Narration Script:

Slide 1: Oracle Database 11g Express Edition – Working with Database Objects -II

Hello and welcome back to the second part of this online, self-paced course titled Oracle Database 11*g* Express Edition – Working with Database Objects . My name is Anupama Mandya, and I will be guiding you through this course.

This course is designed to follow the first part of the course – Working with Database Objects

Slide 2: Course Objectives

Before we begin, let us look at the objectives of this session. At the end of this session, you should be able to:

- Access Data from the Database and
- Manipulate Data within the Database Tables

Slide 3: Accessing Data

Introduction

Here, you learn how to retrieve data from tables. The SQL SELECT statement is used to access and report data back from the Express Edition tables. This is known as "querying" the data. In Express Edition, you can either write SELECT statements using the SQL Command Line Utility tool, or you can use the Query Builder tool in SQL Developer to build queries with a GUI interface. You will learn how to write several reports using these tools.

Overview: The SQL SELECT statement is used to access and report data back from the Express Edition tables. This is known as "querying" the data.The segments around the core here represent the different ways of retrieving data.

Build Queries: All operations on information in an Oracle database are performed using SQL statements. A SQL statement is a string of SQL text.SQL statements are used to access data from existing tables. A query, or SQL SELECT statement, selects data from one or more existing tables or views. With the SELECT statement, you can choose to see all or some of the data in a table. The SELECT statement has three capabilities

Selection : This provides information by identifying rows

Projection : This provides information by identifying the columns

Join : This provides information by retrieving data from multiple tables

You will use the SQL SELECT statement to:

1. Write queries

2.Create SELECT statements that restrict rows

3.Sort output

4.Display data from many tables and

5.Customize output with functions embedded in the SELECT statement

In the SQL*Plus environment, you can enter a query after the SQL> prompt.

In the SQL Developer environment, you can enter a query in the SQL Worksheet.

Join Tables: Sometimes you need to display data from more than one table. To do this, you use SELECT statements. Suppose that you want to select data from multiple tables, you use a JOIN between the tables involved. This provides information from more than one table. Because information comes from more than one table, this is called a JOIN between the tables involved.

For example, in the EMPLOYEES table, the DEPARTMENT_ID column represents the department number for an employee. In the DEPARTMENTS table, there is a DEPARTMENT_ID column as well as a DEPARTMENT_NAME column. You can join the EMPLOYEES and DEPARTMENTS tables by using the DEPARTMENT_ID column to produce a report that shows the employees' names and department names.

Functions to Build Reports: SQL functions provide a powerful way to perform operations when retrieving data from a table. For example, you may need to display employee names in uppercase in a report. Or you may need to display employees' hire dates with the name of the month spelled out.

Many functions work on specific data types. Each value manipulated by Oracle Database XE has a data type. Data types provide a way to define the behavior of data. When you create a table, you must specify a data type for each of its columns.

The main types of data that functions work on are:

Character

Number

Date and

NVL

Work with Groups of Data: You can use group functions in a SQL statement to display information about groups of rows in the database. A group function performs an operation on a set of data. Group functions operate on sets of rows and return one result per group. Group functions are also called multiple-row functions (in contrast with single-row functions, which return one result for each row). You can use a group function to operate on a set of data. The set may be an entire table or only part of the table.

Slide 4: Building Queries

Writing a Basic Query: You may want to see data in the database tables. For ex: You would like to view data in the EMPLOYEES table. You also want to sort data based on your requirements. You can do this by writing a basic query using the SELECT statement. When you write this query, two mandatory clauses in the SELECT statement syntax are required: a SELECT clause and a FROM clause. The syntax for a SELECT statement is as shown. Here,

SELECT - Lists one or more columns

DISTINCT - Eliminates the display of duplicate rows

* - Selects all columns

COLUMN - Selects the named column

ALIAS - Gives selected columns a customized heading and

FROM - Identifies the table or tables containing the columns

The SELECT clause specifies the columns that you want to display. The FROM clause specifies the tables that contain those columns.

Retrieving Columns: You can use the projection capability of SQL to choose the columns in a table that you want to retrieve. You can retrieve selected columns or all columns from a table.

Retrieving Rows: You can use the selection capability of SQL to choose the rows that you want to retrieve from a table. You can specify various criteria to select the rows that you want to see. By inserting a WHERE clause into a SQL statement, you can specify a condition that must be met, and only the rows that meet the condition are returned.

