are a values then we conclude that the decomposition is lossess.

$R_1 = \{A, B, C, D, E\}$, $\{A, B\} \rightarrow \{C\}$, $\{A\} \rightarrow \{D, E\}$

Here the partial dependency exist so it is in 1NF.

$R_2 = \{B, F, G, H\}$, $\{B\} \rightarrow \{F\}$, $\{F\} \rightarrow \{G, H\}$

Here, transitive dependency exist, so it is not in 3NF and there in full functional dependence hence it is in 2NF.

$R_3 = \{D, I, J\}$, $\{D\} \rightarrow \{I, J\}$

Here, D is a super key, Hence R_3 is in BCNF

Q2. Given a set of FDS for the relation schema R (A,B,C,D) with primary key of AB under which R is in 1NF but not in 2NF.

Ans.: Lets us consider the set of dependencies FD

$AB \rightarrow CD$ and $B \rightarrow C$

So, $AB^+ = \{A, B, C, D\}$

$B^+ = \{B, C\}$

$\{AB\}$ is a key for the relation R (A,B,C,D) since $AB \rightarrow ABCD$.

$\{AB\}$ in the candidate key as there is no other minimal key that can determine all the attributes of R.

- Now the FD, $B \rightarrow C$ becomes a partial dependency and hence violates the property of 2NF.
- Again $C \notin B$, that is $B \rightarrow C$ is not trivial FD
- B is not the superkey
- C is not part of some key of R.
- B is the proper subset of the key AB

Q.3. compute the closure of the FD for the relation R (A,B,C,D,E)

$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$.

Sol: $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$

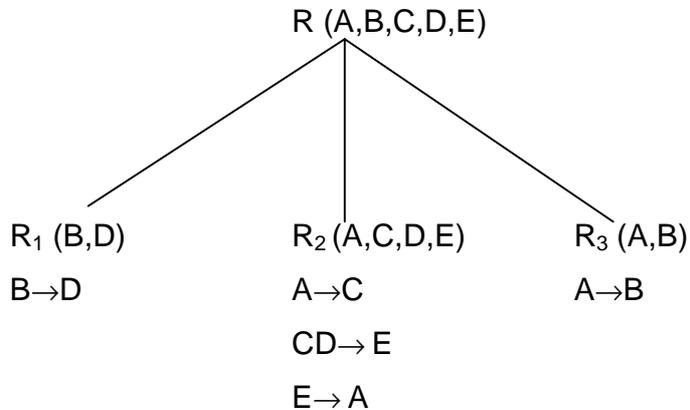
Cond.key: $A^+ = \{A, B, C, D, E\}$

$E^+ = \{A, E\} = \{A, B, C, D, E\}$

$CD^+ = \{A, B, C, D, E\}$

Closing $B=B^+=\{BD\} \leftarrow B$ is not a super key, so R is not in BCNF

(ii) Decompose the relation to obtain BCNF



Q4. Given a set of FDS for relation schema R (A,B,C,D) with primary key AB under which R is in 2NF but not in 3NF.

Ans. Let us consider the set of FDS: $AB \rightarrow CD$ and $C \rightarrow D$

$AB^+ = ABCD$ and $C^+ = CD$

Obviously, AB is the candidate key of this relation.

- Now there is no partial dependency
i.e. $C \not\subseteq AB \Rightarrow R$ is in 2NF
- But $C \rightarrow D$, violates the property of 3NF.

Since

- (i) CEC is false = so it is not trivial FD
- (iii) C is not a super key
- (iv) D is not part of some key of R

Q5. Consider the relation schema R (A,B,C) which has the FDS

$B \rightarrow C$, if A is the candidate key of R if it possible for R to be in BCNF ? if so under what condition. If not, why not ?

Ans. Yes, it is possible for R (A,B,C) to be in BCNF

Cond. : If there exist FDS, $A \rightarrow C$ and $A \rightarrow B$ and $B \rightarrow C$ there should no be or $C \rightarrow B$ any transitive dependency

Verification: For $A \rightarrow C$ and $A \rightarrow B$

A is the super key. Hence it is in BCNF

Q.6: Super we have a relation R (A,B) representations the relationship between two entity sets with key A and B and suppose that R has the FDS : $A \rightarrow B$ and $B \rightarrow A$ Explain what such a pair of dependencies means.

Ans.: Here FDS

$A \rightarrow B$ and

$B \rightarrow A$

- It means that the relation is one to one
- That is each value of entity A corresponds to one values of the entity B and vice versa.

