



Integrated Cloud Applications & Platform Services



Oracle Database 12c R2: SQL Workshop I

Student Guide - Volume II

D80190GC20

Edition 2.0 | April 2017 | D98630

Learn more from Oracle University at education.oracle.com

Author

Apoorva Srinivas

**Technical Contributors
and Reviewers**

Nancy Greenberg
Suresh Rajan
Peckkwai Yap
Bryan Roberts
Sharath Bhujani

Editors

Aju Kumar
Chandrika Kennedy
Kavita Saini

Graphic Designers

Prakash Dharmalingam
Kavya Bellur

Publishers

Asief Baig
Giri Venugopal
Jayanthy Keshavamurthy
Raghunath M
Srividya Rameshkumar
Veena Narasimhan

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction

- Lesson Objectives 1-2
- Lesson Agenda 1-3
- Course Objectives 1-4
- Course Roadmap 1-5
- Appendices and Practices Used in the Course 1-8
- Lesson Agenda 1-9
- Oracle Database 12c: Focus Areas 1-10
- Oracle Database 12c 1-11
- Lesson Agenda 1-13
- Relational and Object Relational Database Management Systems 1-14
- Data Storage on Different Media 1-16
- Relational Database Concept 1-17
- Definition of a Relational Database 1-18
- Data Models 1-19
- Entity Relationship Model 1-20
- Entity Relationship Modeling Conventions 1-22
- Relating Multiple Tables 1-24
- Relational Database Terminology 1-26
- Lesson Agenda 1-28
- Human Resources (HR) application 1-29
- Tables Used in This Course 1-30
- Tables Used in the Course 1-32
- Lesson Agenda 1-33
- Using SQL to Query Your Database 1-34
- How SQL Works 1-35
- SQL Statements Used in the Course 1-36
- Development Environments for SQL 1-37
- Introduction to Oracle Live SQL 1-38
- Lesson Agenda 1-39
- Oracle Database Documentation 1-40
- Additional Resources 1-41
- Summary 1-42
- Practice 1: Overview 1-43

2 Retrieving Data Using the SQL SELECT Statement

- Course Roadmap 2-2
- Objectives 2-3
- Lesson Agenda 2-4
- Basic SELECT Statement 2-6
- Selecting All Columns 2-7
- Selecting Specific Columns 2-8
- Selecting from DUAL 2-9
- Writing SQL Statements 2-10
- Column Heading Defaults 2-11
- Lesson Agenda 2-12
- Arithmetic Expressions 2-13
- Using Arithmetic Operators 2-14
- Operator Precedence 2-15
- Defining a Null Value 2-16
- Null Values in Arithmetic Expressions 2-17
- Lesson Agenda 2-18
- Defining a Column Alias 2-19
- Using Column Aliases 2-20
- Lesson Agenda 2-21
- Concatenation Operator 2-22
- Literal Character Strings 2-23
- Using Literal Character Strings 2-24
- Alternative Quote (q) Operator 2-25
- Duplicate Rows 2-26
- Lesson Agenda 2-27
- Displaying Table Structure 2-28
- Using the DESCRIBE Command 2-29
- Quiz 2-30
- Summary 2-31
- Practice 2: Overview 2-32

3 Restricting and Sorting Data

- Course Roadmap 3-2
- Objectives 3-3
- Lesson Agenda 3-4
- Limiting Rows by Using a Selection 3-5
- Limiting Rows That Are Selected 3-6
- Using the WHERE Clause 3-7
- Character Strings and Dates 3-8
- Comparison Operators 3-9

Using Comparison Operators 3-10
Range Conditions Using the BETWEEN Operator 3-11
Using the IN Operator 3-12
Pattern Matching Using the LIKE Operator 3-13
Combining Wildcard Symbols 3-14
Using NULL Conditions 3-15
Defining Conditions Using Logical Operators 3-16
Using the AND Operator 3-17
Using the OR Operator 3-18
Using the NOT Operator 3-19
Lesson Agenda 3-20
Rules of Precedence 3-21
Lesson Agenda 3-23
Using the ORDER BY Clause 3-24
Sorting 3-25
Lesson Agenda 3-27
SQL Row Limiting Clause 3-28
Using SQL Row Limiting Clause in a Query 3-29
SQL Row Limiting Clause: Example 3-30
Lesson Agenda 3-31
Substitution Variables 3-32
Using the Single-Ampersand Substitution Variable 3-34
Character and Date Values with Substitution Variables 3-36
Specifying Column Names, Expressions, and Text 3-37
Using the Double-Ampersand Substitution Variable 3-38
Using the Ampersand Substitution Variable in SQL*Plus 3-39
Lesson Agenda 3-40
Using the DEFINE Command 3-41
Using the VERIFY Command 3-42
Quiz 3-43
Summary 3-44
Practice 3: Overview 3-45

4 Using Single-Row Functions to Customize Output

Course Roadmap 4-2
Objectives 4-3
HR Application Scenario 4-4
Lesson Agenda 4-5
SQL Functions 4-6
Two Types of SQL Functions 4-7
Single-Row Functions 4-8

Lesson Agenda	4-10
Character Functions	4-11
Case-Conversion Functions	4-13
Using Case-Conversion Functions	4-14
Character-Manipulation Functions	4-15
Using Character-Manipulation Functions	4-16
Lesson Agenda	4-17
Nesting Functions	4-18
Nesting Functions: Example	4-19
Lesson Agenda	4-20
Numeric Functions	4-21
Using the ROUND Function	4-22
Using the TRUNC Function	4-23
Using the MOD Function	4-24
Lesson Agenda	4-25
Working with Dates	4-26
RR Date Format	4-27
Using the SYSDATE Function	4-29
Using the CURRENT_DATE and CURRENT_TIMESTAMP Functions	4-30
Arithmetic with Dates	4-31
Using Arithmetic Operators with Dates	4-32
Lesson Agenda	4-33
Date-Manipulation Functions	4-34
Using Date Functions	4-35
Using ROUND and TRUNC Functions with Dates	4-36
Quiz	4-37
Summary	4-38
Practice 4: Overview	4-39

5 Using Conversion Functions and Conditional Expressions

Course Roadmap	5-2
Objectives	5-3
Lesson Agenda	5-4
Conversion Functions	5-5
Implicit Data Type Conversion of Strings	5-6
Implicit Data Type Conversion to Strings	5-7
Explicit Data Type Conversion	5-8
Lesson Agenda	5-10
Using the TO_CHAR Function with Dates	5-11
Elements of the Date Format Model	5-12
Using the TO_CHAR Function with Dates	5-15

Using the TO_CHAR Function with Numbers	5-16
Using the TO_NUMBER and TO_DATE Functions	5-19
Using TO_CHAR and TO_DATE Functions with the RR Date Format	5-21
Lesson Agenda	5-22
General Functions	5-23
NVL Function	5-24
Using the NVL Function	5-25
Using the NVL2 Function	5-26
Using the NULLIF Function	5-27
Using the COALESCE Function	5-28
Lesson Agenda	5-30
Conditional Expressions	5-31
CASE Expression	5-32
Using the CASE Expression	5-33
Searched CASE Expression	5-34
DECODE Function	5-35
Using the DECODE Function	5-36
Quiz	5-38
Summary	5-39
Practice 5: Overview	5-40

6 Reporting Aggregated Data Using the Group Functions

Course Roadmap	6-2
Objectives	6-3
Lesson Agenda	6-4
Group Functions	6-5
Types of Group Functions	6-6
Group Functions: Syntax	6-7
Using the AVG and SUM Functions	6-8
Using the MIN and MAX Functions	6-9
Using the COUNT Function	6-10
Using the DISTINCT Keyword	6-11
Group Functions and Null Values	6-12
Lesson Agenda	6-13
Creating Groups of Data	6-14
Creating Groups of Data: GROUP BY Clause Syntax	6-15
Using the GROUP BY Clause	6-16
Grouping by More Than One Column	6-18
Using the GROUP BY Clause on Multiple Columns	6-19
Illegal Queries Using Group Functions	6-20
Restricting Group Results	6-22

Restricting Group Results with the HAVING Clause 6-23
Using the HAVING Clause 6-24
Lesson Agenda 6-26
Nesting Group Functions 6-27
Quiz 6-28
Summary 6-29
Practice 6: Overview 6-30

7 Displaying Data from Multiple Tables Using Joins

Course Roadmap 7-2
Objectives 7-3
Lesson Agenda 7-4
Why Join? 7-5
Obtaining Data from Multiple Tables 7-6
Types of Joins 7-7
Joining Tables Using SQL:1999 Syntax 7-8
Lesson Agenda 7-9
Creating Natural Joins 7-10
Retrieving Records with Natural Joins 7-11
Creating Joins with the USING Clause 7-12
Joining Column Names 7-13
Retrieving Records with the USING Clause 7-14
Qualifying Ambiguous Column Names 7-15
Using Table Aliases with the USING Clause 7-16
Creating Joins with the ON Clause 7-17
Retrieving Records with the ON Clause 7-18
Creating Three-Way Joins 7-19
Applying Additional Conditions to a Join 7-20
Lesson Agenda 7-21
Joining a Table to Itself 7-22
Self-Joins Using the ON Clause 7-23
Lesson Agenda 7-24
Nonequiijoins 7-25
Retrieving Records with Nonequiijoins 7-26
Lesson Agenda 7-27
Returning Records with No Direct Match Using OUTER Joins 7-28
INNER Versus OUTER Joins 7-29
LEFT OUTER JOIN 7-30
RIGHT OUTER JOIN 7-31
FULL OUTER JOIN 7-32
Lesson Agenda 7-33

Cartesian Products 7-34
Generating a Cartesian Product 7-35
Creating Cross Joins 7-36
Quiz 7-37
Summary 7-38
Practice 7: Overview 7-39

8 Using Subqueries to Solve Queries

Course Roadmap 8-2
Objectives 8-3
Lesson Agenda 8-4
Using a Subquery to Solve a Problem 8-5
Subquery Syntax 8-6
Using a Subquery 8-7
Rules and Guidelines for Using Subqueries 8-8
Types of Subqueries 8-9
Lesson Agenda 8-10
Single-Row Subqueries 8-11
Executing Single-Row Subqueries 8-12
Using Group Functions in a Subquery 8-13
HAVING Clause with Subqueries 8-14
What Is Wrong with This Statement? 8-15
No Rows Returned by the Inner Query 8-16
Lesson Agenda 8-17
Multiple-Row Subqueries 8-18
Using the ANY Operator in Multiple-Row Subqueries 8-19
Using the ALL Operator in Multiple-Row Subqueries 8-20
Multiple-Column Subqueries 8-21
Multiple-Column Subquery: Example 8-22
Lesson Agenda 8-23
Null Values in a Subquery 8-24
Quiz 8-26
Summary 8-27
Practice 8: Overview 8-28

9 Using Set Operators

Course Roadmap 9-2
Objectives 9-3
Lesson Agenda 9-4
Set Operators 9-5
Set Operator Rules 9-6

Oracle Server and Set Operators	9-7
Lesson Agenda	9-8
Tables Used in This Lesson	9-9
Lesson Agenda	9-13
UNION Operator	9-14
Using the UNION Operator	9-15
UNION ALL Operator	9-16
Using the UNION ALL Operator	9-17
Lesson Agenda	9-18
INTERSECT Operator	9-19
Using the INTERSECT Operator	9-20
Lesson Agenda	9-21
MINUS Operator	9-22
Using the MINUS Operator	9-23
Lesson Agenda	9-24
Matching SELECT Statements	9-25
Matching the SELECT Statement: Example	9-26
Lesson Agenda	9-27
Using the ORDER BY Clause in Set Operations	9-28
Quiz	9-29
Summary	9-30
Practice 9: Overview	9-31

10 Managing Tables Using DML Statements

Course Roadmap	10-2
Objectives	10-3
HR Application Scenario	10-4
Lesson Agenda	10-5
Data Manipulation Language	10-6
Adding a New Row to a Table	10-7
INSERT Statement Syntax	10-8
Inserting New Rows	10-9
Inserting Rows with Null Values	10-10
Inserting Special Values	10-11
Inserting Specific Date and Time Values	10-12
Creating a Script	10-13
Copying Rows from Another Table	10-14
Lesson Agenda	10-15
Changing Data in a Table	10-16
UPDATE Statement Syntax	10-17
Updating Rows in a Table	10-18

Updating Two Columns with a Subquery	10-19
Updating Rows Based on Another Table	10-20
Lesson Agenda	10-21
Removing a Row from a Table	10-22
DELETE Statement	10-23
Deleting Rows from a Table	10-24
Deleting Rows Based on Another Table	10-25
TRUNCATE Statement	10-26
Lesson Agenda	10-27
Database Transactions	10-28
Database Transactions: Start and End	10-29
Advantages of COMMIT and ROLLBACK Statements	10-30
Explicit Transaction Control Statements	10-31
Rolling Back Changes to a Marker	10-32
Implicit Transaction Processing	10-33
State of Data Before COMMIT or ROLLBACK	10-35
State of Data After COMMIT	10-36
Committing Data	10-37
State of Data After ROLLBACK	10-38
State of Data After ROLLBACK: Example	10-39
Statement-Level Rollback	10-40
Lesson Agenda	10-41
Read Consistency	10-42
Implementing Read Consistency	10-43
Lesson Agenda	10-44
FOR UPDATE Clause in a SELECT Statement	10-45
FOR UPDATE Clause: Examples	10-46
LOCK TABLE Statement	10-48
Quiz	10-49
Summary	10-50
Practice 10: Overview	10-51

11 Introduction to Data Definition Language

Course Roadmap	11-2
Objectives	11-3
HR Application Scenario	11-4
Lesson Agenda	11-5
Database Objects	11-6
Naming Rules for Tables and Columns	11-7
Lesson Agenda	11-8
CREATE TABLE Statement	11-9

Creating Tables	11-10
Lesson Agenda	11-11
Data Types	11-12
Datetime Data Types	11-14
DEFAULT Option	11-15
Lesson Agenda	11-16
Including Constraints	11-17
Constraint Guidelines	11-18
Defining Constraints	11-19
Defining Constraints: Example	11-20
NOT NULL Constraint	11-21
UNIQUE Constraint	11-22
PRIMARY KEY Constraint	11-24
FOREIGN KEY Constraint	11-25
FOREIGN KEY Constraint: Keywords	11-27
CHECK Constraint	11-28
CREATE TABLE: Example	11-29
Violating Constraints	11-30
Lesson Agenda	11-32
Creating a Table Using a Subquery	11-33
Lesson Agenda	11-35
ALTER TABLE Statement	11-36
Adding a Column	11-38
Modifying a Column	11-39
Dropping a Column	11-40
SET UNUSED Option	11-41
Read-Only Tables	11-43
Lesson Agenda	11-44
Dropping a Table	11-45
Quiz	11-46
Summary	11-47
Practice 11: Overview	11-48

12 Oracle Cloud Overview

Lesson Objectives	12-2
Lesson Agenda	12-3
Introduction to Oracle Cloud	12-4
Oracle Cloud Services	12-5
Cloud Deployment Models	12-6
Lesson Agenda	12-7
Evolving from On-premises to Exadata Express	12-8

What is in Exadata Express?	12-9
Exadata Express for Users	12-10
Exadata Express for Developers	12-11
Getting Started with Exadata Express	12-12
Oracle Exadata Express Cloud Service	12-13
Getting Started with Exadata Express	12-14
Managing Exadata	12-15
Service Console	12-16
Web Access through Service Console	12-17
Client Access Configuration through Service Console	12-18
Database Administration through Service Console	12-19
SQL Workshop	12-20
Connecting through Database Clients	12-22
Enabling SQL*Net Access for Client Applications	12-23
Downloading Client Credentials	12-24
Connecting Oracle SQL Developer	12-25
Connecting Oracle SQLcl	12-26
Summary	12-27

A Table Descriptions

B Using SQL Developer

Objectives	B-2
What Is Oracle SQL Developer?	B-3
Specifications of SQL Developer	B-4
SQL Developer 3.2 Interface	B-5
Creating a Database Connection	B-7
Browsing Database Objects	B-10
Displaying the Table Structure	B-11
Browsing Files	B-12
Creating a Schema Object	B-13
Creating a New Table: Example	B-14
Using the SQL Worksheet	B-15
Executing SQL Statements	B-19
Saving SQL Scripts	B-20
Executing Saved Script Files: Method 1	B-21
Executing Saved Script Files: Method 2	B-22
Formatting the SQL Code	B-23
Using Snippets	B-24
Using Snippets: Example	B-25
Using the Recycle Bin	B-26

Debugging Procedures and Functions	B-27
Database Reporting	B-28
Creating a User-Defined Report	B-29
Search Engines and External Tools	B-30
Setting Preferences	B-31
Resetting the SQL Developer Layout	B-33
Data Modeler in SQL Developer	B-34
Summary	B-35

C Using SQL*Plus

Objectives	C-2
SQL and SQL*Plus Interaction	C-3
SQL Statements Versus SQL*Plus Commands	C-4
SQL*Plus: Overview	C-5
Logging In to SQL*Plus	C-6
Displaying the Table Structure	C-7
SQL*Plus Editing Commands	C-9
Using LIST, n, and APPEND	C-11
Using the CHANGE Command	C-12
SQL*Plus File Commands	C-13
Using the SAVE and START Commands	C-14
SERVEROUTPUT Command	C-15
Using the SQL*Plus SPOOL Command	C-16
Using the AUTOTRACE Command	C-17
Summary	C-18

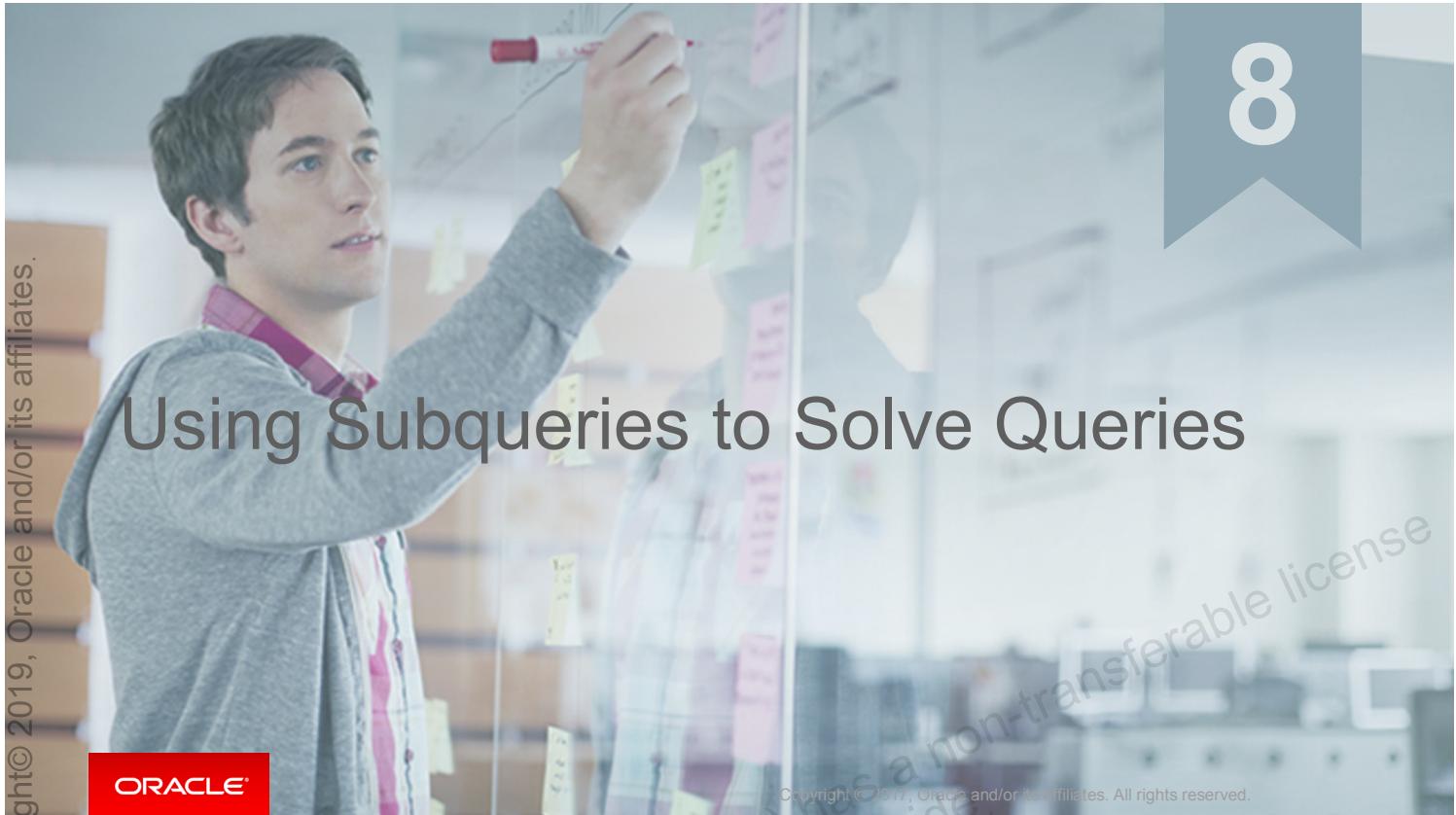
D Commonly Used SQL Commands

Objectives	D-2
Basic SELECT Statement	D-3
SELECT Statement	D-4
WHERE Clause	D-5
ORDER BY Clause	D-6
GROUP BY Clause	D-7
Data Definition Language	D-8
CREATE TABLE Statement	D-9
ALTER TABLE Statement	D-10
DROP TABLE Statement	D-11
GRANT Statement	D-12
Privilege Types	D-13
REVOKE Statement	D-14
TRUNCATE TABLE Statement	D-15

Data Manipulation Language	D-16
INSERT Statement	D-17
UPDATE Statement Syntax	D-18
DELETE Statement	D-19
Transaction Control Statements	D-20
COMMIT Statement	D-21
ROLLBACK Statement	D-22
SAVEPOINT Statement	D-23
Joins	D-24
Types of Joins	D-25
Qualifying Ambiguous Column Names	D-26
Natural Join	D-27
Equijoins	D-28
Retrieving Records with Equijoins	D-29
Additional Search Conditions Using the AND and WHERE Operators	D-30
Retrieving Records with Nonequijoins	D-31
Retrieving Records by Using the USING Clause	D-32
Retrieving Records by Using the ON Clause	D-33
Left Outer Join	D-34
Right Outer Join	D-35
Full Outer Join	D-36
Self-Join: Example	D-37
Cross Join	D-38
Summary	D-39

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Using Subqueries to Solve Queries

ORACLE

Copyright ©2017, Oracle and/or its affiliates. All rights reserved.

Course Roadmap

Lesson 1: Introduction

Unit 1: Retrieving, Restricting, and Sorting Data

Unit 2: Joins, Subqueries, and Set Operators

Unit 3: DML and DDL

▶ Lesson 6: Reporting Aggregated Data Using Group Functions

▶ Lesson 7: Displaying Data from Multiple Tables Using Joins

▶ **Lesson 8: Using Subqueries to Solve Queries**

▶ Lesson 9: Using Set Operators

You are here!



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In Unit 2, you will learn to use:

- SQL statements to query and display data from multiple tables by using joins
- Subqueries when the condition is unknown
- Group functions to aggregate data
- Set operators

Objectives

After completing this lesson, you should be able to do the following:

- Define subqueries
- Describe the types of problems that subqueries can solve
- Identify the types of subqueries
- Write single-row, multiple-row, multiple-column subqueries



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
 - Group functions in a subquery
 - HAVING clause with subqueries
- Multiple-row subqueries
 - Using ALL or ANY operator
- Multiple-column subqueries
- Null values in a subquery



Using a Subquery to Solve a Problem



Suppose the HR manager wants a report of all employees who were hired after Davies. The HR manager submits a request for the report to the IT department.

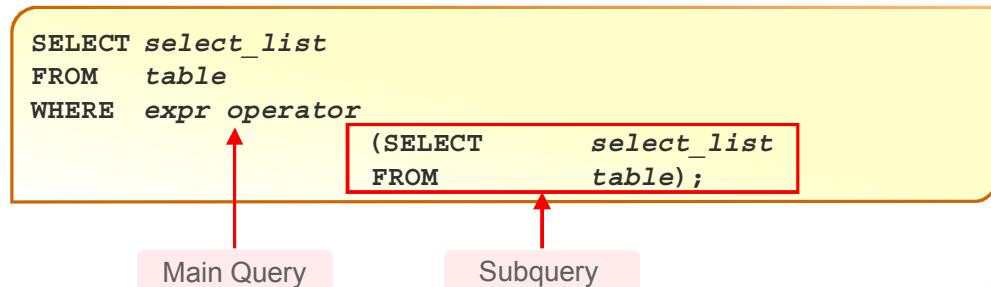
To solve this problem, the IT manager needs *two* queries: one query to find when Davies was hired, and another to find who were hired after Davies.

The IT manager can solve this problem by combining the two queries, placing one query *inside* the other query.

The inner query (or *subquery*) returns a value that is used by the outer query (or *main query*).

Subquery Syntax

- The subquery (inner query) executes *before* the main query (outer query).
- The result of the subquery is used by the main query.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

A subquery is a `SELECT` statement that is embedded in the clause of another `SELECT` statement.

You can place the subquery in a number of SQL clauses, including the following:

- `WHERE` clause
- `HAVING` clause
- `FROM` clause

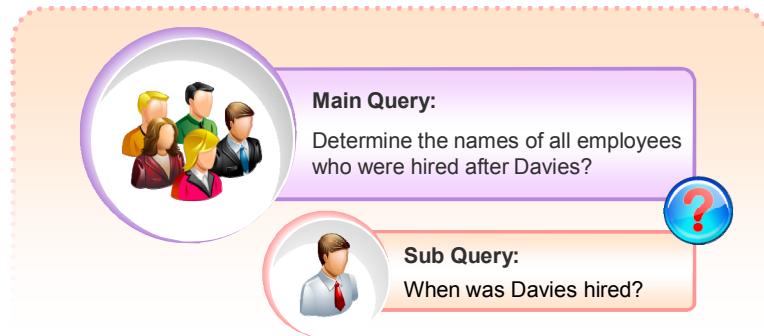
In the syntax:

`operator` includes a comparison condition such as `>`, `=`, or `IN`

The subquery is often referred to as a nested `SELECT`, sub-`SELECT`, or inner `SELECT` statement.

The subquery generally executes first, and its output is used to complete the query condition for the main (or outer) query.

Using a Subquery



```
SELECT last_name, hire_date  
FROM   employees  
WHERE  hire_date > (SELECT hire_date  
                      FROM   employees  
                      WHERE  last_name = 'Davies');
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the inner query determines the hire date of the employee, Davies. The outer query takes the result of the inner query and uses this result to display all the employees who were hired after Davies.

Rules and Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition for readability. (However, the subquery can appear on either side of the comparison operator.)
- Use single-row operators with single-row subqueries and multiple-row operators with multiple-row subqueries.

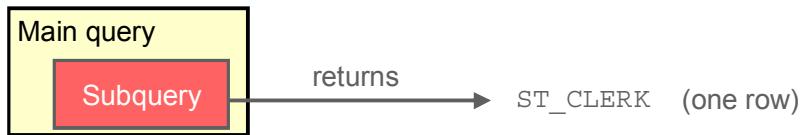


ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Types of Subqueries

- Single-row subquery



- Multiple-row subquery



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- **Single-row subqueries:** Queries that return only one row from the inner SELECT statement
- **Multiple-row subqueries:** Queries that return more than one row from the inner SELECT statement

Note: There are also multiple-column subqueries, which are queries that return more than one column from the inner SELECT statement. They are covered in the *Oracle Database: SQL Workshop II* course.

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
 - Group functions in a subquery
 - HAVING clause with subqueries
- Multiple-row subqueries
 - Using ALL or ANY operator
- Multiple-column subqueries
- Null values in a subquery



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A single-row subquery is one that returns one row from the inner SELECT statement. This type of subquery uses a single-row operator. The table in the slide lists single-row operators.

Example

Display the employees whose job ID is the same as that of employee 141:

```
SELECT last_name, job_id
  FROM employees
 WHERE job_id =
       (SELECT job_id
        FROM   employees
       WHERE  employee_id = 141);
```

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = (SELECT job_id
                  FROM   employees
                  WHERE  last_name = 'Taylor')
AND    salary > (SELECT salary
                  FROM   employees
                  WHERE  last_name = 'Taylor');
```

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using Group Functions in a Subquery

```
SELECT last_name, job_id, salary
  FROM employees
 WHERE salary = 2500
      (SELECT MIN(salary)
       FROM employees);
```

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

HAVING Clause with Subqueries

The Oracle server:

- Executes the subqueries first
- Returns the result into the HAVING clause of the main query

```
SELECT      department_id, MIN(salary)
FROM        employees
GROUP BY    department_id
HAVING      MIN(salary) > 2500
          (SELECT MIN(salary)
           FROM   employees
           WHERE  department_id = 50);
```

DEPARTMENT_ID	MIN(SALARY)
1	(null)
2	17000
3	6000
4	8300
5	8600
6	4200
7	4400

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use subqueries not only in the WHERE clause, but also in the HAVING clause. The Oracle server executes the subquery and the results are returned into the HAVING clause of the main query.

The SQL statement in the slide displays all the departments that have a minimum salary greater than the minimum salary of department 30.

Another Example:

Find the job with the lowest average salary.

```
SELECT      job_id, AVG(salary)
FROM        employees
GROUP BY    job_id
HAVING      AVG(salary) = (SELECT      MIN(AVG(salary))
                           FROM        employees
                           GROUP BY   job_id);
```

What Is Wrong with This Statement?

```
SELECT employee_id, last_name
  FROM employees
 WHERE salary =  
        (SELECT MIN(salary)
          FROM employees
         GROUP BY department_id);
```

ORA-01427: single-row subquery returns more than one row
01427. 00000 - "single-row subquery returns more than one row"
"Cause:
"Action:

Single-row operator with multiple-row subquery



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

No Rows Returned by the Inner Query

```
SELECT last_name, job_id
  FROM employees
 WHERE job_id =
       (SELECT job_id
        FROM   jobs
       WHERE  job_title = 'Architect');
```



The subquery returns no rows because there is no job with the title "Architect."

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
 - Group functions in a subquery
 - HAVING clause with subqueries
- Multiple-row subqueries
 - Use IN, ALL, or ANY
- Multiple-column subqueries
- Null values in a subquery



Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. This returns TRUE if at least one element exists in the result set of the subquery for which the relation is TRUE.
ALL	Must be preceded by =, !=, >, <, <=, >=. This returns TRUE if the relation is TRUE for all elements in the result set of the subquery.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Subqueries that return more than one row are called multiple-row subqueries. You use a multiple-row operator, instead of a single-row operator, with a multiple-row subquery. The multiple-row operator expects one or more values:

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (SELECT MIN(salary)
                   FROM employees
                  GROUP BY department_id);
```

Example

Find the employees who earn the same salary as the minimum salary of any department.

The inner query is executed first, producing a query result. The main query block is then processed and uses the values that were returned by the inner query to complete its search condition. In fact, the main query appears to the Oracle server as follows:

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300,
                   8600, 17000);
```

Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
  FROM employees
 WHERE salary < ANY (SELECT salary
                        FROM employees
                       WHERE job_id = 'IT_PROG')
AND     job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
          9000, 6000, 4200
          (SELECT salary
           FROM   employees
           WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

#	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Multiple-Column Subqueries

- A multiple-column subquery returns more than one column to the outer query.
- Column comparisons in multiple column comparisons can be pairwise or nonpairwise.
- A multiple-column subquery can also be used in the `FROM` clause of a `SELECT` statement.

Syntax:

```
SELECT column, column, ...
  FROM table
 WHERE (column1, column2, ...) IN
       (SELECT column1, column2,
              ...
              FROM table
             WHERE condition);
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Multiple-Column Subquery: Example

Display all the employees with the lowest salary in each department.

```
SELECT first_name, department_id, salary
FROM employees
WHERE (salary, department_id) IN
    (SELECT min(salary), department_id
     FROM employees
     GROUP BY department_id)
ORDER BY department_id;
```

#	FIRST_NAME	DEPARTMENT_ID	SALARY
1	Jennifer	10	4400
2	Pat	20	6000
3	Peter	50	2500
4	Diana	60	4200
5	Jonathon	80	8600
6	Neena	90	17000
7	Lex	90	17000
8	William	110	8300



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
 - Group functions in a subquery
 - HAVING clause with subqueries
- Multiple-row subqueries
 - Using ALL or ANY operator
- Multiple-column subqueries
- Null values in a subquery



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Null Values in a Subquery

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id NOT IN
       (SELECT mgr.manager_id
        FROM   employees mgr);
```



The subquery returns no rows because one of the values returned by a subquery is null.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The SQL statement in the slide attempts to display all the employees who do not have any subordinates. Logically, this SQL statement should have returned 12 rows. However, the SQL statement does not return any rows. One of the values returned by the inner query is a null value and, therefore, the entire query returns no rows.

The reason is that all conditions that compare a null value result in a null. So whenever null values are likely to be part of the results set of a subquery, do not use the NOT IN operator. The NOT IN operator is equivalent to <> ALL.

