Biyani's Think Tank

Concept based notes

Advanced Database System

MCA

Mr Ajay Sharma
Deptt. of IT
Biyani Girls College, Jaipur

Biyani's Group of Girls' Colleges

Published by :
Think Tanks
Biyani Group of Colleges

Concept & Copyright :
©Biyani Shikshan Samiti
Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

Edition : 2012

Leaser Type Setted by :
Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this endeavour. They played an active role in coordinating the various stages of this endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Syllabus

UNIT – I

Object-based Databases : Object-Oriented Databases: Object-oriented data model, Object, Oriented Languages, Persistent Programming Languages. Object-Relational Databases: Nested Relations, Complex Types, . Inheritance, Reference Types, Querying with Complex Types, Functions and Procedures Storage for Object Databases

UNIT - II

Distributed Databases : Distributed Data Storage, Distributed Transactions, Commit protocol,Concurrency Control in Distributed Databases, Availability, Distributed Query Processing

UNIT - III

Parallel Databases : I/O Parallelism, Interquery Parallelism, Intraquery Parallelism, Intraoperation Parallelism, Interoperation Parallelism, Design of Parallel Systems

UNIT - IV

Deductive Databases : Introduction to Recursive Queries, Theoretical Foundations, Recursive Queries with Negation, From Datalog to SQL, Evaluating Recursive Queries

UNIT - V

Information Retrieval and XML Data : Introduction to Information Retrieval, Indexing for Text Search, Web Search Engines, Managing Text in a DBMS, A Data Model for XML, Xquery, Efficient Evaluation of XML Queries.

UNIT - VI

PL/SQL basics, blocks, architecture, variables an constants, attributes, character set, PL/SQL sentence structure, data types, precompiler, conditional and sequential control statements, control structures, conditional control, sequential control, cursors, exceptions, triggers, procedures and packages.

Unit– I

Object-based Databases

1. What do you mean by Object Oriented Databases ?

Ans: Object Oriented Databases are also known as Object Database Management Systems (ODBMS.) Data is stored in form of Objects in these systems, not in form of integers, strings etc. Smalltalk, C++, java and some other languages are used to store data in this form. Data is stored as objects and can be interpreted only using the methods specified by its class.

2. Define Object Oriented Data Models ?

Ans: A data model is a logical organization of the real-world objects(entities), constraints on them and relationships among objects. Object-oriented data model is a model that contains object oriented concepts. This is a collection of objects whose behavior and state, and the relationships are defined according to an object-oriented data model. Some terms used with object oriented data models are :

- **Object:** Any real world entity is uniformly modeled as an object.
- **Message:** Software objects interact and communicate with each other using *messages*
- **Class :** A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind.
- **Inheritance:** A class inherits state and behavior from its superclass. Inheritance provides a powerful and natural mechanism for organizing and structuring software programs.

3. What are Object Oriented Languages ?

Ans: In Object Oriented Programming Languages, we used Objects to design applications and computer programs. Some features are also included in these language techniques like encapsulation, inheritance, polymorphism, data abstraction, messaging . In modern time , many programming languages supports OOP. The first language which supported the primary features of OOL was SIMULA. Languages designed mainly for OO programming, but with some procedural elements are Java, C++, C#, VB.NET, Python.

Languages that are historically procedural languages, but have been extended with some OO features are Visual Basic , Fortran, COBOL 2002, PHP, Perl.

4. What are Persistent Programming Languages?

Ans: Persistent Programming Language is a programming language which is used to handle persistent data. Here, the query language integrated with the host language and both shares the same type system .Objects can be created and stored in the database without any format change. A programmer can modify persistent data without having to write code explicitly to fetch it into memory or store it back to disk.

C++ and Smalltalk which are persistent version of object oriented languages are mostly used because programmers can change data directly from the programming language, without using a data manipulation language like SQL.

5. What is Object Relational Databases ?

Ans: Object relational data models extend the relational data model by providing a much effective system which includes object orientation. Some constructs are also added to relational query languages such as SQL to deal with the added data types. Object relational database systems provide an efficient and easy migration path for users of relational databases who want to use object-oriented features.

6. Define Nested Relations.

Ans: The nested relational model is an extension of the relational model in which domains may be either atomic or relational valued. So, the value of a tuple on an attribute may be a relation, and relations may be stored within relations.

A complex object can be treated by a single tuple of a nested relation.

If we treats a tuple of a nested relation as a data item ,then we have a one-to-one correspondence (equality) between these data items and objects in the user's view of the database.

7. Define Complex Types Systems.

Ans: Complex type systems and object orientation enable E-R model concepts, like generalization , identity of entities, multivalued attributes and specialization. which are represented directly with simple translation to the relational model. No complex translation will required for this task. Example :

```
Create type MyStr char varying.
Create type MyDt
        ( day integer,
          month char (10),
          year integer)
Create type Document
        ( name MyStr,
         author-list setof (MyStr),
         date MyDt,
         keyword-list setof (MyStr))
create table doc of type Document
```

The first statement defines a type MyStr, which is a variable length character string. The next statement defines a type called MyDt, which has three components : day, month and year. The third statement defines a type Document, which contains a name, an author-list, which is a set of authors, a date of type MyDt, and a set of keywords. Finally, the table doc containing tuples of type Document is created. This table definition differs fro table definition in ordinary relational databases, since it allows attributes that are sets and attributes such as MyDt that are structures. The features allow composite attributes and multivalued attributes of E-R diagrams to be represented directly. Type definitions in programming languages, are not stored in a database, and can be seen only by programs that include a text file containing the definitions. Complex type systems usually support other collection types, such as arrays andmultisets.

8. Define Inheritance.

Ans : Inheritance gives the hierarchical relationships between different objects at different levels and gives code reusability. It is the most important feature of the object database . Inheritance can be at the level of types, or at the level of tables.

Example : Suppose that we have the following type definition for PEOPLE :

```
create type Person
( name  MyString,
 social-security integer )
```

We may want to store extra information in the database about Teachers and Students which are part of Person. Since Students and Teachers are also people, we can use inheritance to define the student and teacher types as follows :

```
create type Student
( degree MyString,
department MyString )
under Person

create  type Teacher
( salary integer,
 department MyString )
under Person
```

Both Student and Teacher inherit the attributes of Person – name & social-security. Student and Teacher are said to be subtypes of Person, and Person is a supertype of Student, as well as of Teacher.

Object-relational systems can model such a feature by using inheritance at the level of tables, and allowing an entity to exist in more than one table at once.
Example :

```
Create table people
 ( name MyString,
social-security integer)
```

Then, we define the Students and Teachers tables as follows :

```
create table students
( degree MyString,
 department MyString )
under people

create table teachers
( salary  integer,
 department MyString )
under people
```

The subtables Students & Teachers inherit all the attributes of the table People.

Inheritance makes schema definition more natural. Without inheritance of tables, the schema designer has to link tables corresponding to subtables explicitly with tables corresponding to supertables via primary keys, and to define constraints between the tables to ensure referential constraints.

9. Define Reference Types.

Ans : Object –oriented languages is capable to refer to objects. An attribute of a type can be a reference to an object of a specified type. For example, references to people are of the type ref(Person). We can redefined the author-list field of the type Document as **author-list setof (ref(Person))** . This is a set of references to Person objects.

