KARPAGAM ACADEMY OF HIGHER EDUCATION

Coimbatore-641 021

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF CS, CA & IT

SUBJECT NAME: ORACLE(SQL/PL-SQL) **SEMESTER : V**

SUBJECT CODE: 16CSU504A **CLASS: III- B. Sc (CS)**

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**

COURSE OBJECTIVE

The Objective of Relational Database Management System including relational, object-relational, and object-oriented systems, SQL standards, algebraic query languages, integrity constraints, triggers, functional dependencies, and normal forms. Other topics include tuning database transactions, security from the application perspective, and data warehousing.

COURSE OUTCOME

- Understand the role and nature of relational database management systems (RDBMS) in today's IT environment.
- Translate written business requirements into conceptual entity-relationship data models.
- Convert conceptual data models into relational database schemas using the SQL Data Definition Language (DDL).
- Query and manipulate databases using the SQL Data Manipulation Language (DML).

UNIT-I

Introduction to Oracle as RDBMS SQL Vs. SQL * Plus: SQL Commands and Data types, Operators and Expressions, Introduction to SQL * Plus.

UNIT-II

Managing Tables and Data: Creating and Altering Tables (Including constraints) ,Data Manipulation Command like Insert, update, delete, SELECT statement with WHERE, GROUP BY and HAVING, ORDER BY, DISTINCT, Special operator like IN, ANY, ALL BETWEEN, EXISTS, LIKE, Join, Built in functions

UNIT-III

Other Database Objects - View, Synonyms, Index

UNIT-IV

Transaction Control Statements - Commit, Rollback, Savepoint

UNIT-V

Introduction to PL/SQL SQL v/s PL/SQL, PL/SQL Block Structure, Language construct of PL/SQL (Variables, Basic and Composite Data type, Conditions looping etc.) TYPE and % ROWTYPE , Using Cursor (Implicit, Explicit)

Suggested Readings

Steven Feuerstein., & Bill Pribyl. (2014). Oracle PL/SQL Programming (6th ed.). O'Reilly Media.

Reference Book

Scott Urman, Ron Hardman & Michael McLaughlin. (2004). Oracle Database 10g PL/SQL Programming (4th ed.). Oracle Press.

Websites

W1: https://www.tutorialspoint.com/plsql/
W2: http://plsql-tutorial.com/

Journal

M. M. Astrahan, M.W. Blasgen et. al. System R: relational approach to database management, ACM Transactions on Database Systems, Vol1, Issue 2, June 1976 Pages 97-137.

ESE MARKS ALLOCATION

1.	**Section A** 20 X1 = 20 (Online Examination)	20
2.	**Section B** 5X2 = 10	10
3	**Section C** 5X6 = 30 (Either 'A' or 'B' Choice)	30
	Total	**60**

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE – 641 021

DEPARTMENT OF COMPUTER SCIENCE

LECTURE PLAN

STAFF NAME: Dr. T. GENISH
SUB.CODE : 16CSU504A
CLASS : III B.SC CS

SUBJECT NAME: ORACLE (SQL/PL-SQL)
SEMESTER: V

S.No	Lecture Duration (Hr)	Topics Covered	Reference Materials
Unit I			
1.	1	**Introduction to Oracle as RDBMS** SQL Vs. SQL * Plus	T1: 3-5, T1: 25-28,J1
2.	1	SQL Commands	W1
3.	1	Data types	T1: 167-172
4.	1	Operators and Expressions	T1: 259-261
5.	1	Introduction to SQL * Plus.	T1: 25-28
6.	1	Recapitulation and Possible Questions Discussion	
Total No of hours for Unit 1:6			
Unit II			
1.	1	**Managing Tables and Data:** Creating and Altering Tables (Including constraints)	T1: 440-446
2.	1	Data Manipulation Command: Insert,	
3.	1	Data Manipulation Command: Update, delete,	
4.	1	SELECT statement with WHERE, GROUP BY	W1, W2, T1: 341-345
5.	1	SELECT statement with HAVING, ORDER BY, DISTINCT	
6.	1	Special operators: IN, ANY, ALL	
7.	1	Special operators: BETWEEN, EXISTS, LIKE,	
8.	1	Join, Built in functions	
9.	1	Recapitulation and Possible Questions Discussion	
Total No of hours for Unit 2:9			
Unit -III			
1.	2	**Other Database Objects** - View	W2
2.	2	Synonyms	T1: 43
3.	2	Index	W1

4.	1	Recapitulation and Possible Questions Discussion	
Total No of hours for Unit 3:7			
Unit - IV			
1.	2	**Transaction Control Statements -** Commit	T1: 450,452 W1,W2
2.	1	Rollback	
3.	2	Savepoint	
4.	1	Recapitulation and Possible Questions Discussion	
Total No of hours for Unit 4:6			
Unit – V			
1.	1	**Introduction to PL/SQL** SQL v/s PL/SQL	R1: 3-23
2.	1	PL/SQL Block Structure	
3.	2	Language construct of PL/SQL (Variables, Basic and Composite Data type, Conditions looping etc.)	
4.	1	TYPE and % ROWTYPE	T1:173-179
5.	1	Using Cursor (Implicit, Explicit)	
6.	1	Recapitulation and Possible Questions Discussion	
7.	1	Previous year end-semester question paper discussion	
Total No of hours for Unit 5:08			
Total No. Of Hours Allocated: 36			

SUGGESTED READINGS:

1. Steven Feuerstein., & Bill Pribyl. (2014). Oracle PL/SQL Programming (6th ed.) O'Reilly Media.

REFERENCE BOOK

1. Scott Urman, Ron Hardman & Michael McLaughlin. (2004). Oracle Database 10g PL/SQL Programming (4th ed.). Oracle Press.

WEB SITES:

1. W1: http://www.plsqltutorial.com/
2. W2: https://www.javatpoint.com/pl-sql-tutorial

JOURNALS:

1. Journal: Tong-SengQuah, Mie MieThet Thwin, Prediction of software development faults in PL/SQL files using neural network models, Information and Software Technology, Vol 46, Pages 519-523, 2004.

UNIT I

SYLLABUS

Introduction to Oracle as RDBMS SQL Vs. SQL * Plus: SQL Commands and Data types, Operators and Expressions, Introduction to SQL * Plus.

Introduction to Oracle as RDBMS

A database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

ORACLE is a fourth generation relational database management system. In general, a database management system (DBMS) must be able to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance to the users of the database. A DBMS must also be secure from unauthorized access and provide efficient solutions for failure recovery. The ORACLE Server provides efficient and effective solutions for the major database features.

ORACLE consists of many tools that allow you to create an application with ease and flexibility. You must determine how to implement your requirements using the features available in ORACLE, along with its tools. The features and tools that you choose to use to implement your application can significantly affect the performance of your application.

Several of the more useful features available to ORACLE application developers are integrity constraints, stored procedures and packages, database triggers, cost-based optimizer, shared SQL, locking and sequences.

In Oracle database management, PL/SQL is a procedural language extension to Structured Query Language (SQL). The purpose of PL/SQL is to combine database language and procedural programming language. The basic unit in PL/SQL is called a

block, which is made up of three parts: a declarative part, an executable part, and an exception-building part.

SQL Vs. SQL * Plus

The scope of SQL includes data insert, query, update and delete.

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 iSQL* 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 is a language.
While **SQL*Plus** is a tool.

SQL is the query language used for communication with **Oracle server** to access and modify the data.
SQL* Plus is a command line tool with which you can send SQL queries to the server. Also, it can help you format the query result.

SQL is a language which is invented by **IBM**.
SQL * Plus is a tool to use SQL language for a database from Oracle corporation.

SQL can be simply used to ask queries, i.e. it involves **DML**, **DDL** and **DCL**.
SQL * Plus is command line tool which doesn't involve DML, DDL and DCL.

In SQL, there is no continuation character..
Whereas, in SQL * Plus there is a continuation character.

Keywords cannot be abbreviated in SQL.
But keywords can be abbreviated in SQL*Plus.

SQL uses functions to manipulate the data.
SQL * plus uses commands to manipulate the data.

SQL Commands

SQL, **S**tructured **Q**uery **L**anguage, is a programming language designed to manage data stored in relational databases. SQL operates through simple, declarative statements. This

keeps data accurate and secure, and it helps maintain the integrity of databases, regardless of size.

DML

DML is abbreviation of **Data Manipulation Language**. It is used to retrieve, store, modify, delete, insert and update data in database.

Examples: SELECT, UPDATE, INSERT statements

DDL

DDL is abbreviation of **Data Definition Language**. It is used to create and modify the structure of database objects in database.

Examples: CREATE, ALTER, DROP statements

DCL

DCL is abbreviation of **Data Control Language**. It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.

Examples: GRANT, REVOKE statements

TCL

TCL is abbreviation of **Transactional Control Language**. It is used to manage different transactions occurring within a database.

Examples: COMMIT, ROLLBACK statements

ALTER TABLE

```
ALTER TABLE table_name
ADD column_name datatype;
```

ALTER TABLE lets you add columns to a table in a database.

AND

```
SELECT column_name(s)
FROM table_name
WHERE column_1 = value_1
  AND column_2 = value_2;
```

AND is an operator that combines two conditions. Both conditions must be true for the row to be included in the result set.

AS

```
SELECT column_name AS 'Alias'
FROM table_name;
```

AS is a keyword in SQL that allows you to rename a column or table using an *alias*.

AVG()

```
SELECT AVG(column_name)
FROM table_name;
```

AVG() is an aggregate function that returns the average value for a numeric column.

BETWEEN

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value_1 AND value_2;
```

The BETWEEN operator is used to filter the result set within a certain range. The values can be numbers, text or dates.

CASE

```
SELECT column_name,
  CASE
    WHEN condition THEN 'Result_1'
    WHEN condition THEN 'Result_2'
    ELSE 'Result_3'
  END
FROM table_name;
```

CASE statements are used to create different outputs (usually in the SELECT statement). It is SQL's way of handling if-then logic.

COUNT()

```
SELECT COUNT(column_name)
```

```
FROM table_name;
```

COUNT() is a function that takes the name of a column as an argument and counts the number of rows where the column is not NULL.

CREATE TABLE

```
CREATE TABLE table_name (
  column_1 datatype,
  column_2 datatype,
  column_3 datatype
);
```

CREATE TABLE creates a new table in the database. It allows you to specify the name of the table and the name of each column in the table.

DELETE

```
DELETE FROM table_name
WHERE some_column = some_value;
```

DELETE statements are used to remove rows from a table.

GROUP BY

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name;
```

GROUP BY is a clause in SQL that is only used with aggregate functions. It is used in collaboration with the SELECT statement to arrange identical data into groups.

HAVING

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > value;
```

HAVING was added to SQL because the WHERE keyword could not be used with aggregate functions.

INNER JOIN

```
SELECT column_name(s)
FROM table_1
JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

An inner join will combine rows from different tables if the *join condition* is true.

INSERT

```
INSERT INTO table_name (column_1, column_2, column_3)
VALUES (value_1, 'value_2', value_3);
```

INSERT statements are used to add a new row to a table.

IS NULL / IS NOT NULL

```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NULL;
```

IS NULL and IS NOT NULL are operators used with the WHERE clause to test for empty values.

LIKE

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

LIKE is a special operator used with the WHERE clause to search for a specific pattern in a column.

LIMIT

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

LIMIT is a clause that lets you specify the maximum number of rows the result set will have.

MAX()

```
SELECT MAX(column_name)
FROM table_name;
```

MAX() is a function that takes the name of a column as an argument and returns the largest value in that column.

MIN()

```
SELECT MIN(column_name)
FROM table_name;
```

MIN() is a function that takes the name of a column as an argument and returns the smallest value in that column.

OR

```
SELECT column_name
FROM table_name
WHERE column_name = value_1
  OR column_name = value_2;
```

OR is an operator that filters the result set to only include rows where either condition is true.

ORDER BY

```
SELECT column_name
FROM table_name
ORDER BY column_name ASC | DESC;
```

ORDER BY is a clause that indicates you want to sort the result set by a particular column either alphabetically or numerically.

OUTER JOIN

```
SELECT column_name(s)
FROM table_1
LEFT JOIN table_2
```

```
ON table_1.column_name = table_2.column_name;
```

An outer join will combine rows from different tables even if the join condition is not met. Every row in the *left* table is returned in the result set, and if the join condition is not met, then NULL values are used to fill in the columns from the *right* table.

ROUND()

```
SELECT ROUND(column_name, integer)
FROM table_name;
```

ROUND() is a function that takes a column name and an integer as an argument. It rounds the values in the column to the number of decimal places specified by the integer.

SELECT

```
SELECT column_name
FROM table_name;
```

SELECT statements are used to fetch data from a database. Every query will begin with SELECT.

SELECT DISTINCT

```
SELECT DISTINCT column_name
FROM table_name;
```

SELECT DISTINCT specifies that the statement is going to be a query that returns unique values in the specified column(s).

SUM

```
SELECT SUM(column_name)
FROM table_name;
```

SUM() is a function that takes the name of a column as an argument and returns the sum of all the values in that column.

UPDATE

```
UPDATE table_name
```

```
SET some_column = some_value
WHERE some_column = some_value;
```

UPDATE statements allow you to edit rows in a table.

