UNIT 4 ORACLE

Structure		Page Nos.
4.0	Introduction	51
4.1	Objectives	52
4.2	Database Application Development Features	52
	4.2.1 Database Programming	
	4.2.2 Database Extensibility	
4.3	Database Design and Querying Tools	57
4.4	Overview of Oracle Architecture	50
	4.4.1 Physical Database Structures	
	4.4.2 Logical Database Structures	
	4.4.3 Schemas and Common Schema Objects	
	4.4.4 Oracle Data Dictionary	
	4.4.5 Oracle Instance	
	4.4.6 Oracle Background Processes	
	4.4.7 How Oracle Works?	
4.5	Query Processing and Optimisation	71
4.6	Distributed Oracle	75
	4.6.1 Distributed Queries and Transactions	
	4.6.2 Heterogenous Services	
4.7	Data Movement in Oracle	76
	4.7.1 Basic Replication	
	4.7.2 Advanced Replication	
	4.7.3 Transportable Tablespaces	
	4.7.4 Advanced Queuing and Streams	
	4.7.5 Extraction, Transformation and Loading	
4.8	Database Administration Tools	77
4.9	Backup and Recovery in Oracle	78
4.10	Oracle Lite	79
4.11	Scalability and Performance Features of Oracle	79
	4.11.1 Concurrency	
	4.11.2 Read Consistency	
	4.11.3 Locking Mechanisms	
	4.11.4 Real Application Clusters	
	4.11.5 Portability	
4.12	Oracle DataWarehousing	83
	4.12.1 Extraction, Transformation and Loading (ETL)	
	4.12.2 Materialised Views	
	4.12.3 Bitmap Indexes	
	4.12.4 Table Compression	
	4.12.5 Parallel Execution	
	4.12.6 Analytic SQL	
	4.12.7 OLAP Capabilities	
	4.12.8 Data Mining	
	4.12.9 Partitioning	
4.13	Security Features of Oracle	85
4.14	Data Integrity and Triggers in Oracle	86
4.15	Transactions in Oracle	87
4.16	SQL Variations and Extensions in Oracle	88
4.17	Summary	90
4.18	Solutions/Answers	91

4.0 INTRODUCTION

The relational database concept was first described by Dr. Edgar F. Codd in an IBM research publication titled “System R4 Relational” which in 1970. Initially, it was unclear whether any system based on this concept could achieve commercial success. However, if we look back there have been many products, which support most of the features of relational database models and much more. Oracle is one such product

that was created by Relational Software Incorporated (RSI) in 1977. They released Oracle V.2 as the world's first relational database within a couple of years. In 1983, RSI was renamed Oracle Corporation to avoid confusion with a competitor named RTI. During this time, Oracle developers made a critical decision to create a portable version of Oracle (Version 3) that could run not only on Digital VAX/VMS systems, but also on Unix and other platforms. Since the mid-1980s, the database deployment model has evolved from dedicated database application servers to client/servers to Internet computing implemented with PCs and thin clients accessing database applications via browsers – and, to the grid with Oracle Database 10g.

Oracle introduced many innovative technical features to the database as computing and deployment models changed (from offering the first distributed database to the first Java Virtual Machine in the core database engine). Oracle also continues to support emerging standards such as XML and .NET.

We will discuss some of the important characteristics and features of Oracle in this unit.

4.1 OBJECTIVES

After going through this unit, you should be able to:

- define the features of application development in Oracle;
- identify tools for database design and query;
- describe the Oracle architecture;
- discuss the query processing and optimisation in Oracle;
- define the distributed Oracle database features;
- identify tools for database administration including backup and recovery;
- discuss the features of Oracle including scalability, performance, security etc., and
- define newer applications supported by Oracle.

4.2 DATABASE APPLICATION DEVELOPMENT FEATURES

The main use of the Oracle database system is to store and retrieve data for applications. Oracle has evolved over the past 20 years. The following table traces the historical facts about Oracle.

Year	Feature
1979	Oracle Release 2—the first commercially available relational database to use SQL.
1980-1990	Single code base for Oracle across multiple platforms, Portable toolset, Client/server Oracle relational database, CASE and 4GL toolset, Oracle Financial Applications built on relational database.
1991-2000	Oracle Parallel Server on massively parallel platform, cost-based optimiser, parallel operations including query, load, and create index, Universal database with extended SQL via cartridges, thin client, and application server, Oracle 8 with inclusion of object-relational and Very Large Database (VLDB) features, Oracle8i added a new twist to the Oracle database – a combination of enhancements that made the Oracle8i database the focal point of the world of Internet (the i in 8i) computing. Java Virtual Machine (JVM) was added into the database and Oracle

Year	Feature
	tools were integrated in the middle tier.
2001	Oracle9i Database Server is generally available: Real Application Clusters; OLAP and data mining API in the database.
2003	Oracle Database 10g enables grid (the g in 10g) computing. A grid is simply a pool of computers that provides needed resources for applications on an as-needed basis. The goal is to provide computing resources that can scale transparently to the user community, much as an electrical utility company can deliver power, to meet peak demand, by accessing energy from other power providers' plants via a power grid. Oracle Database 10g further reduces the time, cost, and complexity of database management through the introduction of self-managing features such as the Automated Database Diagnostic Monitor, Automated Shared Memory Tuning, Automated Storage Management, and Automated Disk Based Backup and Recovery. One important key to Oracle Database 10g's usefulness in grid computing is the ability to have a provision for CPUs and data.

However, in this section we will concentrate on the tools that are used to create applications. We have divided the discussion in this section into two categories: database programming and database extensibility options. Later in this unit, we will describe the Oracle Developer Suite, a set of optional tools used in Oracle Database Server and Oracle Application Server development.

4.2.1 Database Programming

All flavors of the Oracle database include different languages and interfaces that allow programmers to access and manipulate data in the database. The following are the languages and interfaces supported by Oracle.

SQL

All operations on the information in an Oracle database are performed using SQL statements. A statement must be the equivalent of a complete SQL sentence, as in:

```
SELECT last_name, department_id FROM employees;
```

Only a complete SQL statement can run successfully. A SQL statement can be thought of as a very simple, but powerful, computer instruction. SQL statements of Oracle are divided into the following categories.

Data Definition Language (DDL) Statements: These statements create, alter, maintain, and drop schema objects. DDL statements also include statements that permit a user the right to grant other users the privilege of accessing the database and specific objects within the database.

Data Manipulation Language (DML) Statements: These statements manipulate data. For example, querying, inserting, updating, and deleting rows of a table are all DML operations. Locking a table and examining the execution plan of a SQL statement are also DML operations.

Transaction Control Statements: These statements manage the changes made by DML statements. They enable a user group changes into logical transactions. Examples include COMMIT, ROLLBACK, and SAVEPOINT.

Session Control Statements: These statements let a user control the properties of the current session, including enabling and disabling roles and changing language settings. The two session control statements are ALTER SESSION and SET ROLE.

System Control Statements: These statements change the properties of the Oracle database instance. The only system control statement is ALTER SYSTEM. It lets users change settings, such as the minimum number of shared servers, kill a session, and perform other tasks.

Embedded SQL Statements: These statements incorporate DDL, DML, and transaction control statements in a procedural language program. Examples include OPEN, CLOSE, FETCH, and EXECUTE.

Datatypes

Each attribute and constant in a SQL statement has a datatype, which is associated with a specific storage format, constraints, and a valid range of values. When you create a table, you must specify a datatype for each of its columns.

Oracle provides the following built-in datatypes:

- Character datatypes
- Numeric datatypes
- DATE datatype
- LOB datatypes
- RAW and LONG RAW datatypes
- ROWID and UROWID datatypes.

New object types can be created from any in-built database types or any previously created object types, object references, and collection types. Metadata for user-defined types is stored in a schema available to SQL, PL/SQL, Java, and other published interfaces.

An object type differs from native SQL datatypes in that it is user-defined, and it specifies both the underlying persistent data (attributes) and the related behaviours (methods). Object types are abstractions of real-world entities, for example, purchase orders.

Object types and related object-oriented features, such as variable-length arrays and nested tables, provide higher-level ways to organise and access data in the database. Underneath the object layer, data is still stored in columns and tables, but you can work with the data in terms of real-world entities – customers and purchase orders, that make the data meaningful. Instead of thinking in terms of columns and tables when you query the database, you can simply select a customer.

PL/SQL

PL/SQL is Oracle's procedural language extension to SQL. PL/SQL combines the ease and flexibility of SQL with the procedural functionality of a structured programming language, such as IF, THEN, WHILE, and LOOP.

When designing a database application, consider the following advantages of using stored PL/SQL:

- PL/SQL code can be stored centrally in a database. Network traffic between applications and the database is reduced, hence, both application and system performance increases. Even when PL/SQL is not stored in the database, applications can send blocks of PL/SQL to the database rather than individual SQL statements, thereby reducing network traffic.

- Data access can be controlled by stored PL/SQL code. In this case, PL/SQL users can access data only as intended by application developers, unless another access route is granted.
- PL/SQL blocks can be sent by an application to a database, running complex operations without excessive network traffic.
- Oracle supports PL/SQL Server Pages, so the application logic can be invoked directly from your Web pages.

The PL/SQL program units can be defined and stored centrally in a database. Program units are stored procedures, functions, packages, triggers, and anonymous transactions.

Procedures and functions are sets of SQL and PL/SQL statements grouped together as a unit to solve a specific problem or to perform a set of related tasks. They are created and stored in compiled form in the database and can be run by a user or a database application. Procedures and functions are identical, except that all functions always return a single value to the user. Procedures do not return values.

Packages encapsulate and store related procedures, functions, variables, and other constructs together as a unit in the database. They offer increased functionality (for example, global package variables can be declared and used by any procedure in the package). They also improve performance (for example, all objects of the package are parsed, compiled, and loaded into memory once).

Java Features and Options

Oracle8i introduced the use of Java as a procedural language with a Java Virtual Machine (JVM) in the database (originally called JServer). JVM includes support for Java stored procedures, methods, triggers, Enterprise JavaBeans (EJBs), CORBA, and HTTP. The Accelerator is used for project generation, translation, and compilation, and can also be used to deploy/install-shared libraries. The inclusion of Java within the Oracle database allows Java developers to level their skills as Oracle application developers as well. Java applications can be deployed in the client, Application Server, or database, depending on what is most appropriate. Oracle data warehousing options for OLAP and data mining provide a Java API. These applications are typically custom built using Oracle's JDeveloper.

Large Objects

Interest in the use of large objects (LOBs) continues to grow, particularly for storing non-traditional data types such as images. The Oracle database has been able to store large objects for some time. Oracle8 added the capability to store multiple LOB columns in each table. Oracle Database 10g essentially removes the space limitation on large objects.

