

# Advanced DBMS

## (Assignment –I)

*Submitted in partial fulfilment of the requirements for the degree of*

**Master of Technology in Information Technology**

by

Vijayananda D Mohire

(Enrolment No.921DMTE0113)



Information Technology Department

Karnataka State Open University

Manasagangotri, Mysore – 570006

Karnataka, India

(2009)

# Advanced DBMS



## **CERTIFICATE**

This is to certify that the Assignment-I entitled (Advanced DBMS, subject code: MT14) submitted by Vijayananda D Mohire having Roll Number 921DMTE0113 for the partial fulfilment of the requirements of Master of Technology in Information Technology degree of Karnataka State Open University, Mysore, embodies the bonafide work done by him under my supervision.

**Place:** \_\_\_\_\_

**Signature of the Internal Supervisor**

**Name**

**Date:** \_\_\_\_\_

**Designation**

**For Evaluation**

<b>Question Number</b>	<b>Maximum Marks</b>	<b>Marks awarded</b>	<b>Comments, if any</b>
1	1		
2	1		
3	1		
4	1		
5	1		
6	1		
7	1		
8	1		
9	1		
10	1		
<b>TOTAL</b>	10		

Evaluator's Name and Signature

Date

## Preface

This document has been prepared specially for the assignments of M.Tech – IT I Semester. This is mainly intended for evaluation of assignment of the academic M.Tech - IT, I semester. I have made a sincere attempt to gather and study the best answers to the assignment questions and have attempted the responses to the questions. I am confident that the evaluator's will find this submission informative and evaluate based on the provide content.

For clarity and ease of use there is a Table of contents and Evaluators section to make easier navigation and recording of the marks. A list of references has been provided in the last page – Bibliography that provides the source of information both internal and external. Evaluator's are welcome to provide the necessary comments against each response, suitable space has been provided at the end of each response.

I am grateful to the Infysys academy, Koramangala, Bangalore in making this a big success. Many thanks for the timely help and attention in making this possible within specified timeframe. Special thanks to Mr. Vivek and Mr. Prakash for their timely help and guidance.

Candidate's Name and Signature

Date

## Table of Contents

FOR EVALUATION.....	4
PREFACE.....	5
QUESTION 1.....	9
ANSWER 1.....	9
QUESTION 2.....	14
ANSWER 2.....	14
QUESTION 3.....	16
ANSWER 3.....	16
QUESTION 4.....	17
ANSWER 4.....	18
QUESTION 5.....	20
ANSWER 5.....	20
QUESTION 6.....	23
ANSWER 6.....	24
ANSWER 6.....	26
QUESTION 7.....	28
ANSWER 7.....	28
QUESTION 8.....	30
ANSWER 8.....	30
QUESTION 9(i) .....	34
ANSWER 9(i).....	34
QUESTION 9(ii) .....	37
ANSWER 9(ii).....	37
QUESTION 10.....	39
ANSWER 10 .....	39
BIBLIOGRAPHY .....	43

## Table of Figures

<b>Figure 1</b> Structure of DBMS (Goyal, 2008) .....	10
<b>Figure 2</b> Extended ER Model Constructs (Mylopoulos, 2004) .....	17
<b>Figure 3</b> Client - Server Architecture .....	18
<b>Figure 4</b> Database Views (Ahmad, 1997).....	24
<b>Figure 5</b> Query Parsing .....	31
<b>Figure 6</b> Query parsing steps .....	31
<b>Figure 7</b> Parsing Tree .....	32
<b>Figure 8</b> Centralized Architecture .....	40
<b>Figure 9</b> Client –Server Architecture.....	40

***ADVANCED DBMS  
RESPONSE TO ASSIGNMENT - I***



**Question 1** What are the basics of DBMS?**Answer 1**

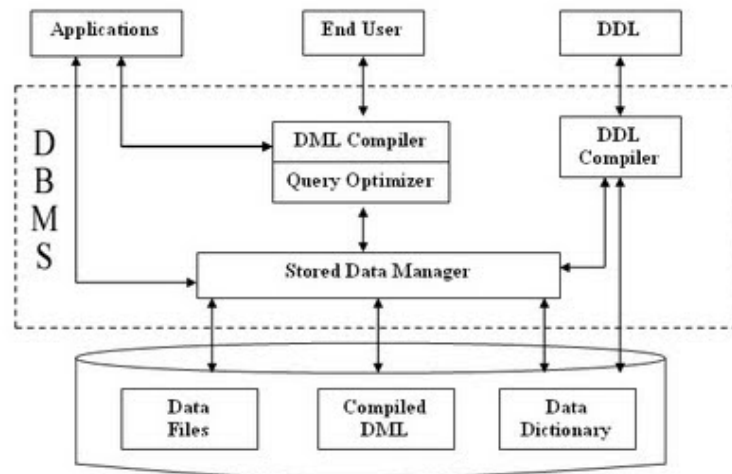
A Database Management System (DBMS) is a software package designed to store and manage databases.

