



Lab Manual

ISC 321 Database System I

Prepared by:

Dr. Hanady Abdulsalam

Eng. Eman Al Roumi

Contents

Laboratory Hardware, Software / Tools Requirements	3
Laboratory Schedule	5
Laboratory Policy	6
Laboratory Grading Policy	7
Introduction	8
Lab #1 – Introduction to Oracle DBMS	9
Lab #2 - Creating a Database.....	12
Lab #3 – Simple Query Statements	23
Lab #4 – Using Simple Functions In Query Statements	27
Lab #5 – Grouping Rows and Subqueries	32
Lab #6 – Creating Sequences and Views - TOAD	37
Lab #7 – Report Builder	49
Lab #8 – Create Simple Oracle Form	61
Lab #9 – Adding Interactive Items to Oracle Form	72
Lab #10 – Adding Non-Input Items to Oracle Forms.....	90
Lab #11 – Introduction to PL/SQL Programming	97
Lab #12 – Applying PL/SQL On Database	103
Appendix A: Rules to fallow by Computer Lab Users	112
Appendix B: Endorsement.....	114

Laboratory Hardware, Software / Tools Requirements:

In this lab the students will be using SQL Plus, Oracle Forms Builder and Oracle Reports Builder.

Connecting to Oracle server

1. Select Local Net Service Name Configuration, then click Next



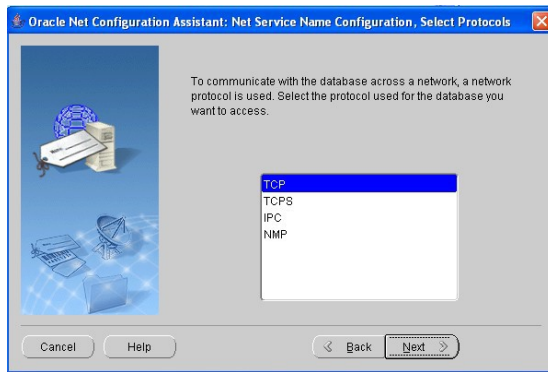
2. Select Add, then click Next



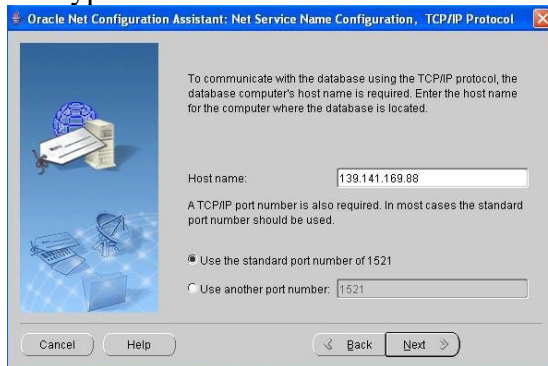
3. Write a server name, then click Next.



4. Select the network protocol.



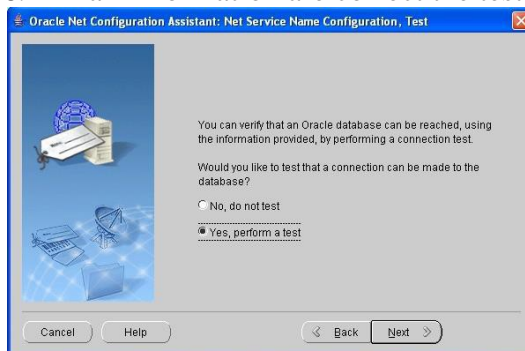
5. Type host name IP address



6. Test service to insure information

7. You will be asked for user name and password

8. If all information are correct the test will succeed, click next then finish



Laboratory Schedule

#	Lab Title	Lab activity
1	Introduction to Oracle DBMS	
2	Creating a Database	
3	Retrieving Information from Database Tables I	
4	Retrieving Information from Database Tables II	
5	Retrieving Information from Database Tables III	
6	Creating Sequences and Views	
7	Report Builder	
8	Oracle Forms Builder I	
9	Oracle Forms Builder II	
10	Oracle Forms Builder III	
11	Introduction to PL/SQL Programming	
12	Applying PL/SQL On Database	

Laboratory Policy

- Follow the laboratory rules listed in appendix “A”
- To pass this course, the student must pass the lab-component of the course.
- Cheating in whatever form will result in F grade.
- Attendance will be checked at the beginning of each Lab.
- Absence for three (03) or more unexcused labs will result in a F grade in the Course.
An official excuse must be shown in one week following return to classes.
- Every unexcused absence leads to a loss of 0.5 % marks.
- Cheating in Lab Work or Lab Project will result F grade in Lab.
- Late Submission of Home Works & Projects will not be accepted.
- There will be no make-up for any Quiz/Exam/Lab.
- Hard work and dedication are necessary ingredients for success in this course.

Laboratory Grading Policy

Activity	Weight
Lab Work	5%
Lab Quizzes	5%
Total	10%

Introduction

This lab is an integral part of the course ISC 321 Database Systems I. The main objectives of the lab are to introduce the students with SQL programming and the fundamental concepts of Oracle Reports Builder and Oracle Forms Builder.

Lab #1 – Introduction to Oracle DBMS

1. Laboratory Objective:

In this lab introduces a brief introduction to DBMS and Oracle 11g

2. Laboratory Learning Outcomes:

- In this lab, you will learn the following:
 - The meaning of DBMS.
 - Oracle DBMS and Oracle 11 g.
 - Relational database.
 - The different between oracle database and oracle instance.
 - The meaning of tables, SQL and schemas.
 - The types of privileges: system privileges and object privileges.
 - The types of SQL Statements.
 - Create user account in Oracle.

3. Introductory Concepts

Database Management System:

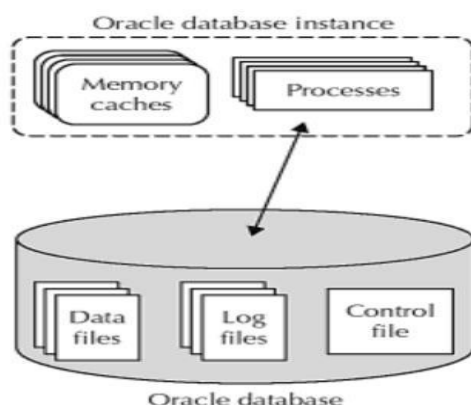
- Database management system (DBMS) is computer software that manages access to databases

Oracle DBMS

- produced and marketed by Oracle Corporation
- A Relational database management system (RDBMS)
 - collections of related information are organized into structures called tables

Oracle Fundamentals

- **Oracle Database:** collection of related OS files that oracle uses to store and manage a set of related information. (physical storage of information)
- **Oracle instance (Oracle Server):** the oracle software that manage the physical db access



- **Tables:** Organized collection of rows (records) columns (attributes).
- **Structured Query Language (SQL):** command language designed to access relational databases.

The diagram illustrates a database table structure. A box represents the table, with 'Columns' labeled above it and 'Rows' labeled to its left. Arrows point from the labels to the corresponding parts of the table. The table has four columns: ID, COMPANYNAME, LASTNAME, and FIRSTNAME. It contains four rows of data.

ID	COMPANYNAME	LASTNAME	FIRSTNAME
1	McDonald Co.	Joy	Harold
2	Car Audio Center	Musial	Bill
3	Wise Trucking	Sams	Danielle
4	Rose Garden Inn	Elias	Juan

- **Schemas:** collection of objects such as tables, views and sequences and owned by a database user.
- **Privileges:** the right to execute particular SQL statements.
 - System privileges: Gaining access to database
 - Object privileges: Manipulating the content of the database objects

Types of SQL Statements

- **Data Definition Language (DDL) statements:** Allow you to define the data structures, such as tables using CREATE, ALTER, DROP, RENAME and TRUNCATE statements.
- **Data Manipulation Language (DML) statements:** to modify the contents of tables using INSERT, UPDATE and DELETE statements.
- **Query statements:** to retrieve rows stored in database tables using the SELECT statement.

Create Account

- You should have system privilege to create user
- The database administrator creates the users
- Syntax


```
CREATE USER user_name
IDENTIFIED BY password;
```
- Where
 - user_name: The name of the user to be created
 - password: the password for the user
- The user must have the necessary permissions to work in the database
- Permissions are granted by a privileged user using the GRANT statement
- Syntax:


```
GRANT privilege [, privilege ...]
TO user_name;
```

Changing Your Password

- You can ONLY change your password, only administrators are allowed to change others passwords
- Syntax:

```
ALTER USER username  
  IDENTIFIED BY new_password;
```

Lab #2 - Creating a Database

1. Laboratory Objective:

This lab introduces the SQL statements to

2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
 - Learn Oracle basic datatypes.
 - Explore how to create tables, constraints.
 - Editing the tables, constraints.
 - Adding, modifying and removing rows from a table.

3. Introductory Concepts (for this laboratory)

Basic Data Types

○ Character Datatypes

- CHAR: stores character values with a fixed length
CHAR(10) = "Rick", "Jon", "Stackowiak"
- VARCHAR2: stores variable-length character strings
VARCHAR2(10) = "Rick", "Jon", "Stackowiak"

○ Numeric Datatypes

- INTEGER: Stores integer numbers.
 - NUMBER: provide a precision of 38 digits and can accept two qualifiers, precision and scale
 - Precision: total number of significant digits in the number
 - scale represents the number of digits to the right of the decimal point.
If no scale is specified, Oracle will use a scale of 0
- NUMBER(3) = 124
NUMBER(5,2) = 124.26

○ Date Datatype

- DATE: stores the date in the form of DD- MON-YY

○ Other Datatypes

- RAW and LONG RAW: store objects with their own internal format, such as bitmaps.
 - RAW can hold 2 KB,
 - LONG RAW can hold 2 GB
- ROWID: represents the specific address of a particular row

Create Table

```
CREATE TABLE table_name (  
    column datatype [CONSTRAINT constraint_def DEFAULT default_exp]  
    [...]  
);
```

○ Where

- *table_name* specifies the name you assign to the table.
- *column* specifies the name you assign to a column.

- *datatype* specifies the type of a column.
- *constraint_def* specifies the definition of a constraint on a column.
- *default_exp* specifies the expression used to assign a default value to a column.

Naming Rules and Guidelines

- Table names and column names:
 - Must begin with a letter.
 - Must be 1 to 30 characters long.
 - Must contain only A-Z, a-z, 0-9, _, \$, and #.
 - Must not duplicate the name of another object owned by the same user.
 - Must not be an Oracle server reserved word.
- Use descriptive names for tables and columns
- Names are **case-insensitive**.

Oracle Server reserved words

ACCESS	DELETE	IS	PCTFREE	TO
ADD	DESC	LEVEL	PRIOR	TRIGGER
ALL	DISTINCT	LIKE	PRIVILEGES	UID
ALTER	DROP	LOCK	PUBLIC	UNION
AND	ELSE	LONG	RAW	UNIQUE
ANY	EXCLUSIVE	MAXEXTENTS	RENAME	UPDATE
AS	EXISTS	MINUS	RESOURCE	USER
ASC	FILE	MLSLABEL	REVOKE	VALIDATE
AUDIT	FLOAT	MODE	ROW	VALUES
BETWEEN	FOR	MODIFY	ROWID	VARCHAR
BY	FROM	NESTED_TABLE_ID	ROWNUM	VARCHAR2
CHAR	GRANT	NOAUDIT	ROWS	VIEW
CHECK	GROUP	NOCOMPRESS	SELECT	WHENEVER
CLUSTER	HAVING	NOT	SESSION	WHERE
COLUMN	IDENTIFIED	NOWAIT	SET	WITH
COLUMN_VALUE	IMMEDIATE	NULL	SHARE	
COMMENT	IN	NUMBER	SIZE	
COMPRESS	INCREMENT	OF	SMALLINT	
CONNECT	INDEX	OFFLINE	START	
CREATE	INITIAL	ON	SUCCESSFUL	
CURRENT	INSERT	ONLINE	SYNONYM	
DATE	INTEGER	OPTION	SYSDATE	
DECIMAL	INTERSECT	OR	TABLE	
DEFAULT	INTO	ORDER	THEN	

Constraints

- ensures that data violating that constraint is never accepted
- Types
 - *NOT NULL*
 - *Unique*: cannot enter values that already exist in another row in the table
 - *Primary key*:

- enforces both the unique and the NOT NULL constraints.
- will create a unique index the specified column(s).
- *Foreign key*: is defined for a table (known as the child) that has a relationship with another table in the database (known as the parent).
- *Check*: is a Boolean expression that evaluates to either TRUE or FALSE.

Constraint Guidelines

- You can name the constraint, or Oracle server generate a name
- Create a constraint at either of the following times:
 - At the same time as the table is created
 - After the table has been created
- View a constraint in the data dictionary

Create Table Example

Create the student table with the following:

Attributes	Data Type	Length	Constraint
SID	Number	8	Primary Key
f_name	Character	20	Not null
major	Character	20	Not null
current_credits	Number	3	

Solution:

```
CREATE TABLE Student
(SID      NUMBER(8)  CONSTRAINT st_SID PRIMARY KEY,
f_name   VARCHAR2(20) CONSTRAINT st_fname NOT NULL,
major    VARCHAR2(20) CONSTRAINT st_major NOT NULL,
current_credits NUMBER(3)
);
```

Getting Information on Tables

- Performing a DESC[RIBE] command on the table
- Example:


```
DESCRIBE student
DESC student
```

Getting Information on Constraints

```
SELECT constraint_name, constraint_type, status
FROM user_constraints
WHERE table_name = 'STUDENT';
```

* Note table name must be in capital letter

Changing a Table

- You change/alter a table using the ALTER TABLE statement.
- You can use ALTER TABLE to perform tasks such as:
 - Add, modify, or drop a column
 - Add or drop a constraint
 - Enable or disable a constraint

Adding a Column

- Use ADD clause to add column

○ Syntax:

```
ALTER TABLE table_name  
ADD (column datatype [CONSTRAINT constraint_def DEFAULT default_exp]);
```

Example:

```
ALTER TABLE Student  
ADD (l_name VARCHAR2(15));
```

Modifying a Column

- Change the size of a column
- Change the precision of a numeric column
- Change the data type of a column
- Change the default value of a column
 - Affects only subsequent insertion to the table

○ Syntax:

```
ALTER TABLE table_name  
MODIFY (column new_datatype );
```

Example:

```
ALTER TABLE Student  
MODIFY (f_name VARCHAR2(15));
```

```
ALTER TABLE Student  
MODIFY (major CHAR(3) );
```

Add a primary key constraint

- To create a PRIMARY KEY constraint on a column when the table is already created.
- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name PRIMARY KEY (column1, column2,..., column_n);
```

- **Note:** If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

- Steps to specify the primary key (more than one column) for the CLASSES table

Step 1: Create the classes table

```
CREATE TABLE Classes  
(dep varchar2(3),  
course number(3),  
current_student number(3),  
num_credits number(1),  
room_id number(8) );
```

Step 2: Alter the classes table to add primary key constraint