Object-Oriented Programming

Support of object structures has been included since Oracle8i to allow an object-oriented approach to programming. For example, programmers can create user-defined data types, complete with their own methods and attributes. Oracle's object support includes a feature called Object Views through which object-oriented programs can make use of relational data already stored in the database. You can also store objects in the database as varying arrays (VARRAYs), nested tables, or index organised tables (IOTs).

Third-Generation Languages (3GLs)

Programmers can interact with the Oracle database from C, C++, Java, COBOL, or FORTRAN applications by embedding SQL in those applications. Prior to compiling applications using a platform's native compilers, you must run the embedded SQL code through a precompiler. The precompiler replaces SQL statements with library calls the native compiler can accept. Oracle provides support for this capability through optional "programmer" precompilers for languages such as C and C++ (Pro*C) and COBOL (Pro*COBOL). More recently, Oracle added SQLJ, a precompiler for Java that replaces SQL statements embedded in Java with calls to a SQLJ runtime library, also written in Java.

Database Drivers

All versions of Oracle include database drivers that allow applications to access Oracle via ODBC (the Open DataBase Connectivity standard) or JDBC (the Java DataBase Connectivity open standard). Also available are data providers for OLE DB and for .NET.

The Oracle Call Interface

This interface is available for the experienced programmer seeking optimum performance. They may choose to define SQL statements within host-language character strings and then explicitly parse the statements, bind variables for them, and execute them using the Oracle Call Interface (OCI). OCI is a much more detailed interface that requires more programmer time and effort to create and debug. Developing an application that uses OCI can be time-consuming, but the added functionality and incremental performance gains often make spending the extra time worthwhile.

National Language Support

National Language Support (NLS) provides character sets and associated functionality, such as date and numeric formats, for a variety of languages. Oracle9i featured full Unicode 3.0 support. All data may be stored as Unicode, or select columns may be incrementally stored as Unicode. UTF-8 encoding and UTF-16 encoding provides support for more than 57 languages and 200 character sets. Oracle Database 10g adds support for Unicode 3.2. Extensive localisation is provided (for example, for data formats) and customised localisation can be added through the Oracle Locale Builder. Oracle Database 10g includes a Globalisation Toolkit for creating applications that will be used in multiple languages.

4.2.2 Database Extensibility

The Internet and corporate intranets have created a growing demand for storage and manipulation of non-traditional data types within the database. There is a need for its extension to the standard functionality of a database for storing and manipulating image, audio, video, spatial, and time series information. These capabilities are enabled through extensions to standard SQL.

Oracle Text and InterMedia

Oracle Text can identify the gist of a document by searching for themes and key phrases in the document.

Oracle interMedia bundles additional image, audio, video, and locator functions and is included in the database license. Oracle interMedia offer the following capabilities:

- The image portion of interMedia can store and retrieve images.
- The audio and video portions of interMedia can store and retrieve audio and video clips, respectively.
- The locator portion of interMedia can retrieve data that includes spatial coordinate information.

Oracle Spatial Option

The Spatial option is available for Oracle Enterprise Edition. It can optimise the display and retrieval of data linked to coordinates and is used in the development of spatial information systems. Several vendors of Geographic Information Systems (GIS) products now bundle this option and leverage it as their search and retrieval engine.

XML

Oracle added native XML data type support to the Oracle9i database and XML and SQL interchangeability for searching. The structured XML object is held natively in object relational storage meeting the W3C DOM specification. The XPath syntax for searching in SQL is based on the SQLX group specifications.

4.3 DATABASE DESIGN AND QUERYING TOOLS

Many Oracle tools are available to developers to help them present data and build more sophisticated Oracle database applications. This section briefly describes the main Oracle tools for application development: Oracle Forms Developer, Oracle Reports Developer, Oracle Designer, Oracle JDeveloper, Oracle Discoverer Administrative Edition and Oracle Portal. Oracle Developer Suite was known as Oracle Internet Developer Suite with Oracle9i.

SQL*Plus

SQL*Plus is an interactive and batch query tool that is installed with every Oracle Database Server or Client installation. It has a command-line user interface, a Windows Graphical User Interface (GUI) and the *i*SQL*Plus web-based user interface.

SQL*Plus has its own commands and environment, and it provides access to the Oracle Database. It enables you to enter and execute SQL, PL/SQL, SQL*Plus and operating system commands to perform the following:

- format, perform calculations on, store, and print from query results,
- examine table and object definitions,
- develop and run batch scripts, and
- perform database administration.

You can use SQL*Plus to generate reports interactively, to generate reports as batch processes, and to output the results to text file, to screen, or to HTML file for browsing on the Internet. You can generate reports dynamically using the HTML output facility of SQL*Plus, or using the dynamic reporting capability of *i*SQL*Plus to run a script from a web page.

Oracle Forms Developer

Oracle Forms Developer provides a powerful tool for building forms-based applications and charts for deployment as traditional client/server applications or as three-tier browser-based applications via Oracle Application Server. Developer is a fourth-generation language (4GL). With a 4GL, you define applications by defining values for properties, rather than by writing procedural code. Developer supports a wide variety of clients, including traditional client/server PCs and Java-based clients. The Forms Builder includes an in-built JVM for previewing web applications.

Oracle Reports Developer

Oracle Reports Developer provides a development and deployment environment for rapidly building and publishing web-based reports via Reports for Oracle's Application Server. Data can be formatted in tables, matrices, group reports, graphs, and combinations. High-quality presentation is possible using the HTML extension Cascading Style Sheets (CSS).

Oracle JDeveloper

Oracle JDeveloper was introduced by Oracle in 1998 to develop basic Java applications without writing code. JDeveloper includes a Data Form wizard, a Beans Express wizard for creating JavaBeans and BeanInfo classes, and a Deployment wizard. JDeveloper includes database development features such as various Oracle drivers, a Connection Editor to hide the JDBC API complexity, database components to bind visual controls, and a SQLJ precompiler for embedding SQL in Java code, which you can then use with Oracle. You can also deploy applications developed with JDeveloper using the Oracle Application Server. Although JDeveloper user wizards to allow programmers create Java objects without writing code, the end result is a generated Java code. This Java implementation makes the code highly flexible, but it is typically a less productive development environment than a true 4GL.

Oracle Designer

Oracle Designer provides a graphical interface for Rapid Application Development (RAD) for the entire database development process—from building the business model to schema design, generation, and deployment. Designs and changes are stored in a multi-user repository. The tool can reverse-engineer existing tables and database schemas for re-use and re-design from Oracle and non-Oracle relational databases.

The designer also include generators for creating applications for Oracle Developer, HTML clients using Oracle's Application Server, and C++. Designer can generate applications and reverse-engineer existing applications or applications that have been modified by developers. This capability enables a process called round-trip engineering, in which a developer uses Designer to generate an application, modifies the generated application, and reverse-engineers the changes back into the Designer repository.

Oracle Discoverer

Oracle Discoverer Administration Edition enables administrators to set up and maintain the Discoverer End User Layer (EUL). The purpose of this layer is to shield business analysts using Discoverer as an ad hoc query or ROLAP tool from SQL complexity. Wizards guide the administrator through the process of building the EUL. In addition, administrators can put limits on resources available to analysts monitored by the Discoverer query governor.

Oracle Portal

Oracle Portal, introduced as WebDB in 1999, provides an HTML-based tool for developing web-enabled applications and content-driven web sites. Portal application systems are developed and deployed in a simple browser environment. Portal includes wizards for developing application components incorporating "servlets" and access to other HTTP web sites. For example, Oracle Reports and Discoverer may be accessed as servlets. Portals can be designed to be user-customisable. They are deployed to the middle-tier Oracle Application Server.

Oracle Portal has enhanced the WebDB, with the ability to create and use portlets, which allow a single web page to be divided into different areas that can independently display information and interact with the user.

☞ Check Your Progress 1

1) What are Data Definition statements in Oracle?

……

……

……

……

……

2) …… manage the changes made by DML statements.

3) What are the in- built data types of ORACLE?

……

……

……

……

……

4) Name the main ORACLE tools for Application Development.

……

……

……

……

……

4.4 OVERVIEW OF ORACLE ARCHITECTURE

The following section presents the basic architecture of the Oracle database. A schematic diagram of Oracle database is given below:

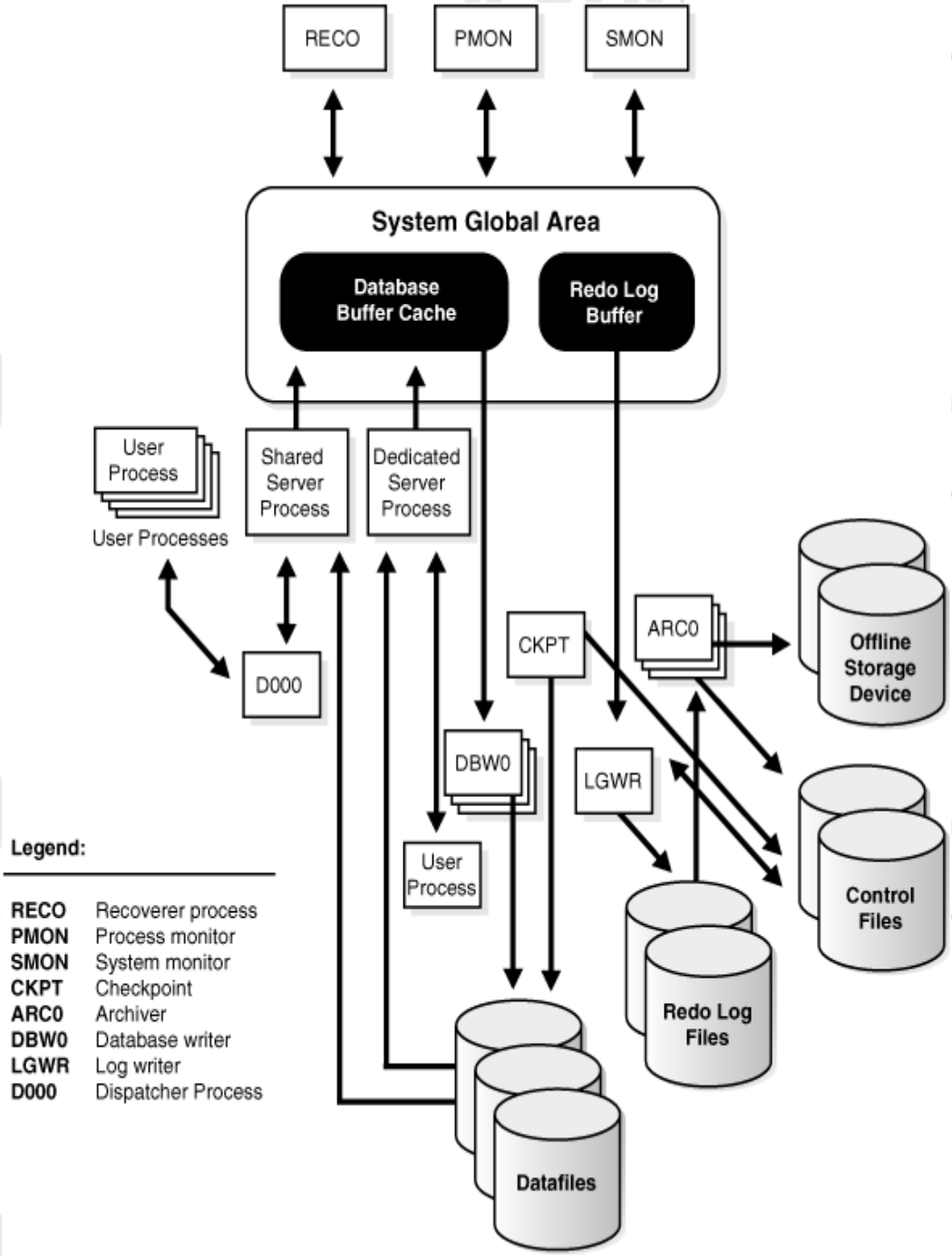

Figure 1: Oracle 10g Architecture (Source: www.oracle.com)

4.4.1 Physical Database Structures

The following sections explain the physical database structures of an Oracle database, including datafiles, redo log files, and control files.

