Date :- 13/August/2019 [MCA]

## => DATABASE MANAGEMENT SYSTEM (DBMS) :-

Legacy System :- These are those system that are in use ~~of some~~ since a very long time and they make use of some old technology and may be sometimes older hardware system like mainframes.

Most of the legacy system were based on CUI interface having a black and white screen and a prompt. for several decades this was the only options available for the developers to design and develop application program or project.

In almost all of these programs or application software file handling was used to manage application data in another words there was no DBMS at that time to handle and manage application data.

All these programs were developed in high level languages like C, C++, Cobol, Pascal, fortran etc using file handling options of these languages and functions like fopen, fclose, write(), read(), gets(), puts(), fprintf(), etc.

Data :- Any collection of RAW FACTS And FIGURES can be termed as data. for example :- height of a student, Account balance, bill amount, etc.

Database :- Any organised collection of related data stored in a particular format for a specific purpose is called as Database. for example :- class attendance register, Account ledger, Account photo, mobile phone etc.

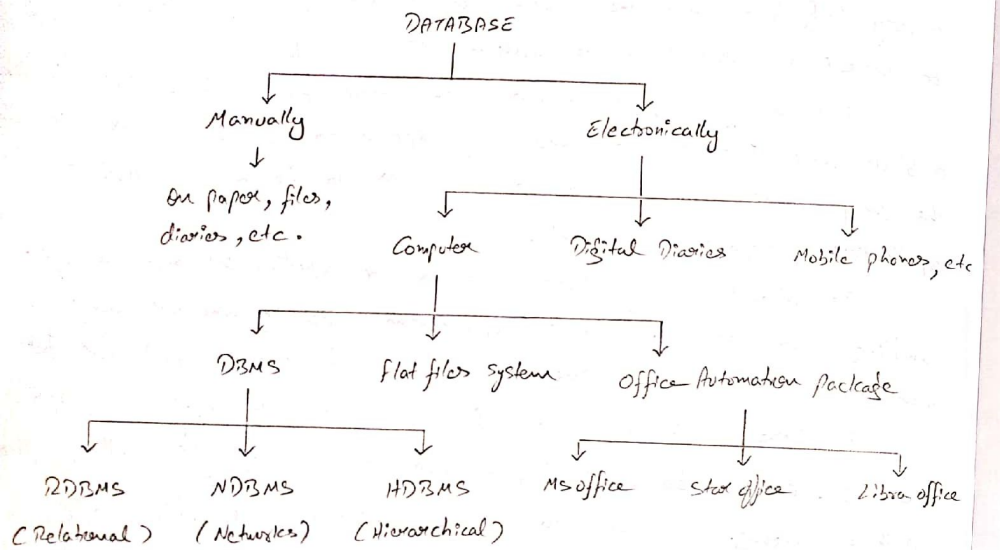
Database Management :- whenever any or all of the following operation are performed on Database. we call it as Database Management.

Operations such as :-

=> Selection / view
=> Deletion
=> Insertion
=> Updation

## => DATABASE MANAGEMENT TECHNIQUES :-

Broadly there are two ways to manage data i.e,
(i) Manually (ii) by using electronic means as shown in the following diagram below.




file database :- It is a database stored in a file called a file. Records follow a uniform format, and there are no structures for ing or recognizing relationships between records. The file is simple. file can be a plain text file, or a binary file. Relationships can erred from the data in the database, but the database format does not make those relationships explicit.
his term has generally implied a small database, but very atabases can also be flat.
we have seen that the data of the flat files is visible command prompt by using C:> type filename.

## => Characteristics and features of flat file :-

(i) The application programs using such flat files are commonly known as TFOS (traditional file oriented system), ffs (flat file system) or legacy system.

(ii) These flat files can have any name or any extension. however some common extension are .dat, .dbf, .rec etc however these extensions have no significance in actual.

(iii) These flat files can contain data either in ASCII format or ANSI or any binary format in the form of class object or structure members via stream of bytes although we can give any type of extension to these files but we should avoid giving executable extensions in case of DOS and windows environment like .exe, .com (binary), .bat (batch executable file).

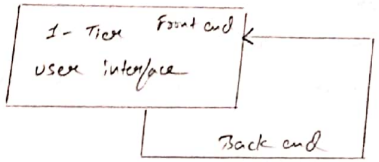
An exe file contains compiled machine code as per processor architecture i.e, 32 bit, 64 bit etc.

## => Drawbacks of flat file :-

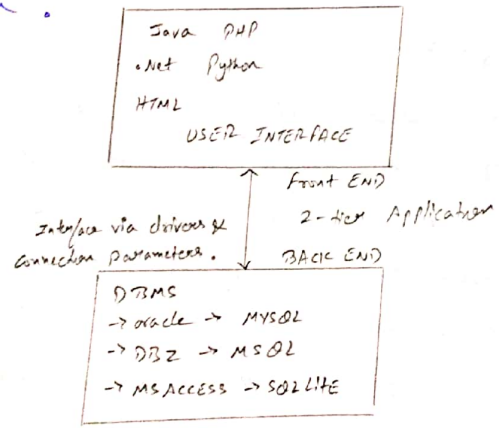
(i) No data security.
(ii) Program data structure and data file dependancy :- once the data was stored in flat files it is very difficult to change the data structure of the application programs.
(iii) Not suitable for client server application since flat files were developed long ago they are not suitable for server based application
(iv) Flat files do not support concurrent transactions as in case of application such as train ticket booking system, online shoping, super market application etc.
(v) Not suitable for data migration, warehousing and performance tuning.