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to =ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id IN
       (SELECT mgr.manager_id
        FROM   employees mgr);
```

Alternatively, a WHERE clause can be included in the subquery to display all employees who do not have any subordinates:

```
SELECT last_name FROM employees
WHERE employee_id NOT IN
      (SELECT manager_id
       FROM   employees
       WHERE  manager_id IS NOT NULL) ;
```

Quiz



Using a subquery is equivalent to performing two sequential queries and using the result of the first query as the search values in the second query.

- a. True
- b. False



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Define subqueries
- Identify the types of problems that subqueries can solve
- Identify the types of subqueries
- Write single-row, multiple-row, multiple-column subqueries



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In this lesson, you should have learned how to use subqueries. A subquery is a SELECT statement that is embedded in the clause of another SQL statement. Subqueries are useful when a query is based on a search criterion with unknown intermediate values.

Subqueries have the following characteristics:

- Can pass one row of data to a main statement that contains a single-row operator, such as =, <>, >, >=, <, or <=
- Can pass multiple rows of data to a main statement that contains a multiple-row operator, such as IN
- Are processed first by the Oracle server, after which the WHERE or HAVING clause uses the results
- Can contain group functions

Practice 8: Overview

This practice covers the following topics:

- Creating subqueries to query values based on unknown criteria
- Using subqueries to find out the values that exist in one set of data and not in another



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Copyright©2017, Oracle and/or its affiliates. All rights reserved.

Using Set Operators

Course Roadmap

Lesson 1: Introduction

Unit 1: Retrieving, Restricting, and Sorting Data

Unit 2: Joins, Subqueries, and Set Operators

Unit 3: DML and DDL

▶ Lesson 6: Reporting Aggregated Data Using Group Functions

▶ Lesson 7: Displaying Data from Multiple Tables Using Joins

▶ Lesson 8: Using Subqueries to Solve Queries

▶ **Lesson 9: Using Set Operators**

You are here!



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe set operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned



ORACLE

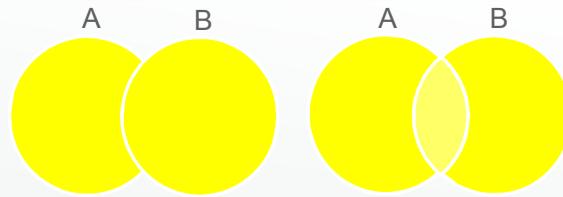
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

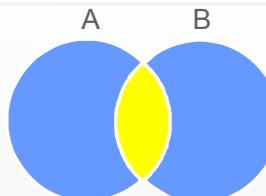
- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using the ORDER BY clause in set operations



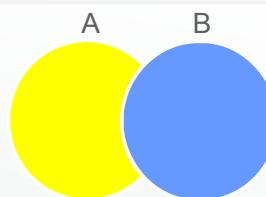
Set Operators



UNION/UNION ALL



INTERSECT



MINUS



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Set operators combine the results of two or more component queries into one result. Queries containing set operators are called *compound queries*.

Operator	Returns
UNION	Rows from both queries after eliminating duplicates
UNION ALL	Rows from both queries, including all duplicates
INTERSECT	Rows that are common to both queries
MINUS	Rows in the first query that are not present in the second query

All set operators have equal precedence. If a SQL statement contains multiple set operators, the Oracle server evaluates them from left (top) to right (bottom), if no parentheses explicitly specify another order. You should use parentheses to specify the order of evaluation explicitly in queries that use the INTERSECT operator with other set operators.

Set Operator Rules

- The expressions in the SELECT lists must match in number.
- The data type of each column in the subsequent query must match the data type of its corresponding column in the first query.
- Parentheses can be used to alter the sequence of execution.
- The ORDER BY clause can appear only at the very end of the statement.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- The expressions in the SELECT lists of the queries must match in number and data type. When you use the UNION, UNION ALL, INTERSECT, and MINUS operators, ensure that the SELECT lists have the same number and data type of columns. The data type sometimes may not be exactly the same. In such cases, the column in the second query must be in the same data type group (such as numeric or character) as the corresponding column in the first query.
- You can use the set operators in subqueries.
- You should use parentheses to specify the order of evaluation in queries that use the INTERSECT operator with other set operators. According to SQL standards, the INTERSECT operator has greater precedence than the other set operators.

Oracle Server and Set Operators

- Duplicate rows are automatically eliminated except in UNION ALL.
- Column names from the first query appear in the result.
- The output is sorted in ascending order by default, except in UNION ALL.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using the ORDER BY clause in set operations



Tables Used in This Lesson

The tables used in this lesson are:

- EMPLOYEES: Provides details about all current employees
- RETIRED_EMPLOYEES: Provides details about all past employees



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Two tables are used in this lesson: EMPLOYEES and RETIRED_EMPLOYEES.

You are already familiar with the EMPLOYEES table that stores employee details, such as a unique identification number, email address, job identification (such as ST_CLERK, SA REP, and so on), salary, manager, and so on.

RETIRED_EMPLOYEES stores the details of the employees who have left the company.

The structure of and data from the EMPLOYEES and RETIRED_EMPLOYEES tables are shown on the following pages.

DESCRIBE employees

DESCRIBE employees		
Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	100	King	AD_PRES	17-JUN-11	90
2	101	Kochhar	AD_VP	21-SEP-09	90
3	102	De Haan	AD_VP	13-JAN-09	90
4	103	Hunold	IT_PROG	03-JAN-14	60
5	104	Ernst	IT_PROG	21-MAY-15	60
6	107	Lorentz	IT_PROG	07-FEB-15	60
7	124	Mourgos	ST_MAN	16-NOV-15	50
8	141	Rajs	ST_CLERK	17-OCT-11	50
9	142	Davies	ST_CLERK	29-JAN-13	50
10	143	Matos	ST_CLERK	15-MAR-14	50
11	144	Vargas	ST_CLERK	09-JUL-14	50
12	149	Zlotkey	SA_MAN	29-JAN-16	80
13	174	Abel	SA REP	11-MAY-12	80
14	176	Taylor	SA REP	24-MAR-14	80
15	178	Grant	SA REP	24-MAY-15	(null)
16	200	Whalen	AD_ASST	17-SEP-11	10
17	201	Hartstein	MK_MAN	17-FEB-12	20
18	202	Fay	MK REP	17-AUG-13	20
19	205	Higgins	AC_MGR	07-JUN-10	110
20	206	Gietz	AC_ACCOUNT	07-JUN-10	110

```
DESCRIBE retired_employees
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(7)
FIRST_NAME		VARCHAR2(20)
LAST_NAME		VARCHAR2(20)
EMAIL		VARCHAR2(25)
RETIRED_DATE		DATE
JOB_ID		VARCHAR2(20)
SALARY		NUMBER(8,2)
MANAGER_ID		NUMBER(4)
DEPARTMENT_ID		NUMBER(6)

SELECT * FROM retired_employees;

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	RETIRED_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID
1	301	Rick	Dayle	RDAYLE	18-MAR-10	AD_PRES	8000	124	90
2	302	Meena	Rac	MRAC	21-SEP-11	AD_VP	11000	149	90
3	303	Mex	Haan	MHAAN	13-JAN-10	AD_VP	9500	149	80
4	304	Alexandera	Runold	ARUNOLD	03-JAN-11	IT_PROG	7500	124	60
5	305	Bruk	Ernst	BERNST	21-MAY-10	IT_PROG	6000	149	60
6	306	Dravid	Aust	DAUST	25-JUN-09	IT_PROG	4800	124	60
7	307	Raj	Patil	RPATIL	05-FEB-12	IT_PROG	4800	201	60
8	308	Rahul	Bose	RBOSE	17-AUG-12	FI_MGR	12008	124	100
9	309	Dany	Fav	DFAV	16-AUG-11	FI_ACCOUNT	9000	101	100
10	310	James	Ken	JKHEN	28-SEP-10	FI_ACCOUNT	8200	101	90
11	311	Shana	Garg	SGARG	30-SEP-10	FI_ACCOUNT	7700	201	100
12	312	Peter	Jois	PJOIS	07-JUN-14	FI_ACCOUNT	7800	124	100
13	313	Lui	Pops	LPOPS	07-DEC-10	FI_ACCOUNT	6900	201	100
14	314	DeL	Raph	DRAPH	07-DEC-12	PU_MAN	11000	101	30
15	315	Alex	Khurl	AKHURL	18-MAY-11	PU_CLERK	3100	149	30

Lesson Agenda

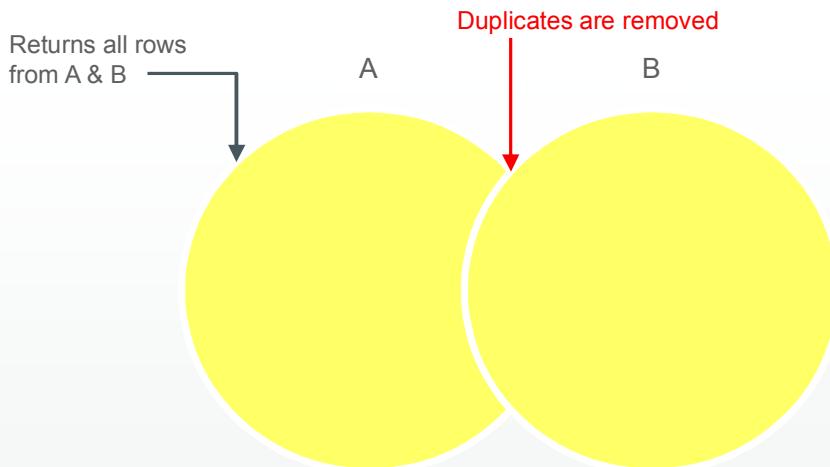
- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using the ORDER BY clause in set operations



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

UNION Operator



The UNION operator returns rows from both queries after eliminating duplicates.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the UNION Operator

Display the job details of all the current and retired employees. Display each job only once.

```
SELECT job_id  
FROM employees  
UNION  
SELECT job_id  
FROM retired_employees
```

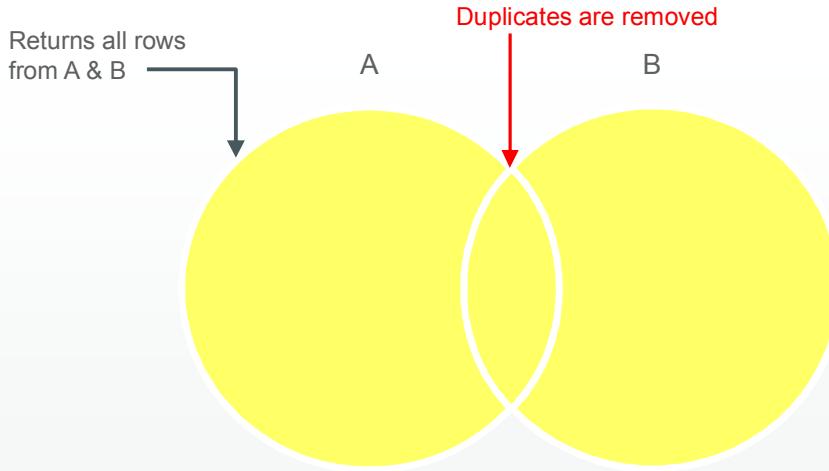
JOB_ID
1 AC_ACCOUNT
2 AC_MGR
3 AD_ASST
4 AD_PRES
5 AD_VP
6 FI_ACCOUNT
7 FI_MGR
8 IT_PROG
9 MK_MAN
10 MK_REP
11 PU_CLERK
12 PU_MAN
13 SA_MAN
14 SA REP
15 ST_CLERK
16 ST_MAN



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

UNION ALL Operator



The UNION ALL operator returns rows from both queries, including all duplications.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the UNION ALL Operator

Display the jobs and departments of all current and previous employees.

```
SELECT job_id, department_id
FROM employees
UNION ALL
SELECT job_id, department_id
FROM retired_employees
ORDER BY job_id;
```

JOB_ID	DEPARTMENT_ID	
1 AC_ACCOUNT	110	
2 AC_MGR	110	
3 AD_ASST	10	
4 AD_PRES	90	
5 AD_PRES	90	
6 AD_VP	90	
7 AD_VP	80	
8 AD_VP	90	
9 AD_VP	90	
...		
28 SA_REP	80	
29 SA_REP	80	
30 SA_REP	(null)	
31 ST_CLERK	50	
32 ST_CLERK	50	
33 ST_CLERK	50	
34 ST_CLERK	50	
35 ST_MAN	50	



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, 35 rows are selected. The combination of the two tables totals to 35 rows. The UNION ALL operator does not eliminate duplicate rows. UNION returns all distinct rows selected by either query. UNION ALL returns all rows selected by either query, including all duplicates. Consider the query in the slide, now written with the UNION clause:

```
SELECT job_id,department_id
FROM      employees
UNION
SELECT job_id,department_id
FROM      retired_employees
ORDER BY job_id;
```

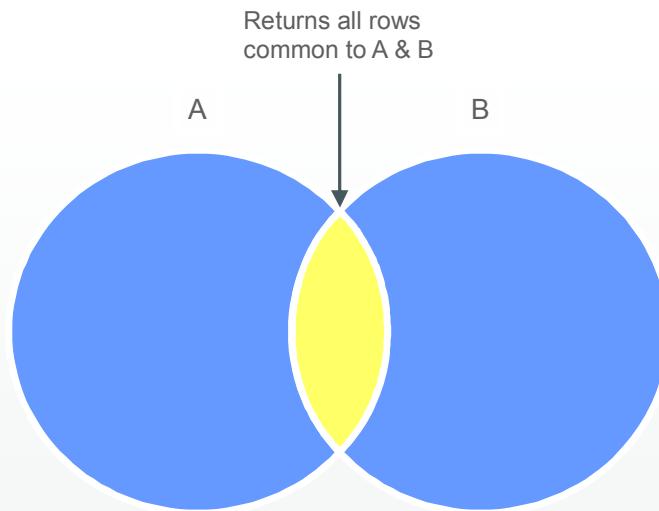
The preceding query returns 19 rows. This is because it eliminates all the duplicate rows.

Lesson Agenda

- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using ORDER BY clause in set operations



INTERSECT Operator



The INTERSECT operator returns rows that are common to both queries.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the INTERSECT Operator

Display the common manager IDs and department IDs of current and previous employees.

```
SELECT manager_id,department_id  
FROM employees  
INTERSECT  
SELECT manager_id,department_id  
FROM retired_employees
```

	MANAGER_ID	DEPARTMENT_ID
1	149	80



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

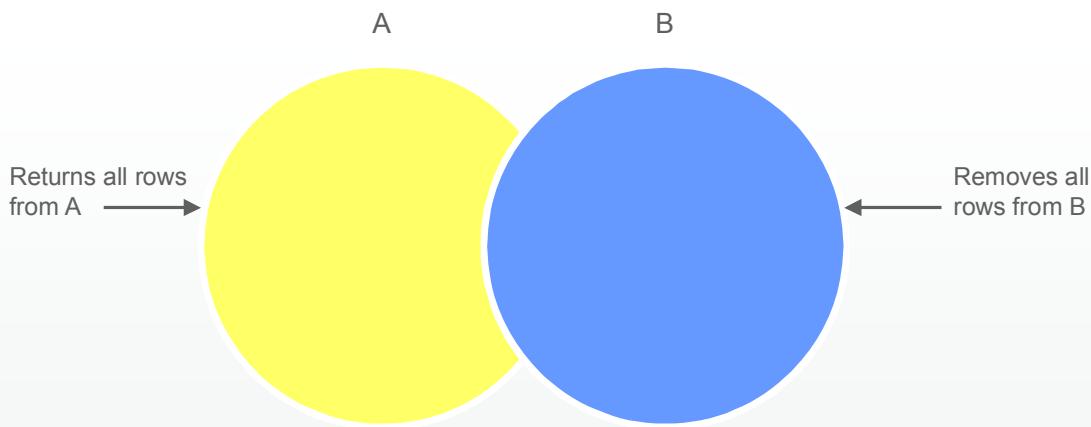
- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using the ORDER BY clause in set operations



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

MINUS Operator



The MINUS operator returns all the distinct rows selected by the first query, but not present in the second query result set.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the MINUS Operator

Display the manager IDs and Job IDs of employees whose managers have never managed retired employees in the Sales department.

```
SELECT manager_id, job_id  
FROM employees  
WHERE department_id = 80  
MINUS  
SELECT manager_id, job_id  
FROM retired_employees  
WHERE department_id = 80;
```

MANAGER_ID	JOB_ID
1	100 SA_MAN
2	149 SA_REP



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using ORDER BY clause in set operations



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Matching SELECT Statements

You must match the data type (using the TO_CHAR function or any other conversion functions) when columns do not exist in one or the other table.

```
SELECT location_id, department_name "Department",
       TO_CHAR(NULL) "Warehouse location"
  FROM departments
 UNION
SELECT location_id, TO_CHAR(NULL) "Department",
       state_province
  FROM locations;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Matching the SELECT Statement: Example

Using the UNION operator, display the employee name, job ID, and hire date of all employees.

```
SELECT FIRST_NAME, JOB_ID, hire_date "HIRE_DATE"
FROM employees
UNION
SELECT FIRST_NAME, JOB_ID, TO_DATE(NULL) "HIRE_DATE"
FROM retired_employees;
```

FIRST_NAME	JOB_ID	HIRE_DATE
1 Alex	PU_CLERK	(null)
2 Alexander	IT_PROG	03-JAN-14
3 Alexandra	IT_PROG	(null)
4 Bruce	IT_PROG	21-MAY-15
5 Bruk	IT_PROG	(null)
6 Curtis	ST_CLERK	29-JAN-13
7 Dany	FI_ACCOUNT	(null)
8 De1	PU_MAN	(null)
...		



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The EMPLOYEES and RETIRED_EMPLOYEES tables have several columns in common (for example, EMPLOYEE_ID, JOB_ID, and DEPARTMENT_ID). However, what if you want the query to display FIRST_NAME, JOB_ID, and HIRE_DATE using the UNION operator, knowing that HIRE_DATE exists only in the EMPLOYEES table?

The code example in the slide matches the FIRST_NAME and JOB_ID columns in the EMPLOYEES and RETIRED_EMPLOYEES tables. NULL is added to the RETIRED_EMPLOYEES SELECT statement to match the HIRE_DATE column in the EMPLOYEES SELECT statement.

In the results shown in the slide, each row in the output that corresponds to a record from the RETIRED_EMPLOYEES table contains a NULL in the HIRE_DATE column.

Lesson Agenda

- Set operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching SELECT statements
- Using the ORDER BY clause in set operations



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Using the ORDER BY Clause in Set Operations

- The ORDER BY clause can appear only once at the end of the compound query.
- Component queries cannot have individual ORDER BY clauses.
- The ORDER BY clause recognizes only the columns of the first SELECT query.
- By default, the first column of the first SELECT query is used to sort the output in ascending order.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use the ORDER BY clause only once in a compound query. Place the ORDER BY clause at the end of the query. The ORDER BY clause accepts the column name or an alias. By default, the output is sorted in ascending order in the first column of the first SELECT query.

Note: The ORDER BY clause does not recognize the column names of the second SELECT query. To avoid confusion over column names, it is a common practice to ORDER BY column positions.

For example, in the following statement, the output will be shown in ascending order of job_id.

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   retired_employees
ORDER BY 2;
```

If you omit ORDER BY, by default, the output will be sorted in ascending order of employee_id. You cannot use the columns from the second query to sort the output.

Quiz



Identify two set operator guidelines.

- a. The expressions in the `SELECT` lists must match in number.
- b. Parentheses cannot be used to alter the sequence of execution.
- c. The data type of each column in the second query must match the data type of its corresponding column in the first query.
- d. The `ORDER BY` clause can be used only once in a compound query, unless a `UNION ALL` operator is used.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to use:

- UNION to return all distinct rows
- UNION ALL to return all rows, including duplicates
- INTERSECT to return all rows that are shared by both queries
- MINUS to return all distinct rows that are selected by the first query, but not by the second
- ORDER BY only at the very end of the statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- The UNION operator returns all the distinct rows selected by each query in the compound query. Use the UNION operator to return all rows from multiple tables and eliminate any duplicate rows.
- Use the UNION ALL operator to return all rows from multiple queries. Unlike the case with the UNION operator, duplicate rows are not eliminated and the output is not sorted by default.
- Use the INTERSECT operator to return all rows that are common to multiple queries.
- Use the MINUS operator to return rows returned by the first query that are not present in the second query.
- Remember to use the ORDER BY clause only at the very end of the compound statement.
- Make sure that the corresponding expressions in the SELECT lists match in number and data type.

Practice 9: Overview

In this practice, you create reports by using:

- The UNION operator
- The INTERSECT operator
- The MINUS operator



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



10

Managing Tables Using DML Statements

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Course Roadmap

Lesson 1: Introduction

Unit 1: Retrieving, Restricting,
and Sorting Data

Unit 2: Joins, Subqueries, and
Set Operators

Unit 3: DML and DDL

▶ **Lesson 10: Managing Tables Using DML
Statements**

▶ Lesson 11: Introduction to Data Definition
Language

You are here!

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

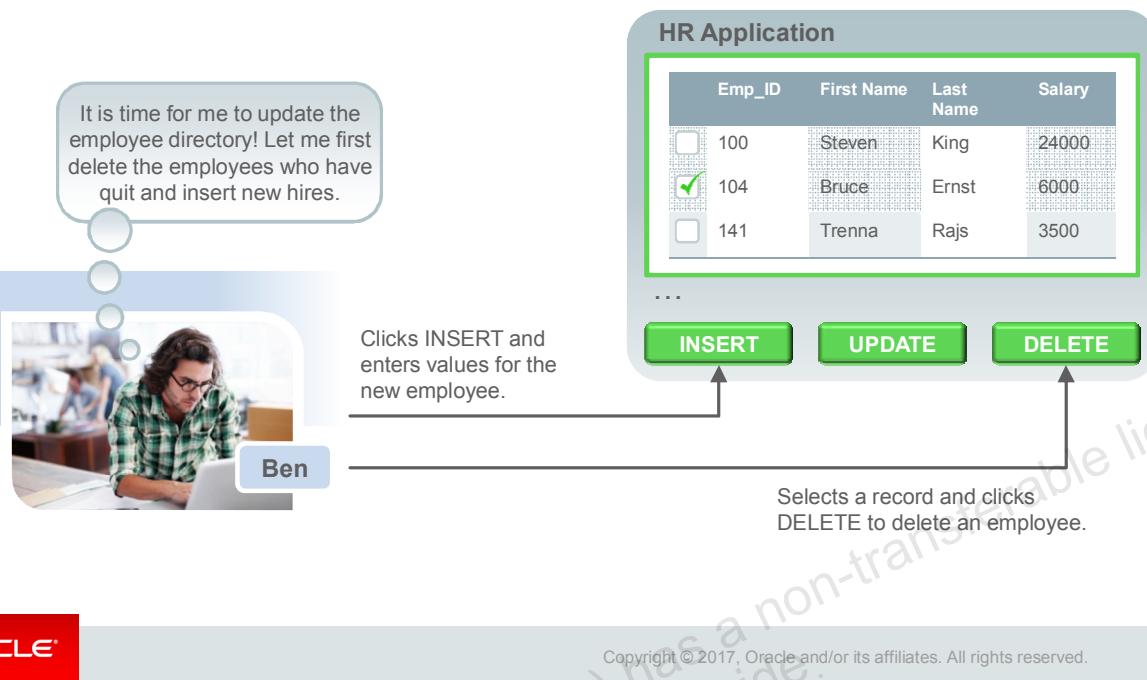
- Describe each data manipulation language (DML) statement
- Control transactions



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

HR Application Scenario



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Ben is an HR manager in USA. Ben wants to update the outdated employee list in his organization because he has hired new employees recently and a few employees have left the organization.

Ben logs in to the HR application and selects the ex-employee records and clicks on DELETE. He then clicks INSERT and enters the details of new hires and clicks SAVE. The employee list is now updated.

When the HR manager performs these transactions in the HR application, data manipulation language (DML) statements are used in the background. DML statements modify the data in the tables. In this lesson, you learn about DML statements and how to use them.

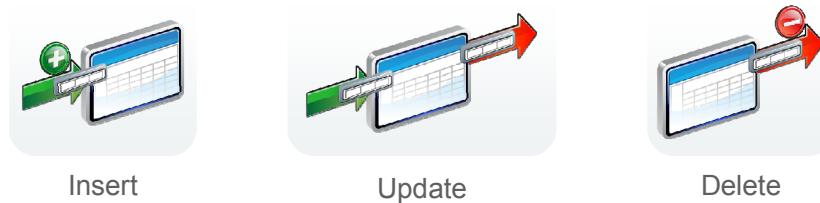
Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



Data Manipulation Language

- A DML statement is executed when you:
 - Add new rows to a table
 - Modify existing rows in a table
 - Remove existing rows from a table
- A *transaction* consists of a collection of DML statements that form a logical unit of work.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data manipulation language (DML) is a core part of SQL. When you want to add, update, or delete data in the database, you execute a DML statement. A collection of DML statements that form a logical unit of work is called a *transaction*.

Consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction might consist of three separate operations: decreasing the savings account, increasing the checking account, and recording the transaction in the transaction journal. The Oracle server must guarantee that all the three SQL statements are performed to maintain the accounts in proper balance. When something prevents one of the statements in the transaction from executing, the other statements of the transaction must be undone.

Note

- Most of the DML statements in this lesson assume that no constraints on the table are violated. Constraints are discussed later in this course.
- In SQL Developer, click the Run Script icon or press F5 to run the DML statements. The feedback messages will be shown on the Script Output tabbed page.

Adding a New Row to a Table

DEPARTMENTS				70 Public Relations	100	1700	New row
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID				
1	10 Administration	200	1700				
2	20 Marketing	201	1800				
3	50 Shipping	124	1500				
4	60 IT	103	1400				
5	80 Sales	149	2500				
6	90 Executive	100	1700				
7	110 Accounting	205	1700				
8	190 Contracting	(null)	1700				

Insert a new row into the DEPARTMENTS table.



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	70 Public Relations	100	1700
2	10 Administration	200	1700
3	20 Marketing	201	1800
4	50 Shipping	124	1500
5	60 IT	103	1400
6	80 Sales	149	2500
7	90 Executive	100	1700
8	110 Accounting	205	1700
9	190 Contracting	(null)	1700



The graphic in the slide illustrates the addition of a new department record to the DEPARTMENTS table.

INSERT Statement Syntax

- Add new rows to a table by using the `INSERT` statement:

```
INSERT INTO table [(column [, column...])]
VALUES      (value [, value...]);
```

- With this syntax, only one row is inserted at a time.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the `INSERT` clause.

```
INSERT INTO departments(department_id,
    department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 row inserted.
```

- Enclose character and date values within single quotation marks.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Because you can insert a new row that contains values for each column, the column list is not required in the `INSERT` clause. However, if you do not use the column list, the values must be listed according to the default order of the columns in the table, and a value must be provided for each column.

```
DESCRIBE departments
```

For clarity, use the column list in the `INSERT` clause.

Enclose character and date values within single quotation marks; however, it is not recommended that you enclose numeric values within single quotation marks.

Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
INSERT INTO departments (department_id,
                        department_name)
VALUES          (30, 'Purchasing');

1 row inserted.
```

- Explicit method: Specify the NULL keyword in the VALUES list.

```
INSERT INTO departments
VALUES          (100, 'Finance', NULL, NULL);

1 row inserted.
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Method	Description
Implicit	Omit the column from the column list.
Explicit	Specify the NULL keyword in the VALUES list; specify the empty string (' ') in the VALUES list for character strings and dates.

Be sure that you can use null values in the targeted column by verifying the NULL status with the DESCRIBE command.

The Oracle server automatically enforces all data types, data ranges, and data integrity constraints. If a column is not explicitly listed, a null value is inserted in the new row unless you have default values for the missing columns that are used.

Common errors that can occur when you are inserting are in the following order:

- Mandatory value missing for a NOT NULL column
- Duplicate value violating any unique or primary key constraint
- Any value violating a CHECK constraint
- Referential integrity maintained for foreign key constraint
- Data type mismatches or values too wide to fit in a column

Note: Use of the column list is recommended because it makes the INSERT statement more readable and reliable, or less prone to mistakes.

Inserting Special Values

The SYSDATE function records the current date and time.

```
INSERT INTO employees (employee_id,
                      first_name, last_name,
                      email, phone_number,
                      hire_date, job_id, salary,
                      commission_pct, manager_id,
                      department_id)
VALUES
      (113,
       'Louis', 'Popp',
       'LPOPP', '515.124.4567',
       SYSDATE, 'AC_ACCOUNT', 6900,
       NULL, 205, 110);
```

1 row inserted.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use functions to enter special values in your table.

The example in the slide records information for employee Popp in the EMPLOYEES table. It supplies the current date and time in the HIRE_DATE column. It uses the CURRENT_DATE function that returns the current date in the session time zone. You can also use the USER function when inserting rows in a table. The USER function records the current username.

Confirming Additions to the Table

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

Inserting Specific Date and Time Values

- Add a new employee.

```
INSERT INTO employees
VALUES      (114,
              'Den', 'Raphealy',
              'DRAPHEAL', '515.127.4561',
              TO_DATE('FEB 3, 2016', 'MON DD, YYYY'),
              'SA REP', 11000, 0.2, 100, 60);
```

1 row inserted.

- Verify your addition.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-16	SA REP	11000	0.2	100	60



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The DD-MON-RR format is generally used to insert a date value. With the RR format, the system provides the correct century automatically.

You may also supply the date value in the DD-MON-YYYY format. This is recommended because it clearly specifies the century and does not depend on the internal RR format logic of specifying the correct century.

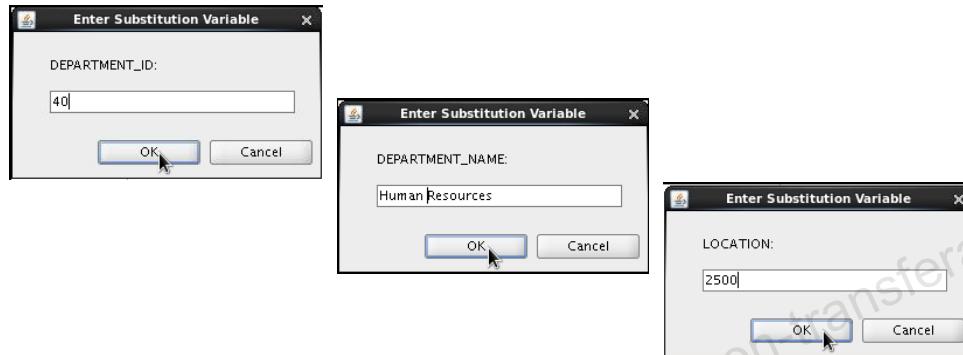
If you want to enter the date in a format other than the default format (for example, with another century or a specific time), you must use the `TO_DATE` function.

The example in the slide records information for employee Raphealy in the `EMPLOYEES` table. It sets the `HIRE_DATE` column to be February 3, 2003.

Creating a Script

- Use the & substitution in a SQL statement to prompt for values.
- & is a placeholder for the variable value.

```
INSERT INTO departments
  (department_id, department_name, location_id)
VALUES (&department_id, '&department_name',&location);
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can save commands with substitution variables to a file and execute the commands in the file. The example in the slide records information for a department in the DEPARTMENTS table.

Run the script file and you are prompted for input for each of the ampersand (&) substitution variables. After entering a value for the substitution variable, click the OK button. The values that you input are then substituted into the statement. This enables you to run the same script file over and over, but supply a different set of values each time you run it.

Copying Rows from Another Table

- Write your `INSERT` statement with a subquery:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

5 rows inserted.

- Do not use the `VALUES` clause.
- Match the number of columns in the `INSERT` clause to those in the subquery.
- Inserts all the rows returned by the subquery in the table, `sales_reps`.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use the `INSERT` statement to add rows to a table where the values are derived from existing tables. In the example in the slide, for the `INSERT INTO` statement to work, you must have already created the `sales_reps` table using the `CREATE TABLE` statement. `CREATE TABLE` is discussed in the lesson titled “Introduction to Data Definition Language.”

In place of the `VALUES` clause, you use a subquery.

Syntax

```
INSERT INTO table [ column (, column) ] subquery;
```

In the syntax:

<code>table</code>	Is the name of the table
<code>column</code>	Is the name of the column in the table to populate
<code>subquery</code>	Is the subquery that returns rows to the table

The number of columns and their data types in the column list of the `INSERT` clause must match the number of values and their data types in the subquery. Zero or more rows are added depending on the number of rows returned by the subquery. To create a copy of the rows of a table, use `SELECT *` in the subquery:

```
INSERT INTO copy_emp
SELECT *
FROM employees;
```

Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



ORACLE®

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Changing Data in a Table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Update rows in the EMPLOYEES table:



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50



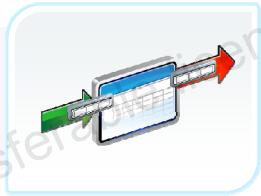
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

UPDATE Statement Syntax

- Modify existing values in a table with the UPDATE statement:

```
UPDATE      table  
SET         column = value [, column = value, ...]  
[WHERE       condition];
```

- Update more than one row at a time (if required).



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can modify the existing values in a table by using the **UPDATE** statement.

In the syntax:

<i>table</i>	Is the name of the table
<i>column</i>	Is the name of the column in the table to populate
<i>value</i>	Is the corresponding value or subquery for the column
<i>Condition</i>	Identifies the rows to be updated and is composed of column names, expressions, constants, subqueries, and comparison operators

Confirm the update operation by querying the table to display the updated rows.

For more information, see the section on “**UPDATE**” in *Oracle Database SQL Language Reference* for 12c database.

Note: In general, use the primary key column in the WHERE clause to identify a single row for update. Using other columns can unexpectedly cause several rows to be updated. For example, identifying a single row in the **EMPLOYEES** table by last name may return more than one employee having the same last name.

Updating Rows in a Table

- Values for a specific row or rows are modified if you specify the WHERE clause:

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
```

1 row updated.

- Values for all the rows in the table are modified if you omit the WHERE clause:

```
UPDATE copy_emp
SET department_id = 110;
22 rows updated
```

- Specify SET *column_name*= NULL to update a column value to NULL.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The UPDATE statement modifies the values of a specific row or rows if the WHERE clause is specified. The example in the slide shows the transfer of employee 113 (Popp) to department 50.