Datafiles

Every Oracle database has one or more physical datafiles. The datafiles contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database.
The characteristics of datafiles are:

- a datafile can be associated with only one database,

- datafiles can have certain characteristics set to let them automatically extend when the database runs out of space,
- one or more datafiles form a logical unit of database storage called a tablespace.

Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory cache for the database, then it is read from the appropriate datafiles and stored in the memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the database writer process (DBW*n*) background process.

Control Files

Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following information:

- database name,
- names and locations of datafiles and redo log files, and
- time stamp of database creation.

Oracle can multiplex the control file, that is, simultaneously maintain a number of identical control file copies, to protect against failure involving the control file.

Every time an instance of an Oracle database is started, its control file identifies the database and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, if a new datafile or redo log file is created), then the control file is automatically modified by Oracle to reflect the change. A control file is also used in database recovery.

Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files is collectively known as the redo log for the database. A redo log is made up of redo entries (also called redo records).

The primary function of the redo log is to record all changes made to the data. If a failure prevents modified data from being permanently written to the datafiles, then the changes can be obtained from the redo log, so work is never lost. To protect against a failure involving the redo log itself, Oracle allows a multiplexed redo log so that two or more copies of the redo log can be maintained on different disks.

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to the datafiles. For example, if an unexpected power shortage terminates database operation, then the data in the memory cannot be written to the datafiles, and the data is lost. However, lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database datafiles, Oracle restores the database to the time when the power failure had occurred. The process of applying the redo log during a recovery operation is called rolling forward.

Archive Log Files

You can also enable automatic archiving of the redo log. Oracle automatically archives log files when the database is in ARCHIVELOG mode.

Parameter Files

Parameter files contain a list of configuration parameters for that instance and database.

Oracle recommends that you create a server parameter file (SPFILE) as a dynamic means of maintaining initialisation parameters. A server parameter file lets you store and manage your initialisation parameters persistently in a server-side disk file.

Alert and Trace Log Files

Each server and background process can write to an associated trace file. When an internal error is detected by a process, it dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, while other information is for Oracle Support Services. Trace file information is also used to tune applications and instances.

The alert file, or alert log, is a special trace file. The alert file of a database is a chronological log of messages and errors.

Backup Files

To restore a file is to replace it with a backup file. Typically, you restore a file when a media failure or user error has damaged or deleted the original file.

User-managed backup and recovery requires you to actually restore backup files before a trial recovery of the backups can be attempted/performed.

Server-managed backup and recovery manages the backup process, such as scheduling of backups, as well as recovery processes such as applying the correct backup file when recovery is needed.

4.4.2 Logical Database Structures

The logical storage structures, including data blocks, extents, and segments, enable Oracles fine-grained control of disk space use.

Tablespaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group together all application objects to simplify administrative operations.

Each database is logically divided into one or more tablespaces. One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. The combined size of the datafiles in a tablespace is the total storage capacity of the tablespace. Every Oracle database contains a SYSTEM tablespace and a SYSAUX tablespace. Oracle creates them automatically when the database is created. The system default is to create a smallfile tablespace, which is the traditional type of Oracle tablespace. The SYSTEM and SYSAUX tablespaces are created as smallfile tablespaces.

Oracle also lets you create bigfile tablespaces up to 8 exabytes (8 million terabytes) in size. With Oracle-managed files, bigfile tablespaces make datafiles completely transparent for users. In other words, you can perform operations on tablespaces, rather than on the underlying datafiles.

Online and Offline Tablespaces

A tablespace can be online (accessible) or offline (not accessible). A tablespace is generally online, so that users can access information in the tablespace. However,

sometimes a tablespace is offline, in order to make a portion of the database unavailable while allowing normal access to the remainder of the database. This makes many administrative tasks easier to perform.

Oracle Data Blocks

At the finest level of granularity, Oracle database data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. The standard block size is specified by the DB_BLOCK_SIZE initialization parameter. In addition, you can specify up to five other block sizes. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

Segments

Next, is the segment, or the level of logical database storage. A segment is a set of extents allocated for a certain logical structure. The following table describes the different types of segments.

Segment	Description
Data segment	Each non-clustered table has a data segment. All table data is stored in the extents of the data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
Index segment	Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
Temporary segment	Temporary segments are created by Oracle when a SQL statement needs a temporary database area to complete execution. When the statement has been executed, the extents in the temporary segment are returned to the system for future use.
Rollback segment	If you are operating in automatic undo management mode, then the database server manages undo space-using tablespaces. Oracle recommends that you use automatic undo management. Earlier releases of Oracle used rollback segments to store undo information. The information in a rollback segment was used during database recovery for generating read-consistent database information and for rolling back uncommitted transactions for users. Space management for these rollback segments was complex, and not done that way. Oracle uses the undo tablespace method of managing undo; this eliminates the complexities of managing rollback segment space. Oracle does use a SYSTEM rollback segment for performing system

Segment	Description
	transactions. There is only one SYSTEM rollback segment and it is created automatically at CREATE DATABASE time and is always brought online at instance startup. You are not required to perform any operation to manage the SYSTEM rollback segment.

Oracle dynamically allocates space when the existing extents of a segment become full. In other words, when the extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on a disk.

4.4.3 Schemas and Common Schema Objects

A schema is a collection of database objects. A schema is owned by a database user and has the same name as that user. Schema objects are the logical structures that refer directly to the database's data. Schema objects include structures like tables, views, and indexes. (There is no relationship between a tablespace and a schema. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas). Some of the most common schema objects are defined in the following section.

Tables

Tables are the basic unit of data storage in an Oracle database. Database tables hold all user-accessible data. Each table has columns and rows.

Indexes

Indexes are optional structures associated with tables. Indexes can be created to increase the performance of data retrieval. An index provides an access path to table data.

When processing a request, Oracle may use some or all of the available indexes in order to locate, the requested rows efficiently. Indexes are useful when applications frequently query a table for a range of rows (for example, all employees with salaries greater than 1000 dollars) or a specific row.

An index is automatically maintained and used by the DBMS. Changes to table data (such as adding new rows, updating rows, or deleting rows) are automatically incorporated into all relevant indexes with complete transparency to the users. Oracle uses B-trees to store indexes in order to speed up data access.

An index in Oracle can be considered as an ordered list of the values divided into block-wide ranges (leaf blocks). The end points of the ranges along with pointers to the blocks can be stored in a search tree and a value in log(n) time for n entries could be found. This is the basic principle behind Oracle indexes.
The following *Figure 2* illustrates the structure of a B-tree index.

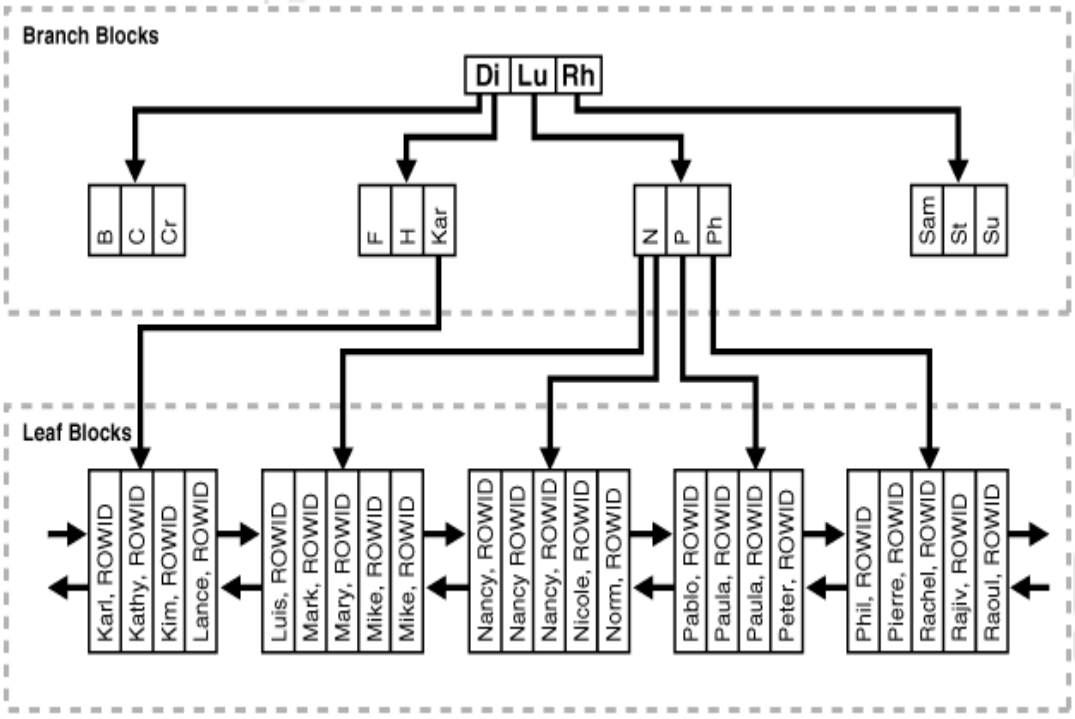

Figure 2: Internal Structure of an Oracle Index (Source: www.oracle.com)

The upper blocks (branch blocks) of a B-tree index contain index data that points to lower-level index blocks. The lowest level index blocks (leaf blocks) contains every indexed data value and a corresponding rowid used to locate the actual row. The leaf blocks are doubly linked. Indexes in columns containing character data are based on the binary values of the characters in the database character set.

Index contains two kinds of blocks:

- Branch blocks for searching, and
- Leaf blocks that store values.

Branch Blocks: Branch blocks store the following:

- the minimum key prefix needed to make a branching decision between two keys, and
- the pointer to the child block containing the key.

If the blocks have *n* keys then they have *n*+1 pointers. The number of keys and pointers is limited by the block size.