Note :- Although flat files have no. of drawbacks but they also have a number of benefits in terms of speed, size and backup facilities. flat files are integrals part of any programming language. In another words, no language can be developed without the support of flat files. for eg :- Java, Python, Android etc all these programming language support traditional file oriented system (TFOS).

The flat files based applications are also known as one tier applications in which front end and back end are same.
for eg :- C++, C, cobol, pascal, fortran, etc.




DATABASE MANAGEMENT SYSTEM :- To overcome the draw backs of flat files Database Management System (DBMS) were developed. the developers and programmers can concentrate more on application features and functionality rather than wasting time on handling flat files. when DBMS is used the application is atleast a two tier application.




Database Management system is a general purpose data processing software that makes the task of data menagement very easy some other definitions of DBMS are :-

A DBMS is a collection of software modules that along with the support of hardware provides excellent data management facilities which are not available in other above discussed method of Database Management.

A Database Management system (DBMS) is a software system that enables users to define, create and maintain the database and provides controlled access to this database through the application program. A DBMS uses Data Definition Language (DDL), Data Manipulation Language (DML) and Data control Language (DCL) for data manipulation. A typical DBMS is based on SQL.

In another words, we can say that SQL is an essential & integral part of any RDBMS some popular example of RDBMS are oracle DB, IBM DB2, MS SQL (Sequel Service), MySQL, MS Access, Sybase, Maria DB etc so SQL is a generic term not specific to any product.

=> DBMS FEATURES AND FACILITIES :-

A DBMS is used to set and monitor a database and helps in inserting updating and retrieval of data from that database. In general most DBMS provides following facilities :-

- Creation of data files / tables (schema definition).
- Deletion of data files.
- Modification in the structure of data files (alteration in table structure)
- Insertion of data.
- updation of data.
- Deletion of data.

- Retrival of data collectively or selectively
- Enforcement of constraints on data values.
- Security of data Access
- Generation of reports
- Mathematical functions on data stored in the database.
- Maintenance of data integrity. (State of being complete/whole)
- Controlled access.

→ Enforcement of constraints on data values :- Oracle provides different types of constraints such as primary key, foreign key, unique, not NULL, check.

→ Security of Data Access :- Oracle provides different levels of security for eg: various types of grant permissions.

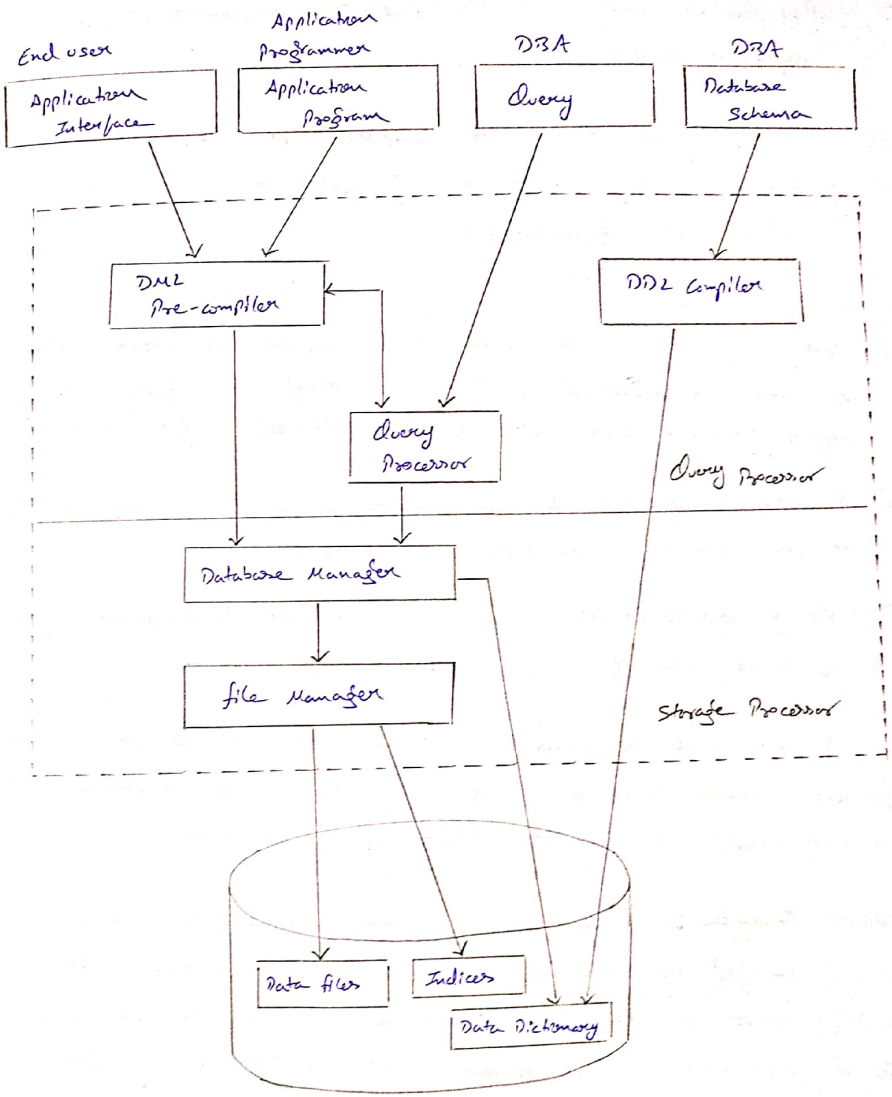
→ Generation of Reports :- DBMS provides simple report generation facility using environment variable parameters However sophisticated report generation can be done only with the help of front end tools.