Serializability

- A schedule S of n transaction is serialization if it is equivalent o some serial schedule fo the same on n transaction.
- All transactions are correct in the sense that if any one of the transaction is executed by itself on a consistent database, the resulting database will be consistent.
- Any serial execution of the transactions is also correct and preserves the consistency of the database, the results obtained are correct.

Ex. Schedule-I

T ₁	T ₂
Read (x)	
Writer (x)	Read (x)
	Write (x)
Write (x)	

T ₁	T ₂
Read (x)	
X=x-50	Read (x)
	Temp =AX0.1
Write(x)	A=A-temp
Read (y)	Write (x)
Y=Y+50	Read (Y)
Write(y)	
	Y=Y+temp
	Write (Y)

(a) Preserves consistency (b) Does not preserve consistency

Basic assumption: Each transaction preserves database consistency.

- Thus serial execution of a set of transactions preserves database consistency.
- A (Possible concurrent) schedule is serializable if it is equivalent to a serial schedule.
- Different forms of schedule equivalence gives rise to the notions of (1) Conflict serializability (2) View Serializability.

Simplified view of transactions:

- We ignore the operations other than read and write instructions .
- We assume that transactions may perform arbitrary computation on data in local buffers in between reads and writes.
- Our simplified schedules consist of any read and write instructions.

1. Conflict serializability

- If a schedule S can be transformed into a serial schedule by a series of swaps of non conflicting instructions, we say that S and S are conflict equivalent.
- We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.
- Schedule-1 can be transformed into schedule 2 , a serial schedule where T₂ follows T₁ by series of swaps of non-conflicting instructions.
- Therefore schedule 1 is conflict serializable.
- Ex. Schedule-1

T ₁	T ₂
Read (x)	
Writer (x)	Read (x)
	Write (x)
Read (Y)	
Write (Y)	Read (Y)
	Write (Y)

Schedule-1

T ₁	T ₂
Read (x)	
Writer (x)	Read (x)
Read (Y)	Writer (x)
Write (Y)	Read (Y)
	Write (Y)

Schedule-2

The example of a schedule that is not conflict serializable

T ₃	T ₄

Read (Q) Writer (Q)	Write (Q)
------------------------	-----------

Schedule-2

A serial schedule where T_2 is followed by t_1

T_1	T_2
Read (x) $X=X-50$ Write (X) Read (Y) $Y=Y+50$ Write (Y)	Read (x) $Temp=X+0.1$ $X=X-temp$ Write (X) $Y=Y+temp$ Write(Y)

Schedule-3: Let T_1 and T_2 be the transactions defined previously. The following schedule is not a serial schedule but is equivalent to schedule-1.

T_1	T_2
Read (x) $X=x-50$ Write(X) Read (y) $Y=Y+50$ Write(y)	Read (x) $Temp =a*0.1$ $Y=X-temp$ Write (x) Read (Y)

	Y=Y+temp Write (Y)
--	-----------------------

In all schedules, 1,2, and 3 the sum X+Y is preserved.

Schedules:

- A schedule is a sequence of instructions that specify the chronological order in which instructions of concurrent transactions are executed.
- A schedule for a set of transactions must consist of all instructions of those transactions.
- A Schedule must preserve the order in which the instructions appear in each individual transaction.
- A transaction that successfully completes its execution will have a commit instruction at the last instruction.
- A transaction that fails to successfully complete its execution will have an abort instruction as the last statement.

Schedule-1

Let T₁ transfers 50 from X to Y and T₂ transfers 10% of the balance from X to Y.

A serial schedule in which T₁ is followed by T₂

T ₁	T ₂
Read (x)	
X=x-50	
Write(X)	
Read (y)	Read (x)
Y=Y+50	Temp = a*0.1
Write(y)	Y=X-temp
	Write (x)
	Read (Y)
	Y=Y+temp

	Write (Y)
--	-----------

Rules for locking Technique by DBMS:

- A transaction T must issue the operation lock (x) before any read (X) or write (X) operations are performed in t.
- A transaction T must issue the operation unlock (X) after all read (X) and write (X) operations are completed in T.
- A transaction T will not issue a lock (X) operation if it is already holds lock on item X.
- A transaction T will not issue on unlock (X) operation unless it already holds the lock on item X.

Shared or Exclusive lock:

- The shared lock is also called a read mode of lock.
- The intention of this mode of locking is to ensure that the data item does not undergo any modifications while it is locked in this mode.
- Any number of transactions can concurrently lock and access a data item locked in a shared mode.