If you omit the WHERE clause, values for all the rows in the table are modified. Examine the updated rows in the COPY_EMP table.

```
SELECT last_name, department_id
FROM copy_emp;
```

For example, an employee who was an SA REP has now changed his job to an IT PROG. Therefore, his JOB_ID needs to be updated and the commission field needs to be set to NULL.

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

Note: The COPY_EMP table has the same data as the EMPLOYEES table.

Updating Two Columns with a Subquery

Update employee 103's job and salary to match those of employee 205.

```
UPDATE employees
SET (job_id,salary) = (SELECT job_id,salary
                        FROM employees
                        WHERE employee_id = 205)
WHERE employee_id = 103;
```

```
1 row updated.
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can update multiple columns in the SET clause of an UPDATE statement by writing multiple subqueries. The syntax is as follows:

```
UPDATE table
SET column =
    (SELECT column
     FROM table
     WHERE condition)
[ ,
column =
    (SELECT column
     FROM table
     WHERE condition)]
[WHERE condition] ;
```

Updating Rows Based on Another Table

Use the subqueries in the UPDATE statements to update row values in a table based on values from another table:

```
UPDATE copy_emp
SET department_id = (SELECT department_id
                      FROM employees
                      WHERE employee_id = 100)
WHERE job_id = (SELECT job_id
                 FROM employees
                 WHERE employee_id = 200);
```

1 row updated.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



Removing a Row from a Table

DEPARTMENTS

#	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Delete a row from the DEPARTMENTS table:

#	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

DELETE Statement

You can remove existing rows from a table by using the `DELETE` statement:

```
DELETE [FROM]    table  
[WHERE    condition];
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the syntax:

table

Is the name of the table

condition

Identifies the rows to be deleted, and is composed of column names, expressions, constants, subqueries, and comparison operators

Note: If no rows are deleted, the message “0 rows deleted” is returned (on the Script Output tab in SQL Developer).

For more information, see the section on “`DELETE`” in *Oracle Database SQL Language Reference* for 12c database.

Deleting Rows from a Table

- Specific rows are deleted if you specify the WHERE clause:

```
DELETE FROM departments  
WHERE department_name = 'Finance';  
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause:

```
DELETE FROM copy_emp;  
22 rows deleted
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can delete specific rows by specifying the WHERE clause in the DELETE statement. The first example in the slide deletes the accounting department from the DEPARTMENTS table. You can confirm the delete operation by trying to display the deleted rows using the SELECT statement. The query returns 0 rows.

```
SELECT *  
FROM departments  
WHERE department_name = 'Finance';
```

However, if you omit the WHERE clause, all rows in the table are deleted. The second example in the slide deletes all rows from the COPY_EMP table, because no WHERE clause was specified.

Example

Remove rows identified in the WHERE clause.

```
DELETE FROM employees WHERE employee_id = 114;
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
```

Deleting Rows Based on Another Table

Use the subqueries in the `DELETE` statements to remove rows from a table based on values from another table:

```
DELETE FROM employees
WHERE department_id IN
    (SELECT department_id
     FROM departments
     WHERE department_name
          LIKE '%Public%');

1 row deleted.
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

TRUNCATE Statement

- Removes all rows from a table, leaving the table empty and the table structure intact
- Is a data definition language (DDL) statement rather than a DML statement; cannot be undone
- Syntax:

```
TRUNCATE TABLE table_name;
```

- Example:

```
TRUNCATE TABLE copy_emp;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



Database Transactions

A database transaction consists of one of the following:

- DML statements that constitute one consistent change to the data
- One DDL statement
- One data control language (DCL) statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Transactions give you more flexibility and control when changing data, and the Oracle server ensures data consistency in the event of user process failure or system failure.

Transactions consist of DML statements that constitute one consistent change to the data. For example, a transfer of funds between two accounts should include the debit in one account and the credit to another account of the same amount. Both actions should either fail or succeed together; the credit should not be committed without the debit.

Transaction Types

Type	Description
Data manipulation language (DML)	Consists of any number of DML statements that the Oracle server treats as a single entity or a logical unit of work
Data definition language (DDL)	Consists of only one DDL statement
Data control language (DCL)	Consists of only one DCL statement

Database Transactions: Start and End

- Begin when the first DML SQL statement is executed
- End with one of the following events:
 - A COMMIT or ROLLBACK statement is issued.
 - A DDL or DCL statement executes (automatic commit).
 - The user exits SQL Developer or SQL*Plus.
 - The system crashes.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When does a database transaction start and end?

A transaction begins when the first DML statement is encountered and ends when one of the following occurs:

- A COMMIT or ROLLBACK statement is issued.
- A DDL statement, such as CREATE, is issued.
- A DCL statement is issued.
- The user exits SQL Developer or SQL*Plus.
- A machine fails or the system crashes.

After one transaction ends, the next executable SQL statement automatically starts the next transaction.

A DDL statement or a DCL statement is automatically committed and, therefore, implicitly ends a transaction.

Advantages of COMMIT and ROLLBACK Statements

Using COMMIT and ROLLBACK statements, you can:

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations



COMMIT

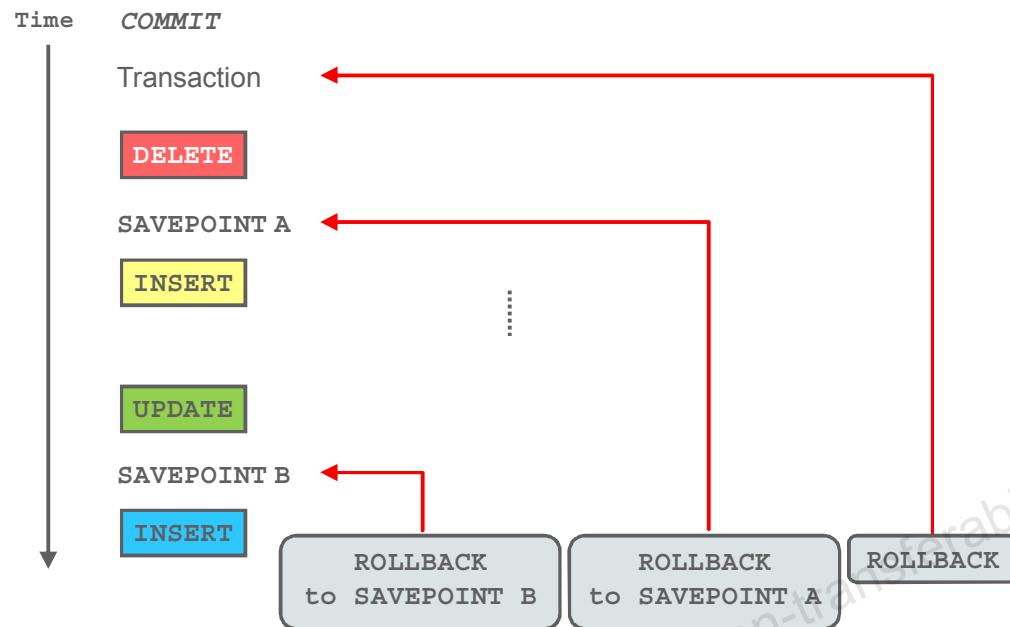


ROLLBACK

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Explicit Transaction Control Statements



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can control the logic of transactions by using the COMMIT, SAVEPOINT, and ROLLBACK statements.

Statement	Description
COMMIT	COMMIT ends the current transaction by making all pending data changes permanent.
SAVEPOINT <i>name</i>	SAVEPOINT <i>name</i> marks a savepoint within the current transaction.
ROLLBACK	ROLLBACK ends the current transaction by discarding all pending data changes.
ROLLBACK TO SAVEPOINT <i>name</i>	ROLLBACK TO <savepoint> rolls back the current transaction to the specified savepoint, thereby discarding any changes and/or savepoints that were created after the savepoint to which you are rolling back. If you omit the TO SAVEPOINT clause, the ROLLBACK statement rolls back the entire transaction. Because savepoints are logical, there is no way to list the savepoints that you have created.

Note: You cannot COMMIT to a SAVEPOINT. SAVEPOINT is not ANSI-standard SQL.

Rolling Back Changes to a Marker

- Create a marker in the current transaction by using the `SAVEPOINT` statement.
- Roll back to that marker by using the `ROLLBACK TO SAVEPOINT` statement.

```
UPDATE...
SAVEPOINT update_done;
```

```
SAVEPOINT update_done
```

```
INSERT...
```

```
ROLLBACK TO update_done;
```

```
Rollback complete.
```

ROLLBACK to this point



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Implicit Transaction Processing

- An automatic commit occurs in the following circumstances:
 - A DDL statement is issued
 - A DCL statement is issued
 - A normal exit from SQL Developer or SQL*Plus, without explicitly issuing COMMIT or ROLLBACK statements
- An automatic rollback occurs when there is an abnormal termination of SQL Developer or SQL*Plus, or a system failure.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Status	Circumstances
Automatic commit	A DDL statement or DCL statement is issued; SQL Developer or SQL*Plus exited normally, without explicitly issuing COMMIT or ROLLBACK commands.
Automatic rollback	Abnormal termination of SQL Developer or SQL*Plus, or system failure.

Note: In SQL*Plus, the AUTOCOMMIT command can be toggled ON or OFF. If set to ON, each individual DML statement is committed as soon as it is executed. You cannot roll back the changes. If set to OFF, the COMMIT statement can still be issued explicitly. Also, the COMMIT statement is issued when a DDL statement is issued or when you exit SQL*Plus. The SET AUTOCOMMIT ON/OFF command is skipped in SQL Developer. DML is committed on a normal exit from SQL Developer only if you have the Autocommit preference enabled. To enable Autocommit, perform the following:

- From the Tools menu, select Preferences. In the Preferences dialog box, expand Database and select Worksheet Parameters.
- In the right pane, select the “Autocommit in SQL Worksheet” option. Click OK.

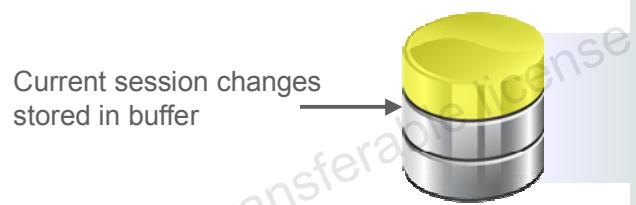
System Failures

When a transaction is interrupted by a system failure, the entire transaction is automatically rolled back. This prevents the error from causing unwanted changes to the data and returns the tables to the state at the time of the last commit. In this way, the Oracle server protects the integrity of the tables.

In SQL Developer, a normal exit from the session is accomplished by selecting Exit from the File menu. In SQL*Plus, a normal exit is accomplished by entering the `EXIT` command at the prompt. Closing the window is interpreted as an abnormal exit.

State of Data Before COMMIT or ROLLBACK

- You can recover the data of the previous state.
- You can review the results of the DML operations by using the `SELECT` statement in the current session.
- Other sessions *cannot* view the results of the DML statements issued by the current session.
- The affected rows are *locked*; other sessions cannot change the data in the affected rows.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Every data change made during the transaction is temporary until the transaction is committed.

The state of the data before `COMMIT` or `ROLLBACK` statements are issued can be described as follows:

- Data manipulation operations primarily affect the database buffer; therefore, the previous state of the data can be recovered.
- The current session can review the results of the data manipulation operations by querying the tables.
- Other sessions cannot view the results of the data manipulation operations made by the current session. The Oracle server institutes read consistency to ensure that each session sees data as it existed at the last commit.
- The affected rows are locked; other sessions cannot change the data in the affected rows.

State of Data After COMMIT

- Data changes are saved in the database.
- The previous state of the data is overwritten.
- All sessions can view the results.
- Locks on the affected rows are released; those rows are available for other sessions to manipulate.
- All savepoints are erased.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Committing Data

- Make the changes:

```
DELETE FROM employees  
WHERE employee_id = 113;  
1 row deleted.  
  
INSERT INTO departments  
VALUES (290, 'Corporate Tax', NULL, 1700);  
1 row inserted.
```

- Commit the changes:

```
COMMIT;
```

```
Commit complete.
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, a row is deleted from the EMPLOYEES table and a new row is inserted into the DEPARTMENTS table. The changes are saved by issuing the COMMIT statement.

Example

Remove departments 290 and 300 from the DEPARTMENTS table and update a row in the EMPLOYEES table. Save the data change.

```
DELETE FROM departments  
WHERE department_id IN (290, 300);  
  
UPDATE employees  
SET department_id = 80  
WHERE employee_id = 206;  
  
COMMIT;
```

State of Data After ROLLBACK

Discard all pending changes by using the ROLLBACK statement:

- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released.

```
DELETE FROM copy_emp;  
ROLLBACK;
```



ROLLBACK

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

State of Data After ROLLBACK: Example

```
DELETE FROM test;  
4 rows deleted.  
  
ROLLBACK;  
Rollback complete.  
  
DELETE FROM test WHERE id = 100;  
1 row deleted.  
  
SELECT * FROM test WHERE id = 100;  
No rows selected.  
  
COMMIT;  
Commit complete.
```

While attempting to remove a record from the TEST table, you may accidentally empty the table. However, you can correct the mistake by rolling back, reissue a proper statement, and make the data change permanent with COMMIT statement.

Statement-Level Rollback

- If a single DML statement fails during execution, only that statement is rolled back.
- The Oracle server implements an implicit savepoint.
- All other changes are retained.
- The user should terminate transactions explicitly by executing a COMMIT or ROLLBACK statement.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

If a single DML statement fails during the execution of a transaction, its effect is undone by a statement-level implicit rollback; however, the changes made by the previous DML statements in the transaction are not discarded. These can be committed or rolled back explicitly by the user.

The Oracle server issues an implicit commit before and after any DDL statement. So, even if your DDL statement does not execute successfully, you cannot roll back the previous statement because the server issued a commit.

Terminate your transactions explicitly by executing a COMMIT or ROLLBACK statement.

Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



Read Consistency

- Read consistency guarantees a consistent view of data at all times.
- Changes made by one user do not conflict with the changes made by another user.
- Read consistency ensures that, on the same data:
 - Readers do not wait for writers
 - Writers do not wait for readers
 - Writers wait for writers



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Database users access the database in two ways:

- Read operations (`SELECT` statement)
- Write operations (`INSERT`, `UPDATE`, `DELETE` statements)

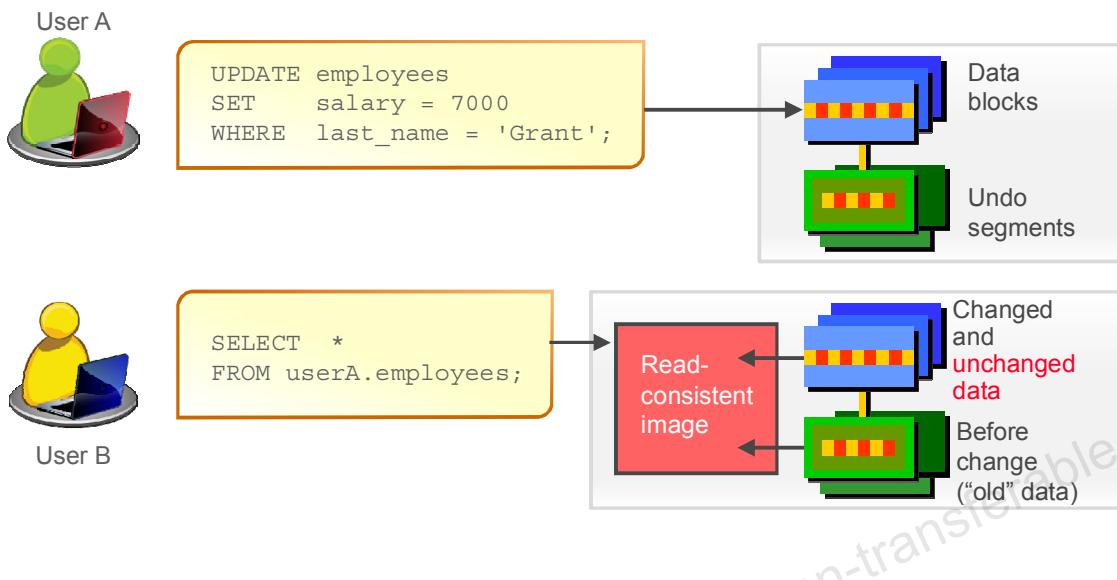
You need read consistency so that:

- The database reader and writer are ensured a consistent view of data
- Readers do not view data that is in the process of being changed
- Writers are ensured that the changes to the database are done in a consistent manner
- Changes made by one writer do not disrupt or conflict with the changes being made by another writer

The purpose of read consistency is to ensure that each user sees data as it existed at the last commit, before a DML operation started.

Note: The same user can log in to different sessions. Each session maintains read consistency in the manner described above, even if they are the same users.

Implementing Read Consistency



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Read consistency is an automatic implementation. It keeps a partial copy of the database in the undo segments. The read-consistent image is constructed from the committed data in the table and the old data that is being changed and is not yet committed from the undo segment.

When an insert, update, or delete operation is made on the database, the Oracle server takes a copy of the data before it is changed and writes it to an *undo segment*.

All readers, except the one who issued the change, see the database as it existed before the changes started; they view the undo segment's "snapshot" of the data.

Before the changes are committed to the database, only the user who is modifying the data sees the database with the alterations. Everyone else sees the snapshot in the undo segment. This guarantees that readers of the data read consistent data that is not currently undergoing change.

When a DML statement is committed, the change made to the database becomes visible to anyone issuing a SELECT statement *after* the commit is done. The space occupied by the *old* data in the undo segment file is freed for reuse.

If the transaction is rolled back, the changes are undone:

- The original, older version of the data in the undo segment is written back to the table.
- All users see the database as it existed before the transaction began.

Lesson Agenda

- Adding new rows in a table
 - INSERT statement
- Changing data in a table
 - UPDATE statement
- Removing rows from a table:
 - DELETE statement
 - TRUNCATE statement
- Database transaction control using COMMIT, ROLLBACK, and SAVEPOINT
- Read consistency
- Manual Data Locking
 - FOR UPDATE clause in a SELECT statement
 - LOCK TABLE statement



FOR UPDATE Clause in a SELECT Statement

- Locks the rows in the EMPLOYEES table where job_id is SA_REP.

```
SELECT employee_id, salary, commission_pct, job_id
  FROM employees
 WHERE job_id = 'SA_REP'
   FOR UPDATE
 ORDER BY employee_id;
```

- Lock is released only when you issue a ROLLBACK or a COMMIT.
- If the SELECT statement attempts to lock a row that is locked by another user, the database waits until the row is available, and then returns the results of the SELECT statement.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

FOR UPDATE Clause: Examples

- You can use the FOR UPDATE clause in a SELECT statement against multiple tables.

```
SELECT e.employee_id, e.salary, e.commission_pct
  FROM employees e JOIN departments d
    USING (department_id)
   WHERE job_id = 'ST_CLERK'
     AND location_id = 1500
    FOR UPDATE
 ORDER BY e.employee_id;
```

- Rows from both the EMPLOYEES and DEPARTMENTS tables are locked.
- Use FOR UPDATE OF *column_name* to qualify the column that you intend to change; then only the rows from that specific table are locked.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the statement locks rows in the EMPLOYEES table with JOB_ID set to ST_CLERK and LOCATION_ID set to 1500, and locks rows in the DEPARTMENTS table with departments in LOCATION_ID set as 1500.

You can use the FOR UPDATE OF *column_name* to qualify the column that you intend to change. The OF list of the FOR UPDATE clause does not restrict you to changing only those columns of the selected rows. Locks are still placed on all rows; if you simply state FOR UPDATE in the query and do not include one or more columns after the OF keyword, the database will lock all identified rows across all the tables listed in the FROM clause.

The following statement locks only those rows in the EMPLOYEES table with ST_CLERK located in LOCATION_ID 1500. No rows are locked in the DEPARTMENTS table:

```
SELECT e.employee_id, e.salary, e.commission_pct
  FROM employees e JOIN departments d
    USING (department_id)
   WHERE job_id = 'ST_CLERK' AND location_id = 1500
    FOR UPDATE OF e.salary
 ORDER BY e.employee_id;
```

In the following example, the database is instructed to wait for five seconds for the row to become available, and then return control to you.

```
SELECT employee_id, salary, commission_pct, job_id  
FROM employees  
WHERE job_id = 'SA_REP'  
FOR UPDATE WAIT 5  
ORDER BY employee_id;
```

LOCK TABLE Statement

- Use the `LOCK TABLE` statement to lock one or more tables in a specified mode.
- This manually overrides automatic locking.
- Tables are locked until you `COMMIT` or `ROLLBACK`.

```
LOCK TABLE table_name
IN [ROW SHARE/ROW EXCLUSIVE/SHARE UPDATE/SHARE/
    SHARE ROW EXCLUSIVE/ EXCLUSIVE] MODE
[NOWAIT];
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use the `LOCK TABLE` statement to manually lock the tables and override automatic locking. The table must be in your schema or you must have the `LOCK ANY TABLE` privilege.

The `lockmode` clause:

SHARE	Permits concurrent queries but prevents update on the locked table
EXCLUSIVE	Permits queries on the locked table but prevents any other activity

For more information about the other `lockmode` clauses, refer to the `LOCK TABLE` statement in *Oracle Database SQL Language Reference* for 12c database.

You can append the optional keyword `NOWAIT` to the `LOCK TABLE` statement to tell the Oracle server not to wait if the table has been locked by another user.

For example, the following statement locks the `EMPLOYEES` table in exclusive mode but does not wait if another user has already locked the table.

```
LOCK TABLE employees
  IN EXCLUSIVE MODE
  NOWAIT;
```

Quiz



The following statements produce the same results:

`DELETE FROM copy_emp;`

`TRUNCATE TABLE copy_emp;`

- a. True
- b. False



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to use the following statements:

Function	Description
INSERT	Adds a new row to the table
UPDATE	Modifies existing rows in the table
DELETE	Removes existing rows from the table
TRUNCATE	Removes all rows from a table
COMMIT	Makes all pending changes permanent
SAVEPOINT	Is used to roll back to the savepoint marker
ROLLBACK	Discards all pending data changes
FOR UPDATE clause in SELECT	Locks rows identified by the SELECT query



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Practice 10: Overview

This practice covers the following topics:

- Inserting rows into the tables
- Updating and deleting rows in the table
- Controlling transactions



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



11

Introduction to Data Definition Language

Course Roadmap

Lesson 1: Introduction

Unit 1: Retrieving, Restricting,
and Sorting Data

Unit 2: Joins, Subqueries, and
Set Operators

Unit 3: DML and DDL

▶ Lesson 10: Managing Tables Using DML
Statements

▶ **Lesson 11: Introduction to Data Definition
Language**

You are here!

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

HR Application Scenario

The JOBS table looks fine except it does not contain a column for JOB_TITLE. What should I do now?

Bob



HR Application

job_ID	Min_Salary	Max_Salary
AD_PRES	20080	40000
SA_MAN	10000	20080
IT_PROG	4000	10000

EDIT

HR Application

Action: Add Column
Name: Job Title
Datatype: varchar2(25)
Default:

SUBMIT

job_ID	Min_Salary	Max_Salary	Job_Title
AD_PRES	20080	40000	NULL
SA_MAN	10000	20080	NULL
IT_PROG	4000	10000	NULL



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

ORACLE

Consider a scenario where Bob, an HR manager is creating tables to store all the employee information in the database. While creating the JOBS table to store all the job information, he forgets to create a column for the job title. So what should Bob do now?

Should he drop the table and create JOBS table again? No!

Bob can alter the table and add a new column (JOB_TITLE) to the existing JOBS table. The statements that allow you to modify the structure of database objects are called data definition language (DDL) statements. Usually, the permission to execute DDL statements is given only to the admin.

Bob submits the request to alter the JOBS table structure along with the details. The DBA receives the request and constructs an appropriate SQL statement to alter the JOBS table.

A new column called Job_Title is added to the JOBS table. The value for Job_Title remains NULL until Bob goes to the application and updates the values for all the records.

In this lesson, you learn more about DDL statements.

Lesson Agenda

- Database objects
 - Naming rules
- CREATE TABLE statement
- Data types
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Database Objects

Object	Description
Table	Is the basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative name to an object



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The Oracle database can contain multiple data objects. Remember to outline each object in the database design so that it can be created during the build stage of database development.

- **Table:** Stores data
- **View:** Is a subset of data from one or more tables
- **Sequence:** Generates numeric values
- **Index:** Improves the performance of some queries
- **Synonym:** Gives an alternative name to an object

Oracle Table Structures

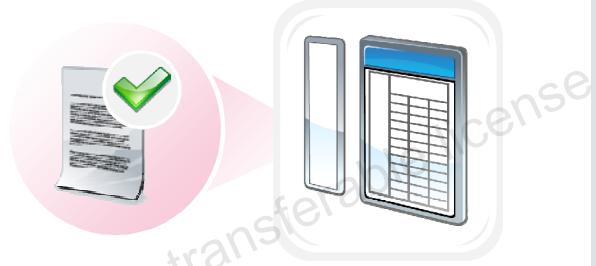
- You can create tables at any time, even when users are using the database.
- You do not need to specify the size of a table. The size is ultimately defined by the amount of space allocated to the database as a whole. It is important, however, to estimate how much space a table will use over time.
- You can also modify the table structure online.

Note: More database objects are covered in the course titled *Oracle Database 12c: Workshop II*.

Naming Rules for Tables and Columns

Ensure that the table names and column names:

- Begin with a letter
- Are 1–30 characters long
- Contain only A–Z, a–z, 0–9, _, \$, and #
- Do *not* duplicate the name of another object owned by the same user
- Are *not* Oracle server–reserved words



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Name the database tables and columns according to the standard rules for naming any Oracle database object.

- Table names and column names must begin with a letter and be 1–30 characters long.
- Names must contain only the characters A–Z, a–z, 0–9, _ (underscore), \$, and # (legal characters, but their use is discouraged).
- Names must not duplicate the name of another object owned by the same Oracle server user.
- Names must not be an Oracle server–reserved word.
 - You may also use quoted identifiers to represent the name of an object. A quoted identifier begins and ends with double quotation marks (""). If you name a schema object using a quoted identifier, you must use the double quotation marks whenever you refer to that object. Quoted identifiers can be reserved words, although this is not recommended.

Naming Guidelines

Use descriptive names for tables and other database objects.

Note: Names are not case-sensitive. For example, EMPLOYEES is treated to be the same name as eMPLOYEES or eMpLOYEES. However, quoted identifiers are case-sensitive.

For more information, see the “Schema Object Names and Qualifiers” section in the *Oracle Database SQL Language Reference* for 12c database.

Lesson Agenda

- Database objects
 - Naming rules
- CREATE TABLE statement
- Data types
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

CREATE TABLE Statement

- You must have:
 - The CREATE TABLE privilege
 - A storage area

```
CREATE TABLE [schema.]table  
  (column datatype [DEFAULT expr] [, ...]);
```

- You specify:
 - The table name
 - The column name, column data type, and column size



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You create tables to store data by executing the SQL CREATE TABLE statement. This statement is one of the DDL statements that are a subset of the SQL statements used to create, modify, or remove Oracle Database structures. These statements have an immediate effect on the database and they also record information in the data dictionary. The data dictionary is an important set of read-only tables that provide database information.

To create a table, a user must have the CREATE TABLE privilege and a storage area in which to create objects. The database administrator (DBA) uses data control language (DCL) statements to grant privileges to users.

In the syntax:

<i>schema</i>	Is the same as the owner's name
<i>table</i>	Is the name of the table
<i>DEFAULT expr</i>	Specifies a default value if a value is omitted in the INSERT statement
<i>column</i>	Is the name of the column
<i>datatype</i>	Is the column's data type and length

Note: The CREATE ANY TABLE privilege is needed to create a table in any schema other than the user's schema.

Creating Tables

- Create the table:

```
CREATE TABLE dept
  (deptno      NUMBER(2),
   dname       VARCHAR2(14),
   loc         VARCHAR2(13),
   create_date DATE DEFAULT SYSDATE);
table DEPT created.
```

- Confirm table creation:

```
DESCRIBE dept
```

Name	Null	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The example in the slide creates the DEPT table with four columns: DEPTNO, DNAME, LOC, and CREATE_DATE.

The CREATE_DATE column has a default value. If a value is not provided for an INSERT statement, the system date is automatically inserted.

To confirm that the table was created, run the DESCRIBE command.

Because creating a table is a DDL statement, an automatic commit takes place when this statement is executed.

Note: You can view the list of tables that you own by querying the data dictionary, as shown in the following example:

```
select table_name from user_tables;
```

Using data dictionary views, you can also find information about other database objects, such as views, indexes, and so on. You will learn about data dictionaries in detail in the *Oracle Database: SQL Workshop II* course.

Lesson Agenda

- Database objects
 - Naming rules
- CREATE TABLE statement
- Data types
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data Types

Data Type	Description
VARCHAR2 (<i>size</i>)	Variable-length character data
CHAR (<i>size</i>)	Fixed-length character data
NUMBER (<i>p, s</i>)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data (up to 2 GB)
CLOB	Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE).
RAW and LONG RAW	Raw binary data
BLOB	Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)).
BFILE	Binary data stored in an external file (up to 4 GB)
ROWID	A base-64 number system representing the unique address of a row in its table



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When you identify a column for a table, you need to provide a data type for the column. There are several data types available:

Data Type	Description
VARCHAR2 (<i>size</i>)	Variable-length character data (A maximum <i>size</i> must be specified; minimum <i>size</i> is 1.) Maximum size is: <ul style="list-style-type: none">• 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED• 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY
CHAR [(<i>size</i>)]	Fixed-length character data of length <i>size</i> bytes (Default and minimum <i>size</i> is 1; maximum <i>size</i> is 2,000.)
NUMBER [(<i>p, s</i>)]	Number having precision <i>p</i> and scale <i>s</i> (Precision is the total number of decimal digits and scale is the number of digits to the right of the decimal point; precision can range from 1 through 38, and scale can range from -84 through 127.)
DATE	Date and time values to the nearest second between January 1, 4712 B.C., and December 31, 9999 A.D.

Data Type	Description
LONG	Variable-length character data (up to 2 GB)
CLOB	A character large object containing single-byte or multibyte characters. Maximum size is $(4 \text{ gigabytes} - 1) * (\text{DB_BLOCK_SIZE})$; stores national character set data.
NCLOB	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is $(4 \text{ gigabytes} - 1) * (\text{database block size})$; stores national character set data.
RAW (size)	Raw binary data of length <code>size</code> bytes. You must specify <code>size</code> for a RAW value. Maximum <code>size</code> is: 32767 bytes if <code>MAX_SQL_STRING_SIZE = EXTENDED</code> 4000 bytes if <code>MAX_SQL_STRING_SIZE = LEGACY</code>
LONG RAW	Raw binary data of variable length up to 2 gigabytes
BLOB	A binary large object. Maximum size is $(4 \text{ gigabytes} - 1) * (\text{DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)})$.
BFILE	Binary data stored in an external file (up to 4 GB)
ROWID	Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn.

Guidelines

- A LONG column is not copied when a table is created using a subquery.
- A LONG column cannot be included in a GROUP BY or an ORDER BY clause.
- Only one LONG column can be used per table.
- No constraints can be defined on a LONG column.
- You might want to use a CLOB column rather than a LONG column.

Datetime Data Types

You can use several datetime data types:

Data Type	Description
TIMESTAMP	Date with fractional seconds
INTERVAL YEAR TO MONTH	Stored as an interval of years and months
INTERVAL DAY TO SECOND	Stored as an interval of days, hours, minutes, and seconds



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data Type	Description
TIMESTAMP	Enables storage of time as a date with fractional seconds. It stores the year, month, day, hour, minute, and the second value of the DATE data type, as well as the fractional seconds value. There are several variations of this data type, such as WITH TIMEZONE and WITH LOCALTIMEZONE.
INTERVAL YEAR TO MONTH	Enables storage of time as an interval of years and months; used to represent the difference between two datetime values in which the only significant portions are the year and month
INTERVAL DAY TO SECOND	Enables storage of time as an interval of days, hours, minutes, and seconds; used to represent the precise difference between two datetime values

Note: These datetime data types are available with Oracle9*i* and later releases. The datetime data types are discussed in detail in the lesson titled “Managing Data in Different Time Zones” in the *Oracle Database: SQL Workshop II* course.

Also, for more information about datetime data types, see the sections on “TIMESTAMP Datatype,” “INTERVAL YEAR TO MONTH Datatype,” and “INTERVAL DAY TO SECOND Datatype” in *Oracle Database SQL Language Reference* for 12c database.

DEFAULT Option

- Specify a default value for a column in the CREATE TABLE statement.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn is an illegal value.
- The default data type must match the column data type.

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);
```

```
Table HIRE_DATES created.
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When you define a table, you can specify that a column should be given a default value by using the DEFAULT option. This option prevents null values from entering the columns when a row is inserted without a value for the column.

The default value can be a literal, an expression, or a SQL function (such as SYSDATE or USER); however, the value cannot be the name of another column or a pseudocolumn (such as NEXTVAL or CURRVAL). The default expression must match the data type of the column.

Consider the following examples:

```
INSERT INTO hire_dates values(45, NULL);
```

The preceding statement will insert the null value rather than the default value.

```
INSERT INTO hire_dates(id) values(35);
```

The preceding statement will insert SYSDATE for the HIRE_DATE column.

Note: In SQL Developer, click the Run Script icon or press F5 to run the DDL statements. The feedback messages will be shown on the Script Output tabbed page.