→ Maintenance of Data Integrity :- Data integrity is maintained through the use of primary key, foreign key constraint and other transaction level features.

→ Controlled Access :- Oracle provides a very robust transaction processing system.

⇒ DBMS Components :- Every DBMS is made up of no. of components just like an OS is made up of components like CPU scheduler, file manager etc. These components can vary from company to company. However, all of them are based on RDBMS concepts suggested by Dr E.F codd and SQL.

The following diagram illustrates architecture of a typical DBMS :-



(01) User Interface :- It defines how user interacts with the DBMS there can be many ways like CUI, GUI, Browser Based Interface, Programming Interface, etc

(02) DDL & DML compiler :- these components are responsible for checking the query for any type of syntax error.

Example : > select from tab ;

OUTPUT : Error

(03) Query processor :- this is the most important component which is responsible for converting a high level language SQL query into internal executable code which is then optimized before actual execution.

(04) Database Manager :- This component is responsible for communication with other components for proper query execution.

(05) File Manager :- This component handles the input/output operations in consultation with OS and Database engine.

All the above components are loaded into RAM as soon as the database instance is up or running. The bottom three components are actually stored on secondary storage devices or stable storage.

ORACLE Instance :- when oracle database server is running it is always identified by instance Name or Host string or Connection string. A single server can have multiple instances and also in case of client server environment oracle database server is identified by instances name.

Error : TNS : protocol adapter error

This error means the oracle engine or oracle server or oracle instance is down.

In technical term, this error means the component shown in the above diagram are not running.

when you install oracle it always ask for a instance name a common password.

Query to shutdown oracle instance,

> shutdown

OUTPUT : Database closed
Database dismounted
Oracle instance shutdown

Query to restart oracle instance

> Startup

OUTPUT : Oracle instance started
Total system Global area 673208 bytes
fixed size 1292076 bytes
⋮
Database mounted
Database opened

The bottom three components are not the part of 04 Database engine rather they are stored on secondary storage devices. In other words, Data files, Indices & Data Dictionary is stored on hard disk.

(01) Data files :- Data files are OS level files that contains user tables in encapsulated and encrypted format. For eg :- In case of oracle database all the tables are hidden inside .DBF files and we can never see any table at OS level file system.

Path : C:/oracle/product/10.2.0/oradata/orcl

| | | |
|---|---|---|
| EXAMPLE01.DBF | UNDOTBS01.DBF | REDO01.LOG |
| SYSAUX01.DBF | USERS01.DBF | |
| TEMP01.DBF | CONTROL01.CTL | |

In case of whatsapp the messages are again stored inside tables which are further encapsulated in .DB files.

(02) Data Dictionary :- It is data about Data or Metadata which contains all critical information regarding database status since oracle is RDBMS so, data dictionary is also stored in the form of

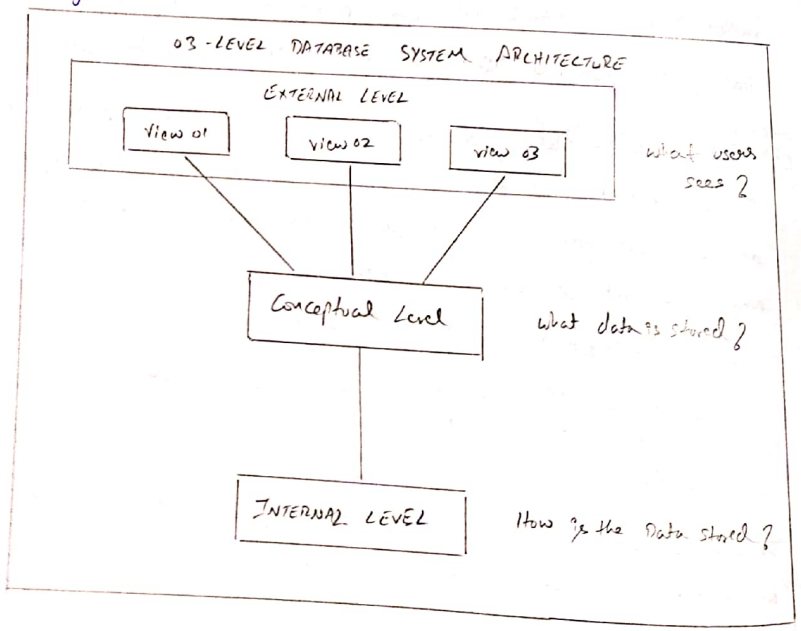
tables. for eg :- when you connect with sys or system user the tables that you see are all Data dictionary table.

