Finacle®
Transforming Banking

Administrator Training Manual

ORACLE FUNDAMENTALS

Infosys

www.infosysinbanking.com

Document number:		VersionRev:	1.00
Authorized by:	R N NAGARAJ	Signature/Date:	

Document revision list

Ver.Rev	Date	Author	Description
1.00	28-10-2003	Anju Jocelin Mathew	*Updated to Finacle 7.0*

Infosys Technologies Limited,
Electronics City,
Hosur Road,
Bangalore – 560 100
India

This Training manual has been written and produced by the *FINACLE -USER EDUCATION TEAM* of Infosys Technologies Limited.

TABLE OF CONTENTS

1 SNAPSHOT

Finacle ™ uses the Oracle Database Engine and sits on UNIX. In order to use the application effectively, knowledge of UNIX and the basic concepts related to a Relational Database Management System (RDBMS) becomes a prerequisite. This document introduces the fundamentals of a relational database management system and in particular, Oracle.

2 SECTION OBJECTIVE

The objective is to teach the user, the fundamentals of relational database concepts and Oracle so that he is able to effectively carry out his duties as a Database Administrator.

3 AN INTRODUCTION TO DATABASE CONCEPTS

This section introduces the fundamentals of database concepts and defines the related terms.

3.1 WHAT IS A DATABASE?

A ***database*** is a collection of *related data. Data* could be known facts that can be recorded and that have an implicit meaning.

For instance, consider the names, telephone numbers and addresses of people. Anybody would normally record this information in an indexed address book or probably in a computer or in an electronic address book. This collection by itself can very well be called a *database.*

Such databases can be either generated or created manually or created by the machine. The library card catalogue could be thought of as an example of a manually maintained database. A computerized database may be created and maintained either by a group of application programs written specifically for the task or by a database management system.

3.2 WHAT IS A DATABASE MANAGEMENT SYSTEM?

A ***database management system*** is a collection of programs that enable users to create and maintain a database. The database management system is a system that facilitates

the processes of ***defining, constructing*** and ***manipulating*** databases for various applications.

- **Defining** a database involves specifying the data types, structures and constraints for the data to be stored in the database.
- **Constructing** the database is the process of storing the data itself on some storage media that is controlled by the database management system.
- **Manipulating** the database includes functions that query the database to retrieve specific data, updating data to reflect changes in and around the application and generating reports from the data.

A database system is an integrated collection of related files, along with details of the interpretation of the data contained therein. The database system allows the users to access and manipulate the data contained in the database in a convenient and effective manner. In addition, the database management system exerts centralized control of the database, prevents fraudulent or unauthorized users from accessing the data and ensures privacy of the data.

4 RELATIONAL DATABASE MANAGEMENT SYSTEMS

Relational databases were a new way of thinking about how data should be structured and stored. The key to this type of database is in understanding the *relationships within data*, then structuring the information base to reflect those relationships. The goal in a relational database is to build a database in which only the data changes and not the structure itself. To appreciate the relational model or approach, it should be compared with the traditional approach. Consider a traditional customer master file. It contains all the normal fields anyone would expect to see, such as customer name, address, city, state, home phone and work phone. There is a separate slot for each item of information; thus, the number of slots depends on the number of different types of data being recorded. This traditional design seemed alright until the use of fax machines became prevalent. Incorporating a fax number into the old model required an additional phone field, which in turn required a complete restructuring of the database. Moreover, this restructuring also demanded a complete redesign of the application code associated with the customer master file. At this juncture, there were two options to choose from,

- If the application code was changed, there were high costs associated with adding new functionality to applications. A mistake could cause an interruption in the service to customers.

- If the application code were left as is, money was saved but there was a different price to pay; access to the fax machine phone numbers of the customers would not be possible. This decision may very well pave way for a technological risk due to lack of adaptation to the changing environment.

Consider a scenario that depicts how cumbersome the traditional approach can get. The employee benefits application has the employee names in a benefits master file. The payroll people maintain employee names in a payroll master file. Suppose a woman changes her last name after getting married and needs her pay check written with her new last name. This change might happen in the payroll master file but may have been missed out in the other files that store her name. It might take months for this inconsistency to be discovered and months again; to ensure this change is incorporated in all files where the employee name is stored.

Traditional systems are ***design-driven***; they require design changes when one needs to capture new kinds of data. Since design changes are expensive, they might not be done at all.

Now, to have a look at the relational approach. Using this approach, system designers isolate types of information that need capturing. They then identify the relationships between those information types and implement a database structure. Using the relational model, what was previously referred to as master files are called tables. There would be a customer table, a phone number table and a phone_number_types table. Each is a separate table within the database. There exist some rules governing the relationships between the customer and phone number data:

- Each customer may have one or more phone numbers
- Each phone number belongs to one and only one customer
- Each phone number must be one and only one type say either a home number, a business number, a fax number or a mobile phone.

In such an approach, when a fax number has to be added, all that needs to be done is to add a row of data to the phone_number_types entity. There is no need to restructure the database or to program any new functionality in such a scenario.

Systems built using the relational model store information once. Changes and additions to that central repository are reflected immediately. Relational systems are ***data-driven***. When additional data needs to be captured, it is not necessary to redesign the system.

4.1 WHAT IS A RELATIONAL DATABASE MANAGEMENT SYSTEM?

A system, which is able to manage its database entirely through its relational capabilities, is called a ***Relational Database Management System.***

MERITS OF A RELATIONAL DATABASE MANAGEMENT SYSTEM

Controlling Redundancy

Centralized control of data avoids unnecessary duplication of data and effectively reduces the total amount of data storage required. It also eliminates the extra processing necessary to trace the required data in a large mass of data. Another advantage of avoiding duplication is the elimination of the inconsistencies that tend to be present in redundant data files. Any redundancies that exist in the database management system are controlled and the system ensures that these multiple copies are consistent.