```
ALTER TABLE Classes
ADD CONSTRAINT cls_pk
PRIMARY KEY (dep,course);
```

Add a foreign key constraint

A **foreign key** is a relationship or link between two tables which ensures that the data stored in a database is consistent.

The foreign key link is set up by matching columns in one table (the **child**) to the primary key columns in another table (the **parent**).

○ Syntax:

```
ALTER TABLE child_table_name
ADD CONSTRAINT constraint_name FOREIGN KEY (column1, column2,...)
REFERENCES parent_table_name (column1, column2,...)
[ ON DELETE referential_action ];
```

○ Example:

CUSTOMER and ORDERS tables

CUSTOMER table includes all customer data

ORDERS table includes all customer orders.

The constraint here is that all orders must be associated with a customer that is already in the CUSTOMER table. In this case, we will place a foreign key on the ORDERS table and have it relate to the primary key of the CUSTOMER table. This way, we can ensure that all orders in the ORDERS table are related to a customer in the CUSTOMER table.

In other words, the ORDERS table cannot contain information on a customer that is not in the CUSTOMER table.

CUSTOMER Table	
Attributes	Description
CID	Primary Key
Name	
Phone	

ORDERS Table	
Attributes	Description
Order_ID	Primary Key
Order_Date	
Customer_ID	Foreign Key
Amount	

How to specify the foreign key when creating the ORDERS table

Step 1: Create the customer table

```
create table customer (  
  cid number(10) constraint pk_cust primary key,  
  name varchar2(20),  
  phone number(8));
```

Step 2: Create the orders table without foreign key constraint

```
create table orders (  
  order_id number(10) constraint pk_ord primary key,  
  order_date date,  
  customer_id number(10),  
  amount number(6,3));
```

Step 3: Alter the orders table to add foreign key constraint

```
ALTER TABLE orders  
ADD CONSTRAINT orders_fkey  
FOREIGN KEY (customer_id) REFERENCES customer (cid)  
ON DELETE CASCADE;
```

Adding a CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- It is a Boolean expression that evaluates to either TRUE or FALSE.
- Syntax:
ALTER TABLE table_name
ADD CONSTRAINT constraint_name CHECK (column_name condition);

- Example:

```
ALTER TABLE Student  
ADD CONSTRAINT student_major  
CHECK ( major IN ( 'CSC ', 'HIS ', 'ECN ', 'MUS ', 'NUT ' ) );
```

```
ALTER TABLE Student  
ADD CONSTRAINT  
student_credits CHECK  
(current_credits > 0);
```

Dropping a Column

- Use DROP COLUMN clause to drop columns you no longer need from the table
- The column may or may not contain data
- Only one column can be dropped at a time
- The table must have at least one column remaining in it after it is altered.
- Syntax:
ALTER TABLE table_name
DROP COLUMN name;
- Example:

```
ALTER TABLE student
DROP COLUMN f_name;
```

Dropping a Constraint

- Syntax:
ALTER TABLE table_name
DROP CONSTRAINT constraint;
- Example:
ALTER TABLE student
DROP CONSTRAINT std_major;

Truncating a Table

- To delete the entire contents of a table
TRUNCATE TABLE table_name;

Drop a Table

- Remove rows and data structure
- Can NOT be rolled back
- Syntax:
DROP TABLE table_name;

Rename Table

- You rename a table using the RENAME statement
- Syntax:
RENAME table_name TO new_table_name;

Adding Rows

- Use INSERT statement to add rows to a table.
- Supply a value for primary key and all columns that are defined as NOT NULL.
- Enclose character and date values within single quotation marks.
- Optionally list the columns in the INSERT clause.
- If columns list is omitted, all the values must be listed in the default order of the column in the table.
- Syntax:
INSERT INTO table_name [(column1, column2, ...)]
VALUES (value1, value2, ...);

```
INSERT INTO student
(sid, f_name, l_name ,major ,current_credits)
VALUES (1000,'Sara','Jassem','CSC',120);
```

```
INSERT INTO student
(sid, f_name, l_name, major, current_credits)
VALUES (1001, 'Maha','Naser', 'ttt',60);
```

Modifying Rows

- Use UPDATE statement to modifying existing rows in a table.

- Syntax:

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

- Example:

```
UPDATE student  
    SET current_credits=66  
    WHERE sid=1001;
```

Removing Rows

- Remove existing rows from table by using the DELETE statement.
- Specific rows are deleted when you specify the WHERE clause.
- All rows in the table are deleted if you omit the WHERE clause.
- Syntax:

```
DELETE [from] table_name  
    [WHERE condition];
```

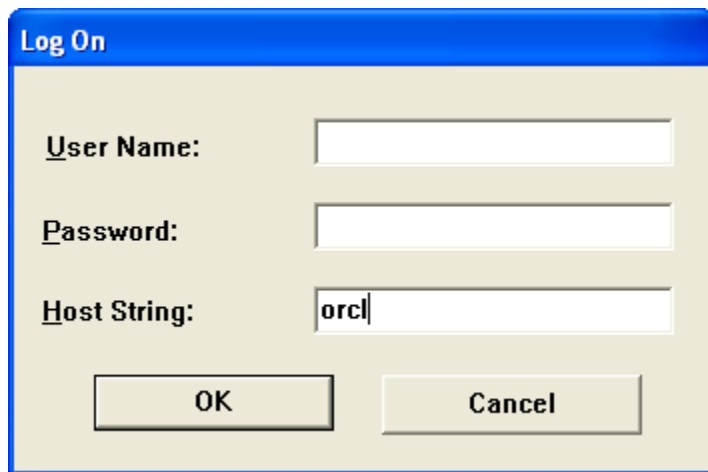
4. Laboratory Instructions

How to use SQL Plus?

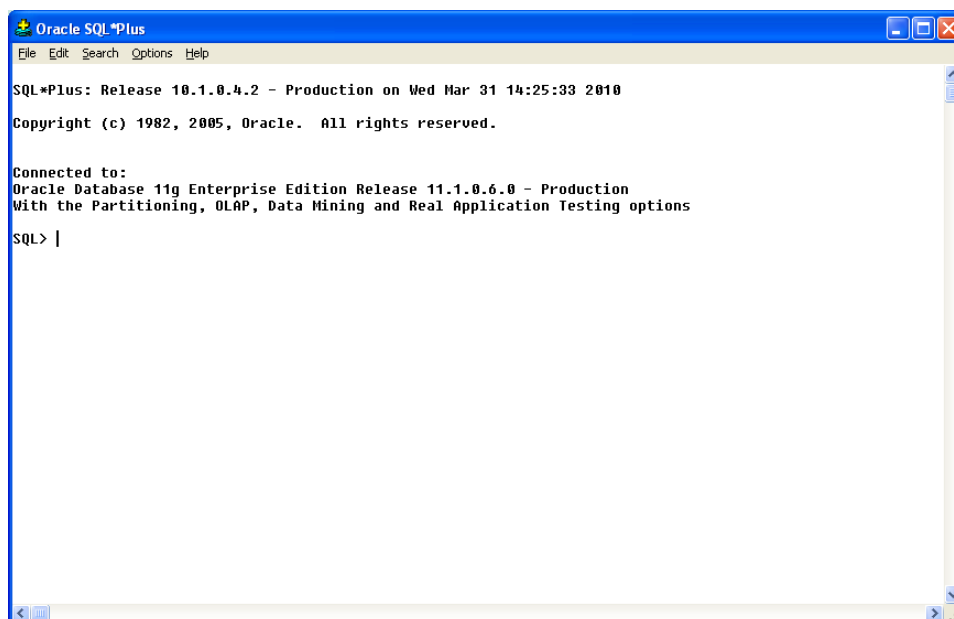
1. Click on the SQL Plus icon in your desktop as shown in the picture below



2. In log on window type your user name and password as given to you by instructor. In the host string type *orcl*



3. The following figure shows the results of logging into Oracle using SQL*Plus

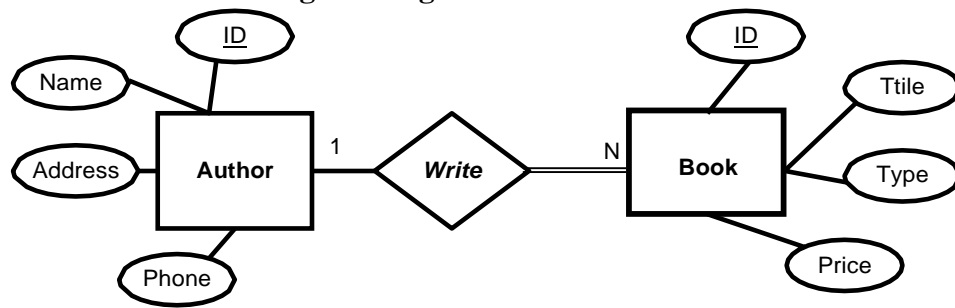


SQL*Plus commands

```
SQL> CLEAR SCREEN
SQL> SHOW USER
SQL> SET PAGESIZE 100
SQL> SET LINESIZE 100
```

5. Laboratory Exercises

Consider the following ER diagram



When reducing this ERD into tables you will get:

Author

<u>ID</u>	Name	Address	Phone

<u>ID</u>	<u>Title</u>	Price	Author_ID

1. Write SQL statements to create the previous tables according to the following constraints:

Author Table

Attributes	Data Type	Length	Description
ID	Integer	5	Primary Key
Name	Character	25	Not Null
Address	Character	35	
Phone	Number	8	

Book Table

Attributes	Data Type	Length	Description
ID	Integer	9	Primary Key 1
Title	Character	35	Primary Key 2
Price	Number	5,3	
Author_ID	Integer	5	Foreign Key

2. Insert the following Data into **Author Table**

<u>ID</u>	Name	Address	Phone
12121	John Smith	123, 4 th Street	52525252

3. Insert the following Data into **Book Table**

<u>ID</u>	Title	Author_ID
123456789	The Secret	12121

4. Modify the book table to add the following:

Attribute	Data Type	Length
Type	Character	20

Lab #3 – Simple Query Statements

1. Laboratory Objective:

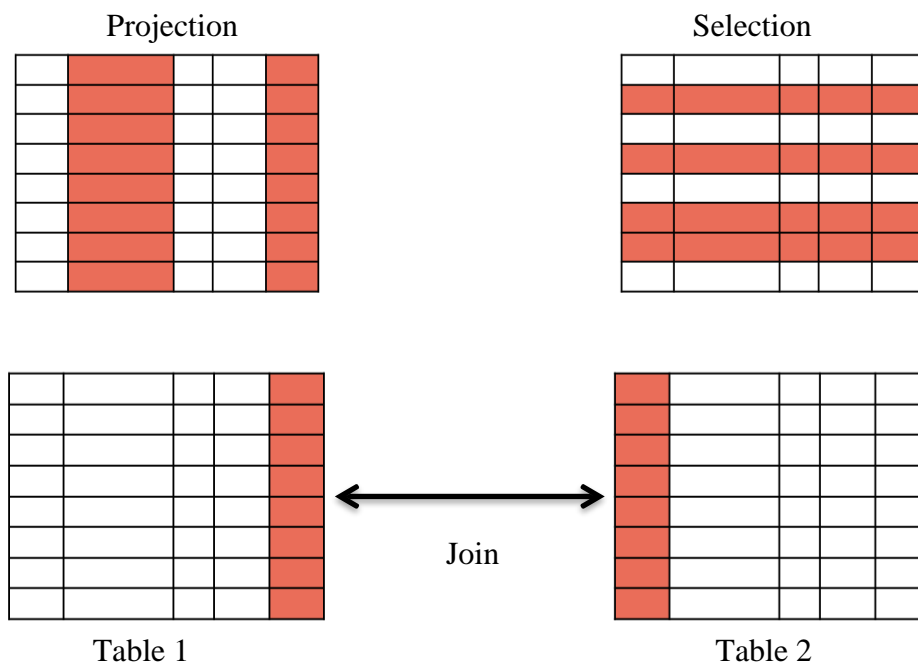
The objective of this laboratory is to introduce students to retrieve information from one database table using select statements.

2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
 - Perform single table query
 - Use aliases to reference tables and columns
 - Merge column output using concatenation operator
 - Limit the rows that are retrieved by a query

3. Introductory Concepts

Capabilities of SQL Select Statements



Using SELECT statement , you can do the following:

Projection: choosing All/ particular columns in table that you want to returned by your query.

Selection: choosing particular rows in a table.

Joining: bring together data that stored in different table.

Basic SELECT Statement

- Used to retrieve information from database tables
- Syntax:

```
SELECT [DISTINCT] *, or column [alias],...,
FROM table1,table2,...,tablen,
[WHERE condition(s)];
```

Select is a list of one or more columns
 "*" Selects all columns
 Distinct suppresses duplicates
 Column/expression name of column or expression
 Alias gives selected column different heading
 From table specifies the table containing the columns

- List all student information
select * from student;
- Display the building and room numbers in every building
select building, room_number
from rooms;

Performing Arithmetic

- You can create expressions with number and date data by using arithmetic operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplications
/	Division

- You can use arithmetic operators in any clause of a SQL statement except the FROM clause
select building, room_number, 0.1*(number_seats + 20)
from rooms;

Using Column Aliases

- Renames a column heading
- Useful with calculations
- Immediately follows the column name
- Required double quotation marks if it contains spaces or special characters or if it is case sensitive

```
select dep "Department Name", course, current_student
from classes;
```

```
select dep, course, 2*current_student Double_Student
from classes;
```

Combining Column Output Using Concatenation

- Links columns or character strings to other columns

- Represented by two vertical bars (||)
 - Creates a resultant column that is a character expression
- ```
select dep|| ' ' || course "course Name"
from classes;
```

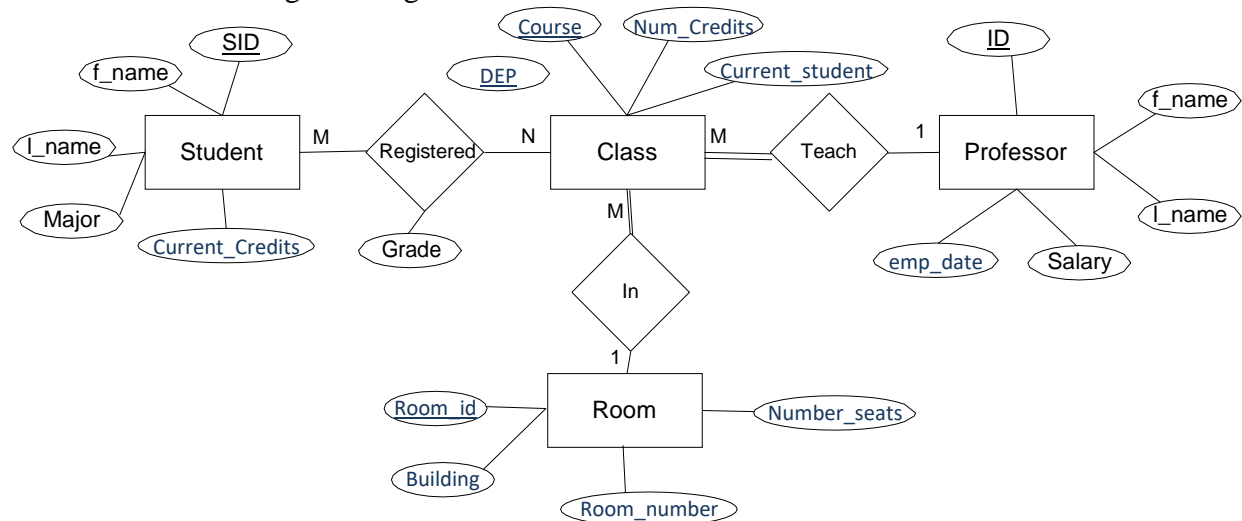
#### The WHERE Clause

- Used to specify the rows you want to retrieve
- The WHERE clause follows the FROM clause
- In a WHERE clause simple conditions based on comparison operator is used to identify specific row.
- The WHERE clause can compare values in columns, literal values, arithmetic expressions, or functions.
- Example:
- Display all rooms numbers and buildings for rooms located in Building 7  