References to tuples of a table can implemented by using the primary key of the table. Each tuple of a table may have a tuple identifier as an implicit attribute, and a reference to a tuple is the tuple identifier. Subtables implicitly inherit the tuple identifier attribute, just as they inherit other attributes from supertables.

10. Define Query with Complex types.

Ans: An extension of the Structured Query Language (SQL) can deal with complex types. It is described in following manner :

- **Relation –Valued Attributes :**
  - Extended SQL allows an expression to appear where a relation name may appear, such as in a FROM clause. The ability to use subexpression makes it possible to take advantage of the structure of nested relation.

    Example :

    ```
    Create table pdoc
    ( name MyString,
    author-list setoff(ref(people)),
    date MyDate,
    keyword-list setof(MyString))
    ```

  To find all documents having a keyword "Database" the query used is :

  ```
  Select name
  From pdoc
  Where "database"  in keyword-list
  ```

  Aggregate functions can be applied to any relation-valued expression.

  Query is:

  ```
  select name, count(author-list)
    from pdoc
  ```

- **Path Expressions :**

  The dot notation for referring to composite attributes can be used with references

  Example:

  ```
  Create table phd-students
  ( advisor ref(people))
  Under people
  ```

  To get the names of the advisors of all phd students , query is :

  ```
  Select phd-students.advisor.name
  From phd-students
  ```

  An expression of the form "phd-students.advisor" is called a path-expression. References can be used to hide join operations and thus the use of references simplifies the query considerably.

- **Nesting and Unnesting :**
  - The transformation of a nested relation into 1NF is called unnesting. The doc relation have two attributes **author-list** and **keyword-list**, which are nested relations.
  - It also have two attributes **name** and **date**, which are not nested.
  - If we want to convert the relation into a single flat relation, with no nested relations or structured types as attributes, then we can use this query :

    ```
    Select name, A as author, date.day, date.month, date.year, K as
    keyword
    From doc as B, B.author-list as A, B.keyword-list as K.
    ```

  The variable B in the from clause is declared to range over doc.

The variable A is declared t range over the authors in author-list of the document and K is declared to range over the keywords in the keyword-list of the document.

- o Nesting is Transforming a 1NF relation into a nested relation.
- o If we use Grouping in SQL then a temporary multiset relation is created for each group and an aggregate function is applied on the temporary relation.
- o By returning the multiset instead of applying the aggregate function, we can create a nested relation.

Example :

```
Select title, author, (day, month, year) as date,
        set (keyword) as keyword-list
From flat-doc
Groupby title, author,date
```

11. Define Functions and Procedures for object databases.

Ans: Object relational systems allow functions to be defined by users. These can be defined in a programming language like C, C++ or in a data manipulation language like SQL.

If we want to create a function that given a document, returns the count of the number of authors then we can write following statement :

```
Create function author-count (one-doc document)
Returns integer as
select count(author-list)
from one-doc
```

The function can be used in a query that returns the name of all documents that have more than one author :

```
Select name
From doc
Where author-count(doc)>1
```

Some databases systems allow us to define functions using a programming language such as C or C++. Functions defined in this way can be more efficient than functions defined using SQL. An example of the use of such functions would be to perform a complex arithmetic computation on the data in a tuple.

A Stored Procedure is different from a simple SELECT statement. Stored procedures can have complex code, means that they contains loops, conditions, and other program logic. Stored Procedures doesn't return any data . Mostly Stored Procedures are used by Database Administrator for maintenance.
Example:

```
Create Procedure  usp_Test AS
Select au_fname, au_lname
From  authors
Order by au_lname;
```

Unit – II

Distributed Databases

1. Define Distributed Databases .

Ans: A distributed database system consists of loosely coupled sites that share no physical components. The data may be stored on same physical location or on different location. All computers are interconnected with network. Each site may participate in the execution of transactions that access data at one site or several sites. The main difference between centralized and distributed database system is that the data is stored on single location in centralized system whereas in the distributed system, the data resides in several locations.

2. Define Distributed Data Storage.

Ans: There are several approaches to storing this relation in the distributed databases .

- **Replication :** In this approach, the system maintains several identical copies of the relation. Each relation is stored at different site.
- **Fragmentation :** In this approach, the relation is partitioned into several fragments. Each fragment is stored at a different site.
- **Replication and fragmentation :** The relation is portioned into several fragments. The system maintains several copies of each fragment.

3. What do you mean by Distributed Transactions?

Ans: A distributed transaction includes one or more statements, that update data individually or as a group, on two or more nodes of a distributed database. There are two types of transactions : Local Transactions & Global Transactions.

Local transactions access and modify data in only one local database. Global transaction access and modify data in several databases. Like other transactions, distributed transactions must have all four ACID properties. Each site has its own Local Transaction Manager whose function is to ensure the ACID properties of those transactions that execute at that site. The various transaction involves in execution of global transaction.

4. Define Commit Protocol.

Ans: All statements in a transaction (distributed or non distributed) must commit or rollback as a unit .The effects of an ongoing transaction should be invisible to all other transactions at all nodes. Two-phase commit allows groups of transactions across several nodes to be treated as a unit, means either all transactions commit, or they all get rolled back.

- **2 Phase Commit (2 PC):**

  The two-phase commit consists of two phases – Prepare Phase & Execute Phase

  Prepare Phase :

  - The coordinator sends a message "prepare for commit" to each node to get ready for committing the transaction.
  - All database who are participating receive the message will force-write all the transaction details to disk and then send a "ready to commit" or "OK" signal to the coordinator.
  - If the force-writing to the disk fails or if the local transaction cannot commit for some reason, the participating database sends a "cannot commit" or "Not OK" signal to the coordinator.

  Commit Phase:

  - If all participating databases reply "OK", the coordinator signals "OK". This means that the transaction is successful.
  - The coordinator sends a "commit" signal for the transaction to all the participating databases.
  - All participating databases completes the transaction by permanently modifying the database.
  - If any databases has given a "Not OK" signal, then the communicator also signals a "Not OK".
  - The "rollback" or undo message will send by coordinator to the local effect of the transaction to each participating database.

- **3 Phase Commit (3 PC):**
  - 3 Phase Commit is a non blocking protocol.
  - It was developed to avoid the failures that occur in two-phase commit transactions.
  - The 3PC also has a coordinator who initiates and coordinates the transaction
  - 3PC has an additional phase called Pre-Commit phase which will remove the uncertainty period for participants that have committed and are waiting for the global abort or commit message from the coordinator.
  - When receiving a pre-commit message, participants know that all others have voted to commit. If a pre-commit message has not been received the participant will abort and release any blocked resources.

5. Define Concurrency Control in Distributed Database.

Ans: To control Concurrency in distributed database, we have 2 options :

- **Locking Protocols** : The various locking protocols used in a distributed environment are:

  - **Single-Lock Manager Approach** –
    The system maintains a single lock manager that resides in a single site. When a transaction needs to lock a data item, it sends a lock request to that site. The lock manager decides that the lock can be granted immediately or not. If the lock can be granted, the lock manager sends a message to the site at which the lock request was initiated. Otherwise, the request is delayed until it can be granted, at which time a message is sent to the site at which the lock request was initiated.

