FILE STRUCTURE

File Structure

Record storage and primary file organization: - In computerized system, database must be physically stored on some computer storage medium. Computer storage media form a storage hierarchy that includes two main categories: -

1. Primary storage: - This storage media can be operated on directly b the CPU. It is fast but has limited storage capacity.

2. Secondary storage: - It includes magnetic disks, tapes and drums. These have usually large storage capacity, lower cost and provide slower access.

Storage of Databases: - Databases typically store large amounts of data that must persist over long periods of time. Most databases are stored permanently on disk secondary storage for the following reasons:

1. Generally, databases are too large to fit entirely in main memory
2. Secondary memory is permanent
3. Cost is less

Placing file records on disks: -

Record and its types: - Data is usually stored in the form of records. Each record consists fo a collection of related data values or items. Records usually describe entities, their attributes, and their relationships. A collection of field names and their corresponding data types constitutes a record type or record format definition.

File: - A file is a sequence of records. All records in a file should be of the same record type. If every record in the file has exactly the same size, the file is said to be made up of fixed-length records. If different records in the file may have different sizes the file is made up of variable-length record.

Fixed-length record: - When every fields of a record occupy fixed memory size that can not be changed from record to record, is called fixed-length record. Main problem with this type of record is that for some record memory will be wasted or some time memory will be overflow.

Variable-length record: - In this type of record, size of every field will be changed according to the data of that field. Therefore, length of each record may vary. As the field size is not fixed, so a delimiter is used to mark the end of the field.

Record blocking: - Block is the unit of data transfer between disk and memory. Records of a disk file must be allocated to disk blocks. When the block size is larger than the record size, each block may contain numerous records.

Suppose the block size is B bytes, record size is R bytes (B>=R), we can fit $bfr = \lfloor (B/R) \rfloor$ records per block. The value bfr is called the blocking factor for the file. In general R may not divide B exactly, so we have some unused space in each block equal to B – (bfr * R) bytes.

If this unused space is less than the space required for a record, then, we can store one part of a record in this space and remaining part in another block. A pointer should be placed in the first block to refer to the block where the remaining part of the record is stored. This type of organization of record is called spanned organization. This is generally used with variable-length record. If we do not allow records to cross block boundaries, the organization is called unspanned. This is often used with fixed-length record.

Allocating file blocks on disk: - There are several standard techniques for allocating the blocks of a file on disk.

i) Contiguous allocation: - In contiguous allocation the file blocks are allocated to consecutive disk blocks. This make reading the whole file very fast but make it difficult to expand the file.

ii) Linked allocation: - In this allocation each file block contains a pointer to the next file block. This makes it easy to expand the file but makes it slow to read the while file.

iii) Clusters allocation: - It is the combination of above to allocation. It allocates clusters of consecutive disk blocks and the clusters are linked together.

iv) Index allocation: - In this allocation, one or more index blocks contain pointer to the actual file blocks.

Operation on file: - Different types of operation applied on a file are:

i) Find: - Search the first record satisfying a search criterion.

ii) Read: - Retrieve records

iii) Delete : - Remove existing record

iv) Modify: - Update some fields of one or more records

v) Insert: - Enter new record into the file.

OVERVIEW OF DATABASE MANAGEMENT SYSTEM

Database : A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. A database has the following implicit properties

1. A database is a logically coherent collection of data with some inherent meaning.
2. A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived application in which these users are interested.
3. A database represents some aspect of the real world.

A database can be of any size and of varying complexity. A database may be generated and maintained manually or by machine.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. A DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing and manipulating databases for various applications. The database and software are together called a database system.

Use of DBMS (Advantage of DBMS):-

1. Controlling Redundancy: - Redundancy means storing of same information in multiple time. It leads to several problems like performing single logical update many times, wastage of memory space and inconsistency. To avoid all such problem we should have a database design that stores each logical data item in only one place in the database.

2. Sharing of data :- The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.

3. Restricting Unauthorized Access: - When multiple users share a database, it is likely that some users are not authorized to access all information in the database. In addition some user may be permitted only to retrieve data, whereas others are allowed to both retrieve and update. Hence the type of access operation can also be controlled. A DBMS should provide a security and authorization subsystem, which is used by the DBA to create account and specify account restrictions.

5. <u>Providing Multiple Interfaces</u>:- Because many types of users, with varying technical knowledge, use a database, a DBMS should provide a variety of user interfaces. The types of interfaces include query languages for casual users, programming language interfaces for application programmers forms for parametric users, menu-driven interfaces for native users and natural language interfaces.

6. <u>Representing Complex Relationships among Data</u>: - A database may include a variety of data that are interrelated in many ways. A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data in easy and efficient manners.

