DATABASE SYSTEMS LAB

Course Code	: 30102422
Credit Hours	: 1
Prerequisite	: 30102421

Instructor Information

Name	:Dr. Jihad Nader				
Office No.	**B17 F4 R8**				
Tel (Ext)	**N/A**				
E-mail	jihadnader@bau.edu.jo				
Office Hours	Sun, Thurs ,Tues (9-10) mon, wed (10-11)				
Class Times	Building	Day	Start Time	End Time	Room No.
	-	Monday	14	17	مختبر المعالجات
	-	Wednesday	14	17	مختبر المعالجات

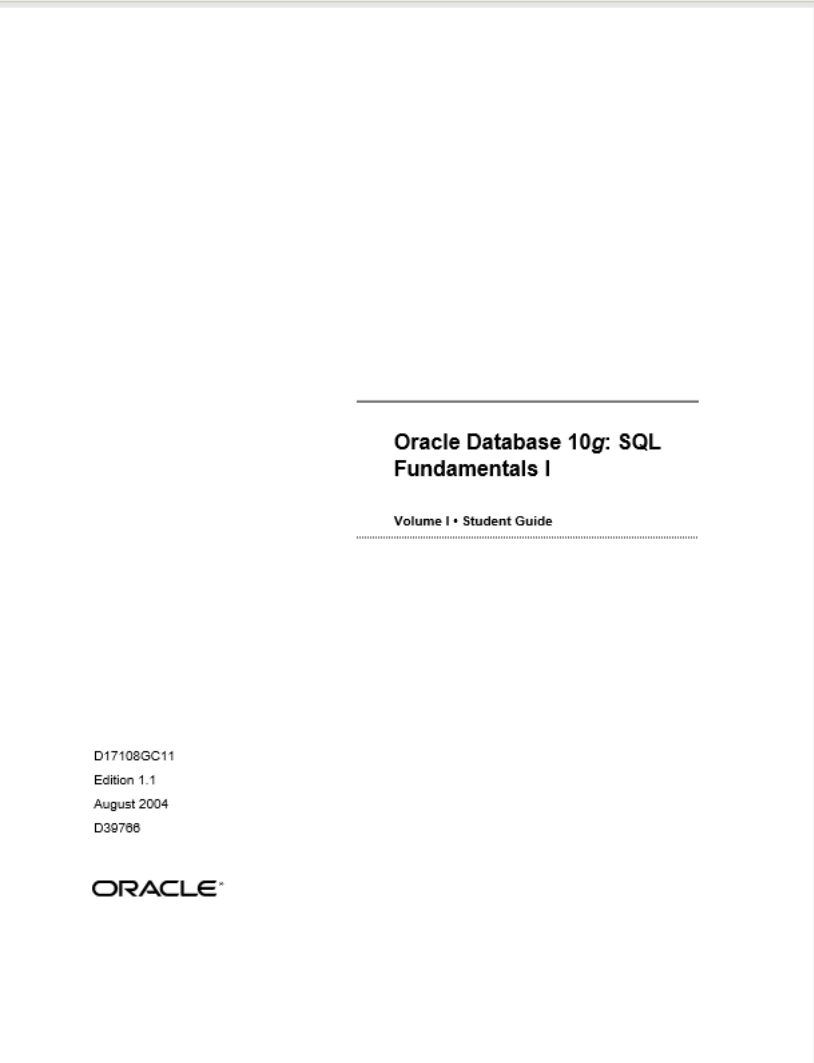
Course Description:

This Lab. practices the concepts introduced in the Database systems course using Oracle Database. The students are expected to implement a database project for some problem.

Course Title: Database Systems Lab **Credit Hour(1)**

[Pre-req. Course Code(30102421)]

Textbook: *Oracle Database 10g: SQL Fundamentals I, Volume I • Student Guide*



Oracle Database 10*g*: SQL Fundamentals I

Volume I • Student Guide

D17108GC11
Edition 1.1
August 2004
D39766

ORACLE

COURSE OBJECTIVES:

Upon completion of this course, students will have gained knowledge of the DBMS (Oracle) concepts and the ability to:

- Understand the concepts of relational databases and the Oracle Database 10g database technology.
- Use the powerful SQL programming language and its features.
- Identify features of Relational Database Management System (RDBMS).
- Categorize the main database objects
- Understand how constraints are created at the time of table creation
- Describe each data manipulation language (DML) statement
- List the capabilities of SQL SELECT statements
- Write SELECT statements to access data from more than one table using equijoins and non-equijoins
- Employ SQL functions to generate and retrieve customized data
- Identify when a subquery can help solve a question
- Write subqueries when a query is based on unknown values
- Use a set operator to combine multiple queries into a single query