  - **Multiple Coordinators –**
    In this approach, the lock manager function is distributed over several sites. Each lock manager watch and check the lock and unlock requests for a subset of data items. Each lock manager resides in a different site.

This approach complicates deadlock handling, because the lock and unlock requests are not made at a single site.

- **Majority Protocol –**
  In this protocol, each site maintains a local lock manager whose task is to administer the lock and unlock requests for those data items that are stored in that site. This scheme deals with replicated data in a decentralized manner, thus avoiding the drawbacks of central control. This protocol is more complicated for implementation. Deadlock handling is also an complicated task because lock and unlock requests are not made at one site.

- **Biased Protocol –**
  This protocol is same as Majority Protocol but the difference is that requests for shared locks treated more effectively than requests for exclusive locks. The system maintains a lock manager at each site. Each manager manages the locks for all the data items stored at that site. Shared and Exclusive locks are handled differently.
  In this protocol overheads on read operations are less than majority protocol, but has additional overheads on writes. Deadlock handling is complex here like majority protocol.

- **Primary Copy –**
  In the case of data replication, one of the replicas (copies) designated as a primary copy. Concurrency control for replicated data handled in a manner similar to that of unreplicated data. Implementation , but if primary site fails, the data item is unavailable, even though other sites may have a replica.

- **Timestamping :** Each transaction is given a unique timestamp that the system uses in deciding the serialization order. There are 2 primary methods for generating unique timestamps: first is centralized and second is distributed.

In the centralized scheme, a single site is chosen for distributing the timestamps. The site can use a logical counter or its own local clock for this purpose.
In the distributed scheme, each site generates a unique local timestamp using either a logical counter or the local clock. We obtain the unique global timestamp by concatenating the unique local timestamp with the site identified, which also must be unique. The order of concatenation is important.

6. Define availability in context of distributed database.

Ans: The time for which system is not fully usable should be extremely low. An approach is based on the idea of dividing the database into fragments and assigning each fragment a controlling entity called an agent to escape from communication failures and network partitions. High data availability is included in this approach

7. What do you mean by Distributed Query Processing?

Ans: In a distributed system, two things matters first is the cost of data transmission over the network and second is the potential gain in performance from having several sites process parts of the query in parallel. The relative cost of data transfer over the network and data transfer to and from disk is different and depends on the network's type and on disk's speed.
Strategy used for query processing are :

- **Query Transformation –** If a relation is replicated, there is a choice of replica (copy) to make. If no replicas are fragmented, then the lowest transmission cost copy is selected . However, if a copy is fragmented, the choice is not so easy to make, we need to perform multiple joins or unions to reconstruct the relation.
- **Simple Join Processing --** A major aspect of the selection of a query-processing strategy is choosing a join strategy. In many strategies it is not compulsory that all strategies are best. Among the factors that must be considered are the volume

of data being shipped, the cost of transmitting a block of data between a pair of sites, and the relative speed of processing at each site.

- **Semi Join Strategy --** This strategy is particularly advantageous when relatively few tuples of a relation contribute to the join. In case of joining of multiple relations, this strategy can be extended to a series of semijoin steps.

Unit III

Parallel Databases

1. Define Parallel Database Systems.

Ans: Parallel Systems are designed to improve processing and I/O speeds by using multiple CPUs and disk in parallel. These systems are used for those applications where query is executed on extremely large databases and large number of transactions processed per second. This capability is not available in centralized and client-server database systems. Data can be partitioned across multiple disks for parallel I/O. Individual operations like sorting, join, aggregation can executed in parallel. Many operations are performed simultaneously in parallel systems. Also used for storing large volume of data and process those queries which consumes time. There are several architectural models for parallel machines like:

- Shared Memory : All the processors share a common memory in this model.
- Shared Disk: All the processors share a common disk.
- Shared Nothing : The processors share neither a common memory nor common disk.
- Hierarchical : Combines the characteristics of all models.

2. Define I/O Parallelism.

Ans: I/O reduces the time required to retrieve relations from disk by dividing (partitioning) the relations on several disks. Generally, Horizontal Partitioning is used for data partitioning in parallel database. Tuples of a relation are divided among multiple disks in horizontal partitioning, because of this, each tuple resides on one disk. Several partitioning techniques are used for data partitioning.

- Round-Robin: In this scheme, tuples are distributed equally across multiple disks, means each disk contains same number of tuples. This scheme is best for those applications where the entire relation is read for each query. No clustering is possible because tuples are scattered across all disks.

- Hash Partitioning : In this scheme, one or more attributes from the given relation's schema are designated as the partitioning attributes. This scheme is best suited for point queries based on the partitioning attribute. Query is directed to a single disk which will save the startup cost of a query on multiple disks. Other disks remains free to process other queries. This partitioning is also useful for sequential scan of the entire relation.

- Range Partitioning: In this scheme, adjacent (contiguous) attribute-values ranges are distributed to each disk. This scheme is best for point and range queries on the partitioning attribute. In both cases, the search is narrowed to exactly those disks that might have any tuples of interest.

3. **What is Interquery Parallelism ?**

Ans: Interquery parallelism is the easiest form of parallelism to support in database system. Different queries or transactions execute equivalent (parallel) with one another in this parallelism. It increases the throughput of transaction. Mainly used to improve a transaction processing system to support huge number of transactions per second. It is complicated in shared-disk or shared-nothing architecture. Locking and logging is performed by processors in by passing messages to each other. Interquery parallelism is supported by Oracle7 & Oracle Rdb.

4. What is Intraquery Parallelism ?

Ans: In Intraquery parallelism, a single query executes on multiple processors and disks in parallel. Basically it is used for speed up long run queries which is not possible Interquery parallelism. We can parallelize a query by parallelizing individual operations. The execution of a single query can be parallelized in two ways :

- Intraoperation Parallelism: By parallelizing the execution for each individual operation, like sort, select, project and join, processing of query can be speed up.

- Interoperation Parallelism: By executing different operations in a query expression in parallel, we can speed up processing of query.
  These two forms can be used simultaneously on a query. In a typical query, number of operations are small compared to the number of tuples processed by each operation, so in this situation the first form can scale better with increasing parallelism but in typical parallel systems where number of processors are small, both forms are important.

5. What is Intraoperation Parallelism?

Ans: Relational operations work on relations containing large set of tuples, so the execution of these operations can parallelize on different subsets of relations. Because the number of tuples in a relation can be large, the degree of parallelism is potentially vast. Parallel versions of some common relational operations are :

A. Parallel Sort
   a. Range-partitioning Sort
   b. Parallel External Sort-Merge
B. Parallel Join
   a. Partitioned Join
   b. Fragment and Replicate Join
   c. Partitioned Parallel Hash-Join
   d. Parallel Nested Loop Join

A. Parallel Sort : To sort each partition separately, the relation has been range partitioned on the attributes on which it is to be sorted, and can concatenate the results to get the full sorted relation. Because the tuples are partitioned on multiple disks, the time required for reading the entire relation is reduced due to parallel access. If the relation has been partitioned in any other way, we can sort it in one of two ways :

a. **Range- Partition Sort :** In this type of sorting it is not compulsory to range the relation on the same processors or disks as those on which that relation is stored. We redistribute the relation using a range-partition strategy, such that all tuples that lie within the i^{th} range are sent to processor Pi, which stores the relation temporarily on disk Di. Each partition of relation is sorted by its processor locally without interaction with the other processors. Each processor executes the same operation on a different data set.

b. **Parallel External Sort-Merge :** In this technique, each processor Pi locally sorts the data on disk Di then the sorted runs on each processor are then merged to get the final sorted output.