Lesson Agenda

- Database objects
 - Naming rules
- CREATE TABLE statement
- Data types
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement

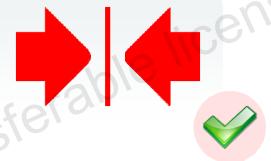


ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Including Constraints

- Constraints enforce rules at the table level.
- Constraints ensure consistency and integrity of the database.
- The following constraint types are valid:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The Oracle server uses constraints to prevent invalid data entry into tables.

You can use constraints to do the following:

- Enforce rules on the data in a table whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.
- Prevent the dropping of a table if there are dependencies from other tables.
- Provide rules for Oracle tools, such as Oracle Developer.

Data Integrity Constraints

Constraint	Description
NOT NULL	Specifies that the column cannot contain a null value
UNIQUE	Specifies a column or combination of columns whose values must be unique for all rows in the table
PRIMARY KEY	Uniquely identifies each row of the table
FOREIGN KEY	Establishes and enforces a referential integrity between the column and a column of the referenced table such that values in one table match values in another table
CHECK	Specifies a condition that must be true

Constraint Guidelines

- You can name a constraint or the Oracle server generates a name by using the `SYS_Cn` format.
- Create a constraint at either of the following times:
 - At the time of table creation
 - After the creation of the table
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

All constraints are stored in the data dictionary.

Constraints are easy to reference if you give them a meaningful name. Constraint names must follow the standard object-naming rules, except that the name cannot be the same as another object owned by the same user. If you do not name your constraint, the Oracle server generates a name with the format `SYS_Cn`, where n is an integer so that the constraint name is unique.

Constraints can be defined at the time of table creation or after the creation of the table. You can define a constraint at the column or table level. Functionally, a table-level constraint is the same as a column-level constraint.

For more information, see the section on “Constraints” in *Oracle Database SQL Language Reference* for 12c database.

Defining Constraints

- Syntax:

```
CREATE TABLE [schema.]table  
  (column datatype [DEFAULT expr]  
   [column_constraint],  
   ...  
   [table_constraint] [,...]);
```

- Column-level constraint syntax:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Table-level constraint syntax:

```
column,...  
[CONSTRAINT constraint_name] constraint_type  
(column, ...),
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The slide gives the syntax for defining constraints when creating a table. You can create constraints at the column level or the table level.

Constraints defined at the column level are included when the column is defined. Table-level constraints are defined at the end of the table definition, and must refer to the column or columns to which the constraint pertains in a set of parentheses. It is mainly the syntax that differentiates the two; otherwise, functionally, a column-level constraint is the same as a table-level constraint. NOT NULL constraints can be defined only at the column level.

Constraints that apply to more than one column must be defined at the table level.

In the syntax:

schema	Is the same as the owner's name
table	Is the name of the table
DEFAULT expr	Specifies a default value to be used if a value is omitted in the INSERT statement
column	Is the name of the column
datatype	Is the column's data type and length
column_constraint	Is an integrity constraint as part of the column definition
table_constraint	Is an integrity constraint as part of the table definition

Defining Constraints: Example

- Example of a column-level constraint:

```
CREATE TABLE employees(
    employee_id  NUMBER(6)
        CONSTRAINT emp_emp_id_pk PRIMARY KEY,
    first_name    VARCHAR2(20),
    ...);
```

1

- Example of a table-level constraint:

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    first_name    VARCHAR2(20),
    ...
    job_id        VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
        PRIMARY KEY (EMPLOYEE_ID));
```

2



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

NOT NULL Constraint

Ensures that null values are not permitted for the column:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4566	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIEZ	515.123.8181	07-JUN-94

NOT NULL constraint
(Primary Key enforces
NOT NULL constraint.)

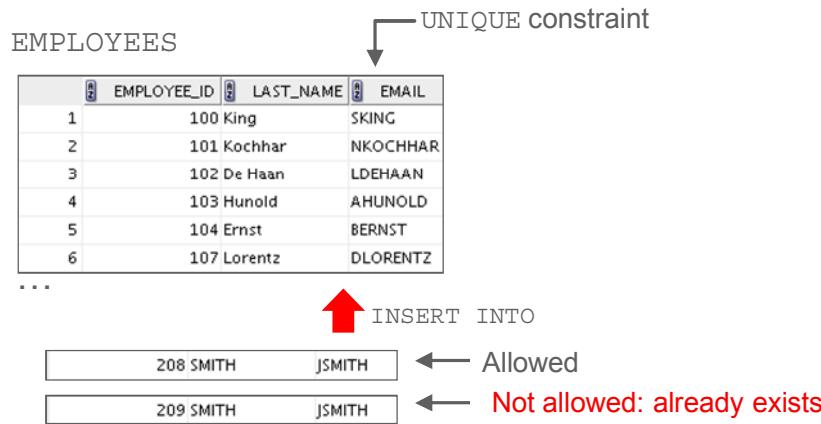
NOT NULL
constraint

Absence of NOT NULL constraint (Any row
can contain a null value for this column.)



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

UNIQUE Constraint



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A **UNIQUE** key integrity constraint requires that every value in a column or a set of columns (key) be unique—that is, no two rows of a table can have duplicate values in a specified column or a set of columns.

The column (or set of columns) included in the definition of the **UNIQUE** key constraint is called the *unique key*. If the **UNIQUE** constraint comprises more than one column, that group of columns is called a *composite unique key*.

UNIQUE constraints enable the input of nulls unless you also define **NOT NULL** constraints for the same columns. In fact, any number of rows can include nulls for columns without the **NOT NULL** constraints because nulls are not considered equal to anything. A null in a column (or in all columns of a composite **UNIQUE** key) always satisfies a **UNIQUE** constraint.

Note: Because of the search mechanism for the **UNIQUE** constraints on more than one column, you cannot have identical values in the non-null columns of a partially null composite **UNIQUE** key.

UNIQUE Constraint

Define at either the table level or the column level:

```
CREATE TABLE employees (
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    CONSTRAINT emp_email_uk UNIQUE(email));
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

PRIMARY KEY Constraint

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

Not allowed
(null value)

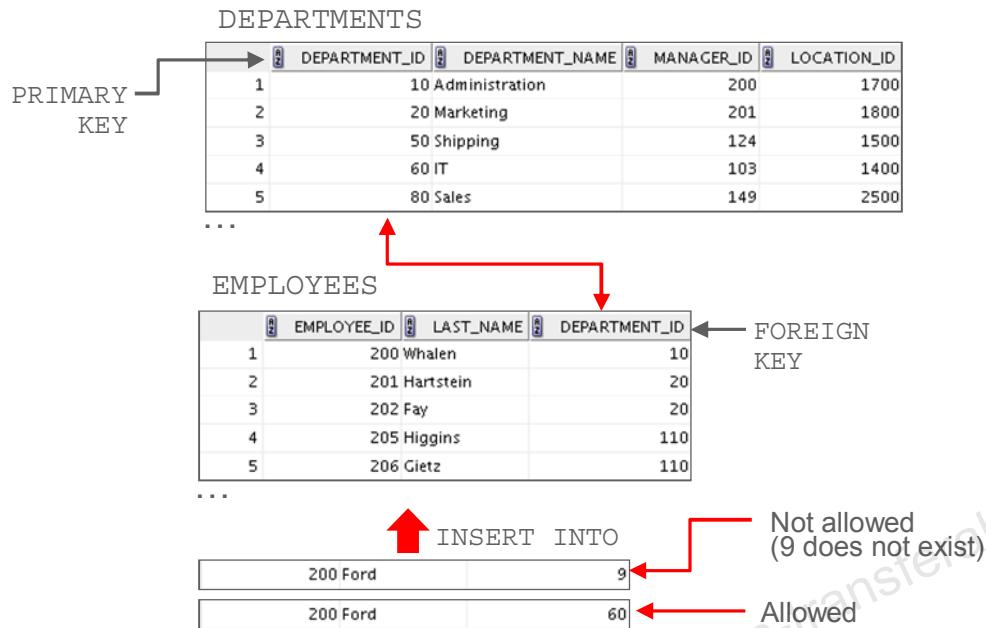
INSERT INTO

Not allowed
(50 already exists)

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

FOREIGN KEY Constraint



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The FOREIGN KEY (or referential integrity) constraint designates a column or a combination of columns as a foreign key, and establishes a relationship with a primary key or a unique key in the same table or a different table.

In the example in the slide, DEPARTMENT_ID has been defined as the foreign key in the EMPLOYEES table (dependent or child table); it references the DEPARTMENT_ID column of the DEPARTMENTS table (the referenced or parent table).

Guidelines

- A foreign key value must match an existing value in the parent table or be NULL.
- Foreign keys are based on data values and are purely logical, rather than physical, pointers.

FOREIGN KEY Constraint

Define at either the table level or the column level:

```
CREATE TABLE employees (
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
        REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

FOREIGN KEY Constraint: Keywords

- FOREIGN KEY: Defines the column in the child table at the table-constraint level
- REFERENCES: Identifies the table and column in the parent table
- ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted
- ON DELETE SET NULL: Converts dependent foreign key values to null



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The foreign key is defined in the child table and the column it references is in the parent table. The foreign key is defined using a combination of the following keywords:

- FOREIGN KEY is used to define the column in the child table at the table-constraint level.
- REFERENCES identifies the table and the column in the parent table.
- ON DELETE CASCADE indicates that when a row in the parent table is deleted, the dependent rows in the child table are also deleted.
- ON DELETE SET NULL indicates that when a row in the parent table is deleted, the foreign key values are set to null.

The default behavior is called the *restrict rule*, which disallows the update or deletion of referenced data.

Without the ON DELETE CASCADE or the ON DELETE SET NULL options, the row in the parent table cannot be deleted if it is referenced in the child table. Also, these keywords cannot be used in column-level syntax.

CHECK Constraint

- Defines a condition that each row must satisfy
- Cannot reference columns from other tables

```
...., salary NUMBER(2)
CONSTRAINT emp_salary_min
    CHECK (salary > 0),...
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When you define a CHECK constraint on a column, each row satisfies the condition. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null). The condition can use the same constructs as the query conditions; however, they must not refer to other values in other rows.

A single column can have multiple CHECK constraints that refer to the column in its definition. There is no limit to the number of CHECK constraints that you can define on a column.

CHECK constraints can be defined at the column level or table level.

```
CREATE TABLE employees
(
    ...
    salary NUMBER(8,2) CONSTRAINT emp_salary_min
        CHECK (salary > 0),
    ...
)
```

CREATE TABLE: Example

```
CREATE TABLE teach_emp (
    empno      NUMBER(5) PRIMARY KEY,
    ename      VARCHAR2(15) NOT NULL,
    job        VARCHAR2(10),
    mgr        NUMBER(5),
    hiredate   DATE DEFAULT (sysdate),
    photo      BLOB,
    sal         NUMBER(7,2),
    deptno    NUMBER(3) NOT NULL
        CONSTRAINT admin_dept_fkey
        REFERENCES
            departments(department_id));
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Violating Constraints

```
UPDATE employees  
SET department_id = 55  
WHERE department_id = 110;
```

```
Error starting at line : 1 in command -  
UPDATE employees  
      SET department_id = 55  
      WHERE department_id = 110  
Error report -  
SQL Error: ORA-02291: integrity constraint (TEACH_A.EMP_DEPT_FK) violated - parent key not found  
02291. 00000 - "integrity constraint (%.%) violated - parent key not found"  
*Cause:   A foreign key value has no matching primary key value.  
*Action:  Delete the foreign key or add a matching primary key.
```

Department 55 does not exist.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Violating Constraints

You cannot delete a row that contains a primary key that is used as a foreign key in another table.

```
DELETE FROM departments  
WHERE department_id = 60;
```

```
Error starting at line : 1 in command -  
DELETE FROM departments  
      WHERE department_id = 60  
Error report -  
SQL Error: ORA-02292: integrity constraint (TEACH_A.EMP_DEPT_FK) violated - child record found  
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"  
*Cause:    attempted to delete a parent key value that had a foreign  
           dependency.  
*Action:   delete dependencies first then parent or disable constraint.
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

If you attempt to delete a record with a value that is tied to an integrity constraint, an error is returned.

The example in the slide tries to delete department 60 from the DEPARTMENTS table, but it results in an error because that department number is used as a foreign key in the EMPLOYEES table. If the parent record that you attempt to delete has child records, you receive the “child record found” violation ORA - 02292.

The following statement works because there are no employees in department 70:

```
DELETE FROM departments  
WHERE department_id = 70;
```

1 row deleted.

Lesson Agenda

- Database objects
 - Naming rules
- Data types
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Creating a Table Using a Subquery

- Create a table and insert rows by combining the CREATE TABLE statement and the AS *subquery* option.

```
CREATE TABLE table
  [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A second method for creating a table is to apply the AS *subquery* clause, which both creates the table and inserts rows returned from the subquery.

In the syntax:

<i>table</i>	Is the name of the table
<i>column</i>	Is the name of the column, default value, and integrity constraint
<i>subquery</i>	Is the SELECT statement that defines the set of rows to be inserted into the new table

Guidelines

- The table is created with the specified column names, and the rows retrieved by the SELECT statement are inserted into the table.
- The column definition can contain only the column name and default value.
- If column specifications are given, the number of columns must equal the number of columns in the subquery SELECT list.
- If no column specifications are given, the column names of the table are the same as the column names in the subquery.
- The column data type definitions and the NOT NULL constraint are passed to the new table. Note that only the explicit NOT NULL constraint will be inherited. The PRIMARY KEY column will not pass the NOT NULL feature to the new column. Any other constraint rules are not passed to the new table. However, you can add constraints in the column definition.

Creating a Table Using a Subquery

```
CREATE TABLE dept80
AS
SELECT employee_id, last_name,
       salary*12 ANNSAL,
       hire_date
  FROM employees
 WHERE department_id = 80;
```

Table DEPT80 created.

```
DESCRIBE dept80
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Database objects
 - Naming rules
- Data types
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change table to read-only status



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

After you create a table, you may need to change the table structure for any of the following reasons:

- You omitted a column.
- Your column definition or its name needs to be changed.
- You need to remove columns.
- You want to put the table into read-only mode

You can do this by using the **ALTER TABLE** statement.

ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns:

```
ALTER TABLE table
ADD      (column datatype [DEFAULT expr]
           [, column datatype] ...);
```

```
ALTER TABLE table
MODIFY   (column datatype [DEFAULT expr]
           [, column datatype] ...);
```

```
ALTER TABLE table
DROP (column [, column] ...);
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Adding a Column

- You use the ADD clause to add columns:

```
ALTER TABLE dept80
ADD          (job_id VARCHAR2(9));
Table DEPT80 altered.
```

- The new column becomes the last column:

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149 Zlotkey	126000	29-JAN-16	(null)	
2	174 Abel	132000	11-MAY-12	(null)	
3	176 Taylor	103200	24-MAR-14	(null)	
4	206 Gietz	99600	07-JUN-10	(null)	



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
```

Table DEPT80 altered.

Size of the last_name
column is modified.

- A change to the default value of a column affects only subsequent insertions to the table.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



You can modify a column definition by using the ALTER TABLE statement with the MODIFY clause. Column modification can include changes to a column's data type, size, and default value.

Guidelines

- You can increase the width or precision of a numeric column.
- You can increase the width of character columns.
- You can decrease the width of a column if:
 - The column contains only null values
 - The table has no rows
 - The decrease in column width is not less than the existing values in that column
- You can change the data type if the column contains only null values. The exception to this is CHAR-to-VARCHAR2 conversions, which can be done with data in the columns.
- You can convert a CHAR column to the VARCHAR2 data type or convert a VARCHAR2 column to the CHAR data type only if the column contains null values or if you do not change the size.
- A change to the default value of a column affects only subsequent insertions to the table.

Dropping a Column

Use the `DROP COLUMN` clause to drop columns that you no longer need from the table:

```
ALTER TABLE dept80
DROP (job_id);
```

Table DEPT80 altered.

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Ziotkey	126000	29-JAN-16
2	174	Abel	132000	11-MAY-12
3	176	Taylor	103200	24-MAR-14
4	206	Gietz	99600	07-JUN-10



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can drop a column from a table by using the `ALTER TABLE` statement with the `DROP COLUMN` clause.

Guidelines

- The column may or may not contain data.
- Using the `ALTER TABLE DROP COLUMN` statement, only one column can be dropped at a time.
- The table must have at least one column remaining in it after it is altered.
- After a column is dropped, it cannot be recovered.
- A primary key that is referenced by another column cannot be dropped, unless the cascade option is added.
- Dropping a column can take a while if the column has a large number of values. In this case, it may be better to set it to be unused and drop it when there are fewer users on the system to avoid extended locks.

Note: Certain columns can never be dropped, such as columns that form part of the partitioning key of a partitioned table or columns that form part of the `PRIMARY KEY` of an index-organized table. For more information about index-organized tables and partitioned tables, refer to the *Oracle Database Concepts* and *Oracle Database Administrator's Guide*.

SET UNUSED Option

- You use the SET UNUSED option to mark one or more columns as unused.
- You use the DROP UNUSED COLUMNS option to remove the columns that are marked as unused.
- You can specify the ONLINE keyword to indicate that DML operations on the table will be allowed while marking the column or columns UNUSED.

```
ALTER TABLE      <table_name>
SET    UNUSED (<column_name> [ , <column_name>]);
OR
ALTER TABLE      <table_name>
SET    UNUSED COLUMN <column_name> [,<column_name>];
```



```
ALTER TABLE <table_name>
DROP    UNUSED COLUMNS;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

DROP UNUSED COLUMNS Option

DROP UNUSED COLUMNS removes from the table all columns that are currently marked as unused. You can use this statement when you want to reclaim the extra disk space from the unused columns in the table. If the table contains no unused columns, the statement returns with no errors.

```
ALTER TABLE dept80  
SET UNUSED (last_name);
```

```
ALTER TABLE dept80  
DROP UNUSED COLUMNS;
```

Note: A subsequent DROP UNUSED COLUMNS will physically remove all unused columns from a table, similar to a DROP COLUMN.

Read-Only Tables

You can use the ALTER TABLE syntax to:

- Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
- Put the table back into read/write mode

```
ALTER TABLE employees READ ONLY;  
  
-- perform table maintenance and then  
-- return table back to read/write mode  
  
ALTER TABLE employees READ WRITE;
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can specify READ ONLY to convert a table into read-only mode. When the table is in READ ONLY mode, you cannot issue any DML statements that affect the table or any SELECT . . . FOR UPDATE statements. You can issue DDL statements as long as they do not modify any data in the table. Operations on indexes associated with the table are allowed when the table is in READ ONLY mode.

Specify READ/WRITE to return a read-only table to read/write mode.

Note: You can drop a table that is in READ ONLY mode. The DROP command is executed only in the data dictionary, so access to the table contents is not required. The space used by the table will not be reclaimed until the tablespace is made read/write again, and then the required changes can be made to the block segment headers, and so on.

For information about the ALTER TABLE statement, see the course titled *Oracle Database 12c: SQL Workshop II*.

Lesson Agenda

- Database objects
 - Naming rules
- Data types
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Dropping a Table

- Moves a table to the recycle bin
- Removes the table and all its data entirely if the PURGE clause is specified
- Invalidates dependent objects and removes object privileges on the table

```
DROP TABLE dept80;
```

```
Table DEPT80 dropped.
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The `DROP TABLE` statement moves a table to the recycle bin or removes the table and all its data from the database entirely. Unless you specify the `PURGE` clause, the `DROP TABLE` statement does not result in space being released back to the tablespace for use by other objects, and the space continues to count toward the user's space quota. Dropping a table invalidates the dependent objects and removes object privileges on the table.

When you drop a table, the database loses all the data in the table and all the indexes associated with it.

Syntax

```
DROP TABLE table [PURGE]
```

In the syntax, *table* is the name of the table.

Guidelines

- All data is deleted from the table.
- Any views and synonyms remain, but are invalid.
- Any pending transactions are committed.
- Only the creator of the table or a user with the `DROP ANY TABLE` privilege can remove a table.

Note: Use the `FLASHBACK TABLE` statement to restore a dropped table from the recycle bin. This is discussed in detail in the course titled *Oracle Database 12c: SQL Workshop II*.

Quiz



Identify three actions that you perform by using constraints.

- a. Enforce rules on the data in a table whenever a row is inserted, updated, or deleted.
- b. Prevent the dropping of a table.
- c. Prevent the creation of a table.
- d. Prevent the creation of data in a table.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to use the CREATE TABLE, ALTER TABLE, and DROP TABLE statement to create a table, modify a table and columns, and include constraints.

- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In this lesson, you should have learned the following:

CREATE TABLE

- Use the CREATE TABLE statement to create a table and include constraints.
- Create a table based on another table by using a subquery.

DROP TABLE

- Remove rows and a table structure.
- When executed, this statement cannot be rolled back.

Practice 11: Overview

This practice covers the following topics:

- Creating new tables
- Creating a new table by using the CREATE TABLE AS syntax
- Verifying that tables exist
- Altering tables
- Adding columns
- Dropping columns
- Setting a table to read-only status
- Dropping tables



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



Oracle Cloud Overview

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Lesson Objectives

After completing this lesson, you should be able to do the following:

- Describe the salient features of Oracle Cloud
- Discuss the features of Oracle Database Exadata Express Cloud Service



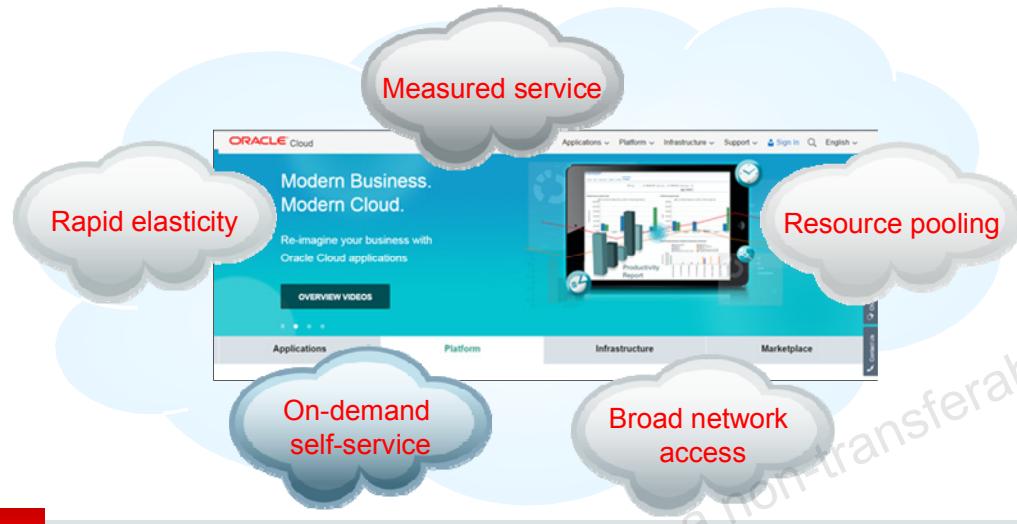
Lesson Agenda

- Overview of Oracle Cloud
- Working with Oracle Database Exadata Express Cloud Service



Introduction to Oracle Cloud

- Any business can now use the enterprise cloud provided by Oracle.
- You can access the Oracle Cloud from cloud.oracle.com.



The Oracle Cloud is an enterprise cloud for business. Oracle Cloud services are built on Oracle Exalogic Elastic Cloud and Oracle Exadata Database Machine, together offering a platform that delivers extreme performance and scalability.

The top two benefits of cloud computing are speed and cost.

As a result, the applications and databases deployed in the Oracle Cloud are portable and you can easily move them to or from a private cloud or on-premise environment.

- You can request and get the cloud services provisioned through a self-service interface.
- You can either use an integrated development and deployment platform to rapidly extend and create new services.

Using Oracle Cloud services, you can benefit from the following five essential characteristics:

- **On-demand self-service:** You can provision, monitor, and manage cloud on your own.
- **Resource pooling:** You can share resources and maintain a level of abstraction between consumers and services.
- **Rapid elasticity:** You can quickly scale up or down as needed.
- **Measured service:** You pay for what you use with either internal chargeback (private cloud) or external billing (public cloud).
- **Broad network access:** You can access the cloud services through a browser on any networked device.

Oracle Cloud Services

Oracle Cloud provides three types of services:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SaaS generally refers to applications that are delivered to end users over the Internet. Oracle CRM On Demand is an example of a SaaS offering that provides both multitenant as well as single-tenant options, depending on the customer's preferences.

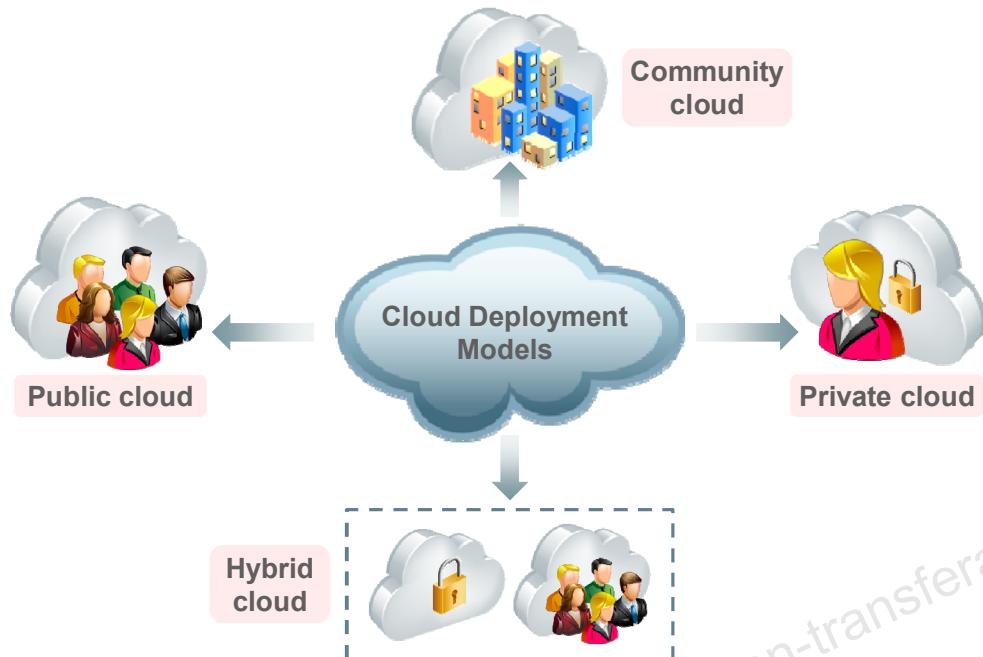
PaaS generally refers to an application development and deployment platform that is delivered as a service to developers, enabling them to quickly build and deploy a SaaS application to end users. The platform typically includes databases, middleware, and development tools, all delivered as a service via the Internet.

IaaS refers to computing hardware (servers, storage, and network) delivered as a service. This service typically includes the associated software as well as operating systems, virtualization, clustering, and so on. Examples of IaaS in the public cloud include Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3).

The Oracle Cloud Database is built as a PaaS model. It provides on-demand access to database services in a self-service, scalable, and metered manner. You can deploy a database within a virtual machine in an IaaS platform.

You can rapidly deploy Oracle Cloud Database on Oracle Exadata, which is a pre-integrated and optimized hardware platform that supports both online transaction processing (OLTP) and Data Warehouse workloads.

Cloud Deployment Models



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- **Private cloud:** A single organization uses a private cloud, which it typically controls, manages, and hosts in private data centers. However, the organization can also outsource hosting and operation to a third-party service provider. Amazon's Virtual Private Cloud is an example of a private cloud in an external provider setting.
- **Public cloud:** Multiple organizations (tenants) use a private cloud on a shared basis. This private cloud is hosted and managed by a third-party service provider. For example: Amazon's Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, and Google AppEngine
- **Community cloud:** A group of related organizations, who want to make use of a common cloud computing environment, uses the community cloud. It is managed by the participating organizations or by a third-party managed service provider. It is hosted internally or externally. For example, a community might consist of the different branches of the military, all the universities in a given region, or all the suppliers to a large manufacturer.
- **Hybrid cloud:** A single organization that wants to adopt both private and public clouds for a single application uses the hybrid cloud. A third model, the hybrid cloud, is maintained by both internal and external providers. For example, an organization might use a public cloud service, such as Amazon Simple Storage Service (Amazon S3), for archived data but continue to maintain in-house (private cloud) storage for operational customer data.

Lesson Agenda

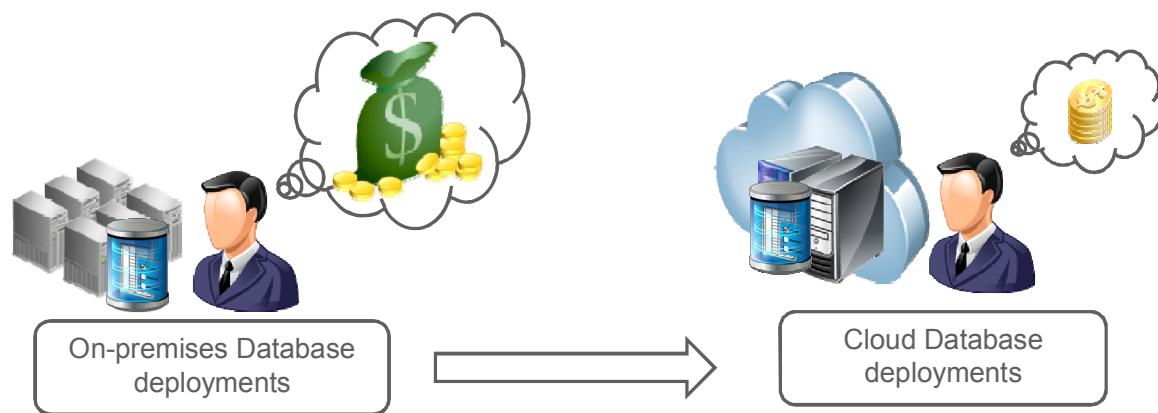
- Overview of Oracle Cloud
- Working with Oracle Database Exadata Express Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Evolving from On-premises to Exadata Express



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Cloud deployments provide end users and enterprises with different capabilities to store and process data. They enable users to have high performance and huge computing resources at a lower price as compared to traditional on-premises deployments.

Exadata Express is a powerful database machine, extended as a cloud service. End users can use it for Oracle 12c database deployments. It delivers a complete database experience for developers and enterprises.

Exadata express being a cloud deployment provides high scalability, performance and availability to its users.

It is fully managed database, therefore you need not worry about patching, upgrading or other DBA tasks.

What is in Exadata Express?

- A fully managed database service
- Provides powerful yet elastic database cloud service for developers
- Provides on-demand access to a shared pool of database resources
- Comes with built-in tools for rapid application development
 - APEX for web application development
 - Compatibility with clients such as SQL Developer, SQLcl



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Exadata Express is a fully managed database service where end users need not worry about upgrades to the database and other components of the service. All enhancements are automatically managed by the cloud service.

Being a cloud deployment, Exadata Express, allows end users to scale their data virtually to unlimited size.

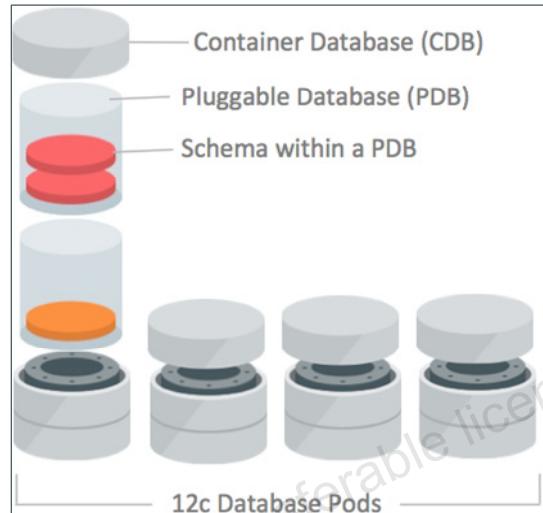
Dynamic provisioning of resources allows users to access huge amount of compute and storage resources in no time.

For developers, it provides built-in application development tool – APEX. APEX(Application Express) is a rapid web application development tool for Oracle database. Developers with minimal development experience can develop and deploy professional applications through web browser using APEX.

Oracle makes a variety of database client drivers and tools available for use with Oracle Database Exadata Express Cloud Service. You can use Exadata Express with Oracle SQL Developer, an IDE used for SQL, PL/SQL development and Oracle SQLcl, an enhanced command line interface.

Exadata Express for Users

- Oracle manages the service as multiple Container databases(CDBs), also known as database pods
- Each CDB can accommodate upto 1000 Pluggable databases(PDBs).
- Each user is provisioned with a PDB on subscribing to the service, where the user can create several schemas.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Exadata Express is ideal for production applications that consist of small to medium sized data as well as developers, testers, evaluators and other users who are looking for a full Oracle Database experience at an affordable entry-level price. It is a fully managed database service, is organized into Container databases(CDBs). These container databases are also known as database pods.

Each container database in turn can contain several Pluggable databases(PDBs). When a user subscribes to the Exadata Service, a pluggable database is provisioned. Within the PDB, the user can create several schemas. However, PDB Services are constrained by CPU, storage and memory.

Exadata Express for Developers

- Developers can connect with a wide range of data sources for their applications
 - JSON Document Storage
 - Document Style data access
 - Oracle Rest Data Services



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSON Document Storage - Oracle Database in Exadata Express provides direct storage, access and management of JSON documents. See JSON Support in Oracle Database New Features Guide 12c Release 2 (12.2).

Document-Style Data Access - Oracle Database in Exadata Express gives you the ability to store and access data as schema-less documents and collections using the Simple Oracle Document Access (SODA) API. See Working with JSON and Other Data Using SODA in Using Oracle Database Exadata Express Cloud Service.

Oracle REST Data Services 3 - Exadata Express includes the newest Oracle REST Data Services (ORDS). With ORDS 3, it's easy to develop modern RESTful interfaces for relational data and now JSON documents stored in Oracle Database.