```
select Room_number, Building
from rooms
where Building='Building 7';
```

#### 4. Laboratory Instructions

Consider the following ER Diagram



| Student    |        |        |       |                 |
|------------|--------|--------|-------|-----------------|
| <u>SID</u> | f_name | L_name | Major | Current_Credits |

| Professor |        |        |        |          |
|-----------|--------|--------|--------|----------|
| <u>ID</u> | f_name | l_name | Salary | emp_date |

| Rooms |  |  |  |  |
|-------|--|--|--|--|
|-------|--|--|--|--|

| <u>Room_id</u> | Building | Room_number | Number_seats |
|----------------|----------|-------------|--------------|
|----------------|----------|-------------|--------------|

| Classes    |               |                 |             |         |         |
|------------|---------------|-----------------|-------------|---------|---------|
| <u>DEP</u> | <u>Course</u> | Current_student | Num_Credits | room_id | prof_id |

| Registered_Students |            |               |       |
|---------------------|------------|---------------|-------|
| <u>Student_ID</u>   | <u>DEP</u> | <u>Course</u> | Grade |

We have 5 different majors/departments

|     |                  |
|-----|------------------|
| CSC | Computer Science |
| HIS | History          |
| ECN | Economics        |
| MUS | Music            |
| NUT | Nutrition        |

## 5. Laboratory Problem Description

- 1) List the course and room id in History classes
- 2) Display all students IDs, first names, and last names whose major is computer science
- 3) Display all grades for student 1009

## 6. Laboratory Exercises

1. Display the full name (first name and last name) and major of all students
2. Display the buildings and the room numbers of all rooms which have 50 seats
3. Display the full name (first name and last name) and passed credits plus 4 of students whose major is music

## Lab #4 – Using Simple Functions In Query Statements

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to the use of different Operators in select statements

### 2. Laboratory Learning Outcomes:

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

- Using SQL operators
- Using comparison operators
- Using Logical operators
- Sorting the result
- Using two tables in select statement

### 3. Introductory Concepts

#### Displaying Distinct Rows

Eliminating Duplicate Rows by using DISTINCT keywords in the SELECT clause

```
select major
from student;
```

```
select distinct major
from student;
```

#### Comparing Values

Comparison operators compare one expression with another.

| Operator | Description                                 |
|----------|---------------------------------------------|
| =        | Equal                                       |
| <> Or != | Not equal                                   |
| >        | Greater than                                |
| <        | Less than                                   |
| >=       | Greater than or equal                       |
| <=       | Less than or equal                          |
| ANY      | Compares one value with any value in a list |
| ALL      | Compare one value with all values in a list |

- Retrieve all student information whose major is not computer science  
Select \*  
from student  
where major != 'CSC';
- Write a query to display room number and building name of rooms having capacity (max seats) more than 50

```
Select building, room_number
from rooms
where number_seats > 50;
```

- Display all classes information where course number is either 101 or 102

```
Select *
from classes
where course = any (101, 102);
```

- Retrieve the student id and first name of all students where student ids is greater than either 1004 or 1006

```
select sid, f_name
from student
where sid >= any (1004, 1006);
```

- Retrieve the student id and first name of all students where student ids is greater than both 1004 or 1006

```
select sid, f_name
from student
where sid >= all (1004, 1006);
```

### Using the SQL Operators

| Operator       | Description                                                                                    |
|----------------|------------------------------------------------------------------------------------------------|
| <b>LIKE</b>    | Matches patterns in strings                                                                    |
| <b>IN</b>      | Matching lists of values                                                                       |
| <b>BETWEEN</b> | Matched range of values                                                                        |
| <b>IS NULL</b> | Matches null values                                                                            |
| <b>NOT</b>     | Reverse the meaning of an operator<br>✓ NOT LIKE<br>✓ NOT IN<br>✓ NOT BETWEEN<br>✓ IS NOT NULL |

### Using The Like Operator

- allows comparisons on parts of character string
  - ( \_ ) replaces single character
  - ( %) replaces an arbitrary number of characters

- Examples:

```
select sid, f_name
from student
where f_name like '%al%';
```

```
select *
```

```
from classes
where dep not like '%S';
```

```
select *
from classes
where dep like '%S_';
```

#### Using The IN Operator

- Retrieve the rows whose column value is in a list
- Can be used with any data type
- IN is equivalent to = ANY
- NOT IN is equivalent to != ALL
- Example:

```
Select *
from classes
where course in (101, 102);
```

#### Using The Between Operator

- To retrieve the rows whose column value is in a specific range
- The range is inclusive
- Example:

```
select *
from registered_students
where grade between 'A' and 'B';

select *
from student
where sid between 1001 and 1005;
```

#### Using The Logical Operators

- Allow to limit rows based on logical conditions

| Operator       | Description                             |
|----------------|-----------------------------------------|
| <b>x AND y</b> | Returns true when both x and y are true |
| <b>x OR y</b>  | Returns true when either x or y is true |
| <b>NOT x</b>   | Returns true if x is false              |

#### Logical Operators Example

```
select *
from registered_students
where student_id > 1006 and grade between 'A' and 'B';

select *
from registered_students
where student_id > 1006 or grade between 'A' and 'B';
```

#### Rules of Precedence



4. Display the full name and the salary of the professor who teaches class HIS101.

5. Get room number and the building for the classes 300 level classes.

**5. Laboratory Exercises**

1. Display room number and building of rooms having seats less than 100 or more than 350
2. List the professor full name who was hired in 2008
3. Display the id, the full names and the grades of students registered in class NUT307  
order by student id

## Lab #5 – Grouping Rows and Subqueries

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to more complicated select statement

### 2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
  - Using table alias
  - Identifying available group functions
  - Group data by using the group by clause
  - Define subqueries
  - Write single-row and multiple-row subqueries

### 3. Introductory Concepts

#### Using Table Aliases

- SQL aliases are used to temporarily rename a table
- Example

```
select c.dep, c.course, r.building, r.room_number
from classes c, rooms r
where c.room_id = r.room_id;
```

In the previous example, the alias c is assigned to classes table and the alias r is assigned to rooms table.

#### Group Functions

Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group.

There are two rules that you must follow when using group functions:

- Can be used in both the SELECT and HAVING clauses (the HAVING clause is covered later).
- Cannot be used in a WHERE clause.

| Functions                           | Purpose                                                                                                                                |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <b>AVG ([DISTINCT/ALL] n)</b>       | Average the value of n, ignoring null values                                                                                           |
| <b>COUNT(*/[DISTINCT/ALL] expr)</b> | Number of rows, where expr evaluates to something other than null (count all select rows using *, including duplicates rows and nulls) |
| <b>MAX([DISTINCT/ALL] expr)</b>     | Maximum value of expr, ignoring null values.                                                                                           |
| <b>MIN([DISTINCT/ALL] expr)</b>     | Minimum value of expr, ignoring null values.                                                                                           |



|                                   |                                                |
|-----------------------------------|------------------------------------------------|
| <b>STDDEV([DISTINCT/ALL] n)</b>   | Standard deviation of n, ignoring null values. |
| <b>SUM([DISTINCT/ALL] n)</b>      | Sum value of n, ignoring null values.          |
| <b>VARIANCE([DISTINCT/ALL] n)</b> | Variance of n, ignoring null values.           |

#### Example of Group Function

- select max(salary), min(salary), avg(salary), sum(salary)  
from professor;
- select count(distinct prof\_id)  
from classes;
- select count(\*)  
from registered\_students  
where student\_id = 1002;

#### Using the GROUP BY

- To group rows into blocks with a common column value
- The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
- GROUP BY returns one record for each group.
- Syntax:

```
SELECT column-name1, column-name2, ... , column-nameN
FROM table-name
[WHERE condition]
GROUP BY column-name1, column-name2, ... , column-nameN
```

- Example:

```
select prof_id
from classes
group by prof_id
order by prof_id;
```

```
PROF_ID

20000
20002
20003
20004
```

```
select prof_id, count(*)
from classes
group by prof_id
order by prof_id;
```

```
PROF_ID COUNT (*)
```

|       |       |
|-------|-------|
| ----- | ----- |
| 20000 | 2     |
| 20002 | 2     |
| 20003 | 3     |
| 20004 | 3     |

### Using Multiple Columns in a Group

```
select dep,course,count(*)
from registered_students
group by dep,course
order by dep;
```

### Using Having

- HAVING filters records that work on GROUP BY results.
- HAVING applies to group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.

### ○ Example:

```
select prof_id,count(*)
from classes, professor
where prof_id = id
group by prof_id;
```

|         |             |
|---------|-------------|
| PROF_ID | COUNT ( * ) |
| -----   | -----       |
| 20003   | 3           |
| 20002   | 2           |
| 20004   | 3           |
| 20000   | 2           |

```
select prof_id,count(*)
from classes, professor
where prof_id = id
group by prof_id
Having count(*)>2;
```

|         |             |
|---------|-------------|
| PROF_ID | COUNT ( * ) |
| -----   | -----       |
| 20003   | 3           |
| 20004   | 3           |

### Subqueries

- Some problems can be solved by combining 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).
- Syntax:



5. Find all room numbers and buildings for rooms in which professor John Smith teach. Order the result according to the room number in ascending order.
  
  
  
  
  
  
  
  
  
  
6. Retrieve the name of students who took A in all their registered classes.

## **5. Laboratory Exercises**

1. Display the maximum, minimum, and average number of seats for all rooms having seats between 10 and 100
2. Display the name and salary of professors having salary greater than Smith's Salary
3. Retrieve the classes (department and course) that have teachers having salary more than 1600 using subquery and without subquery

## Lab #6 – Creating Sequences and Views - TOAD

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to sequences and views

### 2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
  - How to create and use sequence
  - How to create and use views
  - Use TOAD program

### 3. Introductory Concepts

#### Sequence

- It is a database item that generates a sequence of integer.
- Used to create primary key values.
- Syntax:

```
CREATE SEQUENCE [schema.]sequence_name
[INCREMENT BY integer_value]
[START WITH integer_value]
[MAXVALUE integer_value | NOMAXVALUE] [MINVALUE integer_value | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE integer_value | NOCACHE]
[ORDER | NOORDER];
```

- **INCREMENT BY** indicates how the sequence numbers should change. The default increment is 1. A negative number will cause decrement the sequence.
- **START WITH** indicates the initial value of the sequence. The default start value for ascending sequences is MINVALUE and MAXVALUE for descending. Note: START WITH cannot be more than MAXVALUE.
- **MINVALUE** is the lowest number the sequence will generate. The default is 1 for ascending sequences.
- **MAXVALUE** is the highest number the sequence will generate. The default is -1 for ascending sequences.
- **CYCLE** option makes the sequence to restart either type of sequence where its max value or min value is reached.
- **CACHE** allows a preallocated set of sequence numbers to be kept in memory. The default is 20. The value set must be less than max value minus min value.
- **NOCACHE** forces the data dictionary to be updated for each sequence number generated, guaranteeing no gaps in the generated numbers, but decreasing performance of the sequence.

- **ORDER** guarantees that the sequence numbers will be assigned to instances requesting them in the order the request are received. This is useful in applications requiring a history of the sequence in with the transactions took place.

#### Sequence Example

- CREATE SEQUENCE MyFirstSequence;
- CREATE SEQUENCE MySecondSequence  
INCREMENT BY 3  
START WITH 100;
- CREATE SEQUENCE MyThirdSequence  
MAXVALUE 40  
MINVALUE 30  
INCREMENT BY 3  
CACHE 3 CYCLE;

#### To Use Sequence

- select mysecondsequence.currval from dual;
- select mysecondsequence.nextval from dual;
- select \* from user\_sequences

#### Populate Primary Key Using a Sequence

- create table testing  
( id number(5) constraint testing\_pk primary key);
- INSERT INTO testing values (mysecondsequence.nextval);
  - Run the insert statement 4 times
- select \* from testing

#### Views

- Views are customized presentations of data in one or more tables or other views.
- You can think of them as stored queries.
- As with tables, views can be queried, updated, inserted into, and deleted from, with some restrictions.
- All operations performed on a view affect the base tables of the view.
- View used to:
  - Restrict data access
  - Make complex queries easy
  - Provide data independence.
  - Present different views of the same data.

### Creating View