   Some machines, such as the Teradata DBC series machines, use specialized hardware to perform merging.

B. **Parallel Join :** Pairs of tuples are tested to check that they satisfy the join condition or not. If they satisfy the condition, then the pair is added to the join output. Some types of joins are :

a. **Partitioned Join :** Some types of joins like equi-join and natural join, it is possible to partition the two input relations across the processors, and to compute the join locally at each processor. The concept of this partitioning is same as the partitioning of hash-join. There are two different ways of partitioning :

   i. Range partitioning on the join attributes : Same partition vector must be used for both relations.

   ii. Hash partitioning on the join attributes : Same hash function must be used on both relations.

b. **Fragment and Replicate Join :** Partitioning is not applicable to all types of joins. For example it is possible that all tuples in a relation r join with some tuples in relation s. So in this case partitioning is not applicable. We can parallelize such joins using a technique called fragment and replicate.

This technique usually has a higher cost than does partitioning, when both relations are of roughly the same size, since at least on of the relations has to be replicated. A special case of fragment and replicate – asymmetric fragment and replicate which works as follows :

i. Any partitioning technique can be used on relation ,including round-robin partitioning.
ii. The other relation is replicated across all the processors.
iii. Processor then locally computes the joins of relation1 with all of relation2, using any join technique.

c. **Partitioned Parallel Hash-Join :** The hash-join performed at each processor is independent of that performed at other processors and receiving the tuples of relation1 and relation2 is similar to reading them from disk. Hybrid hash-join algorithm is used to cache some of the incoming tuples in memory so that cost of writing and reading can avoid.

d. **Parallel Nested Loop Join :** To understand this technique we use an example. Suppose a relation s is smaller than relation r and the r is stored by partitioning. There is an index on a join attribute r at each of the partitions of relation r. Asymmetric fragment and replicate is used, with relation s being replicated, and with the existing partitioning of relation r.Each processor Pj on which partition of relation s is stored, reads the tuples of relation s stored in Dj , and replicates the tuples to every other processor Pi. When this phase ends, relation s is replicated at all sites that store tuples of relation r.

6. What is Interoperation Parallelism?

Ans: Two types of interoperation parallelism are :–

- **Pipelined Parallelism :** This is an important source of economy of computation for database query processing. Main advantage of this execution in a sequential evaluation is that we can achieve a sequence of such operations without writing any of the intermediate results to disk. Pipelining can be used as a source of parallelism as well, in the same way that instruction pipelines are used as a source of parallelism in hardware design. Pipeline Parallelism is useful with small number of processors, but it does not develop well. Pipeline chains not manage sufficient length to provide a high degree of parallelism, it is also not possible to pipeline relational operators that do not produce output until all inputs have been accessed, minor speedup is get for the frequent cases in which one operator's execution cost is higher than others. Pipelining is used mostly because it not write intermediate results to disk.

- **Independent Parallelism :** In Independent Parallelism, operations in a query expression do not depend on one another. It can be executed parallel. It does not provide a high degree of parallelism. It is fewer useful in a highly parallel system, but it is useful with a lower degree of parallelism.

7. Define design of Parallel Systems.

Ans: For storage of large volumes data, large scale parallel database systems are used.Decision support queries on this data is also processed here. If we are handling large volumes of incoming data, then parallel loading of data from external sources also importantly required. A large parallel database system must also ensures the resilience to failure of some processors or disks and online reorganization of data and schema changes. In case of large number of processors and disks, there are more chances that at least one processor or disk will fail or not work properly. In this situation, the functioning of system will stop. To avoid this type of situation, large scale parallel database systems, such as Tanden and Teradata machines, were designed. Here , data is replicated across at least two processors. If a processor fails, the data that is stored can still be accessed from the other processors. The system keeps track of failed processors and distributes the work among functioning processors.

If data is in large volume then some operations like changes in schema, adding a column in a relation etc can take long time, may be hours or days. So this is not acceptable for the database system to unavailable while these operations are in progress. Parallel database systems like Tandem systems allow such operations to be performed on-line.

Unit– IV

Deductive Databases

1. Define Deductive Database.

Ans: Deductive databases are an extension of relational databases which support more complex data modeling. The main feature of deductive database is their capability or supporting a declarative, rule based style of expressing queries and applications on database. Deductive Databases are coming of age with the emergence of efficient and easy to use systems that support queries, reasoning, and application development on databases through declarative logic-based languages.

2. Define Recursive Queries.

Ans: Recursive processing is one of the most important programming and problem-solving model (example) . Many problems can easily be expressed in terms of recursion. Database systems present the area in which recursive processing may be very useful, as information stored in them could often be read as a graph of some sort that can be the subject of recursive processing. Support for recursive queries in databases is usually not acceptable. This specially concerns alternates of SQL. On the other hand, query languages from the Datalog family provide wide recursive querying capabilities, but are very limited concerning other features and frequently not used by database designers. A full business database system based on Datalog does not exist. Three approaches to recursive query processing implemented for the Stack Based Query Language (SBQL) are transitive closures, fixpoint equation systems and recursive functions. Recursive queries may be very expensive to evaluate so this requires possible optimization techniques for evaluation. Although recursion is important but it is not supports and implements SQL standards. Integration of recursive functions or recursive views with a query language requires generalizations. Currently very few existing query languages have Recursive Queries in Databases.

3. What is Datalog?

Ans: Datalog is a query and rule language used with deductive databases and syntactically is a division (subset) of Prolog. Some queries which are not expressed in relational algebra or SQL and present a more powerful relational language called Datalog. A Datalog program consists of a set of rules. Recursion is established by permiting the relational symbols in the rules. Three different but equivalent approaches to define its semantics are the model-theoretic, proof-theoretic and fixpoint approaches. Datalog inherits these properties from logic programming and its standard language Prolog. The main difference between Datalog and Prolog is that function symbols are not allowed here. Several techniques have been projected for the efficient evaluation of Datalog programs. More recent research on data integration has found Datalog to be a useful conceptual specification tool. In recent years, Datalog has found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.[3]

4. Define Theoretical Foundations.

Ans: Relations in Datalog program are categorized as either output relations or input relations. Output relations are described by rules and input relations have a set of tuples clearly listed. The meaning of a Datalog program is usually defined in two different ways, both of which essentially describe the relation instances for the output relations. There are two approaches to defining a Datalog program .
First approach defines Datalog program is the least model semantics, it gives users a way to understand the program without thinking about how the program is to be executed. That is, the semantics is declarative, like the semantics of relational calculus, and not operational like relational algebra semantics. This is important because the presence of recursive rules makes it difficult to understand a program in terms of an evaluation strategy.

The second approach is least fixpoint semantics, gives a conceptual evaluation strategy to compute the desired relation instances. This serves as the basis for recursive query evaluation in a DBMS. More efficient evaluation strategies are used in an actual implementation, but their correctness is shown by demonstrating their equivalence to the least fixpoint approach. The fixpoint semantics is thus operational and plays a role equivalent to that of relational algebra semantics for nonrecursive queries.