Getting Started with Exadata Express

1. Purchase a subscription.
2. Activate and verify the service.
3. Verify activation.
4. Learn about users and roles.
5. Create accounts for your users and assign them appropriate privileges and roles.
6. Set the password for the database user authorized to perform administrative tasks for your service (PDB_ADMIN).

Note: You can refer to Using Oracle Database Exadata Express Cloud Service (<https://docs.oracle.com/cloud/latest/exadataexpress-cloud/CSDBP/toc.htm>) for details on the subscription process.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The steps in the slide provide you a brief overview of how to get started with Exadata Express. In the following slides, you will learn in detail how to perform the aforementioned steps.

Oracle Exadata Express Cloud Service

You can refer to Working with Oracle Database Exadata Express Cloud Service (<http://oukc.oracle.com/public/redir.html?type=player&offid=1984115860>) to gain an introduction to the service and its features.



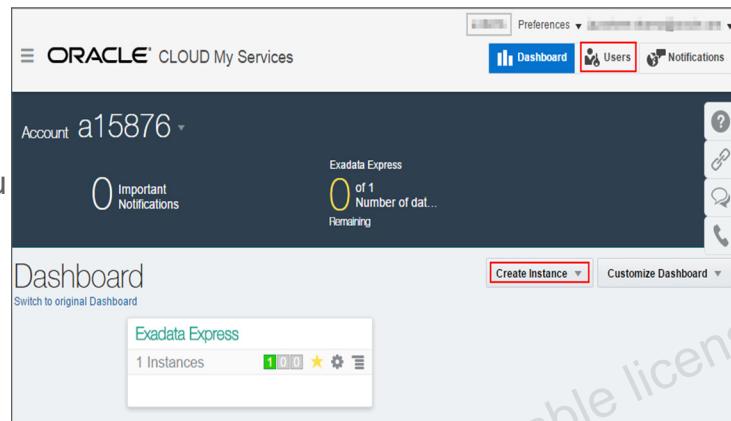
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In this video, you gain an introduction to Oracle Exadata Express Cloud Service and its features. You will take a tour of its service console and also learn about the different database clients such as Oracle SQL Developer, SQL CL, SQL Workshop and SQL * Plus that can be used to connect to Oracle Exadata Express Cloud Service.

Note: This demonstration has audio, which cannot be played inside OU Classroom. However participants can access this link on open internet and review at their convenience.

Getting Started with Exadata Express

- On signing into the service, you get access to the dashboard.
- Dashboard allows you to create database instances and users.
- The number of instances you create is limited by the amount of resources you have access to.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

After successful subscription to the service, you can login to your account and access the dashboard. Based on the type of subscription you can create instances.

The instances would appear on the dashboard. You can see an instance created in the image . To manage the instance, click on the instance.

Managing Exadata

The screenshot shows two main panels. On the left is the 'Service Instances' dashboard for the 'exa4' instance. It displays basic information: Service Type: Exadata Express, Instance Id: 500033811, Status: Active, Size: BASIC, Service SFTP User Name: us148271, and Service SFTP Host & Port: den00rcus.oracle.com. A red box highlights the 'Open Service Console' button. An arrow points from this button to the 'Service Console: exa4' interface on the right. The 'Service Console' interface is titled 'Service Console: exa4'. It contains several sections: 'Web Access' (Develop database and web apps using Oracle Application Express (APEX)), 'Define REST Data Services' (Create and manage RESTful web service interfaces to your database), 'Develop with App Builder' (Declaratively develop and deploy data-driven apps), 'Install Productivity Apps' (Browse and install productivity apps), 'Download Client Credentials' (Download a zip file containing your security credentials and network configuration files), 'Disable Client Access' (Disable SQL*Net access and invalidate all existing client credential files), 'Download Drivers' (Get database drivers for Java, .NET, Node.js, Python, PHP, Ruby, C, C++, Instant Client and more), 'Download Tools' (Get SQL*Plus command-line and developer tools including SQL Developer and JDeveloper), 'Administration' (Manage your cloud database), 'Create Database Schema' (Create a new schema for database objects), 'Set Administrator Password' (Set or reset your database's privileged user (PDB_ADMIN) account password), and 'Create Document Store' (Enable or disable a schema-less document-style interface, with JSON storage and access). Each section includes 'Learn More' and 'Watch Video' links.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

On clicking the instance on the dashboard, you see various details about the dashboard. You can access the services by clicking on 'Open Service Console'.

The service console provides you access to tools for Web Access, Client Access and Administration.

Service Console

- Service Console is the interface to use and manage the Exadata service
- It provides three different perspectives of the instance
 - Web Access
 - Client Access
 - Administration

The screenshot shows the Oracle Service Console interface titled "Service Console: exa4". It is divided into three main sections:

- Web Access:** Develop database and web apps using Oracle Application Express (APEX). Includes links to "Go to SQL Workshop", "Define REST Data Services", and "Install Productivity Apps".
- Client Access:** Enable database client access, then connect using drivers and tools. Includes links to "Download Client Credentials", "Disable Client Access", and "Download Drivers".
- Administration:** Manage your cloud database. Includes links to "Create Database Schema", "Set Administrator Password", and "Create Document Store".

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Web access provides utilities which enable you to develop database and web applications using Oracle Application Express(APEX).

Database clients can connect to Exadata Express service using SQL *Net Access. Some examples of supported database clients are SQLcl, SQL Developer, SQL *Plus, JDBC Thin client,ODP.NET, OCI and Instant Client. Client Access in the Service Console allows you to configure with the client you use.

Administration in the Service Console provides for performing administration tasks such as create new database schemas for database objects, set or reset administration password, create a schema-less documents and collections interface, and use administrative options to manage Oracle Application Express.

Web Access through Service Console

To execute SQL and PL/SQL

For quick development of applications

To define RESTful services on the database

Reuse apps from pre-built APEX applications

Web Access

Go to SQL Workshop

Run SQL commands, execute SQL scripts and browse database objects

Develop with App Builder

Declaratively develop and deploy data-driven apps

Define REST Data Services

Create and manage RESTful web service interfaces to your database

Install Productivity Apps

Browse and install productivity apps

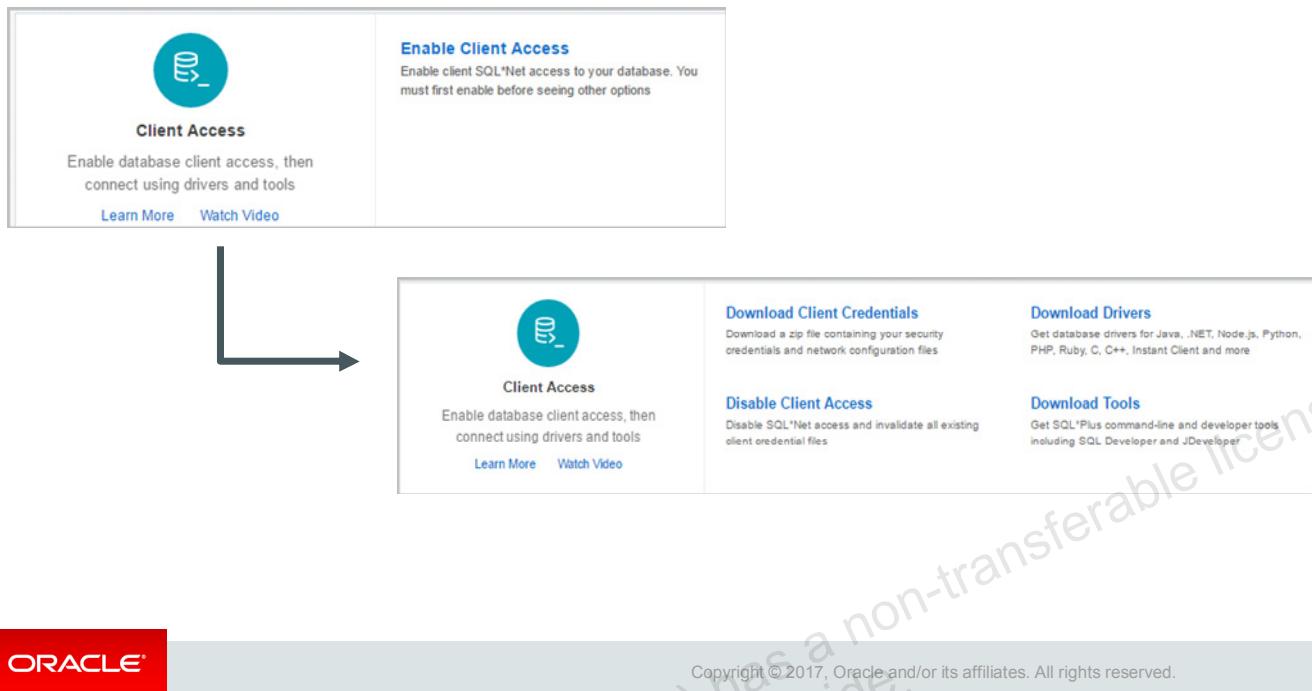
Learn More Watch Video

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Option	Description
Go to SQL Workshop	Allows you go directly to browser-based SQL Workshop, where you can run SQL statements, execute scripts and explore database objects.
Develop with App Builder	Quickly declaratively develop database and websheet applications. You can import files such as database applications and plug-ins. There is a dashboard showing metrics about your applications and workspace utilities to manage defaults, themes, metadata, exports, and more.
Define REST Data Services	Directly access the page to define and manage RESTful web services that view and manipulate data objects within your database.
Install Productivity Apps	Install from a gallery of pre-built Oracle Application Express Productivity Apps.

Client Access Configuration through Service Console



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You have to enable client access to allow SQL*Net Access to your service. Using SQL *Net Access software you can connect the Exadata instance to different clients. This option is only available when client access has not yet been enabled. Once you enable the client access, four options appear in the console:

Option	Description
Download Client Credentials	Download client credentials needed for clients to access your service.
Download Drivers	Go directly to the Oracle Technology Network page to download and install database drivers including for Java, Instant Client, C, C++, Microsoft .NET, Node.js, Python, PHP, Ruby, and more.
Disable Client Access	Use this option to disallow SQL*Net access to your service. This option is only available when client access has been enabled.
Download tools	Go directly to the Oracle Technology Network page to download and install tools such as SQL*Plus, SQLcl, command-line and integrated development environments such as Oracle SQL Developer, JDeveloper, Oracle JET, and more

Database Administration through Service Console

The screenshot shows the Oracle Database Service Console. On the left, there's a navigation bar with a gear icon labeled "Administration" and sub-options "Manage your cloud database", "Learn More", and "Watch Video". The main content area has four cards:

- Create Database Schema**: Create a new schema for database objects.
- Set Administrator Password**: Set or reset your database's privileged user (PDB_ADMIN) account password.
- Create Document Store**: Enable or disable a schema-less document-style interface, with JSON storage and access.
- Manage Application Express**: Use Application Express (APEX) administrative options.

Callout boxes provide additional context:

- An arrow points from the "Create Database Schema" callout to the "Create Database Schema" card.
- An arrow points from the "Set Administrator Password" callout to the "Set Administrator Password" card.
- An arrow points from the "Create Document Store" callout to the "Create Document Store" card.
- An arrow points from the "Manage Application Express" callout to the "Manage Application Express" card.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can perform various administration tasks through 'Administration' in the service console.

Option	Description
Create Database Schema	Create a new schema for database objects. Schema is the set of database objects, such as tables and views that belong to that user account.
Create Document Store	This option enables you to create a document store, using either an existing schema or new schema, and to enable SODA for REST, which enables REST-based operations on the schema using Oracle's SODA for REST API. It also enables SODA for Java, which is Oracle's SODA for Java API for use with Java programs.
Set Administrator Password	Use this option to set the password for the PDB_ADMIN database user that is authorized to perform administrative tasks.
Manage Application Express	Options here allow you to enable application archiving to archive your Oracle Application Express applications to database tables, manage the association between schemas and Oracle Application Express, and manage messages and set preferences for the workspace.

SQL Workshop

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Web Access

Go to SQL Workshop

Run SQL commands, execute SQL scripts and browse database objects

Define REST Data Services

Create and manage RESTful web service interfaces to your database

Install Productivity Apps

Browse and install productivity apps

Develop with App Builder

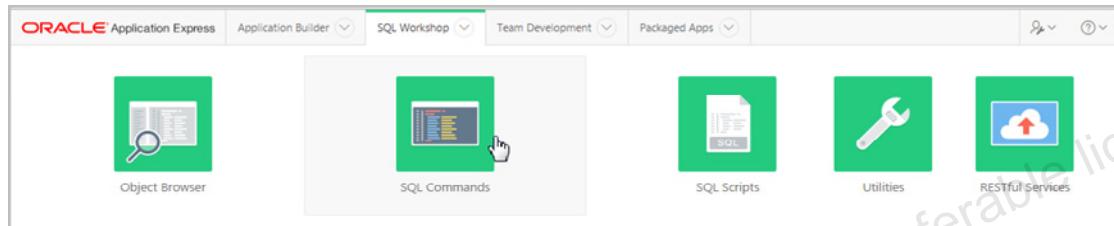
Declaratively develop and deploy data-driven apps

Learn More Watch Video



1

Clicking on SQL workshop will lead you to APEX interface



2

To run SQL or PL/SQL you can use the SQL commands utility

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL Workshop

You can run SQL statements in the editor.

The screenshot shows the Oracle Application Express SQL Workshop interface. At the top, there are tabs for Application Builder, SQL Workshop (which is selected), Team Development, and Packaged Apps. Below the tabs, a toolbar includes a magnifying glass icon, a help icon, and a user profile icon. The main area is titled "SQL Commands" and shows a schema dropdown set to "BZJNMKSSA". A "Rows" input field is set to 10, with "Clear Command" and "Find Tables" buttons nearby. A "Save" button is on the right, and a "Run" button is highlighted in blue. The SQL command entered is "SELECT * FROM emp;". The results tab is selected, displaying a table with the following data:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10
7698	BLAKE	MANAGER	7839	05/01/1981	2850	-	30
7782	CLARK	MANAGER	7839	06/09/1981	2450	-	10
7566	JONES	MANAGER	7839	04/02/1981	2975	-	20
7788	SCOTT	ANALYST	7566	12/09/1982	3000	-	20
7902	FORD	ANALYST	7566	12/03/1981	3000	-	20

Below the results table, there are tabs for Explain, Describe, Saved SQL, and History. The bottom of the interface features a red "ORACLE" logo and a copyright notice: "Copyright © 2017, Oracle and/or its affiliates. All rights reserved."

Connecting through Database Clients

You can connect to Exadata Express through various database clients.

Some of the database clients include:

- SQL*Plus
- SQLcl
- SQL Developer
- .Net and Visual Studio
- JDBC Thin Client



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You must first configure database clients and Oracle Database Exadata Express Cloud Service to communicate with each other.

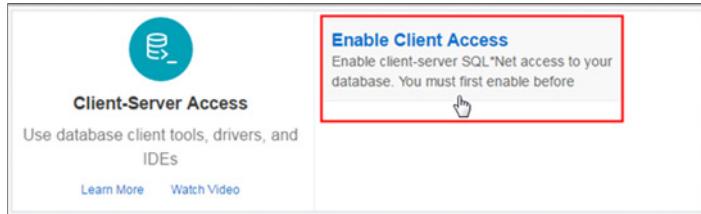
Prerequisite tasks for database client connectivity require you to:

- Enable SQL*Net access to your service.
- Download client credentials.
- Follow set-up instructions for the specific database client you want to connect with.

In the following topics, you will learn how to enable SQL*Net access, download client credentials and make connections using Oracle SQL Developer and SQLcl.

Enabling SQL*Net Access for Client Applications

Enable SQL*Net Access in the Service Console to obtain the various Database Client options.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

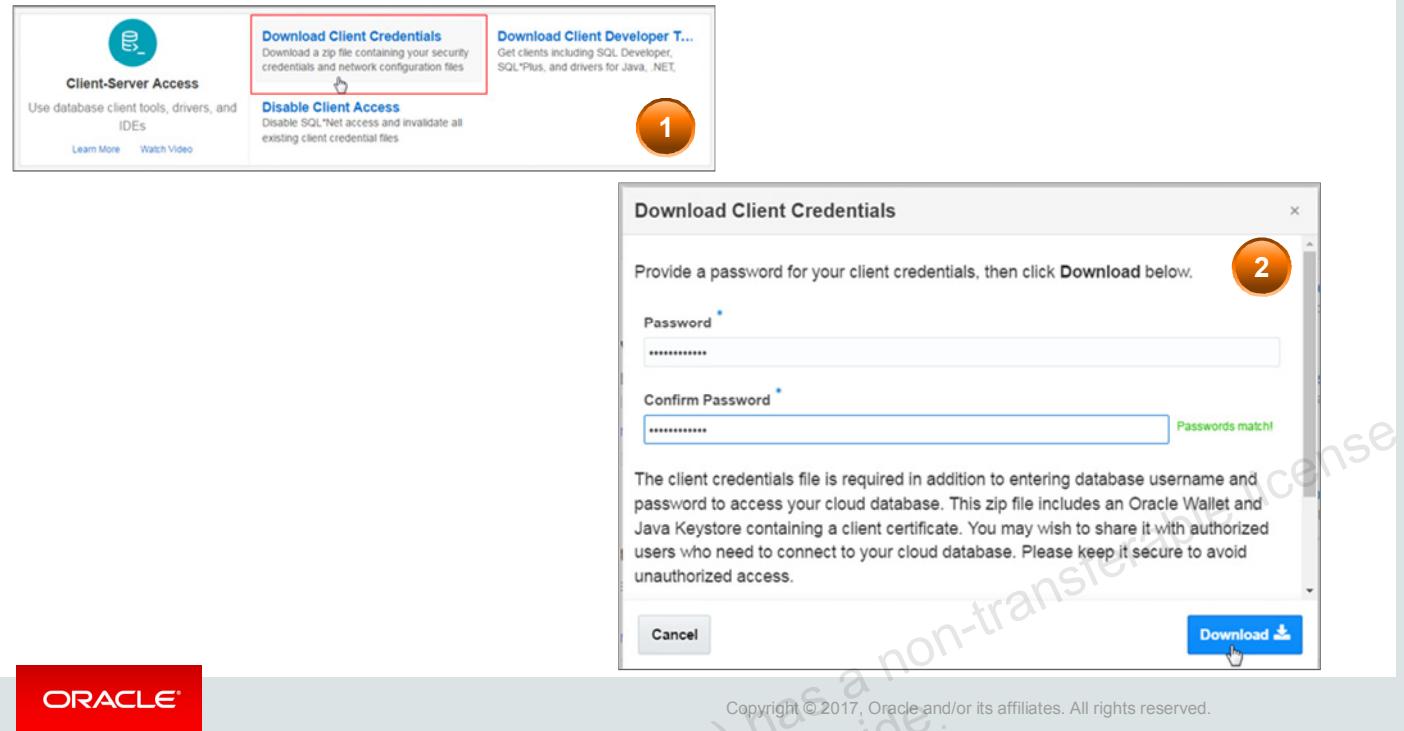
You can connect to your Exadata Express cloud service from diverse database clients over SQL*Net also called as Oracle Database Net Services. Some examples of supported clients include SQL*Plus, SQLcl, SQL Developer, JDBC Thin, ODP.NET, OCI, and Instant Client.

Database drivers for all popular programming and scripting languages such as Python, PHP, Node.js, C/C++, Ruby and Perl are supported. SQL*Net access has to be enabled as a prerequisite for all clients and drivers connecting over SQL*Net.

The Service Administrator must do the following to enable SQL*Net:

- Navigate to the Service Console for **Exadata Express** and open the service console.
- Click **Enable Client Access**.
- Download the client credentials.
- Now, depending on the client-side application and driver being used, you need to configure the application connection string for that application.

Downloading Client Credentials



You can easily download the zip files for client credentials from service console. Its contents include Oracle Wallet and Java Keystore as well as essential client configuration files. While downloading the zip file, you are prompted to enter a password.

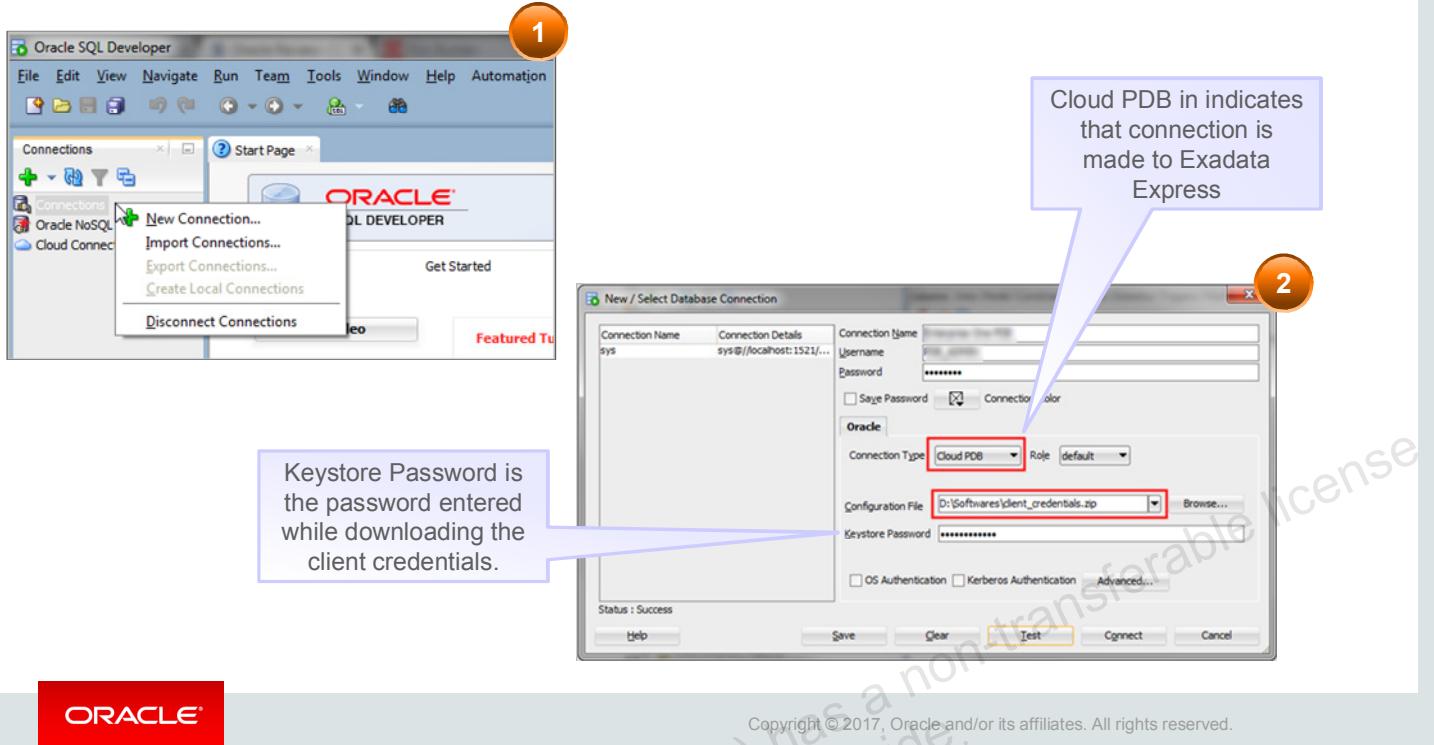
Note that client credential zip files should be carefully managed. Please remember to keep the file secure to avoid unauthorized database access. If you believe the security of this file has been compromised, then immediately disable client access using the cloud service console.

The steps to download the client credentials are as follows.

1. Navigate to Exadata Express and open the service console.
2. Click Download Client Credentials to download a zip file containing your security credentials and network configuration files.
3. Enter a password to create a password-protected Oracle Wallet and Java Keystore files for the service.
4. Click Download and save the downloaded zip file to a secure location that is accessible by your database client(s).

In the following topics, you will learn how to use these credentials to connect to the cloud database.

Connecting Oracle SQL Developer



In order to connect to Oracle Database Exadata Express cloud service, you need to download and install Oracle SQL Developer Release 4.1.5 or later. You should have also downloaded the Client Credentials from Oracle Exadata Express service console. You should then configure an Oracle Cloud connection in the Oracle SQL Developer.

The connection can be created as follows:

1. Run Oracle SQL Developer locally.
2. Under Connections, right click Connections and select **New Connection**.
3. Enter the following details:
 - Connection Name – Enter a name for this cloud connection.
 - Username – Enter username required to sign into Exadata Express.
 - Password – Enter password required to sign into Exadata Express.
 - Connection type – Select **Cloud PDB**.
 - Configuration File - Click Browse and select the **Client Credentials** zip file that you previously downloaded from the Exadata Express service console.
 - Keystore Password – Enter the password provided while downloading the Client Credentials from the Exadata Express service console.
4. Click Test. If the status is Success, click Connect.

If you have connected successfully, the tables and other objects from Exadata Express display under the new connection.

Connecting Oracle SQLcl

```
D:\PDB Service\SQL CL\sqlcl-no-jre-latest\sqlcl\bin>sql /nolog  
SQLcl: Release 4.2.0.16.160.2007 RC on Thu Sep 08 12:18:07 2016  
Copyright (c) 1982, 2016, Oracle. All rights reserved.  
SQL>
```

1

```
SQL> set cloudconfig client_credentials.zip  
Wallet Password: *****  
Using temp directory:C:\Users\APOTHU~1.ORA\AppData\Local\Temp\  
oracle_cloud_config6707346342028726502
```

2

```
SQL> conn pdb_admin/welcome1@dbaccess  
Connected.  
SQL>
```

3



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned about:

- The salient features of Oracle Cloud
- Oracle Database Exadata Express Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Relational database management systems are composed of objects or relations. They are managed by operations and governed by data integrity constraints.

Oracle Corporation produces products and services to meet your RDBMS needs. The main products are the following:

- Oracle Database, which you use to store and manage information by using SQL
- Oracle Fusion Middleware, which you use to develop, deploy, and manage modular business services that can be integrated and reused
- Oracle Enterprise Manager Grid Control, which you use to manage and automate administrative tasks across sets of systems in a grid environment

SQL

The Oracle server supports ANSI-standard SQL and contains extensions. SQL is the language that is used to communicate with the server to access, manipulate, and control data.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Table Descriptions

ORACLE

Copyright ©2017, Oracle and/or its affiliates. All rights reserved.

Schema Description

Overall Description

The Oracle Database sample schemas portray a sample company that operates worldwide to fill orders for several different products. The company has three divisions:

- **Human Resources:** Tracks information about employees and facilities
- **Order Entry:** Tracks product inventories and sales through various channels
- **Sales History:** Tracks business statistics to facilitate business decisions

Each of these divisions is represented by a schema. In this course, you have access to the objects in all the schemas. However, the emphasis of the examples, demonstrations, and practices is on the Human Resources (HR) schema.

All scripts necessary to create the sample schemas reside in the \$ORACLE_HOME/demo/schema/ folder.

Human Resources (HR)

This is the schema that is used in this course. In the Human Resource (HR) records, each employee has an identification number, email address, job identification code, salary, and manager. Some employees earn commissions in addition to their salary.

The company also tracks information about the jobs within the organization. Each job has an identification code, job title, and a minimum and maximum salary range for the job. Some employees have been with the company for a long time and have held different positions within the company. When an employee resigns, the duration the employee was working for, the job identification number, and the department are recorded.

The sample company is regionally diverse, so it tracks the locations of its warehouses and departments. Each employee is assigned to a department, and each department is identified either by a unique department number or a short name. Each department is associated with one location, and each location has a full address that includes the street name, postal code, city, state or province, and the country code.

In places where the departments and warehouses are located, the company records details such as the country name, currency symbol, currency name, and the region where the country is located geographically.

Human Resources (HR) Table Descriptions

DESCRIBE countries

Name	Null	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

SELECT * FROM countries

#	COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	CA	Canada	2
2	DE	Germany	1
3	UK	United Kingdom	1
4	US	United States of America	2

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

SELECT * FROM departments

#	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

DESCRIBE employees

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

SELECT * FROM employees

#	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17-JUN-11	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-09	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-09	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-14	AC_MGR	12008	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-15	IT_PROG	6000	(null)	103	60
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-15	IT_PROG	4200	(null)	103	60
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-15	ST_MAN	5800	(null)	100	50
8	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-11	ST_CLERK	3500	(null)	124	50
9	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-13	ST_CLERK	3100	(null)	124	50
10	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-14	ST_CLERK	2600	(null)	124	50
11	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-14	ST_CLERK	2500	(null)	124	50
12	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-16	SA_MAN	10500	0.2	100	80
13	174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-12	SA REP	11000	0.3	149	80
14	176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-14	SA REP	8600	0.2	149	80
15	178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-15	SA REP	7000	0.15	149	(null)
16	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-11	AD_ASST	4400	(null)	101	10
17	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-12	MK MAN	13000	(null)	100	20
18	202	Pat	Fay	PFAY	603.123.6666	17-AUG-13	MK REP	6000	(null)	201	20
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-10	AC_MGR	12008	(null)	101	110
20	206	William	Gietz	WGIETZ	515.123.8181	07-JUN-10	AC_ACCOUNT	8300	(null)	205	80

DESCRIBE job_history

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT * FROM job_history

#	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-09	24-JUL-14	IT_PROG	60
2	101	21-SEP-09	27-OCT-16	AC_ACCOUNT	110
3	101	28-OCT-09	15-MAR-13	AC_MGR	110
4	201	17-FEB-12	19-DEC-15	MK_REP	20
5	114	24-MAR-14	31-DEC-15	ST_CLERK	50
6	122	01-JAN-15	31-DEC-15	ST_CLERK	50
7	200	17-SEP-95	17-JUN-09	AD_ASST	90
8	176	24-MAR-14	31-DEC-14	SA_REP	80
9	176	01-JAN-15	31-DEC-15	SA_MAN	80
10	200	01-JUL-10	31-DEC-14	AC_ACCOUNT	90

DESCRIBE jobs

Name	Null	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

SELECT * FROM jobs

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1 AD_PRES	President	20080	40000
2 AD_VP	Administration Vice President	15000	30000
3 AD_ASST	Administration Assistant	3000	6000
4 AC_MGR	Accounting Manager	8200	16000
5 AC_ACCOUNT	Public Accountant	4200	9000
6 SA_MAN	Sales Manager	10000	20080
7 SA_REP	Sales Representative	6000	12008
8 ST_MAN	Stock Manager	5500	8500
9 ST_CLERK	Stock Clerk	2008	5000
10 IT_PROG	Programmer	4000	10000
11 MK_MAN	Marketing Manager	9000	15000
12 MK_REP	Marketing Representative	4000	9000

DESCRIBE locations

Name	Null	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

SELECT * FROM locations

#	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1	1400 2014 Jabberwocky Rd	26192	Southlake	Texas	US
2	2	1500 2011 Interiors Blvd	99236	South San Francisco	California	US
3	3	1700 2004 Charade Rd	98199	Seattle	Washington	US
4	4	1800 460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
5	5	2500 Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

```
DESCRIBE regions
```

Name	Null	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

```
SELECT * FROM regions
```

	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Copyright©2017, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- List the key features of Oracle SQL Developer
- Identify the menu items of Oracle SQL Developer
- Create a database connection
- Manage database objects
- Use SQL Worksheet
- Save and run SQL scripts
- Create and save reports
- Browse the Data Modeling options in SQL Developer



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

What Is Oracle SQL Developer?

- Oracle SQL Developer is a graphical tool that enhances productivity and simplifies database development tasks.
- You can connect to any target Oracle database schema by using standard Oracle database authentication.



SQL Developer

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer is a free graphical tool designed to improve your productivity and simplify the development of everyday database tasks. With just a few clicks, you can easily create and debug stored procedures, test SQL statements, and view optimizer plans.

SQL Developer, which is the visual tool for database development, simplifies the following tasks:

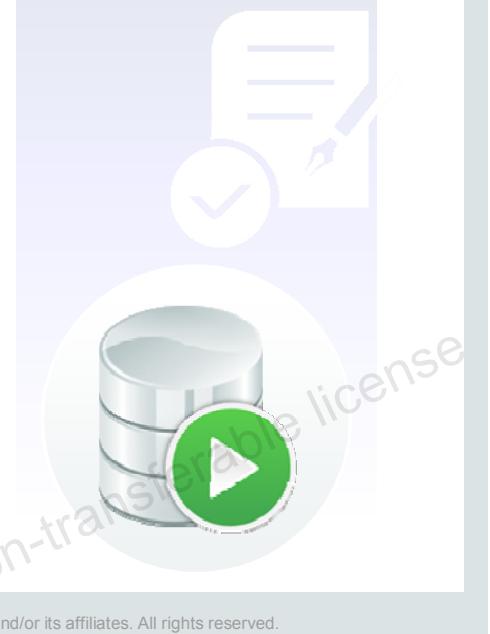
- Browsing and managing database objects
- Executing SQL statements and scripts
- Editing and debugging PL/SQL statements
- Creating reports

You can connect to any target Oracle database schema by using standard Oracle database authentication. When connected, you can perform operations on objects in the database.

SQL Developer is the interface to administer the Oracle Application Express Listener. The new interface enables you to specify global settings and multiple database settings with different database connections for the Application Express Listener. SQL Developer provides the option to drag and drop objects by table or column name onto the worksheet. It provides improved DB Diff comparison options, GRANT statement support in the SQL editor, and DB Doc reporting. Additionally, SQL Developer includes support for Oracle Database 12c features.