```
CREATE [OR REPLACE] [FORCE/NOFORCE] VIEW view [{alias[, alias]}
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]];
[WITH READ ONLY [CONSTRAINT constraint]];
```

- **OR REPLACE**: re-create the view if it already exist.
- **FORCE**: creates the view whether or not the base table exist.
- **NOFORCE**: create the view only if the base tables exist (This is the default)
- **view**: is the name of the view.
- **alias**: specifies names for the expressions selected by the view query. (The number of alias must match the number of expressions selected by the view)
- **subquery**: select statement, you can use alias for the columns selected.
- **WITH CHECK OPTION**: specifies that only rows accessible to view can be inserted or updated.
- **constraint**: is the name assigned to CHECK OPTION constrain

### Creating View –Example

- CREATE or replace VIEW my\_view AS  
select \*  
from student  
where major='CSC';
- CREATE or replace VIEW Sview AS  
select sid,f\_name,l\_name,dep,course,grade  
from student, registered\_students  
where sid=student\_id  
order by sid;

We can query the view above as follows:

- Select \* from my\_view;
- Select \* from sview;

### Commit and Rollback

- Changes happened because of *insert*, *update*, and *delete* statements are temporarily stored in the database system. They become permanent only after the statement *commit*; has been issued.
- You can use the SQL *ROLLBACK* statement to rollback (undo) any changes you made to the database before a COMMIT was issued.

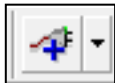
- The SQL COMMIT statement saves any changes made to the database. When a *COMMIT* has been issued, all the changes since the last COMMIT, or since you logged on as the current user, are saved.

### What is toad?

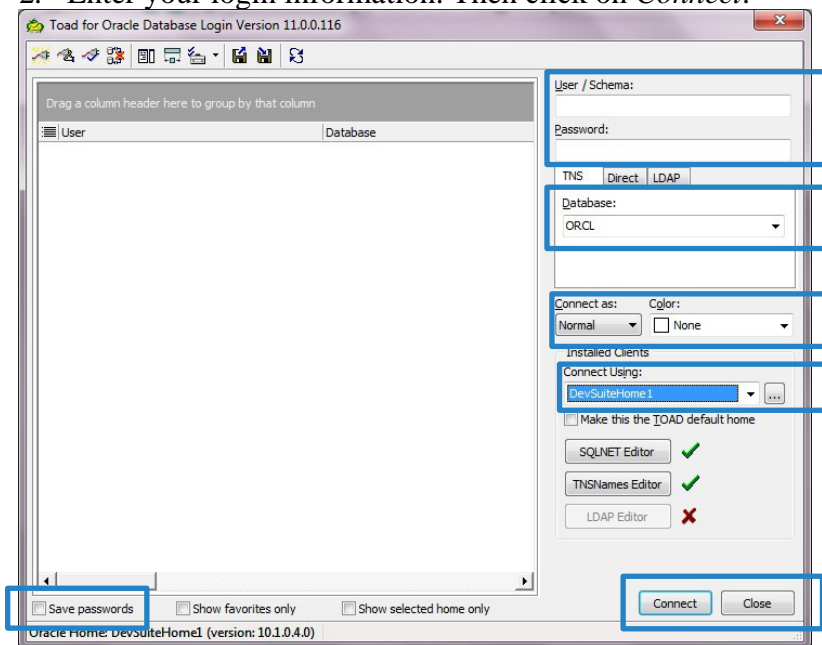
Toad for Oracle provides an efficient and accurate way for database professionals of all skill and experience levels to perform their jobs with an overall improvement in workflow and productivity.

### How to login and connect?

1. Click on *New Connection* in standard toolbar to open Database login screen



2. Enter your login information. Then click on *Connect*.



a. Enter User/Schema and password

b. Select connection method TNS and Database ORCL

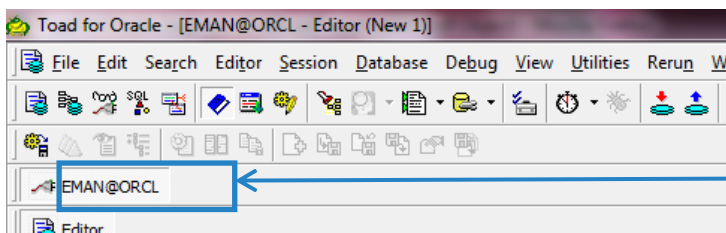
c. Select connection privilege (default is normal) and color

d. Select Oracle Home (DevSuiteHome1)

e. Deselect Save

f. Click on *Connect*

**Figure 1 Login Window**



Connection

The first toolbar (from) contains the following icons:



Open a New SQL Window



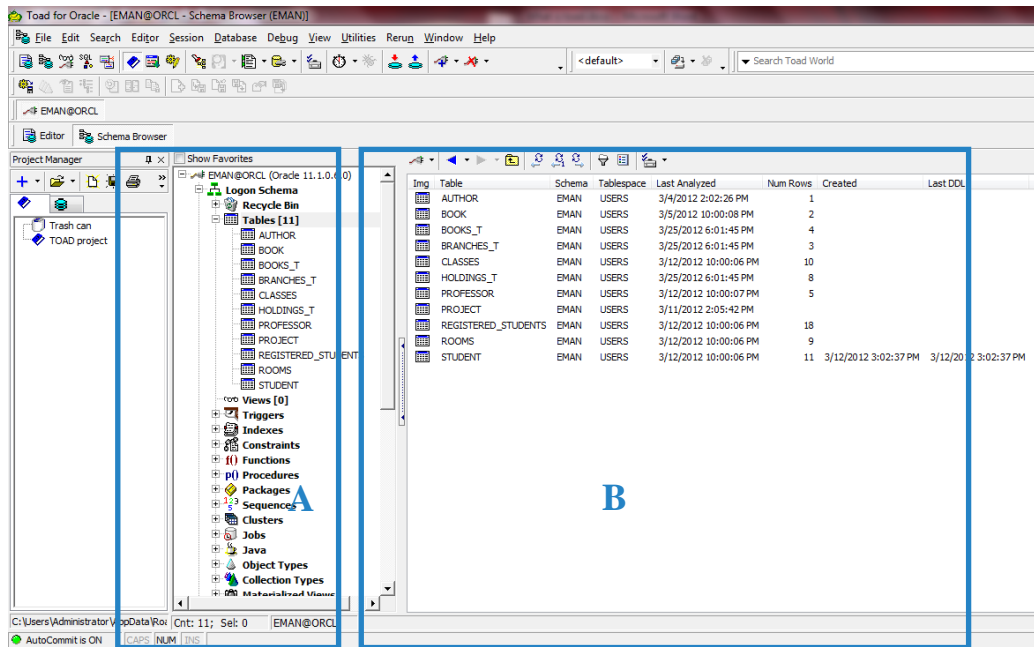
Open a New Schema Browser Window



## Working with Database Objects

### What is Schema Browser?

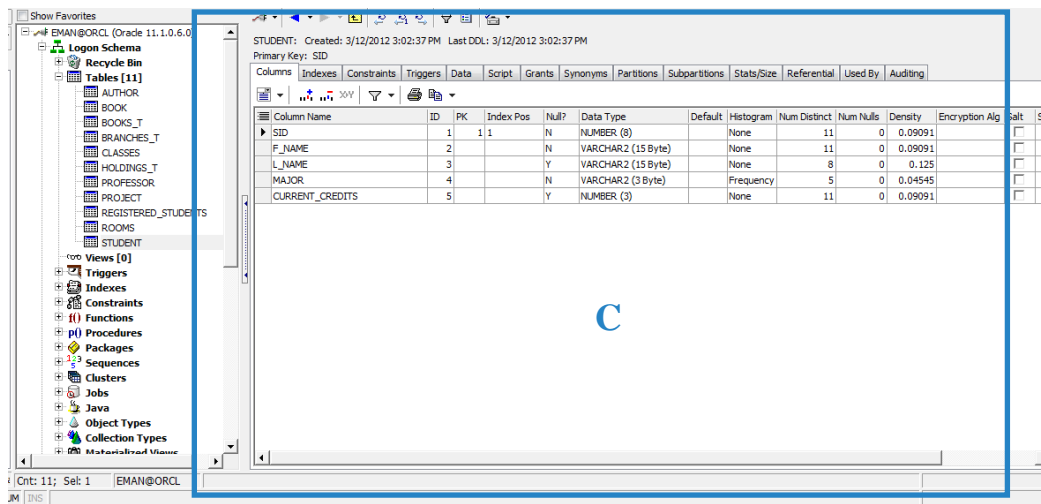
Schema browser allows quick exploring all database objects (view, add, and modify database objects). It also displays detailed information about a selected object. Toad can display your objects in a tree view, a dropdown selector, or a tab/page panel. As shown below.



**Figure 2** Tree view of Schema Browser. (A) List of objects pane. (B) Object details

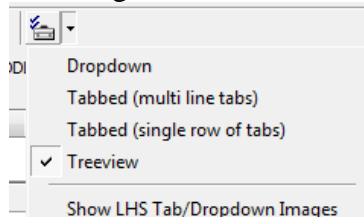
In List of objects pane you can displays the entire objects you can view. To display your tables, select *Logon Schema > Tables*.

The Object details pane displays the details of any object you select. Figure 3 below shows the details of the Student table.



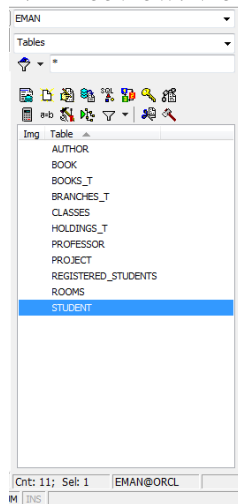
**Figure 4 (C)** The details of student table

To change how Schema Browser views the data, click on the *Browser Style*.

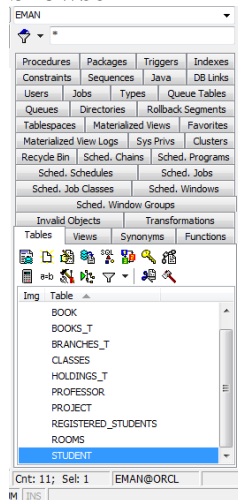


Then select one of the options:

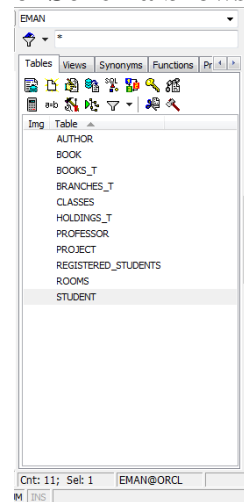
1. Dropdown: views object types in a drop-down field as shown in Figure 5.
2. Tabbed (multi line tabs): displays object types in multiple rows of tabs. Figure 6 shows multi line tabs view.
3. Tabbed (single row of tabs): displays object types as a single line of tabs. You must scroll through the tabs to view all object types. Figure 7 shows single row tabs view.
4. Treeview: views object types in tree view as shown in Figure 8.



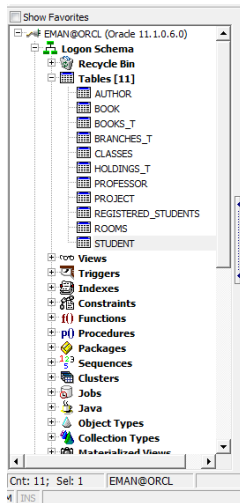
**Figure 5 Drop down view of Schema browser**



**Figure 6 Tabbed (multi line tabs) view of Schema browser**



**Figure 7 Tabbed (single row of tabs) view of Schema browser**



**Figure 8 Treeview of Schema browser**

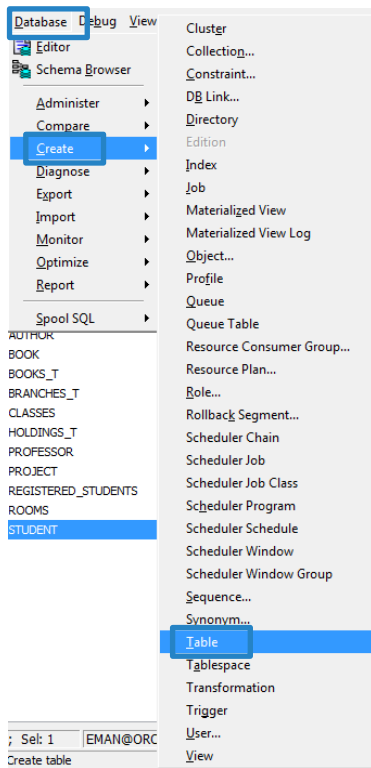
## Create Objects

TOAD allows to easily creating or altering database objects and it will generate the DDL statements. To create table, follow the following steps:

1. On the left hand side of Schema Browser, click on Create Table icon.



OR from the Menu bar, select *Database > Create > Table*



2. Write table name then click on *Add Col*

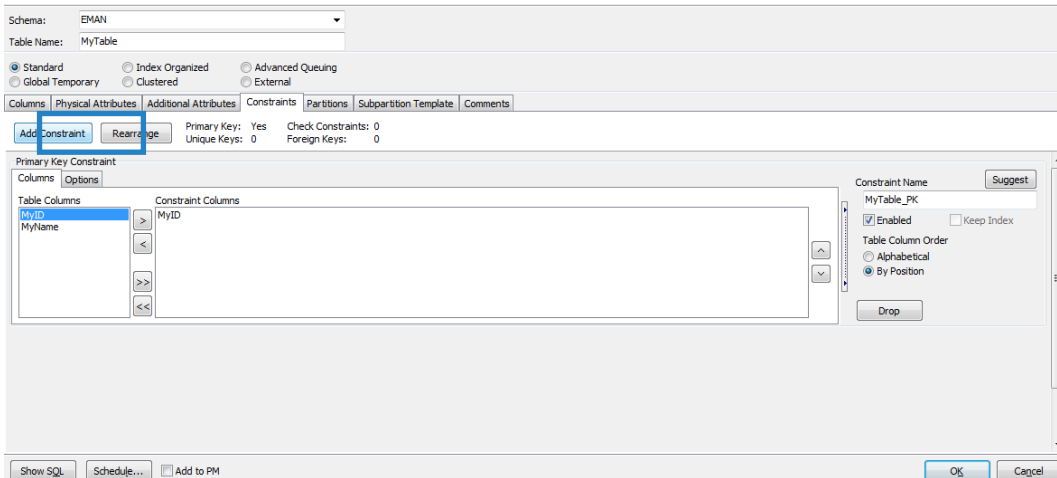
**Figure 9 Create table**

3. Write the column name, and then select the data type from the dropdown menu. Set the other constraints if it is required (Primary key, Not Null... etc.).

| ID | Column Name | Data Type | Size | Byte/Char | Precision | Scale | PK                                  | Not Null                            | Virtual                  | Default / Virtual | Ref                      | Encrypt Using            | Salt                     |