Restricting Unauthorized Access

Data is of vital importance to an organization and may be confidential. Unauthorized persons must not access such confidential data. The database administrator who has the ultimate responsibility for the data in the database management system can ensure that proper access procedures are followed, including proper authentication schemes for access to the database management system and additional checks before permitting access to sensitive data. Different levels of security could be implemented for various types of data and operations. The enforcement of security could be data value dependent (for e.g., a manager has access to the salary details of employees in his department only), as well as data type dependent (but the manager cannot access the medical history of any employee).

Providing Multiple User Interfaces

A database allows the sharing of data under its control by any number of application programs or users.

Enforcing Integrity constraints

Centralized control can also ensure that adequate checks are incorporated in the database management system to provide data integrity. Data integrity means that the data contained in the database is both accurate and consistent. For instance, data values being

entered for storage could be checked to ensure that they fall within a specified range and are of the correct format.

Representing complex relationships among data

A relational database management system uses the relational model to store and manipulate data. The relational model facilitates the representation of complex relationships among data. The relational model is a mathematical model for representation, manipulation and interpretation of data. It provides algorithms for performing various operations and proves the correctness of these algorithms. A discussion of the relational model is beyond the scope of this document.

Provide Backup and Recovery

The data stored in the database is safe due to the backup and recovery facilities offered by a relational database management system.

DEMERITS OF A RELATIONAL DATABASE MANAGEMENT SYSTEM

High cost of hardware, software and training

In addition to the cost of purchasing or developing software, the hardware has to be upgraded to allow for the extensive programs and the workspaces required for their execution and storage. Besides, the users have to be trained to use the system.

Problems associated with centralization and complexity of backup and recovery.

While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that, in case of failure, the data can be recovered. Backup and recovery operations are fairly complex in a database management system environment and this is exacerbated in a concurrent multi-user database system. Centralization also means that the data is accessible from a single source, namely the database. This increases the potential severity of security breaches and disruption of the operation of the organization because of downtimes and failures. The replacement of a monolithic centralized database by a federation of independent and co-operating distributed databases resolves some of the problems resulting from downtimes and failures.

Overhead for security, concurrency control and integrity functions

Overhead is involved when it comes to taking care of the security of the data by offering restricted access to the database. Moreover, the validity of the data needs to be ensured by enforcing integrity constraints. The system should also offer concurrency control because many users will access the database simultaneously. All these incur overheads in processing.

5 ORACLE

Oracle is a relational database management system from Oracle Systems Corporation, based in Redwood Shores, California. Oracle's suite of products have become popular and successful because of aspects like the security mechanisms it offers, the sophisticated backup and recovery routines, open connectivity to the Oracle database to and from other vendors' software and the wide range of development tools available.

5.1 SOME ORACLE PRODUCTS

SQL*PLUS

SQL*Plus is the interface through which data is defined and manipulated in Oracle's relational database. It is an interactive SQL interface with additional features for formatting and editing tools. SQL stands for Structured Query Language. It is the language that the RDBMS understands. All operations that the user wants to perform are translated into SQL statements/queries and given to the RDBMS which acts accordingly.

SQL*NET

Communication software used for communicating between Client/Server applications.

SQL*PRO*

A High level language interface for ORACLE relational database management system through which applications can embed SQL language statements in the host language.

PL/SQL

PL/SQL is a procedural interface for the ORACLE relational database management system to incorporate programming language features into the relational database management system, since SQL does not provide for programming. Finacle ™ makes use of SQL*Plus, PL/SQL, Proc*C and SQL*Net.

5.2 ORACLE ARCHITECTURE

This chapter introduces the related terminology followed by the architecture of Oracle.

RELATED TERMINOLOGY

Object

An object is a structure defined within the Oracle database and referenced in SQL statements within the application. Most objects are tables, but SQL statements can refer to other kinds of objects.

Instance

An instance is a portion of computer memory and auxiliary processes required to access an Oracle database.

Application

An application is a set of Oracle programs that solve a company's or person's business needs.

Database Administrator

A database administrator, DBA in short, is a technical wizard who manages the complete operation of the Oracle database.

Datafile

A datafile is a file on the disk that stores information.

Tablespace

A tablespace is a collection of one or more datafiles. All database objects are stored in tablespaces. It is called a tablespace because it typically holds one or more database objects.

Table

A table holds data. Space is allocated to a table to hold data in the user's database.

Rollback

Rollback is the activity that Oracle performs to restore data to the state in which it was, before a user changed it.

Undo

Undo information is the information the database needs to undo or rollback a user transaction due to a number of reasons.

Dirty Data Block

A dirty data block is a portion of the computer main memory that contains Oracle data whose value has changed from what was originally read from the database. Dirty blocks are data sitting in main memory that has been changed but not written back to the

database. Please note that the database will reside on the hard disk or some other secondary storage and only those parts which the user is using at the moment are brought into the main memory (or the RAM) at the time of performing the operation.

Hot Block

A hot block is a block whose data is changed frequently.

5.2.1 DATABASE

The general purpose of a database is to store and retrieve related information. An Oracle database has a logical and a physical structure. The **physical structure** of the database is the set of operating system files in the database. An Oracle database consists of three file types.

Data files containing the actual data in the database.

Redo logs containing a record of changes made to the database to enable recovery of the data in case of failures.

Control files containing information necessary to maintain and verify database integrity.

The **logical structure** of the Oracle architecture dictates how the physical space of a database is to be used.

The logical Hierarchy exists as follows:

An Oracle database is a group of tablespaces.

A ***tablespace*** may consist of one or more segments.

A ***segment*** is made up of extents.

An ***extent*** is made up of logical blocks.

A ***block*** is the smallest unit for read and write operations.

5.2.2 DATAFILES

Datafiles contain all the database data. The Oracle database is made up of one or more datafiles; datafiles are grouped together to form a tablespace. The datafiles contain all of the data information stored in the database.

USER DATA AND SYSTEM DATA

Two types of data or information are stored within the datafiles associated with a database: user data and system data.

User data is the application data, with all of the application's relevant information. This is the information the organization stores in the database.

The table below shows the common types of user data.