Specifications of SQL Developer

- Is shipped along with Oracle Database 12c Release 1
- Is developed in Java
- Supports Windows, Linux, and Mac OS X platforms
- Enables default connectivity using the JDBC Thin driver
- Connects to Oracle Database version 9.2.0.1 and later



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer is shipped along with Oracle Database 12c Release 1 by default. SQL Developer is developed in Java, leveraging the Oracle JDeveloper integrated development environment (IDE). Therefore, it is a cross-platform tool. The tool runs on Windows, Linux, and Mac operating system (OS) X platforms.

The default connectivity to the database is through the Java Database Connectivity (JDBC) Thin driver, and therefore, no Oracle Home is required. SQL Developer does not require an installer and you need to simply unzip the downloaded file. With SQL Developer, users can connect to Oracle Databases 9.2.0.1 and later, and all Oracle database editions, including Express Edition.

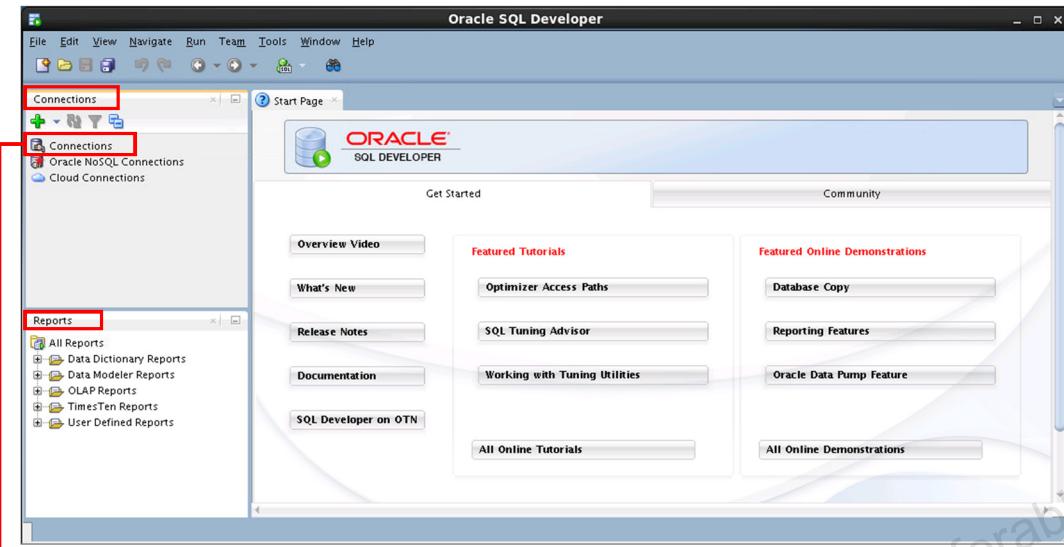
Note: For Oracle Database 12c Release 1, you will have to download and install SQL Developer. SQL Developer is freely downloadable from the following link:

<http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>

For instructions on how to install SQL Developer, see the website at:

<http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html>

SQL Developer 3.2 Interface



You must define a connection to start using SQL Developer for running SQL queries on a database schema.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The SQL Developer interface contains two main navigation tabs:

- **Connections:** By using this tab, you can browse database objects and users to which you have access.
- **Reports:** Identified by the Reports icon, this tab enables you to run predefined reports or create and add your own reports.

General Navigation and Use

SQL Developer uses the left side for navigation to find and select objects, and the right side to display information about selected objects. You can customize many aspects of the appearance and behavior of SQL Developer by setting preferences.

Note: You need to define at least one connection to be able to connect to a database schema and issue SQL queries or run procedures and functions.

Menus

The following menus contain standard entries, plus entries for features that are specific to SQL Developer:

- **View:** Contains options that affect what is displayed in the SQL Developer interface
- **Navigate:** Contains options for navigating to panes and for executing subprograms
- **Run:** Contains the Run File and Execution Profile options that are relevant when a function or procedure is selected, and also debugging options
- **Versioning:** Provides integrated support for the following versioning and source control systems—Concurrent Versions System (CVS) and Subversion
- **Tools:** Invokes SQL Developer tools such as SQL*Plus, Preferences, and SQL Worksheet. It also contains options related to migrating third-party databases to Oracle.

Note: The Run menu also contains options that are relevant when a function or procedure is selected for debugging.

Creating a Database Connection

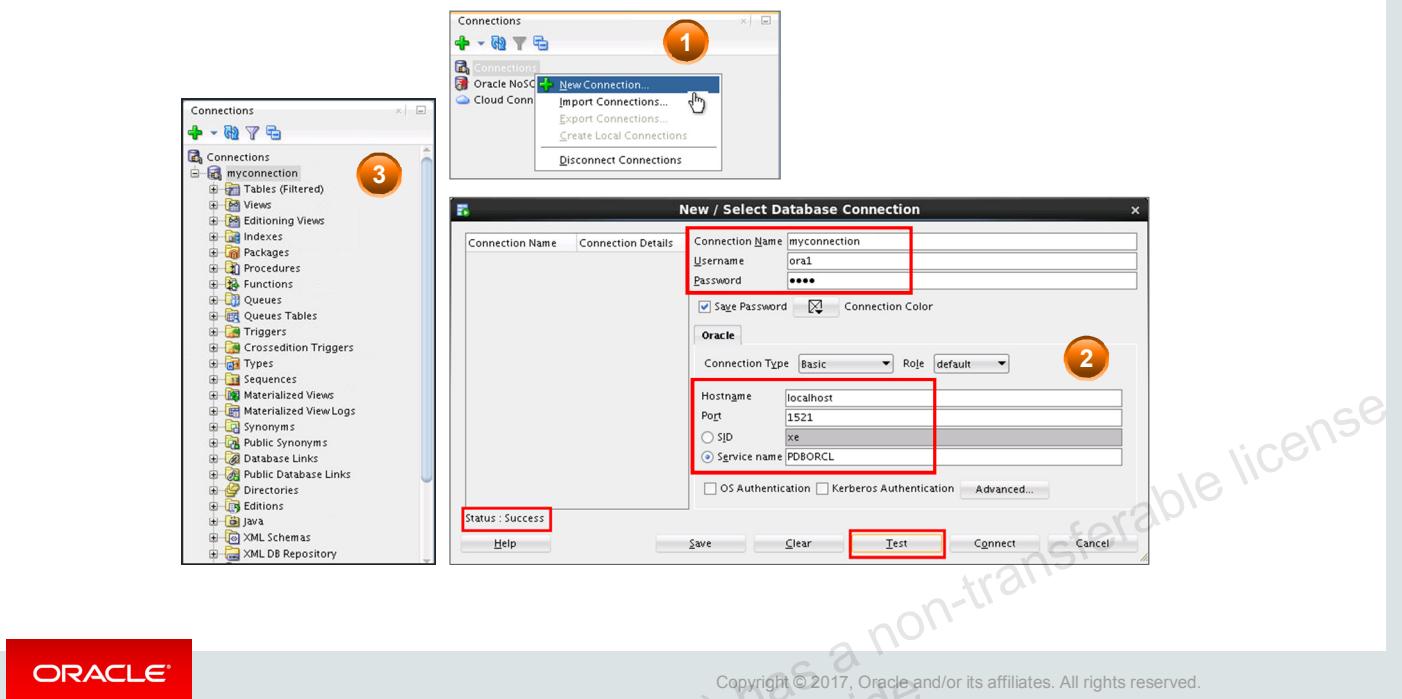
- You must have at least one database connection to use SQL Developer.
- You can create and test connections for:
 - Multiple databases
 - Multiple schemas
- SQL Developer automatically imports any connections defined in the `tnsnames.ora` file on your system.
- You can export connections to an Extensible Markup Language (XML) file.
- Each additional database connection created is listed in the Connections Navigator hierarchy.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Creating a Database Connection



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

To create a database connection, perform the following steps:

1. On the Connections tabbed page, right-click Connections and select New Connection.
2. In the New / Select Database Connection window, enter the connection name. Enter the username and password of the schema that you want to connect to.
 - a. From the Role drop-down list, you can select either *default* or SYSDBA. (You choose SYSDBA for the *sys* user or any user with database administrator privileges.)
 - b. You can select the connection type as:
 - Basic:** In this type, enter host name and SID for the database that you want to connect to. Port is already set to 1521. You can also choose to enter the Service name directly if you use a remote database connection.
 - TNS:** You can select any one of the database aliases imported from the *tnsnames.ora* file.
 - LDAP:** You can look up database services in Oracle Internet Directory, which is a component of Oracle Identity Management.
 - Advanced:** You can define a custom Java Database Connectivity (JDBC) URL to connect to the database.
 - Local/Bequeath:** If the client and database exist on the same computer, a client connection can be passed directly to a dedicated server process without going through the listener.

- c. Click Test to ensure that the connection has been set correctly.
- d. Click Connect.

If you select the Save Password check box, the password is saved to an XML file. So, after you close the SQL Developer connection and open it again, you are not prompted for the password.

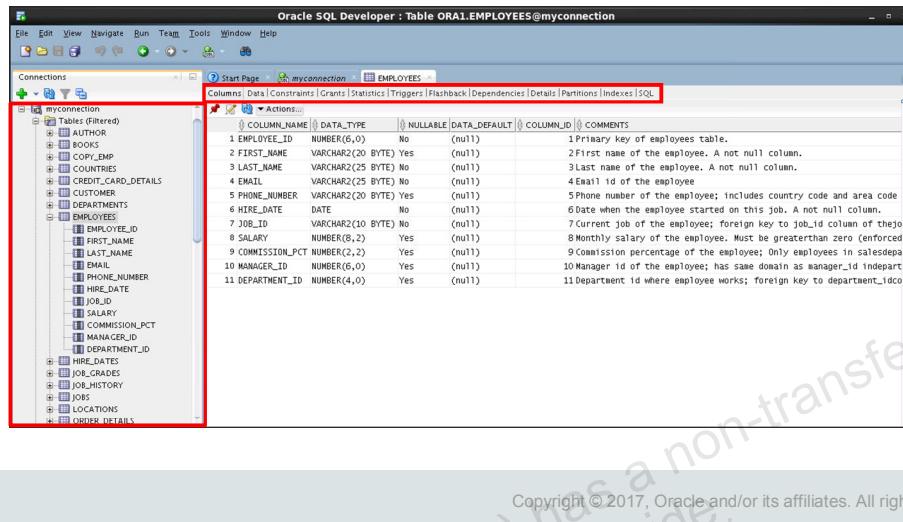
3. The connection gets added in the Connections Navigator. You can expand the connection to view the database objects and view object definitions (dependencies, details, statistics, and so on).

Note: From the same New>Select Database Connection window, you can define connections to non-Oracle data sources using the Access, MySQL, and SQL Server tabs. However, these connections are read-only connections that enable you to browse objects and data in that data source.

Browsing Database Objects

Use the Connections Navigator to:

- Browse through many objects in a database schema
- Review the definitions of objects at a glance



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

After you create a database connection, you can use the Connections Navigator to browse through many objects in a database schema, including Tables, Views, Indexes, Packages, Procedures, Triggers, and Types.

SQL Developer uses the left side for navigation to find and select objects, and the right side to display information about the selected objects. You can customize many aspects of the appearance of SQL Developer by setting preferences.

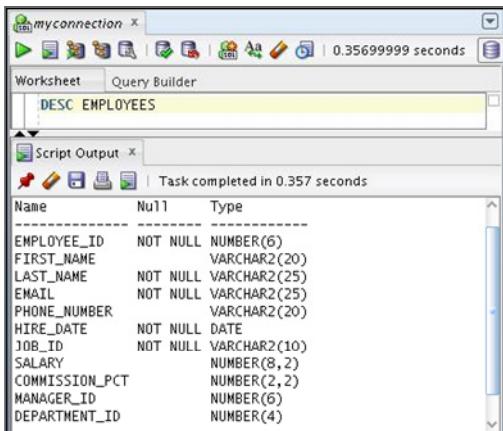
You can see the definition of the objects broken into tabs of information that is pulled out of the data dictionary. For example, if you select a table in the Navigator, details about columns, constraints, grants, statistics, triggers, and so on are displayed on an easy-to-read tabbed page.

If you want to see the definition of the `EMPLOYEES` table as shown in the slide, perform the following steps:

1. Expand the Connections node in the Connections Navigator.
2. Expand Tables.
3. Click `EMPLOYEES`. By default, the Columns tab is selected. It shows the column description of the table. Using the Data tab, you can view the table data and also enter new rows, update data, and commit these changes to the database.

Displaying the Table Structure

Use the DESCRIBE command to display the structure of a table:



The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a window titled "myconnection x". Below it, a "Worksheet" tab is active, showing the command "DESC EMPLOYEES". To the right of the worksheet is a "Script Output" tab which displays the results of the describe command. The output is a table with three columns: Name, Null, and Type. The data shows the structure of the EMPLOYEES table, including columns like EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID.

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

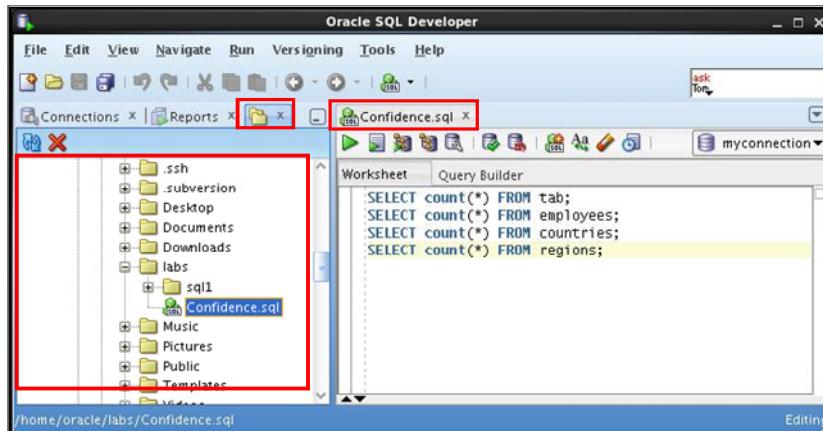


ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Browsing Files

Use the File Navigator to explore the file system and open system files.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

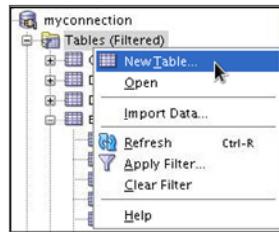
Browsing Database Objects

You can use the File Navigator to browse and open system files.

- To view the File Navigator, click the View tab and select Files, or select View > Files.
- To view the contents of a file, double-click a file name to display its contents in the SQL Worksheet area.

Creating a Schema Object

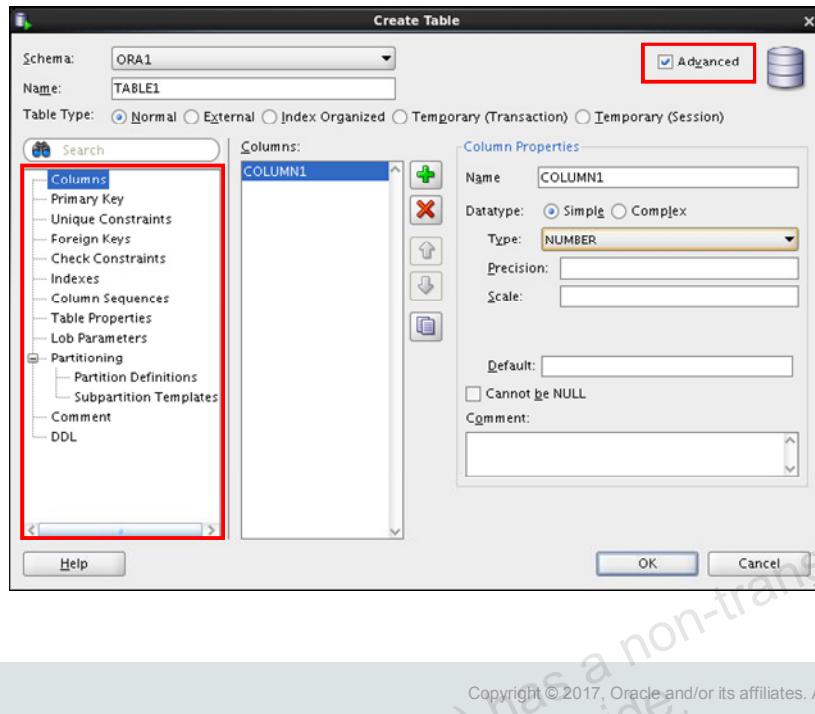
- SQL Developer supports the creation of any schema object by:
 - Executing a SQL statement in SQL Worksheet
 - Using the context menu
- Edit the objects by using an edit dialog box or one of the many context-sensitive menus.
- View the data definition language (DDL) for adjustments such as creating a new object or editing an existing schema object.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Creating a New Table: Example



In the Create Table dialog box, if you do not select the Advanced check box, you can create a table quickly by specifying columns and some frequently used features.

If you select the Advanced check box, the Create Table dialog box changes to one with multiple options, in which you can specify an extended set of features while you create the table.

The example in the slide shows how to create the DEPENDENTS table by selecting the Advanced check box.

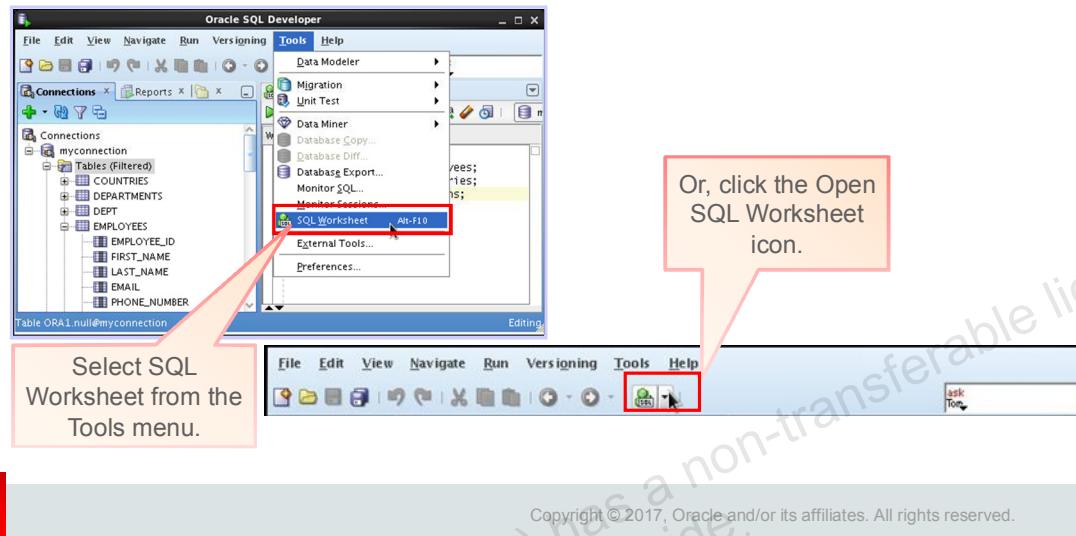
To create a new table, perform the following steps:

1. In the Connections Navigator, right-click Tables and select Create TABLE.
2. In the Create Table dialog box, select Advanced.
3. Specify the column information.
4. Click OK.

Although it is not required, you should also specify a primary key by using the Primary Key tab in the dialog box. Sometimes, you may want to edit the table that you have created; to do so, right-click the table in the Connections Navigator and select Edit.

Using the SQL Worksheet

- Use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements.
- Specify any actions that can be processed by the database connection associated with the worksheet.



When you connect to a database, a SQL Worksheet window for that connection automatically opens. You can use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements. The SQL Worksheet supports SQL*Plus statements to a certain extent. SQL*Plus statements that are not supported by the SQL Worksheet are ignored and not passed to the database.

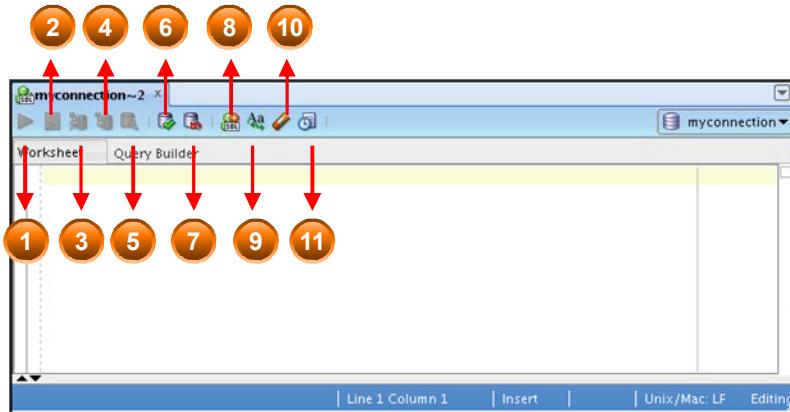
You can specify the actions that can be processed by the database connection associated with the worksheet, such as:

- Creating a table
- Inserting data
- Creating and editing a trigger
- Selecting data from a table
- Saving the selected data to a file

You can display a SQL Worksheet by using one of the following:

- Select Tools > SQL Worksheet.
- Click the Open SQL Worksheet icon.

Using the SQL Worksheet



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

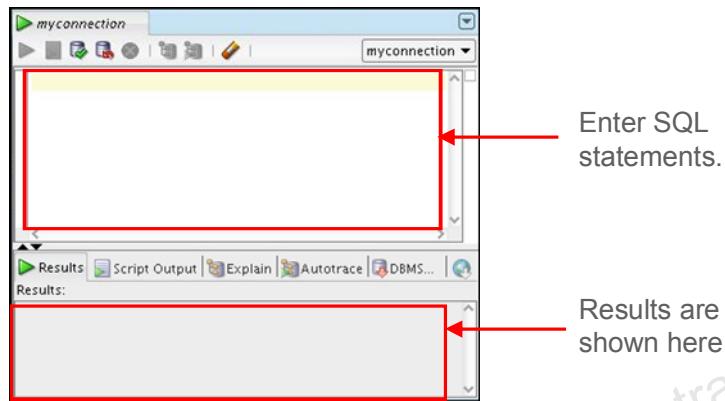
You may want to use the shortcut keys or icons to perform certain tasks such as executing a SQL statement, running a script, and viewing the history of the SQL statements that you have executed. You can use the SQL Worksheet toolbar that contains icons to perform the following tasks:

1. **Run Statement:** Executes the statement where the cursor is located in the Enter SQL Statement box. You can use bind variables in the SQL statements, but not substitution variables.
2. **Run Script:** Executes all the statements in the Enter SQL Statement box by using the Script Runner. You can use substitution variables in the SQL statements, but not bind variables.
3. **Autotrace:** Generates trace information for the statement
4. **Explain Plan:** Generates the execution plan, which you can see by clicking the Explain tab
5. **SQL Tuning Advisory:** Analyzes high-volume SQL statements and offers tuning recommendations
6. **Commit:** Writes any changes to the database and ends the transaction
7. **Rollback:** Discards any changes to the database, without writing them to the database, and ends the transaction

8. **Unshared SQL Worksheet:** Creates a separate unshared SQL Worksheet for a connection
9. **To Upper/Lower/InitCap:** Changes the selected text to uppercase, lowercase, or initcap, respectively
10. **Clear:** Erases the statement or statements in the Enter SQL Statement box
11. **SQL History:** Displays a dialog box with information about the SQL statements that you have executed

Using the SQL Worksheet

- Use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements.
- Specify any actions that can be processed by the database connection associated with the worksheet.



ORACLE

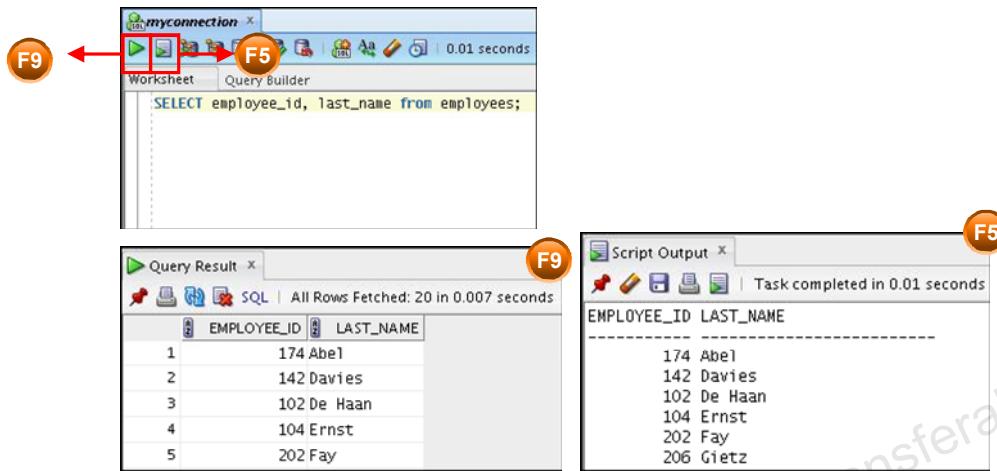
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When you connect to a database, a SQL Worksheet window for that connection automatically opens. You can use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements. All SQL and PL/SQL commands are supported as they are passed directly from the SQL Worksheet to the Oracle database. The SQL*Plus commands that are used in SQL Developer must be interpreted by the SQL Worksheet before being passed to the database.

The SQL Worksheet currently supports a number of SQL*Plus commands. Commands that are not supported by the SQL Worksheet are ignored and not sent to the Oracle database. Through the SQL Worksheet, you can execute the SQL statements and some of the SQL*Plus commands.

Executing SQL Statements

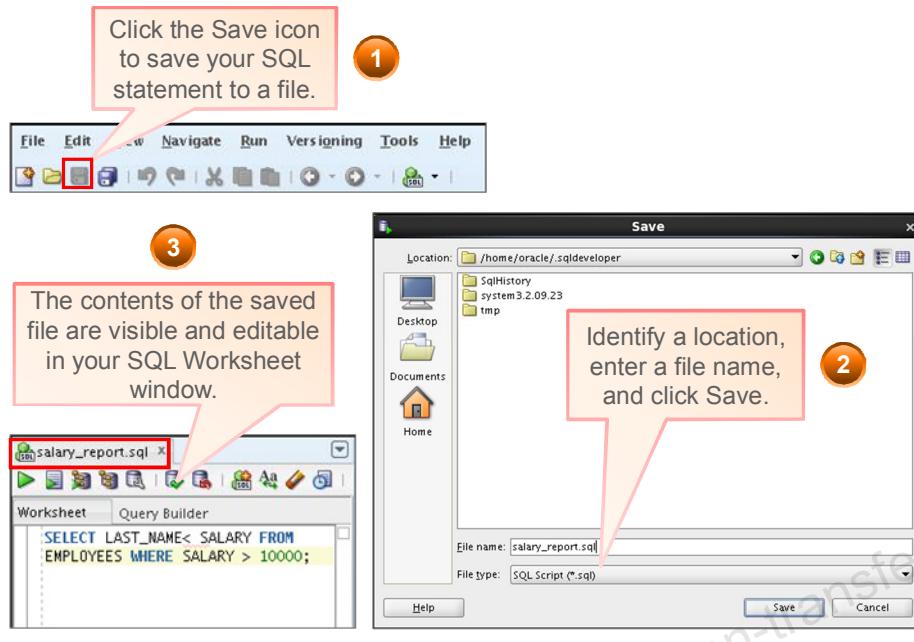
Use the Enter SQL Statement box to enter single or multiple SQL statements.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Saving SQL Scripts



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can save your SQL statements from the SQL Worksheet to a text file. To save the contents of the Enter SQL Statement box, perform the following steps:

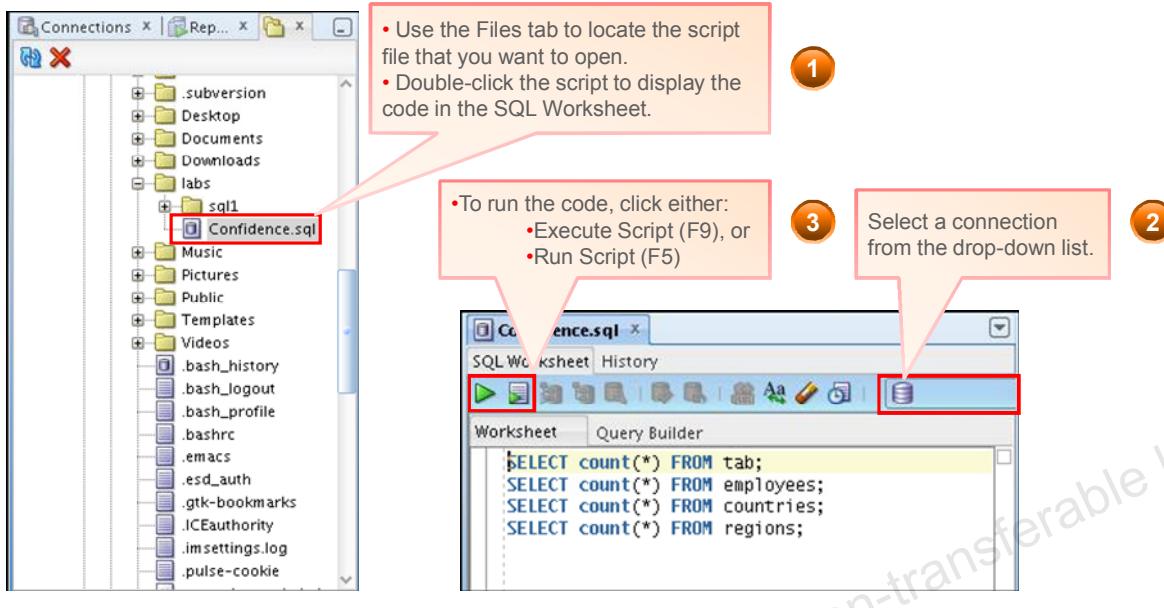
1. Click the Save icon or use the File > Save menu item.
2. In the Save dialog box, enter a file name and the location where you want the file saved.
3. Click Save.

After you save the contents to a file, the Enter SQL Statement window displays a tabbed page of your file contents. You can have multiple files open at the same time. Each file is displayed as a tabbed page.

Script Pathing

You can select a default path to look for scripts and to save scripts. Under Tools > Preferences > Database > Worksheet Parameters, enter a value in the "Select default path to look for scripts" field.

Executing Saved Script Files: Method 1



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

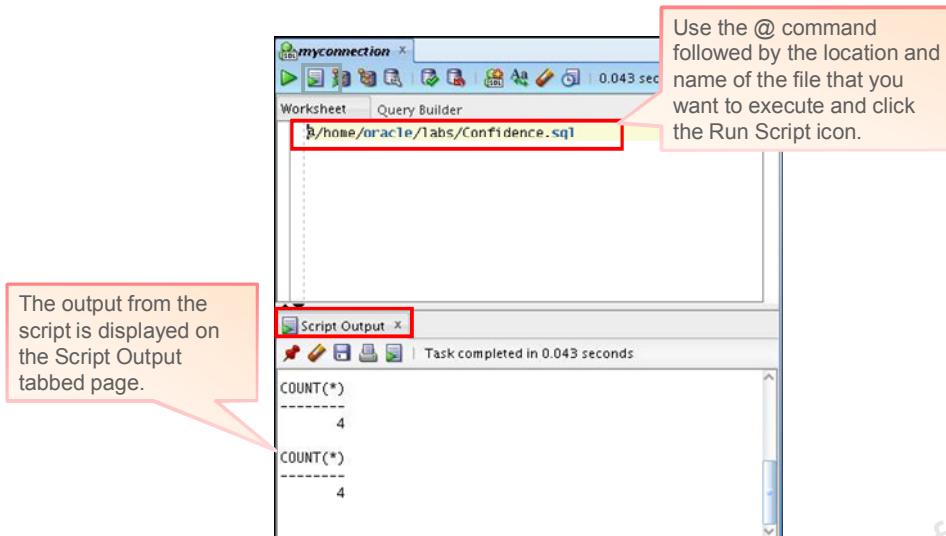
To open a script file and display the code in the SQL Worksheet area, perform the following steps:

1. In the files navigator, select (or navigate to) the script file that you want to open.
2. Double-click the file to open it. The code of the script file is displayed in the SQL Worksheet area.
3. Select a connection from the connection drop-down list.
4. To run the code, click the Run Script (F5) icon on the SQL Worksheet toolbar. If you have not selected a connection from the connection drop-down list, a connection dialog box will appear. Select the connection that you want to use for the script execution.

Alternatively, you can also do the following:

1. Select File > Open. The Open dialog box is displayed.
2. In the Open dialog box, select (or navigate to) the script file that you want to open.
3. Click Open. The code of the script file is displayed in the SQL Worksheet area.
4. Select a connection from the connection drop-down list.
5. To run the code, click the Run Script (F5) icon on the SQL Worksheet toolbar. If you have not selected a connection from the connection drop-down list, a connection dialog box will appear. Select the connection that you want to use for the script execution.

Executing Saved Script Files: Method 2



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

To run a saved SQL script, perform the following steps:

1. Use the @ command followed by the location and the name of the file that you want to run in the Enter SQL Statement window.
2. Click the Run Script icon.

The results from running the file are displayed on the Script Output tabbed page. You can also save the script output by clicking the Save icon on the Script Output tabbed page. The File Save dialog box appears and you can identify a name and location for your file.