WHERE

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value;
```

WHERE is a clause that indicates you want to filter the result set to include only rows where the following *condition* is true.

WITH

```
WITH temporary_name AS (
  SELECT *
  FROM table_name)
SELECT *
FROM temporary_name
WHERE column_name operator value;
```

WITH clause lets you store the result of a query in a temporary table using an alias. You can also define multiple temporary tables using a comma and with one instance of the WITH keyword.

The WITH clause is also known as common table expression (CTE) and subquery factoring.

SQL Data Types

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

These are the general data types in SQL.

Data-type	**Syntax**	**Explanation**
Integer	INTEGER	The integer data type is used to specify an

		integer value.
Smallint	SMALLINT	The smallint data type is used to specify small integer value.
Numeric	NUMERIC(P,S)	It specifies a numeric value. Here 'p' is precision value and 's' is scale value.
Real	REAL	The real integer is used to specify a single precision floating point number.
Decimal	DECIMAL(P,S)	It specifies a decimal value. Here 'p' is precision value and 's' is scale value.
Double precision	DOUBLE PRECISION	It specifies double precision floating point number.
Float	FLOAT(P)	It specifies floating-point value e.g. 12.3, 4.5 etc. Here, 'p' is precision value.
Character	CHAR(X)	Here, 'x' is the character's number to store.
Character varying	VARCHAR2(X)	Here, 'x' is the character's number to store
Bit	BIT(X)	Here, 'x' is the number of bits to store
Bit varying	BIT VARYING(X)	Here, 'x' is the number of bits to store (length can vary up to x).
Date	DATE	It stores year, month and days values.
Time	TIME	It stores hour, minute and second values
Timestamp	TIMESTAMP	The timestamp data type is used to store year, month, day, hour, minute and second values.
Time with time zone	TIME WITH TIME ZONE	It is exactly same as time but also store an offset from UTC of the time specified.
Timestamp with time zone	TIMESTAMP with TIME ZONE	It is same as timestamp but also stores an offset from UTC of the time specified.

The **UTC offset** is the difference in hours and minutes from Coordinated Universal Time (**UTC**) for a particular place and date.

Text data types:

Data type	**Description**
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type

TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items

Number data types:

Data type	**Description**
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is

	specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Date data types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS
TIME()	A time. Format: HH:MI:SS
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

SQL Operators

SQL statements generally contain some reserved words or characters that are used to perform operations such as comparison and arithmetical operations etc. These reserved words or characters are known as operators.

Generally there are three types of operators in SQL:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators

SQL Arithmetic Operators:

Let's assume two variables "a" and "b". Here "a" is valued 50 and "b" valued 100.

Example:

Operators	Descriptions	Examples
+	It is used to add containing values of both operands	a+b will give 150
-	It subtracts right hand operand from left hand operand	a-b will give -50
*	It multiply both operand?s values	a*b will give 5000
/	It divides left hand operand by right hand operand	b/a will give 2
%	It divides left hand operand by right hand operand and returns reminder	b%a will give 0

SQL Comparison Operators:

Let's take two variables "a" and "b" that are valued 50 and 100.

Operator	Description
=	Examine both operands value that are equal or not,if yes condition become true.
!=	This is used to check the value of both operands equal or not,if not condition become true.
>	Examine the left operand value is greater than right Operand, if yes condition becomes true
<	Examines the left operand value is less than right Operand, if yes condition becomes true
>=	Examines that the value of left operand is greater than or equal to the value of right operand or not,if yes condition become true
<=	Examines that the value of left operand is less than or equal to the value of right operand or not, if yes condition becomes true

SQL Logical Operators:

This is the list of logical operators used in SQL.

Operator	Description
ALL	this is used to compare a value to all values in another value set.

AND	this operator allows the existence of multiple conditions in an SQL statement.
ANY	this operator is used to compare the value in list according to the condition.
BETWEEN	this operator is used to search for values, that are within a set of values
IN	this operator is used to compare a value to that specified list value
NOT	the NOT operator reverse the meaning of any logical operator
OR	this operator is used to combine multiple conditions in SQL statements
EXISTS	the EXISTS operator is used to search for the presence of a row in a specified table
LIKE	this operator is used to compare a value to similar values using wildcard operator

ALL

TRUE if all of the subquery values meet the condition

SELECT * FROM Products
WHERE Price > ALL (SELECT Price FROM Products WHERE Price > 50);

ProductID	ProductName	SupplierID	CategoryID	Price
1	Toy	1	1	18
2	Plastic	1	1	19
3	Steel	1	2	10

AND

SELECT * FROM Customers
WHERE City = "London" AND Country = "UK";

CustomerID	CustomerName	City	Country
4	**ContactName**	London	UK
11	Thomas Hardy	London	UK
16	Victoria Ashworth	London	U

ANY

TRUE if any of the subquery values meet the condition

SELECT * FROM Products
WHERE Price > ANY (SELECT Price FROM Products WHERE Price > 50);

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
9	A	4	6	18 - 500 g pkgs.	97
18	B	7	8	16 kg pkg.	62.5
20	C	8	3	30 gift boxes	81

BETWEEN

TRUE if the operand is within the range of comparisons

SELECT * FROM Products
WHERE Price BETWEEN 50 AND 60;

ProductID	ProductName	SupplierID	CategoryID	Price
51	A	24	7	53
59	B	28	4	55

IN

TRUE if the operand is equal to one of a list of expressions

SELECT * FROM Customers
WHERE City IN ('Paris','London');

CustomerID	CustomerName	ContactName	Address	City
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London

NOT

Displays a record if the condition(s) is NOT TRUE

SELECT * FROM Customers
WHERE City NOT LIKE 's%';

CustomerID	CustomerName	Address	City
1	Alfreds Futterkiste	Obere Str. 57	Berlin

2	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222	México D.F.
3	Antonio Moreno Taquería	Mataderos 2312	México D.F.

OR

TRUE if any of the conditions separated by OR is TRUE

SELECT * FROM Customers
WHERE City = "London" OR Country = "UK";

CustomerID	CustomerName	Address	City	Country
4	A	120 Hanover Sq.	London	UK
11	B	Fauntleroy Circus	London	UK
38	C	Garden House Crowther Way	Cowes	UK

EXISTS

TRUE if the subquery returns one or more records

SELECT * FROM Products
WHERE EXISTS (SELECT Price FROM Products WHERE Price > 50);

ProductID	ProductName	SupplierID	Unit	Price
1	A	1	10 boxes x 20 bags	18
2	B	1	24 - 12 oz bottles	19
3	C	1	12 - 550 ml bottles	10

LIKE

TRUE if the operand matches a pattern.

SELECT * FROM Customers
WHERE City LIKE 's%';

CustomerID	Address	City	Country
7	24, place Kléber	Strasbourg	France
15	Av. dos Lusíadas, 23	São Paulo	Brazil
21	Rua Orós, 92	São Paulo	Brazil

SQL Expressions

An expression is a combination of one or more values, operators and SQL functions that evaluate to a value. These SQL EXPRESSIONs are like formulae and they are written in query language.

Syntax

Consider the basic syntax of the SELECT statement as follows −

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONDITION|EXPRESSION];
```

There are different types of SQL expressions, which are mentioned below −

- Boolean
- Numeric
- Date

Boolean Expressions

SQL Boolean Expressions fetch the data based on matching a single value. Following is the syntax −

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHING EXPRESSION;
```

The following table is a simple example showing the usage of various SQL Boolean Expressions −

```
SQL> SELECT * FROM CUSTOMERS WHERE SALARY = 10000;
+----+-------+-----+---------+----------+
| ID | NAME  | AGE | ADDRESS | SALARY   |
+----+-------+-----+---------+----------+
|  7 | Muffy |  24 | Indore  | 10000.00 |
```

Numeric Expression

These expressions are used to perform any mathematical operation in any query. Following is the syntax −

```
SELECT numerical_expression as  OPERATION_NAME
[FROM table_name
WHERE CONDITION] ;
```

Here, the numerical_expression is used for a mathematical expression or any formula. Following is a simple example showing the usage of SQL Numeric Expressions −

```
SQL> SELECT (15 + 6) AS ADDITION
+----------+
| ADDITION |
+----------+
|      21 |
+----------+
```

There are several built-in functions like avg(), sum(), count(), etc., to perform what is known as the aggregate data calculations against a table or a specific table column.

Date Expressions

Date Expressions return current system date and time values −

```
SQL>  SELECT CURRENT_TIMESTAMP;
+---------------------+
| Current_Timestamp   |
+---------------------+
| 2009-11-12 06:40:23 |
+---------------------+
```

Another date expression is as shown below −

```
SQL>  SELECT  GETDATE();;
+-------------------------+
| GETDATE                 |
+-------------------------+
| 2009-10-22 12:07:18.140 |
+-------------------------+
```

Introduction to SQL * Plus

SQL*Plus is essentially an interactive query tool with some scripting capabilities. We can enter a SQL statement, such as a SELECT query, and view the results. We can execute *data definition language* (DDL) statements to create tables and other objects. DBAs can use SQL*Plus to start up, shut down, and otherwise administer a database.

In spite of all, GUI-based SQL generators contained in products such as PowerBuilder, Clear Access, and Crystal Reports, it is quicker and easier to build up and test a complex query in SQL*Plus before transferring it to whatever development tool.

*Uses for SQL*Plus*

Originally developed simply as a way to enter queries and see results, SQL*Plus has been enhanced with scripting and formatting capabilities and can be used for many different purposes. The basic functionality is simple. With SQL*Plus, you can do the following:

- Issue a SELECT query and view the results.
- Insert, update, and delete data from database tables.
- Submit PL/SQL blocks to the Oracle server for execution.
- Issue DDL statements, such as those used to create, alter, or drop database objects (e.g., tables, indexes, and users), as well as any other types of SQL statements that Oracle supports.
- Execute SQL*Plus script files.
- Write output to a file.
- Execute procedures and functions that are stored in a database.

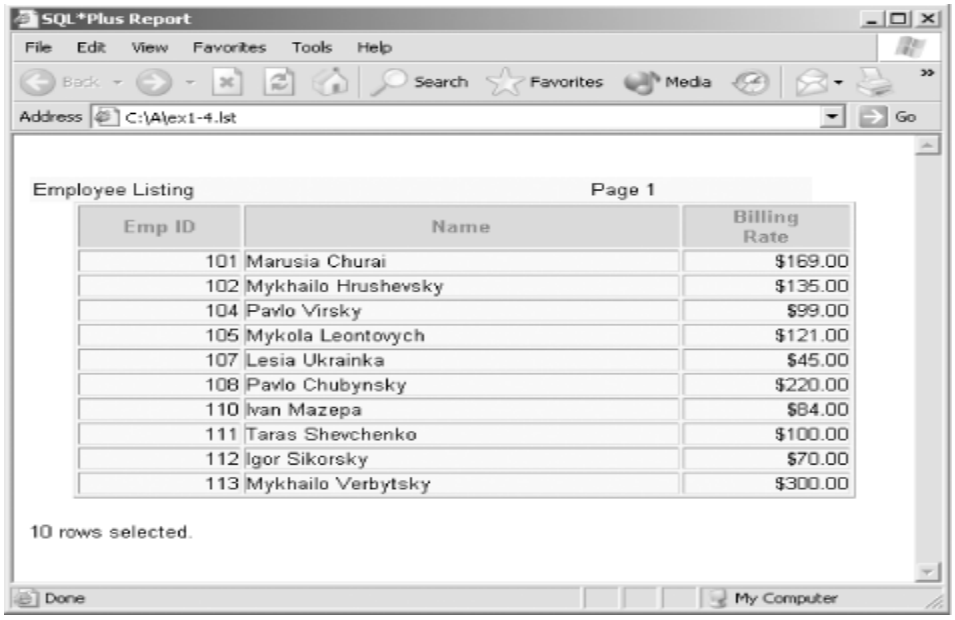
Beginning with SQL*Plus in Oracle8*i* Database, you can use the SET MARKUP HTML command to generate HMTL output, such as that shown below.

QL*Plus report formatted in HTML

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=US-ASCII">
<meta name="generator" content="SQL*Plus 10.1.0">
<style type='text/css'> body {font:10pt Arial,Helvetica,
sans-serif; color:black; background:White;}
...
<tr>
<td align="right">
     101
</td>
```

```
<td>
Marusia Churai
</td>
<td align="right">
 $169.00
</td>
</tr>
<tr>
<td align="right">
      102
</td>
<td>
Mykhailo Hrushevsky
</td>
...
```

By writing such HTML output to a file, you can easily generate ad hoc reports for users to view from a corporate intranet. One DBA whom I spoke with regularly refreshes the phone list on his departmental intranet using this mechanism. The output is rendered in a browser.



Of course, it's rare that you would issue such a simple statement, or just one statement, when you add a new user. Usually, you also want to assign a default tablespace and often a quota on that tablespace. You may also want to grant the privilege needed to connect to the database. Whenever you have a task that requires a sequence of statements to be executed, you can simplify things by taking advantage of SQL*Plus's scripting capabilities. The statements in Example 1-5, when placed in a script file, allow you to add a new user with just one command.

Example 1-5. Script to create a new database user

```
CREATE USER &&1 IDENTIFIED BY &&2
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  QUOTA &&3.M ON users;

GRANT CONNECT TO &&1;
```

The &&1, &&2, and &&3 in Example 1-5 are SQL*Plus user variables marking the locations at which to insert parameters that you pass to the script. Assuming that you give the name *create_user.s ql* to the file shown in Example 1-5, and assuming that you are the DBA, you can issue the following command from SQL*Plus whenever you need to add a user to your database:

Example 1-6 shows how this works, by creating a user named sql_dude with a password of yooper and a quota of 10 megabytes.

Example 1-6. Running a script to create a new database user

```
SQL> @ex1-5 sql_dude yooper 10
old   1: CREATE USER &&1 IDENTIFIED BY &&2
new   1: CREATE USER sql_dude IDENTIFIED BY yooper
old   4:    QUOTA &&3.M ON users
new   4:    QUOTA 10M ON users

User created.

old   1: GRANT CONNECT TO &&1
new   1: GRANT CONNECT TO sql_dude

Grant succeeded.
```

Example 1-7. "Hello World" written as a PL/SQL block and executed from SQL*Plus

```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
  2   DBMS_OUTPUT.PUT_LINE('Hello World!');
  3  END;
  4  /
Hello World!
```

SQL*Plus's Relation to SQL, PL/SQL, and the Oracle Database

SQL*Plus is often used in conjunction with two other products, both of which have the letters "SQL" in their names. The first is SQL itself. Without a doubt, the most common use of SQL*Plus is to submit SQL statements to the database for execution. The second product is Oracle's PL/SQL procedural language. Table 1-1 provides a short summary of each of these three products.

Table 1-1. The three SQLs: SQL, PL/SQL, and SQL*Plus

Product	Description
SQL	SQL is an ANSI and ISO standard language used to insert, delete, update, and retrieve data from relational databases. SQL is also used to manage relational databases.
PL/SQL	PL/SQL is a proprietary procedural language developed by Oracle as an extension to SQL, for use in coding business rules and other procedural logic at the database level. Like SQL, PL/SQL executes inside the database engine.
SQL*Plus	SQL*Plus is an Oracle-developed tool that allows you to interactively enter and execute SQL commands and PL/SQL blocks.

Because these three products all have "SQL" as part of their names, people occasionally get confused about the relationships among them and about which statements get executed where. SQL*Plus does have its own set of commands that it recognizes and executes (for example, SET SERVEROUTPUT ON), but any SQL statements and PL/SQL blocks are sent to the database server for execution. Figure 1-2 illustrates this relationship.

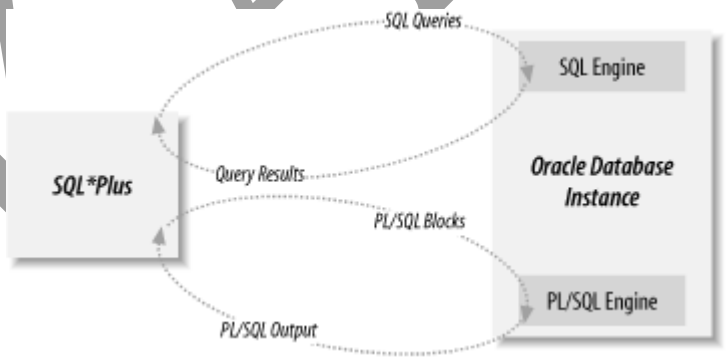


Figure 1-2. Relationships among SQL*Plus, SQL, and PL/SQL

Think of SQL*Plus as kind of a middleman, standing between you and Oracle and helping you to communicate with your database. You type in a SQL query, SQL*Plus takes it and sends it to the database, the database returns the results to SQL*Plus, and SQL*Plus displays those results in a format you can understand.