Exclusive mode: The exclusive lock is also called an update or write lock.

The intention of this mode of locking is to provide exclusive use of the data item to one transaction.

If a transaction T locks a data item Q in an exclusive mode no other transaction can access Q. not even read (Q)

Status	Unlock	Shared	Exclusive
Unlock		Yes	Yes
Shared	Yes	Yes	No
Exclusive	Yes	No	No

Concurrency control techniques:

- The goal of concurrency control is to develop protocols that will assure serializability and hence ensuring database consistency.

- A policy in which only one transaction can execute at a time generates serial schedules but provides a poor degree of concurrency.

Techniques of concurrency control.

- Locking : Some of the main techniques used in controlling concurrent execution of transactions are based on the concept of locking.
- A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it.
- Generally there is one lock for each data item in the database.
- Lock is used as a means of synchronizing the access by concurrent transactions to the data base items.
- We use locking to guarantee serializability of transaction schedules.

Types of locks:

There are several types of locks used in concurrency control.

(1) Binary locks (2) Shared /Exclusive (Read/Write)

(1) Binary locks:

A binary lock can have two states or values

Locked -1

Unlocked -0

- If the value of the lock 'X' is 1, item X can not be accessed by a database operation that request the item.
- If the value of the lock on 'X' is 0, item X can be accessed when requested.

Ex. B : Lock (x)=0 (Unlocked)

Lock (X)=1 (Locked item)

Go to B

End.

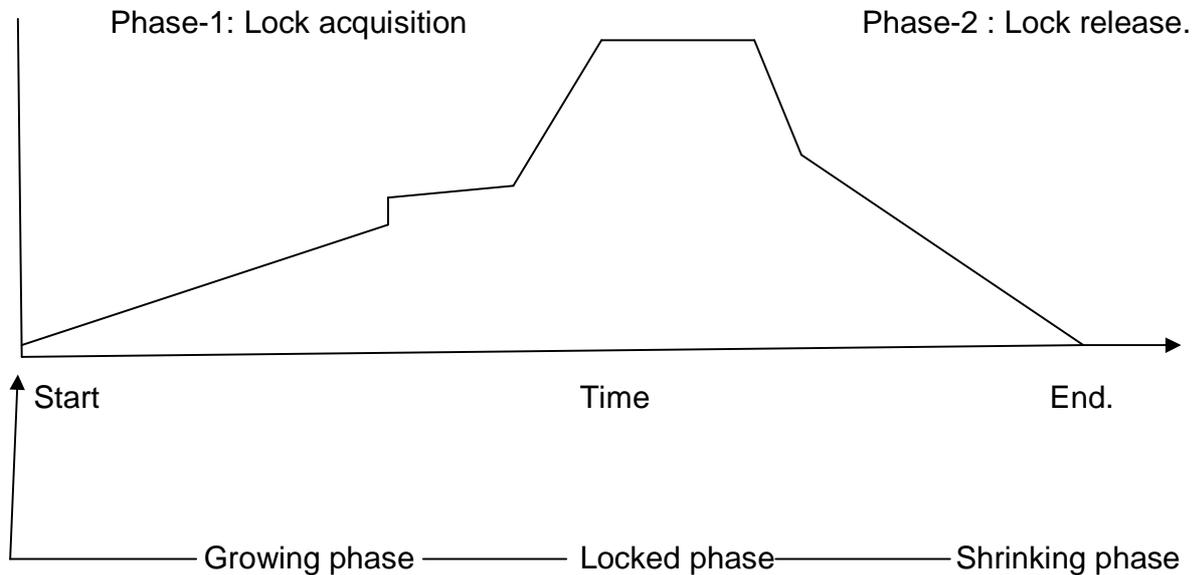
Two phase locking (2PL)

- Two phase locking is a method or a protocol of controlling concurrent execution of transactions (Processing) in which all locking operations precede the first unlocking operations.
 - A transaction is said to follow the two phase locking protocol if all locking operations (such as read lock or write lock) precede the first lock unlock operation in the transaction.
 - Two phase locking is the standard protocol used to maintain level 3 consistency.
 - 2PL defines how transaction acquire and releasing locks.
 - The two phase locking (2PL) has the two phases as (1) growing phase and (2) Shrinking phase
1. In the growing phase, the transaction acquires all the required data locks (on desired data) without unlocking any data.
Once all locks have been acquired, the transaction is in its locked point.
In this phase the lock is assigned to the transaction.
 2. In the shrinking phase (in which) a transaction released all locks and can not obtain any newlock.