A7. <u>Enforcing Integrity Constraints</u>: - Most database applications will have certain integrity constraints like data type of data item, uniqueness constraint of data item, relationship between records of different file etc. These integrity constraints must be specified during the database design.

8. <u>Provide Backup and Recovery</u>: - A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.

<u>DBMS users</u>: - Many persons are involved in the design, use and maintenance of a large database. Different types of DBMS users are:

1. Database Administrator(DBA): -
2. Database Designer
3. End users

<u>Three-Level architecture of DBMS</u>: - The goal of the three-level architecture is to separate the user applications and the physical database. In this architecture, level or schema can be defined at the following three levels

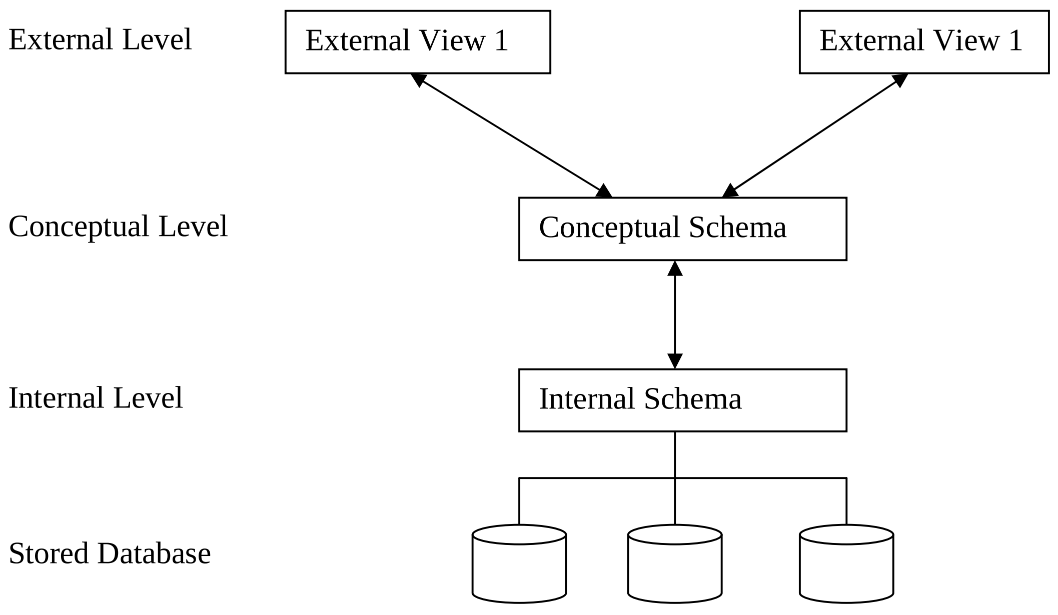

Fig: Three level architecture of DBMS (ANSI-SPARC)

1. Internal Level : - The internal level has a internal schema, which describes the physical storage structre of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. Conceptual Level: - The conceptual schema or level describes the structure of the whole database for a community of users. The conceptual schema hides the4 details of physical storage structures and concentrates on describing entities, data types, relationships, user operations and constraints.
3. External or view level: - The external level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from the user group.

Data Independence: - Data independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. There are two types of data independence :

1. Logical data independence: - It is the capacity to change the conceptual schema without having to change external schemas or application programme. We may change the

conceptual schema to expand the database, to change constraints, or to reduce the database.

2. Physical data independence: - It is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files had to be reorganized.

DBMS Languages: - The DBMS must provide appropriate languages and interfaces for each category of users. Different types of DBMS languages are:

1. Data Definition Language (DDL): - It is used by the DBA and by database designers to define both schemas. The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schemas constructs and to store the schema description in the DBMS catalog.
2. Data Manipulation Language (DML): - Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion and modification of
3. the data. The DBMS provides a set of operations or a language called the data manipulation language (DML) for these purposes.

Traditional Data Models: - A data model is a set of concepts that can be used to describe the structure of database. By structure of a database, we mean the data types, relationships, and constraints that should hold on the data. Most data models also include a set of operations for specifying retrievals and updates on the database. There are three different types of data model

a) Relational b) Hierarchical c) Network

Relational Data Model: - The Relational Data Model represents a database as a collection of tables, which consist of rows and columns. In relational database terminology, a row is called a tuple, a column name is called an attribute and table is called a relation. Most relational databases have high-level query languages and support a limited form of user views.