|----|-------------|-----------|------|-----------|-----------|-------|-------------------------------------|-------------------------------------|--------------------------|-------------------|--------------------------|--------------------------|--------------------------|
| 1  | MyID        | NUMBER    | 0    |           | 5         | 0     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |                   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2  | MyName      | VARCHAR2  | 20   |           | 0         | 0     | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/> |                   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

**Figure 10 Adding columns**

4. To add foreign key constraints, follow the steps shown below
  - a. Click on the constraints tab
  - b. The Constraints tab shows all constraint for the table. To add new constraint, click on *Add Constraint*



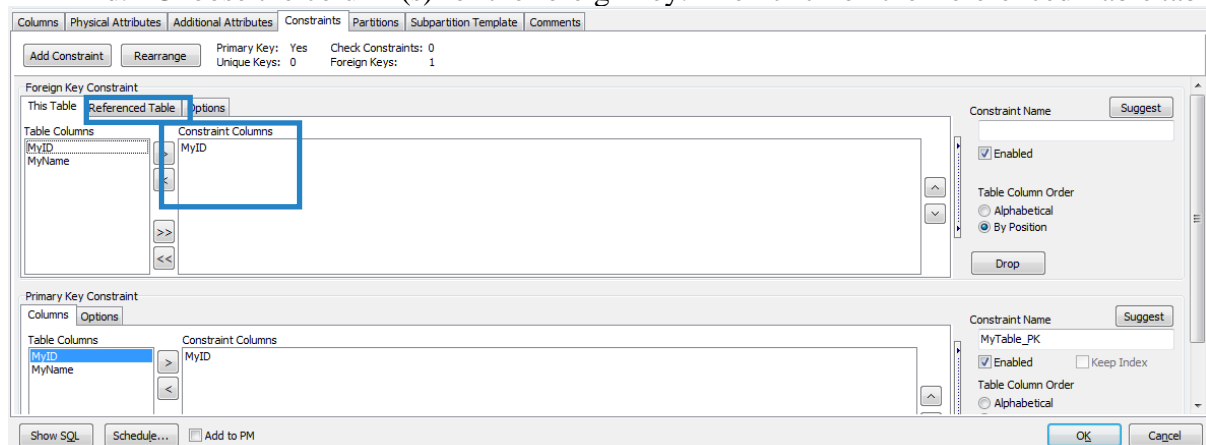
**Figure 11 Constraint Tab**

- c. Select *Foreign key* and then press *Ok*



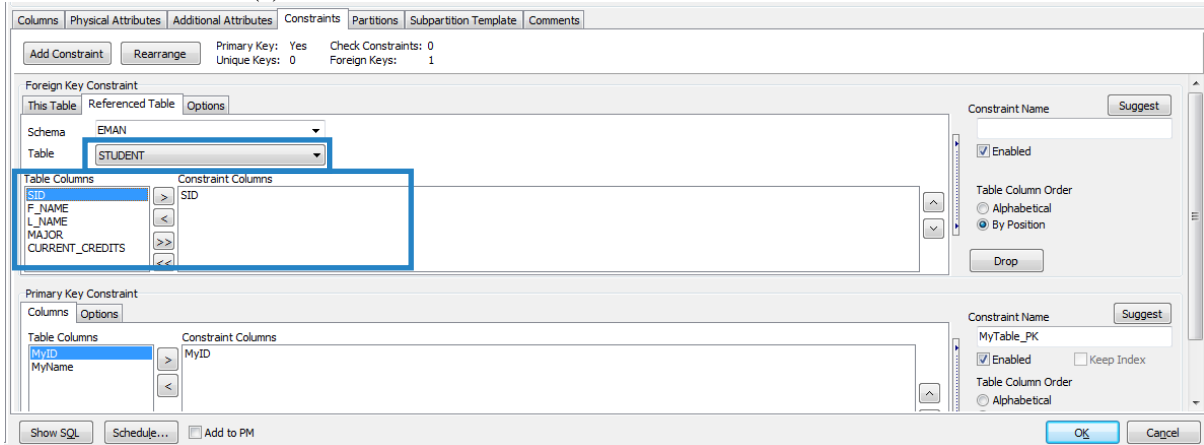
**Figure 12 Add Constraint Options**

- d. Choose the column(s) for the foreign key. Then click on the Referenced Table tab



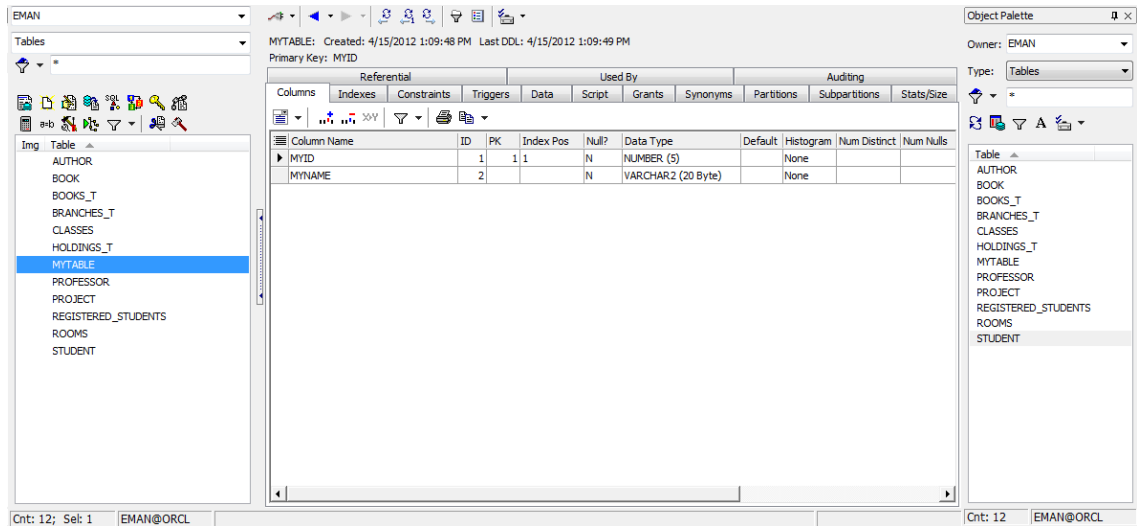
**Figure 13 Adding Foreign Key Constraint**

- e. Select the referenced table from dropdown menu for all tables. Then choose the column(s).



**Figure 14 Adding Foreign Key Constraint - Referencing Table**

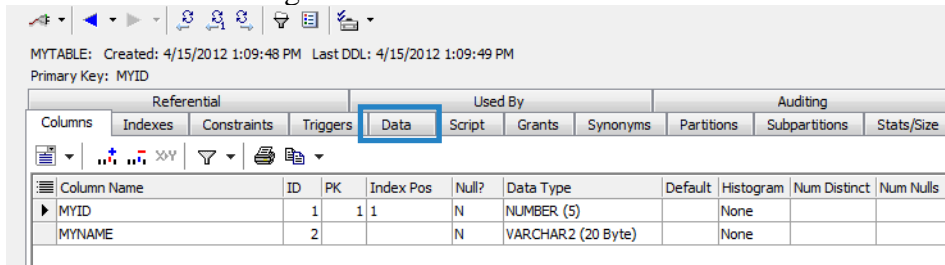
5. Click Ok to create the table with constraints



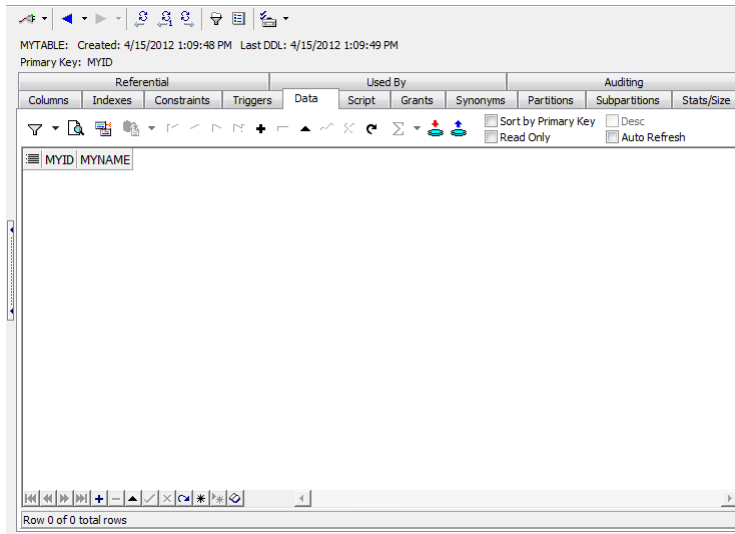
**Figure 15 The new created table in Browser view**

## Insert or Delete Rows

1. On the right hand side of Schema Browser (Object detail), click on Data tab. The Data tab is shown in Figure 16

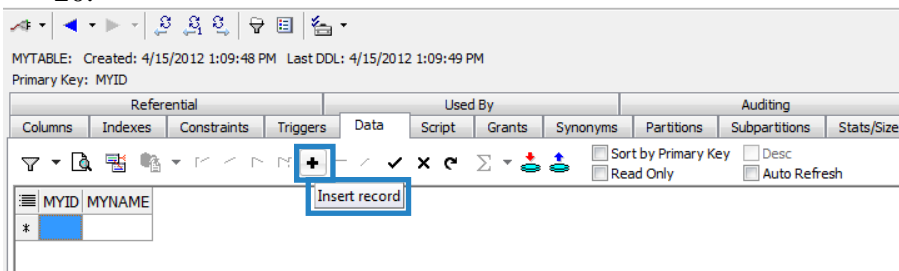


**Figure 17 Object Detail**

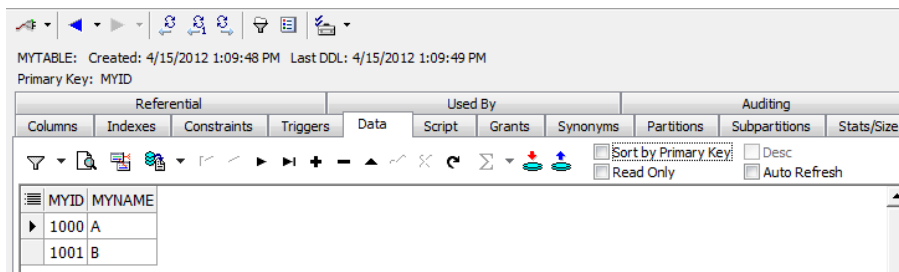


**Figure 18 Data Tab**

2. To Insert record, click on the + icon as shown below, and then write the values for each attribute. To insert another record, repeat the steps. The final records are shown in Figure 20.

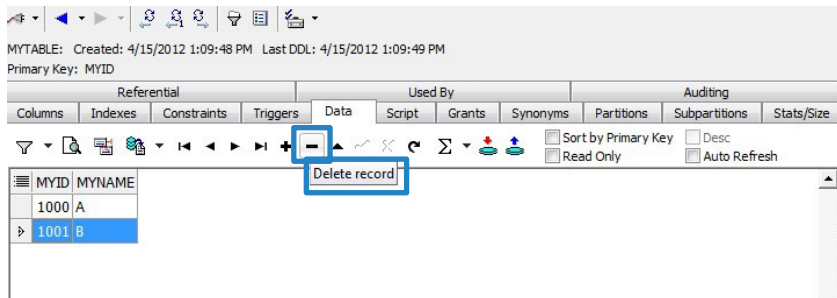


**Figure 19 Insert Record**

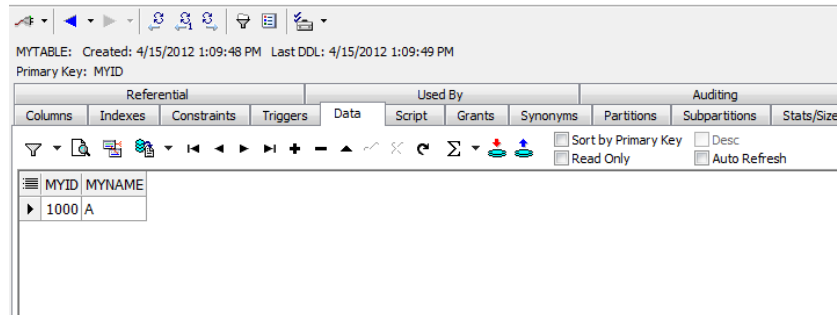


**Figure 20 Data inserted**

3. To delete a record, first you have to select it and then click on the - icon as shown below in Figure 21. Figure 22 shows the record after deleting.



**Figure 21 Deleting a record**



**Figure 22 After record is deleted**



## Lab #7 – Report Builder

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to the basics of Oracle Report Builder

### 2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
  - Create report using report builder

### 3. Introductory Concepts (for this laboratory)

#### What is Report Builder?

a powerful enterprise reporting tool that enables you to rapidly develop and deploy sophisticated Web and paper reports against any data source

---

#### What is Report?

A report consists of objects that collectively define the report:

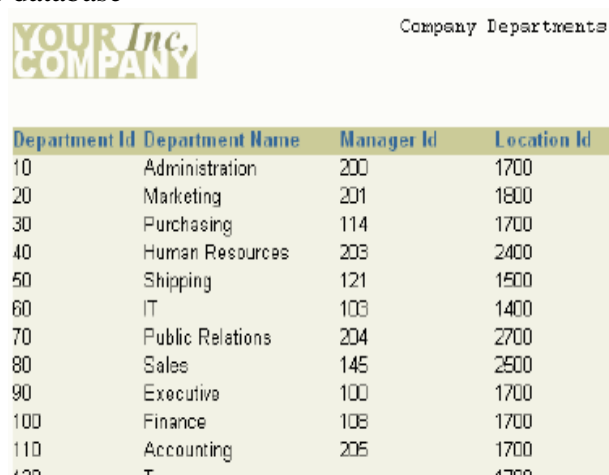
- data model objects (queries, groups, columns, links, user parameters)
- layout objects (repeating frames, frames, fields, boilerplate, anchors)
- parameter form objects (parameters, fields, boilerplate)
- PL/SQL objects (program units, triggers)
- references to external PL/SQL libraries, if any
- code shown in the Web Source view (for JSP-based Web reports)

---

#### Report Styles

##### ○ **tabular reports**

A tabular report is the most basic type of report. Each column corresponds to a column selected from the database



| Department Id | Department Name  | Manager Id | Location Id |
|---------------|------------------|------------|-------------|
| 10            | Administration   | 200        | 1700        |
| 20            | Marketing        | 201        | 1800        |
| 30            | Purchasing       | 114        | 1700        |
| 40            | Human Resources  | 203        | 2400        |
| 50            | Shipping         | 121        | 1500        |
| 60            | IT               | 109        | 1400        |
| 70            | Public Relations | 204        | 2700        |
| 80            | Sales            | 145        | 2500        |
| 90            | Executive        | 100        | 1700        |
| 100           | Finance          | 108        | 1700        |
| 110           | Accounting       | 205        | 1700        |

##### ○ **group above reports**

A group above report contains multiple groups in its data model. It is a "master/detail" report, where there may be a lot of information in the master group. For every master

group, the related values of the detail group(s) are fetched from the database and are displayed below the master information.

| YOUR <i>INC</i> COMPANY |                   |             |
|-------------------------|-------------------|-------------|
| Sales Rep 7499          |                   |             |
| Custid                  | Dollars           | % of Total: |
| 104                     | \$7160.00         | 90.96%      |
| 107                     | \$710.00          | 9.02%       |
| <b>Total:</b>           | <b>\$7870.00</b>  |             |
| <b>% of Total:</b>      | <b>7.60%</b>      |             |
| Sales Rep 7521          |                   |             |