When using a WHERE clause:

- Make sure, The WHERE clause directly follows the FROM clause in the SQL statement syntax.
- Also,The WHERE clause should consist of the WHERE keyword and a condition or conditions.
- and The condition in a WHERE clause should specify a comparison of values that limits the rows that are returned by a query

Syntax for the where clause is as shown:

The condition consists of three components:

1. Column name

2. Comparison operator

3. Element that can be another column name, constant, or list of values

Slide 5: The following are some demonstrations and simulations about the Building Queries topic where some procedural tasks are demonstrated in a step-by-step fashion and you are also invited to perform some steps based on the instructions given to you.

Slide 9: Let us now take a brief pause by taking a Quiz

Slide 11: Joining Tables

Joins

In this topic, you learn about Joins. A natural join enables you to display data from two tables when a value in one column of one table corresponds directly to a value in another column in the second table. In a natural join, the two tables include one or more columns that have the same name and data types. A natural join retrieves all rows from the two tables that have equal values in all matched columns. Frequently, this type of join involves primary key and foreign key columns.

Syntax

SELECT [DISTINCT] * | column [alias], ...

FROM table NATURAL JOIN ;

Joining Two Tables with a USING Clause

The USING clause enables you to specify the columns to be used for a join between two tables. The column names must be the same for both tables and must have compatible data types. Use the USING if your tables contain more than one column whose names match to explicitly identify the name of the columns that you want to join. The syntax is as shown

Joining Tables and Identifying Columns:

You use the ON clause to specify the join condition when joining two tables or joining a table to itself. This enables you to separate the join condition from any search or filter conditions in the WHERE clause. The column names need not match between the tables; however, the data types must match.

For example, you might need to evaluate the hire dates and start dates of all employees. You can find the hire dates in the EMPLOYEES table and the start dates in the JOB_HISTORY table. These two columns are named differently. The ON clause is as follows:

FROM employees JOIN job_history

ON employees.hire_date = job_history.start_date

The syntax is as shown

Applying Additional Conditions to a Join: You can apply additional conditions to the join. For example, you may want to join the EMPLOYEES and DEPARTMENTS tables and, in addition, display only employees who have a manager ID of 149. To add additional conditions to the ON clause, you can add AND clauses. Alternatively, you can use a WHERE clause to apply additional conditions.

Here is an example for using the AND Clause

```
SELECT e.employee_id, e.last_name, e.department_id,
    d.department_id, d.location_id
FROM employees e JOIN departments d
ON (e.department_id = d.department_id)
AND e.manager_id = 149
```

An example Using the WHERE clause is

```
SELECT e.employee_id, e.last_name, e.department_id,
    d.department_id, d.location_id
FROM employees e JOIN departments d
ON (e.department_id = d.department_id)
WHERE e.manager_id = 149
```

Aliases

In the examples shown, aliases are used to identify the table names. In the FROM clause, an abbreviation is provided after the table name. This is called an alias. After the alias is set up in the FROM clause, you can refer to it throughout the statement.

Joining Multiple Tables: A three-way join is a join of three tables. You can join as many tables as needed to retrieve information. For example, you might want to find employees, their dependents' names, and the department names for those employees. This requires accessing three tables:- EMPLOYEES, DEPENDENTS, and DEPARTMENTS

The ON clause can also be used to join columns that have different names (in the same table or in a different table). For example, you can perform a self-join of the EMPLOYEES table based on the EMPLOYEE_ID and MANAGER_ID columns.

Slide 12: Following are the demonstrations for the JOINING Tables topic

Slide 17: Let us now take a brief pause by taking a Quiz

Slide 19: In this topic, we discuss how to customize reports using SQL Functions. The different types of functions discussed here are Character Functions, Numeric Functions, DateTime Functions, Conversion Functions and NVL(Null Value) Function.

Character Functions: Character functions accept character input. Most return character values, but some return numeric values. Character functions that return character values return values of the same data type as their input argument. Character functions that return number values can take any character data type as their input argument.

There are two types of single-row character functions:

1. Case-manipulation functions: You use case-manipulation functions to change the case of character strings. The types of case-manipulation functions are LOWER, UPPER , and INITCAP

2. Character-manipulation functions: You use character-manipulation functions to manipulate character strings. The types of character-manipulation functions are CONCAT, SUBSTR , INSTR , LPAD , RPAD , TRIM , and REPLACE .