5. Define Recursive queries with negation.

Ans: To understand recursive queries with negation, we use a table and some rules :

Part	Subpart	Qty
Trike	Wheel	3
Trike	Frame	1
Frame	Seat	1
Frame	Pedal	1
Wheel	Spoke	2
Wheel	Tire	1
Tire	Rim	1
Tire	tube	1

Assembly Relation (table)

Now Consider the following rules :

Big(Part) :- Assembly(Part, Subpart, Qty), Qty > 2,
not Small(Part).

Small(Part) :- Assembly(Part, Subpart, Qty),
not Big(Part).

Suppose these rules attempt to divide Parts attribute into two classes (big & small)

In the first rule, Big part must use more than 2 copies of some subpart and this is not categorized as small parts. In the second rule, Small is defined as the set of parts that is not categorized as big parts.According to this example, a single part strike uses more than 2 copies (3 copies) of some subpart.

If we apply the first rule and then the second rule, this tuple is in Big.To apply the first rule, we consider the tuples in Assembly, choose those with Qty >2 (which is just htrikei), discard those that are in the current instance of Small (both Big and Small are initially empty), and add the tuples that are left to Big. Therefore,an application of the first rule adds strike to Big. Proceeding similarly, we can see that if the second rule is applied before the first, strike is added to Small instead of Big.

In this example there are 2 fixpoints and both are same in numbers (equal) . The first fixpoint has a Big tuple that does not show in the second fixpoint, so it is not smaller than the second fixpoint. The second fixpoint has a Small tuple that does not appear in the first fixpoint; therefore, it is not smaller than the first fixpoint. The order in which we apply the rules determines which fixpoint is computed, and this situation is very unacceptable.

The main problem here is the use of not. When we apply the first rule, some inferences are rejected because of the existence of tuples in Small. Parts that satisfy the other conditions in the body of the rule are candidates for addition to Big, and we eliminate the parts in Small from this set of candidates. Thus, some inferences that are possible if Small is empty are rejected if Small contains tuples (generated by applying the second rule before the first rule).If not is used, the addition of tuples to a relation can reject the inference of other tuples. Without not, this situation can never arise, the insertion of more tuples to a relation can never disallow the inference of other tuples.

6. Define efficient evaluation of recursive queries.

Ans: After studying the evaluation of recursive queries , we found many problems with newly **i**ntroduced fixpoint operation . A simple approach to evaluating recursive queries is to compute the fixpoint by repeatedly applying the rules. We perform multiple iterations to reach the least fixpoint . Iteration is a application of all program rules. But this approach has two disadvantages:

- **Repeated inferences:** inferences are repeated across iteration, means the same tuple is continually using the same rule and the same tuples for tables in the body.

- **Unnecessary inferences: If** we want to find some data from a column of a table, then calculating whole table is not good approach.

Fixpoint Evaluation without repeated interferences :

Naive fixpoint Evaluation means computation of fixpoint by regularly applying all rules. Naive evaluation is guaranteed to compute the least fixpoint, but every application of a rule repeats all inferences made by earlier applications of this rule. To solve this repetition of inferences consists of remembering which inferences were carried out in earlier rule applications and not carrying them out again. It is possible by keeping track of those components tuples who were generated for the first time in the most recent applicaton of the recursive rule. The refinement of fixpoint evaluation is called Seminaive fixpoint evaluation.

To implement Seminaive fixpoint evaluation for general Datalog programs, we apply all the recursive rules in a program together in an **iteration**.
A delta version of every recursive predicate is used to keep track of the tuples generated for this predicate in the most recent iteration. The delta versions are updated at the end of each iteration.The original program rules are rewritten to ensure that every inference uses at least one delta tuple, that is, one tuple that was not known before the previous iteration. This property ensures that the inference could not have been performed in earlier iterations.

The Magic Sets Algorithm
Magic set algorithm is one output relation and one recursive rule. The algorithm can be applied to any Datalog program. The input to the algorithm is a program and a **query form**.The output of the algorithm is a rewritten program.
Magic tuple can be added to the rewritten program and evaluate the least fixpoint of the program Magic Sets algorithm has turned out to be quite effective for computing correlated nested SQL queries, even if there is no recursion.

Unit V

Information Retrieval and XML Data

1. Explain overview of XML.

Ans: XML means eXtensible Markup Language. XML is used to exchange data across the web. Data can be represented in hierarchical format . Elements and tags are also used here. XML is a metalanguage and it is a proper subset of SGML. XML is also a data description language. XML can also used to represent relations, which could be useful for importing data into or exporting data out of a relational database. Like conventional database data, a XML document usually has some information. This information can be specified by Document type definition (DTD) or an XML Schema.

Document Type Definition (DTD) : DTD describes the rules for interpreting an XML document. It guarantees that the contents of an XML document must follows the given rules.

XML Schema : In an XML documents data elements, attributes and their relationships are defined by schema. We can differentiate DTD and XML Schema by their rules and discriptions.

2. Define Information retrieval and indexing for text search.

Ans: In XML, searching and data retrieval is easily possible. Search engines use the XML tags to specify to search on specific field instead of whole text searching of a page. XML documents provide *structured* data storage. The search indexing program and search engine must understand the DTD or other schema of the documents in the indexed collection (this is easier on local sites and intranets than on the entire Web). Tools can convert from one known structure to another, but it can't be completely automatic, human judgement is required.

Some differences between querying and text searching are:

- Text search works on indexed data, so only the indexing application must recognize the fields and hierarchies and store the information as metadata for each word entry. In a structured query, the system cannot assume a structured index: the query itself must convey more about the structure of the document.
- Text search applies to documents as a whole, while query languages are concerned with different kinds of data like a single tagged field or a record made up of several fields, and there may be several of these records in a single document, or fields from several tables joined together.
- Text search returns a list of documents with some information about them as the result. Query languages will often return the data extracted from a document, such as a tagged field or multiple records.
- In addition to "selection" (finding matches), query languages often must perform computations and formatting transformations on found items, combine data from multiple sources and even update documents automatically. Text search does none of these.
- Search engines will have to deal with each standard structure in the collection of indexed documents.

3. **Explain Web Search Engine.**

Ans: Web search engine is created to search information on the World Wide Web. A list of result represent the result of search which is called search engine result pages (SERPs). This information contains web pages, information , images etc. Data mining is also performed by some search engines. Real-time information is also maintained by search engines by using algorithms.

Web search engines store information about many web pages, which they receive from the HTML and XML. Web crawler is used to receive these pages. Web crawler is a web browser which follows each link on the site.

For indexing, the contents of all pages are analyzed . Data about web pages are stored in an index database. This can be used in future for queries . A query can be a single word or it may be a combination of multiple words. Indexes are used for fast data retrieval. Partial or all information about web page is stored by some search engines like Google etc and some search engines atore each word of each page like Altavista.

When a query is entered into a search engine by user then the engine checks its index and give a best matching web page list according to the query's criteria.Concept based searching is also possible in which the search includes statistical analysis on pages for the words which are used in search criteria. A user can type a question and can get desired result in natural language queries.