Rules: The two phase locking is governed by the following rules.

- Two transactions can't have conflicting locks.
- No unlock operation can precede a lock operation in the same transaction.
- No data are affected until all locks are obtained, that is until the transaction is in its locked point.

Somatic of two phase locking (2 PL)



- In growing phase the number of locks increases from zero to the maximum for the transaction.
- In shrinking phase the no. of locks are decreased from maximum to zero.
- Both these phases are monotonic. The no of locks are only increasing in the first phase and decreasing in the second phase.
- Once the transaction has started releasing locks, it is not allowed to request any further locks.

Strictly two phase locking:

As given by the semantics of two phase locking in fig.1 . in case of a strict two phase locking the inter leaving is not allowed

Fig.2 shows a strict two phase locking in which transaction T_1 would obtain on exclusive lock on A first and then Read and write A.

T_1	T_2
X (A)	
Read (A)	
Write A	Time

Fig.3 shows the example of strict 2PL with serial execution in which first strict locking is done, then transaction T2 would request on exclusive lock on A.

2PL guarantees serializability. But it (2PL) does not prevent dead locks therefore is used in conjunction with a dead lock prevention technique.

T ₁	T ₂
X (A)	
Read (A)	
Write A	
X (B)	
Read (B)	X (A)
Write B	Read (A)
Commit	Write A
	X (B)
	Read (B)
	Write B
	Commit

Transaction processing & concurrency control.

Transactions: A transaction is a logical unit of database processing that includes one or more database access operations.

- A transaction can be defined as an action or series of actions that are carried out by a single user or operations performed by application programs for accessing the contents of the database.
- A transaction is a program unit whose execution may change of the database.
- A transaction is a program unit whose execution may change the contents of database.
- Transactions execution preserves the consistency of database.
- A transaction is said to be atomic if each transaction could access the share data without interfering with the other transactions and whenever a transaction successfully completes its execution, its effect should be permanent.

Transaction operations: The operations such as retrieval , insertion, deletion and modification can be carried out by using transactions.

These operations are performed in the form of read/write.

1. Read (X) : R(X)

2. Write (X) : W (X)

1. Read (X) : Reads a data base item X in to a program variable Y.

2. Write (X): Writes the values of X the program variable Y in to the database item named X.

Example: T1

1. Read (X)

2. X=X-50

3. Write (X)

4. Read (Y)

5. Y=Y-50

6. Write (Y)

Issues of transaction management:

- Failure

- Concurrent execution.

1. **Failure:** There are various kinds of failures.

- System crash / Hardware failure.

- Transaction or system error

- Local error or execution condition detected by transaction

- Dist failure

- Physical problems and catastrophes.

- Concurrency control enforcement.

2. **Concurrent execution:**

- Inconsistencies caused by conflicting updates from concurrent users.

Example:

T₁

T₂

1. Read (X)

2. X=X-50

3. Write (X)

Read (X) , Read(Y), Print (X+Y)

4. Read (Y)

5. $Y=Y-50$

6. Write (Y)

- When the transaction T1 is executed and in between steps 3 and 6 of T1, if another transaction T2 is allowed to access the partially updated database, then it will cause inconsistency in the database, as the sum of X and Y will be less than it should be.

ACID properties of Transactions:

- A transaction is a unit of program execution that access or updates various data items.
- To preserve the integrity of data the database system must ensure the following properties, called ACID properties.
- These ACID properties called ACIDITY of a transaction, to ensure that a database remains in stable state after the transaction is executed.

Atomicity: A transaction is an atomic unit of processing .

The atomicity property of a transaction requires that all operations of a transaction are performed completely or not performed (the transaction is aborted)

Consistency: The consistency property of a transaction implies that if the database was in a consistent state before the start of the transaction, then on termination of a transaction, the database will also be in a consistent state.

Isolation: The isolation property of a transaction indicates that actions performed by a transaction will be isolated or hidden from the outside the transaction until the transaction terminates.

Durability: The durability property of a transaction ensures that the commit action of a transaction on its termination will be reflected in the database.

These changes must not be lost because of any failure.

States of transaction:

- A transaction which successfully completes its execution is said to be committed.
- Otherwise the transaction is said to be aborted.

A transaction can be in one of the following states:

Active: The initial state, after the transaction starts its operation.

Partially Committed: After the final statement has been executed.

Failed: After detecting that the normal execution can no longer, proceed.