Type of Data	Contains Information About

Customer Information	Last name, first name, phone number
Product Information	Product Name, availability, price
Medical Information	Lab results, doctor's name, nurse's name
Inventory Information	Quantity in stock, quantity backordered
Financial Information	Stock price, interest rate

System data is the information the database needs to manage the user data and to manage itself. System data tells Oracle the valid users of the database, their passwords, how many datafiles are part of the database and where these datafiles are located.

The table that follows shows some common types of system data.

Type of Data	Contains Information About
Tables	The fields of the table and the type of information they hold
Space	Amount of physical space the database objects take
Users	Names, passwords and privileges
Datafiles	Number, location. time last used

5.2.3 TABLESPACES

Oracle groups datafiles under the umbrella of a database object called a tablespace. Before inserting data into an Oracle database, a tablespace must be created, then a table within that tablespace to hold the data. When creating a table, the information about the type of data should also be included.

The best analogy to explain a database, tablespace, datafile, table and data is an image of a filing cabinet. The drawers within the cabinet are tablespaces; the folders in those drawers are datafiles; the pieces of paper in each folder are the tables; the information written on the paper in each folder is the data. Tablespaces are a way to group datafiles

Tablespace Names and Contents

System Tablespace

The system tablespace is a required part of every Oracle database. This is where Oracle stores all the information it needs to manage itself, such as names of tablespaces and what datafiles each tablespace contains.

Temp Tablespace

The temp tablespace is where Oracle stores all its temporary tables. This is the database's whiteboard or scratch paper. Oracle has a need for some periodic disk space. In the case of a very active database, there might be more than one temp tablespace.

Tools Tablespace

The tools tablespace is where the database objects needed to support tools such as Oracle Reports, are stored. Like any Oracle application, Oracle Reports needs to store tables in the database. Most DBAs place the tables needed to support tools in this tablespace.

Users Tablespace

The users' tablespace holds users' personal information.

Data and Index Tablespaces

Indexes are special database objects that enable Oracle to quickly find data stored within the table. In Oracle, looking at every row in a database is called a full table scan. Using an index search is called an index scan.

Rollback Tablespace

All Oracle databases need a location to store undo information. This tablespace, which holds rollback segments, is typically called the rollback tablespace.

5.2.4 REDO LOGS

In addition to the datafiles associated with the tablespace, Oracle has other operating system files associated with it called ***online redo logs***. Another common term for redo logs is ***transaction logs***. *These are special operating system files in which Oracle records all changes or transactions that happen to the database*. As changes are made to the database, these changes occur in memory (main memory or RAM). Oracle handles these changes in memory for performance reasons. A disk input/output is one thousand times slower than an action in memory. Since a copy of all transactions is always recorded to the online redo logs, Oracle can take its time recording back to the original datafile (in secondary memory) the changes that occurred in (main) memory. Eventually, the final value of the changed data is recorded back to the physical datafile. Since all the transactions are recorded in the online redo logs, the database is always able to recover itself using these logs. It is a requirement that every Oracle database have at least two online redo logs.

The Working of Redo Logs

Redo logs work in a circular fashion. If a database has two online redo logs, logA and logB, as transactions create, delete and modify the data in the database; they are recorded first in logA. When logA is filled up, a log switch occurs. All new transactions are then recorded in logB. When logB fills up, another log switch occurs. Now all transactions are recorded in logA again.

An aspect worthy of attention is that, since redo logs are used in a cyclic fashion, when Oracle reuses logA, the transaction information in logA is overwritten.

The Oracle database runs in either ARCHIVELOG or NOARCHIVELOG mode; these have a direct correlation to the online redo logs.

ARCHIVELOG Mode: Full Recoverability

When a database is running in ARCHIVELOG mode, all transaction redo logs are saved. There exists a copy of every transaction that runs in the database, so even though the redo logs work in a circular fashion, a copy of the redo log is made before it is overwritten. In the event that the database needs to switch before the copy has been made, Oracle will freeze up until this action has completed. Oracle will not allow the old transaction log to be overwritten until it has a copy of it. This way, the database is able to protect the data against all types of failures and hence is the safest mode to run the database in.

NONARCHIVELOG Mode

When a database is running in NONARCHIVELOG mode (the default), old redo logs are not saved. Because no transaction logs are saved, recovery is not possible.

5.2.5 CONTROL FILES

Every database must have at least one control file; though it is highly recommended to have more. It is good to have two or more control files in case one is damaged while the database operates. If there is a single control file, without an additional control file, keeping the database accessible to the users will not be possible.

A control file is a very small file that contains key information about all the files associated with an Oracle database. Control files maintain the integrity of the database and help to identify which redo logs are needed in the recovery process. Before the database is allowed to begin running, it goes to the control file to determine if the database is in acceptable shape. For instance, if a datafile is missing or a particular file has been altered while the database was not using it, then the control file informs the database that it has failed inspection. If Oracle determines that the database is not in acceptable shape, it will not permit the database to run.

Whenever a database checkpoint (explained in the next section) occurs or there is a change to the structure of the database, the control file is updated. It is advisable to have at least two control files stored on different disks.

5.2.6 PROGRAMS

These can be called processes. Every time an application accesses a database, it communicates with Oracle via a process. There are two types of Oracle processes: the user and server processes.

User (Client) Process

User processes request information from the server processes. Oracle Forms, Oracle Reports are examples of user processes. These are common tools any user of the data within the database uses to communicate with the database.

Server Processes

Server processes take requests from user processes and communicate with the database. Through this communication, user processes work with the data in the database. A way to think of the client/server process is to imagine a customer in a restaurant. The customer communicates to the waiter who takes his order. That person communicates the request to the kitchen. The kitchen staff's job is to prepare the food, let the waiter know when it is ready, and stock inventory. The waiter then delivers the meal back to the customer. In this analogy, the waiter represents the client process and the kitchen staff represents the server processes.

5.2.7 DATABASE SUPPORT PROCESSES

Server processes take requests from user / client processes; they communicate with the database on behalf of user processes. There is a special set of server processes that help the database operate.