DBMS Software that controls the organization, storage, retrieval, security and integrity of data in a database. It accepts requests from the application and instructs the operating system to transfer the appropriate data. The major DBMS vendors are Oracle, IBM, Microsoft and Sybase (see Oracle database, DB2, SQL Server and ASE). MySQL is a very popular open source product.

**STRUCTURE OF DBMS** (Goyal, 2008)

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users. The various components of DBMS are shown in Figure 1

**1. DDL Compiler** - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and



**Figure 1** Structure of DBMS (Goyal, 2008)

constraints etc.

**2. DML Compiler and Query optimizer** - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

**3. Data Manager** - The Data Manager is the central software component of the DBMS also known as Database Control System.

**4. Data Dictionary** - Data Dictionary is a repository of description of data in the database. It contains information about

- Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
- Relationships between database transactions and data items referenced by them which is useful in determining which transactions are affected when certain data definitions are changed.
- Constraints on data i.e. range of values permitted.
- Detailed information on physical database design such as storage structure, access paths, files and record sizes.
- Access Authorization - is the Description of database users their responsibilities and their access rights.
- Usage statistics such as frequency of query and transactions.

Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.

**5. Data Files** - It contains the data portion of the database.

**6. Compiled DML** - The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

**7. End Users** - They are already discussed in previous section.

Major Features of a DBMS

### **Data Security**

The DBMS can prevent unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or a

subset of it known as a "subschema." For example, in an employee database, some users may be able to view salaries while others may view only work history and medical data.

**Data Integrity**

The DBMS can ensure that no more than one user can update the same record at the same time. It can keep duplicate records out of the database; for example, no two customers with the same customer number can be entered.

**Interactive Query**

Most DBMSs provide query languages and report writers that let users interactively interrogate the database and analyze its data. This important feature gives users access to all management information as needed.

**Interactive Data Entry and Updating**

Many DBMSs provide a way to interactively enter and edit data, allowing you to manage your own files and databases. However, interactive operation does not leave an audit trail and does not provide the controls necessary in a large organization. These controls must be programmed into the data entry and update programs of the application.

This is a common misconception about desktop computer DBMSs. Complex business systems can be developed, but not without programming. This is not the same as creating lists of data for your own record keeping.

**Data Independence**

With DBMSs, the details of the data structure are not stated in each application

program. The program asks the DBMS for data by field name; for example, a coded equivalent of "give me customer name and balance due" would be sent to the DBMS. Without a DBMS, the programmer must reserve space for the full structure of the record in the program. Any change in data structure requires changing all application programs.

Evaluator's Comments if any:

**Question 2** What is information integration?**Answer 2**

Information integration is an important feature of Oracle9i database. Oracle provides many features that allows customers to synchronously and asynchronously integrate their data including Oracle streams, Oracle Advanced Queuing, replication and distributed SQL. Oracle also provides gateways to non-Oracle database servers, providing seamless interoperation in heterogeneous environment.

Asynchronous integration with Oracle streams:

Oracle streams enables the propagation and management of data, transactions and events in a data stream within a database, or from one database to another. The stream routes published information to subscribed destinations. The result is a new feature that provides greater functionality and flexibility than traditional solutions for capturing and managing events, and sharing the events with other databases and applications. Oracle streams provide both message queuing and replication functionality within the Oracle database. To simplify deployment of these specialized information integration solutions, Oracle provides customized APIs and tools for message queuing, replication and data protection solutions.

These include:

- Advanced queuing for message queuing
- Replication with Oracle streams

#### Asynchronous integration with Oracle Advanced replication

Oracle also includes Advanced Replication. This is a powerful replication feature first introduced in Oracle 7. Advanced replication is useful for replicating object between Oracle9i database and older versions of the Oracle server. Advanced replication also provides Oracle materialized views; a replication solution targeted for mass deployments and disconnected replicas.

#### Synchronous Integration with distributed SQL

Oracle distributed SQL enables synchronous access from one Oracle database, to other Oracle or non-Oracle databases. Distributed SQL supports location independence, making remote objects appear local, facilitating movement of objects between databases without application code changes. Distributed SQL supports both queries and DML operations, and can intelligently optimize execution plans to access data in the most efficient manner.




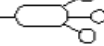





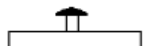

Evaluator's Comments if any:

**Question 3** What is the meaning of extended ER models?

**Answer 3**

- The Extended Entity-Relationship (EER) model is a conceptual (or semantic) data model, capable of describing the data requirements for a new information system in a direct and easy to understand graphical notation.
- Data requirements for a database are described in terms of a conceptual schema, using the EER model.
- EER schemata are comparable to UML class diagrams.