Hierarchical Data Model: - In the hierarchical model there are two main data structuring concepts : records and parent-child relationship. Records of the same type are grouped into record types. A parent-child relationship type (PCR type) is a 1:N relationship between two record types. The record type on the 1-side is called the parent record type and the one on the N-side is called the child record type of the PCR type. Properties of a Hierarchical schema are

a) One record type, called the root of the hierarchical schema, does not participate as a child record type in any PCR type.
b) Every record type except the root participates as a child record type in exactly one PCR type
c) A record type can participate as parent record type in any number of PCR types
d) A record type that does not participate as parent record type in any PCR type is called a leaf of the hierarchical schema

If a record type participates as parent in more than one PCR type, then its child record types are ordered. The order is displayed, by convention, from left to right in a hierarchical diagram.

Entity Relationship Model (ER Model): - At the present time, the ER model is used mainly during the process of database design.

1. Entities: - The basic object that the ER model represents is an entity, which is a "thing" in the real world with an independent existence. An entity may be an object with a physical existence – a particular person, car etc, or it may be an object with a conceptual existence like a company, a job or a university course etc.
2. Weak Entity: - Some entity may not have any key attributes of their own. This implies that we may not be able to distinguish between some entities because the combinations of values of their attributes can be identical. Such entity is called weak entity. Weak entity is identified by being related to specific entities from another entity type in combination with some of their attribute values. Weak entity always has a total participation constraint with respect to its identifying relationship. Weak Entity type has a partial key, which is the set of attributes that can uniquely identify weak entities related to the same owner entity.
3. Attribute: - Each entity has particular properties called attributes that describe it. For example a student entity may be described by RollNo, Name, Class, Address etc. Different types of attributes are

   a) Composite Attribute: - An attribute, which is composed of more basic attributes, is called composed attribute. For example Address attribute of a student.

b) Atomic Attribute: - The attributes that are not divisible are called simple or atomic attributes. For example RollNo attribute of a student.

c) Single-valued Attribute: - Most attributes have a single value for a particular entity, such attributes are called single-valued attribute. For example Date_of_Birth attribute of a person.

d) Multivalued Attribute: - The attribute, which has a set of values for the same entity is called multivalued attribute. A multivalued attribute may have lower and upper bounds on the number of values for an individual entity. For example Subject attribute of a student.

e) Derived Attribute: - In some cases two or more attribute values are related. The value of one attribute has to be calculated from the value of another attribute. Such type of attribute is called derived attribute. For example Age attribute of a person can be calculated from current date and his date of birth.

f) Key Attribute: - An entity type usually has an attribute whose values are distinct for each individual entity. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely. Sometimes several attributes together can form a key, meaning that the combination of the attribute values must be distinct for each individual entity. Some entity types have more than one key attribute. In this case, each of the keys is called a candidate key. When a relation schema has several candidate keys, the choice of one to become primary key is arbitrary, however, it is usually better to choose a primary key with a single attribute or a small number of attributes.

3. Degree of a Relationship Type: The degree of a relationship type is the number of participating entity types. A relationship type of degree two is called binary and of degree three is called ternary.

4. Constraints on Relationship Types: - There are two types of constraints on relationship types.

a) Cardinality Ratio: - The cardinality ratio constraints specify the number of relationship instances that an entity can participate in. Different ratios are 1:1   1:N   and M: N

b) Participation constraint: - Participation constraint specifies whether the existence of an entity on its being related to another entity via the relationship type. There are two types of participation constraints, total and partial.

<u>Relational Algebra</u>: - Relational algebra is a collection of operations on relation. Each operation takes one or more relations as its operand and produces another relation as its result. These operations are used to select tuples from individual relations and to combine related tuples from several relations for the purpose of specifying a query, a retrieval request, on the database. Different types of operation in relational algebra are

1. <u>SELECT operation</u>: - The algebraic SELECT operator yields a horizontal subset of a fiven relation that is subset of tuples for which a specified predicates is specified. The predicate is expressed as a Boolean combination of terms, each term being a simple comparison that can be established as tru or false for a given tuple by inspecting that tuple in isolation.

   The SELECT operator is unary that is it is applied on a single relation. Hence SELECT cannot be used to select tuples from more than one relation. The number of tuples in the resulting relation is always less that or equal to the number of tuples in the original relation.

2. <u>PROJECT operation</u>: - The PROJECT operator yields a vertical subset of a given relation, that is that subset obtained by selecting specified attribute in a specified left to right order and then eliminating duplicate tuples within the attributes selected. The number of tuples in a relation resulting from a PROJECT operation will be les than or equal to the number of tuples in the original relation.