Formatting the SQL Code

The screenshot shows the Oracle SQL Developer interface. A context menu is open over a SQL statement in the query editor. The menu items include: Run Statement (F9), Run Script (F5), Create Report..., Save as Snippet..., Autotrace..., Explain Plan..., SQL Tuning Advisor..., Commit (F11), Rollback (F12), To Upper/Lower/InitCap (Ctrl+Quote), Clear (Ctrl-D), SQL History (F8), Cut (Ctrl-X), Copy (Ctrl-C), Paste (Ctrl-V), Select All (Ctrl-A), Debug (Ctrl+Shift-F10), Refactoring, Format (Ctrl-F7) [highlighted with a red box], Advanced Format..., Code Template, Popup Describe, Open Declaration, and YES The SQL statement in the editor is:

```
select employee_id, first_name, salary from employees e, departments d  
where e.department_id=d.department_id and e.salary>3000;
```

Below the editor, the formatted SQL statement is shown:

```
SELECT employee_id,  
       first_name,  
       salary  
  FROM employees e,  
        departments d  
 WHERE e.department_id=d.department_id  
   AND e.salary>3000;
```

Two callout boxes point to the SQL code: one labeled "Before formatting" pointing to the original code, and another labeled "After formatting" pointing to the formatted code.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

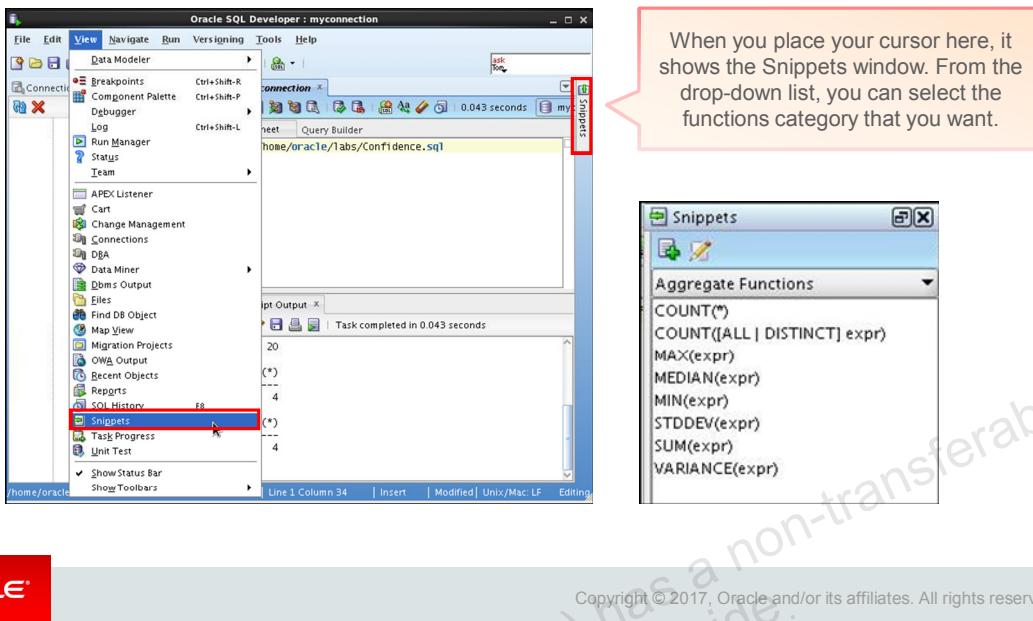
You may want to format the indentation, spacing, capitalization, and line separation of the SQL code. SQL Developer has a feature for formatting SQL code.

To format the SQL code, right-click in the statement area and select Format.

In the example in the slide, before formatting, the SQL code has the keywords not capitalized and the statement not properly indented. After formatting, the SQL code is beautified with the keywords capitalized and the statement properly indented.

Using Snippets

Snippets are code fragments that may be just syntax or examples.



When you place your cursor here, it shows the Snippets window. From the drop-down list, you can select the functions category that you want.

You may want to use certain code fragments when you use the SQL Worksheet or create or edit a PL/SQL function or procedure. SQL Developer has a feature called Snippets. Snippets are code fragments such as SQL functions, optimizer hints, and miscellaneous PL/SQL programming techniques. You can drag snippets to the Editor window.

To display Snippets, select View > Snippets.

The Snippets window is displayed on the right. You can use the drop-down list to select a group. A Snippets button is placed in the right window margin, so that you can display the Snippets window if it becomes hidden.

Using Snippets: Example

The screenshot shows two instances of the Oracle SQL Developer interface. In the top instance, a red callout box labeled "Inserting a snippet" points to the "Worksheet" tab where the code "SELECT CONCAT(char1, char2)" is being typed. To the right is the "Snippets" window, which displays a list of character functions. The "CONCAT(char1, char2)" function is selected. In the bottom instance, another red callout box labeled "Editing the snippet" points to the "Worksheet" tab where the code has been completed to "SELECT CONCAT(first_name, last_name) FROM employees;". The "Snippets" window remains visible on the right side of the interface.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

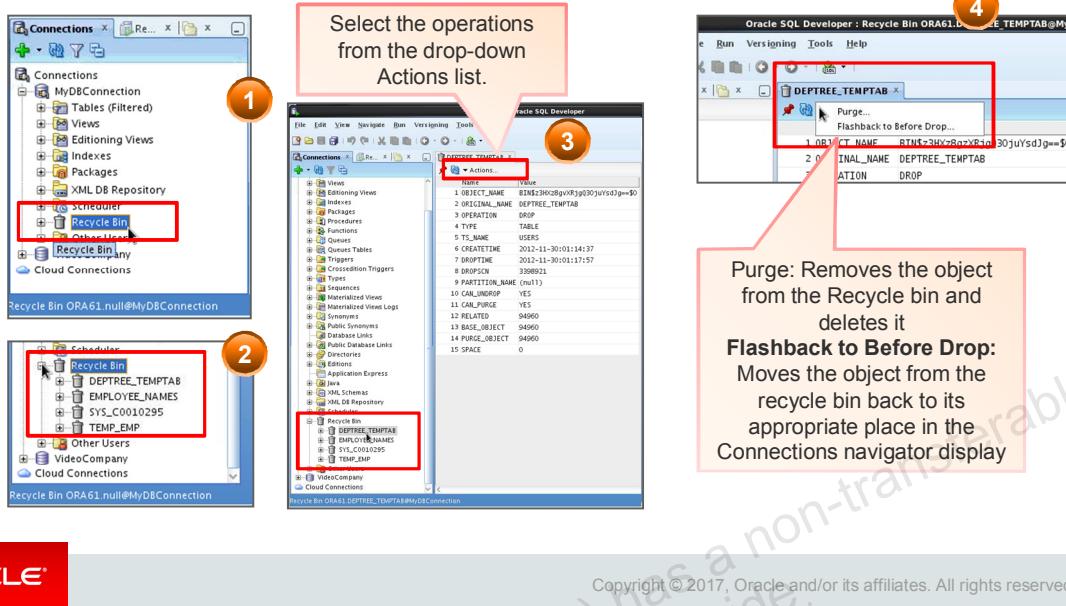
To insert a Snippet into your code in a SQL Worksheet or in a PL/SQL function or procedure, drag the snippet from the Snippets window to the desired place in your code. Then you can edit the syntax so that the SQL function is valid in the current context. To see a brief description of a SQL function in a tool tip, place the cursor over the function name.

The example in the slide shows that `CONCAT (char1, char2)` is dragged from the Character Functions group in the Snippets window. Then the `CONCAT` function syntax is edited and the rest of the statement is added as in the following:

```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

Using the Recycle Bin

The recycle bin holds objects that have been dropped.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

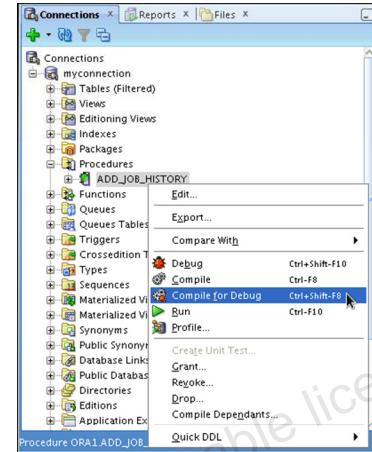
The recycle bin is a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space. They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints.

To use the recycle bin, perform the following steps:

1. In the Connections navigator, select (or navigate to) the recycle bin.
2. Expand Recycle Bin and click the object name. The object details are displayed in the SQL Worksheet area.
3. Click the Actions drop-down list and select the operation that you want to perform on the object.

Debugging Procedures and Functions

- Use SQL Developer to debug PL/SQL functions and procedures.
- Use the Compile for Debug option to perform a PL/SQL compilation so that the procedure can be debugged.
- Use the Debug menu options to set breakpoints, and to perform Step Into and Step Over tasks.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

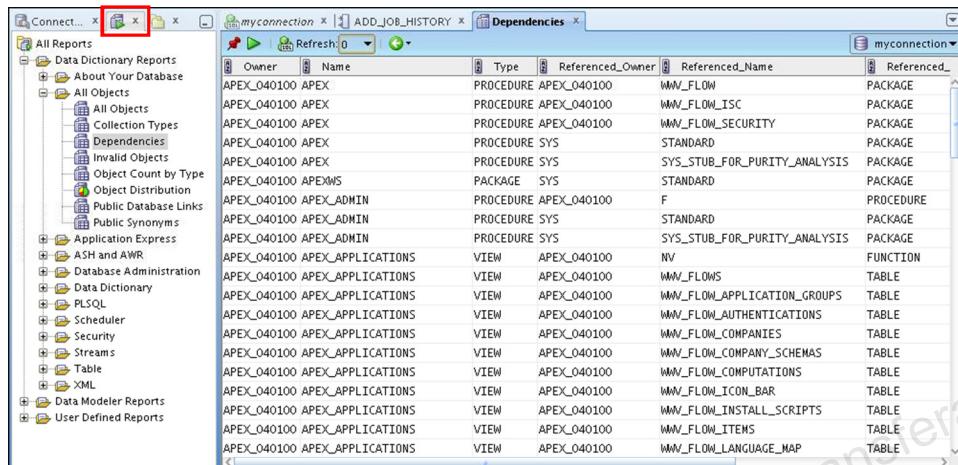
In SQL Developer, you can debug PL/SQL procedures and functions. Using the Debug menu options, you can perform the following debugging tasks:

- **Find Execution Point** goes to the next execution point.
- **Resume** continues execution.
- **Step Over** bypasses the next method and goes to the next statement after the method.
- **Step Into** goes to the first statement in the next method.
- **Step Out** leaves the current method and goes to the next statement.
- **Step to End of Method** goes to the last statement of the current method.
- **Pause** halts execution, but does not exit, thus allowing you to resume execution.
- **Terminate** halts and exits the execution. You cannot resume execution from this point; instead, to start running or debugging from the beginning of the function or procedure, click the Run or Debug icon on the Source tab toolbar.
- **Garbage Collection** removes invalid objects from the cache in favor of more frequently accessed and more valid objects.

These options are also available as icons on the Debugging tab of the output window.

Database Reporting

SQL Developer provides a number of predefined reports about the database and its objects.



The screenshot shows the Oracle SQL Developer interface. The left sidebar contains a tree view of report categories under 'All Reports'. The main pane displays a table of database objects. The table has columns: Owner, Name, Type, Referenced_Owner, Referenced_Name, and Referenced_Type. The data in the table includes various procedures, views, and packages from the APEX_040100 schema, such as APEX_040100.APEX, APEX_040100.APPLICATIONS, and APEX_040100.FUNCTION. The bottom right corner of the interface features the ORACLE logo and a copyright notice: 'Copyright © 2017, Oracle and/or its affiliates. All rights reserved.'

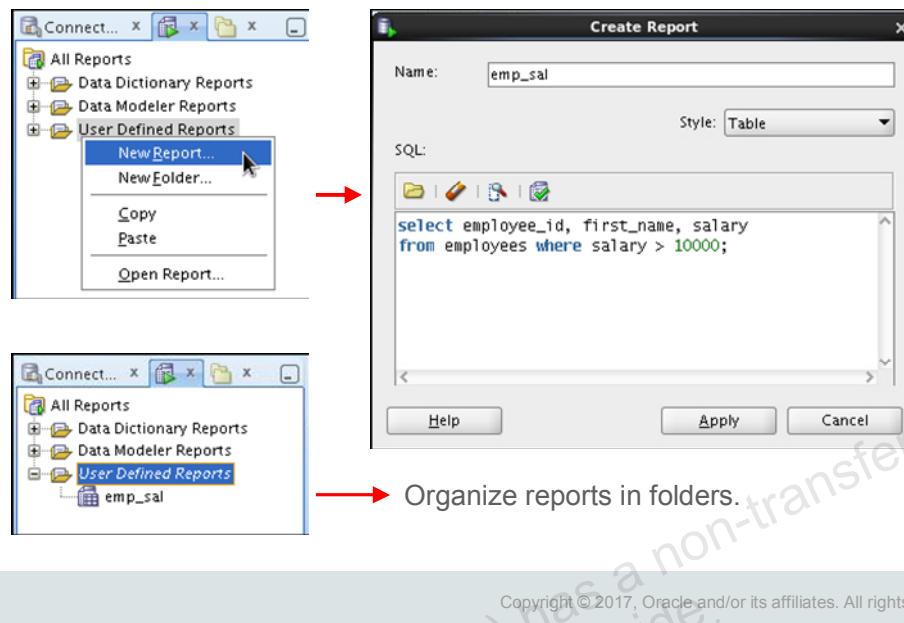
SQL Developer provides many reports about the database and its objects. These reports can be grouped into the following categories:

- About Your Database reports
- Database Administration reports
- Table reports
- PL/SQL reports
- Security reports
- XML reports
- Jobs reports
- Streams reports
- All Objects reports
- Data Dictionary reports
- User-Defined reports

To display reports, click the Reports tab on the left of the window. Individual reports are displayed in tabbed panes on the right of the window; for each report, you can select (using a drop-down list) the database connection for which to display the report. For reports about objects, the objects shown are only those visible to the database user associated with the selected database connection, and the rows are usually ordered by Owner. You can also create your own user-defined reports.

Creating a User-Defined Report

Create and save user-defined reports for repeated use.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

User-defined reports are reports created by SQL Developer users. To create a user-defined report, perform the following steps:

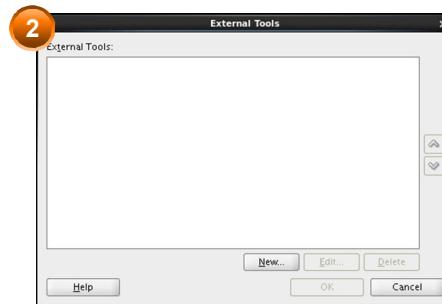
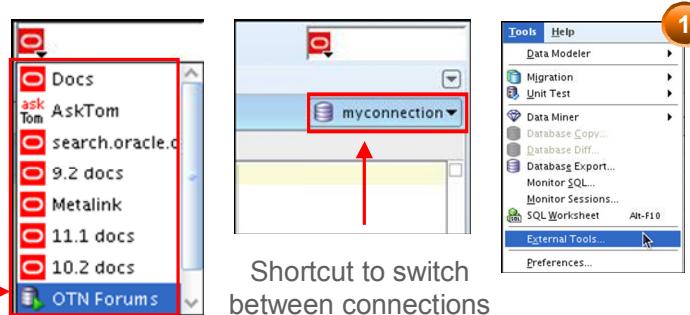
1. Right-click the User Defined Reports node under Reports and select Add Report.
2. In the Create Report dialog box, specify the report name and the SQL query to retrieve information for the report. Then click Apply.

In the example in the slide, the report name is specified as `emp_sal`. An optional description is provided indicating that the report contains details of employees with `salary >= 10000`. The complete SQL statement for retrieving the information to be displayed in the user-defined report is specified in the SQL box. You can also include an optional tool tip to be displayed when the cursor stays briefly over the report name in the Reports navigator display.

You can organize user-defined reports in folders and you can create a hierarchy of folders and subfolders. To create a folder for user-defined reports, right-click the User Defined Reports node or any folder name under that node and select Add Folder. Information about user-defined reports, including any folders for these reports, is stored in a file named `UserReports.xml` in the directory for user-specific information.

Search Engines and External Tools

Links to popular search engines and discussion forums



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

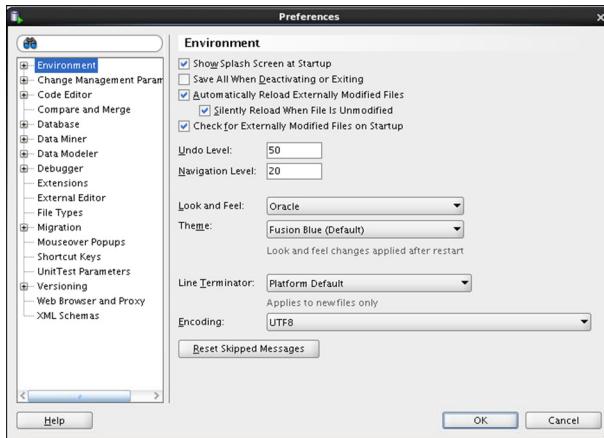
To enhance the productivity of developers, SQL Developer has added quick links to popular search engines and discussion forums such as AskTom, Google, and so on. Also, you have shortcut icons to some of the frequently used tools such as Notepad, Microsoft Word, and Dreamweaver, available to you.

You can add external tools to the existing list or even delete shortcuts to the tools that you do not use frequently. To do so, perform the following steps:

1. From the Tools menu, select External Tools.
2. In the External Tools dialog box, select New to add new tools. Select Delete to remove any tool from the list.

Setting Preferences

- Customize the SQL Developer interface and environment.
- In the Tools menu, select Preferences.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



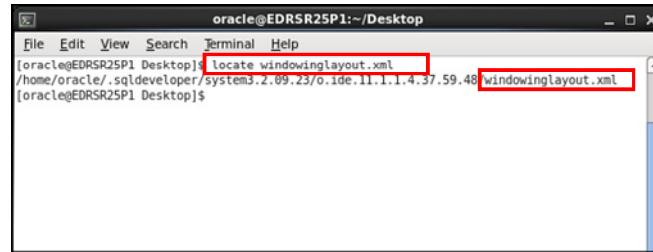
You can customize many aspects of the SQL Developer interface and environment by modifying SQL Developer preferences according to your needs. To modify SQL Developer preferences, select Tools, and then Preferences.

The preferences are grouped into the following categories:

- Environment
- Change Management Parameter
- Code Editors
- Compare and Merge
- Database
- Data Miner
- Data Modeler
- Debugger
- Extensions
- External Editor
- File Types

- Migration
- Mouseover Popups
- Shortcut Keys
- Unit Test Parameters
- Versioning
- Web Browser and Proxy
- XML Schemas

Resetting the SQL Developer Layout



```
oracle@EDRSR2SP1:~/Desktop
File Edit View Search Terminal Help
[oracle@EDRSR2SP1 Desktop] locate windowinglayout.xml
/home/oracle/.sqldeveloper/system3.2.09.23/o.ide.11.1.1.4.37.59.48>windowinglayout.xml
[oracle@EDRSR2SP1 Desktop]$
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

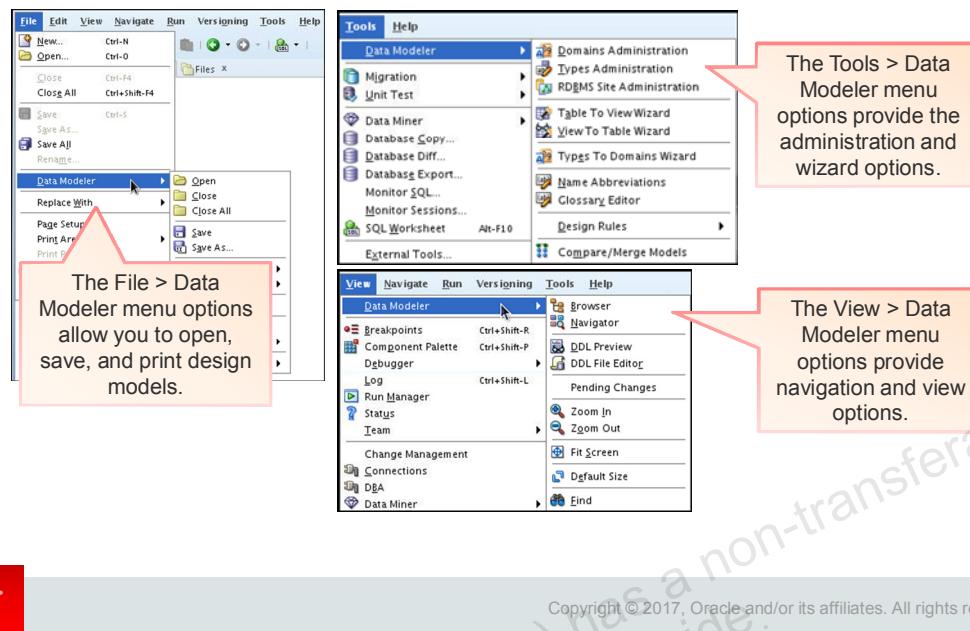


While working with SQL Developer, if the Connections Navigator disappears or if you cannot dock the Log window in its original place, perform the following steps to fix the problem:

1. Exit SQL Developer.
2. Open a terminal window and use the locate command to find the location of `windowinglayout.xml`.
3. Go to the directory that has `windowinglayout.xml` and delete it.
4. Restart SQL Developer.

Data Modeler in SQL Developer

SQL Developer includes an integrated version of SQL Developer Data Modeler.



Summary

In this appendix, you should have learned how to use SQL Developer to do:

- Browse, create, and edit database objects
- Execute SQL statements and scripts in SQL Worksheet
- Create and save custom reports
- Browse the Data Modeling options in SQL Developer



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Using SQL*Plus

Copyright ©2017, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Objectives

After completing this appendix, you should be able to do the following:

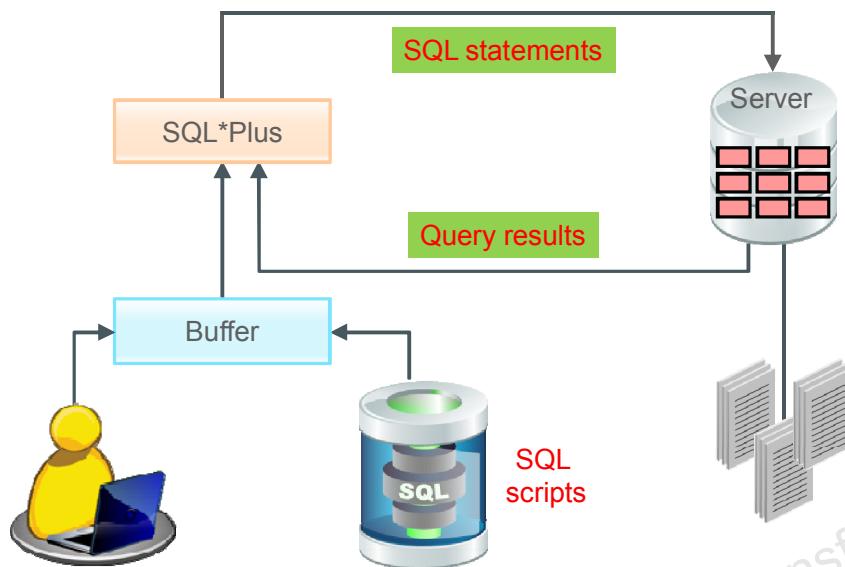
- Log in to SQL*Plus
- Edit SQL commands
- Format the output by using SQL*Plus commands
- Interact with script files



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL and SQL*Plus Interaction



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



SQL Statements Versus SQL*Plus Commands

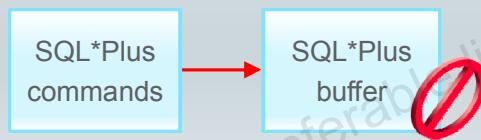
SQL

- A language
- ANSI-standard
- Keywords cannot be abbreviated.
- Statements manipulate data and table definitions in the database.



SQL*Plus

- An environment
- Oracle-proprietary
- Keywords can be abbreviated.
- Commands do not allow manipulation of values in the database.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The following table compares SQL and SQL*Plus:

SQL	SQL*Plus
Is a language for communicating with the Oracle server to access data	Recognizes SQL statements and sends them to the server
Is based on American National Standards Institute (ANSI)-standard SQL	Is the Oracle-proprietary interface for executing SQL statements
Manipulates data and table definitions in the database	Does not allow manipulation of values in the database
Is entered into the SQL buffer on one or more lines	Is entered one line at a time, not stored in the SQL buffer
Does not have a continuation character	Uses a dash (-) as a continuation character if the command is longer than one line
Cannot be abbreviated	Can be abbreviated
Uses a termination character to execute commands immediately	Does not require termination characters; executes commands immediately
Uses functions to perform some formatting	Uses commands to format data

SQL*Plus: Overview

- Log in to SQL*Plus.
- Describe the table structure.
- Edit your SQL statement.
- Execute SQL from SQL*Plus.
- Save SQL statements to files and append SQL statements to files.
- Execute saved files.
- Load commands from the file to buffer to edit.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL*Plus

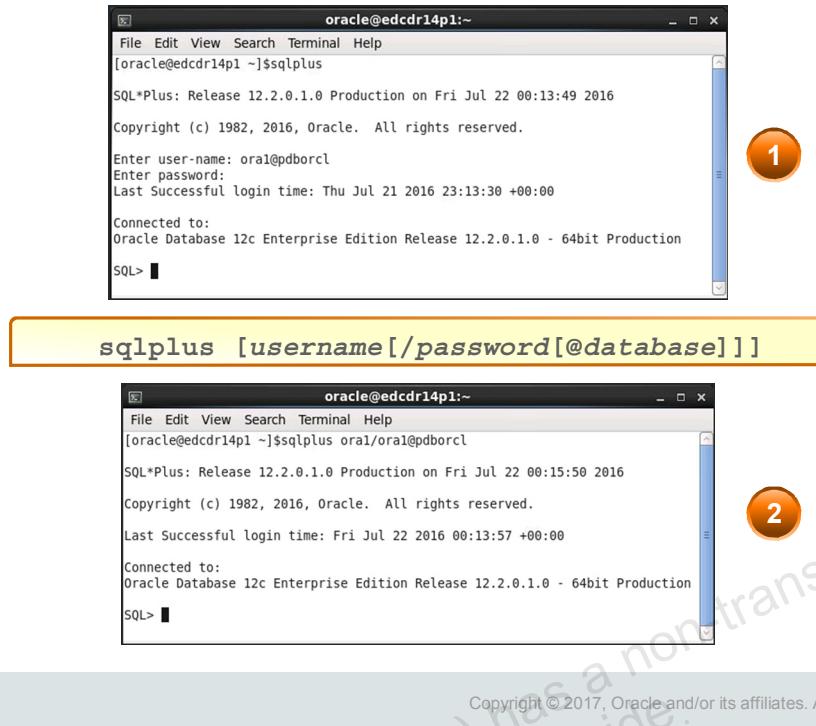
SQL*Plus is an environment in which you can:

- Execute SQL statements to retrieve, modify, add, and remove data from the database
- Format, perform calculations on, store, and print query results in the form of reports
- Create script files to store SQL statements for repeated use in the future

SQL*Plus commands can be divided into the following main categories:

Category	Purpose
Environment	Affect the general behavior of SQL statements for the session
Format	Format query results
File manipulation	Save, load, and run script files
Execution	Send SQL statements from the SQL buffer to the Oracle server
Edit	Modify SQL statements in the buffer
Interaction	Create and pass variables to SQL statements, print variable values, and print messages to the screen
Miscellaneous	Connect to the database, manipulate the SQL*Plus environment, and display column definitions

Logging In to SQL*Plus



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

How you invoke SQL*Plus depends on the type of operating system that you are running Oracle Database on.

To log in from a Linux environment, perform the following steps:

1. Right-click your Linux desktop and select terminal.
2. Enter the `sqlplus` command shown in the slide.
3. Enter the username, password, and database name.

In the syntax:

<code>username</code>	Your database username
<code>password</code>	Your database password (Your password is visible if you enter it here.)
<code>@database</code>	The database connect string

Note: To ensure the integrity of your password, do not enter it at the operating system prompt. Instead, enter only your username. Enter your password at the password prompt.

Displaying the Table Structure

Use the SQL*Plus DESCRIBE command to display the structure of a table:

```
DESC [RIBE] tablename
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In SQL*Plus, you can display the structure of a table by using the DESCRIBE command. The result of the command is a display of column names and data types, as well as an indication if a column must contain data.

In the syntax:

tablename The name of any existing table, view, or synonym that is accessible to the user

To describe the DEPARTMENTS table, use the following command:

```
SQL> DESCRIBE DEPARTMENTS
```

Name	Null	Type
DEPARTMENT_ID		NOT NULL NUMBER(4)
DEPARTMENT_NAME		NOT NULL VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Displaying the Table Structure

```
DESCRIBE departments
```

Name	Null	Type
<hr/>		
DEPARTMENT_ID	NOT NULL	NUMBER (4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2 (30)
MANAGER_ID		NUMBER (6)
LOCATION_ID		NUMBER (4)

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL*Plus Editing Commands

- A [PPEND] *text*
- C [HANGE] / *old* / *new*
- C [HANGE] / *text* /
- CL [EAR] BUFF [ER]
- DEL
- DEL *n*
- DEL *m n*



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL*Plus commands are entered one line at a time and are not stored in the SQL buffer.

Command	Description
A [PPEND] <i>text</i>	Adds <i>text</i> to the end of the current line
C [HANGE] / <i>old</i> / <i>new</i>	Changes <i>old</i> text to <i>new</i> in the current line
C [HANGE] / <i>text</i> /	Deletes <i>text</i> from the current line
CL [EAR] BUFF [ER]	Deletes all lines from the SQL buffer
DEL	Deletes current line
DEL <i>n</i>	Deletes line <i>n</i>
DEL <i>m n</i>	Deletes lines <i>m</i> to <i>n</i> inclusive

Guidelines

- If you press Enter before completing a command, SQL*Plus prompts you with a line number.
- You terminate the SQL buffer either by entering one of the terminator characters (semicolon or slash) or by pressing Enter twice. The SQL prompt appears.

SQL*Plus Editing Commands

- I [NPUT]
- I [NPUT] *text*
- L[IST]
- L[IST] *n*
- L[IST] *m n*
- R[UN]
- *n*
- *n text*
- 0 *text*



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Command	Description
I [NPUT]	Inserts an indefinite number of lines
I [NPUT] <i>text</i>	Inserts a line consisting of <i>text</i>
L[IST]	Lists all lines in the SQL buffer
L[IST] <i>n</i>	Lists one line (specified by <i>n</i>)
L[IST] <i>m n</i>	Lists a range of lines (<i>m</i> to <i>n</i>) inclusive
R[UN]	Displays and runs the current SQL statement in the buffer
<i>n</i>	Specifies the line to make the current line
<i>n text</i>	Replaces line <i>n</i> with <i>text</i>
0 <i>text</i>	Inserts a line before line 1

Note: You can enter only one SQL*Plus command for each SQL prompt. SQL*Plus commands are not stored in the buffer. To continue a SQL*Plus command on the next line, end the first line with a hyphen (-).

Using LIST, n, and APPEND

```
LIST
 1  SELECT last_name
 2* FROM employees
```

```
1
1* SELECT last_name
```

```
A , job_id
1* SELECT last_name, job_id
```

```
LIST
 1  SELECT last_name, job_id
 2* FROM employees
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- Use the L [IST] command to display the contents of the SQL buffer. The asterisk (*) beside line 2 in the buffer indicates that line 2 is the current line. Any edits that you made apply to the current line.
- Change the number of the current line by entering the number (n) of the line that you want to edit. The new current line is displayed.
- Use the A [PPEND] command to add text to the current line. The newly edited line is displayed. Verify the new contents of the buffer by using the LIST command.

Note: Many SQL*Plus commands, including LIST and APPEND, can be abbreviated to just their first letter. LIST can be abbreviated to L; APPEND can be abbreviated to A.

Using the CHANGE Command

```
LIST  
1* SELECT * from employees
```

```
c/employees/departments  
1* SELECT * from departments
```

```
LIST  
1* SELECT * from departments
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

- Use L [IST] to display the contents of the buffer.
- Use the C [HANGE] command to alter the contents of the current line in the SQL buffer. In this case, replace the employees table with the departments table. The new current line is displayed.
- Use the L [IST] command to verify the new contents of the buffer.

SQL*Plus File Commands

- `SAVE filename`
- `GET filename`
- `START filename`
- `@ filename`
- `EDIT filename`
- `SPOOL filename`
- `EXIT`



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SQL statements communicate with the Oracle server. SQL*Plus commands control the environment, format query results, and manage files. You can use the commands described in the following table:

Command	Description
<code>SAV[E] filename [.ext]</code> <code>[REP[LACE] APP[END]]</code>	Saves the current contents of the SQL buffer to a file. Use APPEND to add to an existing file; use REPLACE to overwrite an existing file. The default extension is .sql.
<code>GET filename [.ext]</code>	Writes the contents of a previously saved file to the SQL buffer. The default extension for the file name is .sql.
<code>STA[RT] filename [.ext]</code>	Runs a previously saved command file
<code>@ filename</code>	Runs a previously saved command file (same as START)
<code>ED[IT]</code>	Invokes the editor and saves the buffer contents to a file named afiedt.buf
<code>ED[IT] [filename [.ext]]</code>	Invokes the editor to edit the contents of a saved file
<code>SPO[OL] [filename [.ext]] OFF OUT</code>	Stores query results in a file. OFF closes the spool file. OUT closes the spool file and sends the file results to the printer.
<code>EXIT</code>	Quits SQL*Plus

Using the SAVE and START Commands

LIST

```
1  SELECT last_name, manager_id, department_id  
2* FROM employees
```

SAVE my_query

Created file my_query

START my_query

LAST_NAME	MANAGER_ID	DEPARTMENT_ID
King		90
Kochhar	100	90
...		
107 rows selected.		



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

SAVE

Use the **SAVE** command to store the current contents of the buffer in a file. Thus, you can store frequently used scripts for use in the future.

START

Use the **START** command to run a script in SQL*Plus. You can also, alternatively, use the symbol @ to run a script.