COURSE SYLLABUS

Week	Course Topic	Notes
Week 1	Creating and Managing Tables: - Database Objects - Naming Conventions - The Create Table Statement - Creating a Table by Using a Subquery - Querying the Data Dictionary - The Alter Table Statement - Truncating a Table - Adding Comments to a Table	

Week 2	Including Constraints
	- Defining Constraints
	  o The Not Null Constraint
	  o The Unique Constraint
	  o The Primary Key Constraint
	  o The Foreign Key Constraint
	  o The Check Constraint
	- Adding a Constraint
	- Dropping a Constraint
	- Enabling and Disabling Constraints
	- Viewing Constraints	

Week 3	Manipulating Data - Data Manipulating Language. - The Insert Statement - Copying Rows from another Table - The Update Statement - The Delete Statement - Database Transactions - Commit and Rollback Statements	
Week 4	Writing Basic SQL Statements - Selecting Specific Columns - Arithmetic Expressions - Concatenation Operator - Using Column Aliases - Eliminating Duplicate Rows	

COURSE SYLLABUS

Week	Course Topic	Notes
Week 5	Restricting and Sorting Data - Where Clause - Comparison Operators - Special Operators - Logical Operator (And, Or, Not) - Order By Clause	
Week 6	Displaying Data from Multiple Tables - Cartesian Product. - Types of Joins - Table Aliases.	
Week 7	Single-Row Functions - Character Functions. - Number Functions - Date Functions ▪	
Week 8	Midterm Exam	Midterm Exam
Week 9	Project Proposal	
Week 10	Single-Row Functions - Conversion Functions - General Functions	

COURSE SYLLABUS

Week	Course Topic	Notes
Week 11	Aggregating Data using Group Functions - Types of Group Functions (AVG, SUM, MAX, MIN, COUNT). - Creating Groups of data: Group By Clause. - Excluding Group Results: Having Clause. - Nested Group Functions	
Week 12	Subqueries - Types of Subqueries ▪ Single-Row Subqueries ▪ Multiple-Row Subqueries	
Week 13	Multiple-Column Subqueries - Column Comparisons - Null Values in a subquery - Using a subquery in the From Clause	
Week 14	Using the Set Operators - Union / Union All - Intersect - Minus	
Week 15	Project Discussion	
Week 16	Final Exam	Final Exam