3. <u>JOIN</u>:- The JOIN operation is used to combine related tuples from two relations into single tuples. The result of the JOIN operation is a relation Q with m+n attributes that is m attributes from first relation and n attributes from another relation. The tuples in the relation resulting from a JOIN operation are those, which will satisfy the condition given in the join operation. Different types of JOIN operation are :

a) <u>Theta join</u>: - A join condition is of the form

<Condition> AND <condition> AND………. AND <condition> where each condition is of the form $A_i \theta B_j$. A_i is an attribute of R. B_j is an attribute of S, A_i and B_j has the same domain and θ is one of the comparison operators { =, <, <=, >, >=, #}. A

join operation with such a general join condition is called a theta join.

b) Equi Join: - The most common join operation involves join condition with equality comparisons only. Such a join where only comparison operator used is equal sign is called an Equijoin. The relation produced by Equijoin, contain one or more pairs of attributes that have identical values in every tuple because the equality join condition is specified on these two attributes.

c) Natural join: - It is basically an equijoin but it eliminates the duplicate attribute in the result. It is denoted by *.

d) Outer join: - Generally Join operation select all the tuples from the two relation which will satisfy the join condition. But outer join is used to keep all tuples in R or S or both in the result, whether or not they have matching tuples in the other relation. Different types of outer joins are

i) Left outer join: - The left outer join operation keeps every tuples in the first or left relation R in R S. If no matching tuple is found in S, then the attributes of S in the result are filled with null values.

ii) Right outer join:- Right outer join( ) keeps every tuple in the second or right relations S in the result of R S.

iii) Full outer join:- The full outer join( ) keeps all tuples in both the left and right relations when no matching tuples are found, assigning them with null values as needed.

e) SET operation:- Set operations are the standard mathematical operation on sets. They apply to the relational model because a relation is defined to be a set of tuples and they are used whenever we process the tuples in two relation as sets. Set operations are binary that is they are applied to two sets. The two relation on which these operations are applied, must be union compatib le. Union compatible means the two relation must have the same type of tuples. There are three set operations :

i) UNION: - The union of two relations A and B, A UNION B is the set of all tuples t belonging to either A or B (or both). Duplicate tuples are eliminated

ii) INTERSECTION: - The intersection of two relations A and B, A INTERSECTION B, is the set of all tuples t belonging to both A and B.

iii) DIFFERENCE: - The difference between two relations A and B, A MINUS B, is the set of all tuples t belonging to A and not to B

iv) Cartesian Product: - Cartesian product is a binary set operation but the relations on which it is applied do not have to be union compatible. This operation is used to combine tuples from two relations so that related tuples can be identified. In general, the result of R(A1,A2,…….An) X (B1, B2…….Bm) is the relation Q with (n+m) attributes Q(A1, A2…….An, B1, B2…….Bm) in that order. The resulting relation Q has one tuple for each combination of tuples one from R and one from S. The Cartesian product creates tuples with the combined attributes of two relations. We can select only related tuples from the two relations by specifying an appropriate selection condition.

Structured Query Language (SQL): - SQL is a comprehensive database language; it has statements for data definition, query, and update. Hence, it is both a DDL and DML. It is a very powerful language in the sense that most of the operations in RDBMS can be performed using SQL. An important feature of SQL is that it is a non-procedural language. In a non-procedural language, we have to describe what to do rather than how to do.

Advantage of SQL: -

Data types in SQL: -

Data Type	Specification	Description
Char	CHAR (size)	Char data, size if specified in number
Varchar	VARCHAR(size)	Same as above
Date	DATE	Used for specifying dates
Number	NUMBER	Used to specify numeric data, NUMBER(size) or NUMBER(size, dec) number with digits equal to specified size.
Integer	INTEGER	Same as number but without decimal digit
Raw	RAW(size)	Raw binary data
Long raw	LONG RAW	Raw large binary data
Blob, Clob		Large binary and character data

Bfile		External files

Different SQL commands are: -

1. <u>CREATE TABLE command</u>: - This command is used to specify a new relation by giving it a name and specifying each of its attributes. Each attribute is given a name, a data type to specify its domain of values, and some constraints on the attribute. The syntax is

```
CREATE TABLE tablename (attributename   datatype    constraint,
                        attributename   datatype    constraint,
                        -  -   -   -   -   -   -   -   -   -   -   - ) ;
```

for example we can create a student table as follows

```
create table student (  Roll      number(3)   primary key,
                        name    varchar(30)  not null,
                        class     varchar (10) );
```

2. <u>DROP TABLE command</u>: - We can delete the table or relation and its definition using the DROP TABLE comand

```
DROP TABLE tablename ;
```

We can drop the student table as DROP TABLE student ;

3.<u>ALTER TABLE command</u>: - To add attributes to an existing relation, we can use ALTER TABLE command.

```
ALTER TABLE tablename ADD attributename datatype ;
```

We can add an extra attribute RESULT to the student table as follows:

```
ALTER TABLE student ADD result varchar(10) ;
```

4. <u>INSERT command</u>: - INSERT command is used to add a single record or tuple to a relation. If we want to insert record consisting of values of all attributes then no need to specify the attribute name, otherwise specify the name of those attributes for which we want to insert values.