Aborted: When the normal execution can no longer be performed . A transaction may be aborted when the transaction itself defeats an error during execution which it can recover from.

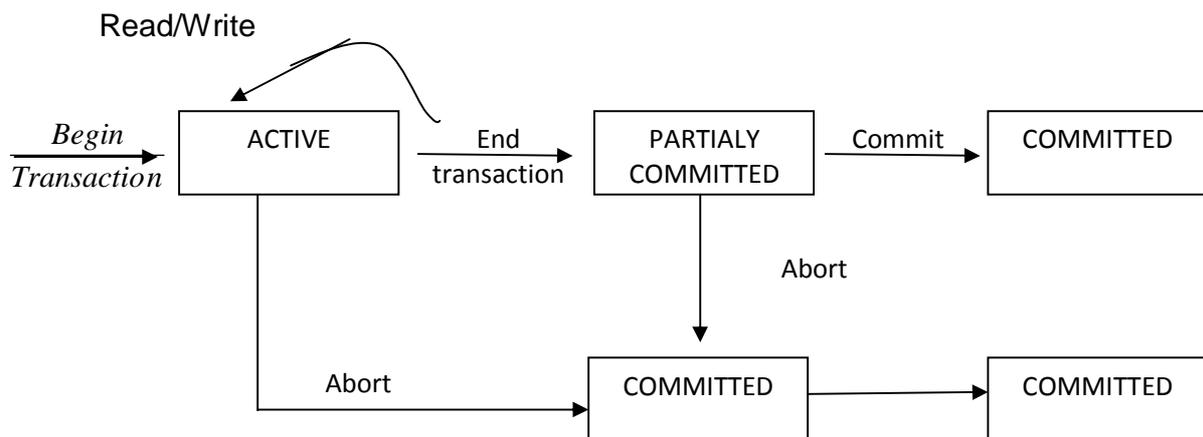
After the transaction has been rolled back and the database restored to its state prior to the start of the transaction.

There are two options after it has been aborted.

Restart the transaction (Only if no internal- logical error)

Kill the transaction.

Committed: After successful completion of the transaction . The transaction also said to be in a committed state if it has partially committed and it can be ensured that it will never be aborted.



(States of transaction)

Transaction execution with SQL:

- The ANSI has defined standards that govern SQL database transactions.
 - Transaction support is provided by two SQL statements. (a) Commit (b) Rollback
 - The ANSI standards require that when a transaction sequence is initiated by a user or an application program, it must continue through all succeeding SQL statement. Until one of the following events occur.
- (a) **Commit** : A commit statement is reached, in which case all the (transaction) changes are permanently recorded in the database.
- The commit statement automatically ends the SQL transaction.
 - The commit operations indicate successful end of transaction.
- (b) **The Rollback**: Statement is reached, in which all the change are aborted and the database is rolled back to its previous consistent state.
- The rollback operation indicates successful end of transaction.
- (c) The end of program is successfully reached, in which case all changes are permanently recorded within the database. The action is equivalent to Commit.
- (d) The program is abnormally terminated; in which case all the changes made in the database are aborted and the database is rolled back to its previous consistent state. This action is equivalent to rollback.

Ex.: Let us consider the example of commit, which updates an example loan balance (Emp-loan-bal) and the projects cost (Proj-Cost) in the tables employee and project respectively.

```
UPDATA    EMPLOYEE
          SET EMP-LOAN-BAL = EMP-LOAN-BAL-10000
          WHERE EMP-ID= 106519
```

```
UPDATA    PROJECT
          SET PROJ-COST = PROJ-COST +40000
          WHERE PROJ-ID= PROJ-1.
```

```
COMMIT;
```

Time stamp ordering: (Time stamp protocol)

- Time stamp is a unique identifier created by the DBMS to identify a transaction.
- In this scheme, the transaction manager assigns unique time stamp $TS(T)$ to each transaction.
- The idea for this scheme is to order the transaction based on their time stamp.
- In the time stamp based method a serial order is created among the concurrent transaction by assigning each of the transaction a unique non decreasing number.
- The usual value assigned to each transaction is the system clock time value at start of the transaction.

Time stamp ordering rules: The following rules are checked when transaction T attempts to change a data item.

If the rule indicates ABORT, then transaction T is rolled back and aborted.

Rule-1: If T attempts to read a data item which has already been written to by a younger transaction then ABORT T.

Rule-2: If T attempts to write a data item which has been seen or written to by a younger transaction then ABORT T.