The syntax for functions is as shown. Here

function name - the name of the function

Column - any named database column

Expression - any string or calculated expression

arg1,arg2 - any arguments used by the function

The SUBSTR function accepts as arguments a string, a character position, and a length, and returns the substring that starts at the specified position in the string and has the specified length.

Numeric Functions: Numeric Functions are used to accept numeric input. The numeric functions return numeric values. Each numeric function returns a single value for each row that is evaluated.

The three most commonly used numeric functions are Round, Mod and Trunc

ROUND: Rounds a column, an expression, or a value to n decimal places

MOD: Returns the remainder of x divided by y

TRUNC: Truncates a column, an expression, or a value to n decimal places

DateTime Functions: Datetime functions operate on date, timestamp, and interval values. Each datetime function returns a single value for each row that is evaluated. Oracle database 11g XE stores dates in an internal format representing the century, year, month, day, hours, minutes, and seconds.

For example, if today is 20th June,2011, at 7:10 in the morning, the current date is stored internally as June 20, 2011, 07:10

The default display and input format for any date in XE is DD-MON-YY.Here, DD stands for the day of the month, MON stands for the first three letters of the month name, and YY stands for the year (which also includes the century). Valid Oracle dates are between January 1, 4712 B.C., and December 31, A.D. 9999

Lets look at some of the Common Date Functions: Date functions enhance your ability to calculate information about dates. Common DATE functions are

- MONTHS_BETWEEN - Finds number of months between two dates
- ADD_MONTHS - Adds months to a date
- NEXT_DAY - Finds the next day
- LAST_DAY - Finds the last day of the month
- TRUNC - Truncates a date
- ROUND - Rounds a date

SYSDATE is a function that contains no arguments. You can use SYSDATE just as you use a column name. The SYSDATE is displayed for every row in the table used.

Conversion Function: Conversion functions convert one data type to another. You may want to convert Dates to Characters or Numbers to Characters using standard Formats. This is done using the TO_CHAR function. The query in Example uses the TO_CHAR function to convert HIRE_DATE values to character values that have the two standard formats DS (Date Short) and DL (Date Long).

You can also convert characters to numbers using the TO_NUMBER function. The query in Example uses the TO_NUMBER function to convert POSTAL_CODE values (which are of type VARCHAR2) to values of type NUMBER, which it uses in calculations.

NVL Function: The NULL-related functions facilitate the handling of NULL values. An empty field is said to contain NULL. A NULL is defined as a value that is unavailable, unassigned, unknown, or inapplicable. A NULL is not the same as a zero or a space. Zero is a number, and a space is a character. Columns of any data type can contain NULLs unless the column was defined as NOT NULL when the table was created. Also, primary key columns cannot have NULLs. The syntax for a Null Value Function is as shown

Slide 20: Following are some demonstrations and simulations about Using Functions to Customize Reports topic where some procedural tasks are demonstrated in a step-by-step fashion and you are also invited to perform some steps based on the instructions given to you.

Slide 25: Let us now take a brief pause by taking a Quiz

Slide 27: Working with Groups of Data

Grouping Functions: To display information about groups of rows in the database, you can use Group Functions in a SQL statement. Group functions operate on sets of rows and return one result per group.

In this topic, you learn about the various group functions that are available for use. You also learn how to divide data into groups using the GROUP By clause. To restrict groups that are retrieved, the HAVING Clause can be used. There are several group functions that are available for use. All are ANSI-standard SQL functions.

The various types of SQL GROUP Functions are :

- AVG
- SUM
- MAX
- MIN
- STDDEV
- VARIANCE
- COUNT

You can find the average of a group of values by using the AVG function.

You can find the total of a group of values by using the SUM function.

MAX finds a largest value. MIN returns a smallest value.

STDDEV returns a standard deviation. VARIANCE returns the statistical variance.

AVG , SUM , STDDEV , and VARIANCE can be used with numerical data only. You can use MAX and MIN on character, numeric, and date data types.

You can use the COUNT function to return the number of values in a column.

The syntax for using a group function is as shown

You can use the SQL GROUP functions in SELECT statements to:

Find the number of rows in a table

Add the salaries

Find the average, maximum, and minimum salaries and

Analyze salary information by department

To specify a group function, you write the name of the function followed by an argument in parentheses. The argument can be a column name, an expression, or a constant.

The COUNT function returns the number of non-null values in a column. If you count the values of a primary key column, you will find the number of rows in a table because a primary key column cannot contain nulls.

Dividing Data Into Groups: You might need to divide a table of information into groups to produce meaningful results when using group functions. For example, you might want to find the average salary for each department or the oldest hire in each job category.