Database Writer (DBWR)

The database writer is a mandatory process that writes changed data blocks back to the database files. It is one of the only two processes that are allowed to write to the datafiles that make up the Oracle database. On certain operating systems, Oracle allows multiple database writers for performance reasons.

Checkpoint (CKPT)

Checkpoint is an optional process. When users are working with an Oracle database, they make requests to look at data. The data is read from the database files and put into an area of memory where users can look at it. Some of these users eventually make changes to the data that must be recorded back onto the original datafiles. When the redo logs switch, a checkpoint occurs. When this switch happens, Oracle writes any dirty data blocks' information back to disk. In addition, it notifies the control file of the redo log switch.

The log writer normally performs these tasks. For performance reasons, the database administrator can make changes to the database to enable the checkpoint process. Its sole job is to take the checkpoint responsibility away from the log writer.

Log Writer (LGWR)

The log writer is a mandatory process that writes redo entries to the redo logs. Since a copy of every transaction is written in the redo log, Oracle does not have to spend its resources constantly writing data changes back to the datafiles immediately. This results in improved performance. It is also the only process in an Oracle database that reads the redo logs.

System Monitor (SMON)

System monitor is a mandatory process that performs any recovery that is needed at start up.

Process Monitor (PMON)

Process monitor is a mandatory process that performs recovery for a failed user of the database. It assumes the identity of the failed user, releasing all the database resources that user was holding and it rolls back the aborted transaction.

Archiver (ARCH)

Archiver is an optional process. The redo logs are written in a sequential manner. When a log fills up, there is a log switch to the next available redo log. If the database is running in ARCHIVELOG mode, the database makes a copy of the redo log. This is done so that when the database switches back to this redo log, there is a copy of the contents of this file for recovery purposes. This is the job of the archiver process. It makes a copy of the file.

5.3 MEMORY STRUCTURES

In this section, the manner in which the client and server processes communicate with each other and among themselves through memory structures will be discussed.

Oracle uses two types of memory structures: the system global area (SGA) and the program global area (PGA).

5.3.1 SYSTEM GLOBAL AREA (SGA)

SGA is a place in memory where the Oracle database stores pertinent information about itself. This memory structure is then accessible to all the user processes and server

processes. This mechanism is advantageous since memory is the quickest and most efficient way to allow processes to communicate.

Since SGA is the mechanism by which the various client and server processes communicate, it is important to understand its various components. The SGA is broken down into the following key components.

Data Buffer Cache

The data buffer cache is where Oracle stores the most recently used blocks of database data. This is the data cache. When information is put into the database, it is stored in data blocks in secondary memory (hard disk drive). The data buffer cache is an area of main memory in which Oracle places these data blocks so that a user process can look at them. Before any user process can look at a piece of data, the data must first reside in the data buffer cache. There is a physical limit on the size of the data buffer cache. Thus, as Oracle fills it up, it leaves the hottest blocks in the cache and moves out the cold blocks. This is done using the Least Recently Used Algorithm.

If a client process needs information that is not in the cache, the database goes out to the physical disk drive, reads the needed data blocks, then places them in the data buffer cache.

Dictionary Cache

A dictionary cache contains rows out of the data dictionary. The data dictionary contains all the information Oracle needs to manage itself, such as what users have access to the database, what database objects they own and where those objects are located.

Redo Log Buffer

Before any transaction can be recorded into the redo log (the file in secondary memory), it must first be recorded in the redo log buffer (in main memory). This is an area of memory set aside for this event. Then, the database periodically flushes this buffer to the online redo logs.

Shared SQL Pool

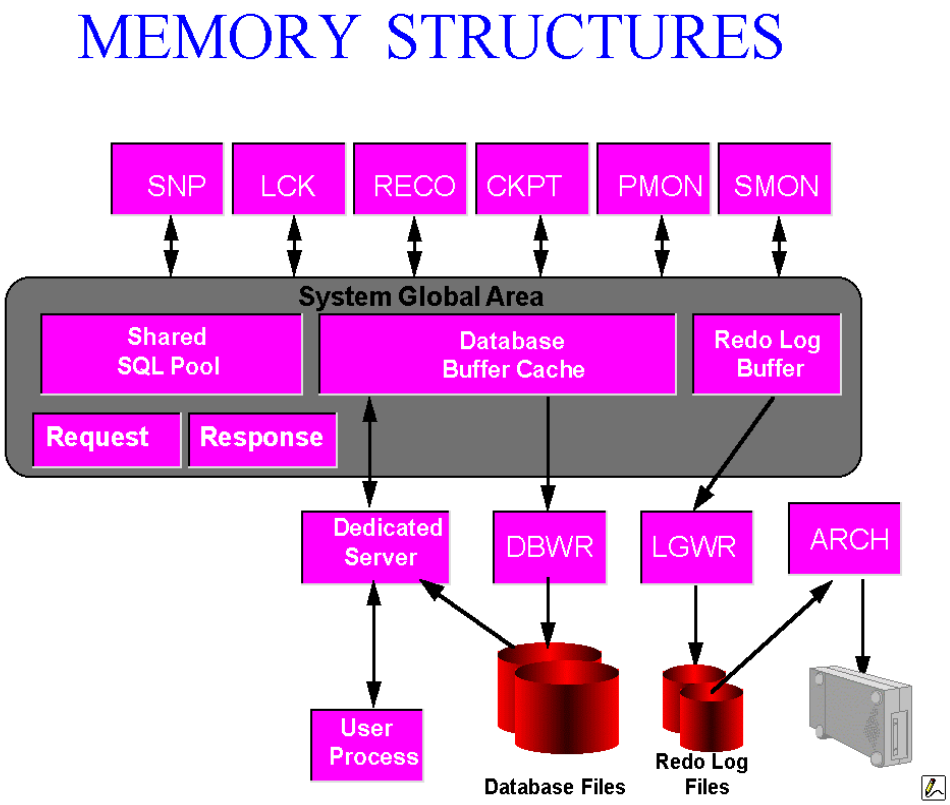
This is where all programs are stored. Programs within an Oracle database are based on a standard language called SQL. This cache contains all the parsed SQL statements that are ready to run.