Construct	Graphical representation
Entity	
Relationship	
Simple attribute	
Composite attribute	
Cardinality of a	
Cardinality of an attribute	
Internal identifier	 
External identifier	
Generalization	
Subset	

**Figure 2** Extended ER Model Constructs (Mylopoulos, 2004)

The EER model mainly consists of entity super types, entity subtypes and entity clustering which are modeled in ER diagram.

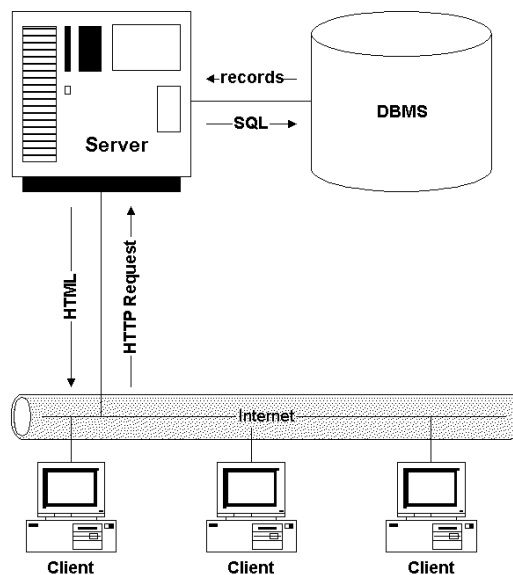
Evaluator's Comments if any:

**Question 4** What are client server systems?

**Answer 4** ( One response is Similar to answer 10, alternate described below).

Alternate response:

Client/Server technology is the computer architecture used in almost all automated library systems now being offered to libraries. (KSOU, 2009)



**Figure 3** Client - Server Architecture

This is a computer architecture that divides functions into client and server subsystems, with standard communication method to facilitate the sharing of information between them. Among the characteristics of a client/server architecture following are the key ones:

- The client and server can be distinguished from one another by the differences in tasks they perform.

- The client and server usually operate on different computer platforms
- Either the client or the server may be upgraded without affecting the other
- Client may connect to one or more servers; servers may connect to multiple clients concurrently. Clients always initiate the dialogue by requesting a service
- Client/server is most easily differentiated from hierarchical processing, which used a host and slave , by the way a PC functions within a system. In client/server the PC based client communicates with the server as a computer; in hierarchical processing the PC emulates a dumb terminal to communicate with the host. In client/server the client controls part of the activity, but in hierarchical processing the host controls all activities. A client PC almost always does the following in a client/server environment: screen handling, menu or command interpretation, data entry, help processing, and error recovery.

The dividing line between the client and server can be anywhere along a broad continuum: at one end only the user interface has been moved onto the

client; at the other, almost all applications have been moved onto the client and the database may be distributed.

Evaluator's Comments if any:

**Question 5** What is normalization?

**Answer 5**

In the field of relational database design, normalization (Anonymous, 2009) is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics—insertion, update, and deletion anomalies—that could lead to a loss of data integrity. E.F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1970. Codd went on to define the Second Normal Form (2NF) and Third Normal Form (3NF) in 1971, and Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form in 1974. Higher normal forms were defined by other theorists in

subsequent years, the most recent being the Sixth Normal Form (6NF) introduced by Chris Date, Hugh Darwen, and Nikos Lorentzos in 2002.

Informally, a relational database table (the computerized representation of a relation) is often described as "normalized" if it is in the Third Normal Form. Most 3NF tables are free of insertion, update, and deletion anomalies, i.e. in most cases 3NF tables adhere to BCNF, 4NF, and 5NF (but typically not 6NF).

The normal forms (abbrev. NF) of relational database theory provide criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies. The higher the normal form applicable to a table, the less vulnerable it is to inconsistencies and anomalies. Each table has a "highest normal form" (HNF): by definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF; also by definition, a table fails to meet the requirements of any normal form higher than its HNF.

The normal forms are applicable to individual tables; to say that an entire database is in normal form  $n$  is to say that all of its tables are in normal form  $n$ .

Newcomers to database design sometimes suppose that normalization proceeds in an iterative fashion, i.e. a 1NF design is first normalized to 2NF, then to 3NF, and so on. This is not an accurate description of how normalization typically works. A sensibly designed table is likely to be in 3NF on the first attempt; furthermore, if it is 3NF, it is overwhelmingly likely to have an HNF of 5NF.

Achieving the "higher" normal forms (above 3NF) does not usually require an extra expenditure of effort on the part of the designer, because 3NF tables usually need no modification to meet the requirements of these higher normal forms.