Leaf Blocks: All leaf blocks are at the same depth from the root branch block. Leaf blocks store the following:

- the complete key value for every row, and
- ROWIDs of the table rows

All key and ROWID pairs are linked to their left and right siblings. They are sorted by (key, ROWID).

Views

Views are customised presentations of data in one or more tables or other views. A view can also be considered as a stored query. Views do not actually contain data. Rather, they derive their data from the tables on which they are based, (referred to as the base tables of the views). Views can be queried, updated, inserted into, and deleted from, with some restrictions.

Views provide an additional level of table security by restricting access to a pre-determined set of rows and columns of a table. They also hide data complexity and store complex queries.

Clusters

Clusters are groups of one or more tables physically stored together because they share common columns and are often used together. Since related rows are physically stored together, disk access time improves.

Like indexes, clusters do not affect application design. Whether a table is part of a cluster is transparent to users and to applications. Data stored in a clustered table is accessed by SQL in the same way as data stored in a non-clustered table.

Synonyms

A synonym is an alias for any table, view, materialised view, sequence, procedure, function, package, type, Java class schema object, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary.

4.4.4 Oracle Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference on the database. For example, a data dictionary stores information about the logical and physical structure of the database. A data dictionary also stores the following information:

- the valid users of an Oracle database,
- information about integrity constraints defined for tables in the database, and
- the amount of space allocated for a schema object and how much of it is in use.

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific actions, (such as, when the structure of the database is altered). The database relies on the data dictionary to record, verify, and conduct on-going work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exist and that users have proper access to them.

4.4.5 Oracle Instance

An Oracle database server consists of an Oracle database and an Oracle instance. Every time a database is started, a system global area (SGA) (please refer to *Figure1)* is allocated and Oracle background processes are started. The combination of the background processes and memory buffers is known as an Oracle instance.

Real Application Clusters: Multiple Instance Systems

Some hardware architectures (for example, shared disk systems) enable multiple computers to share access to data, software, or peripheral devices. Real Application Clusters (RAC) take advantage of such architecture by running multiple instances that share a single physical database. In most applications, RAC enables access to a single database by users on multiple machines with increased performance.

An Oracle database server uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of computers that constitute the database system. Processes are jobs that work in the memory of these computers.

Instance Memory Structures

Oracle creates and uses memory structures to complete several jobs. For example, memory stores program code being run and data shared among users. Two basic memory structures associated with Oracle are: the system global area and the program global area. The following subsections explain each in detail.

System Global Area

The System Global Area (SGA) is a shared memory region that contains data and control information for one Oracle instance. Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. Each instance has its own SGA.

Users currently connected to an Oracle database share the data in the SGA. For optimal performance, the entire SGA should be as large as possible (while still fitting in to the real memory) to store as much data in memory as possible and to minimise disk I/O.

The information stored in the SGA is divided into several types of memory structures, including the database buffers, redo log buffer, and the shared pool.

Database Buffer Cache of the SGA

Database buffers store the most recently used blocks of data. The set of database buffers in an instance is the database buffer cache. The buffer cache contains modified as well as unmodified blocks. Because the most recently (and often, the most frequently) used data is kept in the memory, less disk I/O is required, and the performance is improved.

Redo Log Buffer of the SGA

The redo log buffer stores redo entries – a log of changes made to the database. The redo entries stored in the redo log buffers are written to an online redo log, which is used if database recovery is necessary. The size of the redo log is static.

Shared Pool of the SGA

The shared pool contains shared memory constructs, such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.

Statement Handles or Cursors

A cursor is a handle or name for a private SQL area in which a parsed statement and other information for processing the statement are kept. (Oracle Call Interface, OCI, refers to these as statement handles). Although most Oracle users rely on automatic cursor handling of Oracle utilities, the programmatic interfaces offer application designers more control over cursors.

For example, in precompiler application development, a cursor is a named resource available to a program and can be used specifically to parse SQL statements embedded within the application. Application developers can code an application so that it controls the phases of SQL statement execution and thus improves application performance.

Program Global Area

The Program Global Area (PGA) is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the configuration of Oracle.

4.4.6 Oracle Background Processes

An Oracle database uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of computers that constitute the database system. Processes are jobs that work in the memory of these computers.

The architectural features discussed in this section enables the Oracle database to support:

- many users concurrently accessing a single database, and
- the high performance required by concurrent multiuser, multiapplication database systems.

Oracle creates a set of background processes for each instance. The background processes consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. They asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability. There are numerous background processes, and each Oracle instance can use several background processes.

Process Architecture

A process is a "thread of control" or a mechanism in an operating system that can run a series of steps. Some operating systems use the terms job or task. A process generally has its own private memory area in which it runs. An Oracle database server has two general types of processes: user processes and Oracle processes.

User (Client) Processes

User processes are created and maintained to run the software code of an application program or an Oracle tool (such as Enterprise Manager). User processes also manage communication with the server process through the program interface, which is described in a later section.

Oracle Processes

Oracle processes are invoked by other processes to perform functions on behalf of the invoking process. Oracle creates server processes to handle requests from connected user processes. A server process communicates with the user process and interacts with Oracle to carry out requests from the associated user process. For example, if a user queries some data not already in the database buffers of the SGA, then the associated server process reads the proper data blocks from the datafiles into the SGA.

Oracle can be configured to vary the number of user processes for each server process. In a dedicated server configuration, a server process handles requests for a single user process. A shared server configuration lets many user processes share a small number of server processes, minimising the number of server processes and maximising the use of available system resources.

On some systems, the user and server processes are separate, while on others they are combined into a single process. If a system uses the shared server or if the user and server processes run on different machines, then the user and server processes must be separate. Client/server systems separate the user and server processes and run them on different machines.

Let us discuss a few important process briefly next.

Database Writer (DBWR)

The Database Writer process writes database blocks from the database buffer cache in the SGA to the datafiles on disk. An Oracle instance can have up to 10 DBWR processes, from DBW0 to DBW9, to handle the I/O load to multiple datafiles. Most instances run one DBWR. DBWR writes blocks out of the cache for two main reasons:

- If Oracle needs to perform a checkpoint (i.e., to update the blocks of the datafiles so that they "catch up" to the redo logs). Oracle writes the redo for a transaction when it's committed, and later writes the actual blocks. Periodically, Oracle performs a checkpoint to bring the datafile contents in line with the redo that was written out for the committed transactions.
- If Oracle needs to read blocks requested by users into the cache and there is no free space in the buffer cache. The blocks written out are the least recently used blocks. Writing blocks in this order minimises the performance impact of losing them from the buffer cache.

Log Writer (LGWR)

The Log Writer process writes the redo information from the log buffer in the SGA to all copies of the current redo log file on disk. As transactions proceed, the associated redo information is stored in the redo log buffer in the SGA. When a transaction is committed, Oracle makes the redo information permanent by invoking the Log Writer to write it to disk.

System Monitor (SMON)

The System Monitor process maintains the overall health and safety of an Oracle instance. SMON performs crash recovery when the instance is started after a failure and coordinates and performs recovery for a failed instance when there is more than one instance accessing the same database, as with Oracle Parallel Server/Real Application Clusters. SMON also cleans up adjacent pieces of free space in the datafiles by merging them into one piece and gets rid of space used for sorting rows when that space is no longer needed.

Process Monitor (PMON)

The Process Monitor process watches over the user processes that access the database. If a user process terminates abnormally, PMON is responsible for cleaning up any of the resources left behind (such as memory) and for releasing any locks held by the failed process.

Archiver (ARC)

The Archiver process reads the redo log files once Oracle has filled them and writes a copy of the used redo log files to the specified archive log destination(s).

An Oracle instance can have up to 10 processes, numbered as described for DBWR above. LGWR will start additional Archivers as needed, based on the load, up to the limit specified by the initialisation parameter LOG_ARCHIVE_MAX_PROCESSES.

Checkpoint (CKPT)

The Checkpoint process works with DBWR to perform checkpoints. CKPT updates the control file and database file headers to update the checkpoint data when the checkpoint is complete.

Recover (RECO)

The Recover process automatically cleans up failed or suspended distributed transactions.

4.4.7 How Oracle Works?

The following example describes the most basic level of operations that Oracle performs. This illustrates an Oracle configuration where the user and associated server process are on separate machines (connected through a network).

1) An instance has started on the computer running Oracle (often called the host or database server).

2) A computer running an application (a local machine or client workstation) runs the application in a user process. The client application attempts to establish a connection to the server using the proper Oracle Net Services driver.

3) The server is running the proper Oracle Net Services driver. The server detects the connection request from the application and creates a dedicated server process on behalf of the user process.

4) The user runs a SQL statement and commits the transaction. For example, the user changes a name in a row of a table.

5) The server process receives the statement and checks the shared pool for any shared SQL area that contains a similar SQL statement. If a shared SQL area is found, then the server process checks the user's access privileges to the requested data, and the previously existing shared SQL area is used to process the statement. If not, then a new shared SQL area is allocated for the statement, so it can be parsed and processed.

6) The server process retrieves any necessary data values from the actual datafile (table) or those stored in the SGA.

7) The server process modifies data in the system global area. The DBW*n* process writes modified blocks permanently to the disk when doing so is efficient. Since, the transaction is committed, the LGWR process immediately records the transaction in the redo log file.

8) If the transaction is successful, then the server process sends a message across the network to the application. If it is not successful, then an error message is transmitted.

9) Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

☞ **Check Your Progress 2**

1) What are the different files used in ORACLE?

……

……

……

……

2) ……Storage Structure enables Oracle to have fine – gained control of disk space use.

3) ……..are the basic unit of data storage in an ORACLE database.

4) What are the advantages of B+ tree structure?

……

……

……

……

5) What does a Data Dictionary store?

……

……

……

……

6) ……is a handle for a private SQL area in which a parsed statement and other information for processing the statement are kept.

7) ……is a memory buffer that contains data and control information for a server process.

8) What is a process in Oracle?

……

……

……

……

9) Expand the following:

a) LGWR
b) ARC
c) RECO.