You use the GROUP BY clause to organize rows in a table into groups. You can then use the group functions to return information for each group. For example, you can group data in the EMPLOYEES table by department number and then return the average salary for each department. The syntax is as shown

You place the GROUP BY clause between the WHERE and ORDER BY clauses in a SQL statement. You specify the GROUP BY keyword followed by a column name or expression that indicates how you want to group the rows

Here are some Guidelines:

The SELECT clause can contain only the column or columns that appear in the GROUP BY clause, in addition to any group function or functions.

The GROUP BY clause must contain any column or columns that appear in the SELECT clause that are not listed in the group function. By default, rows are sorted in ascending order of the columns that are included in the GROUP BY list. The ORDER BY clause is optional. If used, it overrides the default sorting.

Restricting Groups: In the same way that you use the WHERE clause to restrict rows that are retrieved, you can use the HAVING clause to restrict groups that are retrieved. The syntax for using the HAVING clause is as shown. The HAVING clause is placed immediately after the GROUP BY clause in the syntax. You specify the HAVING keyword followed by a condition that refers to a group of rows. You cannot use a HAVING clause in a statement without a GROUP BY clause.

Slide 28: Following is a simulation and demonstration about the topic Working with Groups of Data

Slide 31: Let us now take a brief pause by taking a quiz

Slide 33: Manipulating Data

Introduction: Data manipulation language (DML) statements access and manipulate data in existing tables. You can add data to a table by using the INSERT statement; you can modify data using an UPDATE statement and also delete unwanted data using a DELETE statement.

The effect of a DML statement is not permanent until you commit the transaction that includes it. A **transaction** is a sequence of SQL statements that Oracle Database Express Edition treats as a unit. Until a transaction is committed, it can be rolled back (undone).

All these changes can be made using one of these tools:

In the SQL*Plus environment, you can enter a statement after the SQL>prompt

In the SQL Developer environment, you can enter a transaction control statement in the SQL Worksheet.

Overview: The segments around the core describe these various Data Manipulation Statements

Add Data: Oracle Database 11g Express Edition allows you to store data by inserting rows of information into them. You can add data into the table using an INSERT statement.

Modify Data: During everyday operations in a company, data constantly requires changes. You can make changes to existing records in a table by performing an UPDATE operation.

Remove Data: You can remove data from the database at any time by deleting the data from a table. This may be required to remove redundant or duplicate information. When removing data, you must ensure that dependent data is also updated as needed. The DELETE statement is used to remove data

Save and Discard Changes: Transactional mode is a stateful transaction mode in which you can, for example, perform an update and then select data for review before issuing a COMMIT to save changes. When making changes to the data, you can choose to make changes permanent by performing a COMMIT. You can also choose to discard your changes by performing a ROLLBACK.

Slide 34: Adding Data

Adding Data: In this topic, you learn about Adding Data into a database table by Inserting Rows of Information into them. The INSERT statement inserts rows into an existing table. Before you insert a row into a table, you must know what columns the table has and what their valid values are. The records being inserted must match the structure of the table. Each record should have values that match the different columns in the table. Data and constraints are validated at the time of insert, and the data is automatically saved.

Inserting A Row: You can also use an INSERT INTO statement to insert one record at a time into a table. A sample INSERT INTO syntax is as shown. Here, table_name in the INSERT statement specifies the name of the table to which you are adding a row. Notice that the INSERT keyword is always followed by the INTO clause. The identifiers col1, col2 , and coln identify the column names into which you are inserting values. If you omit this parameter, the server assumes that you will provide a value or expression in the VALUES clause for every column in the table.

The VALUES clause specifies the data values to be inserted in the corresponding columns. The values must be specified in the same sequence as the column list in the INSERT INTO clause. The data type of

the inserted value must match the data type for the column. You must enclose date and character values in single quotation marks.

Any mandatory or NOT NULL columns must be specified during inserts.

Slide 35: Following is a demonstration about this topic Adding Data to the Database

Slide 37: Modify Data

About Modifying Data: You can modify records in a table using the UPDATE statement. While performing an UPDATE, you need to specify the column(s) and row that you are modifying as well as the new value(s). You can use one UPDATE statement to modify multiple records in a table based on a specific condition. When a condition is not specified, the entire table is updated**.**

Syntax of an Update Statement: The syntax of the Update statement is as shown. Each value must be valid for its column_name. If you include the WHERE clause, the statement updates column values only in rows that satisfy the condition.