The main normal forms are summarized below.

**Table 1** Normal Forms

Normal form	Defined by	Brief definition
First normal form (1NF)	E.F. Codd (1970)	Table faithfully represents a relation and has no repeating groups
Second normal form (2NF)	E.F. Codd (1971)	No non-prime attribute in the table is functionally dependent on a part (proper subset) of a candidate key
Third normal form (3NF)	E.F. Codd (1971)	Every non-prime attribute is non-transitively dependent on every key of the table
Boyce-Codd normal form (BCNF)	Raymond F. Boyce and E.F. Codd (1974)	Every non-trivial functional dependency in the table is a dependency on a superkey

Fourth normal form (4NF)	Ronald Fagin (1977)	Every non-trivial multivalued dependency in the table is a dependency on a superkey
Fifth normal form (5NF)	Ronald Fagin (1979)	Every non-trivial join dependency in the table is implied by the superkeys of the table
Domain/key normal form (DKNF)	Ronald Fagin (1981)	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints
Sixth normal form (6NF)	Chris Date, Hugh Darwen, and Nikos Lorentzos (2002)	Table features no non-trivial join dependencies at all (with reference to generalized join operator)

Evaluator's Comments if any:

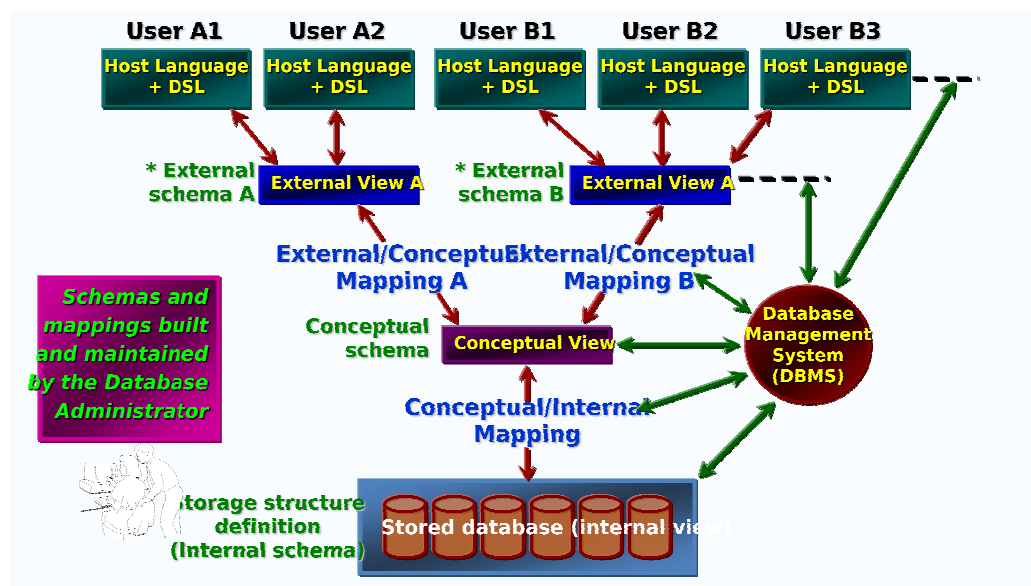
**Question 6** What are views? Explain its various types?

**Answer 6** (Ahmad, 1997)

In 1975, the American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) developed standard based on a three level approach.

Three level approach with data dictionary:

1. External level or View
  2. Conceptual level or View
  3. Internal level or View
- The External level is the way users perceive the data
  - The Internal level is the way the DBMS and the Operating System perceive the data
  - The Conceptual level provides both the mapping and the independence between the external and the internal levels



**Figure 4** Database Views (Ahmad, 1997)



### External View :

- The user's view of the database
- Consists of different external views of the database
  1. entities
  2. attributes
- Relationships
- Different representation of the same data, e.g. date

### Conceptual View:

- Describes what data is stored in the database and their relationships
- Independent of any storage considerations e.g. no. of bytes occupied by Staff No.
- Consists of the logical structure of the database as seen by the DBA
  1. all entities, attributes and their relationships
  2. constraints on data
  3. meaning of data
  4. security and integrity

### Internal View:

- The physical representation of the database in the computer
- Describes how data is stored in the data-base to achieve optimal performance and storage space utilization

1. storage allocation for data and indexes
2. record description for storage
3. record placement
4. data compression & encryption techniques

**Answer 6** (Alternate answer)

In database theory, a view consists of a stored query accessible as a virtual table composed of the result set of a query. Unlike ordinary tables (base tables) in a relational database, a view does not form part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in subsequent invocations of the view.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table
- Views can join and simplify multiple tables into a single virtual table
- Views can act as aggregated tables, where the database engine aggregates data (sum, average etc) and presents the calculated results as part of the data

- Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents
- Depending on the SQL engine used, views can provide extra security
- Views can limit the degree of exposure of a table or tables to the outer world

Just as functions (in programming) can provide abstraction, so database users can create abstraction by using views. In another parallel with functions, database users can manipulate nested views, thus one view can aggregate data from other views. Without the use of views the normalization of databases above second normal form would become much more difficult. Views can make it easier to create lossless join decomposition.