Example :- • select * from tab; • select * from user_tables;
• select * from cat; • select * from all_users;
• select * from v$database;

all are DB tables which may not be visible in the user list but still you can select data from them.

=> Three Layer Architecture of DBMS :-

Most of the DBMS are design & Develop as a three layer architecture for scalability and better performance. The following diagram illustrates these three layers :-




(i) External view / High Level view :- At this level, user can have multiple views of same data.

For Example :- select * from emp ;
↳ this is view 1

select ecode, ename from emp ;
↳ this is view 2

(ii) Middle view / Logical view / conceptual view :- At this Level user can see the data type and size of the table fields and columns. In other words, there can be only one conceptual view per table whereas there can be 'n' external view per table.

for example :- Describe emp ;

(iii) Internal view / Physical view / Lower Level :- This layer is not visible to the user and tables are implemented using various storage structures and implementation algorithms.

for Example :- In case of oracle all tables are encapsulated inside .DBF files which is hidden from the user.

Similarly, as in case in MS Access at OS Level we can see .MDB file

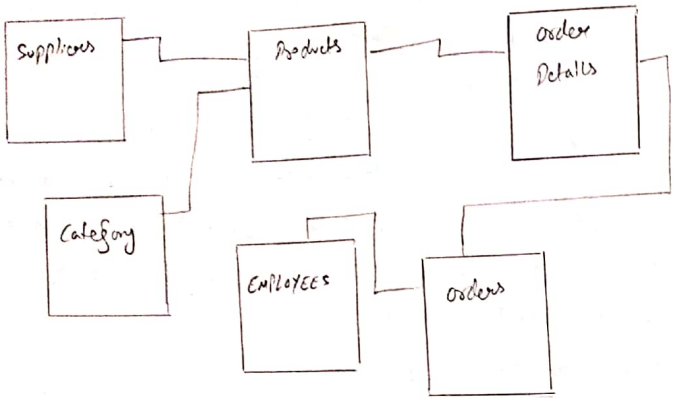
Schema : Schema is of two types :- (i) Table schema
(ii) Overall Database Schema

Table Schema is also known as table definition or table design or table structure. It illustrates the fields or columns of a table.

Example :- Describe table name ;

Overall Database Schema defines & illustrates the relationships among the relationships among the table of a project using referential integrity Concepts (primary key, foreign key concepts). Database Schema can be

viewed in graphical mode in Database like MS Access, where as in Database oracle it is not directly visible however database schema is always shown in detail in software engineering document SRS.




In general there are three types of DBMS users,

(i) Simple or Naive users.
(ii) Developers and Application programmers.
(iii) DBA.

Simple or Naive users :- These are the most common type of users who are sometimes not even aware of presence of DBMS. They simply make use of application program for there day to day operations / tasks.

Developers and Application programmers :- There users are aware of presence of DBMS and its schema. There developers write SQL queries embedded inside there code.

DBA :- A DBA is the custodian of project database and they perform any or all of the following operations :-

- Creation & Deletion of Database users.
- Granting of various types of permissions.
- Creating database schema or tables.
- taking the backup of the database.
- Performing recovery operations if required.
- fine tuning the database.
- Any other operation related to performance optimization.

SQL prompt

↙ ↘

SQL queries — Confirmation msg?

Internal commands — (no confirmation msg?)

'/' use to run last SQL query not for internal command.

Exporting and importing Oracle Data :-

In order to export oracle data EXP and IMP utility are used. these are .exe files not to be executed on SQL prompt.

Steps for Exporting oracle Data :-

Step (01) Open command prompt i.e cmd

Step (02) type C:\exp and press enter

Step (03) Specify the username of the user whose data you want to export

Step (04) specify different options including filename, location etc.

Steps to import oracle Data :-

Step (01) Create same user on the target machine using sys or system user if not created earlier.

Step (02) open command prompt i.e cmd

Step (03) type C:\imp and press enter.

## Data Modelling and ER Diagrams :-

Modelling is a technique in which real world objects are mapped or modelled into some other types of representative tokens or elements. for eg :- A 2D Map for a House or a 3D model of a complex.

Similarly, in case of software we need to model real world objects, process or concepts into some computer related objects.

for example :- When a book is issued from library two activities are performed, (i) a physical book is issued
(ii) a virtual book is issued by the software

ER Diagram (Entity Relationship Diagram) is a graphical tool to model conceptual entities of a software project domain. In other words, an ER diagram is used to show case logical or conceptual model of a software problem context.

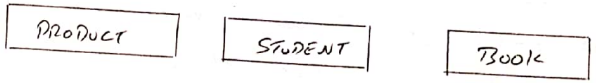
In order to draw an ER diagram detailed requirement analysis is to be done. ER Diagram is one of the many graphical model available in SRS like flow chart, DFD, use case diagrams, UML diagrams etc.