POSSIBLE QUESTIONS
UNIT I

2 marks Questions:

1. What is a SQL command?
2. Define TCL.
3. Define SQL*Plus.
4. What is meant by SQL expression?
5. What is meant by DML?
6. Define SQL operators.

6 marks Questions:

1. Differentiate SQL and SQL*Plus.
2. List and explain SQL commands.
3. Describe about SQL datatypes.
4. Explain in detail about SQL expressions.
5. Elaborate SQL*Plus.
6. Explain DDL commands with example.

Karpagam Academy of Higher Education
Department of CS
Subject-Oracle (SQL-PL/SQL) **Class III B.Sc CS** **2016-2019 Batch**
Objective Type Questions

Unit-1

sno	Questions	opt1	opt2	opt3	opt4	Answer
1	_____was adopted by the ANSI and ISO.	PSQL	SQL	R-SQL	Sequel	SQL
2	_____is a collection of high-level data description constructs that hide many low-level storage details	Data Model	ER Model	Network Model	none	Data Model
3	Database management System based on _____	Network model	Hierarchical model	Relational model	Object-based model	Relational model
4	A widely used Semantic model called _____	Network model	ER Model	Object-based model	Hierarchical model	ER Model
5	_____is a more abstract	ER model	Semantic data model	Conceptual data Model	Physical data model	Semantic data model
6	_____model is used to pictorially denote entities & relationships	Physical data model	ER model	network model	structure chart	ER model
7	A description Of data in terms of a data model is called_____	Schema	relation	record	entities	Schema
8	Field is otherwise known as _____	Column	Entity	Relationship	Relation	Column
9	Column is otherwise known as _____	Entity	Relationship	Relation	attribute	attribute
10	_____is a software designed to assist in maintaining and utilizing large collections of data.	Database	DBMS	Entities	attributes.	DBMS
11	_____model used Object store & versant.	Network Model	Hierarchical Model	Object Oriented Model	Record based Model	Object Oriented Model

12	____is used to define the external and conceptual model	DDL	DML	DCL	TCL	DDL
13	Conceptual model otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema
14	Physical model specifies _____details	Information	data	Storage	relationships	Storage
15	The _____ enviroment involves dumb terminals	mainframe	client/server	internet computing	LAN	mainframe
16	A host computer in internet computing enirоment is called _	server	data server	PC	web server	web server
17	____ is the primary unit of storage in a database	table	column	row	number	table
18	database design involves conversion of ____ to stuctured database model.	business process	business model	entity	relationships	business model
19	The architecture of a hierarchical database is based on ____the concept of relationships.	set structure	tree/node	parent/child	server/client	parent/child
20	The relationship between tables in the network model is called a	parent/child	set structure	client/server	tree/node	set structure
21	Set structures can represent a __ relationship between tables	one-to-one	one-to-many	many-to-many	many-to-one	one-to-many
22	SQL has been developed and used for ___ model	relational	Hierarchical	network	flat file	relational
23	A class is the equivalent of a _____ in a relational database	row	column	table	primary key	table
24	SQL3 is also referred to as	SQL97	SQL98	SQL99	SQL100	SQL99
25	_____ is the process of creating an interface for the end user through which the database can be accessed	dabase design	business model	interface design	Application design	Application design

26	The process of reducing data redundancy in a relational database is called	data security	data accuracy	data protection	normalization	normalization
27	Static, or _____ data is seldom or never modified once stored in the database.	dynamic	historic	information	transactional	historic
28	_____ or transactional data, is data that is frequently modified once stored in the database.	dynamic	historic	information	transactional	dynamic
29	BPR is	Business product re-engineering	Business product repair	Business process re-engineering	Business procedure re-engineering	Business process re-engineering
30	___ is the process of ensuring that data is consistent between related tables	primary key	database security	performance	Referential integrity	Referential integrity
31	Foreign keys are defined in ___ tables	parent	child	one	database	child
32	_____is an object in the real world	Entity	Attribute	Relationship	Property	Entity
33	in database model the data is stored in objects	hierarchical	network	relational	object_oriented	object_oriented
34	In relational model the data is stored in ___	table	files	objects	sets	table
35	Infromation about the conceptual,external and physical schemas is stored in	Directory	System Catalogs	IMS	Information System	System Catalogs
36	Conceptual schema otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema
37	Physical Schema specifies _____details	Information	data	Storage	relationships	Storage
38	_____is used to speed up data retrieval operations.	DML Operations	Select Operation	Indexes	select operation with where	Indexes
39	A Structure of database using the given data model is called a _____	Database	Relation	Schema	design	Schema

40	SQL was developed as an integral part of	A hierarchical database	A relational database	A OO database	A network database	A relational database
41	Which of the following is CORRECT about database management system's languages?	Data definition languages are used to specify the conceptual	Data manipulation languages are used to create	Data manipulation languages are used for	Data definition languages are only used to	Data manipulation languages are used for
42	An E-R modelling for given application leads to	conceptual data model	logical data model	external data model	internal data model	conceptual data model
43	A conceptual data model is converted using a Relational Data Base Management System to a	logical data model	external data model	internal data model	an entity-relation data model	logical data model
44	A subset of logical data model accessed by programmers is called a	conceptual data model	external data model	internal data model	an entity-relation data model	external data model
45	When a logical model is mapped into a physical storage such as a disk store the resultant data model is known as	conceptual data model	external data model	internal data model	disk data model	internal data model
46	By data integrity we mean	maintaining consistent data values	integrated data values	banning improper access to	not leaking data values	maintaining consistent data values
47	Data integrity is ensured by	good data editing	propagating data changes to all data items	preventing unauthorized access	preventing data duplication	propagating data changes to all data
48	By data security in DBMS we mean	preventing access to data	allowing access to data only to	preventing changing data	introducing integrity constraints	allowing access to data only to
49	By redundancy in a file based system we mean that	unnecessary data is stored	same data is duplicated in many files	data is unavailable	files have redundant data	same data is duplicated in many files
50	Data integrity in a file based system may be lost because	the same variable may have different	files are duplicated	unnecessary data is	redundant data is stored in files	the same variable may have different
51	Data availability is often difficult in file based system	as files are duplicated	as unnecessary data are stored in files	as one has to search different files and	redundant data are stored in files	as one has to search different files and these files
52	An entity is	an inanimate object in an application	a collection of items in an application	a data structure	a distinct real world item in an application	a distinct real world item in an application

53	A relationship is	an item in an application	a meaningful dependency between entities	a collection of related entities	related data	a meaningful dependency between entities
54	Pick the relationship from the following:	a classroom	teacher	attends	cost per dozen	attends
55	Pick the meaningful relationship between entities	vendor supplies goods	vendor talks with customers	vendor complains to vendor	vendor asks prices	vendor supplies goods
56	The entity set is a	set of entities	collection of different entities	collection of related entities	collection of similar entities	collection of similar entities
57	Pick entity set from the following	all vendors supplying to an organization	vendors and organizations they supply	vendors and transporters	a vendor supplying to many organizations	all vendors supplying to an organization
58	The expansion of E-R diagram is	Entity-Relationship diagram	Entity-Relative diagram	Entity-Relation diagram	Entity-Rationalized diagram	Entity-Relationship diagram
59	In an E-R diagram entities are represented by	circles	rectangles	diamond shaped box	ellipse	rectangles
60	In an E-R diagram relationship is represented by	circles	rectangles	diamond shaped box	ellipse	diamond shaped box

UNIT II

SYLLABUS

Managing Tables and Data: Creating and Altering Tables (Including constraints) ,Data Manipulation Command like Insert, update, delete, SELECT statement with WHERE, GROUP BY and HAVING, ORDER BY, DISTINCT, Special operator like IN, ANY, ALL BETWEEN, EXISTS, LIKE, Join, Built in functions

SQL Create Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways.

The first way specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

The SQL GROUP BY Statement

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

GROUP BY Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
ORDER BY *column_name(s);*

Example
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;

SQL - WHERE Clause

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.

Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

SELECT column1, column2, columnN
FROM table_name
WHERE [condition]

Example
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS

WHERE SALARY > 2000;

HAVING

The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > value;

HAVING was added to SQL because the WHERE keyword could not be used with aggregate functions.

ORDER BY

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

SELECT column_name
FROM table_name
ORDER BY column_name ASC | DESC;

ORDER BY is a clause that indicates you want to sort the result set by a particular column either alphabetically or numerically.

SQL - Distinct Keyword

The SQL DISTINCT keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only those unique records instead of fetching duplicate records.

Syntax

The basic syntax of DISTINCT keyword to eliminate the duplicate records is as follows −

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

IN

TRUE if the operand is equal to one of a list of expressions

SELECT * FROM Customers

WHERE City IN ('Paris','London');

CustomerID	CustomerName	ContactName	Address	City
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London

NOT

Displays a record if the condition(s) is NOT TRUE

SELECT * FROM Customers

WHERE City NOT LIKE 's%';

CustomerID	CustomerName	Address	City
1	Alfreds Futterkiste	Obere Str. 57	Berlin
2	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222	México D.F.
3	Antonio Moreno Taquería	Mataderos 2312	México D.F.

OR

TRUE if any of the conditions separated by OR is TRUE

SELECT * FROM Customers

WHERE City = "London" OR Country = "UK";

CustomerID	CustomerName	Address	City	Country
4	A	120 Hanover Sq.	London	UK
11	B	Fauntleroy Circus	London	UK
38	C	Garden House Crowther Way	Cowes	UK

EXISTS

TRUE if the subquery returns one or more records

SELECT * FROM Products

WHERE EXISTS (SELECT Price FROM Products WHERE Price > 50);

ProductID	ProductName	SupplierID	Unit	Price
1	A	1	10 boxes x 20 bags	18
2	B	1	24 - 12 oz bottles	19
3	C	1	12 - 550 ml bottles	10

LIKE

TRUE if the operand matches a pattern.

SELECT * FROM Customers

WHERE City LIKE 's%';

CustomerID	Address	City	Country
7	24, place Kléber	Strasbourg	France
15	Av. dos Lusíadas, 23	São Paulo	Brazil
21	Rua Orós, 92	São Paulo	Brazil

SQL INNER JOIN (simple join)

SQL INNER JOINS return all rows from multiple tables where the join condition is met.

Syntax

The syntax for the INNER JOIN in SQL is:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Example

Let's look at an example of how to use the INNER JOIN in a query.

In this example, we have a table called *customers* with the following data:

customer_id	last_name	first_name	favorite_website
4000	Jackson	Joe	techonthenet.com
5000	Smith	Jane	digminecraft.com
6000	Ferguson	Sam	bigactivities.com
7000	Reynolds	Allen	checkyourmath.com
8000	Anderson	Paige	NULL
9000	Johnson	Derek	techonthenet.com

And a table called *orders* with the following data:

order_id	customer_id	order_date
1	7000	2016/04/18
2	5000	2016/04/18
3	8000	2016/04/19
4	4000	2016/04/20
5	NULL	2016/05/01

Enter the following SQL statement:

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

Output

customer_id	order_id	order_date
4000	4	2016/04/20
5000	2	2016/04/18
7000	1	2016/04/18
8000	3	2016/04/19

SQL LEFT OUTER JOIN

Another type of join is called a LEFT OUTER JOIN. This type of join returns all rows from the LEFT-hand table specified in the ON condition and only those rows from the other table where the joined fields are equal (join condition is met).

Syntax

The syntax for the LEFT OUTER JOIN in SQL is:

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

In some databases, the OUTER keyword is omitted and written simply as LEFT JOIN.

The SQL LEFT OUTER JOIN would return the all records from *table1* and only those records from *table2* that intersect with *table1*.

Using the same *customers* table as the previous example:

customer_id	last_name	first_name	favorite_website
4000	Jackson	Joe	techonthenet.com
5000	Smith	Jane	digminecraft.com
6000	Ferguson	Sam	bigactivities.com
7000	Reynolds	Allen	checkyourmath.com
8000	Anderson	Paige	NULL
9000	Johnson	Derek	techonthenet.com

And the *orders* table with the following data:

order_id	customer_id	order_date
1	7000	2016/04/18
2	5000	2016/04/18
3	8000	2016/04/19
4	4000	2016/04/20
5	NULL	2016/05/01

Enter the following SQL statement:

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
```

```
LEFT OUTER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

There will be 6 records selected. These are the results that you should see:

customer_id	order_id	order_date
4000	4	2016/04/20
5000	2	2016/04/18
6000	NULL	NULL
7000	1	2016/04/18
8000	3	2016/04/19
9000	NULL	NULL

SQL RIGHT OUTER JOIN

Another type of join is called a SQL RIGHT OUTER JOIN. This type of join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where the joined fields are equal (join condition is met).

Syntax

The syntax for the RIGHT OUTER JOIN in SQL is:

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

In some databases, the OUTER keyword is omitted and written simply as RIGHT JOIN.

Using the same *customers* table as the previous example:

customer_id	last_name	first_name	favorite_website
4000	Jackson	Joe	techonthenet.com
5000	Smith	Jane	digminecraft.com
6000	Ferguson	Sam	bigactivities.com
7000	Reynolds	Allen	checkyourmath.com
8000	Anderson	Paige	NULL
9000	Johnson	Derek	techonthenet.com

And the *orders* table with the following data:

order_id	customer_id	order_date
1	7000	2016/04/18
2	5000	2016/04/18
3	8000	2016/04/19
4	4000	2016/04/20
5	NULL	2016/05/01

Enter the following SQL statement:

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
```

RIGHT OUTER JOIN orders

ON customers.customer_id = orders.customer_id

ORDER BY customers.customer_id;

There will be 5 records selected. These are the results that you should see:

customer_id	order_id	order_date
NULL	5	2016/05/01
4000	4	2016/04/20
5000	2	2016/04/18
7000	1	2016/04/18
8000	3	2016/04/19

SQL FULL OUTER JOIN

Another type of join is called a SQL FULL OUTER JOIN. This type of join returns all rows from the LEFT-hand table and RIGHT-hand table with NULL values in place where the join condition is not met.

Syntax

The syntax for the SQL FULL OUTER JOIN is:

SELECT columns

FROM table1

FULL [OUTER] JOIN table2

ON table1.column = table2.column;

In some databases, the OUTER keyword is omitted and written simply as FULL JOIN.

Using the same *customers* table as the previous example:

customer_id	last_name	first_name	favorite_website
4000	Jackson	Joe	techonthenet.com
5000	Smith	Jane	digminecraft.com
6000	Ferguson	Sam	bigactivities.com
7000	Reynolds	Allen	checkyourmath.com
8000	Anderson	Paige	NULL
9000	Johnson	Derek	techonthenet.com

And the *orders* table with the following data:

order_id	customer_id	order_date
1	7000	2016/04/18
2	5000	2016/04/18
3	8000	2016/04/19
4	4000	2016/04/20
5	NULL	2016/05/01

Enter the following SQL statement:

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
FULL OUTER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

There will be 7 records selected. These are the results that you should see:

customer_id	order_id	order_date
NULL	5	2016/05/01
4000	4	2016/04/20
5000	2	2016/04/18
6000	NULL	NULL
7000	1	2016/04/18
8000	3	2016/04/19
9000	NULL	NULL

Built in functions

Numeric Functions

Function	Description
ABS	Returns the absolute value of a number
AVG	Returns the average value of an expression
CEILING	Returns the smallest integer value that is greater than or equal to a number
COUNT	Returns the count of an expression
FLOOR	Returns the largest integer value that is equal to or less than a number
MAX	Returns the maximum value of an expression
MIN	Returns the minimum value of an expression

RAND	Returns a random number or a random number within a range
ROUND	Returns a number rounded to a certain number of decimal places
SIGN	Returns a value indicating the sign of a number
SUM	Returns the summed value of an expression

Conversion Functions

Function	Description
CAST	Converts an expression from one data type to another
CONVERT	Converts an expression from one data type to another

POSSIBLE QUESTIONS
UNIT-II

2 marks questions

1. What is known as constraints?
2. Define primary key.
3. Define foreign key.
4. What is the use of WHERE clause?
5. Define FULL OUTER JOIN.

6 marks questions

1. Elaborate Constraints with suitable queries.
2. Explain about Data Manipulation Language.
3. Explain in detail about special operators.
4. Discuss the concept of join.
5. Explain about Built in functions in SQL.

Karpagam Academy of Higher Education
Department of CS
Subject-Oracle (SQL-PL/SQL) Class III B.Sc CS 2016-2019 Batch
Objective Type Questions
UNIT-II

SNO	Questions	opt1	opt2	opt3	opt4	Answer
1	The rows of a relation	must be in specified order	may be in any order	in ascending order of key	in descending order of key	may be in any order
2	The columns of a relation	must be in specified order	may be in any order	with key field in first column	with largest width column last	may be in any order
3	A_____is a dependency of one non primary key attribute on another non primary key attribute.	Functional dependency	Transitive dependency	Partial dependency	Non primary key dependency	Partial dependency
4	functional depency of the form X →Y where Y is a subset of X are called _____	Functional dependency	Transitive dependency	Partial dependency	Trivial dependency	Trivial dependency
5	If B is a subset of A ,then A→B indicates_____	Reflexivity	Augmentation	Transitivity	Decomposition	Reflexivity
6	If A→B, then AC→BC indicates_____	Reflexivity	Augmentation	Transitivity	Decomposition	Augmentation
7	If A→B and B→C, then A→C indicates_____	Reflexivity	Augmentation	Transitivity	Decomposition	Transitivity
8	If A→BC, then A→B and A→C indicates_____	Reflexivity	Augmentation	Transitivity	Decomposition	Decomposition
9	If A→B and A→C, then A→BC indicates_____	Reflexivity	Augmentation	Transitivity	Union	Union
10	If A→B and C→D, then AC→BD indicates_____	Composition	Augmentation	Transitivity	Union	Composition
11	_____technique is used to reduce the redundancy	Closure set	Decomposition	Normalization	Null Values	Normalization

12	if and only if the right-hand side is not a subset of the left-hand side, then functional dependency is said to be as _____	Non-trivial	Trivial	Transitive	Augmentation	Non-trivial
13	The Closure of F denoted as_____	Fc	F—	FXX	F+	F+
14	All decomposition are used to eliminate_____	duplicates	null values	empty values	not null values	duplicates
15	_____is a kind of IC, that generalizes the concept of a key.	Decomposition	Functional dependency	cursors	Triggers	Functional dependency
16	_____Consists of replacing the relation schema by two relation schemas that each contain a subsets of the attributes.	Decomposition	Functional dependency	cursors	Triggers	Decomposition
17	X is a proper subset of some key K. Such a dependency is sometimes called_____	dependency	Partial dependency	transitive dependency	Decompostion	Partial dependency
18	X is not a proper subset of some key K. Such a dependency is sometimes called_____	dependency	Partial dependency	transitive dependency	Decompostion	transitive dependency
19	Indicated by using arrow from entities to relationships in the ER diagram.	Arrow	Thick line	Dotted line	Shaded line	Arrow
20	Aggragation is indicated by _____in ER diagram	Solid line	Thick line	Thin line	Dotted line	Dotted line
21	ISA is indicated by _____symbol	Rectangle	Ellipse	Triangle	Diamond	Triangle
22	_____is a set of associated values	Entity	Attribute	Relationships	Domain	Domain
23	_____consists of a relation schema and a relation instance.	relation	table	domain	entity	relation
24	An instance of a relation is a set of _____	tuple	domain	attribute	relationships	tuple
25	Each tuple is a _____	Column	row	table	instance	row

26	_____is an object in the real world	Entity	Attribute	Relationship	Property	Entity
27	Collection Of Similar entities are_____	Attributes	Entity	Entity Set	Relationship	Entity Set
28	An Entity is described using a set Of _____	Entity	Entity Set	Attributes	Relationship	Attributes
29	_____is used to uniquely identify an entity in the set.	Key	Lock	Attributes	Entity	Key
30	DBMS is a collection of …….. that enables user to create and maintain a database.	Keys	Translators	Program	Language Activity	Program
31	In a relational schema, each tuple is divided into fields called	Relations	Domains	Queries	All of the above	Domains
32	In an ER model, ……. is described in the database by storing its data.	Entity	Attribute	Relationship	Notation	Entity
33	DFD stands for	Data Flow Document	Data File Diagram	Data Flow Diagram	Non of the above	Data Flow Diagram
34	A top-to-bottom relationship among the items in a database is established by a	Hierarchical schema	Network schema	Relational Schema	All of the above	Hierarchical schema
35	…… table store information about database or about the system.	SQL	Nested	System	None of these	System
36	……..defines the structure of a relation which consists of a fixed set of attribute-domain pairs.	Instance	Schema	Program	Super Key	Schema
37	…… clause is an additional filter that is applied to the result.	Select	Group-by	Having	Order by	Having
38	A logical schema	is the entire database	is a standard way of organizing information into	Describes how data is actually stored on disk.	All of the above	is a standard way of organizing information into accessible parts.

39	…… is a full form of SQL.	Standard query language	Sequential query language	Structured query language	Server side query language	Structured query language
40	A relational database developer refers to a record as	a criteria	a relation	a tuple	an attribute	a tuple
41	……. keyword is used to find the number of values in a column.	TOTAL	COUNT	ADD	SUM	COUNT
42	An advantage of the database management approach is	data is dependent on programs	data redundancy increases	data is integrated and can be accessed by multiple programs	none of the above	data is integrated and can be accessed by multiple programs
43	The collection of information stored in a database at a particular moment is called as ……	schema	instance of the database	data domain	independence	instance of the database
44	A …… is used to define overall design of the database	schema	application program	data definition language	code	schema
45	Key to represent relationship between tables is called	primary key	secondary key	foreign key	none of the above	foreign key
46	Grant and revoke are ……. statements	DDL	TCL	DCL	DML	DCL
47	DBMS helps achieve	Data independence	Centralized control of data	Neither A nor B	Both A and B	Both A and B
48	……. command can be used to modify a column in a table	alter	update	set	create	alter
49						
50	The candidate key is that you choose to identify each row uniquely is called ……..	Alternate Key	Primary Key	Foreign Key	None of the above	Primary Key
51	…….. is used to determine whether of a table contains duplicate rows.	Unique predicate	Like Predicate	Null predicate	In predicate	Unique predicate

52	To eliminate duplicate rows …… is used	NODUPLICATE	ELIMINATE	DISTINCT	None of these	DISTINCT
53	DCL stands for	Data Control Language	Data Console Language	Data Console Level	Data Control Level	Data Control Language
54	…… is the process of organizing data into related tables.	Normalization	Generalization	Specialization	None of the above	Normalization
55	A ……. Does not have a distinguishing attribute if its own and mostly are dependent entities, which are part of some another entity.	Weak entity	Strong entity	Non attributes entity	Dependent entity	Weak entity
56	…….. is the complex search criteria in the where clause.	Sub string	Drop Table	Predict	Predicate	Predicate
57	…… is preferred method for enforcing data integrity	Constraints	Stored Procedure	Triggers	Cursors	Constraints
58	The number of tuples in a relation is called its ……. While the number of attributes in a relation is called it's ……..	Degree, Cardinality	Cardinality, Degree	Rows, Columns	Columns, Rows	Cardinality, Degree
59	The language that requires a user to specify the data to be retrieved without specifying exactly how to get it is	Procedural DML	Non-Procedural DML	Procedural DDL	Non-Procedural DDL	Procedural DML
60	Which two files are used during operation of the DBMS?	Query languages and utilities	DML and query language	Data dictionary and transaction log	Data dictionary and query language	Data dictionary and transaction log
61						
62						
63						
64						
65						
66						
67						

UNIT III

SYLLABUS

Other Database Objects - View, Synonyms, Index

VIEW

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows −

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

Example

Consider the CUSTOMERS table having the following records −

```
+----+----------+-----+-----------+----------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |

+----+----------+-----+-----------+----------+

|1|Ramesh|32|Ahmedabad|2000.00|

|2|Khilan|25|Delhi|1500.00|

|3| kaushik  |23|Kota|2000.00|

|4|Chaitali|25|Mumbai|6500.00|

|5|Hardik|27|Bhopal|8500.00|

|6|Komal|22| MP        |4500.00|

|7|Muffy|24|Indore|10000.00|

+----+----------+-----+-----------+----------+
```

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS

SELECT name, age

FROM  CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result.

```
+----------+-----+
| name     | age |
+----------+-----+
| Ramesh   |  32 |
| Khilan   |  25 |
| kaushik  |  23 |
| Chaitali |  25 |
| Hardik   |  27 |
| Komal    |  22 |
| Muffy    |  24 |
```

```
+----------+-----+
```

The WITH CHECK OPTION

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following code block has an example of creating same view CUSTOMERS_VIEW with the WITH CHECK OPTION.

```
CREATE VIEW CUSTOMERS_VIEW AS

SELECT name, age

FROM  CUSTOMERS

WHERE age IS NOT NULL

WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View

A view can be updated under certain conditions which are given below −

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

```
SQL > UPDATE CUSTOMERS_VIEW
  SET AGE =35
  WHERE name ='Ramesh';
```

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we cannot insert rows in the CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

Deleting Rows into a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

```
SQL > DELETE FROM CUSTOMERS_VIEW
  WHERE age =22;
```

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
```

```
| 3 | kaushik  |  23 | Kota     |  2000.00 |
| 4 | Chaitali |  25 | Mumbai   |  6500.00 |
| 5 | Hardik   |  27 | Bhopal   |  8500.00 |
| 7 | Muffy    |  24 | Indore   | 10000.00 |
+----+----------+-----+----------+----------+
```

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below −

```
DROP VIEW view_name;
```

Following is an example to drop the CUSTOMERS_VIEW from the CUSTOMERS table.

```
DROP VIEW CUSTOMERS_VIEW;
```

Force VIEW Creation

FORCE keyword is used while creating a view, forcefully. This keyword is used to create a View even if the table does not exist. After creating a force View if we create the base table and enter values in it, the view will be automatically updated.

Syntax for forced View is,

```
CREATEor REPLACE FORCEVIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;
```

Update a VIEW

UPDATE command for view is same as for tables.

Syntax to Update a View is,

```
UPDATEview-name SETVALUE
WHERE condition;
```

NOTE: If we update a view it also updates base table data automatically.

Read-Only VIEW

We can create a view with read-only option to restrict access to the view.

Syntax to create a view with Read-Only Access

```
CREATEor REPLACE FORCEVIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition WITHread-only;
```

The above syntax will create view for **read-only** purpose, we cannot Update or Insert data into read-only view. It will throw an **error**.

Types of View

There are two types of view,

- Simple View
- Complex View

Simple View	**Complex View**
Created from one table	Created from one or more table
Does not contain functions	Contain functions
Does not contain groups of data	Contains groups of data

Synonyms

A **synonym** is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

You generally use synonyms when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the object.

Create Synonym (or Replace)

You may wish to create a synonym so that users do not have to prefix the table name with the schema name when using the table in a query.

Syntax

The syntax to create a synonym in Oracle is:

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym_name
```

```
FOR [schema .] object_name [@ dblink];
```

OR REPLACE

Allows you to recreate the synonym (if it already exists) without having to issue a DROP synonym command.

PUBLIC

It means that the synonym is a public synonym and is accessible to all users. Remember though that the user must first have the appropriate privileges to the object to use the synonym.

schema

The appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.

object_name

The name of the object for which you are creating the synonym. It can be one of the following:

- table
- view
- sequence
- stored procedure
- function
- package
- materialized view
- java class schema object
- user-defined object
- synonym

Example

Let's look at an example of how to create a synonym in Oracle.

For example:

```
CREATE PUBLIC SYNONYM suppliers

FOR app.suppliers;
```

This first CREATE SYNONYM example demonstrates how to create a synonym called *suppliers*. Now, users of other schemas can reference the table called *suppliers* without having to prefix the table name with the schema named *app*. For example:

```
SELECT *

FROM suppliers;
```

If this synonym already existed and you wanted to redefine it, you could always use the *OR REPLACE* phrase as follows:

```
CREATE OR REPLACE PUBLIC SYNONYM suppliers
FOR app.suppliers;
```

Drop synonym

Once a synonym has been created in Oracle, you might at some point need to drop the synonym.

Syntax

The syntax to drop a synonym in Oracle is:

```
DROP [PUBLIC] SYNONYM [schema .] synonym_name [force];
```

PUBLIC
Allows you to drop a public synonym. If you have specified *PUBLIC*, then you don't specify a *schema*.

force
It will force Oracle to drop the synonym even if it has dependencies. It is probably not a good idea to use *force* as it can cause invalidation of Oracle objects.

Example

Let's look at an example of how to drop a synonym in Oracle.

For example:

```
DROP PUBLIC SYNONYM suppliers;
```

This DROP statement would drop the synonym called *suppliers* that we defined earlier.

SQL Index

Index in sql is created on existing tables to retrieve the rows quickly.

When there are thousands of records in a table, retrieving information will take a long time. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly. Indexes can be created on a single column or a group of columns. When a index is created, it first sorts the data and then it assigns a ROWID for each row.

For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and are then referred to one or more specific page numbers.

An index helps to speed up **SELECT** queries and **WHERE**clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data.

Creating an index involves the **CREATE INDEX** statement, which allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in an ascending or descending order.

Indexes can also be unique, like the **UNIQUE** constraint, in that the index prevents duplicate entries in the column or combination of columns on which there is an index.

Create an Index

Syntax

The syntax for creating an index in Oracle/PLSQL is:

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (column1, column2, ... column_n)
[ COMPUTE STATISTICS ];
```

UNIQUE

It indicates that the combination of values in the indexed columns must be unique.

index_name

The name to assign to the index.

table_name

The name of the table in which to create the index.

column1, column2, ... column_n

The columns to use in the index.

COMPUTE STATISTICS

It tells Oracle to collect statistics during the creation of the index. The statistics are then used by the optimizer to choose a "plan of execution" when SQL statements are executed.

Example

Let's look at an example of how to create an index in Oracle/PLSQL.

For example:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name);
```

In this example, we've created an index on the supplier table called supplier_idx. It consists of only one field - the supplier_name field.