Just as rows in a base table lack any defined ordering, rows available through a view do not appear with any default sorting. A view is a relational table, and the relational model defines a table as a set of rows. Since sets are not ordered - by definition - the rows in a view are not ordered, either. Therefore, an ORDER BY clause in the view definition is meaningless. The SQL standard (SQL:2003) does not allow an ORDER BY clause in a subselect in a CREATE VIEW statement, just as it is not allowed in a CREATE TABLE statement. However, sorted data can be

obtained from a view, in the same way as any other table - as part of a query statement. In Oracle 10g, a view can be created with an ORDER BY clause in a subquery.

Evaluator's Comments if any:

**Question 7** What is Dynamic SQL?

### **Answer 7**

Dynamic SQL is a programming technique that enables you to build SQL statements dynamically at runtime. You can create more general purpose, flexible applications by using dynamic SQL because the full text of a SQL statement may be unknown at compilation. For example, dynamic SQL lets you create a procedure that operates on a table whose name is not known until runtime.

Oracle includes two ways to implement dynamic SQL in a PL/SQL application:

- Native dynamic SQL, where you place dynamic SQL statements directly into PL/SQL blocks.
- Calling procedures in the DBMS\_SQL package.

You should use dynamic SQL only if you cannot use static SQL to accomplish your goals, or if using static SQL is cumbersome compared to dynamic SQL. However, static SQL has limitations that can be overcome with dynamic SQL.

You can use dynamic SQL to create applications that execute dynamic queries, whose full text is not known until runtime. Many types of applications need to use dynamic queries, including:

- Applications that allow users to input or choose query search or sorting criteria at runtime
- Applications that allow users to input or choose optimizer hints at run time
- Applications that query a database where the data definitions of tables are constantly changing
- Applications that query a database where new tables are created often

Language features:

- If you use C/C++, you can call dynamic SQL with the Oracle Call Interface (OCI), or you can use the Pro\*C/C++ precompiler to add dynamic SQL extensions to your C code.
- If you use COBOL, you can use the Pro\*COBOL precompiler to add dynamic SQL extensions to your COBOL code.
- If you use Java, you can develop applications that use dynamic SQL with

JDBC.

Factors to be considered while designing Dynamic SQL. Careful analysis of the below and their affects on the performance needs to be addressed.

1. Permissions
2. Query execution plans
3. Network traffic
4. Using Output parameters
5. How logic is encapsulated

Evaluator's Comments if any:

**Question 8** What is parsing?

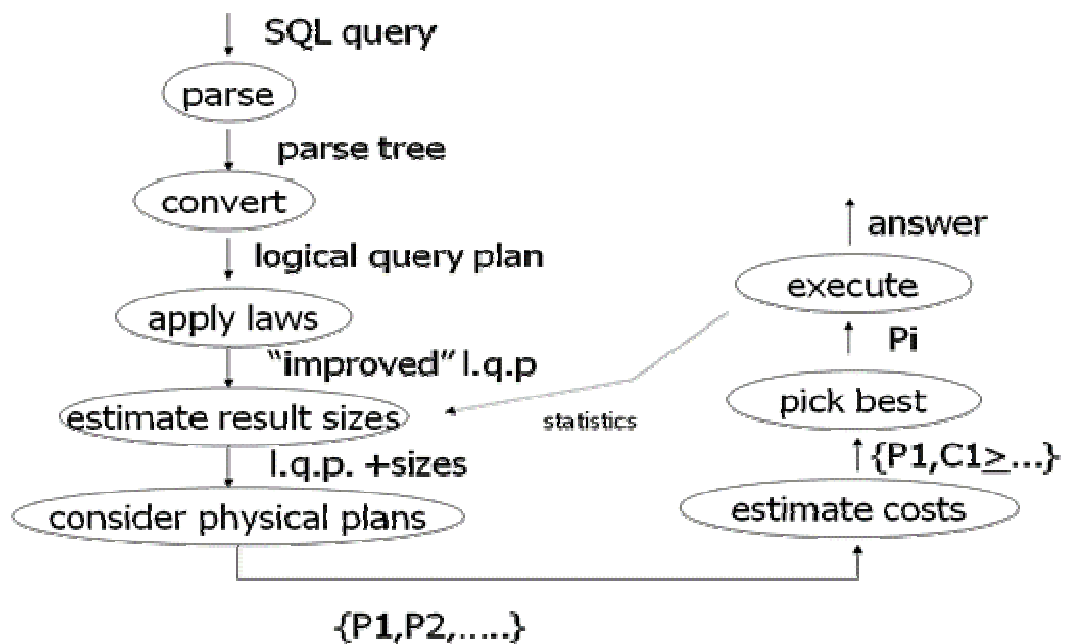
**Answer 8**

Parsing is a subset activity of Database Query Compiler and does the following

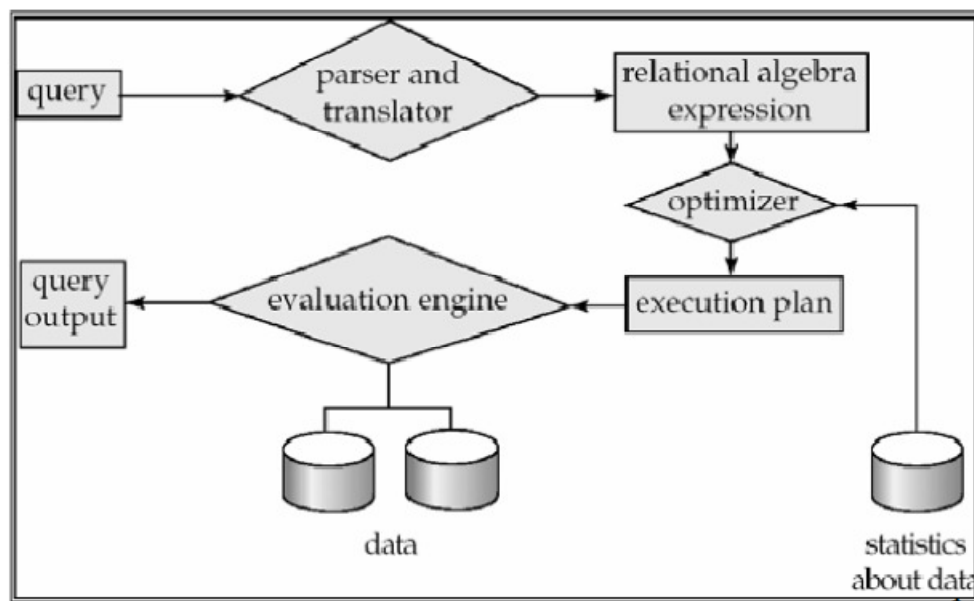
- Parses SQL query into **parse tree**
- Transforms parse tree into **expression tree (logical query plan)**

- Transforms logical query plan into **physical query plan**

Above sequence is pictorially depicted in Figure



**Figure 5** Query Parsing



**Figure 6** Query parsing steps

## Query and parse tree

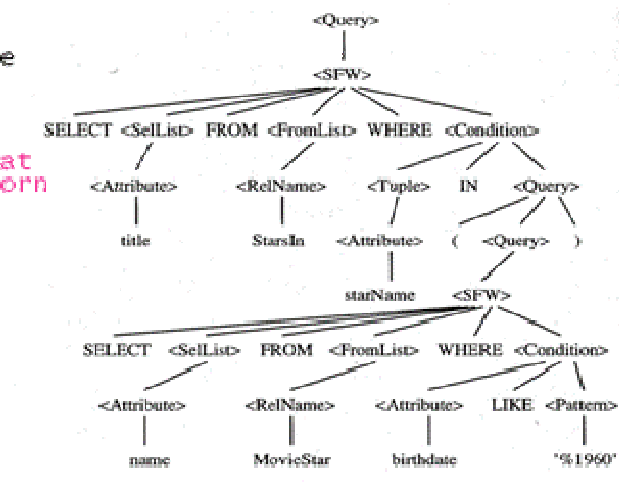
```
StarsIn(
  title,year,starName
)
```

```
MovieStar(
  name,address,gender,bdate
)
```

Query:

Give titles of movies that  
have at least one star born  
in 1960

```
SELECT title
FROM StarsIn
WHERE starName IN (
  SELECT name
  FROM MovieStar
  WHERE
    birthdate LIKE '%1960%'
);
```



**Figure 7** Parsing Tree

In Oracle, every statement, be it a DDL, DML, or anything else, gets parsed. Parsing basically means what Oracle understands about the statement and based on that, how it decides to execute it. This process is also known as Optimization of the query. As the name says, the idea is how best Oracle can process the query or in other words, optimize its execution.