| Custid                  | Dollars           | % of Total: |
| 106                     | \$9024.40         | 91.25%      |
| 100                     | \$764.00          | 7.73%       |
| 101                     | \$101.40          | 1.03%       |
| <b>Total:</b>           | <b>\$9889.80</b>  |             |
| <b>% of Total:</b>      | <b>9.55%</b>      |             |
| Sales Rep 7654          |                   |             |
| Custid                  | Dollars           | % of Total: |
| 102                     | \$27775.50        | 100.00%     |
| <b>Total:</b>           | <b>\$27775.50</b> |             |
| <b>% of Total:</b>      | <b>26.81%</b>     |             |

### ○ Group left report

- A group left report also contains multiple groups in its data model, dividing the rows of a table based on a common value in one of the columns.
- Use this type of report to restrict a column from repeating the same value several times while values of related columns change.
- The data model for group above and group left reports is the same, but the layouts differ; group above reports display the master information at the top while group left reports display break columns to the side.

| YOUR <i>INC</i> COMPANY |               |                 |
|-------------------------|---------------|-----------------|
| Customer Id             | Order Id      | Order Total     |
| 101                     | 2458          | 78279.6         |
|                         | 2447          | 33893.6         |
|                         | 2430          | 29669.9         |
|                         | 2413          | 48662           |
|                         | <b>Total:</b> | <b>190395.1</b> |
| 102                     | 2397          | 42269.2         |
|                         | 2432          | 10523           |
|                         | 2414          | 10794.6         |
|                         | 2431          | 6610.6          |
|                         | <b>Total:</b> | <b>69211.4</b>  |
| 103                     | 2454          | 6853.4          |
|                         | 2433          | 78              |
|                         | 2437          | 13650           |
|                         | 2416          | 310             |
|                         | <b>Total:</b> | <b>20591.4</b>  |

### ○ Form like report

A form-like report displays one record per page, displaying field values to the right of field labels.

### ○ Form letter report

A form letter report contains database values embedded in boilerplate text



## ○ Mailing label report

A mailing label report prints mailing labels in multiple columns on each page. Using the Report Wizard, you can specify the format for your mailing labels

Hermann Baer  
Schwanthalerstr. 7031  
Munich, Bavaria 80925

Adam Fripp  
2011 Interiors Blvd  
South San Francisco,

Nancy Greenberg  
2004 Chiarade Rd  
Seattle, Washington 98199

Michael Hartstein  
147 Spadina Ave  
Toronto, Ontario M5

## ○ Matrix report

A matrix (cross-product) report is a cross-tabulation of four groups of data:

- One group of data is displayed across the page.
- One group of data is displayed down the page.
- One group of data is the cross-product, which determines all possible locations where the across and down data relate and places a cell in those locations.
- One group of data is displayed as the "filler" of the cells.

|    | ANALYST   | CLERK     | MANAGER   | PRESIDENT | SALESMAN  |            |
|----|-----------|-----------|-----------|-----------|-----------|------------|
| 10 | \$0.00    | \$1300.00 | \$2450.00 | \$5000.00 | \$0.00    | \$8750.00  |
| 20 | \$6000.00 | \$1900.00 | \$2975.00 | \$0.00    | \$0.00    | \$10875.00 |
| 30 | \$0.00    | \$950.00  | \$2850.00 | \$0.00    | \$5600.00 | \$9400.00  |
|    | \$6000.00 | \$4150.00 | \$8275.00 | \$5000.00 | \$5600.00 | \$29025.00 |

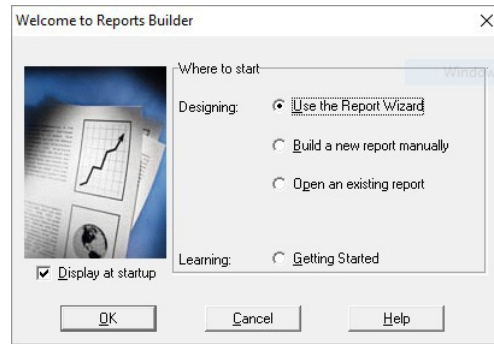
### Example

*Suppose we want a report that displays professors' names and salary and all courses they teach*

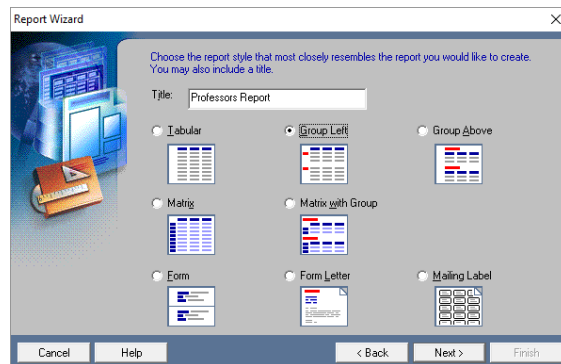
*In addition to the total number of courses each professor teach*

### How to Create Report Using The Wizard?

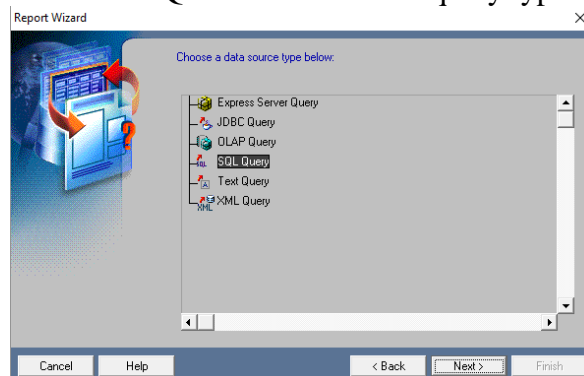
1. Launch Reports Builder



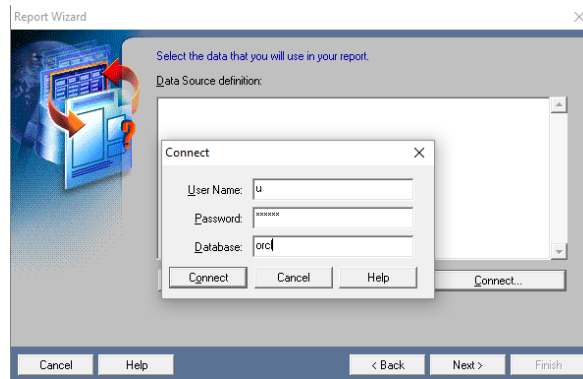
2. Select “Use the Report Wizard” and click OK
3. If the Welcome page displays, click **Next**
4. On the Report Type page, select **Create Paper Layout Only**, then click **Next**.
5. Type the title of the report you want in the *title bar*.
6. Choose the format you want your report to be showed with from the formats listed below the title bar.



7. Click on next
8. On the next window choose SQL-Statement as the query type and click on next.



9. Click on **connect** to get the connect pop on window. Type the user name, password and database and click on connect



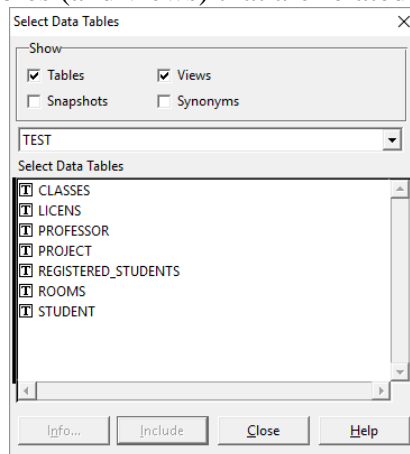
10. The program will verify the user name and password and connect to the specified database.

*If you have the query you want to run the report from, stored in an .sql file, then click on **Import SQL query** button and brows for your file.*

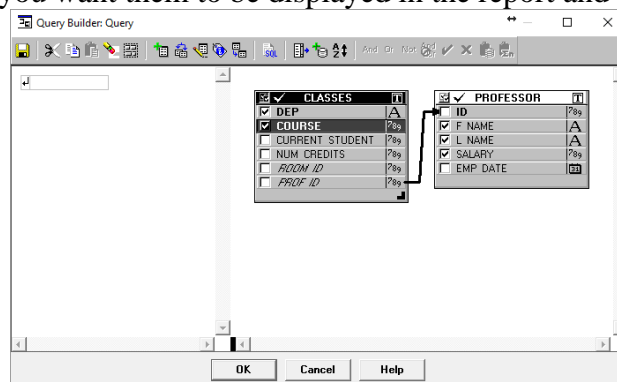
*If you want to build up a new query, then click on the **Query builder**.*

**Note:** remember that you can always write you own code in the *Query Statement Text Area*

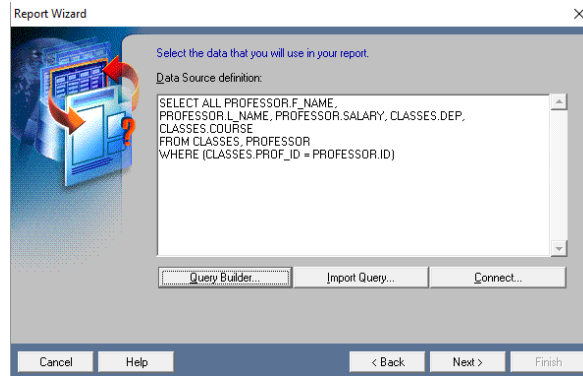