If a transaction t aborts, then all other transactions which have seen a data item written to by T must also abort.

In addition other aborting transactions can cause further aborts on other transactions.

This is a cascading roll back.

The time stamp ordering algorithm:

- The idea for this scheme is to order the transactions based on their time spaces/ stamps.
- A schedule in which the transactions participate in the serializable and the equivalent serial schedule has the transactions in order of their time stamp values.
- This is called time stamp ordering.
- The algorithm must ensure that for each item accessed by conflicting operations the schedule, the order in which the item is accessed, the item is accessed does not violate serializability order.
- In this algorithm, the item X is accessed with two time stamp values (TS)

- (i) Read-Ts (X): The read time stamp of item X; this is the largest time stamp among all the time stamp of transaction that have successfully read item X.
i.e. $Read\ Ts(X) = TS(T)$, where T is youngest transaction that has read X successfully.
- (ii) Write-Ts (X) : The entire time stamp of item X , this is the largest time stamp among al the time stamp of transaction that have successfully written item X, that is $write-TS(X) = TS(T)$, where T is youngest transaction that has written X successfully.

Basic time stamp ordering (Basic To)

- Whenever some transaction T tries to issue a read (X) or write (X) operation, the basic time stamp ordering algorithm compares the time stamp of T with read- $Ts(X)$ and write $-Ts(X)$ to ensure that the time stamp order of transaction execution is not violated.
- If this order is violated them transaction T is aborted and submitted to the system as a new transaction with a new time stamp.
- If T is aborted and rollback, any transaction T1 that may have used a value written by T must be rollback.
- The effect is known as cascading rollback.
- The concurrency control algorithm must check whether conflicting operations violate the timestamp ordering in the following two cases.
 - Transaction T issues a write-item(X) or write (X).
 - (a) If $read-Ts(X) > Ts(T)$ or if $write -Ts(X) > Ts(T)$, then abort and rolled back T and reject the operation.
 - (b) If the condition (a) is not satisfied them execute write(X) operation of T and set write $Ts(X)$ to $Ts(T)$
 - Transaction T issues a read (X).
 - (c) If $read-Ts(X) > Ts(T)$, then abort and rolled back T and reject the operation.
 - (d) If $read-Ts(X) > Ts(T)$, then abort and rolled back T and reject the operation. Of T and set read $Ts(X)$ to $Ts(T)$ the current read $Ts(X)$ which ever is larger.

Thomas write rule: A modifications of basic time stamp ordering algorithm is known a stomas write rule.

- It does not enforce conflict serializability, However.
- It rejects fewer write operations by modifying the check for the write (X) operation as follows.
 1. If $\text{read-Ts}(X) > \text{Ts}(T)$, then abort and rollback T and reject the operation.
 2. IF $\text{write Ts}(X) > \text{Ts}(T)$, then do not execute write operation but continue processing operation, this is because some transaction with time stamp ordering has already written the value of X, hence we must ignore the write (X) operation of T because it is already, outdated and absolved.
 3. If neither the condition in part (1) nor the condition in part (2) occurs, then execute the write-item (x) operation of T and set $\text{write-Ts}(X)$ to $\text{Ts}(T)$.

Strict Time stamp ordering: A variation of basic Time stamp ordering is called strict time ordering that ensures that the schedules are both strict for easy recoverability and conflict serializable.

In this variation a transaction T that issues read (X) or write (X) such that $\text{Ts}(T) > \text{write-Ts}(X)$ has its read or write operation delayed until the transaction T that write the value of X has committed or aborted. To implement this algorithm it is necessary to simulate the locking of an item X that has been written by transaction T until T is either committed or aborted.

This algorithm does not cause deadlock , since T waits T only for $\text{Ts}(T) > \text{Ts}(T)$.

Multiversion concurrency control technique:

- In this technique multiple versions (values) of an item are maintained i.e. the old and updated value of a data item is kept.
- When a transaction requires access to an item, an appropriate version is chosen to maintain the serializability of the currently executing schedule.
- When a transaction writes an item it writes a new version and the old version of the item is retained.
- Some multiversion concurrency control algorithms use the concept of view serializability rather than conflict serializability.
- The main drawback of multiverison technique is that more storage spaces are needed to maintain multiple versions of the data items.

- The order versions may have to maintained any way for recovery or other purposes.