We could also create an index with more than one field as in the example below:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name, city);
```

We could also choose to collect statistics upon creation of the index as follows:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name, city)
  COMPUTE STATISTICS;
```

Create a Function-Based Index

In Oracle, you are not restricted to creating indexes on only columns. You can create function-based indexes.

Syntax

The syntax for creating a function-based index in Oracle/PLSQL is:

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (function1, function2, ... function_n)
[ COMPUTE STATISTICS ];
```

UNIQUE

It indicates that the combination of values in the indexed columns must be unique.

index_name

The name to assign to the index.

table_name

The name of the table in which to create the index.

function1, function2, ... function_n

The functions to use in the index.

COMPUTE STATISTICS

It tells Oracle to collect statistics during the creation of the index. The statistics are then used by the optimizer to choose a "plan of execution" when SQL statements are executed.

Example

Let's look at an example of how to create a function-based index in Oracle/PLSQL.

For example:

```
CREATE INDEX supplier_idx
  ON supplier (UPPER(supplier_name));
```

In this example, we've created an index based on the uppercase evaluation of the *supplier_name* field.

However, to be sure that the Oracle optimizer uses this index when executing your SQL statements, be sure that UPPER(supplier_name) does not evaluate to a NULL value. To ensure this, add **UPPER(supplier_name) IS NOT NULL** to your WHERE clause as follows:

```
SELECT supplier_id, supplier_name, UPPER(supplier_name)
FROM supplier
WHERE UPPER(supplier_name) IS NOT NULL
ORDER BY UPPER(supplier_name);
```

Rename an Index

Syntax

The syntax for renaming an index in Oracle/PLSQL is:

```
ALTER INDEX index_name
  RENAME TO new_index_name;
```

index_name

The name of the index that you wish to rename.

new_index_name

The new name to assign to the index.

Example

Let's look at an example of how to rename an index in Oracle/PLSQL.

For example:

```
ALTER INDEX supplier_idx
 RENAME TO supplier_index_name;
```

In this example, we're renaming the index called *supplier_idx* to *supplier_index_name*.

Collect Statistics on an Index

If you forgot to collect statistics on the index when you first created it or you want to update the statistics, you can always use the ALTER INDEX command to collect statistics at a later date.

Syntax

The syntax for collecting statistics on an index in Oracle/PLSQL is:

```
ALTER INDEX index_name
 REBUILD COMPUTE STATISTICS;
```

index_name

The index in which to collect statistics.

Example

Let's look at an example of how to collect statistics for an index in Oracle/PLSQL.

For example:

```
ALTER INDEX supplier_idx
 REBUILD COMPUTE STATISTICS;
```

In this example, we're collecting statistics for the index called supplier_idx.

Drop an Index

Syntax

The syntax for dropping an index in Oracle/PLSQL is:

```
DROP INDEX index_name;
```

index_name

The name of the index to drop.

Example

Let's look at an example of how to drop an index in Oracle/PLSQL.

For example:

```
DROP INDEX supplier_idx;
```

In this example, we're dropping an index called supplier_idx.

UNIT-III

POSSIBLE QUESTIONS

2 marks questions

1. Define view.
2. How to create a view?
3. Define WITH CHECK OPTION.
4. What is the use of FORCE keyword?
5. Define synonyms.
6. What is an index?

6 marks questions

1. Discuss the concept of VIEW with examples.
2. Explain about Synonyms.
3. Elaborate Index.

Karpagam Academy of Higher Education
Department of CS
Subject-Oracle (SQL-PL/SQL) Class III B.Sc CS 2016-2019 Batch
Objective Type Questions

UNIT-III

sno	Questions	opt1	opt2	opt3	opt4	Answer
1	What does SQL stand for?	Structured Query Language	Strong Question Language	Structured Question Language	Structure Query Language	Structured Query Language
2	Create.Alter,Drop commands are_____ language commands	DML	TCL	DCL	DDL	DDL
3	Insert,Select,Update,delete commands are _____language commands	DML	TCL	DCL	DDL	DML
4	Selection Operation is used to ____from a relation	Select the columns	Select the rows	Select the table	none.	Select the rows
5	_____command is used to remove the table definition information	Delete	Remove	Destroy	Drop	Drop
6	_____is used to modify the structure of an existing table.	Modify	Alter	Change	Recreate	Alter
7	View can be dropped using _____command	Delete View	Remove View	Replace View	Drop View	Drop View
8	_____columns are not allowed to contain null values	Primary Key	Candidate key	foreign Key	Unique Key	Primary Key

9	_____are valuable to give the security to our original table	Original table	Duplicate table	Views	Tables	Views
10	_____clause is used to modify a particular row.	Update	Modify	Alter	Where	Where
11	_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.	integrity Constraint	restriction	key	check	integrity Constraint
12	A set of fields that uniquely identifies a tuple according to a key constraint is called_____ for the relation	primary key	Candidate key	foreign key	Super key	Candidate key
13	_____is used to refer the primary key in another entity	Candidate key	referential key	foreign key	Primary key	foreign key
14	_____value is unknown or not applicable	not null	zero	unknown	null	null
15	_____statement is used to define a new table.	Create	Produce	Insert	Add	Create
16	Rows are inserted using the _____command	Create	Insert	Add	Make	Insert
17	rows are deleted using the _____command	Delete	drop	remove	alter	Delete

18	Modify the column values in an existing row using_____ command	Modify	Alter	Update	Change	Update
19	_____command is used to remove the table definition information	Delete	Remove	Destroy	Drop	Drop
20	___is used to modify the structure of an existing table.	Modify	Alter	Change	Recreate	Alter
21	View can be dropped using _____command	Delete View	Remove View	Replace View	Drop View	Drop View
22	Duplicates are eliminated by using_____	Remove	Distinct	RM	Redundant	Distinct
23	Group function is otherwise known_____	Collection	Aggregate	function	Count	Aggregate
24	Pattern matching has done through_____	Comparison	arithmetic	Logical	Aggregate	Aggregate
25	Which keyword is used to check if an element is in a given set?	not	in	not exist	except	in
26	Any two tables that are Union-Compatible that is, have the same number of _____	Columns	rows	Columns and rows	null values	Columns
27	_____keyword is used to eliminates the duplicates	Union	Union all	Intersect all	Except all	Union
28	_____keyword is used to retain the duplicates	Union	Union all	Intersect	Except	Union all

29	_____is a query that has another query embedded within it.	Query	Subquery	QBE	QUEL	Subquery
30	_____ to calculate the number of values in the Column.	Count	aggregate	Cal	Calculate	Count
31	__ is used to calculate the sum of all values in the column	Total	Sum	Count	Collection	Sum
32	____is used to calculate the average of all values in the column	Total	Sum	Average	avg	avg
33	Which function is used to extract the maximum values in the relations?	max	maximum	excess	large	max
34	Which function is used to extract the maximum values in the relations?	Small	minimum	lower	min	min
35	Which clause is used, when we are using Group by clause, instead of where clause?	where	Having	Distinct	Group	Having
36	_____is used when the column value is either unknown or inapplicable.	zero	all	Empty	null	null
37	_____keyword specifies that the join condition is equality on all common attributes and the where clause is not required.	full outerjoin	left outer join	Natural	Right outerjoin	Natural
38	_____constraints for a single table.	Assertion	table constraints	default	union	table constraints

39	_____keyword is used to assign a default value with a domain.	Static	Default	Permanent	distinct	Default
40	Which constraint is used to check the ranges in the column values?	range	verify	check	condition	check
41	_____operator is another set comparison operator such as IN.	Exists	IN	avail	present	Exists
42	Which keyword is similar to NOT IN?	Exist	IN	Not EXIST	Except	Except
43	Which keyword is similar to IN?	Exist	IN	Not EXIST	Except	Exist
44	With SQL, how can you return the number of records in the "Persons" table?	SELECT COLUMNS(*) FROM Persons	SELECT COUNT() FROM Persons	SELECT COUNT(*) FROM Persons	SELECT COLUMNS() FROM Persons	SELECT COUNT(*) FROM Persons
45	How can you change "Hansen" into "Nilsen" in the "LastName" column in the Persons table?	UPDATE Persons SET LastName='Nilsen' WHERE LastName='Hansen'	MODIFY Persons SET LastName='Hansen' INTO LastName='Nilsen	UPDATE Persons SET LastName='Hansen' INTO LastName='Nilsen'	MODIFY Persons SET LastName='Nilsen' WHERE LastName='Hansen'	UPDATE Persons SET LastName='Nilsen' WHERE LastName='Hansen'

46	With SQL, how can you return all the records from a table named "Persons" sorted descending by "FirstName"?	SELECT * FROM Persons SORT 'FirstName' DESC	SELECT * FROM Persons ORDER BY FirstName DESC	SELECT * FROM Persons ORDER FirstName DESC	SELECT * FROM Persons SORT BY 'FirstName' DESC	SELECT * FROM Persons ORDER BY FirstName DESC
47	Which SQL keyword is used to sort the result-set?	ORDER	SORT	SORT BY	ORDER BY	ORDER BY
48	With SQL, how do you select all the records from a table named "Persons" where the value of the	SELECT * FROM Persons WHERE FirstName	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE FirstName
49	You need to calculate the total of all salaries in the accounting department. Which group function should you use?	SUM	COUNT	TOTAL	LARGEST	SUM
50	SELECT ROUND (45.953, -1), TRUNC (45.936, 2) FROM dual; which values are displayed?	46 and 45	46 and 45.93	50 and 45.93	50 and 45.9	46 and 45.93
51	Select operator is not a unary operator	TRUE	FALSE	not opeator	operator	FALSE
52	Project operator chooses subset of attributes or columns of a relation	TRUE	FALSE	not opeator	operator	TRUE

53	…… database is used as template for all databases created.	Master	Model	Tempdb	None of the above	Model
54	One aspect that has to be dealt with by the integrity subsystem is to ensure that only valid values can be assigned to each data items. This is referred to as	Data Security	Domain access	Data Control	Domain Integrity	Domain Integrity
55	…….. operator is basically a join followed by a project on the attributes of first relation.	Join	Semi-Join	Full Join	Inner Join	Semi-Join
56	Which of the following is not a binary operator in relational algebra?	Join	Semi-Join	Assignment	Project	Project
57	Centralizing the integrity checking directly under the DBMS …….. duplication and ensures the consistency and validity of the database.	Increases	Skips	Does not reduce	Reduces	Reduces
58	Which of the following is/are the DDL statements?	Create	Drop	Alter	All of the above	All of the above
59	In snapshot, ……. clause tells oracle how long to wait between refreshes.	Complete	Force	Next	Refresh	Refresh

60	…… defines rules regarding the values allowed in columns and is the standard mechanism for enforcing database integrity.	Column	Constraint	Index	Trigger	Constraint

UNIT IV

SYLLABUS

Transaction Control Statements - Commit, Rollback, Savepoint

Commit, Rollback and Savepoint SQL commands

Transaction Control Language(TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements. It also allows statements to be grouped together into logical transactions.

COMMIT Statement

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Syntax

The syntax for the COMMIT statement in Oracle/PLSQL is:

```
COMMIT [ WORK ] [ COMMENT clause ] [ WRITE clause ] [ FORCE clause ];
```

Parameters or Arguments

WORK

Optional. It was added by Oracle to be SQL-compliant. Issuing the COMMIT with or without the WORK parameter will result in the same outcome.

COMMENT clause

Optional. It is used to specify a comment to be associated with the current transaction. The comment that can be up to 255 bytes of text enclosed in single quotes. It is stored in the system view called DBA_2PC_PENDING along with the transaction ID if there is a problem.

WRITE clause

Optional. It is used to specify the priority that the redo information for the committed transaction is to be written to the redo log. With this clause, you have two parameters to specify:

- *WAIT* or *NOWAIT* (*WAIT* is the default if omitted)
  - *WAIT* - means that the commit returns to the client only after the redo information is persistent in the redo log.
  - *NOWAIT* - means that the commit returns to the client right away regardless of the status of the redo log.
- *IMMEDIATE* or *BATCH* (*IMMEDIATE* is the default if omitted)
  - *IMMEDIATE* - forces a disk I/O causing the log writer to write the redo information to the redo log.
  - *BATCH* - forces a "group commit" and buffers the redo log to be written with other transactions.

FORCE clause

Optional. It is used to force the commit of a transaction that may be corrupt or in doubt. With this clause, you can specify the FORCE in 3 ways:

- *FORCE 'string', [integer]* or *FORCE CORRUPT_XID 'string'* or *FORCE CORRUPT_XID_ALL*
  - *FORCE 'string', [integer]* - allows you to commit a corrupt or in doubt transaction in a distributed database system by specifying the transaction ID in single quotes as *string*. You can find the transaction ID in the system view called DBA_2PC_PENDING. You can specify *integer* to assign the transaction a system change number if you do not wish to commit the transaction using the current system change number.
  - *FORCE CORRUPT_XID 'string'* - allows you to commit a corrupt or in doubt transaction by specifying the transaction ID in single quotes

as *string*. You can find the transaction ID in the system view called V$CORRUPT_XID_LIST.

- *FORCE CORRUPT_XID_ALL* - allows you to commit all corrupted transactions.

Note

- You must have DBA privileges to access the system views - DBA_2PC_PENDING and V$CORRUPT_XID_LIST.
- You must have DBA privileges to specify certain features of the COMMIT statement.

Example

Let's look at an example that shows how to issue a commit in Oracle using the COMMIT statement.

For example:

```
COMMIT;
```

This COMMIT example would perform the same as the following:

```
COMMIT WORK WRITE WAIT IMMEDIATE;
```

In this example, the WORK keyword is implied and the omission of the WRITE clause would default to WRITE WAIT IMMEDIATE so the first 2 COMMIT statements are equivalent.

Comment

Let's look at an example of a COMMIT that shows how to use the *COMMENT* clause:

For example, you can write the COMMIT with a comment in two ways:

```
COMMIT COMMENT 'This is the comment for the transaction';
```

OR

```
COMMIT WORK COMMENT 'This is the comment for the transaction';
```

Since the WORK keyword is always implied, both of these COMMIT examples are equivalent. The COMMIT would store the comment enclosed in quotes along with the transaction ID in the DBA_2PC_PENDING system view, if the transaction was in error or in doubt.

Force

Finally, look at an example of a COMMIT that shows how to use the FORCE clause.

For example, you can write the COMMIT of an in-doubt transaction in two ways:

```
COMMIT FORCE '22.14.67';
```

OR

```
COMMIT WORK FORCE '22.14.67';
```

Since the WORK keyword is always implied, both of these COMMIT examples would force the commit of the corrupted or in doubt transaction identified by the transaction ID '22.14.67'.

ROLLBACK

This command restores the database to last commited state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

ROLLBACKTOsavepoint_name;

Syntax

The syntax for the ROLLBACK statement is:

```
ROLLBACK [ WORK ] [ TO [SAVEPOINT] savepoint_name  | FORCE 'string' ];
```

Parameters or Arguments

WORK

Optional. It was added by Oracle to be SQL-compliant. Issuing the ROLLBACK with or without the WORK parameter will result in the same outcome.

TO SAVEPOINT savepoint_name

Optional. The ROLLBACK statement undoes all changes for the current session up to the savepoint specified by *savepoint_name*. If this clause is omitted, then all changes are undone.

FORCE 'string'

Optional. It is used to force the rollback of a transaction that may be corrupt or in doubt. With this clause, you specify the transaction ID in single quotes as *string*. You can find the transaction ID in the system view called DBA_2PC_PENDING.

Note

- You must have DBA privileges to access the system views - DBA_2PC_PENDING and V$CORRUPT_XID_LIST.
- You can not rollback a transaction that is in doubt to a savepoint.

Example

Let's look at an example that shows how to issue a rollback in Oracle using the ROLLBACK statement.

For example:

```
ROLLBACK;
```

This ROLLBACK example would perform the same as the following:

```
ROLLBACK WORK;
```

In this example, the WORK keyword is implied so the first 2 ROLLBACK statements are equivalent. These examples would rollback the current transaction.

Savepoint

Let's look at an example of a ROLLBACK that shows how to use the rollback to a specific savepoint.

For example, you can write the ROLLBACK to a savepoint in two ways:

```
ROLLBACK TO SAVEPOINT savepoint1;
```

OR

```
ROLLBACK WORK TO SAVEPOINT savepoint1;
```

Since the WORK keyword is always implied, both of these ROLLBACK examples would rollback the current transaction to the savepoint called savepoint1.

Force

Finally, look at an example of a ROLLBACK that shows how to force the rollback of a transaction that is in doubt.

For example, you can write the ROLLBACK of an in-doubt transaction in two ways:

```
ROLLBACK FORCE '22.14.67';
```

OR

```
ROLLBACK WORK FORCE '22.14.67';
```

Since the WORK keyword is always implied, both of these ROLLBACK examples would force the rollback of the corrupted or in doubt transaction identified by the transaction ID '22.14.67'.

SAVEPOINT

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

SAVEPOINTsavepoint_name;

In short, using this command we can name the different states of our data in any table and then rollback to that state using the ROLLBACK command whenever required.

Using Savepoint and Rollback

Following is the table class,

id	name
1	Abhi
2	Adam
4	Alex

Lets use some SQL queries on the above table and see the results.

INSERTINTO class VALUES(5,'Rahul');

COMMIT;

UPDATE class SET name ='Abhijit'WHERE id ='5';

SAVEPOINT A;

INSERTINTO class VALUES(6,'Chris');

SAVEPOINT B;

INSERTINTO class VALUES(7,'Bravo');

SAVEPOINT C;

SELECT*FROM class;

NOTE: SELECT statement is used to show the data stored in the table.

The resultant table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris
7	Bravo

Now let's use the ROLLBACK command to roll back the state of data to the savepoint B.

ROLLBACKTO B;

SELECT*FROM class;

Now class table will look like,

id	name

1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

Now let's again use the ROLLBACK command to roll back the state of data to the savepoint A

ROLLBACKTO A;

SELECT*FROM class;

Now the table will look like,

id	name
1	Abhi
2	Adam
4	Alex

5	Abhijit

UNIT-IV

POSSIBLE QUESTIONS

2 marks questions

1. Define commit.
2. What is meant by Rollback?

3. Define Savepoint.

4. Give an example query for TO SAVEPOINT.

6 marks questions

1. Elaborate COMMIT with examples.

2. Explain about Rollback.

3. Elaborate Savepoint.

Karpagam Academy of Higher Education
Department of CS
Subject-Oracle (SQL-PL/SQL) Class III B.Sc CS 2016-2019 Batch
Objective Type Questions

UNIT-IV

sno	Questions	opt1	opt2	opt3	opt4	Answer
1	____property enables us to recover any instance of the decomposed relation from corresponding instance of the smaller relations.	dependency preservation	lossless-join	normal forms	decomposition	lossless-join
2	BCNF stands for_____	Boyce Codd Normal Form	Boy Codd Normal Form	Basic Codd Normal Form	Basic Codd Normalization Form	Boyce-Codd Normal Form
3	Boye Codd Normal Form (BCNF) is needed when	two non-key attributes are dependent	there is more then one possible composite key	there are two or more possible composite overlapping keys and one attribute of a composite key is dependent on an attribute of another composite key	there are two possible keys and they are dependent on one another	there are two or more possible composite overlapping keys and one attribute of a composite key is dependent on an attribute of another composite key

4	A relation is said to be in BCNF when	it has overlapping composite keys	it has no composite keys	it has no multivalued dependencies	it has no overlapping composite keys which have related attributes	it has no overlapping composite keys which have related attributes
5	A 3 NF relation is converted to BCNF by	removing composite keys	removing multivalued dependencies	dependent attributes of overlapping composite keys are put in a separate relation	dependent non-key attributes are put in a separate table	dependent attributes of overlapping composite keys are put in a separate relation
6	BCNF is needed because	otherwise tuples may be duplicated	when a data is deleted tuples may be lost	updating is otherwise difficult	when there is dependent attributes in two possible composite keys one of the attributes is unnecessarily duplicated in the tuples	when there is dependent attributes in two possible composite keys one of the attributes is unnecessarily duplicated in the tuples
7	Fourth normal form (4 NF) relations are needed when	there are multivalued dependencies between attributes in composite key	there are more than one composite key	there are two or more overlapping composite keys	there are multivalued dependency between non-key attributes	there are multivalued dependencies between attributes in composite key

8	A 3 NF relation is split into 4 NF	by removing overlapping composite keys	by splitting into relations which do not have more than one independent multivalued dependency	removing multivalued dependency	by putting dependent non-key attribute in a separate table	by removing overlapping composite keys
9	A third Normal Form (3 NF) relation should	be in 2 NF	not have complete key	not be 1 NF	should not have non-key attributes depend on key attribute	be in 2 NF
10	The process of normalization	is automatic using a computer program	requires one to understand dependency between attributes	is manual and requires semantic information	is finding the key of a relation	requires one to understand dependency between attributes
11	A relation is said to be in 1NF if	there is no duplication of data	there are no composite attributes in the relation	there are only a few composite attributes	all attributes are of uniform type	there are no composite attributes in the relation
12	The number of normal forms which has been proposed and discussed in the book are	3	4	5	6	6

13	A relation which is in a higher normal form	implies that it also qualifies to be in lower normal form	does not necessarily satisfy the conditions of lower normal form	is included in the lower normal form	is independent of lower normal forms	implies that it also qualifies to be in lower normal form
14	Given an attribute x, another attribute y is dependent on it, if for a given x	there are many y values	there is only one value of y	there is one or more y values	there is none or one y value	there is only one value of y
15	Given the following relation vendor order (vendor no, order no, vendor name, qty supplied , price/unit) the second normal form relations are	vendor (vendor no, vendor name) qty (qty supplied, price/unit) order (order no, qty supplied)	vendor (vendor no, vendor name) order (order no, qty supplied, price/unit)	vendor (vendor no, vendor name) order (order no, qty supplied, price/unit) vendor order (vendor no, order no)	vendor (vendor no, vendor name, qty supplied, price/unit) vendor order (order no, vendor no)	vendor (vendor no, vendor name) order (order no, qty supplied, price/unit) vendor order (vendor no, order no)
16	__technique is used to reduce the redundancy	Closure set	Decomposition	Normalization	Null Values	Normalization
17	3NF is also referred to as_____	LCNF	BCNF	information-preserving	desirable form	BCNF
18	Projection-join normal form also known as _____	1NF	2NF	3NF	5NF	5NF
19	if and only if the right-hand side is not a subset of the left-hand side, then functional dependency is said to be as _____	Non-trivial	Trivial	Transitive	Augmentation	Non-trivial
20	A relation is in _____ if and only if the nonkey attributes are mutually independent	3NF	1NF	2NF	5NF	3NF

21	_____attribute does not participate in the primary key of the relation concerned	key attribute	Non-key attribute	Variable	none	Non-key attribute
22	A relation is in _____if and only if , in every legal value of that relation, every tuple contains exactly one value for each attribute.	2NF	3NF	1NF	4NF	1NF
23	_technique is used to eliminate the redundancy	Null Values	Decomposition	Normalizations	Concatenation	Normalizations
24	A relation is in _____ if and only if it is in 1NF and every nonkey attribute is irreducibly dependent on the primary key.	1NF	2NF	3NF	4NF	2NF
25	A relation is in _____ if and only if it is in 2NF and every nonkey attribute is non transitively dependent on the primary key.	1NF	2NF	3NF	4NF	3NF
26	A relation is _____ if and only if the only determinants are candidate keys.	1NF	2NF	3NF	4NF	2NF
27	In the functional dependency, left-hand side indicates_____	determinants	Dependencies	trivial	non-trivial	determinants
28	In the functional dependency, right-hand side indicates_____	determinants	Dependencies	trivial	non-trivial	Dependencies

29	_____property anables us to enforce any constraint on the original relation by simply enforcing some constraints on each of the smaller relations.	dependency preservation	lossless-join	normal forms	decomposition	dependency preservation
30	_____ package constructs are declared in the package specifiaction and defined in package body.	public	private	internal	external	public
31	_____ package constructs are not declared in the package specifiaction but defined in package body.	public	private	internal	external	private
32	Global package item are declared in package ____ for external users to use it.	defintion	specification	body	declaration	specification
33	Variables declared in package specification are intialised to ___ by default	zero	space	null	one	null
34	which command is used to invoke a procedure in a package	call	execute	invoke	/	execute
35	____is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased database performance.	Denormalization	normalization	functional dependency	Decomposition	Denormalization
36	in select statement Duplicates are eliminated by using _____ keyword.	Remove	Distinct	RM	Redundant	Distinct

37	Group function is otherwise known as_____	Collection	Aggregate	function	Count	Aggregate
38	string Pattern matching is done through_____operator.	Comparison	arithmetic	Logical	like	like
39	Which keyword is used to check if an element is in a given set?	not	in	not exist	except	in
40	Any two tables that are Union-Compatible if they have the same number of _____	Columns	rows	Columns and rows	null values	Columns
41	Which clause is used, when we are using Group by clause, instead of where clause?	where	Having	Distinct	Group	Having
42	_____clause is used to group the values under particular characteristics	Order By	Group by	Count	Sum	Order By
43	which of the following is not a set operator	union	minus	intersect	plus	plus
44	check constraint is ___ level constraint	table	column	row	database	column
45	a view is updatable when it is based upon ____ tables	two	one	three	any numberof	one
46	____ clause is used in to make a sequence repeat the series.	cycle	nocycle	cache	start with	cycle
47	what is the maximum value for a descending sequence	-1	0	null	10^27	-1
48	___ is used and maintained automatically by oracle server	view	table	index	sequence	index

49	The procedure has only ____ priveledge	alter	delete	execute	select	execute
50	……. provides option for entering SQL queries as execution time, rather than at the development stage.	PL/SQL	SQL*Plus	SQL	Dynamic SQL	Dynamic SQL
51	The RDBMS terminology for a row is	tuple	relation	attribute	degree	tuple
52	To change column value in a table the …… command can be used	create	insert	alter	update	update
53	A set of possible data values is called	attribute	degree	tuple	domain	domain
54	…… is critical in formulating database design.	row column order	number of tab	functional dependency	normalizing	functional dependency
55	A ……. represents the number of entities to which another entity can be associated	mapping cardinality	table	schema	information	mapping cardinality
56	Which two files are used during operation of the DBMS	Query languages and utilities	DML and query language	Data dictionary and transaction log	Data dictionary and query language	Data dictionary and transaction log
57	A …….. is a set of column that identifies every row in a table	composite key	candidate key	foreign key	super key	super key
58	The relational model is based on the concept that data is organized and stored in two-dimensional tables called …….	Fields	Records	Relations	Keys	Relations

59	…….. contains information that defines valid values that are stored in a column or data type.	View	Rule	Index	Default	Index
60	insert into <table_name> values <list of values>	TRUE	FALSE	Table	None	TRUE
61						
62						
63						
64						
65						
66						
67						
68						

UNIT V

SYLLABUS

Introduction to PL/SQL SQL v/s PL/SQL, PL/SQL Block Structure, Language construct of PL/SQL (Variables, Basic and Composite Data type, Conditions looping etc.) TYPE and % ROWTYPE , Using Cursor (Implicit, Explicit)

SQL v/s PL/SQL

Difference between SQL and PL/SQL

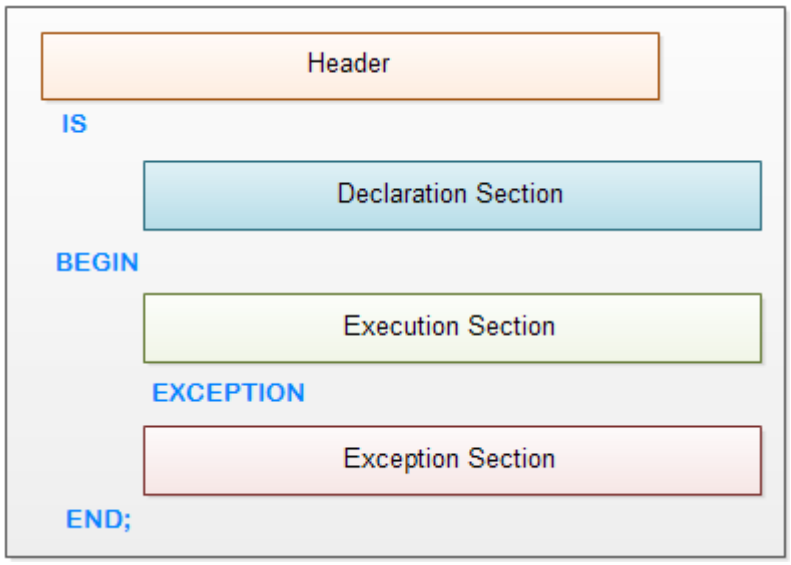
SQL	PL/SQL
• SQL is a single query that is used to perform DML and DDL operations.	• PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
• It is declarative, that defines what need to be done, rather than how things need to be done.	• PL/SQL is procedural that defines how the things needs to be done.
• Execute as a single statement.	• Execute as a whole block.
• Mainly used to manipulate data.	• Mainly used to create an application.
• Interaction with a Database server.	• No interaction with the database server.

• Cannot contain PL/SQL code in it.	• It is an extension of SQL, so that it can contain SQL inside it.

PL/SQL BLOCK STRUCTURE

PL/SQL is a **block-structured** language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts.

S.No	Sections & Description
1	**Declarations** This section starts with the keyword **DECLARE**. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.
2	**Executable Commands** This section is enclosed between the keywords **BEGIN**and **END** and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a **NULL command** to indicate that nothing should be executed.
3	**Exception Handling** This section starts with the keyword **EXCEPTION**. This optional section contains **exception(s)** that handle errors in the program.



Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using **BEGIN**and **END**. Following is the basic structure of a PL/SQL block −

```
DECLARE
<declarations section>
BEGIN
<executable command(s)>
EXCEPTION
<exception handling>
END;
```

The 'Hello World' Example

```
DECLARE

message  varchar2(20):='Hello, World!';

BEGIN

dbms_output.put_line(message);

END;

/
```

The **end;** line signals the end of the PL/SQL block. To run the code from the SQL command line, you may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the following result −

```
Hello World

PL/SQL procedure successfully completed.
```

The PL/SQL Identifiers

PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words. The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters.

By default, **identifiers are not case-sensitive**. So you can use **integer** or **INTEGER** to represent a numeric value. You cannot use a reserved keyword as an identifier.

The PL/SQL Delimiters

A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL −

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator
'	Character string delimiter
.	Component selector
(,)	Expression or list delimiter

:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
||	Concatenation operator
******	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)

--	Single-line comment indicator
..	Range operator
<, >, <=, >=	Relational operators
<>, '=, ~=, ^=	Different versions of NOT EQUAL

The PL/SQL Comments

Program comments are explanatory statements that can be included in the PL/SQL code that you write and helps anyone reading its source code. All programming languages allow some form of comments.

The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /* and */.