**ER Diagram :-** An ER Diagram models the entire Software project Data requirement by using only following 2 components i.e
(i) Entities (ii) Relationship

In other words, the above two components are used to describe all objects process and concepts of a software application for which the data is to be stored.

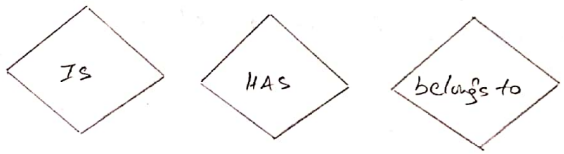
**Entities :-** An entity is an object or a concept for which the data is to be captured and stored. For eg :- student, book, bank account, employee, product, ticket etc.

An Entity is shown in an rectangular box in an ER Diagram.

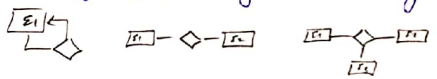



Entity is of two type (i) Strong entity
(ii) Weak entity

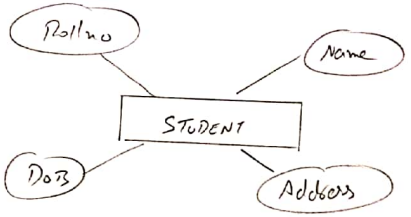
**Relationship :-** A relationship is an association or a link between two or more entities. It is shown as a Diamond box in an ER Diagram




Relationship is of three types : (i) unary (ii) Binary (iii) N-nary



**Attributes :-** An Attributes is a characteristic or property of an entity which can become a table column. It is shown as a oval attached to an Entity.




Attributes are of many types, (i) Simple or Atomic
(ii) Complex or Composite
(iii) Stored or derived

(i) Simple or Atomic Attribute :- An attribute which can directly become a table column is known as simple or atomic column.

(ii) Complex or Composite Attribute :- An attribute which can not directly become a table column rather it can be broken into sub columns is known as Composite Attribute. Example : Name can be broken down into first name, middle name, last name, etc.

(iii) Stored or Derived Attribute :- An attribute which directly becomes a table column with a direct value is known as stored attribute where as a column for which a direct value is not stored in a table rather it is extracted from some other column is known as derived attribute. for example : Age is derived from DoB, etc.

→ CoDDs Rules By Dr E.F Codd :-

A RDBMS is an DBMS which is based on relational theory and rules proposed by Dr E.F codd. In other words, a DBMS is said to be a RDBMS if it follow codds rules there are total 12 rules beside rule no. zero.

A RDBMS is said to be a RDBMS only if completly relational, only relational and nothing but relational.

Rule 00 : This rule states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities.

Rule 01 : The Information Rule

→ All data should be presented to the user in a table form.
→ All information in the database to be represented in one and only one way, Namely by values in column positions within rows of tables.

RULE 02 : Guaranteed Access Rule

→ Data should be accessible without ambiguity
→ Uses tablename, primary key, column name to uniquely access single piece of data [atomic data]

For eg :- Select location from department where dname = 'HR';

OUTPUT :

| Location (Atomic Data) |
|---|
| Mumbai |

RULE 03 : Systematic Treatment of NULL Values

→ A field should be allowed to remain empty
→ Involves the support of a null value, which is distinct from an empty string or a number with a value of zero.
→ As per this rule, database should have provision of NULL values.

For eg :- Display names of those departments that are not assigned any locations.

>> Select dname from department where location is NULL;

Similarly, NULL is not included in most of the SQL functions

>> Select Count(*) from test; // to count all rows of a table or a column of a table.

>> Select NVL(dcode, 0) from test;

OUTPUT :

| NVL(dcode,0) |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 0 |

>> Select avg(nvl(dcode,0)) from test; // finding average when there is a null value present.

Note : NVL() function is used to replace NULL value with another value.

RULE 04 : Dynamic on-line Catalog based on the relational Model

-> Every DBMS must support an online data dictionary which should be maintained in the form of tables only as per rule no.

for_eg : >> select * from cat ;
>> select * from user_tables ;
>> select * from tab ;
>> select * from all_users ;
>> select * from user_constraints ;

RULE 05 : Comprehensive Data sub language Rule

-> It state that the database must support at least one clearly defined language.

RULE 06 : View updating Rule

-> Every DBMS must support SQL views.

-> A view is an alias for a query however it doesnot stores any type of data. they are of 2 types :-
(i) Simple views (ii) Complex views

Simple view :- a view that doesnt contains any type of formulah or expression is known as Simple view.

Complex view :- a view that contains any type of formulah or expression is known as Complex view.

>> Create view myview as select balance, balance+500, balance-500 from Account ; // CREATE VIEW

OUTPUT : Error : must have this expression with a column alias.

>> Select balance, balance+500 Revised, balance-500 Revised2 from Account ; OR

>> Select balance, balance+500 as MyBal from Account ; OR

>> Select balance, balance+500 as "My new Balance" from Account ;

Note : we use double quotes here in DBMS only not any place else.