4.5 QUERY PROCESSING AND OPTIMISATION

This section provides the various stages of the execution of a SQL statement in each stage of DML statement processing. The following stages are necessary for each type of statement processing:

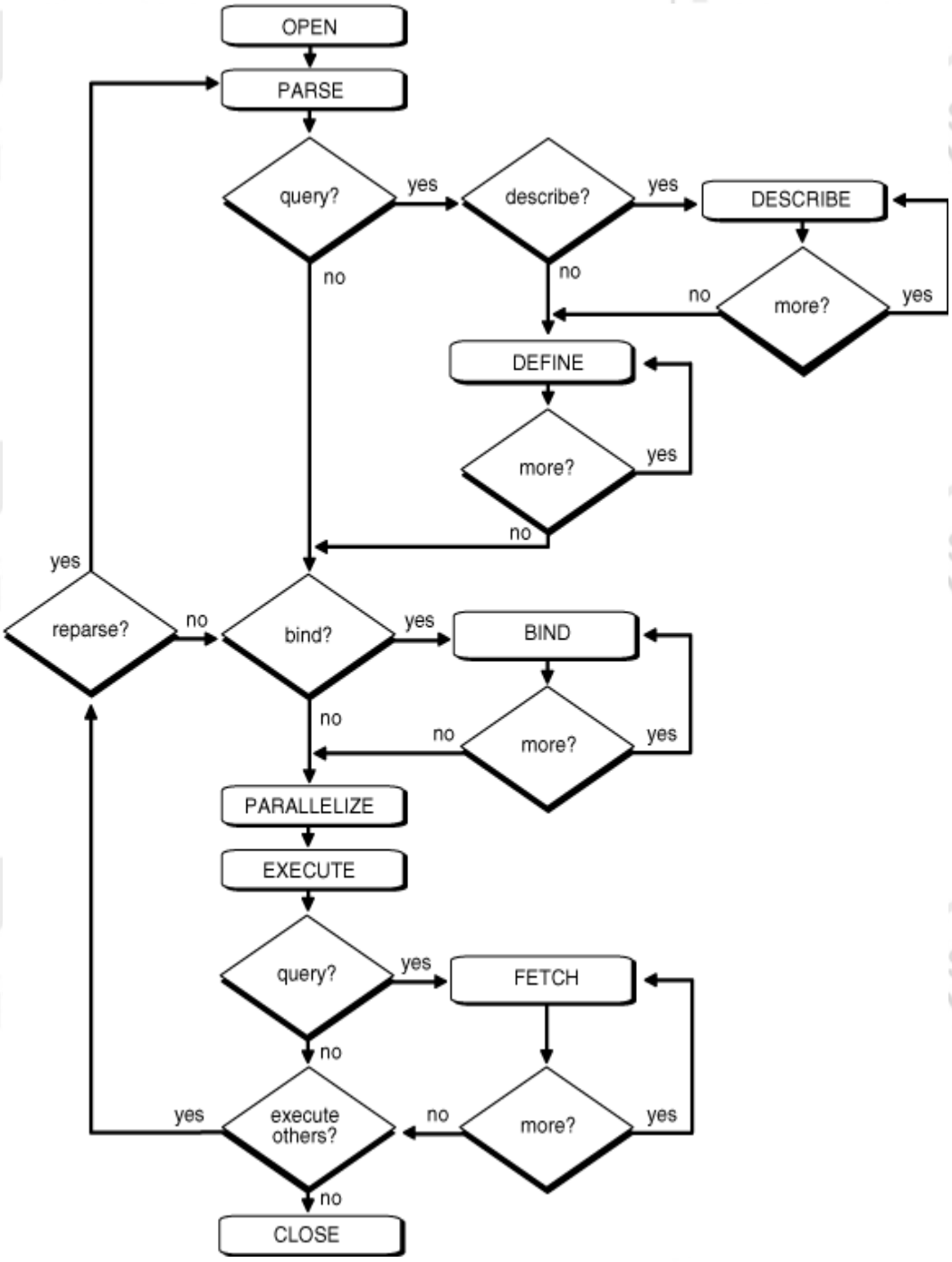

Figure 3: Stages in Query Processing (Source: www.oracle.com)

Stage 1: Create a Cursor

A program interface call creates a cursor. This cursor is not the Oracle PL/SQL cursor but the reference to memory space that is used to keep and manipulate the data in the primary memory. The cursor is created independent of any SQL statement: it is created in expectation of any SQL statement. In most applications, the cursor creation is automatic. However, in pre-compiler programs, cursor creation can either occur implicitly or be explicitly declared.

Stage 2: Parse the Statement

During parsing, the SQL statement is passed from the user process to Oracle, and a parsed representation of the SQL statement is loaded into a shared SQL area. Many errors may be detected during this stage of statement processing.
Parsing is the process of:

- translating a SQL statement, verifying it to be a valid statement,
- performing data dictionary lookups to check table and column definitions,
- acquiring parse locks on required objects so that their definitions do not change during the statement's parsing,
- checking privileges to access referenced schema objects,
- determining the optimal execution plan for the statement,
- loading it into a shared SQL area, and
- routing all or part of distributed statements to remote nodes that contain referenced data.

Oracle parses a SQL statement only if a shared SQL area for a similar SQL statement does not exist in the shared pool. In this case, a new-shared SQL area is allocated, and the statement is parsed.

The parse stage includes processing requirements that need to be done only once, no matter, how many times the statement is executed. Oracle translates each SQL statement only once, re-executing that parsed statement during subsequent references to the statement.

Although parsing a SQL statement validates that statement, parsing only identifies errors that can be found before the execution of the statement. Thus, some errors cannot be caught by parsing. For example, errors in data conversion or errors in data (such as an attempt to enter duplicate values in a primary key) and deadlocks are all errors or situations that can be encountered and reported only during the execution stage.

Note: Queries are different from other types of SQL statements because, if successful, they return data as results. Whereas, other statements simply return success or failure, for instance, a query can return one row or thousands of rows. The results of a query are always in tabular format, and the rows of the result are fetched (retrieved), either a row at a time or in groups.

Several issues are related only to query processing. Queries include not only explicit SELECT statements but also the implicit queries (subqueries) in other SQL statements. For example, each of the following statements, require a query as a part of its execution:

INSERT INTO table SELECT...

UPDATE table SET x = y WHERE...

DELETE FROM table WHERE...

CREATE table AS SELECT...

In particular, queries:

- require read consistency,
- can use temporary segments for intermediate processing, and
- can require the describe, define, and fetch stages of SQL statement processing.

Stage 3: Describe Results of a Query

The describe stage is necessary only if the characteristics of a query's result are not known; for example, when a query is entered interactively by a user. In this case, the describe stage determines the characteristics (datatypes, lengths, and names) of a query's result.

Stage 4: Define Output of a Query

In the define stage of a query, you specify the location, size, and datatype of variables defined to receive each fetched value. Oracle performs datatype conversion if necessary.

Stage 5: Bind any Variables

At this point, Oracle knows the meaning of the SQL statement but still does not have enough information to execute the statement. Oracle needs values for any variables listed in the statement; for example, Oracle needs a value for DEPT_NUMBER. The process of obtaining these values is called binding variables.

A program must specify the location (memory address) of the value. End users of applications may be unaware that they are specifying bind variables, because the Oracle utility can simply prompt them for a new value.

Because you specify the location (binding by reference), you need not rebind the variable before re-execution. You can change its value and Oracle will look up the value on each execution, using the memory address.

You must also specify a datatype and length for each value (unless they are implied or defaulted) in order for Oracle to perform datatype conversions.

Stage 6: Parallelise the Statement

Oracle can parallelise queries (SELECTs, INSERTs, UPDATEs, MERGEs, DELETEs), and some DDL operations such as index creation, creating a table with a subquery, and operations on partitions. Parallelisation causes multiple server processes to perform the work of the SQL statement so it can be completed faster.

Stage 7: Execute the Statement

At this point, Oracle has all the necessary information and resources, so the statement is executed. If the statement is a query or an INSERT statement, no rows need to be locked because no data is being changed. If the statement is an UPDATE or DELETE statement, however, all rows that the statement affects are locked from use by other users of the database until the next COMMIT, ROLLBACK, or SAVEPOINT for the transaction. This ensures data integrity.

For some statements you can specify a number of executions to be performed. This is called array processing. Given *n* number of executions, the bind and define locations are assumed to be the beginning of an array of size *n*.

Stage 8: Fetch Rows of a Query

In the fetch stage, rows are selected and ordered (if requested by the query), and each successive fetch retrieves another row of the result until the last row has been fetched.

Stage 9: Close the Cursor

The final stage of processing a SQL statement is closing the cursor.

Query Optimisation

An important facet of database system performance tuning is the tuning of SQL statements. SQL tuning involves three basic steps:

- Identifying high load or top SQL statements that are responsible for a large share of the application workload and system resources, by reviewing past SQL execution history available in the system.
- Verifying that the execution plans produced by the query optimiser for these statements perform reasonably.

- Implementing corrective actions to generate better execution plans for poorly performing SQL statements.

These three steps are repeated until the system performance reaches a satisfactory level or no more statements can be tuned.

A SQL statement can be executed in many different ways, such as full table scans, index scans, nested loops, and hash joins. The Oracle query optimiser determines the most efficient way to execute a SQL statement after considering many factors related to the objects referenced and the conditions specified in the query. This determination is an important step in the processing of any SQL statement and can greatly affect execution time.

The query optimiser determines most efficient execution plan by considering available access paths and by factoring in information based on statistics for the schema objects (tables or indexes) accessed by the SQL statement. The query optimiser also considers hints, which are optimisation suggestions placed in a comment in the statement.

The query optimiser performs the following steps:

1) The optimiser generates a set of potential plans for the SQL statement based on available access paths and hints.

2) The optimiser estimates the cost of each plan based on statistics in the data dictionary for data distribution and storage characteristics of the tables, indexes, and partitions accessed by the statement.

   The cost is an estimated value proportional to the expected resource use needed to execute the statement with a particular plan. The optimiser calculates the cost of access paths and join orders based on the estimated computer resources, which includes I/O, CPU, and memory.

   Serial plans with higher costs take more time to execute than those with lesser costs. When using a parallel plan, however, resource use is not directly related to elapsed time.

3) The optimiser compares the costs of the plans and chooses the one with the lowest cost.

4.6 DISTRIBUTED ORACLE

One of the strongest features of the Oracle database is its ability to scale up to handling extremely large volumes of data and users. Oracle scales not only by running on more and more powerful platforms, but also by running in a distributed configuration. Oracle databases on separate platforms can be combined to act as a single logical distributed database. This section describes some of the basic ways that Oracle handles database interactions in a distributed database system.

4.6.1 Distributed Queries and Transactions

Data within an organisation is often spread among multiple databases for reasons of both capacity and organisational responsibility. Users may want to query this distributed data or update it as if it existed within a single database. Oracle, first, introduced distributed databases in response to the requirements for accessing data on multiple platforms in the early 1980s. Distributed queries can retrieve data from multiple databases. Distributed transactions can insert, update, or delete data on distributed databases. Oracle's two-phase commit mechanism guarantees that all the database servers that are part of a transaction will either commit or roll back the

transaction. Background recovery processes can ensure database consistency in the event of system interruption during distributed transactions. Once the failed system comes back online, the same process will complete the distributed transactions.

Distributed transactions can also be implemented using popular transaction monitors (TPs) that interact with Oracle via XA, an industry standard (X/Open) interface. Oracle8i added native transaction coordination with the Microsoft Transaction Server (MTS), so you can implement a distributed transaction initiated under the control of MTS through an Oracle database.