```
DECLARE
  -- variable declaration
message  varchar2(20):= 'Hello, World!';
BEGIN
  /*
*  PL/SQL executable statement(s)
  */
dbms_output.put_line(message);
END;
```

```
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
Hello World


PL/SQL procedure successfully completed.
```

PL/SQL Program Units

A PL/SQL unit is any one of the following −

- PL/SQL block
- Function
- Package
- Package body
- Procedure
- Trigger
- Type
- Type body

LANGUAGE CONSTRUCT OF PL/SQL

Variables

A variable is a name given to a storage area that our programs can manipulate. Each variable in PL/SQL has a specific data type, which determines the size and the layout of the variable's memory; the range of values that can be stored within that memory and the set of operations that can be applied to the variable.

The name of a PL/SQL variable consists of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters. By default, variable names are not case-sensitive. You cannot use a reserved PL/SQL keyword as a variable name.

PL/SQL programming language allows to define various types of variables, such as date time data types, records, collections, etc. which we will cover in subsequent chapters. For this chapter, let us study only basic variable types.

Variable Declaration in PL/SQL

PL/SQL variables must be declared in the declaration section or in a package as a global variable. When you declare a variable, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.

The syntax for declaring a variable is −

```
variable_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial_value]
```

Where, *variable_name* is a valid identifier in PL/SQL, *datatype*mustbe a valid PL/SQL data type or any user defined data type which we already have discussed in the last chapter. Some valid variable declarations along with their definition are shown below −

```
sales number(10, 2);
pi CONSTANT double precision := 3.1415;
name varchar2(25);
address varchar2(100);
```

When you provide a size, scale or precision limit with the data type, it is called a **constrained declaration**. Constrained declarations require less memory than unconstrained declarations. For example −

```
sales number(10, 2);
name varchar2(25);
address varchar2(100);
```

Initializing Variables in PL/SQL

Whenever you declare a variable, PL/SQL assigns it a default value of NULL. If you want to initialize a variable with a value other than the NULL value, you can do so during the declaration, using either of the following −

- The **DEFAULT** keyword
- The **assignment** operator

For example −

```
counterbinary_integer := 0;
greetings varchar2(20) DEFAULT 'Have a Good Day';
```

You can also specify that a variable should not have a **NULL**value using the **NOT NULL** constraint. If you use the NOT NULL constraint, you must explicitly assign an initial value for that variable.

It is a good programming practice to initialize variables properly otherwise, sometimes programs would produce unexpected results. Try the following example which makes use of various types of variables −

```
DECLARE
a integer :=10;
b integer :=20;
c integer;
f real;
BEGIN
c:= a + b;
dbms_output.put_line('Value of c: '|| c);
f:=70.0/3.0;
dbms_output.put_line('Value of f: '|| f);
END;
/
```

When the above code is executed, it produces the following result −

```
Value of c: 30
Value of f: 23.333333333333333333

PL/SQL procedure successfully completed.
```

Variable Scope in PL/SQL

PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block. If a variable is declared within an inner block, it is not accessible to the outer block. However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks. There are two types of variable scope −

- **Local variables** − Variables declared in an inner block and not accessible to outer blocks.
- **Global variables** − Variables declared in the outermost block or a package.

Following example shows the usage of **Local** and **Global**variables in its simple form −

```
DECLARE
--Global variables
num1 number :=95;
num2 number :=85;
BEGIN
dbms_output.put_line('Outer Variable num1: '|| num1);
dbms_output.put_line('Outer Variable num2: '|| num2);
  DECLARE
--Local variables
num1 number :=195;
num2 number :=185;
BEGIN
dbms_output.put_line('Inner Variable num1: '|| num1);
dbms_output.put_line('Inner Variable num2: '|| num2);
END;
END;
/
```

When the above code is executed, it produces the following result −

```
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185

PL/SQL procedure successfully completed.
```

Assigning SQL Query Results to PL/SQL Variables

You can use the **SELECT INTO** statement of SQL to assign values to PL/SQL variables. For each item in the **SELECT list**, there must be a corresponding, type-compatible variable in the **INTO list**. The following example illustrates the concept. Let us create a table named CUSTOMERS −

```
CREATE TABLE CUSTOMERS(
   ID   INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY   DECIMAL (18,2),
   PRIMARY KEY (ID)
);


TableCreated
```

Let us now insert some values in the table −

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1,'Ramesh',32,'Ahmedabad',2000.00);
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2,'Khilan',25,'Delhi',1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3,'kaushik',23,'Kota',2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4,'Chaitali',25,'Mumbai',6500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5,'Hardik',27,'Bhopal',8500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6,'Komal',22,'MP',4500.00);
```

The following program assigns values from the above table to PL/SQL variables using the **SELECT INTO clause** of SQL −

```
DECLARE
c_idcustomers.id%type:=1;
c_namecustomers.name%type;
c_addrcustomers.address%type;
c_salcustomers.salary%type;
BEGIN
   SELECT name, address, salary INTO c_name,c_addr,c_sal
   FROM customers
```

```
  WHERE id =c_id;
dbms_output.put_line
('Customer '||c_name||' from '||c_addr||' earns '||c_sal);
END;
/
```

When the above code is executed, it produces the following result −

```
Customer Ramesh from Ahmedabad earns 2000
PL/SQL procedure completed successfully
```

Data Types

The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values. We will focus on the **SCALAR** and the **LOB**data types in this chapter. The other two data types will be covered in other chapters.

S.No	Category & Description
1	**Scalar** Single values with no internal components, such as a **NUMBER, DATE,** or **BOOLEAN**.
2	**Large Object (LOB)** Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.
3	**Composite**

	Data items that have internal components that can be accessed individually. For example, collections and records.
4	**Reference** Pointers to other data items.

PL/SQL Scalar Data Types and Subtypes

PL/SQL Scalar Data Types and Subtypes come under the following categories −

S.No	Date Type & Description
1	**Numeric** Numeric values on which arithmetic operations are performed.
2	**Character** Alphanumeric values that represent single characters or strings of characters.
3	**Boolean** Logical values on which logical operations are performed.
4	**Datetime** Dates and times.

PL/SQL provides subtypes of data types. For example, the data type NUMBER has a subtype called INTEGER. You can use the subtypes in your PL/SQL program to make the data types

compatible with data types in other programs while embedding the PL/SQL code in another program, such as a Java program.

PL/SQL Numeric Data Types and Subtypes

Following table lists out the PL/SQL pre-defined numeric data types and their sub-types −

S.No	Data Type & Description
1	**PLS_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
2	**BINARY_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
3	**BINARY_FLOAT** Single-precision IEEE 754-format floating-point number
4	**BINARY_DOUBLE** Double-precision IEEE 754-format floating-point number
5	**NUMBER(prec, scale)** Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0
6	**DEC(prec, scale)** ANSI specific fixed-point type with maximum precision of 38

	decimal digits
7	**DECIMAL(prec, scale)** IBM specific fixed-point type with maximum precision of 38 decimal digits
8	**NUMERIC(pre, secale)** Floating type with maximum precision of 38 decimal digits
9	**DOUBLE PRECISION** ANSI specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
10	**FLOAT** ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
11	**INT** ANSI specific integer type with maximum precision of 38 decimal digits
12	**INTEGER** ANSI and IBM specific integer type with maximum precision of 38 decimal digits
13	**SMALLINT** ANSI and IBM specific integer type with maximum precision of

	38 decimal digits
14	**REAL** Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits)

Following is a valid declaration −

```
DECLARE
num1 INTEGER;
num2 REAL;
num3 DOUBLE PRECISION;
BEGIN
null;
END;
/
```

When the above code is compiled and executed, it produces the following result −

```
PL/SQL procedure successfully completed
```

PL/SQL Character Data Types and Subtypes

Following is the detail of PL/SQL pre-defined character data types and their sub-types −

S.No	Data Type & Description
1	**CHAR** Fixed-length character string with maximum size of 32,767 bytes

2	**VARCHAR2** Variable-length character string with maximum size of 32,767 bytes
3	**RAW** Variable-length binary or byte string with maximum size of 32,767 bytes, not interpreted by PL/SQL
4	**NCHAR** Fixed-length national character string with maximum size of 32,767 bytes
5	**NVARCHAR2** Variable-length national character string with maximum size of 32,767 bytes
6	**LONG** Variable-length character string with maximum size of 32,760 bytes
7	**LONG RAW** Variable-length binary or byte string with maximum size of 32,760 bytes, not interpreted by PL/SQL
8	**ROWID** Physical row identifier, the address of a row in an ordinary table

9	**UROWID** Universal row identifier (physical, logical, or foreign row identifier)

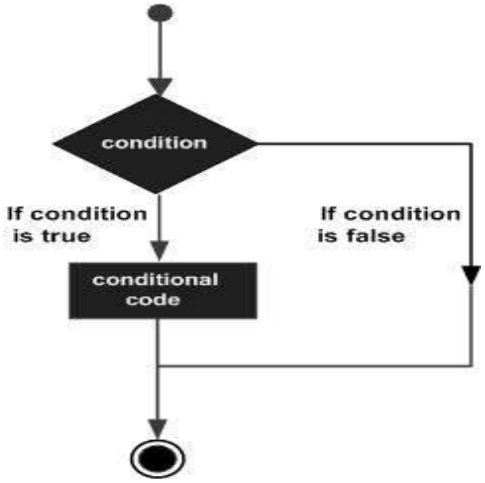
PL/SQL BOOLEAN DATA TYPES

The BOOLEAN data type stores logical values that are used in logical operations. The logical values are the Boolean values TRUE and FALSE and the value NULL.

Conditions

Decision-making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical conditional (i.e., decision making) structure found in most of the programming languages −



PL/SQL programming language provides following types of decision-making statements. Click the following links to check their detail.

S.No	Statement & Description
1	**IF - THEN statement** The **IF statement** associates a condition with a sequence of statements enclosed by the keywords **THEN** and **END IF**. If the condition is true, the statements get executed and if the condition is false or NULL then the IF statement does nothing.
2	**IF-THEN-ELSE statement** **IF statement** adds the keyword **ELSE** followed by an alternative sequence of statement. If the condition is false or NULL, then only the alternative sequence of statements get executed. It ensures that either of the sequence of statements is executed.
3	**IF-THEN-ELSIF statement** It allows you to choose between several alternatives.
4	**Case statement** Like the IF statement, the **CASE statement** selects one sequence of statements to execute. However, to select the sequence, the CASE statement uses a selector rather than multiple Boolean expressions. A selector is an expression whose value is used to select one of several alternatives.
5	**Searched CASE statement** The searched CASE statement **has no selector**, and it's WHEN clauses contain search conditions that yield Boolean values.
6	**nested IF-THEN-ELSE** You can use one **IF-THEN** or **IF-THEN-ELSIF**statement inside

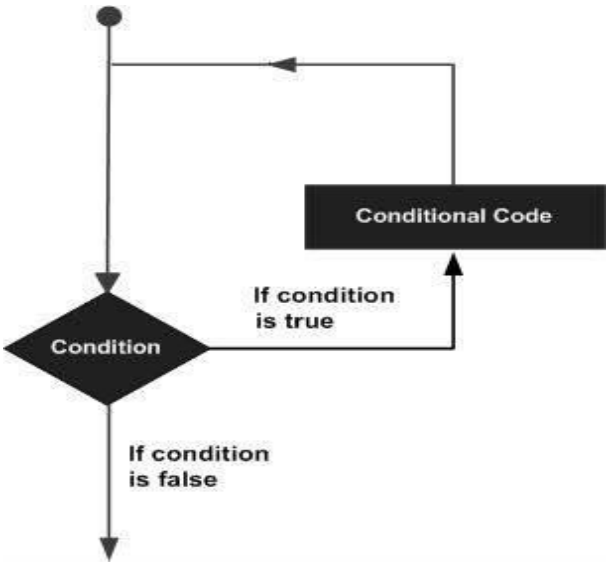
	another **IF-THEN** or **IF-THEN-ELSIF**statement(s).

LOOPING

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −



PL/SQL provides the following types of loop to handle the looping requirements. Click the following links to check their detail.

S.No	Loop Type & Description
1	**PL/SQL Basic LOOP** In this loop structure, sequence of statements is enclosed between

	the LOOP and the END LOOP statements. At each iteration, the sequence of statements is executed and then control resumes at the top of the loop.
2	**PL/SQL WHILE LOOP** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
3	**PL/SQL FOR LOOP** Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
4	**Nested loops in PL/SQL** You can use one or more loop inside any another basic loop, while, or for loop.

Labeling a PL/SQL Loop

PL/SQL loops can be labeled. The label should be enclosed by double angle brackets (<< and >>) and appear at the beginning of the LOOP statement. The label name can also appear at the end of the LOOP statement. You may use the label in the EXIT statement to exit from the loop.

The following program illustrates the concept −

```
DECLARE
i number(1);
j number(1);
BEGIN
<<outer_loop>>
  FOR i IN 1..3 LOOP
```

```
<<inner_loop>>
    FOR j IN 1..3 LOOP
dbms_output.put_line('i is: '|| i ||' and j is: '|| j);
END loop inner_loop;
END loop outer_loop;
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
i is: 1 and j is: 1
i is: 1 and j is: 2
i is: 1 and j is: 3
i is: 2 and j is: 1
i is: 2 and j is: 2
i is: 2 and j is: 3
i is: 3 and j is: 1
i is: 3 and j is: 2
i is: 3 and j is: 3

PL/SQL procedure successfully completed.
```

TYPE and % ROWTYPE

The %TYPE attribute provides the datatype of a variable or table column. This is particularly useful when declaring variables that will hold values of a table column. For example, suppose you want to declare variables as the same datatype as the employee_id and last_name columns in employees table. To declare variables named empid and emplname that have the same datatype as the table columns, use dot notation and the %TYPE attribute.