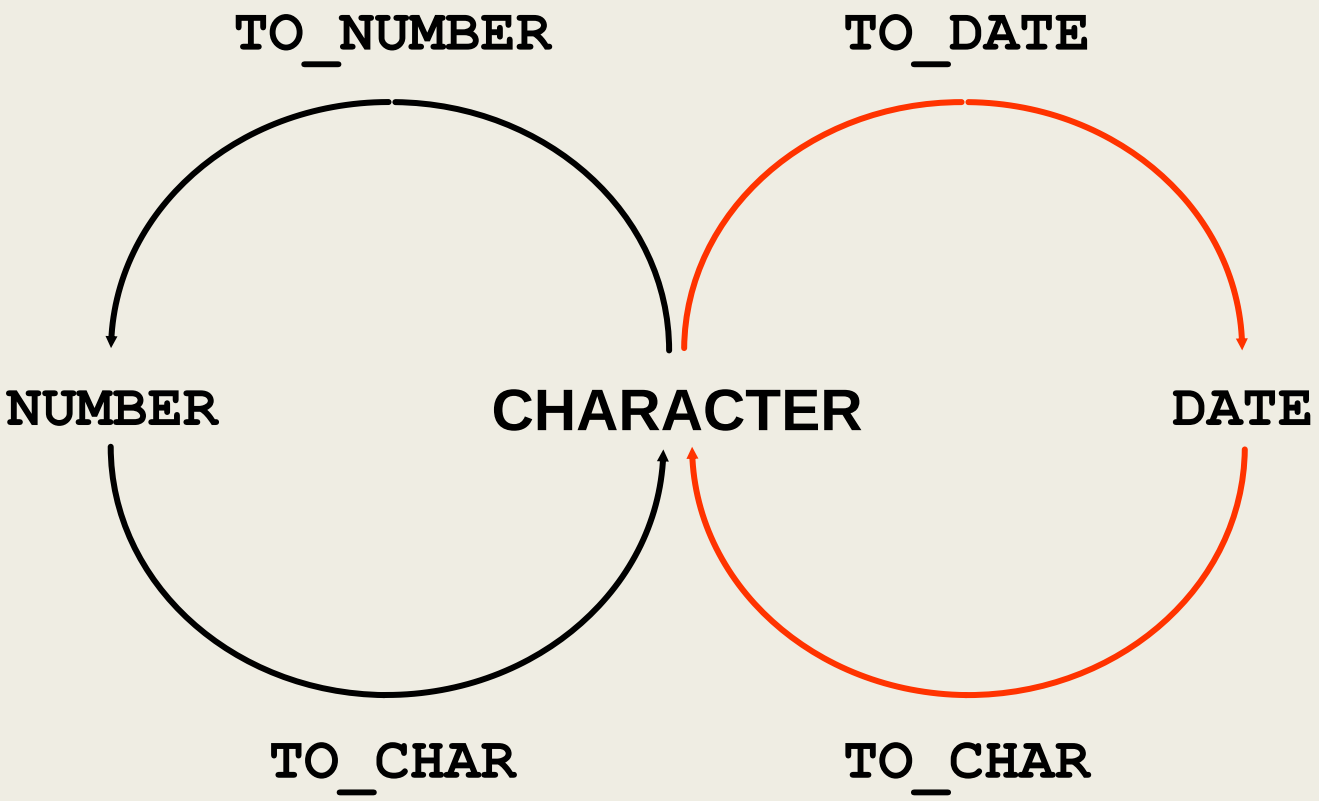
Week 9

Chapter 6:

Single Row Functions

USING SINGLE-ROW FUNCTIONS TO CUSTOMIZE OUTPUT

Explicit Data Type Conversion



Using the `TO_CHAR` Function with Dates

```
TO_CHAR(date, 'format_model')
```

The format model:

- *Must be enclosed by single quotation marks*
- *Is case sensitive*
- *Can include any valid date format element*
- *Has an `fm` element to remove padded blanks or suppress leading zeros*
- *Is separated from the date value by a comma*

Elements of the Date Format Model

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Elements of the Date Format Model

- *Time elements format the time portion of the date:*

HH24:MI:SS AM	**15:45:32 PM**

- *Add character strings by enclosing them in double quotation marks:*

DD "of" MONTH	**12 of OCTOBER**

- *Number suffixes spell out numbers:*

ddspth	**fourteenth**

Using the TO_CHAR Function with Dates

```
SELECT ename,
       TO_CHAR(hiredate, 'DD /  Month / YYYY')
       AS HIREDATE
FROM   emp;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

...

20 rows selected.

Using the TO_CHAR Function with Numbers

```
TO_CHAR(number, 'format_model')
```

These are some of the format elements that you can use with the TO_CHAR function to display a number value as a character:

Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as thousands indicator

Using the TO_CHAR Function with Numbers

```
SELECT ename "foamily name",TO_CHAR(sal, '$99,999.00') SALARY
FROM   emp
WHERE  ename = 'Scott';
```

Family Name	SALARY
Scott	$3,500.00

Using the `TO_NUMBER` and `TO_DATE` Functions

- *Convert a character string to a number format using the `TO_NUMBER` function:*

```
TO_NUMBER(char[, 'format_model'])
```

- *Convert a character string to a date format using the `TO_DATE` function:*