INSERT INTO tablename VALUES(value1, value2, ……. Value n)

Or

INSERT INTO tablename(attribute1, attribute2,…….. attribute n) VALUES(value1, value2,………value n)

For example add a new record in student table

INSERT INTO student VALUES(1, “AAAA”, “B.Sc. 1st Year”, “1st class”);

5. <u>DELETE command</u>: - The DELETE command removes tuples from a relation. It includes a where-clause, to select the tuples to be deleted. Tuples are deleted from only one table at a time. A missing where-clause specifies that all tuples in the relation are to be deleted.

DELETE FROM tablename

Or

DELETE FROM tablename WHERE expression;

Example DELETE FROM student

Or DELETE FROM student WHERE roll = 2;

<u>UPDATE command</u>: - The UPDATE command is used to modify attribute values of one or more selected tuples. Where-clause can be used to specify the tuples to be modified from a single relation. The syntax is

UPDATE tablename

SET attributename= value

WHERE expression ;

UPDATE student SET result =”1st class” WHERE roll = 3 ;

<u>SQL Operators and their precedence</u>: - SQL operators are used to perform some functions that

would have been possible only with SQL. The operators are of various types such as value operator, logical operator.

1. Value operator: -

   a) Operator ( ) : - It is used to override normal precedence rule.
   b) Operator +, - : - It is used to give sign for a number expression
   c) Operator *, / :- Multiplication and Division
   d) Operator +, - :- Addition and Subtraction.
   e) Operator | | :- Concatenation character expression.

2. Logical operator : -

   a) Operator ( ) : - Used for overriding normal precedence rule.
   b) Operator = :- Tests for equality
   c) Operator != or < > : - Tests for inequality
   d) Operator >=, >, <, <= : - Compares two values
   e) BETWEEN : - Tests whether a certain value falls between two given values
   f) IN : - Tests the membership of an element in a set.
   g) NOT IN :- Complement of IN.
   h) ANY :- Compares a value with each value in a set and returns true if any value is compared according to given condition.
   i) ALL :- Compares a value with every value in the set and returns true if the given condition is satisfied for every value.
   j) EXISTS :- Tests whether a subquery returns at least one row. It is tru if it returns at least one row, otherwise it is false.
   k) NOT EXISTS :- Reverse of exists.
   l) LIKE :- Used in pattern matching
   m) NOT LIKE :- Reverse of Like

   n) IS NULL : - Tests whether the value of column is null
   o) IS NOT NULL :- Revere of Is Null
   p) AND :- Combines two logical conditions. Returns true if both condition are true
   q) OR :- Combines two logical conditions. Returns true if any of them is true.

r) NOT :- Reveres a result of logical expression

<u>Build-in-Functions</u>: -

a) COUNT:- The COUNT function returns the number of tuples or values specified in a query.
b) SUM:- The SUM function return summation of the specified attribute's value
c) MAX:- The MAX function return maximum value of the specified attribute
d) MIN:- The MIN function return minimum value of the specified attribute
e) AVG:- The AVG function return average value of the specified attribute

<u>Embedded SQL</u>: - SQL can also be used in conjunction with a general-purpose programming language such as PASCAL, COBOL etc. The programming language is called the host language. Any SQL statement can be embedded in a host language program. The embedded SQL statement is distinguished from programming language statements by prefixing it with a special character or command so that a preprocessor can separate the embedded SQL statement from the host language code.

In general, different systems follow different conventions for embedding SQL statements. Within an embedded SQL command, we may refer to programme variables, which are prefixed by a "%" sign. This allows program variables and database schema objects, such as attributes and relations, to have the same names without any ambiguity.

<u>Integrity Constraints</u>: - Intetrity constraints are specified on a database schema and are expected to hold on every database instance of that schema. There are three types of integrity constraints :

i) <u>Key Constraint</u>: - Key constraints specify the candidate keys of each relation schema. Candidate key values must be unique for every tuple in any relation instance of that relation schema.
ii) <u>Entity Integrity Constraint</u>: - The entity integrity constraint states that no primary key value can be null
iii) <u>Referential Integrity Constraint</u>: - It is a constraint that is specified between two relations and is used to maintain the consistency among tuples of the two relations.

Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

Functional Dependency(FD): - A functional dependency is a constraint between two sets of attributes from the database. A functional dependency, denoted by X Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation instance r of R. The constraint states that for any two tuples t1 and t2 in such that t1[X] = t2[X], we must also have t1[Y] = t2[Y]. This means that the values of the Y component of a tuple in r depend on, or are determined by the values of the X component or, alternatively, the values of the X component of a tuple uniquely determine the values of the Y component. We also say that there is a functional dependency from X to Y or that Y is functionally dependent on X.