Using %TYPE With Table Columns in PL/SQL

```
DECLARE -- declare variables using %TYPE attribute
empidemployees.employee_id%TYPE;  -- employee_iddatatype is NUMBER(6)
emplnameemployees.last_name%TYPE;  -- last_namedatatype is VARCHAR2(25)
BEGIN
```

```
empid    := 100301;  -- this is OK because it fits in NUMBER(6)
--   empid  := 3018907;  -- this is too large and will cause an overflow
emplname := 'Patel'; --  this is OK because it fits in VARCHAR2(25)
   DBMS_OUTPUT.PUT_LINE('Employee ID: ' || empid);  -- display data
   DBMS_OUTPUT.PUT_LINE('Employee name: ' || emplname); -- display data
END;
/
```

Using the %ROWTYPE Attribute to Declare Variables

For easier maintenance of code that interacts with the database, you can use the %ROWTYPE attribute to declare a variable that represents a row in a table. A PL/SQL record is the datatype that stores the same information as a row in a table.

In PL/SQL, records are used to group data. A record consists of a number of related fields in which data values can be stored. The record can store an entire row of data selected from the table or fetched from a cursor or cursor variable.

```
DECLARE

customer_reccustomers%rowtype;

BEGIN

   SELECT * into customer_rec

   FROM customers

   WHERE id = 5;

dbms_output.put_line('Customer ID: ' || customer_rec.id);

dbms_output.put_line('Customer Name: ' || customer_rec.name);

dbms_output.put_line('Customer Address: ' || customer_rec.address);

dbms_output.put_line('Customer Salary: ' || customer_rec.salary);

END;

/
```

Cursors

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

- Implicit cursors
- Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK_ROWCOUNT** and **%BULK_EXCEPTIONS**, designed for use with the **FORALL**statement. The following table provides the description of the most used attributes −

S.No	Attribute & Description
1	**%FOUND**

	Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as **sql%attribute_name** as shown below in the example.

Example

We will be using the CUSTOMERS table we had created and used in the previous chapters.

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
```

```
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program will update the table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT**attribute to determine the number of rows affected −

```
DECLARE
total_rows number(2);
BEGIN
   UPDATE customers
   SET salary = salary +500;
   IF sql%notfound THEN
dbms_output.put_line('no customers selected');
   ELSIF sql%found THEN
total_rows:=sql%rowcount;
dbms_output.put_line(total_rows||' customers selected ');
END IF;
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
6 customers selected

PL/SQL procedure successfully completed.
```

If you check the records in customers table, you will find that the rows have been updated −

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
```

```
|  1 | Ramesh   |  32 | Ahmedabad |  2500.00 |
|  2 | Khilan   |  25 | Delhi     |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2500.00 |
|  4 | Chaitali |  25 | Mumbai    |  7000.00 |
|  5 | Hardik   |  27 | Bhopal    |  9000.00 |
|  6 | Komal    |  22 | MP        |  5000.00 |
+----+----------+-----+-----------+----------+
```

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is −

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor includes the following steps −

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

```
CURSOR c_customers IS
   SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

```
FETCH c_customers INTO c_id,c_name,c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

```
CLOSE c_customers;
```

Example

Following is a complete example to illustrate the concepts of explicit cursors

```
DECLARE
c_idcustomers.id%type;
c_namecustomerS.No.ame%type;
c_addrcustomers.address%type;
  CURSOR c_customersis
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customersintoc_id,c_name,c_addr;
    EXIT WHEN c_customers%notfound;
dbms_output.put_line(c_id||' '||c_name||' '||c_addr);
END LOOP;
```

```
   CLOSE c_customers;

END;

/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
1 Ramesh Ahmedabad
2 Khilan Delhi
3 kaushik Kota
4 Chaitali Mumbai
5 Hardik Bhopal
6 Komal MP

PL/SQL procedure successfully completed.
```

POSSIBLE QUESTIONS

UNIT-V

2 marks questions

1. Define PL/SQL.

2. What is meant by datatype?

3. Define Large Object.

4. Define Looping.

5. Define cursor.

6. What are the types of cursor?

6 marks questions

1. Explain about PL/SQL Block Structure.

2. Explain various types of datatypes.

3. Describe Conditions looping.

4. Elaborate TYPE and % ROWTYPE.

5. Explain cursor with examples.

Karpagam Academy of Higher Education
Department of CS
Subject-Oracle (SQL-PL/SQL) Class III B.Sc CS 2016-2019 Batch
Objective Type Questions

Unit-V

sno	Questions	opt1	opt2	opt3	opt4	Answer
1	Index which has an entry for some of key value is classified as	linear index	dense index	non dense index	cluster index	non dense index
2	Primary indexes, secondary indexes and cluster indexes are all types of	ordered indexes	unordered indexes	linear indexes	relative search indexes	ordered indexes
3	In multilevel indexes, primary index created for its first level is classified as	of multilevel index	third level of multilevel index	second level of multilevel index	first level of multilevel index	second level of multilevel index
4	Indexes which specifies address of records on disk with a physical pointer are classified as	structural index	hashing index	physical index	logical index	physical index
5	Example of non dense index	ternary index	secondary index	primary index	clustering index	clustering index
6	The method of access which uses key transformation is known as	direct	hash	random	sequential	hash

7	The physical location of a record is determined by a mathematical formula that transforms	B-Tree File	Hashed File	Indexed File	Sequential fil	Hashed File
8	What is the purpose of index in sql server	To enhance the query performanc e	To provide an index to a record	To perform fast searches	All of the mer	All of the mentioned
9	How many types of indexes	1	2	3	4	2
10	How non clustered index po	It never points to anything	It points to a data row	It is used for pointing data rows containing key values	None of the mentioned	It is used for pointing data rows containing key values
11	Which one is true about clus	Clustered index is not associated with table	Clustered index is built by default on unique key columns	Clustered index is not built on unique key columns	None of the mentioned	Clustered index is built by default on unique key columns
12	What is true about indexes?	Indexes enhance the performanc e even if the table is updated frequently	It makes harder for sql server engines to work to work on index which have large keys	It doesn’t make harder for sql server engines to work to work on index which have large keys	None of the n	It makes harder for sql server engines to work to work on index which have large keys

13	Does index take space in the	It stores memory as and when required	Yes, Indexes are stored on disk	Indexes are never stored on disk	Indexes take no space	Yes, Indexes are stored on disk
14	What are composite indexes	Are those which are composed by database for its internal use	A composite index is a combination of index on 2 or more columns	Composite index can never be created	None of the mentioned	A composite index is a combination of index on 2 or more columns
15	In _____ index	Clustered	Non clustered	Column store	Row store	Column store
16	A _____ inde	Clustered	Non Clustered	Covering	B-Tree	Covering
17	used to preserve the integrity of a document or a	Message dig	Message summ	Encrypted me	None of the n	Encrypted message
18	A hash function must meet _____ criteria.	Two	Three	Four	Five	Three
19	Index which has an entry for some of key value is classified as	linear index	dense index	non dense ind	cluster index	non dense index
20	In data file, first record of any of block is called	anchor recor	dense record	non dense rec	none of above	anchor record
21	File which has secondary index for its every field is classified as	fully inverte	fully indexed fi	secondary ind	primary index	fully inverted file
22	First field in primary index having same data type as in ordering field is considered as	indexed key	ternary key	secondary key	primary key	primary key

23	In multilevel indexes, primary index created for its second level is classified as	second level of multilevel index	first level of multilevel index	zero level of multilevel index	third level of multilevel index	third level of multilevel index
24	The SQL database language includes statements for:	Database de	Database mani	Database cont	All of the abo	All of the above.
25	A command to remove a relation from an SQL database	Delete table table name	Drop table table name	Erase table table name	Alter table table name	Drop table table name
26	Which SQL Query is use to remove a table and all its data from the database?	Create Table	Alter Table	Drop Table	None of these	Drop Table
27	A type of query that is placed within a WHERE or HAVING clause of another query is called	Super query	Sub query	Master query	Multi-query	Sub query
28	Aggregate functions are functions that take a _____ as input and return a single value.	Collection o	Single value	Aggregate val	Both a & b	Collection of values
29	Which of the following should be used to find the mean of the salary ?	Mean(salary	Avg(salary)	Sum(salary)	Count(salary)	Avg(salary)
30	All aggregate functions except _____ ignore null values in their input collection.	Count(attrib	Count(*)	Avg	Sum	Count(*)

31	A Boolean data type that can take values true, false, and_____ .	1	0	Null	Unknown	Unknown
32	If we do want to eliminate duplicates, we use the keyword _____in the aggregate expression.	Distinct	Count	Avg	Primary key	Distinct
33	The ____ connective tests for set membership, where the set is a collection of values produced by a select clause.	Or	Not in	In	and	In
34	The ____ connective tests for the absence of set membership.	Or	Not in	In	and	Not in
35	We can test for the nonexistence of tuples in a subquery by using the _____ construct.	Not exist	Not exists	Exists	Exist	Not exists
36	Dates must be specified in the format	mm/dd/yy	yyyy/mm/dd	dd/mm/yy	yy/dd/mm	yyyy/mm/dd
37	Which of the following is used to store movie and image files ?	Clob	Blob	Dlob	None of the a	Blob
38	Entities are identified from the word statement of a problem by	picking words which are adjectives	picking words which are nouns	picking words which are verbs	picking words which are pronouns	picking words which are nouns

39	Relationships are identified from the word statement of a problem by	picking words which are adjectives	picking words which are nouns	picking words which are verbs	picking words which are pronouns	picking words which are pronouns
40	One entity may be	related to only one other entity	related to itself	related to only two other entities	related to many other entities	related to many other entities
41	By relation cardinality we mean	number of items in a relationship	number of relationships in which an entity can appear	number of items in an entity	number of related occurrences for each of the two entities	number of related occurrences for each of the two entities
42	If an entity appears in only one relationship then it is	a 1:1 relationship	a 1:N relationship	a N:1 relationship	a N:M relationship	a 1:1 relationship
43	If an entity appears in N relationships then it is	a 1:1 relationship	a 1:N relationship	a N:1 relationship	a N:M relationship	a N:M relationship
44	If an entity appears in not more than 5 relationships then it is a	1:1 relationship	1:5 relationship	5:1 relationship	5:5 relationship	1:5 relationship
45	A relation is	an entity	a relationship	members of a relationship set	members of an entity set or a relationship set	an entity
46	Rows of a relation are called	tuples	a relation row	a data structure	an entity	tuples
47	The database schema is written in	HLL	DML	DDL	DCL	DDL

48	The way a particular application views the data from the database that the application uses is a	module	relational mode	schema	sub schema	sub schema
49	The relational model feature is that there	is no need for primary key data	is much more data independence than some other database models	are explicit relationships among records.	are tables with many dimensions	is much more data independence than some other database models
50	Which of the following are the properties of entities?	Groups	Table	Attributes	Switchboards	Attributes
51	Which database level is closest to the users?	External	Internal	Physical	Conceptual	External
52	Which are the two ways in which entities can participate in a relationship?	Passive and	Total and partia	Simple and C	All of the abo	Total and partial
53	…….. data type can store unstructured data	RAW	CHAR	NUMERIC	VARCHAR	RAW
54	insert into <table_name> (column list) values <list of values>	TRUE	FALSE	Table	None	TRUE
55	……. first proposed the process of normalization in DBMS.	Edgar. W	Edgar F. Codd	Edward Steph	Edward Codd	Edgar F. Codd
56	Which of the following is not comparison operator?	<>	<	=<	>=	=<

57	An outstanding functionality of SQL is its support for automatic …… to the target data.	programmin	functioning	navigation	notification	notification
58	……..specifies a search condition for a	GROUP BY	HAVING Clau	FROM Clause	WHERE Clau	HAVING Clause
59	Drop Table cannot be used to drop a table referenced by a …… constraint.	Local Key	Primary Key	Composite Ke	Foreign Key	Foreign Key
60	The user defined data type can be created using	Create datat	Create data	Create definet	Create type	Create type

Register Number_____
[16CSU504A]

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore-641021.
B.Sc COMPUTER SCIENCE
FIRST INTERNAL EXAMINATION - JULY 2018
Fifth Semester
Oracle (SQL/PL-SQL)

Date & Session: .7.2018 & N **Duration: 2 Hours**
Maximum : 50 Marks **Class : III-B. Sc(CS) A & B**

SECTION A – (20 X 1 = 20 Marks)
ANSWER ALL THE QUESTIONS

1. A _____ is used to store and retrieve related information.
   a. **Database** b. Record c. Data d. Procedures
2. Multiuser environment deals with_____.
   a. Access b. Application c. Integrity d. **Concurrent access**
3. Oracle is a _____ generation relational database management system.
   a. Third b. **Fourth** c. Fifth d. Second
4. _____ is a procedural language extension to SQL.
   a. **PL/SQL** b. DDL c. DML d. DCL
5. iSQL*Plus supports_____.
   a. Windows GUI b**.Web based user interface**
   c. LINUX based interface d. UNIX UI
6. SQL is invented by _____.
   a. **IBM** b. Microsoft c. Oracle d. Intel
7. _____ is used to retrieve, store, modify data in a database.
   a. DDL b. TCL c. DCL d. **DML**
8. _____ is the operator used to combine two conditions.
   a. AS b. BETWEEN c. **AND** d. CASE
9. The text data type CHAR() can hold up to _____ characters.
   a. **255** b. 254 c. 65535 d. 65536
10. The _____ operator is used to filter the result set within a certain range.
    a. JOIN b. HAVING c. **BETWEEN** d. LIKE
11. _____ is an interactive tool with scripting capabilities.
    a. **SQL*Plus** b. Html c. Xml d. Report
12. _____ is an ANSI and ISO standard language.
    a. PL/SQL b. **SQL** c. SQL*Plus d. Html
13. DCL is used to create _____ statements.
    a. Integrity b. Constraint c. **Grant** d. Commit
14. Rollback satatement is an example for _____.
    a. PL b. SQL c. Oracle d. **TCL**
15. _____ is a combination of values, operators and functions.
    a. Operators b. **Expressions** c. Operands d. Statements

16. Boolean expressions fetch the data based on matching _____.
    a. **Single value** b. Two values c. Three values d. Null value
17. Example for the aggregate data calculation is _____.
    a. **count()** b. as() c. like() d. group()
18. Selection Operation is used to ____from a relation.
    a. Select the Column **b. Select the row**
    c. Select the table d. Select DMBS
19. _____command is used to remove the table definition information.
    a. Delete b. Remove c. Destroy **d. Drop**
20. What does SQL stand for?
    a. Structured Query Language b. Similar Query Language
    c. Simple Query Language d. Structure Query Language

SECTION – B (3 X 2 =6 Marks)
ANSWER ALL THE QUESTIONS

21. Define SQL*Plus.

22. What is meant by SQL expression?

23. What is meant by DML?

SECTION – C (3 X 8 =24 Marks)
ANSWER ALL THE QUESTIONS

24. (a) Differentiate SQL and SQL*Plus. [OR]

    (b) List and explain SQL commands.

25. (a) Describe about SQL datatypes. [OR]

    (b) Explain in detail about SQL expressions.

26. (a) Elaborate SQL*Plus. [OR]

    (b) Explain DDL commands with example.