4. **Explain XML Data Model.**

Ans: The XML data model is a simple model, it is also abstract. Restricted applications can share some common information. The main structure of an XML document is tree-like, but there is also a way to make connections between nodes in a tree. The XML has a number of different types of nodes: **Element Nodes:** This contains type, attribute name and attribute values which consists of string The type, attribute names and attribute values consist of strings of characters. Some restrictions type and attribute names are that they must consist of letters, digits,dashes and dots, they must be at least one character long and they must start with a letter or a dash. There are no restrictions on attribute values, they may be empty. Attribute names can not start with the four letters "xml-" because these are reserved for xml-link. "id" is reserved for the ID of the element se we cannot use this also.

Document Node : This is a special kind of element node . It has an optional URL. The aim of the URL is to specify a special data model for the node and its children.The type and URL must be character strings. No constraints are allowed with URL. XML tree's root node may be an unnamed document node, which does not have a type and a URL.

Comment Node : This node always a leaf node and only a comment. These nodes are used to include illustrative notes for human use,

Data Node : These are also always leaf nodes. All other nodes have delimiters but this node have not any delimiter. Whatever is written between '< >' (angular brackets) is data. Data nodes contains minimum one characters, they cannot left blank.

5. Explain XQuery.

Ans: XQyery is a query and functional programming language which is used to query on collection of XML data. XQuery works for XML just like SQL works for databases. It is built on XPath expressions. All major databases supports XQuery and it is a W3C Recommendation. It is used for finding and retrieving elements and attributes from XML documents.

Basically XQuery is used to generate summary reports, extract information to use in a web service, convert XML data to XHTML and search web documents for appropriate information.

Unit VI

PL/SQL

1. Explain Basics of PL/SQL.

Ans: PL/SQL is a development tool that supports SQL data manipulation features an also provide facilities of conditional checking, branching and looping. It allows declaration and use of variables . It also allow handle errors and display proper message according to user. It sends entire block of SQL statements to the Oracle engine. Because of this network traffic is reduces and code executes much faster in comparison of execution of a single statement. PL/SQL is a block structured language which combines features of SQL with procedural statements.

2. Define blocks and architecture of PL/SQL.

Ans: PL/SQL allow the creation of structured logical blocks of code that describe processes.

Set of SQL statements are written in a PL/SQL block. A PL/SQL block has following Sections :

- Declare Section : This is an optional section . Used to declare variables, exceptions, constants, cursors etc.
- Begin Section : It is a compulsory section. All executable code is written in this section. It contains data manipulation commands, loops, branching constructs etc.
- Exception Section : It is also an optional section. Used to handle all exceptions arise at run time during execution of a program.
- End Section : This section marks the end of PL/SQL block. It is also an compulsory part.

Structure of block :

```
[Declare]
          ---
          ---
<Begin>
          ---
          ---
[Exception]
          ---
          ---
<End>;
```

3. Explain data types in PL/SQL .

Ans: PL/SQL uses mostly data types of SQL. The data types which are available in PL/SQL are NUMBER for numeric values, CHAR & VARCHAR2 both for text values, DATE for date values, BOOLEAN for true, false. %TYPE is used to declare variables based on definitions of columns in a table. %ROWTYPE is used to declare variables which will store values of entire row of a table. Raw types are used to store binary data. It is used to store fixed length binary data. Rowid data type is same as ROWID pseudo-column type. LOB types are used to store large objects. Types of LOB are BLOB, CLOB and BFILE .

4. Explain sequential and control statements .

Ans: Control structures are the most important PL/SQL extension to SQL. Not only does PL/SQL let you manipulate Oracle data, it lets you process the data using conditional, iterative, and sequential flow-of-control statements such like:

- IF-THEN-ELSE

- CASE
- FOR-LOOP
- WHILE-LOOP
- EXIT-WHEN

5. Define cursors in PL/SQL.

Ans: Cursors are the pointers to the memory area. Cursor point to the result set of the query. Used for Row Level Processing of record returned by the query. Oracle uses work areas to execute SQL statements and store processing information. A PL/SQL construct called a cursor lets you name a work area and access its stored information. There are two kinds of cursors: implicit and explicit.

Implicit cursor:

- They are declared implicitly by PL/SQL block, whenever any DML command is included within the block.
- One implicit cursor associated with one DML statement.
- The implicit cursor works for the following statement:
    - SELECT, INSERT, UPDATE, DELETE
- There will be no users interference for implicit cursors.
- That means implicit cursors are declared & manipulated automatically by oracle server.
- A user can only trace the cursor activity like:
    - How many records were affected by cursor?
    - Cursor is affecting a row or not?
    - Whether the cursor is open in memory or not?
- Four cursor attributes are used to trace cursor activities:
    - SQL%FOUND
    - SQL%NOTFOUND
    - SQL%ISOPEN
    - SQL%ROWCOUNT
- All cursor attribute prefixed by SQL.
- Oracle server names them automatically.

Explicit Cursor:

- Explicit cursors are declared & manipulated by users only.
- The set of rows returned by a multi-row query is called the result set.
- Its size is the number of rows that meet your search criteria.
- This allows your program to process the rows one at a time.
- Explicit cursor stages:
  - Declaration of the cursor
  - Execution of the query
  - Fetching
  - Closing

6. Explain Exception handling in PL/SQL.

Ans: Exception is a specific condition in your program which may or may not cause the unexpected result. It is possible to handled the exception.In oracle we have 2 types of exceptions:

- Inbuilt
- User defined.

Inbuild exceptions: Inbuild exceptions are those which re defined and thrown by the oracle
itself. There are following inbuilt exceptions in oracle :

- NO_DATA_FOUND
- TOO_MANY_ROWS
- INVALID_CURSOR
- CURSOR_ALREADY_OPEN
- ZERO_DIVIDE

The above mentioned errors are thrown by oracle and can be caught by the programmer to avoid termination of program or to display user defined error message.

User Defined Exceptions: The user defined exceptions are raised to throw an exception which is not identified by or mentioned in the inbuild exception list. In order to built an raise a user defined exceptions use following steps:

- Declaration of exception
- Raising the UDE

- Handling the exception

7. Explain Procedures in PL/SQL.

Ans: Procedures are used to bundle set of statements which are to be execute together. Procedures can contain DML statements. They can be described (DESC) to view the arguments and other details.An existing procedure's definition can be changed or altered.

Syntax:

```
Create [or Replace] procedure <procedure name>(parameter list)
is/as
<Local declaration>
Begin
        ------
        ------   statements
Exception
        ------
        ------   statements
End [procedure name];
```

- Or replace option is used to replace the existing definition of the procedure.
- Procedure name must be unique for the schema.
- Parameter list is optional that means a procedure can be created without parameters.
- Syntax to define parameters:
- Parameter name   mode   datatype
- Parameter name must not be same as of column name.
- Datatype size must not be specified.
- We can use %type and %rowtype to declare datatypes.
- There are 2 types of parameters:
  - Formal parameters:- These parameters are used in the sub program's specification and body
  - Actual Parameter:- These parameters are passed at the time of calling of the sub program.

- Mode: Parameters can be defined in one of the 3 modes:
  - IN mode
  - OUT mode
  - IN OUT mode

Execute statement is used to call a procedure.

Syntax:

Execute procedurename (parameters list).