In short, the SGA is the place in memory where information is placed so that client and server processes can access it. It is broken up into the data cache, the redo log cache, the dictionary cache and the shared SQL cache.

5.3.2 PROGRAM GLOBAL AREA (PGA)

PGA is an area of memory that is used by a single Oracle process. The program global area is not shared. It contains data and control information for a single process. It contains information such as process session variables and internal arrays.

The diagram explaining the memory structures follows:



5.4 AN ORACLE INSTANCE

An Oracle instance is a set of Oracle server processes, that have their own system global area, and a set of database files associated with them. If a computer has two databases on it and each of these has its own SGA and a separate set of Oracle server processes, then there are two instances of the database. Each instance is identified by what is known as the SID – system identifier. This is set by the variable called **ORACLE_SID** on most UNIX computers. Say a database called trg exists. The processes related to the trg database would be named ora_trg_dbwr, ora_trg_pmon, ora_trg_smon and ora_trg_lgwr.

5.5 DATABASE OBJECTS

This section deals with the major database objects that will be encountered when working with Oracle. Each of these objects has a specific purpose.

5.5.1 TERMINOLOGY

Table

A table is a database object that holds data. Information about every table is stored in the data dictionary; with this information, Oracle allows the maintenance of data residing in the table.

View

A view allows the customized selection of fields from one or more tables based on user-defined conditions and uses a SQL query that is stored in the database.

Index

Index entries for a table allow Oracle rapid access to the data in the tables. An index is created on one or more of columns in the table. An index is essentially a mechanism by which Oracle correlates data values with RowID values – and thus with physical locations of data, thus making data access faster.

Synonym

A synonym is an alternate name for an object in the database.

Grants

Grants are privileges given out by owners of objects, allowing other users to work with their data.

Data Dictionary.

The data dictionary is maintained by Oracle containing information relevant to the tables that reside in the database.

Role

A role is a group of privileges that are collected together and granted to users. Once privileges are granted to a role, a user inherits the role's privileges by becoming a member of that role. This way, instead of updating every user's account on an individual basis, the role can be managed.

5.5.2 TERMINOLOGY UNLEASHED

Some of the terminology that was introduced in the previous section is explained in detail.

TABLES

A table is a database object that holds data. The data dictionary holds information about every table. Oracle uses its data dictionary to ensure the correct type of data is placed in Oracle tables. The table is made up of many columns. Each of the columns has a data type associated with it. This data type is the roadmap that Oracle follows so it knows how to correctly manipulate the contents.

VIEWS

A view is a database object that allows the creation of a customized slice of a table or a collection of tables. Unlike a table, a view contains no data, just a SQL query. The data that is retrieved from this query is presented like a table. Like a table, insert, update, delete and select is possible from a view.

Views can provide an additional level of security by allowing users to see information required by them alone. Views help to hide data complexity. This can be achieved by having a view that is a combination of one or more tables and the user will only have to query from this view using simple SQL statements.

INDEXES

An index placed on a table helps retrieve data faster. A well-placed index will make a slow application run faster.

SYNONYMS

A synonym is a database object that allows the creation of alternate names for Oracle tables and views. Synonyms can be used when the true owner or name of the table has to be hidden and when the table needs to be given a name that is less complicated than its original name.

6 STRUCTURED QUERY LANGUAGE

This section discusses SQL, the Structured Query Language.

6.1 TERMINOLOGY

The terminology used in the document has been described below.

DDL

Data Definition Language is the SQL construct used to define data in the database. When data is defined, entries are made in Oracle's data dictionary.

DML

Data Manipulation Language is the SQL construct used to manipulate the data in the database.

Commit

The word commit is used to instruct Oracle to save the data in the buffers to the database.

Query

The action of trying to get information out of an Oracle database using a program such as SQL*Plus is called a query.

Functions

Functions are operations performed on data to alter the data's characteristics.

6.2 TYPES OF SQL STATEMENTS

SQL statements fall into three major categories: Data Definition Language, Data Manipulation Language and Data Control Language.

Data Definition Language

Data Definition Language allows the following tasks:

- Creation of a database object
- Dropping a database object
- Alteration of a database object
- Grant of privileges on a database object
- Revoking of privileges on a database object

When a DDL SQL statement is issued, Oracle commits the current transaction before and after every DDL statement.

Given below are some important DDL statements

SQL COMMAND	PURPOSE
alter procedure	Recompiles a stored procedure
alter table	Adds a column, redefines a column, changes storage allocation
analyze	Gathers performance statistics for objects to be fed to the cost-based optimizer
create table	Creates a table
create index	Creates an index
drop index	Drops an index
drop table	Drops a table from the database
truncate	Deletes all the rows from a table

Data Manipulation Language

Data Manipulation Language allows the user to insert, update, delete and select data in the database. It allows the user to work with the contents of the database.

Given below are some important DML statements:

SQL COMMAND	PURPOSE
insert	Add rows of data to a table
delete	Delete rows of data from a table
update	Change data in a table
select	Retrieve rows of data from a table/view
commit work	Make changes permanent for the current transaction(s)
rollback	Undo all changes since the last commit

Data Control Language

Data Control Language allows the user to control the data stored in the database.

grant	Grants privileges or roles to a user or another role
revoke	Removes privileges from a user or database role

6.3 DATABASE OBJECT NAMING RULES

- The name of the database object must be maximum of 30 characters long. (Except for database names which are limited to 8 characters).
- A name must not contain a question mark.
- A name is not case sensitive.
- A name must begin with an alphabet.
- A name may only contain the characters A-Z, 0-9, _,$ and #.
- A name must not duplicate an Oracle Reserved word.