Normal forms: - Normalization of data can be looked on as a process during which unsatisfactory relation schemas are decomposed b breaking up their attributes into smaller relation schemas that posses desirable properties.

i. First Normal Form (1NF): - The first normal form is defined to disallow multivalued attributes, composite attributes and their combinations. The only attribute values permitted by 1NF are single atomic values. For example

STUDENT (Rollno, Name, Subject, Address).

In the STUDENT relation Subject attribute is multivalued attribute and Address is composite attribute. So this relation is not under 1NF. We decompose it according to the 1NF.

R (Rollno, Name, Dist, State, Vill, Pin)   S (Rollno, Subject)

STUDENT

R(Rollno, Name, Dist, State, Vill, Pin)   S(Rollno, Subject)

ii. Second Normal Form(2NF): - The second normal form is based on the concept of a full functional dependency. A functional dependency X Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more, that is, for any attribute A ∈ X. A functional dependency X Y is a partial dependency if there is some attribute A ∈ X that can be removed from X and the dependency will still hold.

A relation schema R is in 2NF if every nonprime attributes A in R is fully functionally dependent on the primary key of R. For example:

PARTS (PartNo, SupplyNo, PartName, SupplyName, Qty).

In this PARTS table F1, F2, F3 functional dependency hold. Now if we remove PartNo attribute from PARTS then the functional dependencies F1 and F3 will not hold. Again if we remove SupplyNo attribute from PARTS then the functional dependencies F2 and F3 will not hold. That means if we remove any attribute either PartNo or SupplyNo the functional dependency F3 certainly will not hold. So F3 is the full functional dependency and F1 & F2 are partial functional dependency. If the table has partial functional dependency then that table is not under 2NF. So we decompose it into three tables as follows

R (<u>PartNo</u>, PartName)   S (<u>SupplyNo</u>, SupplyName)   T(<u>PartNo, SupplyNo</u>, Qty)

PARTS

R(<u>PartNo</u> PartName)   S(<u>SupplyNo</u>, SupplyName)   T(<u>PartNo, SupplyNo</u>, Qty)

iii) <u>Third Normal Form (3NF)</u>: - The third normal form is based on the concept of a transitive dependency. A functional dependency X Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is not a subset of any key of R, and both X Z and Z Y hold.

A relation is in 3NF if it is in 2NF and no nonprime attribute of R is transitively dependent on the primary key. For example

EMPLOYEE (<u>EmpCode</u>, EmpName, Salary DeptCode DeptName DeptLocation)

In this relation EmpCode can determine the value of EmpName, Salary and DeptCode. So these three attributes are depending on EmpCode. On the other hand DeptCode can determine the value of the attributes DeptName and DeptLocation. So both the attributes are depending on DeptCode but DeptCode is not a primary key. That means some nonprime attribute (DeptName, DeptLocation) are depending on another nonprime attribute (DeptCode) and that non prime attribute is depending on primary key (EmpCode). This relation is transitive dependency, so it is not under 3NF. We decompose it as follows:
R (EmpCode EmpName Salary DeptCode)   S (DeptCode DeptName DeptLocation)

EMPLOYEE

R (EmpCode Empname Salary DeptCode)   S (DeptCode DeptName DeptLocation)

iv. Boyce Code Normal Form (BCNF): - A relation is in BCNF if whenever a functional dependency X A holds in R the X is super key of R.

Multivalued Dependency: - If we have two or more multi valued independent attributes in the same relation schema, we get into a problem of having to repeat every value of one of the attributes with every value of the other attribute to keep the instances consistent. This constraint is specified by a multi valued dependency. For example: COURSE(C_id Subject Teacher)

Here C_id is a single valued attribute. Subject and Teacher attributes are multi valued attributes because under one course many subject and many teacher are there. Subject and Teacher attributes are independent of each other but depending on C_id (primary key). So multi valued dependency exist in this relation.

Back up and Recovery: - A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing. Alternatively, the recovery subsystem cou8ld ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

Hardware protection and redundancy: -

Transaction: - A transaction is an atomic unit of work that is either completed in its entirety or not done at all. A transaction has the following stages :

1. BEGIN_TRANSACTION: - This marks the beginning of transaction execution.
2. READ OR WRITE: - These specify read or write operations on the database items that are executed as part of a transaction.
3. END_TRANSACTION: - This specifies that READ or WRITE transaction operations have ended and marks the end of transaction execution.
4. COMMIT_TRANSACTION: This signals a successful end of the transaction so that any changeds executed by the transaction can be safely committed to the database and will not be undone.
5. ROLLBACK (OR ABORT): - This signals that the transaction has ended unsuccessfully, so that any changes or effects that the transaction may have applied to the database must be undone.