8. Define Packages in PL/SQL.

Ans: Packages are bundles of the program units. Packages are used to group together procedures and functions that belongs to the same category. Used to reduce I/O traffic. Packages can include procedures and functions. Individual management of sub programs is not required because the package manages them. Granting of privilege is done only on the package (all procedures and functions will be automatically accessed by the grantee).We cannot add any existing procedure or function to a package. Over loading of sub programs is possible in packages. The packages is permanently stored in the database. Sub programs can be called by any user. Packages has two part definition:

- Package Specification.
- Package Body.

Both parts require a separate create statement. Specification must be created before its body.

Specification should be successfully compiled before compiling package body.

Package Specification : Packages specification includes:

- Variables (to be used for all sub programs within package)
- Exception
- Procedure definition
- Function definition
- Constants

Variables, exceptions etc. declared in the package specification can be used within any sub program without re declaring them.

Package Body: A package body is consist of the coding for a procedure and the functions that exist in the package.

9. Explain Triggers.

Ans: Triggers are the program units that get fired automatically on some database event. Triggers are the event activated program units. Triggers are available for all the DML events. Unlike sub programs, triggers are fired automatically. Owner of the table can write triggers on the objects owned by him. All DML commands are allowed in triggers.DML triggers can be written on tables and views.TCL commands are not allowed in triggers.

Some checks are possible with only triggers like :

- Time control (logon/ logoff)
- Tracing user activity.
- Multiple table inserts.
- Complex view updation.

Trigger syntax has 3 sections.

- **Trigger Specification** : This part includes-
  - Name of the Trigger.
  - Timing (Before, After, Instead of)
  - Event (Insert, update, delete)
  - Object name (tables, views).

- **Trigger Restriction**: This part includes –
  - WHEN <condition>: to restrict the trigger to get fired on specific condition. It checks the criteria in the specification part only that means there is not need to scan the body of the trigger.
  - FOR EACH ROW: It will determine that the trigger will be fired for each row affected by the trigger. If FOR EACH ROW clause is omitted then the trigger will be fired once for the entire statement.

    Both clauses are optional

- **Trigger Body :** This part includes –

- It includes all executable statements to be fired when the trigger gets executed.
- To declare variables, constants, cursors and exceptions use the keyword DECLARE.
- BEGIN and END parts includes the references clause.
- Reference clause : There are 2 keywords which are used to point the existing and new records.

:OLD. :	Points existing records Useful for Delete and Update
:NEW :	Points new records Useful for Insert and Update

- Types of Triggers are :
  - Timing Based Triggers
    - Before Level Trigger
    - After Level Trigger
  - Row Level & Statement Level Trigger
    - Row Level Trigger
    - Statement Level Trigger
  - Instead Of Trigger

In triggers we cannot issue commit or rollback statements. A trigger may be used to provide referential integrity, to enforce complex business rules , to audit changes on data.

Multiple Choice Questions

(1) If you were collecting and storing information about your music collection, an album would be considered a

a) Relation
b) Entity
c) Instance
d) Attribute

Ans: B

(2) If you were to assign a unique number or code to each album you owned in your record collection, this code would be considered a

a) Entity
b) Instance
c) Associate
d) Primary key

Ans : D

(3) Which of the following contains information about the structure of a database ?

a) Database Management System
b) Data Dictionary
c) Data Repository
d) Data Warehouse

Ans: B

(4) To store your picture in a database requires a __________ field.

a) BLOB
b) TEXT
c) IMAGE
d) LOGICAL

Ans: A

(5) A DBMS that runs on many different operating systems is said to be __________

a) Sclable

b) Robust

c) Platform independent

d) Distributed

Ans: C

(6) Some of the columns of a relation are at different sites is which of the following ?

a) Data Replication

b) Horizontal Partitioning

c) Vertical Partitioning

d) Horizontal and Vertical Partitioning

Ans : C

(7) What is right about Distributed Database?

a) Data is stored on a single disk on a single location.

b) Data is stored on multiple disks on multiple locations.

c) Data is stored on all clients .

d) Data is stored on both servers and clients.

Ans : B

(8) A homogeneous distributed database is which of the following.

a) The same DBMS is used at each location and data are not distributed across all nodes.

b) The same DBMS is used at each location and data are distributed across all nodes.

c) A different DBMS is used at each location and data are not distributed across all nodes.

d) A different DBMS is used at each location and data are distributed across all nodes.

Ans: B

(9) A heterogeneous distributed database is which of the following.

a) The same DBMS is used at each location and data are not distributed across all nodes.

b) The same DBMS is used at each location and data are distributed across all nodes.

c) A different DBMS is used at each location and data are not distributed across all nodes.

d) A different DBMS is used at each location and data are distributed across all nodes.

Ans: D

(10) What is correct about Object Oriented database ?

a) Data is stored in form of objects.

b) Data is stored in form of tables.

c) Data is stored in form of objects and tables.

d) Data is stored on both client and server.

Ans: A.

(11) What is correct about Inheritance ?

a) Inheritance creates composite relationships between different objects.

b) Inheritance used tables to store data.

c) Inheritance gives the hierarchical relationships between different objects.

d) In Inheritance there is no concept of relationship

Ans: C

(12) Which term is used with Distributed database storage :

a) Replication

b) Fragmentation

c) Replication and Fragmentation

d) None of these

Ans: C

(13) What is correct about timestamping?

a) Each transaction has given a unique timestamp.
b) Each transaction has given a common timestamp.
c) Some transactions can use timestamp, not all.
d) Transactions are capable to generate timestamp itself.

Ans: A

(14) What is ACID ?

a) Atomicity , Consistency, Integrity, Durability
b) Availability , Consistency, Isolation, Durability
c) Availability, Concurrency, Integrity, Durability
d) Atomicity, Consistency, Isolation, Durability

Ans: D

(15) ___________ is a query and rule language used with deductive databases and syntactically is a division (subset) of Prolog.

a) Analog
b) Datalog
c) SQL Log
d) Archive Log

Ans: B

(16) What is XML ?

a) eXtensible Markup Language
b) Xtensible Markup Language
c) aXtensible Markup Language
d) eXtensible Mark Language

Ans: A

(17) What is XQuery?

a) XQyery is only a query.
b) XQuery is a programming language.
c) XQuery is a Query and functional programming language

d) XQuery is a not used with XML.

Ans: C

(18) Which data type is not used in PL/SQL ?

a) Varchar2
b) Number
c) Int
d) Char

Ans: C

(19) Which attribute is not used with Cursors?

a) SQL%NOTFOUND
b) SQL%FOUND
c) SQL%ISOPEN
d) SQL%ISCLOSE

Ans: D

(20) Which is not a system defined exception ?

a) TOO_MANY-ROWS
b) NO_DATA_FOUND
c) NO_ROWS_SELECTED
d) INVALID_CURSOR

Ans: C

(21) What is true about procedures ?

a) We cannot replace an existing procedure.
b) We cannot used exception handling part in procedures.
c) We can call procedure with SQL statement.
d) We can use %TYPE & %ROWTYPE with procedures.