Optimistic Concurrency Control:

- In optimistic concurrency control techniques no. checking is done before the transaction is executed.
- This also called validation or certification techniques.
- In this technique, updates in transaction are not applied directly to the database items until the transaction reaches its end.
- During transaction execution, all updates are applied to local copies of the data items that are kept for the transaction.
- After the transaction is executed, a validation phase checks whether any of the transactions updates violates serializability.
- If serializability is not violated, the transaction is committed and the database is updated form the local copies; otherwise the transaction is aborted and then restarted later.

There are three phases for this concurrency control protocol.

1. **Read phase:** A transaction can read values to the committed data items from the database. However updates are applied only to local copies (versions) of the data items kept in the transaction work space.
2. **Validation phase:** Checking is performed to ensure that serializability will not be violated if the transaction updates applied to the database.
3. **Write phase:** If the validation phase is successful, the transaction updates are applied to the database; otherwise , the updates are discarded and the transaction is restarted.

This technique is called optimistic because they assume that little interference will occur and hence there is no need to do checking during transaction execution.

In validation phase for transaction T_i , the protocol check that T_i does not interfere with any committed transaction or with any other transactions currently in their validation phase.

The validation phase for T_i checks that for each such transaction T_j that is either committed or is in its validation phase, one of the following condition holds.

1. Transaction Tj completes its write phase before Ti starts its read phase.
2. Ti starts its write phase after Tj completes its write phase and read-set of Tj has no items in common with the write – set of Tj.
3. Both the read set and write set of Ti have no items in common with the write set of Ti and Tj completes its read phase before Ti completes its read phase.

When the validating transaction Ti, the first condition is checked first for each transaction Tj, since condition (i) is the simplest condition to check.

Only if the condition (i) is false condition (ii) checked and if (ii) is false is condition (iii) the most complex to evaluate checked. If one of these conditions holds there is no interference and Ti is validated successfully.

If non of these three conditions holds, the validation of transaction Ti fails and it is aborted and restarted later because interference may have occurred.

Recovery techniques:

Deferred update: Deferred update or No UNDO/REDO is an algorithm to support ABORT and machine failure.

- While a transaction runs, no changes made by the transaction are recorded in the database.
- On a commit:
- The new data is recorded in a log file and flashed to disk
- The new data is then recorded in the database itself.
- On as abort, do nothing (the database has not been check)
- On a system restart after a failure , REDO the log.

Example: Let us consider the following transaction Ti

T₁
Read (A)
Write(10,B)
Write (20,C)
Commit

Using deferred update , the process is:

Time	Action	Log
T ₁	Start	-
T ₂	Read(A)	-
T ₃	Write(10,B)	B=10
T ₄	Write(20,C)	C=20
T ₅	Commit	Commit

	Before	After
Disk	B=6 A=5 C=2	B=10 A=5 C=10

If the DBMS is failed and is restarted

- The disks are physically or logically damaged then recovery from the log is impossible and instead a restore from a dump is needed.
- If the disks are ok then the database consistency must be maintained. Writes to the disk which was in progress at the time of failure may have only been partially done.
- Parse the log file and where a transaction has been ended with COMMIT apply the data part of the log to the database.
- If a log entry for a transaction ends with any thing other than commit , do nothing for that transaction.
- Flush the data to the disk and then truncate the log to zero.
- The process or reapplying transaction form the log is some times referred to as roll forwarded.

Immediate Update:

Immediate update or UNDO/REDO, is another algorithm to support ABORT and machine failure scenarios.

- While a transaction runs, changes made by that transaction can be written to the database any time . However, the original and new data being written must both be stored in the log before storing it on the database disk.
- On a commit
- All the updates which is not yet been recorded on the disk is first stored in the log file and then flushed to disk.
- The new data is then recorded in the database itself.
- On an abort, REDO all the changes which that transaction has made to the database disk using the log entries.
- On a system restart after a failure , REDO committed changes from Log.

Example: Using immediate update and the transaction T_1 again , the process is:

Time	Action	Log
T_1	Start	-
T_2	Read(A)	-
T_3	Write(10,B)	Was B=6 now10
T_4	Write(20,C)	Was C=2, now 20
T_5	Commit	Commit

	Before	During	After
Disk	B=6 A=5 C=2	B=10 A=5 C=2	B=10 A=5 C=20

Recovery: Database recovery from failure such as system crashes, transaction errors etc.

A database may be in inconsistent state when:

- Dead lock has occurred.
- A transaction is aborted after updating the database
- Software or hardware errors are occurred.
- In correct updates have been applied to the database.

If the database is in inconsistent state, it should be recovered to a consistent state.