4.6.2 Heterogeneous Services

Heterogeneous Services allow non-Oracle data and services to be accessed from an Oracle database through generic connectivity via ODBC and OLE-DB included with the database. Optional Transparent Gateways use agents specifically tailored for a variety of target systems. Transparent Gateways allow users to submit Oracle SQL statements to a non-Oracle distributed database source and have them automatically translated into the SQL dialect of the non-Oracle source system, which remains transparent to the user. In addition to providing underlying SQL services, Heterogeneous Services provide transaction services utilising Oracle's two-phase commit with non-Oracle databases and procedural services that call for third-generation language routines on non-Oracle systems. Users interact with the Oracle database as if all objects are stored in the Oracle database, and Heterogeneous Services handle the transparent interaction with the foreign database on the user's behalf.

4.7 DATA MOVEMENT IN ORACLE

Moving data from one Oracle database to another is often a requirement when using distributed databases, or when a user wants to implement multiple copies of the same database in multiple locations to reduce network traffic or increase data availability. You can export data and data dictionaries (metadata) from one database and import them into another. Oracle Database 10g introduces a new high speed data pump for the import and export of data. Oracle also offers many other advanced features in this category, including replication, transportable tablespaces, and Advanced Queuing.

The ensuing sections describe the technology used to move data from one Oracle database to another automatically.

4.7.1 Basic Replication

You can use basic replication to move recently, added and updated data from an Oracle "master" database to databases on which duplicate sets of data reside. In basic replication, only the single master is updated. You can manage replication through the Oracle Enterprise Manager (OEM or EM). While replication has been a part of all recent Oracle releases, replication based on logs is a more recent addition, appearing for the first time in Oracle9i Release 2.

4.7.2 Advanced Replication

You can use advanced replication in multi-master systems in which any of the databases involved can be updated and in which conflict-resolution features are required to resolve inconsistencies in the data. Because, there is more than one master database, the same data may be updated on multiple systems at the same time. Conflict resolution is necessary to determine the "true" version of the data. Oracle's advanced replication includes a number of conflict-resolution scenarios and also allows programmers to write their own conflict-resolution scenarios.

4.7.3 Transportable Tablespaces

Transportable tablespaces were introduced in Oracle8i. Instead of using the export/import process, which dumps data and the structures that contain it into an intermediate files for loading, you simply put the tablespaces in read-only mode, move or copy them from one database to another, and mount them. You must export the data dictionary (metadata) for the tablespace from the source and import it at the target. This feature can save a lot of time during maintenance, because it simplifies the process. Oracle Database 10g allows you to move data with transportable tablespaces between different platforms or operating systems.

4.7.4 Advanced Queuing and Streams

Advanced Queuing (AQ), first introduced in Oracle8, provides the means to asynchronously send messages from one Oracle database to another. Because messages are stored in a queue in the database and sent asynchronously when a connection is made, the amount of overhead and network traffic is much lower than it would be using traditional guaranteed delivery through the two-phase commit protocol between source and target. By storing the messages in the database, AQ provides a solution with greater recoverability than other queuing solutions that store messages in file systems.

Oracle messaging adds the capability to develop and deploy a content-based publish and subscribe solution using the rules engine to determine relevant subscribing applications. As new content is published to a subscriber list, the rules on the list determine which subscribers should receive the content. This approach means that a single list can efficiently serve the needs of different subscriber communities.

In the first release of Oracle9i, AQ added XML support and Oracle Internet Directory (OID) integration. This technology is leveraged in Oracle Application Interconnect (OAI), which includes adapters to non-Oracle applications, messaging products, and databases.

The second release of Oracle9i introduced Streams. Streams have three major components: log-based replication for data capture, queuing for data staging, and user-defined rules for data consumption. Oracle Database 10g includes support for change data capture and file transfer solutions via Streams.

4.7.5 Extraction, Transformation and Loading

Oracle Warehouse Builder is a tool for the design of target data stores including data warehouses and a metadata repository, but it also provides a front-end to building source-to-target maps and for generating extraction, transformation, and loading (ETL) scripts. OWB leverages key embedded ETL features first made available in the Oracle9i database.

4.8 DATABASE ADMINISTRATION TOOLS

Oracle includes many features that make the database easier to manage. We will discuss two types of tools: Oracle Enterprise Manager, and add-on packs in this section. The tools for backup and recovery, and database availability will be explained in the next section.

Oracle Enterprise Manager

As part of every Database Server, Oracle provides the Oracle Enterprise Manager (EM), a database management tool framework with a graphical interface to manage database users, instances, and features (such as replication) that can provide additional

information about the Oracle environment. EM can also manage Oracle's Application Server, Collaboration Suite, and E-Business Suite.

Prior to the Oracle8i database, the EM software was installed on Windows-based systems. Each repository was accessible only by a single database manager at a time. EM evolved to a Java release providing access from a browser or Windows-based system. Multiple database administrators could then access the EM repository at the same time.

More recently, an EM HTML console was released with Oracle9iAS. This console has important new application performance management and configuration management features. The HTML version supplemented the Java-based Enterprise Manager earlier available. Enterprise Manager 10g, released with Oracle Database 10g, also comes in Java and HTML versions. EM can be deployed in several ways: as a central console for monitoring multiple databases leveraging agents, as a "product console" (easily installed with each individual database), or through remote access, also known as "studio mode". The HTML-based console includes advanced management capabilities for rapid installation, deployment across grids of computers, provisioning, upgrades, and automated patching.

Oracle Enterprise Manager 10g has several additional options (sometimes called packs) for managing the Oracle Enterprise Edition database. These options, which are available for the HTML-based console, the Java-based console, or both, include:

- Database Diagnostics Option
- Application Server Diagnostics Option
- Database Tuning Option
- Database Change Management Option
- Database Configuration Management Option
- Application Server Configuration Management Option.

Standard management pack functionality for managing the Standard Edition is now also available for the HTML-based console.

4.9 BACKUP AND RECOVERY IN ORACLE

As every database administrator knows, backing up a database is a rather mundane but necessary task. An improper backup makes recovery difficult, if not impossible. Unfortunately, people often realise the extreme importance of this everyday task only when it is too late – usually after losing critical business data due to the failure of a related system.

Recovery Manager

Typical backups include complete database backups (the most common type), tablespace backups, datafile backups, control file backups, and archived redo log backups. Oracle8 introduced the Recovery Manager (RMAN) for server-managed backup and recovery of the database. Previously, Oracle's Enterprise Backup Utility (EBU) provided a similar solution on some platforms. However, RMAN, with its Recovery Catalogue we stored in an Oracle database, provides a much more complete solution. RMAN can automatically locate, back up, restore, and recover datafiles, control files, and archived redo logs. RMAN, since Oracle9i, can restart backups, restore and implement recovery window policies when backups expire. The Oracle Enterprise Manager Backup Manager provides a GUI-based interface to RMAN. Oracle Enterprise Manager 10g introduces a new improved job scheduler that can be used with RMAN and other scripts, and that can manage automatic backups to disk.

Incremental Backup and Recovery

RMAN can perform incremental backups of Enterprise Edition databases. Incremental backups, back up, only the blocks modified since the last backup of a datafile, tablespace, or database; thus, they are smaller and faster than complete backups. RMAN can also perform point-in-time recovery, which allows the recovery of data until just prior to an undesirable event (such as the mistaken dropping of a table).

Oracle Storage Manager and Automated Disk Based Backup and Recovery

Various media-management software vendors support RMAN. Since Oracle8i, a Storage Manager has been developed with Oracle to provide media-management services, including the tracking of tape volumes, for up to four devices. RMAN interfaces automatically with the media-management software to request the mounting of tapes as needed for backup and recovery operations.

Oracle Database 10g introduces Automated Disk Based Backup and Recovery. The disk acts as a cache, and archives and backups can then be copied to tape. The disk "cache" can also serve as a staging area for recovery.

4.10 ORACLE LITE

Oracle Lite is Oracle's suite of products for enabling mobile use of database-centric applications. The key components of Oracle Lite include the Oracle Lite Database, Mobile Development Kit, and Mobile Server (an extension of the Oracle Application Server).

Although the Oracle Lite Database engine runs on a much smaller platform than other Oracle implementations (it requires a 50K to 1 MB footprint depending on the platform), Mobile SQL, C++, and Java-based applications can run against the database. ODBC is therefore, supported. Java support includes Java stored procedures and JDBC. The database is self-tuning and self-administering. In addition to Windows-based laptops, Oracle Lite also supports for handheld devices running on WindowsCE, Palm's Computing Platform, and Symbian EPOC.

4.11 SCALABILITY AND PERFORMANCE FEATURES OF ORACLE

Oracle includes several software mechanisms to fulfil the following important requirements of an information management system:

- Data concurrency of a multiuser system must be maximised.
- Data must be read and modified in a consistent fashion. The data a user is viewing or changing is not changed (by other users) until the user is finished with the data.
- High performance is required for maximum productivity from the many users of the database system.

4.11.1 Concurrency

A primary concern of a multiuser database management system is controlling concurrency, which is the simultaneous access of the same data by many users. Without adequate concurrency controls, data could be updated or changed improperly, compromising data integrity.

One way to manage data concurrency is to make each user wait for a turn. The goal of a database management system is to reduce that wait so it is either nonexistent or negligible to each user. All data manipulation language statements should proceed with as little interference as possible, and undesirable interactions among concurrent transactions must be prevented. Neither performance nor data integrity can be compromised.

Oracle resolves such issues by using various types of locks and a multiversion consistency model. These features are based on the concept of a transaction. It is the application designer's responsibility to ensure that transactions fully exploit these concurrency and consistency features.

4.11.2 Read Consistency

Read consistency, as supported by Oracle, does the following:

- Guarantees that the set of data seen by a statement is consistent with respect to a single point in time and does not change during statement execution (statement-level read consistency).
- Ensures that readers of database data do not wait for writers or other readers of the same data.
- Ensures that writers of database data do not wait for readers of the same data.
- Ensures that writers only wait for other writers if they attempt to update identical rows in concurrent transactions.

The simplest way to think of Oracle's implementation of read consistency is to imagine each user operating a private copy of the database, hence the multiversion consistency model.

To manage the multiversion consistency model, Oracle must create a read-consistent set of data when a table is queried (read) and simultaneously updated (written). When an update occurs, the original data values changed by the update are recorded in the database undo records. As long as this update remains part of an uncommitted transaction, any user that later queries the modified data views the original data values. Oracle uses current information in the system global area and information in the undo records to construct a read-consistent view of a table's data for a query.

Only when a transaction is committed are the changes of the transaction made permanent. Statements that start *after* the user's transaction is committed only see the changes made by the committed transaction.

Transaction is the key to Oracle's strategy for providing read consistency. This unit of committed (or uncommitted) SQL statements dictates the start point for read-consistent views generated on behalf of readers and controls modified data can be seen by other transactions of the time span when, database for reading or updating.

By default, Oracle guarantees statement-level read consistency. The set of data returned by a single query is consistent with respect to a single point in time. However, in some situations, you might also require transaction-level read consistency. This is the ability to run multiple queries within a single transaction, all of which are read-consistent with respect to the same point in time, so that queries in this transaction do not see the effects of intervening committed transactions. If you want to run a number of queries against multiple tables and if you are not doing any updating, you would prefer a read-only transaction.