Ans: D

(22) A package can include :

a) Procedures
b) Functions.
c) Exceptions
d) All of these

Ans: D

(23) Triggers are used to :
a) Apply restrictions on a table
b) Drop user of a database.
c) Give rights to users
d) Take rights back from users.
Ans: A

(24) Which operation is not possible with triggers ?
a) Tracing of user activities.
b) Creation of a new user
c) Time control
d) Multiple table insert
Ans: B

(25) Basically types of triggers are :
a) One
b) Two
c) Three
d) Four
Ans: C

(26) Which term is not related to triggers ?
a) Specification
b) Restriction
c) Isolation
d) References Clause
Ans: C

(27) Which statement is not correct about triggers ?
a) User can execute create by EXECUTE statement
b) Triggers are event driven programs.
c) FOR EACH ROW is optional for triggers.
d) We can modify an existing trigger.
Ans: A

(28) Which is not a type of trigger ?
- a) Statement Level Trigger
- b) Object Level Trigger
- c) Instead of Trigger
- d) Row Level Trigger

Ans: B

(29) Which Loop is not available in PL/SQL
- a) LOOP
- b) DO WHILE LOOP
- c) WHILE LOOP
- d) FOR LOOP

Ans: B

(30) Bind variable is not used with which mode in procedures ?
- a) IN
- b) OUT
- c) IN OUT
- d) Used with all of these.

Ans: A

Key words

ODBMS	:	Object Database Model System
OODM	:	Object Oriented Data Model
OOL	:	Object Oriented Languages

- **Inheritance** : Inheritance gives the hierarchical relationships between different objects at different levels and gives code reusability.
- **Distributed Database**: A distributed database system consists of loosely coupled sites that share no physical components.
- **Replication :** In this approach, the system maintains several identical copies of the relation. Each relation is stored at different site.
- **Fragmentation :** In this approach, the relation is partitioned into several fragments. Each fragment is stored at a different site.
- **Timestamping :** Each transaction is given a unique timestamp that the system uses in deciding the serialization order.
- **Parallel Database Systems**: Parallel Systems are designed to improve processing and I/O speeds by using multiple CPUs and disk in parallel.
- **Shared Memory :** All the processors share a common memory in this model.
- **Shared Disk:**All the processors share a common disk.
- **Shared Nothing :** The processors share neither a common memory nor common disk.
- **Interquery Parallelism**: Different queries or transactions execute equivalent (parallel) with one another in this parallelism.
- **Intraquery Parallelism**: A single query executes on multiple processors and disks in Parallel
- **Range- Partition Sort :** In this type of sorting it is not compulsory to range the relation on the same processors or disks as those on which that relation is stored.
- **Parallel Join :** Pairs of tuples are tested to check that they satisfy the join condition or not. If they satisfy the condition, then the pair is added to the join output.
- **Pipelined Parallelism:** Main advantage of this execution in a sequential

evaluation is that we can achieve a sequence of such operations without writing any of the intermediate results to disk.

- **Independent Parallelism**: Operations in a query expression do not depend on one another. It can be executed parallel.
- **Define Deductive Database**: Deductive databases are an extension of relational databases which support more complex data modeling.
- **Datalog:** Datalog is a query and rule language used with deductive database and syntactically is a division (subset) of Prolog.
- **XML :** eXtensible Markup Language
- **DTD:** Document Type Definition
- **Web Search Engine**:Used to search information on the World Wide Web
- **XQuery** : It is a query and functional programming language which is used to query on collection of XML data.
- **PL/SQL** : Procedural Language / Structured Query Language.
- **BLOB:** Binary Large Object
- **CLOB :** Character Large Object
- **Implicit Cursor :** These are created by default when INSERT, UPDATE, DELETE, SELECT statements are executed.
- **Explicit Cursor :** Created by user. Functionality is same as Implicit Cursors. Inbuilt exceptions: Defined and thrown by the oracle itself.
- **User Defined Exceptions**: Explicitly defined and based on business rules
- **Formal parameters**: These parameters are used in the sub program's specification and body
- **Actual Parameters** :These parameters are passed at the time of calling of the sub program.

- **Packages** : Packages are used to group together procedures and functions.
- **Triggers** :Triggers are the program units that get fired automatically on some database event.
- **Procedures :** A procedure is a named PL/SQL block which performs one or more specific tasks.

ASSIGNMENTS – PL/SQL:

Exercise – I (PL/SQL Block)

Q1. Create a program which will accept empno from user and display his name, job and salary.
Q2. Create a program which will accept empno, ename, job, sal from user and insert this new record in emp table.
Q3. Create a program which will accept empno from user and delete record of this employee.
Q4. Create a program which will accept empno from user and increase his salary by 2000. Display ename, job, old and new salary.

Exercise –I I (IF-- Else)

Q1. Create a program which will accept empno from user and check his salary. If salary is more than 2000, then increase it by 3000.
Q2. Create a program which will accept empno from user and check his deptno. If deptno is 10 then decrease salary by 1000, if deptno is 20 then decrease by 2000 and if deptno = 30 then decrease by 3000.
Q3. Create a program which will accept ename from user and modify his record according to following details:
If working as Clerk in deptno = 10 then change job to LDC.
If working as Manager in deptno = 20 then change job to EXECUTIVE.
If working as Salesman in deptno = 30 then change job to SALESREP
Display NO UPDATION message in all other cases.

Exercise – III (Cursor)

Q1. Create a program which will accept dname from user and display ename, job, sal of those employees who are working in this department.
Q2. Create a program which will display ename, dname, grade of all employees.
Q3. Create a program which will accept month from user and display ename, job, hiredate of those employees who have joined in this month.
Q4. Create a program which will display 5 topmost earners.

Exercise – IV (Exceptions)

Q1. Create a program which will accept empno from user and display name, sal, deptno of that employee. Write code for all possible exceptions.

Q2. Create a program which will accept empno from user . Increase his salary by 2000. Also store updated values in INCREMENT table (empno, ename, sal). An employee can get increment maximum 3 times otherwise an exception will raise.

Exercise- V (Procedures & Packages)

Q1. Create a procedure which will accept name from user and display ename, job, sal of those employees who are managed by him.

Q2. Create a procedure which will accept grade from user and display ename, sal, losal, hisal of those employees who are getting salary of that grade.

Q3. Create a procedure which will display details of employees location and grade wise in following format::

```
                              ABC LTD.
**************************************************************
Dname:        Sales              Location:     New York
**************************************************************
         Ename              Designation          Salary
         *******            ***********          ******
         Allen              Manager              5000
         Ford               Salesman             3000
         Rock               Salesman             2500
**************************************************************
Dname:        Research           Location:     London
**************************************************************
         Ename              Designation          Salary
         *******            ***********          ******
         James              Manager              6000
         Scott              Analyst              5000
         Martin             Clerk                3000
```

****

Q4. Create a package which will store:

a. A procedure which will accept empno and delete that employee.

b. A procedure which will accept empno and decrease his salary by 10%.

c. A procedure which will accept empno, ename, job, salary and insert these values in emp table.

Exercise – VI (Triggers)

Q1. Create a trigger which will restrict user to perform any dml operation on emp table.

Q2. Create a trigger which will restrict user to delete records of all Clerks.

Q3. Create a trigger on emp table which will store deleted values into Backup table.(empno, ename, sal, deptno).

Q4. Create a trigger which will check that incoming salary of manager should not more than 5000.

===