Datatypes

Data Type	Description	Max Values	Example
char(n)	Fixed length character data of length n bytes. All n bytes are used , where necessary filled with blanks	255 bytes	char(40)
varchar(n)	Currently a synonym for varchar2	2000 bytes	varchar(80)
varchar2(n)	Variable length character string of length n bytes. Only uses the number of bytes actually filled.	2000 bytes	varchar2(80)
number(v,n)	numeric datatype of integer and real values. v = Total no of Places. n = Places after decimal point.	v=38 n=-84 to +127	number(8) number(12,4)
date	Date type for storing date and time.	31.12.4712	date
raw(n)	Datatype for storing binary data.	2000 bytes	raw(1000)
long	Datatype for storing data of variable length.	2 GB	long
long raw	Datatype for storing binary data up to 2GB. Column size is variable	2 GB	long raw

6.4 TABLE CREATION AND ALTERATION

TABLE CREATION

Syntax **CREATE TABLE** tablename

```
( columnname datatype_and_size [constraint]
 [,columnname datatype_and_size [constraint]].....
                );
```

Example

```
CREATE  TABLE  dept
(
deptno NUMBER(2) NOT NULL,
dname CHAR(14), loc VARCHAR2(13)
);
```

Constraints

Constraints are rules which the database enforces to maintain the integrity of the data.

NULL | NOT NULL specifies if a column may or may not support NULL values.

NULL means absence of value. Specifically, it doesn't mean zero or blank.

UNIQUE forces values of a column to be unique.

PRIMARY KEY identifies a column as a primary key.

A primary key is a combination of one or more columns of the table which uniquely identifies each row of the table.

FOREIGN KEY (column…) REFERENCES user.table (column…) identifies the column(s) as a foreign key of user.table(column(s)).

A foreign key is a combination of columns with values based on the primary key values from another table (called the parent table). The foreign key constraint specifies that the values of the foreign key correspond to actual values of the primary key in the parent table.

CHECK condition specifies a condition that the column must satisfy for the row to exist in the table.

ALTERING A TABLE

Tables can be altered in one of the three ways : by adding a column to an existing table, by changing a column's definition or by dropping a column.

Syntax **ALTER TABLE** tablename ([ADD][MODIFY]) (column definitions);

Example ALTER TABLE dept ADD (tot_space NUMBER(3,1));

Note :

One cannot use the MODIFY option

- to reduce the width of a column unless all the columns are empty.
- change the datatype unless the column is empty.
- to make it NOT NULL unless all rows have values in that column.

The **DESC** tablename can be used to view the structure of the table.

Example DESC dept;

6.5 SEQUENCE GENERATOR

These are used to generate sequential numbers and are helpful in multi-user environments to generate and return unique sequential numbers. Sequence generators are objects that are stored as part of the database.

Syntax **CREATE SEQUENCE** <seq_name>

[INCREMENT BY n]

[START WITH n]

[NOMAXVALUE]

[NOCYCLE]

[CACHE 20];

Example CREATE SEQUENCE delseq ;

To use the contents of a sequence **seq_name.CURRVAL** or **seq_name.NEXTVAL** can be used to access the current value and next incremental value of the sequence generator.

Example INSERT INTO DELORD (ordno,dordno)

VALUES (delseq.NEXTVAL, po.vordno);

6.6 SYNONYMS

A synonym is an alias for database object. Synonyms do not require additional storage other than its definition in the data dictionary.

Generally they are used for security reasons to mask the owner and name of the object and to simplify SQL statements.

CREATE PUBLIC SYNONYM pucust FOR scott.customer;

6.7 THE SELECT STATEMENT

This command is used to query the contents of a table.

WHOLE TABLE

Syntax SELECT * FROM tablename;

Example SELECT * from SOL;

SELECT LIST

Syntax SELECT col1,col2,col3.... FROM tablename [WHERE condition];

Example SELECT sol_id,sol_desc FROM sol WHERE sol_id = '27';

SELECT sol_id,sol_desc FROM sol;

6.8 COMPARISON OPERATORS

The following are the comparison operators:

Equal	=
Greater than	>
Less than	<
Greater or Equal to	>=
Less or Equal to	<=
Not Equal	<> or != or ^=

Between	BETWEEN ..... AND .....
Partial Equality	LIKE 'pattern' (_ underscore for single character, % any sequence of zero or more characters)
Equal to one item in list	IN (list_of_values)
Negation	NOT
To compare with NULL	IS NULL/ IS NOT NULL

Examples

SELECT sol_id,sol_desc FROM sol WHERE sol_id IN('027','027A');

SELECT sol_desc FROM sol WHERE city_code LIKE '%BAD';

SELECT cust_name FROM cmg WHERE cust_emp_id BETWEEN 200 AND 400;

SELECT dname FROM dept WHERE loc IS NULL;

SELECT deptno,loc FROM dept WHERE dname LIKE '__HI';

Compound conditions can be specified in the where clause by using the operators AND and OR. Also, note that the user can select expressions from a table and provide aliases for viewing the data.

Example

SELECT * FROM sol

WHERE sol_id = '027' OR state_code LIKE 'M%';

Alias

Alias is a temporary name assigned to a table or a column within a SQL statement .You can use the AS keyword to separate the column definition from its alias.

SELECT cust_name AS cname ,cust_id AS custno from cmg where cust_name LIKE 'AN%';

Here cname ,custno are the aliases used.

To arrange the data in sorted order (of a specified column) for viewing, use the **ORDER BY** clause.

Example SELECT * FROM sol ORDER BY sol_id;

6.9 JOINS

Joins are used to combine columns of different tables. There are different types of joins

EQUI JOINS (RELATIONAL ALGEBRA EQUI JOIN)

When two tables are joined together using equality of values in one or more columns they make an equi join. Table prefixes are used to prevent ambiguity and the where clause is used to specify which columns are to be joined.

Example: List the name of the department with employee information.

```
SELECT empno,ename,job,emp.deptno,dname FROM emp,dept
WHERE emp.deptno = dept.deptno;
```

CARTESIAN JOINS (RELATIONAL ALGEBRA CARTESIAN PRODUCT)

When no '**WHERE**' clause is specified, each row of one table matches every row of the other table.

Example SELECT ename,deptno , loc FROM emp,dept;

OUTER JOIN

An **outer join** extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

Example

Display the list of employees working in each department, also list the departments which do not contain employees.