OUTPUT : My New Balance
7777
5674
5544

>> Create view Myview as select balance, balance+500 MyBal1, balance-500 MyBal2 from Account ;

OUTPUT : view created

>> select * from myview ;

>> select * from all_views ; // Data-Dictionary of view

Note : we can perform all DDL, DML, DCL operations on views however the effect is reflected in base table.

>> create view myview3 as select ecode, ename, salary+100 as newsal from emp ;

>> select * from emp ;

| Ecode | Ename | Salary |
|---|---|---|
| 101 | Amit | 7878 |
| 201 | Kapil | 17878 |
| 301 | Sample | |

>> select * from myview3 ;

| Ecode | Ename | NewSal |
|---|---|---|
| 101 | Amit | 7978 |
| 201 | Kapil | 17978 |
| 301 | Sample | |

>> insert into myview3 values (501, 'abcd', 999) ;

Error : virtual column not allowed here

Note : we can't insert value inside a Complex column other than NULL value if there is not a constraint.

RULE 07 : High level Insert, update & Delete

→ The system must support set-at-a-time Insert, update & Delete operators.

Example : update emp set Salary = 7000;

RULE 08 : Physical Data Independence

→ Isolated from the physical Method of storing and retrieving information from the database.

RULE 09 : Logical Data Independence

→ How a user views data should not change when the logical structure of the database changes.

→ Difficult to satisfy.

Example : > select * from Emp;

OUTPUT :

| Ecode | Ename | Salary | Pno |
|---|---|---|---|
| 101 | AMIT | 7000 | 8888 |
| 201 | KAPIL | 7000 | 8888 |
| 301 | SAMPLE | 7000 | 8888 |

> Select Salary + Pno from Emp;

OUTPUT :

| Salary + PNO |
|---|
| 15888 |
| 15888 |
| 15888 |

> Alter table emp drop column Pno;

OUTPUT : Table altered.

> Select Salary + Pno from emp;

OUTPUT : ERROR : 'PNO' : invalid identifier ORA-00904

> Select 'The salary of' || ename || 'is' || salary from emp;

OUTPUT : The salary of || ename || is || salary

The salary of AMIT is 7000
The salary of kapil is 7000
The salary of sample is 7000

Note : || can not be used after select & before from.

RULE 10 : Integrity Independence

→ The database language should support constraints on user input that maintain database integrity.

→ Integrity is of two types :- (i) Entity Integrity (Primary key)
(ii) Referential Integrity (Foreign key)

RULE 11 : Distribution Independence

→ A user should be totally unaware of whether or not the database is distributed.

RULE 12 : No sub version Rule

→ No way to modify the database structure other than through the multiple row database language.

» TRANSACTIONS :- A transaction is a small unit of program that can make changes to the database, a transaction may only read a data item or it may read and modify a transaction may be made up of multiple statements as follows, Example :- Transaction to transfer $50 from account A to acount B :

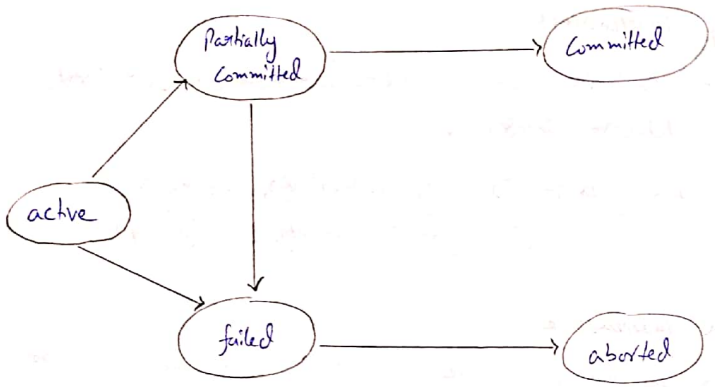
(i) read (A) (iii) write (A) (v) B := B+50
(ii) A := A-50 (iv) read (B) (vi) write (B)

Two main issues to deal with :-

→ failures of various kinds, such as hardware failures and system crashes.

→ Concurrent execution of multiple transaction.

Transactions States :- During its execution a transaction may go through no. of states as shown as in the diagram.




ACID Property :- During its execution no. of problems can occur in the system therefore DBMS must ensure the integrity of database in order to do so every DBMS implements ACID property.

- Atomicity :- either all operations of the transaction are properly reflected in the database or none are.
- Consistency :- Execution of a transaction in isolation preserves the consistency of the database.
- Isolation :- Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transaction.
- Durability :- After a transaction completes successfully the changes it has made to the database persist even if there are system failures.