```
TO_DATE(char[, 'format_model'])
```

Examples

1. The following statement uses implicit conversion to combine a string and a number into a number:
   - *SELECT TO_CHAR('01110' + 1) FROM dual;*
   - *1111*
2. In the next example, the output is blank padded to the left of the currency symbol.

- SELECT TO_CHAR(-10000,'L99G999D99MI') "Amount"

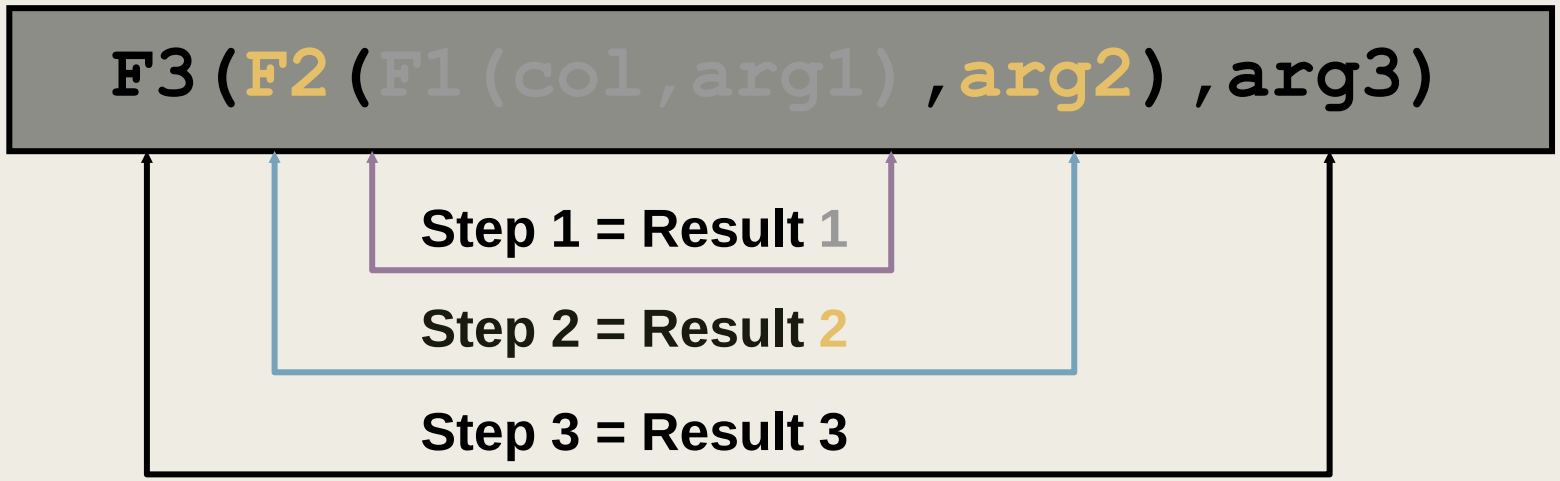
  FROM DUAL;
- Amount
- ----------
- $10,000.00-

Examples

- SELECT TO_CHAR(12345, '00000000') FROM DUAL;-→ 00012345
- SELECT TO_CHAR(12345.67, '99999.9') FROM DUAL;→ 12345.7
- SELECT TO_CHAR(SYSDATE, 'YYYY_MM_DD') FROM DUAL;→ 2014_12_27
- SELECT TO_CHAR(SYSDATE, 'Month') FROM DUAL;→ December

Nesting Functions

- *Single-row functions can be nested to any level.*
- *Nested functions are evaluated from the deepest level to the least deep level.*



Nesting Functions

```
SELECT last name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8
Hunold	HUNOLD_US
Ernst	ERNST_US
Lorentz	LORENTZ_US

General Functions

The following functions work with any data type and pertain to using nulls:

- *NVL (expr1, expr2)*
- *NVL2 (expr1, expr2, expr3)*

NVL Function

Converts a null value to an actual value:

- *Data types that can be used are date, character, and number.*
- *Data types must match:*
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

```
SELECT   TO_CHAR(NEXT_DAY(ADD_MONTHS
         (hiredate, 6), 'FRIDAY'),
         'fmDay, Month DDth, YYYY')
         "Next 6 Month Review"
  FROM      emp
  ORDER BY  hiredate;
```

Next 6 Month Review
Friday, January 3RD, 2003
Friday, February 7TH, 2003
Friday, June 20TH, 2003
Friday, July 25TH, 2003
Friday, August 22ND, 2003
Friday, August 29TH, 2003
Friday, October 3RD, 2003
Friday, November 7TH, 2003

Using the NVL Function

```
SELECT ename, sal, NVL(comm, 0),
   (sal*12) + (sal*12*NVL(comm, 0)) AN_SAL
FROM emp;
```

1

2

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...

20 rows selected.

1

2

Using the NVL2 Function

```
SELECT last name,  salary, commission pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	.2	SAL+COMM
Abel	11000	.3	SAL+COMM
Taylor	8600	.2	SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.

Conditional Expressions

- *Provide the use of IF-THEN-ELSE logic within a SQL statement*
- *Use two methods:*
  - `CASE` expression
  - `DECODE` function

CASE Expression

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
          [WHEN comparison_expr2 THEN return_expr2
           WHEN comparison_exprn THEN return_exprn
           ELSE else_expr]
END
```

Using the CASE Expression

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END     "REVISED_SALARY"
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

DECODE Function

Facilitates conditional inquiries by doing the work of a CASE expression or an IF-THEN-ELSE statement:

```
DECODE(col/expression, search1, result1
                  [, search2, result2,...,]
                  [, default])
```

Using the DECODE Function

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

Using the DECODE Function

Display the applicable tax rate for each employee in department 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Summary

In this lesson, you should have learned how to:

- *Perform calculations on data using functions*
- *Modify individual data items using functions*
- *Manipulate output for groups of rows using functions*
- *Alter date formats for display using functions*
- *Convert column data types using functions*
- *Use* `NVL` *functions*
- *Use IF-THEN-ELSE logic*