SELECT ename, dept.deptno, loc FROM emp RIGHT JOIN dept

WHERE emp.deptno = dept.deptno;

Use RIGHT JOIN to get all rows of the table on the right side of the JOIN operator, else use LEFT JOIN.

SELF JOIN

When one row of a table is joined with another of the same table the user is performing a SELF JOIN.

Example: List the name of each employee along with his/her manager from emp table.

In this case since both the employee and his/her manager are parts of the same table

a self join is made use of.

Example SELECT worker.ename, manager.ename

FROM emp worker, emp manager □ *alias* ←

WHERE worker.mgr = manager.empno;

Any other join condition involving an operator other than the equivalency (=) is a **NON EQUI JOIN**.

6.10 SET OPERATORS

Set operators combine 2 or more queries into a single result. The data type of the corresponding columns must be the same.

UNION

Rows of the first query plus the rows of the second query, less duplicate rows.

Example List the employees from accounts and research tables whose salary is greater than 2000.

```
SELECT ename,sal FROM account WHERE  sal > 2000

UNION

SELECT ename , sal FROM research WHERE sal > 2000;
```

INTERSECTION

Returns the rows that are common to both queries.

Example List the jobs that are common in research,accounts and sales department.

```
SELECT job FROM research

INTERSECT

SELECT job FROM accounts

INTERSECT

SELECT job FROM sales;
```

MINUS

Returns rows that are unique to the first query.

Example List the jobs which are done in accounts only and not in sales.

```
SELECT job FROM account

MINUS

SELECT job FROM sales;
```

6.11 GROUPING FUNCTIONS

Group functions are utilized for summarizing information from a set of rows.

The grouping functions are

a) SUM([DISTINCT|ALL] column) returns the sum of values of **column**.

Example SELECT SUM(sal) Total FROM emp;

Output Total

29025

b) **MAX([DISTINCT|ALL] column)** returns the maximum value of **column**.

Example SELECT MAX(sal) "Maximum" FROM emp;

Output Maximum

2000

c) **MIN([DISTINCT|ALL] column)** returns the minimum value of **column**.

Example SELECT MIN(sal) "Minimum" FROM emp;

Output Minimum

20

d) **COUNT({* | [DISTINCT |ALL] column})** returns the number of rows in the output of a query in which **'column'** is non-null.

Examples

COUNT (DISTINCT deptno) returns the number of unique departments.

COUNT (*) returns the total number of rows including NULL and duplicates.

COUNT (deptno) returns the number of departments minus NULLs.

e) **AVG ([DISTINCT|ALL] column*)*** returns the average value of **column**.

Example SELECT AVG (sal) "Average" FROM emp;

Output Average

2074.44

6.12 GROUP BY CLAUSE

GROUP BY clause is used to group the selected rows based on the value of the specified column for each row and return a single row of summary information for each group. This column induces a control break using the specified field as the control field.

Example	SELECT sol_id, COUNT (*) FROM gam GROUP BY sol_id;

The above query will give a summary of the number of accounts for a particular SOL.

The selected columns must form the list of columns specified in the GROUP BY clause or should be grouping functions.

6.13 HAVING CLAUSE

The HAVING CLAUSE filters the rows returned by the GROUP BY clause.

Example	SELECT sol_id, COUNT (*) FROM gam GROUP BY sol_id HAVING COUNT (*) > 1000;

This query returns those SOLs that have more than 1000 Accounts.

6.14 NESTED QUERY

In this type of query, the result of one query is dynamically substituted in the condition of another query. ORACLE supports infinite level of nesting.

The general syntax for a nested or sub query is

SELECT col1[,col2,....]

FROM table1

WHERE column operator (SELECT column FROM table2 WHERE condition);

Example

SELECT name,target FROM salesrep

WHERE target > (SELECT target FROM salesrep WHERE title = 'VP SALES');

6.15 CO RELATED SUBQUERIES

SQL sub queries that perform the subquery repeatedly , once for each row of the outer query are called CO RELATED SUB QUERIES.

6.16 VIEWS

A VIEW is a virtual window through which a user can change information of the base table.

The properties of a view are,

- It looks like a table. However, it does not exist as such.
- Its data is derived from the table(s).
- NO copy of the data is maintained.
- It is used to provide simplicity, by presenting only what is needed.
- It provides security, by preventing unauthorized users from seeing the columns of a table.

The user cannot perform inserts, updates or deletes for a view if the view query contains,

- joins
- set operators
- group functions
- GROUP BY, CONNECT BY or START WITH clauses
- the DISTINCT operator

Example CREATE VIEW clerk (id_number,person,department,position) *AS* SELECT empno,ename,deptno,job

FROM emp

WHERE job = 'CLERK'

WITH CHECK OPTION;

The **WITH CHECK OPTION** specifies that inserts and updates performed through the view must result in rows that the view query can select. In the above example any row other than the one having job = 'CLERK' cannot be inserted into the table.

6.17 INDEXES AND CLUSTERS

Indexes and Clusters are optional structures that provide a faster access to data and efficient utilization of storage space.

Indexes and Clusters are transparent to users and applications.

INDEXES

Unique and Non Unique Indexes

Unique indexes guarantee that no two rows of a table have duplicate values in the columns that define the index.

Non Unique indexes do not impose this restriction.

Composite Indexes

A Composite index is one that is created on multiple columns in a table.

When to use Indexes?

Create indexes on tables that are queried for a small percentage of rows in a table.

Which columns to select for indexing?

- Columns that are frequently accessed in WHERE clauses.
- Columns that frequently used to join tables.
- Do not index columns that are frequently modified.

Example

```
CREATE INDEX  ordcust ON  orders(custcode,ordno)

TABLESPACE  users

STORAGE(INITIAL 20K, NEXT 20K, PCTINCREASE 75)

PCTFREE   0;
```

CLUSTERS

A Cluster is a group of tables that share the same data blocks, because they share the same columns and are often used together.

Advantages

Disk I/O is reduced and access time improves for joins of clustered tables.