## » CONCURRENCY AND ITS RELATED PROBLEMS :-

Concurrency means that more than one transactions are running in the system and they may access common data items. In case of concurrency we need to decide that in what manner transactions are executed. It is also known as transaction schedule. A schedule can be of following types :- (i) Serial schedule

(ii) A non-serial but equivalent schedule

(iii) A non-serial inconsistent schedule

Example : Let $T_1$ transfer $50 from A to B and $T_2$ transfer 10% of the balance from A to B. A serial schedule in which $T_1$ is followed by $T_2$

| $T_1$ | $T_2$ |
|---|---|
| read (A) | |
| A := A-50 | |
| write (A) | |
| read (B) | |
| B := B+50 | |
| write (B) | |
| | read (A) |
| | temp := A * 0.1 |
| | A := A - temp |
| | write (A) |
| | read (B) |
| | B := B + temp |
| | write (B) |

In order to increase system throughput transactions may be executed in interleaved manner i.e non serial manner but if the non serial execution is not controlled then many problems can occur.

for example :- consider the following two schedule / transaction :-

» Next Page

| Transaction $T_5$ | Transaction $T_6$ |
|---|---|
| Read (A) | Sum := 0 |
| A := A-100 | Read (A) |
| write (A) | Sum := Sum + A |
| Read (B) | Read (B) |
| B := B+100 | Sum := Sum + B |
| write (B) | write (Sum) |

If the above two transactions are executed in serial manner then the sum value will always be same. However, if they are executed in the following manner :- [Non serial inconsistent schedule]

| Schedule | Transaction $T_5$ | Transaction $T_6$ | value of Database item: A | B | Sum |
|---|---|---|---|---|---|
| Read (A) | Read (A) 500 | | 500 | 1000 | — |
| Sum := 0 | | Sum := 0 | | | 0 |
| Read (A) | | Read (A) 500 | | | |
| A := A-100 | A := A-100 400 | | | | |
| write (A) | write (A) 400 | | 400 | | |
| Sum := Sum+A | | Sum := Sum+A 500 | | | 500 |
| Read (B) | Read (B) 1000 | | | | |
| B := B+100 | B := B+100 1100 | | | 1100 | |
| write (B) | write (B) 1100 | | | | |
| Read (B) | | Read (B) 1100 | | | |
| sum := Sum+B | | Sum := Sum+B 1600 | | | |
| write (Sum) | | write (Sum) 1600 | | | 1600 |

TIME ↓

LOST UPDATE PROBLEM :- Another problem that can occur in case of uncontrolled non serial execution is lost update problem in which the changes made up by any one transaction is lost.

| Transaction $T_3$ | Transaction $T_4$ |
|---|---|
| Read (A) | Read (A) |
| A := A+10 | A := A * 1.1 |
| write (A) | write (A) |

| Schedule | Transaction $T_3$ | Transaction $T_4$ | value of A |
|---|---|---|---|
| Read (A) | | Read (A) | 200 |
| A := A * 1.1 | | A := A * 1.1 | |
| Read (A) | Read (A) | | |
| A := A+10 | A := A+10 | | |
| write (A) | write (A) | | 210 |
| write (A) | | write (A) | 220 |

TIME ↓

→ HANDLING CONCURRENCY RELATED PROBLEMS :-

The above problems are occuring because more than one transactions are executing without any control. Therefore, we need to ensure that these simultaneous execution gives the same result as a serial execution. Therefore a concept known as conflict serializability.

Two statements are said to conflict if all following three conditions are true :-

(i) More than one transactions are executing with statements in different transaction.

(ii) Both statements Access / refers to same data item.

(iii) Allleast one of them performs write operations on the data item.

Note :- The sequence order of statements can never be changed.

» CONFLICT SERIALIZABILITY :-

→ If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent.

→ Two schedule are said to be conflict equivalent if the order of any two conflicting operations is same in both the schedules.

→ We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule S'. In such a case we can reorder the NON CONFLICTING Operations in S until we form the equivalent serial schedule S'.

For example :- Schedule 3 is conflict equivalent to schedule 6

| $T_1$ | $T_2$ |
|---|---|
| read (A) | |
| write (A) | |
| | read (A) |
| | write (A) |
| read (B) | |
| write (B) | |
| | read (B) |
| | write (B) |

Schedule 3

| $T_1$ | $T_2$ |
|---|---|
| read (A) | |
| write (A) | |
| read (B) | |
| write (B) | |
| | read (A) |
| | write (A) |
| | read (B) |
| | write (B) |

Schedule 6

TWO PHASE LOCKING :-

To implement serializability locks are used. There are two types of locks :- (i) Shared lock or Read lock => Locks (A)

(ii) Exclusive Lock or write Lock => Lockx (A)

→ Shared lock is a read only lock which can be granted multiple times on a single data items with the help of a counter. However during shared lock no write lock can be granted.

→ Exclusive lock is granted for updating the values and only one exclusive lock can be granted at a time on a data item also during exclusive lock, shared lock cannot be granted.

For example :- Consider the following schedule that uses locking :-

Transaction $T_5$

```
Lockx (A)
Read (A)
A := A-100
write (A)
unlock (A)
Lockx (B)
Read (B)
B := B+100
write (B)
unlock (B)
```

Transaction $T_6$

```
Lockx (sum)
sum := 0
Locks (A)
Read (A)
sum := sum + A
unlock (A)
Locks (B)
Read (B)
sum := sum + B
write (sum)
unlock (B)
unlock (sum)
```