11. Once you click on query builder, a pop up window showing all your tables and views will be shown, include the tables (and views) that are related in your query.



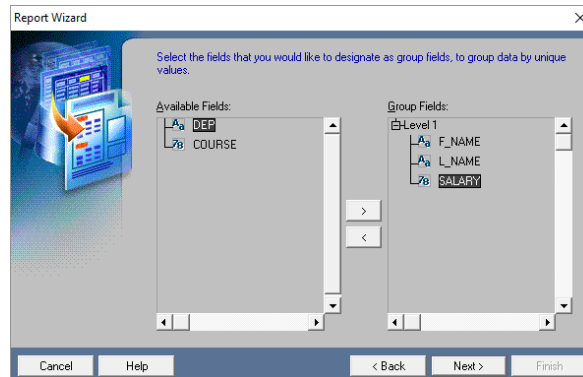
12. Check all fields you want them to be displayed in the report and click **OK**.



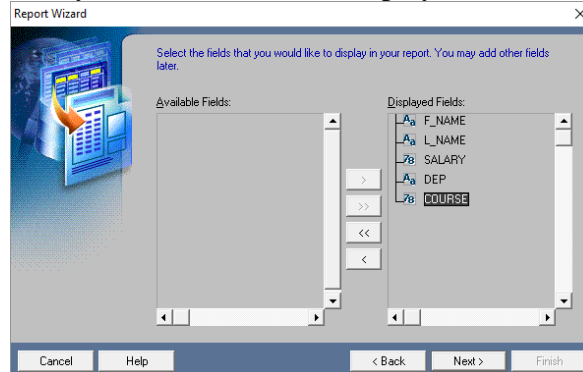
13. The query that relate all tables and fields will appear in the query statement box



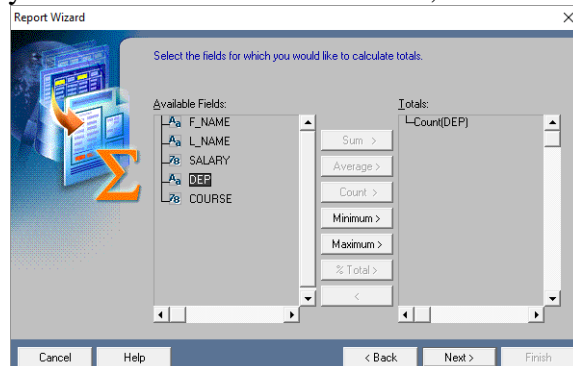
14. Click **next**, then choose the group fields by which output data is grouped and then click **next**



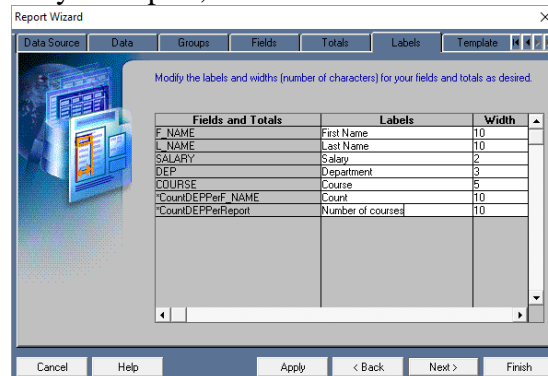
15. Now include the fields you want them to be displayed and click **Next**



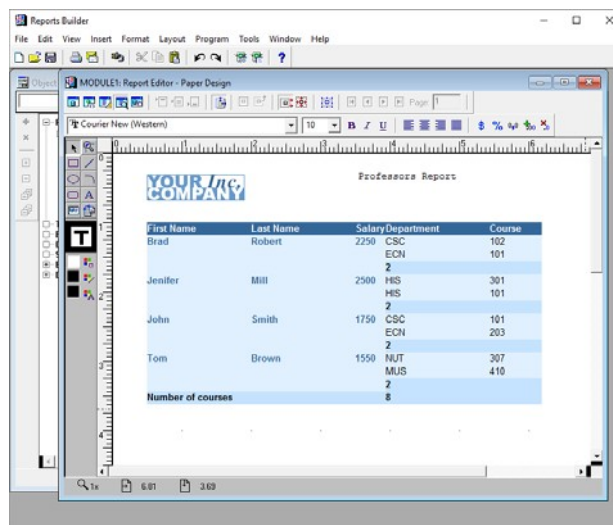
16. This screen is to include any aggregate function needed in your report,  
 17. Click on the field you want to have the function on, and then choose the function



18. This screen enables you to modify either the labels or the size of the displayed fields. If there are any modifications, then do them and click **Next**
19. Choose a template for your report, then click **Finish**.

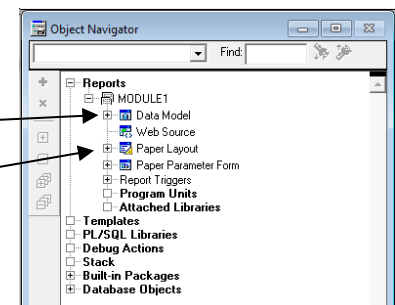


The resultant report is the following:



### Controlling Report

- Shows all related queries, parameters, groups, etc...
- To control the layout of the report



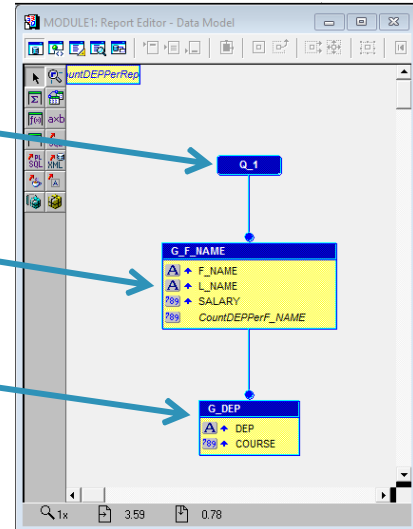
## Data Model

To view or update any object property, double click on it

Query controlling the report

Main group

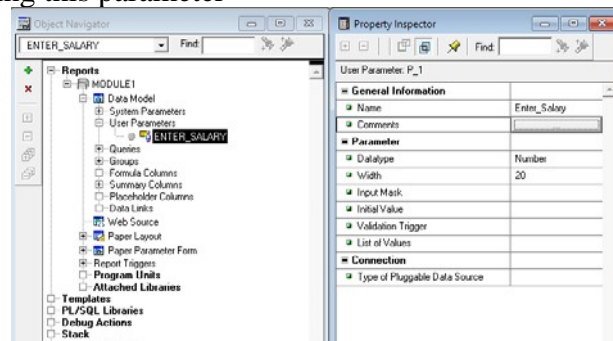
Extended group



**User parameters:** to take an input parameter from the user and return the related output.

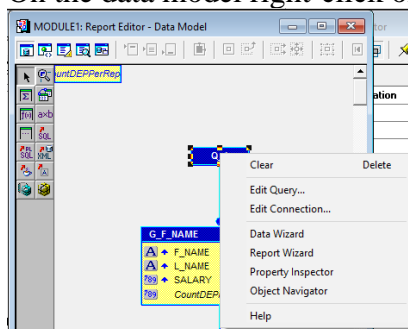
**How to create new user parameter?**

- double click on user parameters, a new parameter will be listed under it.
- double click on this parameter
- You can change the name, the type or the size of it.
- To complete the process, you have to update your query to include the condition including this parameter



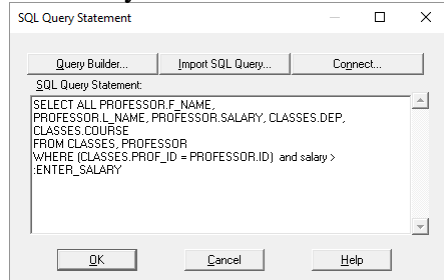
How to use the parameter?

- Change the name of the parameter (optional) ... say you named it “ENTER\_SALARY”
- On the data model right-click on Q\_1 and select edit query





- Add the following to the query
  - and salary > :ENTER\_SALARY



- Run the report by clicking on

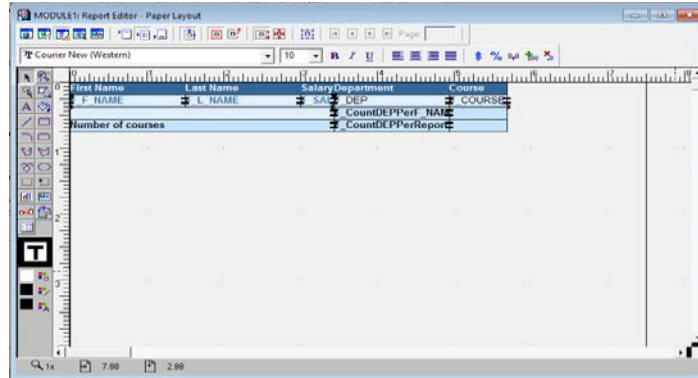


### Output

| First Name        | Last Name | Salary | Department | Course |
|-------------------|-----------|--------|------------|--------|
| Brad              | Robert    | 2250   | ECN        | 101    |
|                   |           |        | CSC        | 102    |
|                   |           | 2      |            |        |
| Jenifer           | Mill      | 2500   | HIS        | 301    |
|                   |           |        | HIS        | 101    |
|                   |           | 2      |            |        |
|                   |           |        |            |        |
| Number of courses |           | 4      |            |        |

### Layout Model

- Control the layout of the report

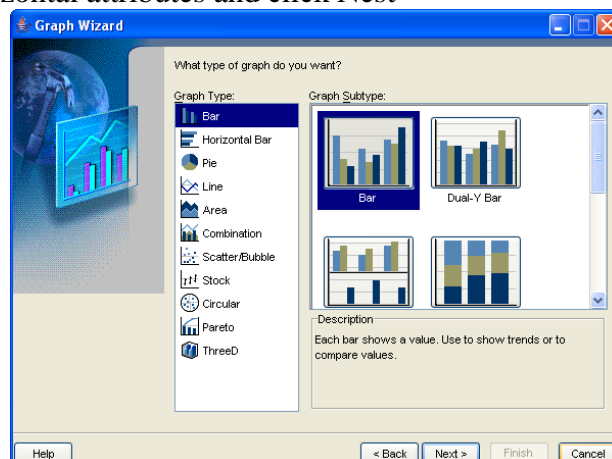


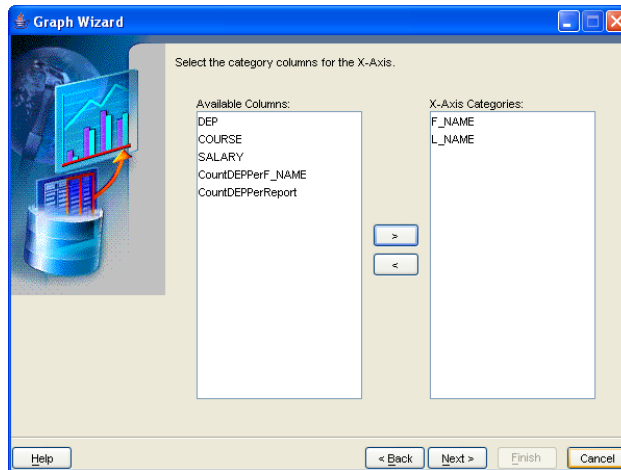
### Graph Wizard

- provides an easy way for you to add a wide variety of graphs to report
- **Usage notes**
  - When you specify a graph title, subtitle, footnote, or axis title in the Graph Wizard, you can insert lexical references (to user parameters, system parameters, and columns) in the text that will display their value at runtime
  - When you specify dates in your graphs, the date values may display in a different format in a graph than in other fields in your report output
- **Primary Graph Types**
  - Bar graph, line graph, area graph, pie graph, funnel graph, combination graph

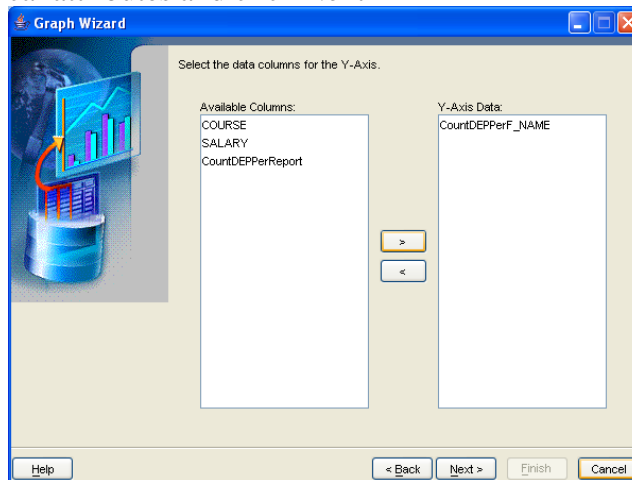
### Adding Graph to Report

- In the Paper Layout view, click the Graph tool in the tool palette.
- Drag a square in the area where the graph should appear to display the Graph Wizard.
- In the Graph Wizard, specify information for the graph. Click **Help on any wizard** page for assistance
- Click Next
- Select the graph type and click Next
- Include the horizontal attributes and click Nest

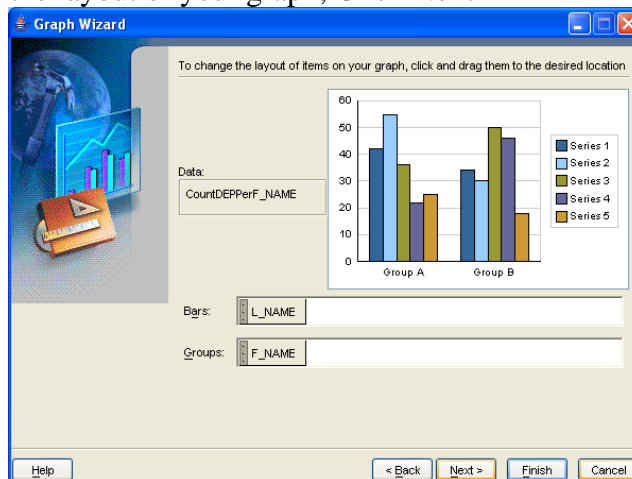




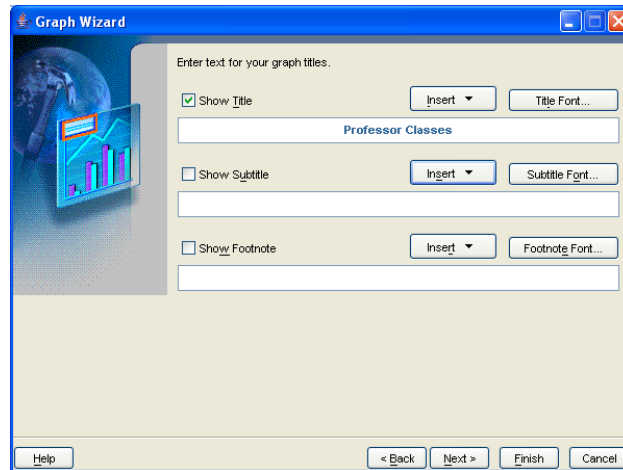
- Specify the vertical attributes and click Next



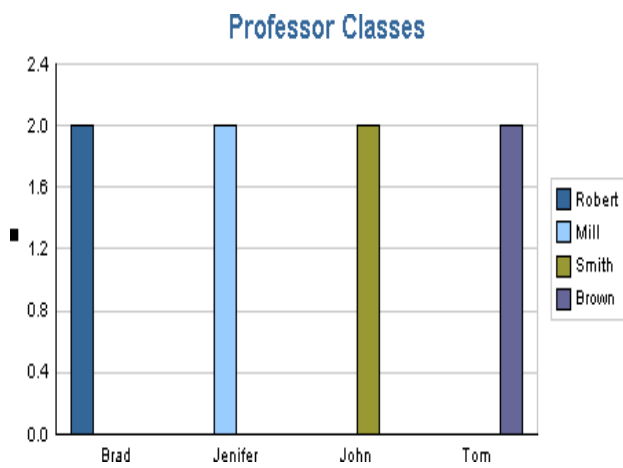
- You can change the layout of your graph, Click Next



- Include the title, subtitle and footnote then click Finish



○ Graph output:



## Lab #8 – Create Simple Oracle Form

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to Oracle Forms Builder.

### 2. Laboratory Learning Outcomes:

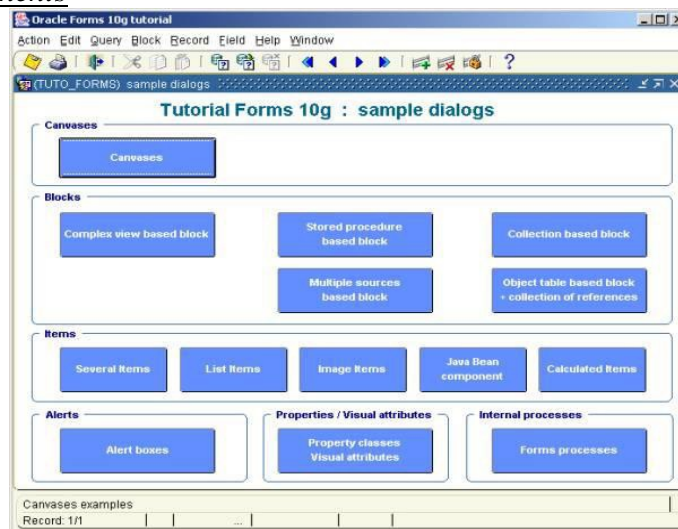
- After completing this lesson, you should be able to do the following:
  - Create simple form

### 3. Introductory Concepts (for this laboratory)

#### What is Form Builder?

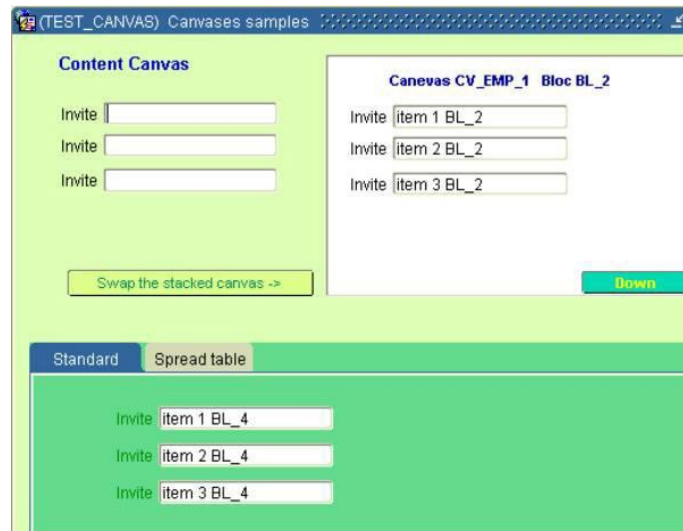
- Used to develop form-based applications for presenting and manipulating data in a variety of ways.
- With Form Builder you can:
  - Provide an interface for users to insert, delete, update and query data
  - Present data as text, image, and custom controls
  - Control forms across several windows and database transactions