Each cluster key is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value.

Example

To create tables emp and dept clustered around department number.

```
CREATE CLUSTER PERSONNEL(DEPT_NO NUMBER(2));

CREATE TABLE EMP(
        EMPNO NUMBER(4) PRIMARY KEY,
        DEPTNO NUMBER(2) NOT NULL,
        ENAME  VARCHAR2(15),
        SAL NUMBER(10,2),
        COMM NUMBER(5))
        CLUSTER PERSONNEL(DEPTNO);
CREATE TABLE DEPT(
        DEPTNO NUMBER(2) PRIMARY KEY,
        DNAME  VARCHAR2(20),
        LOC  VARCHAR2(15))
        CLUSTER PERSONNEL(DEPTNO);
CREATE INDEX IDX_PERSONNEL  ON CLUSTER PERSONNEL;
```

6.18 DATA MANIPULATION COMMANDS

Some of the Data Manipulation Commands have been discussed below:

INSERT

This statement is used to insert values to the table.

Syntax INSERT INTO tablename

VALUES (list_of _values_for_the_columns);

Example INSERT INTO emp

VALUES(8762,'MANOHAR','15-AUG-96','Asst Mgr',14500);

Note:

Values have to be inserted in the order of the columns itself.

If any other format is given as input the names of the columns have to be specified.

Insert will automatically check for not NULL and UNIQUE constraints , if applicable.

All data types to be included in quotes(`) , except numeric data types.

DELETE

Using the DELETE statement row can be deleted from a table.

Syntax DELETE FROM tablename WHERE condition_is_true;

Example DELETE FROM dept WHERE deptno = 67;

To delete all rows,

DELETE FROM dept;

UPDATE

This statement is used change existing values in a specified table.

Syntax UPDATE tablename

SET column = value|expression(query)

WHERE condition_is_true;

Example UPDATE emp SET sal = sal + 1000 WHERE deptno = 20;

7 FUNCTIONS

Functions allow the user to

- combine values to create new ones.
- change value formats.
- modify values.

7.1 CHARACTER FUNCTIONS

Listed below are some character functions:

INITCAP(ename) capitalizes first letter of each word.

LENGTH(ename) computes the number of characters in the string.

SUBSTR(ename,s,n) extracts n characters starting from s character.

LOWER converts all characters to lower case.

UPPER converts all characters to upper case.

LEAST returns the value from a series of arguments that comes first alphabetically.

GREATEST returns the value from a series of arguments that comes last alphabetically.

7.2 NUMERIC FUNCTIONS

Listed below are some numeric functions:

LEAST(a1,a2.....) returns the least of the arguments.

GREATEST(a1,a2...) returns the largest value from the arguments.

ABS returns the absolute value from the arguments

ROUND rounds the argument to the closest integer.

SIGN returns -1 if the number is negative

0 if the number is zero

+1 if the number is positive.

TRUNC Truncates the argument to the closest integer.

7.3 DATE FUNCTIONS

Listed below are some date functions:

ADD_MONTHS(hiredate,5)

adds 5 months to the hiredate.

MONTHS_BETWEEN(sysdate,hiredate)

calculates the number of months between sysdate and hiredate.

NEXT_DAY(hiredate, 'Friday')

finds the date of the Friday after the employee was hired.

TO_CHAR(date, date_picture)

date will be displayed as a character string according to the format of date_picture.

TO_DATE(character string,date_picture)

character will be converted to an ORACLE date according to the format of the date picture.

date_picture

DAYS		
dd	number	12
dy	abbreviated	fri
day	spelled out	Friday
ddspth `	spelled out, ordinal	twelfth
MONTH		
mm	number	03
mon	abbreviated	mar
month	spelled out	march
YEAR		
yy	year	96
yyyy	year and century	1996

When the user capitalizes the picture , output reflects the same.

7.4 GENERAL FUNCTIONS

Listed below are some general functions:

NVL(arg1 , arg2) If arg1 is NULL then arg2 is returned.

DECODE(grade,'A',1000,'B',2000,0)

If grade has value 'A' , then 1000 else if value 'B' then 2000, default then 0.

8 FLASHBACK

This document introduced the fundamentals of database concepts, talked about relational database management systems, and has dealt with Oracle in particular. The memory structures in Oracle were also discussed apart from the Structured Query Language commands and built-in functions.

QUIZ

TOPIC : ORACLE FUNDAMENTALS

Sl.No	Question	True	False
1	Database is a collection of related data		
2	A DBMS without relational capabilities is called an RDBMS		
3	NUMBER(n) is a valid data type in RDBMS		
4	"DATE" is an invalid data type		
5	The data type for a column can be changed at any time		
6	Table is a physical object in database		
7	More than one databases can be created in a single system		
8	View has a copy of a table data.		
9	SQL*Plus is a product of Oracle		
10	SQL commands cannot be saved in a file		
11	Datafiles are the physical components of a database		
12	SGA is allocated when the database is brought down		
13	PMON is a oracle background process		
14	import/export are oracle utilities		
15	Oracle memory structure contains datafiles		
16	A table can have many rows		
17	Dual table is a table with only one row		
18	Bringing up the system will automatically bring up the database		
19	Oracle works on flat file concept		
20	Each database is identified by using database name		

EXERCISES

TOPIC : ORACLE FUNDAMENTALS

Sl.No	Question
1.	Expand the following: (A) RDBMS (B) SGA (C) PL/SQL
2.	Who are the two users created by default whenever a database is created?
3.	What are the advantages of RDBMS?
4.	What are the three major classifications of SQL statements?
5.	How do you select all the entries from a table?
6.	What is meant by self join?
7.	How do you display the maximum account balance by selecting it from the table?
8.	Write a query to display the account no and account balance by selecting from the table General_Account_Master.
9.	What does "set pause on" do?
10.	How do you save the result of a query into a file?
11.	Name some of the database objects.
12.	How to bring up and shutdown the database?
13.	How do you close the database by aborting an instance?
14.	Why do you open the database?
15.	Write in brief about the roles of Database Administrator.