If we try to execute the above schedule then again we will see that inconsistent read problem is still occuring :-

| Schedule | Transaction $T_5$ | Transaction $T_6$ |
|---|---|---|
| Lockx(sum) | | Lockx(sum) |
| sum := 0 | | sum := 0 |
| Locks(A) | | Locks(A) |
| Read(A) | | Read(A) |
| sum := sum + A | | sum := sum + A |
| unlock(A) | | unlock(A) |
| Lockx(A) | Lockx(A) | |
| Read(A) | Read(A) | |
| A := A-100 | A := A-100 | |
| write(A) | write(A) | |
| unlock(A) | unlock(A) | |
| Lockx(B) | Lockx(B) | |
| Read(B) | Read(B) | |
| B := B+100 | B := B+100 | |
| write(B) | write(B) | |
| unlock(B) | unlock(B) | |
| Lockx(B) | | Lockx(B) |
| Read(B) | | Read(B) |
| sum := sum+B | | sum := sum+B |
| write(sum) | | write(sum) |
| unlock(B) | | unlock(B) |
| unlock(sum) | | unlock(sum) |

TIME ↓

As we can see that simple locking is not preventing the problem. The problem is that $T_6$ locked & released item A & B as and when required. therefore, a technique known as two phase locking is used.

In this phase there are 2 stages :-

1. Growing phase or locking phase
2. Shrinking phase or unlocking phase

During growing phase only locks are acquire & no lock can be released where as in contraction phase locks can only be released & no new lock can be granted.

This will ensure correct execution of transaction however the throughput may be reduced because of serial execution.

Example :- The following schedule follows 2 phase locking protocol

| Transaction $T_{20}$ | Transaction $T_{21}$ |
|---|---|
| Lockx(A) | Lockx(sum) |
| Read(A) | sum := 0 |
| A := A-100 | Locks(A) |
| write(A) | Read(A) |
| Lockx(B) | sum := sum+A |
| unlock(A) | Locks(B) |
| Read(B) | Read(B) |
| B := B+100 | sum := sum+B |
| write(B) | write(sum) |
| unlock(B) | unlock(B) |
| | unlock(A) |
| | unlock(sum) |

The above schedule is executed as follows :-

| Schedule | Transaction $T_{20}$ | Transaction $T_{21}$ |
|---|---|---|
| Lockx(sum) | | Lockx(sum) |
| sum := 0 | | sum := 0 |
| Locks(A) | | Locks(A) |
| Read(A) | | Read(A) |
| sum := sum+A | | sum := sum+A |
| Lockx(B) | | Locks(B) |
| Read(B) | | Read(B) |
| sum := sum+B | | sum := sum+B |
| write(sum) | | write(sum) |
| unlock(B) | | unlock(B) |
| unlock(A) | | unlock(A) |
| unlock(sum) | | unlock(sum) |

TIME ↓

| | |
|---|---|
| Lock-X (A) | Lock-X (A) |
| Read (A) | Read(A) |
| A := A-100 | A := A-100 |
| write (A) | write (A) |
| Lock-X (B) | Lock-X (B) |
| unlock (A) | unlock (A) |
| Read (B) | Read (B) |
| B := B+100 | B := B+100 |
| write (B) | write (B) |
| unlock (B) | unlock (B) |

→ Computing the closure [ $f^+$ of $x$ ] :-

Sometimes you may need to identify the primary key of the Relation R on the basis of no. of FD's moreover you may also need to find some hidden FDs which can be very difficult if you use inference axiom.

So in such cases it is better to find the closure $f^+$ of attribute $x$ under $f$.

For Example : Consider the relation R (ABCDEH) with FDs f as

{ A → BC, CD → E, E → C, D → AEH, ABH → BD, DH → BC }

Now identify that BCD is the primary key or not.

As we know that a primary key attribute is an attribute on which all non prime attributes are dependend.

For Example : R (Rollno, Name, DoB)

Rollno → Name, Rollno → DoB, Rollno → Name, DoB

Out of 3 Attributes 2 are depended on Rollno. Therefore, rollno is the candidate key (Primary key) of the above relation.

So, In general an attribute or group of attributes on which all other remaining attributes are dependent is termed as candidate key.

In order to simplify this process the closure $f^+$ of $x$ is computed.

To compute the closure following algo is used.

INPUT : A Set of functional dependence f and a set of attributes X

OUTPUT : The closure $X^+$ of X under the FD in f

```
X+ := X ;  ( initialise X+ to X )
change := true ;
While change do
 begin
 change := false
  For each FD W → Z in f do
      begin
      if W ⊆ X+ then do
          begin
          X+ := X+ ∪ Z ;
          change := true ;
          end
  end
end
( X+ now contains the closure of X under f )
```

So, using above algorithm :-

① check BCD is a primary key or not

$BCD^+ = BCD$

$BCD^+ = BCDE$ (using $E \rightarrow C$)

$BCD^+ = ABCDEH$ (using $D \rightarrow AEH$)

Since, BCD+ contains all attributes so it is a primary key.

(2) check if D can be a primary key or not.

$D^+ = D$

$D^+ = ADEH$ (using $D \rightarrow AEH$)

$D^+ = ABCDEH$ (using $DH \rightarrow BC$)

Since, $D^+$ contains all attributes so it can be a primary key.

(3) check if C can be a primary key or not

$C^+ = C$

Since, $C^+$ contains only one attribute so it can not be a primary key.