Transaction logs (**System logs**): - To be able to recover from failures that affect transactions, the system maintains a log to keep track of all transaction operations that affect the values of database items. This information may be needed to permit recovery from failure. The log is kept on disk, so it is not affected by any type of failure except for disk. In addition, the log is periodically backed up to archival storage ro guard against catastrophic failure. Each entries in the log is called log records. Different types of log records are:

1. [start_transaction T] : indicate that transaction T has started execution.
2. [write-item, T, X, old_value, new_value]: Indicates that transaction T has changed the value of database item X from old_value to new_value.
3. [read_item, T, X] : Indicates that transaction T has read the value of database item X.
4. [commit, T] : Indicates that transaction T has completed successfully, and affirms that its effect can be committed to the database.
5. [abort, T] : Indicates that transaction T has been aborted.

Database failure and recovery : - Whenever a transaction is submitted to a DBMS for execution, the system is responsible for making sure that either

1. all the operations in the transaction are completed successfully and effect is recorded permanently in the database, OR

2. the transaction has no effect whatsoever on the database. This may happen if a transaction fails after executing some of its operations.

Types of failures: - Failures are generally classified as transaction, system and media failures.

1. A computer failure: A hardware, software or network error occurs in the computer system during transaction execution.

2. A transaction or system error: Some operation in the transaction may cause it to fail, such as integer overflow or division by zero.

3. Local errors or exception conditions detected by the transaction: During transaction execution, certain conditions may occur that necessitate cancellation of the transaction.

4. Concurrency control enforcement: - The concurrency control method may decide to abort the transaction, to be restarted later.

5. Disk failure: - Some disk blocks may lose their data because of a read or write malfunction or because of disk read/write head crash.

6. Physical problems and catastrophes: This refers to an endless list of problem that

includes power or air conditioning failure, fire, theft etc.

Recovery technique: - Different recovery techniques are

1. Recovery techniques based on Deferred update: - The idea behind deferred update techniques is to defer or postpone any actual updates to the database until the transaction completes its execution successfully and reaches it commit point. During transaction execution, the updates are recorded inly in the log. After the transaction reaches its commit point and the log is force-written to disk, the update are recorded in the database. If the transaction fails before reaching its commit point, there is no need to undo any operation.

2. Recovery techniques based on Immediate update: - In these techniques, when a transaction issues an update command, the database can be updated immediately without any need to wait for the transaction to reach its commit point. The update operation is recorded in the log. Provision must be made for undoing the effect of update operations that have been applied to the database by a failed transaction.

3. Shadow paging: - Shadow paging considers the database to be made up of a number of fixed-size disk page for recovery purposes. A directory is maintained to keep the information of each page. When a transaction begins executing, the current directory – whose entries point to the most recent or current database pages is copied into a shadow directory. The shadow directory is then saved on disk while the current directory is used by the transaction.

During transaction execution, the shadow directory is never modified. When a write_item operation is performed, a new copy of the modified database is created, but the old copy of the page is not overwritten. Instead, the new page is written elsewhere on unused disk block. The current directory entry is modified to point to the new disk page where as shadow directory is not modified and continues to point to the old unmodified disk page.

If the transaction completes successfully then shadow page directory is deleted to make the update permanent in the database. If the transaction fails to complete then the current directory is deleted and shadow page directory is considered as current directory to bring the database to the previous stage of the execution of the transaction.

Security importance of data security : -Different issues of database security are :

1. Legal and ethical issues regarding the right to access certain information. Some information may be deemed to be private and cannot be accessed legally by unauthorized persons.

2. Policy issues at the governmental, institutional, or corporate level as to what kinds of

information should not be made publicly available.

3. System-related issues such as the system levels at which various security functions should be enforced.

4. The need in some organizations to identify multiple security levels and to categorize the data and users based on these classification.

Database security and the DBA: - Database administrator (DBA) is the central authority for managing a database system. The DBA's responsibilities include granting privileges to users who need to use the system and classifying users and data in accordance with the policy of the organization. The DBA has a DBA account in the DBMS, sometime called a system or super user account. DBA privileged commands include commands for granting and revoking privileges to individual accounts, users, or user groups and for performing the following types of actions:

1. Account creation: This action creates a new account and password for a user or a group of users to enables access to the DBMS

2. Privilege granting: This action permits the DBA to grant certain privileges to certain accounts.

3. Privilege revocation: This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.

4. Security level assignments: This action consists of assigning user accounts to the appropriate security classification level.

Protecting data in the database: -

Granting and revoking privileges

Questions:

Q: What are the functions of transaction log?