Slide 38: Following are a demonstration and simulation about the topic Modifying Data where some procedural tasks are demonstrated in a step-by-step fashion and you are also invited to perform some steps based on the instructions given to you.

Slide 41: Remove Data

Removing Data from a Table: In this module, you learn how to remove data from a database table. You can delete rows from a table using the DELETE statement. You can use the Oracle SQL Developer interface to remove rows. You can also remove rows from a table by using the DELETE statement, which does the following: It

- Allows you to remove all rows from a table
- Enables you to delete rows conditionally
- Prevents you from deleting rows that are protected by referential integrity constraints

Delete Statement: The syntax of the DELETE statement is as shown

Use of the keyword FROM is optional. The FROM keyword is used to improve readability.

table_name identifies the name of the table from which the rows are to be deleted.

The WHERE clause specifies the condition that identifies the rows to be deleted. The WHERE clause can accept any SQL condition, including a subquery. If you omit the WHERE clause, all rows in the table are deleted.

Oracle SQL Developer allows you to remove rows very easily. However, each row is removed individually. To remove multiple rows with a single DELETE statement, you can use the Oracle SQL Developer Worksheet or Query Builder interfaces.

Slide 42: Following is a demonstration about Deleting Rows of Data from a Table

Slide 44: Save and Discard Changes

About Transaction Control Statements: In this module, you learn what Transaction Control statements. What is a Transaction? A transaction is a sequence of one or more SQL statements that Oracle Database XE treats as a unit: either all of the statements are performed, or none of them are. You need transactions to model business processes that require that several operations be performed as a unit. For example, when a manager leaves the company, a row must be inserted into the JOB_HISTORY table to show when the manager left, and for every employee who reports to that manager, the value of MANAGER_ID must be updated in the EMPLOYEES table. To model this process in an application, you must group the INSERT and UPDATE statements into a single transaction.

The basic transaction control statements are:

SAVEPOINT, which marks a savepoint in a transaction-a point to which you can later roll back. Savepoints are optional, and a transaction can have multiple savepoints.

COMMIT, which ends the current transaction, makes its changes permanent, and releases its locks.

ROLLBACK, which rolls back (undoes) either the entire current transaction or only the changes that are made after a specified Savepoint.

In the SQL Developer environment, you can enter a transaction control statement in the SQL Worksheet. SQL Developer also has Commit Changes and Rollback Changes icons

Committing Transactions: Committing a transaction makes its changes permanent, erases its savepoints, and releases its locks. To explicitly commit a transaction, use either the COMMIT statement or (in the SQL Developer environment) the Commit Changes icon.

Before you commit a transaction:

■ Your changes are visible to you, but not to other users of the database instance.

■ Your changes are not final-you can undo them with a ROLLBACK statement.

After you commit a transaction:

■ Your changes are visible to other users, and to their statements that run after you commit your transaction

■ Your changes are final-you cannot undo them with a ROLLBACK statement

Rolling Back: Rolling back a transaction undoes its changes. You can roll back the entire current transaction, or you can roll it back only to a specified savepoint. To roll back the current transaction only to a specified savepoint, you must use the ROLLBACK statement with the TO SAVEPOINT clause. To roll back the entire current transaction, use either the ROLLBACK statement without the TO SAVEPOINT clause or (in the SQL Developer environment) the Rollback Changes icon.

Rolling back the entire current transaction:

■ Ends the transaction

■ Reverses all of its changes

■ Erases all of its savepoints

■ Releases any transaction locks

Other transactions that have requested access to rows locked after the specified savepoint must continue to wait until the transaction is either committed or rolled back. Other transactions that have not requested the rows can request and access the rows immediately.

To see the effect of a rollback in SQL Developer, you might have to click the Refresh icon.

Slide 45: Following are a simulation and demonstration about the topic Transaction Control Statements.

Slide 48: Let us now take a brief pause by taking a quiz

Slide 50: Course Review

Well, we have now come to the end of this session. In this session, we talked about Working with Database Objects which included

- Accessing Data from the Database and
- Manipulating Data within the Database Tables

Slide 51: How Can I Learn More?

Oracle offers a variety of additional channels from which you can learn more about Oracle Database 11g Express Edition or about any Oracle products. We at Oracle Education know your time is valuable and limited, and so we thank you for participating in this self-paced training. We hope this course has met your expectations and learning objectives, and wish you the best of luck in all of your endeavors. Thanks again.