This basis of recovery is to have back ups of the data in the database.

Types of recovery:

- **Dump:** A full backup of the database. In this method .
 - The entire contents of the database is backed up in an auxiliary storage
 - Backup is taken only at the consistent state of the database
 - Dumping takes a long time to perform
 - Dumping is expensive and can be performed repeatedly
- **Transaction log: (Journal)**
 - A log is a sequence of log records recording all the update activities in the database since the last consistent state.
 - Database can have state before and after each transaction (information)
 - When the database is returned to a consistent state, the log may be truncated to remove committed transaction.
 - When the database is returned to a consistent state the process is often referred to as check pointing.
- **Checkpoint:** Checkpoint is a scheme, which is used to limit the volume of log information that has to be handed and process in the event of a system failure involving loss of volatile information.
 - It is performed periodically. It performs the following sequence of actions to take place.
 - Output all log records currently residing in main memory into stable storage.
 - Output all modified buffer blocks to the disk
 - Write a log record <check point> on to stable storage.

Recovery using check point:

During recovery we need to consider only the most recent transaction T_i that started before the check point and transaction that started after T_i .

- Same backwards from end of log to find the most recent <check point> record.
- Continue scanning backwards to a record < T_i start> is found
- Need only consider the part of log following above start record.
- Earlier part of part of log can be ignored during recovery and can be erased whenever desired.

- For all transactions (starting from T_i or later) with no $\langle T_i \text{ commit} \rangle$, execute undo T_i . Done only in case of immediate modification.
- Canning forward in the log, for all transactions starting from T_i or later with a $\langle T_i \text{ commit} \rangle$ execute redo (T_i)

IF the DBMS fails and restarted:

- The disks are physically or logically damaged then the recovery from log is impossible and instead a restore from a dump is needed.
- If the disks are ok then the database consistency must be maintained. Writes to disk which was in progress at the time of failure may have only been partially done .
- Parse the log file and where a transaction has been ended, with commit apply the new data part of the log to the database.
- IF a log entry for a transaction ends with any thing other than commit, apply the old data party the log to the database.
- Flush the data to the disk and then truncate the log to Zero.

Roll Back

The process of undoing changes done to the disk under immediate update is frequently referred to as rollback.

- Where the DBMS does not prevent one transaction from reading un committed modifications (a dirty read) of another transaction (i.e. the uncommitted dependency problem,) then aborting the first transaction also means aborting all transactions which have performed their dirty reads.
- As a transaction is aborted it can therefore cause aborts in other dirty reader transactions, which in turn can cause other aborts in other dirty reader transaction.

This is referred to as cascade roll back.

What is an index ?

Index: An index is a data structure or table, which is maintained to determine the address (location) of rows (Records) in a file that satisfy some conditions.

- An index table consists of the value of the key attribute for a particular record and the pointer to the location where the record is stored.
- The record is retrieved from the disk first by searching the index for the address of the record in the index table and then reading the record from this address.

What are the uses of indices ?

- An index on a file speeds up selection on the search key fields for the index.
- An index contains a collection of data entries and supports efficient retrieval of data entries with a given key value .
- Updating a particular record is fast.
- Inserting records is also fast.

What are different types of indexing ? Indices ?

There are two basic kinds of indices

- (1) Ordered indices: Based on sort ordering of the values.
- (2) Hash indices: Based on the values being distributed uniformly across a range of buckets.

The bucket to which the value is assigned is determined by a function called a hash function.

Types of ordered indices: There are two types of ordered indices.

- (1) **Primary index:** The file in which the records are sequentially ordered, the index whose search key specifies the sequential order of the file is called the primary index.

The primary index is called also clustering index.

A primary index is an ordered file whose records are of fixed length with two fields. The first field is the same data type as the ordering key field (primary key) and the second field is the pointer to a disk block (a block address)

There is one index entry in the index file for each block in the data file.

- (2) **Secondary index:** An index whose search key specifies an order different from the sequential order of the file is called secondary index or no clustering index.

A secondary index is also an order file with two fields . The first field is of some data type as some non ordering field of the data file that is an indexing field. The second field is either a block pointer or record pointer.

The secondary index is used to search a file on the basis of secondary keys.

(3) **Dense and sparse indices:** These are the ordered indices that we can use.

Dense index: An index record (or index entry) appears for every search key value in the file

The index record contains the search key value and pointer to the first data record with that search key value.

Sparse index: An index record is created for only some of the values. Each index record contains a search key value and a pointer to the first data record for that search key value.