A: The main function of transaction log is to keeps track of all transaction operation that affect the values of database items. The information may be needed to recover from transaction failures.

Q: What is database recovery? What is meant by forward and back ward recovery.

A: Database recovery means getting the original database back after any type of hardware or software failure.

Q: Difference between authorization and authentication

Authentication verifies who you are. For e.g. login to a system or access mail.

Authorization verifies what you are authorized to do. For e.g. you are allowed to login but you are not authorized to browser / or any other file system.

Authorization occurs after successful authentication. Authorization can be controlled at file system level or using various application level configuration options.

Usually the connection attempt must be both authenticated and authorized by the system. You can easily find out why connection attempts are either accepted or denied with the help of these two factors.

Q: What types of problem may arise if relations are not normalized

A: The problems are:

1. Redundancy can not be reduced
2. inconsistency will exist
3. memory will be wasted
4. updating need many time

Q: What do you mean by transaction recovery?

A:

Q: What are the characteristics of the transaction?

A: The characteristics of a transaction are:

- **Atomicity.** A transaction must be an atomic unit of work (either all of its data modifications are performed, or none of them is performed).
- **Consistency.** When completed, a transaction must leave all data in a consistent state.
- **Isolation.** Modifications made by concurrent transactions must be isolated from the modifications made by any other concurrent transactions. A transaction either sees data in the state it was in before another concurrent transaction modified it or it sees the data after the second transaction has completed, but it does not see an intermediate state.
- **Durability.** After a transaction has completed, its effects are permanently in place in the system. The modifications persist even in the event of a system failure.

Q: Describe the three-tier ANSI-SPARC ARCHITECTURE OF DATABASE SYSTEM. Write its advantage.

A:

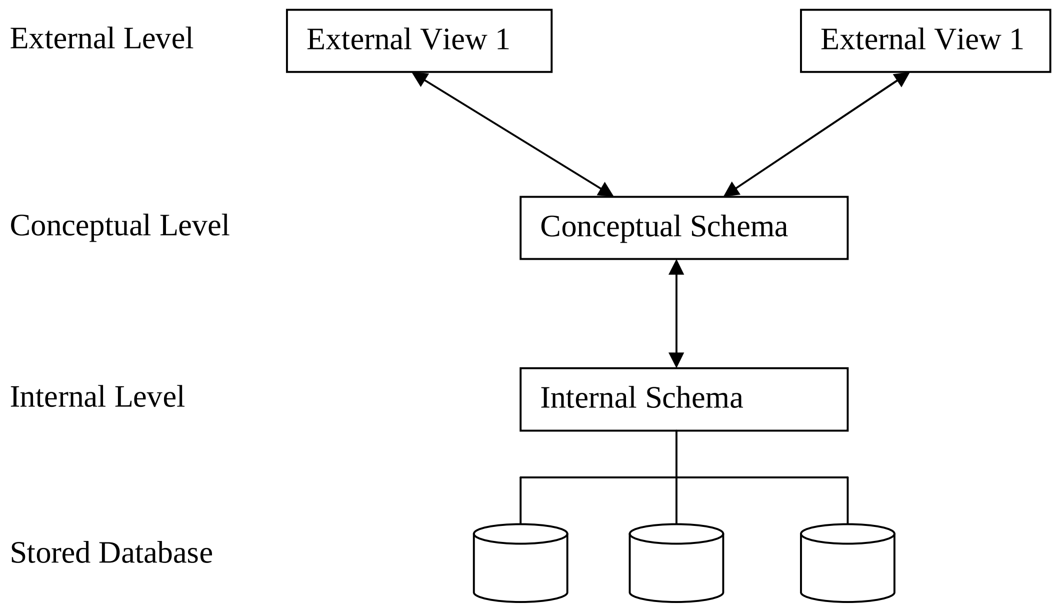

Fig: Three level architecture of DBMS (ANSI-SPARC)

4. Internal Level : - The internal level has a internal schema, which describes the physical storage structre of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
5. Conceptual Level: - The conceptual schema or level describes the structure of the whole database for a community of users. The conceptual schema hides the4 details of physical storage structures and concentrates on describing entities, data types, relationships, user operations and constraints.
6. External or view level: - The external level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from the user group.

Q: Show that if a relation schema is in BCNF, then it is also in 3NF

A: According to 3NF every non-prime attribute of a relation is non-transitively dependent directly dependent on every key of that table.

According to BCNF a table is in BCNF if and only if, for every one of its non-trivial functional dependencies $X \rightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.

From both the definition it is clear that when a relation is in BCNF, all non prime attributes are directly depending on key attribute or super key, which is indirectly following rule of 3NF. So when a relation is in BCNF we can say that it is also in 3NF.