```
@my_query
```

SERVEROUTPUT Command

- Use the SET SERVEROUT [PUT] command to control whether to display the output of stored procedures or PL/SQL blocks in SQL*Plus.
- The DBMS_OUTPUT line length limit is increased from 255 bytes to 32767 bytes.
- The default size is now unlimited.
- Resources are not pre-allocated when SERVEROUTPUT is set.
- Because there is no performance penalty, use UNLIMITED unless you want to conserve physical memory.

```
SET SERVEROUT [PUT] {ON | OFF} [SIZE {n | UNL [IMITED]}]
[FOR [MAT] {WRA [PPED] | WOR [D_WRAPPED] | TRU [NCATED]}]
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Most of the PL/SQL programs perform input and output through SQL statements, to store data in database tables or query those tables. All other PL/SQL input/output is done through APIs that interact with other programs. For example, the DBMS_OUTPUT package has procedures, such as PUT_LINE. To see the result outside of PL/SQL requires another program, such as SQL*Plus, to read and display the data passed to DBMS_OUTPUT.

SQL*Plus does not display DBMS_OUTPUT data unless you first issue the SQL*Plus command SET SERVEROUTPUT ON as follows:

```
SET SERVEROUTPUT ON
```

Note

- SIZE sets the number of bytes of the output that can be buffered within the Oracle Database server. The default is UNLIMITED. n cannot be less than 2000 or greater than 1,000,000.
- For additional information about SERVEROUTPUT, see *Oracle Database PL/SQL User's Guide and Reference 12c*.

Using the SQL*Plus SPOOL Command

```
SPO[OL] [file_name[.ext]] [CRE[ATE] | REP[LACE] |  
APP[END]] | OFF | OUT]
```

Option	Description
file_name [. ext]	Spools output to the specified file name
CRE [ATE]	Creates a new file with the name specified
REP [LACE]	Replaces the contents of an existing file. If the file does not exist, REPLACE creates the file.
APP [END]	Adds the contents of the buffer to the end of the file that you specify
OFF	Stops spooling
OUT	Stops spooling and sends the file to your computer's standard (default) printer



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The SPOOL command stores query results in a file or optionally sends the file to a printer. The SPOOL command has been enhanced. You can now append to, or replace an existing file, where previously you could use SPOOL only to create (and replace) a file. REPLACE is the default.

To spool the output generated by commands in a script without displaying the output on screen, use SET TERMOUT OFF. SET TERMOUT OFF does not affect the output from commands that run interactively.

You must use quotation marks around file names that contain white space. To create a valid HTML file using SPOOL APPEND commands, you must use PROMPT or a similar command to create the HTML page header and footer. The SPOOL APPEND command does not parse HTML tags. SET SQLPLUSCOMPAT [IBILITY] to 9.2 or earlier to disable the CREATE, APPEND, and SAVE parameters.

Using the AUTOTRACE Command

- It displays a report after the successful execution of SQL DML statements such as SELECT, INSERT, UPDATE, or DELETE.
- The report can now include execution statistics and the query execution path.

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]  
[STATISTICS]
```

```
SET AUTOTRACE ON  
-- The AUTOTRACE report includes both the optimizer  
-- execution path and the SQL statement execution  
-- statistics
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

EXPLAIN shows the query execution path by performing an EXPLAIN PLAN. STATISTICS displays SQL statement statistics. The formatting of your AUTOTRACE report may vary depending on the version of the server to which you are connected and the configuration of the server. The DBMS_XPLAN package provides an easy way to display the output of the EXPLAIN PLAN command in several predefined formats.

Note

- For additional information about the package and subprograms, refer to *Oracle Database PL/SQL Packages and Types Reference 12c*.
- For additional information about the EXPLAIN PLAN, refer to *Oracle Database SQL Reference 12c*.
- For additional information about Execution Plans and the statistics, refer to *Oracle Database Performance Tuning Guide 12c*.

Summary

In this appendix, you should have learned how to use SQL*Plus as an environment to do the following:

- Execute SQL statements
- Edit SQL statements
- Format the output
- Interact with script files



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Copyright©2017, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- Execute a basic SELECT statement
- Create, alter, and drop a table using data definition language (DDL) statements
- Insert, update, and delete rows from one or more tables using data manipulation language (DML) statements
- Commit, roll back, and create savepoints by using transaction control statements
- Perform join operations on one or more tables



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Basic SELECT Statement

- Use the SELECT statement to:
 - Identify the columns to be displayed
 - Retrieve data from one or more tables, object tables, views, object views, or materialized views
- A SELECT statement is also known as a query because it queries a database.
- Syntax:

```
SELECT { * | [DISTINCT] column|expression [alias],... }  
FROM    table;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In its simplest form, a SELECT statement must include the following:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

In the syntax:

SELECT	Is a list of one or more columns
*	Selects all columns
DISTINCT	Suppresses duplicates
column / expression	Selects the named column or the expression
alias	Gives different headings to the selected columns
FROM table	Specifies the table containing the columns

Note: Throughout this course, the words *keyword*, *clause*, and *statement* are used as follows:

- A *keyword* refers to an individual SQL element—for example, SELECT and FROM are keywords.
- A *clause* is a part of a SQL statement (for example, SELECT employee_id, last_name).
- A *statement* is a combination of two or more clauses (for example, SELECT * FROM employees).

SELECT Statement

- Select all columns:

```
SELECT *  
FROM job_history;
```

#	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-01	24-JUL-06	IT_PROG	60
2	101	21-SEP-97	27-OCT-01	AC_ACCOUNT	110
3	101	28-OCT-01	15-MAR-05	AC_MGR	110
4	201	17-FEB-04	19-DEC-07	MK_REP	20
5	114	24-MAR-06	31-DEC-07	ST_CLERK	50
6	122	01-JAN-07	31-DEC-07	ST_CLERK	50
7	200	17-SEP-95	17-JUN-01	AD_ASST	90
8	176	24-MAR-06	31-DEC-06	SA_REP	80
9	176	01-JAN-07	31-DEC-07	SA_MAN	80
10	200	01-JUL-02	31-DEC-06	AC_ACCOUNT	90

- Select specific columns:

```
SELECT manager_id, job_id  
FROM employees;
```

#	MANAGER_ID	JOB_ID
1	(null)	AD_PRES
2	100	AD_VP
3	100	AD_VP
4	102	IT_PROG
5	103	IT_PROG
6	103	IT_PROG
7	100	ST_MAN
8	124	ST_CLERK
9	124	ST_CLERK
10	124	ST_CLERK



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can display all columns of data in a table by following the `SELECT` keyword with an asterisk (*) or by listing all the column names after the `SELECT` keyword. The first example in the slide displays all the rows from the `job_history` table. Specific columns of the table can be displayed by specifying the column names, separated by commas. The second example in the slide displays the `manager_id` and `job_id` columns from the `employees` table.

In the `SELECT` clause, specify the columns in the order in which you want them to appear in the output. For example, the following SQL statement displays the `location_id` column before displaying the `department_id` column:

```
SELECT location_id, department_id FROM departments;
```

Note: You can enter your SQL statement in a SQL Worksheet and click the Run Statement icon or press F9 to execute a statement in SQL Developer. The output displayed on the Results tabbed page appears as shown in the slide.

WHERE Clause

- Use the optional WHERE clause to:
 - Filter rows in a query
 - Produce a subset of rows
- Syntax:

```
SELECT * FROM table  
[WHERE condition];
```

- Example:

```
SELECT location_id from departments  
WHERE department_name = 'Marketing';
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

ORDER BY Clause

- Use the optional ORDER BY clause to specify the row order.
- Syntax:

```
SELECT * FROM table  
[WHERE condition]  
[ORDER BY {<column>|<position>} [ASC|DESC] [, ...] ];
```

- Example:

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id ASC, salary DESC;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

GROUP BY Clause

- Use the optional GROUP BY clause to group columns that have matching values into subsets.
- Each group has no two rows having the same value for the grouping column or columns.
- Syntax:

```
SELECT <column1, column2, ... column_n>
  FROM table
  [WHERE condition]
  [GROUP BY <column> [, ...] ]
  [ORDER BY <column> [, ...] ] ;
```

- Example:

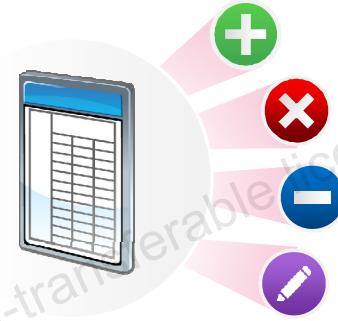
```
SELECT department_id, MIN(salary), MAX (salary)
  FROM employees
 GROUP BY department_id ;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data Definition Language

- DDL statements are used to define, structurally change, and drop schema objects.
- The commonly used DDL statements are:
 - CREATE TABLE, ALTER TABLE, and DROP TABLE
 - GRANT, REVOKE
 - TRUNCATE



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

DDL statements enable you to alter the attributes of an object without altering the applications that access the object. You can also use DDL statements to alter the structure of objects while database users are performing work in the database. These statements are most frequently used to:

- Create, alter, and drop schema objects and other database structures, including the database itself and database users
- Delete all the data in schema objects without removing the structure of these objects
- Grant and revoke privileges and roles

Oracle Database implicitly commits the current transaction before and after every DDL statement.

CREATE TABLE Statement

- Use the CREATE TABLE statement to create a table in the database.
- Syntax:

```
CREATE TABLE tablename (
{column-definition | Table-level constraint}
[ , {column-definition | Table-level constraint} ] * )
```

- Example:

```
CREATE TABLE teach_dept (
department_id NUMBER(3) PRIMARY KEY,
department_name VARCHAR2(10));
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Use the CREATE TABLE statement to create a table in the database. To create a table, you must have the CREATE TABLE privilege and a storage area in which to create objects.

The table owner and the database owner automatically gain the following privileges on the table after it is created:

- INSERT
- SELECT
- REFERENCES
- ALTER
- UPDATE

The table owner and the database owner can grant the preceding privileges to other users.

ALTER TABLE Statement

- Use the ALTER TABLE statement to modify the definition of an existing table in the database.
- Example1:

```
ALTER TABLE teach_dept  
ADD location_id NUMBER NOT NULL;
```

- Example 2:

```
ALTER TABLE teach_dept  
MODIFY department_name VARCHAR2(30) NOT NULL;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The ALTER TABLE statement allows you to make changes to an existing table.

You can:

- Add a column to a table
- Add a constraint to a table
- Modify an existing column definition
- Drop a column from a table
- Drop an existing constraint from a table
- Increase the width of the VARCHAR and CHAR columns
- Change a table to have read-only status

Example 1 in the slide adds a new column called location_id to the teach_dept table.

Example 2 updates the existing department_name column from VARCHAR2 (10) to VARCHAR2 (30), and adds a NOT NULL constraint to it.

DROP TABLE Statement

- The DROP TABLE statement removes the table and all its data from the database.
- Example:

```
DROP TABLE teach_dept;
```

- DROP TABLE with the PURGE clause drops the table and releases the space that is associated with it.

```
DROP TABLE teach_dept PURGE;
```

- The CASCADE CONSTRAINTS clause drops all referential integrity constraints from the table.

```
DROP TABLE teach_dept CASCADE CONSTRAINTS;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The DROP TABLE statement allows you to remove a table and its contents from the database, and pushes it to the recycle bin. Dropping a table invalidates dependent objects and removes object privileges on the table.

Use the PURGE clause along with the DROP TABLE statement to release back to the tablespace the space allocated for the table. You cannot roll back a DROP TABLE statement with the PURGE clause, nor can you recover the table if you have dropped it with the PURGE clause.

The CASCADE CONSTRAINTS clause allows you to drop the reference to the primary key and unique keys in the dropped table.

GRANT Statement

- The GRANT statement assigns the privilege to perform the following operations:
 - Insert or delete data.
 - Create a foreign key reference to the named table or to a subset of columns from a table.
 - Select data, a view, or a subset of columns from a table.
 - Create a trigger on a table.
 - Execute a specified function or procedure.
- Example:

```
GRANT SELECT any table to PUBLIC;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can use the GRANT statement to:

- Assign privileges to a specific user or role, or to all users, to perform actions on database objects
- Grant a role to a user, to PUBLIC, or to another role

Before you issue a GRANT statement, check that the `derby.database.sql` authorization property is set to True. This property enables SQL Authorization mode. You can grant privileges on an object if you are the owner of the database.

You can grant privileges to all users by using the PUBLIC keyword. When PUBLIC is specified, the privileges or roles affect all current and future users.

Privilege Types

Assign the following privileges by using the GRANT statement:

- ALL PRIVILEGES
- DELETE
- INSERT
- REFERENCES
- SELECT
- UPDATE



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

REVOKE Statement

- Use the REVOKE statement to remove privileges from a user to perform actions on database objects.
- Revoke a *system privilege* from a user:

```
REVOKE DROP ANY TABLE  
  FROM hr;
```

- Revoke a *role* from a user:

```
REVOKE dw_manager  
  FROM sh;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The REVOKE statement removes privileges from a specific user (or users) or role to perform actions on database objects. It performs the following operations:

- Revokes a role from a user, from PUBLIC, or from another role
- Revokes privileges for an object if you are the owner of the object or the database owner

Note: To revoke a role or system privilege, you must have been granted the privilege with the ADMIN OPTION.

TRUNCATE TABLE Statement

- Use the TRUNCATE TABLE statement to remove all the rows from a table.
- Example:

```
TRUNCATE TABLE employees_demo;
```

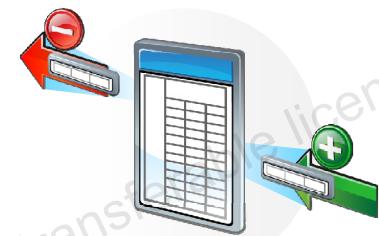
- By default, Oracle Database performs the following tasks:
 - De-allocates space used by the removed rows
 - Sets the NEXT storage parameter to the size of the last extent removed from the segment by the truncation process



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data Manipulation Language

- DML statements query or manipulate data in the existing schema objects.
- A DML statement is executed when:
 - New rows are added to a table by using the `INSERT` statement
 - Existing rows in a table are modified using the `UPDATE` statement
 - Existing rows are deleted from a table by using the `DELETE` statement
- A *transaction* consists of a collection of DML statements that form a logical unit of work.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Data Manipulation Language (DML) statements enable you to query or change the contents of an existing schema object. These statements are most frequently used to:

- Add new rows of data to a table or view by specifying a list of column values or using a subquery to select and manipulate existing data
- Change column values in the existing rows of a table or view
- Remove rows from tables or views

A collection of DML statements that forms a logical unit of work is called a transaction. Unlike DDL statements, DML statements do not implicitly commit the current transaction.

INSERT Statement

- Use the `INSERT` statement to add new rows to a table.
- Syntax:

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

- Example:

```
INSERT INTO departments  
VALUES (200, 'Development', 104, 1400);  
1 rows inserted.
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The `INSERT` statement adds rows to a table. Make sure to insert a new row containing values for each column and to list the values in the default order of the columns in the table. Optionally, you can also list the columns in the `INSERT` statement.

Example:

```
INSERT INTO job_history (employee_id, start_date, end_date,  
job_id)  
VALUES (120, '25-JUL-06', '12-FEB-08', 'AC_ACCOUNT');
```

The syntax discussed in the slide allows you to insert a single row at a time. The `VALUES` keyword assigns the values of expressions to the corresponding columns in the column list.

UPDATE Statement Syntax

- Use the UPDATE statement to modify the existing rows in a table.
- Update more than one row at a time (if required).

```
UPDATE    table  
SET        column = value [, column = value, ...]  
[WHERE    condition];
```

- Example:

```
UPDATE    copy_emp  
SET  
[22 rows updated]
```

- Specify SET *column_name*= NULL to update a column value to NULL.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The UPDATE statement modifies the existing values in a table. Confirm the update operation by querying the table to display the updated rows. You can modify a specific row or rows by specifying the WHERE clause.

Example:

```
UPDATE employees  
SET      salary = 17500  
WHERE   employee_id = 102;
```

In general, use the primary key column in the WHERE clause to identify the row to update. For example, to update a specific row in the employees table, use employee_id to identify the row instead of employee_name, because more than one employee may have the same name.

Note: Typically, the condition keyword is composed of column names, expressions, constants, subqueries, and comparison operators.

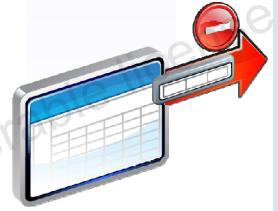
DELETE Statement

- Use the DELETE statement to delete the existing rows from a table.
- Syntax:

```
DELETE [FROM] table  
[WHERE condition];
```

- Write the DELETE statement using the WHERE clause to delete specific rows from a table.

```
DELETE FROM departments  
WHERE department_name = 'Finance';  
1 rows deleted
```



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The DELETE statement removes existing rows from a table. You must use the WHERE clause to delete a specific row or rows from a table based on the condition. The condition identifies the rows to be deleted. It may contain column names, expressions, constants, subqueries, and comparison operators.

The first example in the slide deletes the finance department from the departments table. You can confirm the delete operation by using the SELECT statement to query the table.

```
SELECT *  
FROM departments  
WHERE department_name = 'Finance';
```

If you omit the WHERE clause, all rows in the table are deleted. Example:

```
DELETE FROM copy_emp;
```

The preceding example deletes all the rows from the copy_emp table.

Transaction Control Statements

- Transaction control statements are used to manage the changes made by DML statements.
- The DML statements are grouped into transactions.
- Transaction control statements include:
 - COMMIT
 - ROLLBACK
 - SAVEPOINT



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

COMMIT Statement

- Use the COMMIT statement to:
 - Permanently save the changes made to the database during the current transaction
 - Erase all savepoints in the transaction
 - Release transaction locks
- Example:

```
INSERT INTO departments
VALUES      (201, 'Engineering', 106, 1400);
COMMIT;
1 rows inserted.
committed.
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The COMMIT statement ends the current transaction by making all the pending data changes permanent. It releases all row and table locks, and erases any savepoints that you may have marked since the last commit or rollback. The changes made using the COMMIT statement are visible to all users.

Oracle recommends that you explicitly end every transaction in your application programs with a COMMIT or ROLLBACK statement, including the last transaction, before disconnecting from Oracle Database. If you do not explicitly commit the transaction and the program terminates abnormally, the last uncommitted transaction is automatically rolled back.

Note: Oracle Database issues an implicit COMMIT before and after any data definition language (DDL) statement.

ROLLBACK Statement

- Use the ROLLBACK statement to undo changes made to the database during the current transaction.
- Use the TO SAVEPOINT clause to undo a part of the transaction after the savepoint.
- Example:

```
UPDATE      employees
SET         salary = 7000
WHERE        last_name = 'Ernst';
SAVEPOINT   Ernst_sal;

UPDATE      employees
SET         salary = 12000
WHERE        last_name = 'Mourgos';

ROLLBACK TO SAVEPOINT Ernst_sal;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The ROLLBACK statement undoes work done in the current transaction. To roll back the current transaction, no privileges are necessary.

Using ROLLBACK with the TO SAVEPOINT clause performs the following operations:

- Rolls back only the portion of the transaction after the savepoint
- Erases all savepoints created after that savepoint. The named savepoint is retained, so you can roll back to the same savepoint multiple times.

Using ROLLBACK without the TO SAVEPOINT clause performs the following operations:

- Ends the transaction
- Undoes all the changes in the current transaction
- Erases all savepoints in the transaction

SAVEPOINT Statement

- Use the SAVEPOINT statement to name and mark the current point in the processing of a transaction.
- Specify a name to each savepoint.
- Use distinct savepoint names within a transaction to avoid overriding.
- Syntax:

```
SAVEPOINT savepoint;
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Joins

Use a join to query data from more than one table:

```
SELECT table1.column, table2.column  
FROM   table1, table2  
WHERE  table1.column1 = table2.column2;
```

- Write the join condition in the WHERE clause.
- Prefix the column name with the table name when the same column name appears in more than one table.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When data from more than one table in the database is required, a *join* condition is used. Rows in one table can be joined to rows in another table according to common values that exist in the corresponding columns (usually primary and foreign key columns).

To display data from two or more related tables, write a simple join condition in the WHERE clause.

In the syntax:

table1.column

Denotes the table and column from which data is retrieved

table1.column1 =

Is the condition that joins (or relates) the tables together

table2.column2

Guidelines

- When writing a SELECT statement that joins tables, precede the column name with the table name for clarity and to enhance database access.
- If the same column name appears in more than one table, the column name must be prefixed with the table name.
- To join n tables together, you need a minimum of $n-1$ join conditions. For example, to join four tables, a minimum of three joins is required. This rule may not apply if your table has a concatenated primary key, in which case more than one column is required to uniquely identify each row.

Types of Joins

- Natural join
- Equijoin
- Nonequijoin
- Outer join
- Self-join
- Cross join



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Use table prefixes to improve performance.
- Use table aliases instead of full table name prefixes.
- Table aliases give a table a shorter name.
 - This keeps SQL code smaller and uses less memory.
- Use column aliases to distinguish columns that have identical names but reside in different tables.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When joining two or more tables, you need to qualify the names of the columns with the table name to avoid ambiguity. Without the table prefixes, the DEPARTMENT_ID column in the SELECT list could be from either the DEPARTMENTS table or the EMPLOYEES table. Therefore, it is necessary to add the table prefix to execute your query. If there are no common column names between the two tables, there is no need to qualify the columns. However, using a table prefix improves performance, because you tell the Oracle server exactly where to find the columns.

Qualifying column names with table names can be very time consuming, particularly if table names are lengthy. Therefore, you can use *table aliases*, instead of table names. Just as a column alias gives a column another name, a table alias gives a table another name. Table aliases help to keep SQL code smaller, thereby using less memory.

The table name is specified in full, followed by a space, and then the table alias. For example, the EMPLOYEES table can be given an alias of e, and the DEPARTMENTS table an alias of d.

Guidelines

- Table aliases can be up to 30 characters in length, but shorter aliases are better than longer ones.
- If a table alias is used for a particular table name in the FROM clause, that table alias must be substituted for the table name throughout the SELECT statement.
- Table aliases should be meaningful.
- A table alias is valid only for the current SELECT statement.

Natural Join

- The NATURAL JOIN clause is based on all the columns in the two tables that have the same name.
- It selects rows from tables that have the same names and data values of columns.
- Example:

```
SELECT country_id, location_id, country_name, city
FROM countries NATURAL JOIN locations;
```

COUNTRY_ID	LOCATION_ID	COUNTRY_NAME	CITY
1 US	1400	United States of America	Southlake
2 US	1500	United States of America	San Francisco
3 US	1700	United States of America	Seattle
4 CA	1800	Canada	Toronto
5 UK	2500	United Kingdom	Oxford

ORACLE

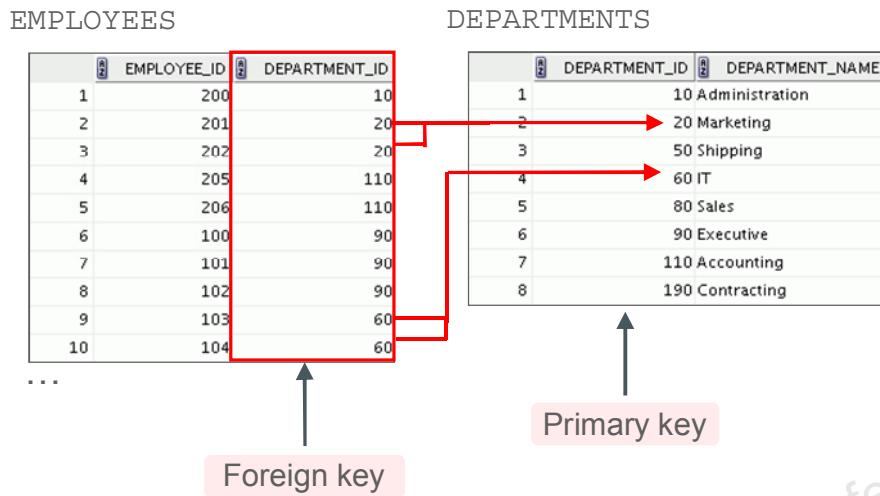
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You can join tables automatically based on the columns in the two tables that have matching data types and names. You do this by using the NATURAL JOIN keywords.

Note: The join can happen only on those columns that have the same names and data types in both tables. If the columns have the same name but different data types, the NATURAL JOIN syntax causes an error.

In the example in the slide, the COUNTRIES table is joined to the LOCATIONS table by the COUNTRY_ID column, which is the only column of the same name in both tables. If other common columns were present, the join would have used them all.

Equijoins



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

An **equijoin** is a join with a join condition containing an equality operator. An equijoin combines rows that have equivalent values for the specified columns. To determine an employee's department name, you compare the values in the DEPARTMENT_ID column in the EMPLOYEES table with the DEPARTMENT_ID values in the DEPARTMENTS table. The relationship between the EMPLOYEES and DEPARTMENTS tables is an **equijoin**; that is, values in the DEPARTMENT_ID column in both tables must be equal. Often, this type of join involves primary and foreign key complements.

Note: Equijoins are also called *simple joins*.

Retrieving Records with Equijoins

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e JOIN departments d
 WHERE e.department_id = d.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200 Whalen	10	10	1700
2	201 Hartstein	20	20	1800
3	202 Fay	20	20	1800
4	144 Vargas	50	50	1500
5	143 Matos	50	50	1500
6	142 Davies	50	50	1500
7	141 Raji	50	50	1500
8	124 Mourgos	50	50	1500
9	103 Hunold	60	60	1400
10	104 Ernst	60	60	1400
11	107 Lorentz	60	60	1400
...				



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide:

- **The SELECT clause specifies the column names to retrieve:**
 - Employee last name, employee ID, and department ID, which are columns in the EMPLOYEES table
 - Department ID and location ID, which are columns in the DEPARTMENTS table
- **The FROM clause specifies the two tables that the database must access:**
 - EMPLOYEES table
 - DEPARTMENTS table
- **The WHERE clause specifies how the tables are to be joined:**

e.department_id = d.department_id

Because the DEPARTMENT_ID column is common to both tables, it must be prefixed with the table alias to avoid ambiguity. Other columns that are not present in both the tables need not be qualified by a table alias, but it is recommended for better performance.

Note: When you use the Execute Statement icon to run the query, SQL Developer suffixes a “_1” to differentiate between the two DEPARTMENT_IDS.

Additional Search Conditions Using the AND and WHERE Operators

```
SELECT d.department_id, d.department_name, l.city
FROM departments d JOIN locations l
ON d.location_id = l.location_id
AND d.department_id IN (20, 50);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	CITY
1	20	Marketing	Toronto
2	50	Shipping	South San Francisco

```
SELECT d.department_id, d.department_name, l.city
FROM departments d JOIN locations l
ON d.location_id = l.location_id
WHERE d.department_id IN (20, 50);
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In addition to the join, you may have criteria for your WHERE clause to restrict the rows in consideration for one or more tables in the join. The example in the slide performs a join on the DEPARTMENTS and LOCATIONS tables and, in addition, displays only those departments with ID equal to 20 or 50. To add additional conditions to the ON clause, you can add AND clauses. Alternatively, you can use a WHERE clause to apply additional conditions.

Both queries produce the same output.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

Retrieving Records with Nonequijoins

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON    e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

#	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Fay	6000	C

...

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The example in the slide creates a nonequijoin to evaluate an employee's salary grade. The salary must be *between* any pair of the low and high salary ranges.

It is important to note that all employees appear exactly once when this query is executed. No employee is repeated in the list. There are two reasons for this:

- None of the rows in the job grade table contain grades that overlap. That is, the salary value for an employee can lie only between the low salary and high salary values of one of the rows in the salary grade table.
- All of the employees' salaries lie within the limits that are provided by the job grade table. That is, no employee earns less than the lowest value contained in the `LOWEST_SAL` column or more than the highest value contained in the `HIGHEST_SAL` column.

Note: Other conditions (such as `<=` and `>=`) can be used, but `BETWEEN` is the simplest. Remember to specify the low value first and the high value last when using the `BETWEEN` condition. The Oracle server translates the `BETWEEN` condition to a pair of `AND` conditions. Therefore, using `BETWEEN` has no performance benefits, but should be used only for logical simplicity.

Table aliases have been specified in the example in the slide for performance reasons, not because of possible ambiguity.

Retrieving Records by Using the USING Clause

- You can use the USING clause to match only one column when more than one column matches.
- You cannot specify this clause with a NATURAL join.
- Do not qualify the column name with a table name or table alias.
- Example:

```
SELECT country_id, country_name, location_id, city
FROM   countries JOIN locations
USING (country_id) ;
```

#	COUNTRY_ID	COUNTRY_NAME	LOCATION_ID	CITY
1	US	United States of America	1400	Southlake
2	US	United States of America	1500	South San Francisco
3	US	United States of America	1700	Seattle
4	CA	Canada	1800	Toronto
5	UK	United Kingdom	2500	Oxford

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the COUNTRY_ID columns in the COUNTRIES and LOCATIONS tables are joined and thus the LOCATION_ID of the location where an employee works is shown.

Retrieving Records by Using the ON Clause

- The join condition for the natural join is basically an equijoin of all columns with the same name.
- Use the `ON` clause to specify arbitrary conditions or specify columns to join.
- The `ON` clause makes code easy to understand.

```
SELECT e.employee_id, e.last_name, j.department_id,  
FROM   employees e JOIN job_history j  
ON     (e.employee_id = j.employee_id);
```

#	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	101	Kochhar	110
2	101	Kochhar	110
3	102	De Haan	60
4	176	Taylor	80
5	176	Taylor	80
6	200	Whalen	90
7	200	Whalen	90
8	201	Hartstein	20



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Use the `ON` clause to specify a join condition. With this, you can specify join conditions separate from any search or filter conditions in the `WHERE` clause.

In this example, the `EMPLOYEE_ID` columns in the `EMPLOYEES` and `JOB_HISTORY` tables are joined using the `ON` clause. Wherever an employee ID in the `EMPLOYEES` table equals an employee ID in the `JOB_HISTORY` table, the row is returned. The table alias is necessary to qualify the matching column names.

You can also use the `ON` clause to join columns that have different names. The parentheses around the joined columns, as in the example in the slide, `(e.employee_id = j.employee_id)`, is optional. So, even `ON e.employee_id = j.employee_id` will work.

Note: When you use the Execute Statement icon to run the query, SQL Developer suffixes a '`_1`' to differentiate between the two `employee_ids`.

Left Outer Join

- A join between two tables that returns all matched rows, as well as the unmatched rows from the left table is called a LEFT OUTER JOIN.
- Example:

```
SELECT c.country_id, c.country_name, l.location_id, l.city
FROM   countries c [LEFT OUTER JOIN] locations l
ON    (c.country_id = l.country_id) ;
```

COUNTRY_ID	COUNTRY_NAME	LOCATION_ID	CITY
1 CA	Canada	1800	Toronto
2 DE	Germany	(null)	(null)
3 UK	United Kingdom	2500	Oxford
4 US	United States of America	1400	Southlake
5 US	United States of America	1500	South San Francisco
6 US	United States of America	1700	Seattle

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Right Outer Join

- A join between two tables that returns all matched rows, as well as the unmatched rows from the right table is called a **RIGHT OUTER JOIN**.
- Example:

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1 Whalen	10	Administration
2 Hartstein	20	Marketing
3 Fay	20	Marketing
4 Davies	50	Shipping
...		
18 Higgins	110	Accounting
19 Gietz	110	Accounting
20 (null)	190	Contracting



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Full Outer Join

- A join between two tables that returns all matched rows, as well as the unmatched rows from both tables is called a FULL OUTER JOIN.
- Example:

```
SELECT e.last_name, d.department_id, d.manager_id,
       d.department_name
  FROM employees e FULL OUTER JOIN departments d
     ON (e.manager_id = d.manager_id) ;
```

	LAST_NAME	DEPARTMENT_ID	MANAGER_ID	DEPARTMENT_NAME
1	King	(null)	(null)	(null)
2	Kochhar	90	100	Executive
3	De Haan	90	100	Executive
4	Hunold	(null)	(null)	(null)
...				
19	Higgins	(null)	(null)	(null)
20	Gietz	110	205	Accounting
21	(null)	190	(null)	Contracting
22	(null)	10	200	Administration



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Self-Join: Example

```
SELECT worker.last_name || ' works for '
    || manager.last_name
  FROM employees worker JOIN employees manager
 WHERE worker.manager_id = manager.employee_id
  ORDER BY worker.last_name;
```

#	WORKER.LAST_NAME "WORKSFOR' MANAGER.LAST_NAME
1	Abel works for Zlotkey
2	Davies works for Mourgos
3	De Haan works for King
4	Ernst works for Hunold
5	Fay works for Hartstein
6	Gietz works for Higgins
7	Grant works for Zlotkey
8	Hartstein works for King
9	Higgins works for Kochhar

...



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Cross Join

- A CROSS JOIN is a JOIN operation that produces the Cartesian product of two tables.
- Example:

```
SELECT department_name, city  
FROM department CROSS JOIN location;
```

#	DEPARTMENT_NAME	CITY
1	Administration	Oxford
2	Administration	Seattle
3	Administration	South San Francisco
4	Administration	Southlake
5	Administration	Toronto
6	Marketing	Oxford
7	Marketing	Seattle
8	Marketing	South San Francisco
9	Marketing	Southlake
10	Marketing	Toronto

...

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The CROSS JOIN syntax specifies the cross product. It is also known as a Cartesian product. A cross join produces the cross product of two relations, and is essentially the same as the comma-delimited Oracle Database notation.

You do not specify any WHERE condition between the two tables in the CROSS JOIN.

Summary

In this appendix, you should have learned how to use:

- The SELECT statement to retrieve rows from one or more tables
- DDL statements to alter the structure of objects
- DML statements to manipulate data in the existing schema objects
- Transaction control statements to manage the changes made by DML statements
- Joins to display data from multiple tables



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.