4.11.3 Locking Mechanisms

Oracle also uses locks to control concurrent access to data. When updating information, the data server holds that information with a lock, until, the update is submitted or committed. Until that happens, no one else can make changes to the locked information. This ensures the data integrity of the system.

Oracle provides unique non-escalating row-level locking. Unlike other data servers that "escalate" locks to cover entire groups of rows or even the entire table, Oracle always locks only the row of information being updated. Because Oracle includes the locking information with the actual rows themselves, Oracle can lock an unlimited number of rows so users can work concurrently without unnecessary delays.

Automatic Locking

Oracle locking is performed automatically and requires no user action. Implicit locking occurs for SQL statements as necessary, depending on the action requested. Oracle's lock manager automatically locks table data at the row level. By locking table data at the row level, contention for the same data is minimised.

Oracle's lock manager maintains several different types of row locks, depending on the type of operation that established the lock. The two general types of locks are: exclusive locks and share locks. Only one exclusive lock can be placed on a resource (such as a row or a table); however, many share locks can be placed on a single resource. Both exclusive and share locks always allow queries on the locked resource but prohibit other activity on the resource (such as updates and deletes).

Manual Locking

Under some circumstances, a user might want to override default locking. Oracle allows manual override of automatic locking features at both the row level (by first querying for the rows that will be updated in a subsequent statement) and the table level.

4.11.4 Real Application Clusters

Real Application Clusters (RAC) comprises several Oracle instances running on multiple clustered machines, which communicate with each other by means of an interconnect. RAC uses cluster software to access a shared database that resides on a shared disk. RAC combines the processing power of these multiple interconnected computers to provide system redundancy, near linear scalability, and high availability. RAC also offers significant advantages for both OLTP and data warehouse systems and all systems and applications can efficiently exploit clustered environments.

You can scale applications in RAC environments to meet increasing data processing demands without changing the application code. As you add resources such as nodes or storage, RAC extends the processing powers of these resources beyond the limits of the individual components.

4.11.5 Portability

Oracle provides unique portability across all major platforms and ensures that your applications run without modification after changing platforms. This is because the Oracle code base is identical across platforms, so you have identical feature functionality across all platforms, for complete application transparency. Because of this portability, you can easily upgrade to a more powerful server as your requirements change.

☞ Check Your Progress 3

1) What is Parsing?

……

……

……

……

2) What are the three basic steps involved in SQL tuning?

……

……

……

……

3) The oracle ……determines the most efficient way to execute a SQL statement after considering many factors related to the object references and the conditions specified in the Query.

4) What are heterogeneous services?

……

……

……

5) Name the basic technology used to move data from are ORACLE database to another.

……

……

……

……

6) What is Oracle Enterprise manager?

……

……

……

7) What are the different types of backup?

……

……

……

8) …….. is Oracle's suite of products for enabling mobile use of database – centric Applications.

9) …….. comprises several ORACLE instances running on multiple clustered machines.

4.12 ORACLE DATA WAREHOUSING

A data warehouse is a relational database designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organisation to consolidate data from several sources.

In addition to a relational database, a data warehouse environment includes an extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users.

4.12.1 Extraction, Transformation and Loading (ETL)

You must load your data warehouse regularly so that it can serve its purpose of facilitating business analysis. To do this, data from one or more operational systems must be extracted and copied into the warehouse. The process of extracting data from source systems and bringing it into the data warehouse is commonly called ETL, which stands for extraction, transformation, and loading.

4.12.2 Materialised Views

A materialised view provides indirect access to table data by storing the results of a query in a separate schema object. Unlike an ordinary view, which does not take up any storage space or contain any data, a materialised view contains the rows resulting from a query against one or more base tables or views. A materialised view can be stored in the same database as its base tables or in a different database.

Materialised views are stored in the same database as their base tables can improve query performance through query rewrites. Query rewrites are particularly useful in a data warehouse environment.

4.12.3 Bitmap Indexes

Data warehousing environments typically have large amounts of data and *ad hoc* queries, but a low level of concurrent database manipulation language (DML) transactions. For such applications, bitmap indexing provides:

- reduced response time for large classes of ad hoc queries,
- reduced storage requirements compared to other indexing techniques,
- dramatic performance gains even on hardware with a relatively small number of CPUs or a small amount of memory, and
- efficient maintenance during parallel DML and loads.

Fully indexing a large table with a traditional B-tree index can be prohibitively expensive in terms of space because the indexes can be several times larger than the data in the table. Bitmap indexes are typically only a fraction of the size of the indexed data in the table.

4.12.4 Table Compression

To reduce disk use and memory use (specifically, the buffer cache), you can store tables and partitioned tables in a compressed format inside the database. This often leads to better scaleup for read-only operations. Table compression can also speed up query execution. There is, however, a slight cost in CPU overhead.

4.12.5 Parallel Execution

When Oracle runs SQL statements in parallels, multiple processes work together simultaneously to run a single SQL statement. By dividing the work, necessary to run a statement among multiple processes, Oracle can run the statement more quickly than if only a single process ran it. This is called parallel execution or parallel processing.

Parallel execution dramatically reduces response time for data-intensive operations on large databases, because statement processing can be split up among many CPUs on a single Oracle system.

4.12.6 Analytic SQL

Oracle has many SQL operations for performing analytic operations in the database. These include ranking, moving averages, cumulative sums, ratio-to-reports, and period-over-period comparisons.

4.12.7 OLAP Capabilities

Application developers can use SQL online analytical processing (OLAP) functions for standard and ad-hoc reporting. For additional analytic functionality, Oracle OLAP provides multidimensional calculations, forecasting, modeling, and what-if scenarios. This enables developers to build sophisticated analytic and planning applications such as sales and marketing analysis, enterprise budgeting and financial analysis, and demand planning systems. The data can also be stored in either relational tables or multidimensional objects.

Oracle OLAP provides the query performance and calculation capability, previously found only in multidimensional databases to Oracle's relational platform. In addition, it provides a Java OLAP API that is appropriate for the development of internet-ready analytical applications. Unlike other combinations of OLAP and RDBMS technology, Oracle OLAP is not a multidimensional database using bridges to move data from the relational data store to a multidimensional data store. Instead, it is truly an OLAP-enabled relational database. As a result, Oracle provides the benefits of a multidimensional database along with the scalability, accessibility, security, manageability, and high availability of the Oracle database. The Java OLAP API, which is specifically designed for internet-based analytical applications, offers productive data access.

4.12.8 Data Mining

With Oracle Data Mining, data never leaves the database — the data, data preparation, model building, and model scoring results all remain in the database. This enables Oracle to provide an infrastructure for application developers to integrate data mining seamlessly with database applications. Some typical examples of the applications that data mining are used in are call centers, ATMs, ERM, and business planning applications. Data mining functions such as model building, testing, and scoring are provided through a Java API.

4.12.9 Partitioning

Partitioning addresses key issues in supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions. SQL queries and DML statements do not need to be modified in order to access partitioned tables. However, after partitions are defined, DDL statements can access and manipulate individual partitions rather than entire tables or indexes. This is how partitioning can simplify the manageability of large database objects. Also, partitioning is entirely transparent to applications.

Partitioning is useful for many different types of applications, particularly applications that manage large volumes of data. OLTP systems often benefit from improvements in manageability and availability, while data warehousing systems benefit from performance and manageability.

4.13 SECURITY FEATURES OF ORACLE

Oracle includes security features that control the accesssing and using of a database. For example, security mechanisms in Oracle:

- prevent unauthorised database access,
- prevent unauthorised access to schema objects, and
- audit user actions.

Associated with each database user is a schema by the same name. By default, each database user creates and has access to all objects in the corresponding schema.

Database security can be classified into two categories: system security and data security.

System security includes mechanisms that control the access and use of the database at the system level. For example, system security includes:

- valid user name/password combinations,
- the amount of disk space available to a user's schema objects, and
- the resource limits for a user.

System security mechanisms check whether a user is authorised to connect to the database, whether database auditing is active, and the system operations that a user has been permitted to perform.

Data security includes mechanisms that control the access and use of the database at the schema object level. For example, data security includes:

- users with access to a specific schema object and the specific types of actions permitted to each user on the schema object (for example, user MOHAN can issue SELECT and INSERT statements but not DELETE statements using the employees table),
- the actions, if any, that are audited for each schema object, and
- data encryption to prevent unauthorised users from bypassing Oracle and accessing the data.

Security Mechanisms

The Oracle database provides discretionary access control, which is a means of restricting access to information based on privileges. The appropriate privilege must be assigned to a user in order for that user to access a schema object. Appropriately privileged users can grant other users privileges at their discretion.

Oracle manages database security using several different facilities:

- authentication to validate the identity of the entities using your networks, databases, and applications,
- authorisation processes to limit access and actions, limits that are linked to user's identities and roles,
- access restrictions on objects, like tables or rows,
- security policies, and
- database auditing.

4.14 DATA INTEGRITY AND TRIGGERS IN ORACLE

Data must adhere to certain business rules, as determined by the database administrator or application developer. For example, assume that a business rule says that no row in the inventory table can contain a numeric value greater than nine in the sale_discount column. If an INSERT or UPDATE statement attempts to violate this integrity rule, then Oracle must undo the invalid statement and return an error to the

application. Oracle provides integrity constraints and database triggers to manage data integrity rules.

Database triggers let you define and enforce integrity rules, but a database trigger is not the same as an integrity constraint. Among other things, a database trigger does not check data already loaded into a table. Therefore, it is strongly recommended that you use database triggers only when the integrity rule cannot be enforced by integrity constraints.

Integrity Constraints

An integrity constraint is a declarative way to define a business rule for a column of a table. An integrity constraint is a statement about table data that is always true and that follows these rules:

- If an integrity constraint is created for a table and some existing table data does not satisfy the constraint, then the constraint cannot be enforced.

- After a constraint is defined, if any of the results of a DML statement violate the integrity constraint, then the statement is rolled back, and an error is returned.

Integrity constraints are defined with a table and are stored as part of the table's definition in the data dictionary, so that all database applications adhere to the same set of rules. When a rule changes, it only needs be changed once at the database level and not for each application.

The following integrity constraints are supported by Oracle:

- NOT NULL: Disallows nulls (empty entries) in a table's column.
- UNIQUE KEY: Disallows duplicate values in a column or set of columns.
- PRIMARY KEY: Disallows duplicate values and nulls in a column or set of columns.
- FOREIGN KEY: Requires each value in a column or set of columns to match a value in a related table's UNIQUE or PRIMARY KEY. FOREIGN KEY integrity constraints also define referential integrity actions that dictate what Oracle should do with dependent data if the data references is altered.
- CHECK: Disallows values that do not satisfy the logical expression of the constraint.