Parsing is of two types, Hard parse and Soft parse. If the said query is found in Oracle's cache, the query optimization is not needed. Oracle can pick up the optimized query and can execute it. If the query is run for the first time or the query's cached version is obsolete or flushed out from Oracle's cache, query needs to be optimized and the process is called Hard parse of the query. Hard parse, in general, is unavoidable as for the very first time, each query needs to



be parsed , at least for once. But in the subsequent executions, query should be simply soft parsed and executed.

The mechanism which works in the backend for doing all this is called Optimizer. There are two versions of it, Rule Based and Cost Based. The Rule Based optimizer(RBO) is made deprecated from Oracle version release 10g. This was not very much efficient as it was "statement driven". The Cost Based is now the only and supported mode of optimizer which is, as the name suggests, cost based and takes into the account the resource consumption of the query.

Evaluator's Comments if any:

**Question 9(i)**

Write a short note on –  
Concurrency control by time stamps

**Answer 9(i)**

Timestamps are processed so that their execution is equivalent to a serial execution in their timestamp order. The timestamp-based concurrency control allows a transaction to read or write a data item  $x$  only if  $x$  had been last written by an older transaction. Otherwise, it rejects the operation and restarts the transaction.

Scheduler tasks:

- Scheduler assign each transaction  $T$  a unique number, it's timestamp  $TS(T)$ .
- The timestamp ordering protocol ensures that any conflicting read and write operations are executed in timestamp order.
- Each transaction is issued a timestamp when it enters the system. If an old transaction  $T1$  has timestamp  $TS(T1)$ , a new transaction  $T2$  is assigned time-stamp  $TS(T2)$  such that  $TS(T1) < TS(T2)$

**Two approaches to generate Timestamps:**

1. Use the value of system, clock as the timestamp; that is, a

transaction's timestamp is equal to the value of the clock when the transaction enters the system.

2. Use a logical counter that is incremented after a new timestamp has been assigned; that is, a transaction's is equal to the value of the counter when the transaction enters the system.

The Rules of Timestamp-Based Scheduling, below example depicts the rules

#### Four Classes

- the scheduler receives a request  $RT(X)$
  - the scheduler receives a request  $WT(X)$
  - the scheduler receives a request to commit  $T$
  - the scheduler receives a request to abort  $T$  or decides to rollback  $T$
- Scheduler receives a request  $RT(X)$ 
    - (a) If  $TS(T) \geq WT(X)$ , the read is physically realizable.
      - i. If  $C(X)$  is true, grant the request.
      - ii. If  $C(X)$  is false, delay  $T$  until  $C(X)$  becomes true or the transaction that wrote  $X$  aborts.
    - (b) If  $TS(T) < WT(X)$ , the read is physically unrealizable, abort  $T$  and restart it with a new, larger timestamp.

- Scheduler receives a request  $WT(X)$

(a) If  $TS(T) \geq RT(X)$  and  $TS(T) \geq WT(X)$ , the write is physically realizable and must be performed.

i. Write the new value for  $X$

ii. Set  $WT(X) := TS(T)$ , and

iii. Set  $C(X) := \text{false}$ .

(b) If  $TS(T) \geq RT(X)$ , but  $TS(T) < WT(X)$ , then the write is physically realizable, but there is already a later value in  $X$ .

i. If  $C(X)$  is true, then the previous writers of  $X$  is committed, and ignore the write by  $T$ .

ii. If  $C(X)$  is false, we must delay  $T$ .

(c) If  $TS(T) \leq RT(X)$ , the write is physically unrealizable, and  $T$  must be rolled back.

- The scheduler receives a request to commit  $T$ .

It must find all the database elements  $X$  written by  $T$ , and set  $C(X) := \text{true}$ .

- The scheduler receives a request to abort  $T$  or decides to rollback  $T$ .

Any transaction that was waiting on an element  $X$  that  $T$  wrote must repeat its attempt to read or write, and see whether the action is now legal

after the aborted transaction's writes are cancelled.

Evaluator's Comments if any:

**Question 9(ii)** Write a short note on –  
Concurrency control by validation

**Answer 9(ii)**

- Another type of optimistic concurrency control
- Maintains a record of what active transactions are doing
- Just before a transaction starts to write, it goes through a “validation phase”
- If there is a risk of physically unrealizable behavior, the transaction is rolled back

Scheduler tasks:

- Keep track of each transaction T's
  - Read set  $RS(T)$ : the set of elements T read
  - Write set  $WS(T)$ : the set of elements T write
- Execute transactions in three phases:
  - *Read*. T reads all the elements in  $RS(T)$
  - *Validate*. Validate T by comparing its  $RS(T)$  and  $WS(T)$  with those in other transactions. If the validation fails, T is rolled back
  - *Write*. T writes its values for the elements in  $WS(T)$

Steps:

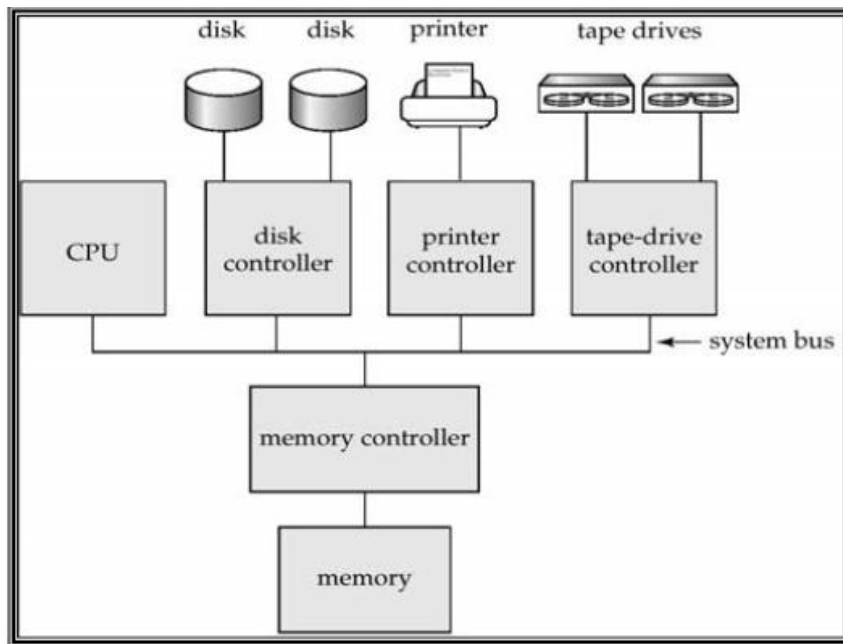
- **START**: the set of transactions that have started, but not yet completed validation. For each T, maintain  $(T, START(T))$
- **VAL**: the set of transactions that have been validated, but not yet finished. For each T, maintain  $(T, START(T), VAL(T))$
- **FIN**: the set of transaction that have completed. For each T, maintain  $(T, START(T), VAL(T), FIN(T))$

Evaluator's Comments if any:

**Question 10** What is centralized and client server architectures?**Answer 10****Centralized Systems:**

- Run on a single computer system and do not interact with other computer systems.
- Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal –however, all processing is done at centralized site.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.

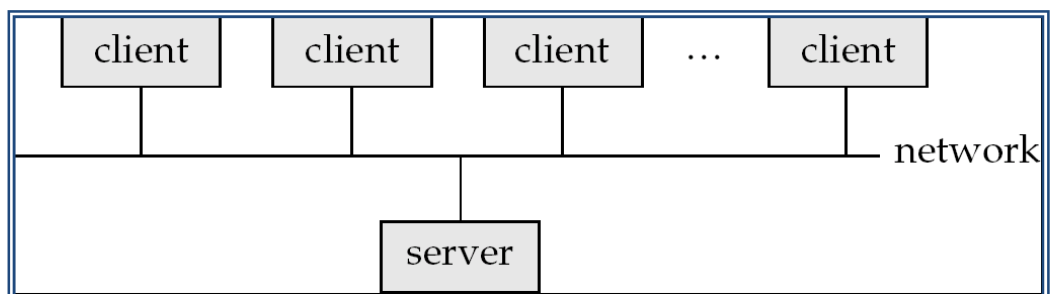
Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system via terminals. Often called *server* systems.



**Figure 8** Centralized Architecture

### Client-Server Systems:

- Server systems satisfy requests generated at  $m$  client systems, whose general structure is shown below:



**Figure 9** Client –Server Architecture

### Specialized Servers with Specialized functions

- Print server



- File server
- DBMS server
- Web server
- Email server

Clients can access the specialized servers as needed

Clients features:

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.

Server features;

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: for Java programming access

- Client and server must install appropriate client module and server module software for ODBC or JDBC

Evaluator's Comments if any:

## Bibliography

Ahmad, E. M. (1997). *CMPB244: Database Design*. Pahang, Malaysia: Universiti Tenaga Nasional.

Anonymous. (2009). *Database normalization*. Retrieved 2009, from Wikipedia:  
[http://en.wikipedia.org/wiki/Database\\_normalization](http://en.wikipedia.org/wiki/Database_normalization)

Goyal, A. (2008, Feb). *STRUCTURE OF DBMS*. Retrieved Oct 2009, from DBMSBASICS:  
<http://dbmsbasics.blogspot.com/2008/02/structure-of-dbms.html>

KSOU. (2009). *Advanced DBMS*. Delhi: Virtual Education Trust.

Mylopoulos, J. (2004). *Conceptual Modeling*. Toronto, Canada: Department of Computer Science, University of Toronto .