Keys

Key is used to define several types of integrity constraints. A key is the column or set of columns included in the definition of certain types of integrity constraints. Keys describe the relationships between the different tables and columns of a relational database. Individual values in a key are called key values.
The different types of keys include:

- **Primary key:** The column or set of columns included in the definition of a table's PRIMARY KEY constraint. A primary key's value uniquely identifies the rows in a table. Only one primary key can be defined for each table.

- **Unique key:** The column or set of columns included in the definition of a UNIQUE constraint.

- **Foreign key**: The column or set of columns included in the definition of a referential integrity constraint.

- **Referenced key:** The unique key or primary key of the same or a different table referenced by a foreign key.

Triggers

Triggers are procedures written in PL/SQL, Java, or C that run (fire) implicitly, whenever a table or view is modified or when some user actions or database system action occurs.

Triggers supplement the standard capabilities of Oracle to provide a highly customised database management system. For example, a trigger can restrict DML operations against a table to those issued during regular business hours.

4.15 TRANSACTIONS IN ORACLE

A transaction is a logical unit of work that comprises one or more SQL statements run by a single user. According to the ANSI/ISO SQL standard, with which Oracle is compatible, a transaction begins with the user's first executable SQL statement. A transaction ends when it is explicitly committed or rolled back by that user.

Transactions let users guarantee consistent changes to data, as long as the SQL statements within a transaction are grouped logically. A transaction should consist of all necessary parts for one logical unit of work – no more and no less. Data in all referenced tables are in a consistent state before the transaction begins and after it ends. Transactions should consist of only SQL statements that make one consistent change to the data.

Consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operations: decrease in the savings account, increase in the checking account, and recording the transaction in the transaction journal.

The transfer of funds (the transaction) includes increasing one account (one SQL statement), decreasing another account (one SQL statement), and recording the transaction in the journal (one SQL statement). All actions should either fail or succeed together; the credit should not be committed without the debit. Other non-related actions, such as a new deposit to one account, should not be included in the transfer of funds transaction. Such statements should be in other transactions.

Oracle must guarantee that all three SQL statements are performed to maintain accounts accurately. When something prevents one of the statements in the transaction from running (such as a hardware failure), then the other statements of the transaction must be undone. This is called rolling back. If an error occurs in making any of the updates, then no updates are made.

Commit and Undo Transactions

The changes made by the SQL statements that constitute a transaction can be either committed or rolled back. After a transaction is committed or rolled back, the next transaction begins with the next SQL statement.

To commit, a transaction makes permanent the changes resulting from all SQL statements in the transaction. The changes made by the SQL statements of a transaction become visible to other user sessions' transactions that start only after the transaction is committed.

To undo a transaction, retracts, any of the changes resulting from the SQL statements in the transaction. After a transaction is rolled back, the affected data is left unchanged, as if the SQL statements in the transaction were never run.

Savepoints

Savepoints divide a long transaction with many SQL statements into smaller parts. With savepoints, you can arbitrarily mark your work at any point within a long transaction. This gives you the option of later rolling back all work performed from the current point in the transaction to a declared savepoint within the transaction.

4.16 SQL VARIATIONS AND EXTENSIONS IN ORACLE

Oracle is compliant to industry-accepted standards and participates actively in SQL standards committees. Industry-accepted committees are the American National Standards Institute (ANSI) and the International Standards Organisation (ISO), affiliated to the International Electrotechnical Commission (IEC). Both ANSI and the ISO/IEC have accepted SQL as the standard language for relational databases. Oracle 10g conforms Core SQL: 2003 for most of the features except the following partially supported and not supported features.

Oracle partially supports these sub-features:

- CHARACTER VARYING data type (Oracle does not distinguish a zero-length VARCHAR string from NULL).
- Character literals (Oracle regards the zero-length literal "as being null").
- Correlation names in FROM clause (Oracle supports correlation names, but not the optional AS keyword).
- WITH HOLD cursors (in the standard, a cursor is not held through a ROLLBACK, but Oracle does hold through ROLLBACK).
- ALTER TABLE statement: ADD COLUMN clause (Oracle does not support the optional keyword COLUMN in this syntax).
- User-defined functions with no overloading.
- User-defined procedures with no overloading.
- In the standard, the mode of a parameter (IN, OUT or INOUT) comes before the parameter name, whereas in Oracle it comes after the parameter name.
- The standard uses INOUT, whereas Oracle uses IN OUT.
- Oracle requires either IS or AS after the return type and before the definition of the routine body, while the standard lacks these keywords.

Oracle does not support the following sub-features:

- Rename columns in the FROM clause.
- DROP TABLE statement: RESTRICT clause.
- DROP VIEW statement: RESTRICT clause.
- REVOKE statement: RESTRICT clause.
- Features and conformance views.
- Distinct data types.

Oracle supports the following subfeatures in PL/SQL but not in Oracle SQL:

- RETURN statement.

Oracle's triggers differ from the standard as follows:

- Oracle does not provide the optional syntax FOR EACH STATEMENT for the default case, the statement trigger.
- Oracle does not support OLD TABLE and NEW TABLE; the transition tables specified in the standard (the multiset of before and after images of affected rows) are not available.
- The trigger body is written in PL/SQL, which is functionally equivalent to the standard's procedural language PSM, but not the same.

- In the trigger body, the new and old transition variables are referenced beginning with a colon.
- Oracle's row triggers are executed as the row is processed, instead of buffering them and executing all of them after processing all rows. The standard's semantics are deterministic, but Oracle's in-flight row triggers are more performant.
- Oracle's before row and before-statement triggers may perform DML statements, which is forbidden in the standard. On the other hand, Oracle's after-row statements may not perform DML, while it is permitted in the standard.
- When multiple triggers apply, the standard says they are executed in order of definition; in Oracle the execution order is non-deterministic.

In addition to traditional structured data, Oracle is capable of storing, retrieving, and processing more complex data.

- Object types, collection types, and REF types provide support for complex structured data. A number of standard-compliant multiset operators are now supported by the nested table collection type.
- Large objects (LOBs) provide support for both character and binary unstructured data. A single LOB reach a size of 8 to 128 terabytes, depending on the size of the database block.
- The XML datatype provides support for semistructured XML data.

☞ Check Your Progress 4

1) What is ETL?

……

……

2) What is Parallel Execution?

……

……

3) With Oracle ……data never leaves the database.

4) What does system security include?

……

……

5) How does Oracle manage database security?

……

……

6) ……is a declarative way to defined a business rule for a column of a table.

7) What are the different integrity constraints supported by oracle?

……

……

8) Name the different types of keys.

……

……

9) What is a trigger?

……

……

10) ……divides a long transaction with many SQL statements into smaller parts.

4.17 SUMMARY

This unit provided an introduction to Oracle, one of the commercial DBMS in the market. Some other products include MS SQL server, DB2 by IBM and so on. Oracle being a commecial DBMS supports the features of the Relational Model and also some of the Object oriented models. It has a very powerful Database Application Development Feature that is used for database programming. Oracle supports the Standard Query Language (SQL) and the use of embedded languages including C, JAVA etc.

The Oracle Architecture defines the physical and logical database components of the database that is stored in the Oracle environment. All database objects are defined using a schema definition. This information is stored in the data dictionary. This data dictionary is used actively to find the schema objects, integrity and security constraints on those objects etc. Oracle defines an instance as the present database state. There are many Oracle backend processes that support various operations on oracle database. The database is stored in the SGA area of the memory.

Oracle is relational as the basic technology supports query optimisation. Query optimisation is needed for commercial databases, as the size of such databases may be very high. Oracle supports indexing using the B-tree structure, in addition the bit wise index makes indexing even faster. Oracle also supports distributed database technology. It supports replication, transportability of these replicas and advanced queuing and streams.

Oracle has many tools for system administration. They support basic used management to backup and recovery tools. Oracle Lite is the version of Oracle used for mobile database. Oracle uses standard implementation methods for concurrency control. Oracle has Data Warehousing capabilities. It contains Extraction, Transformation and Loading (ETL) tools, materialised views, bitmap indexes, table compression and Data mining and OLAP in support to a Data Warehouse. Oracle supports both system and database security features.

The unit also explained data integrity, transactions, SQL Variations and Extensions in Oracle.

4.18 SOLUTIONS/ANSWERS

Check Your Progress 1

1) These statements create, alter, maintain and drop schemes objects.

2) Transaction Control Statements.

3) (a) Character
   (b) Numeric
   (c) DATE data type
   (d) LOB data type
   (e) RAW and LONGRAW
   (f) ROWID and UNROWID data type

4) ORACLE forms Developer
   ORACLE Reports Developer
   ORACLE Designer
   ORACLE J Developer
   ORACLE Discoverer Administrative Edition

ORACLE Portal.

Check Your Progress 2

1) (a) Data files
   (b) Redo log files
   (c) Control files

2) Logical.

3) Tables

4)
   (a) All Leaf Block have same depth,
   (b) B-Tree indexes are balanced,
   (c) Blocks of the B+ tree are 3 quarters full on the average,
   (d) B+ tree have excellent retrieval performance,
   (e) Inserts, updates and deletes are all efficient, and
   (f) Good Performance.

5)
   (1) Valid users of Oracle data types,
   (2) Information about integrity Constraints defined for the tables, and
   (3) The Amount of space Allocated for a scheme Object.

6) Cursor

7) Program Global Area

8) A process is a thread of control or a mechanism in an operating system that can run a series of steps.

9)
   (a) Log writers,
   (b) Archiver,
   (c) Recover.

Check Your Progress 3

1) Translating a SQL statement and verifying it to be a valid statement.

2)
   (1) Identifying high load on top of SQL Statements.
   (2) Verifying that the execution plans perform reasonably, and
   (3) Implementing corrective actions to generate better execution plans.

3) Query optimiser

4) It allows non – oracle data and services to be accessed from an Oracle database through generic connectivity.

5)
   (1) Basic replication,
   (2) Advanced replication,
   (3) Transportable replication,
   (4) Advanced queuing and streams, and
   (5) Extraction, transformation, loading.

6) A complete interface for enterprise wide application development

7)
   (a) Complete database,
   (b) Tablespace,
   (c) Data file,
   (d) Control file, and
   (e) Achieved Redo Log.

8) Oracle LITE.

9) Real application clusters.

Check Your Progress 4

1) ETL means extraction, transformation and loading. It is basically the process of extracting data from source systems and transferring it to the data warehouse.

2) The process of making multiple processes run together.

3) Data Mining.

4) It includes the mechanisms that control access and use of the database at the system level.

5) Oracle manages database security using:
   - Authentication,
   - Authorisation,
   - Access restrictions,
   - Security policies, and
   - Database Auditing.

6) Integrity constraints.

7)
   - NOT NULL
   - UNIQUE KEY
   - PRIMARY KEY
   - FOREIGN KEY
   - CHECK

8)
   - Primary
   - Unique
   - Foreign
   - Referenced

9) It is a event driven procedure.

10) Save points.