  - Use integrated menus
  - Send data to Oracle Reports

#### Form Builder Components



- Canvases are background objects on which you place the interface objects and graphic elements that end users interact with when they use a Form Builder application.
- Blocks are logical containers for Form Builder items, and are the basic unit of information in a form module. A form module typically contains multiple blocks.
- Items are interface objects that display information to end users and allow them to interact with the application.

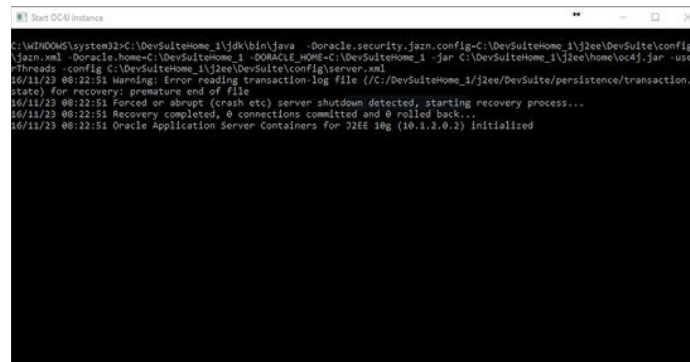
## Canvases



- **Content Canvas: (light green)**
  - Most common canvas type
  - Contains the content pane of the window
  - Must define at least one content canvas for each window you create.
- **Stacked Canvas: (white)**
  - Displayed on the top of the content canvas assigned to the current window.
  - You can display more than one stacked canvas in a window at the same time.
- **Tab Canvas: (dark green)**
  - Made up of one or more tab pages
  - Like stacked canvases, tab canvases are displayed on top of a content canvas

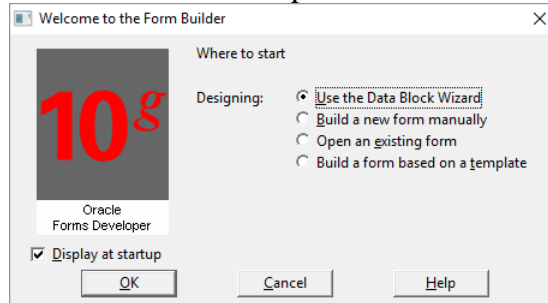
## Getting Started

- Execute the patch file
- The batch file executed in a separate window, you can minimize it if desired

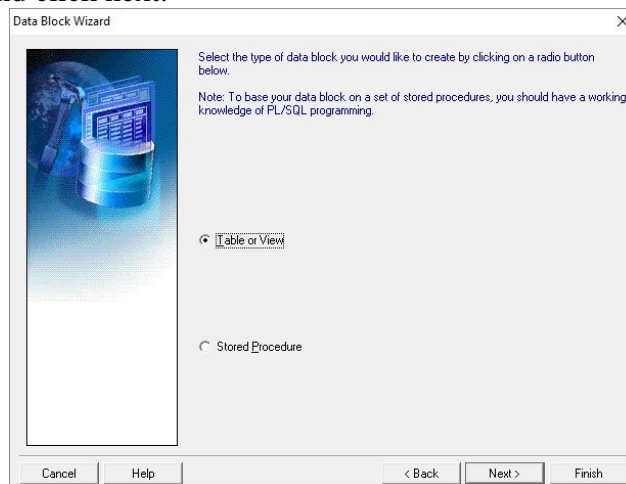


### Creating a New Form Module

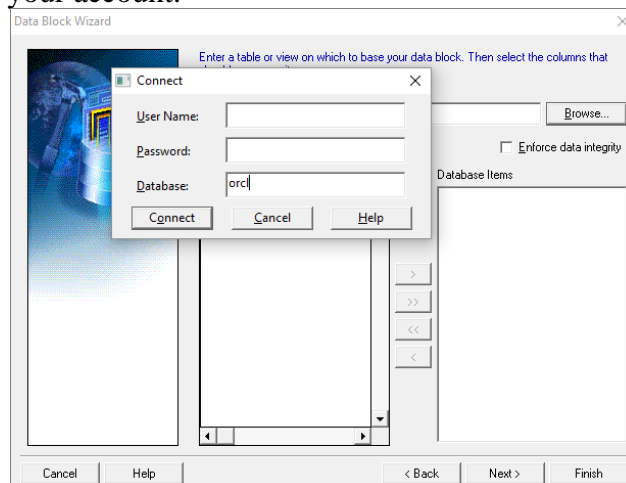
- Invoke the Form Builder. This will take you to the Form Builder Welcome page
- Select the “Use the Data Block Wizard” option



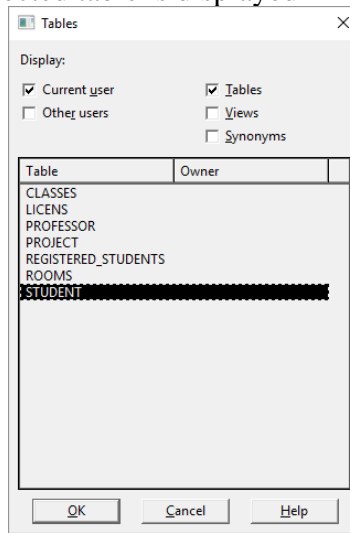
- Click Next
- Creating a New Form Module
- The Type page asks you if you want the form to be built on table or procedure, choose table or view, and click next.



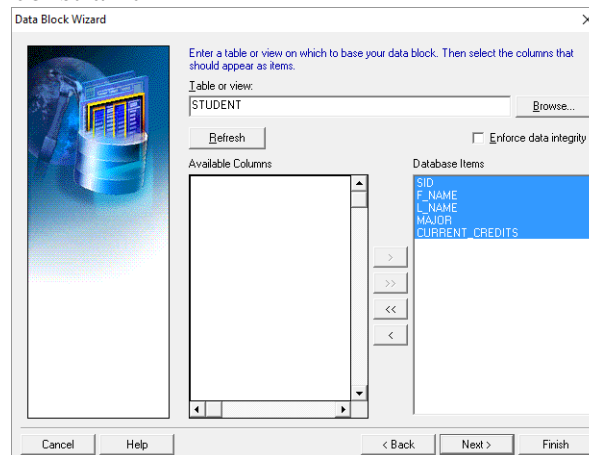
- To choose the table of view you want, click on Browse, and the connection window will show up. After connecting to the database, you will have the list of tables and views that are available in your account.



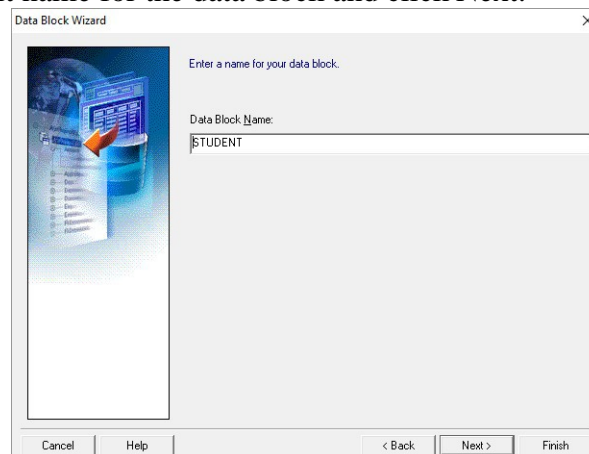
- Select the table for the data source name, then click Ok
- A list of columns in the selected table is displayed



- Select the columns you want to include in the data block.
- Select the “enforce Data integrity” check box if you want the wizard to enforce the database integrity constraint

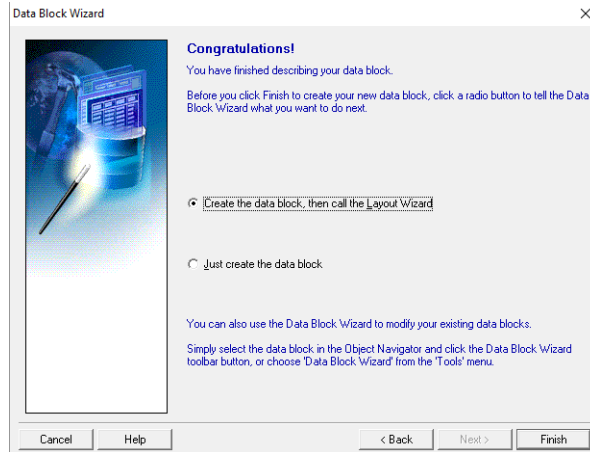


- Accept the default name for the data block and click Next.

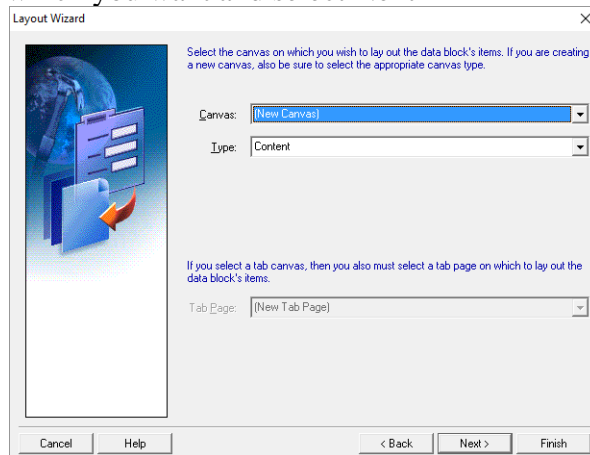


- Select the “Create the data block, then call the Layout Wizard”

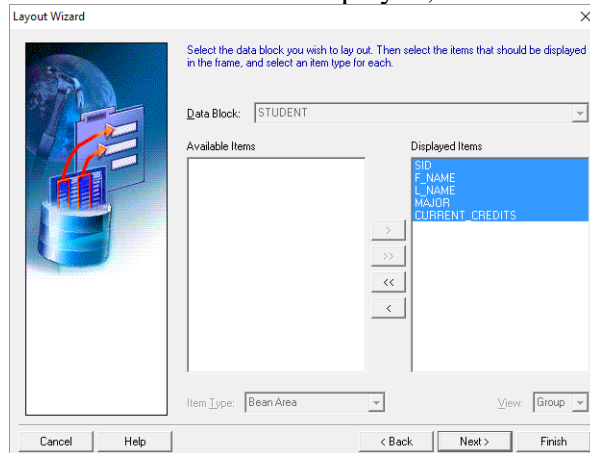




- Select the canvas which you want and select Next



- Choose the data base fields needed to be displayed, and select the type for each



- Change the width and name displayed fields if needed

Enter a prompt, width, and height for each item. The units for item width and height are Points.

| Name            | Prompt          | Width | Height |
|-----------------|-----------------|-------|--------|
| SID             | Student ID      | 83    | 16     |
| F_NAME          | First Name      | 132   | 16     |
| L_NAME          | Last Name       | 132   | 16     |
| MAJOR           | Major           | 33    | 16     |
| CURRENT_CREDITS | Current Credits | 41    | 16     |

Buttons: Cancel, Help, < Back, Next >, Finish

- Select a layout style for your frame. The options are:
  - Form (*usually used to create single-record data blocks*)
  - Tabular (*usually used to create multirecord data blocks*)

Select a layout style for your frame by clicking a radio button below.

☒ Form

☐ Tabular

Buttons: Cancel, Help, < Back, Next >, Finish

- Enter a title in the Frame Title field
- Enter the number of records that you want to display at run time in the Records Displayed field
- You can select the Display Scrollbar check box to display a scroll bar next to the frame (common for multi data blocks)

Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

Frame Title:

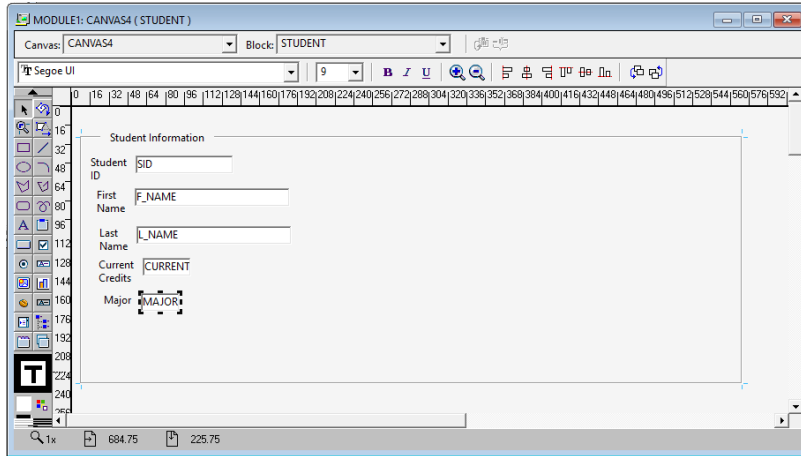
Records Displayed:

Distance Between Records:

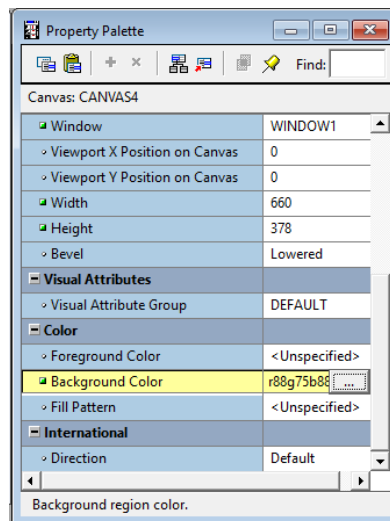
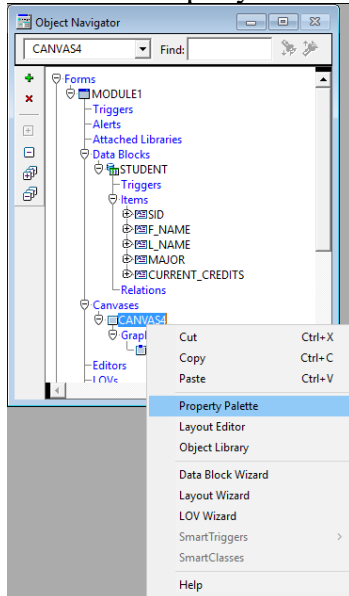
☐ Display Scrollbar

Buttons: Cancel, Help, < Back, Next >, Finish

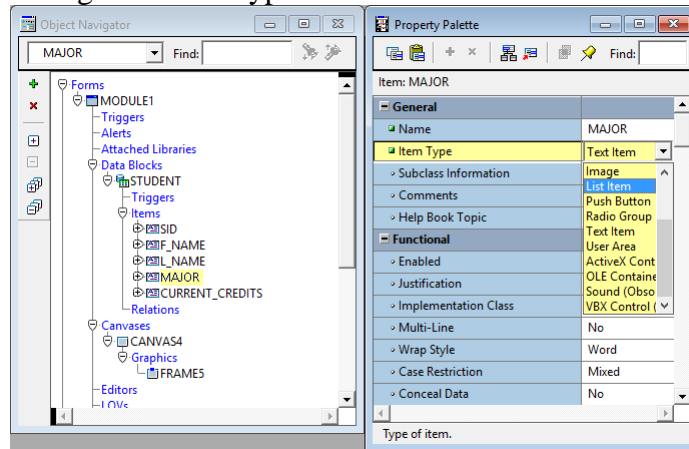
- Finish page. Click Finish to create a new frame and lay out the selected items for the new data block.



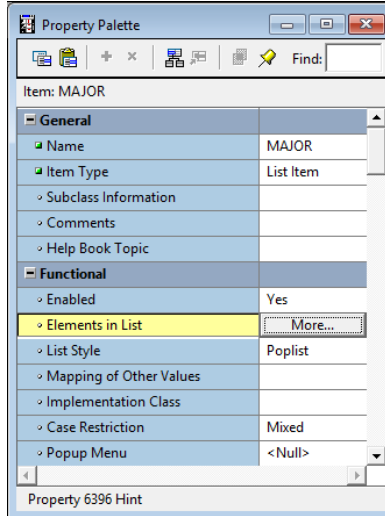
- You can change the colors of the items you have in the canvas by right click on Canvas and select Property Palette



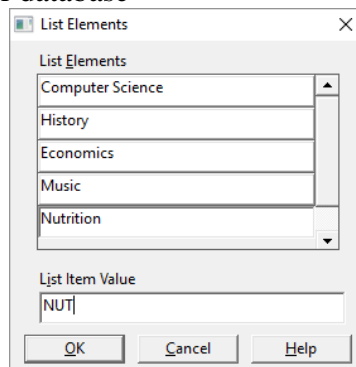
- **Customize the form:** Create **drop down list** for the major.
  1. Right click on major item and choose property palette
  2. Change the Item Type to be *List Item*



3. Click on elements on list and choose more

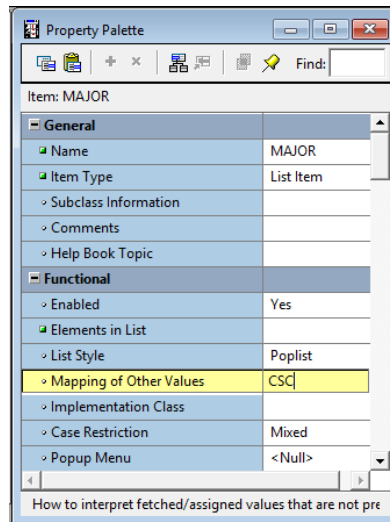


4. Write the values that will appear in the list and its corresponding values you want to deal with in your database



| List Item Value | List Elements    |
|-----------------|------------------|
| CSC             | Computer Science |
| HIS             | History          |
| ECN             | Economics        |
| MUS             | Music            |
| NUT             | Nutrition        |

5. Write the default value of Major in Mapping of Other Values

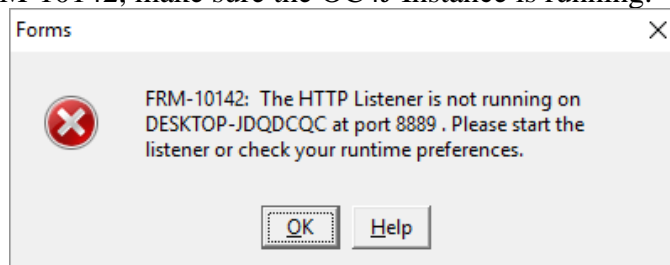


### Compile and Running



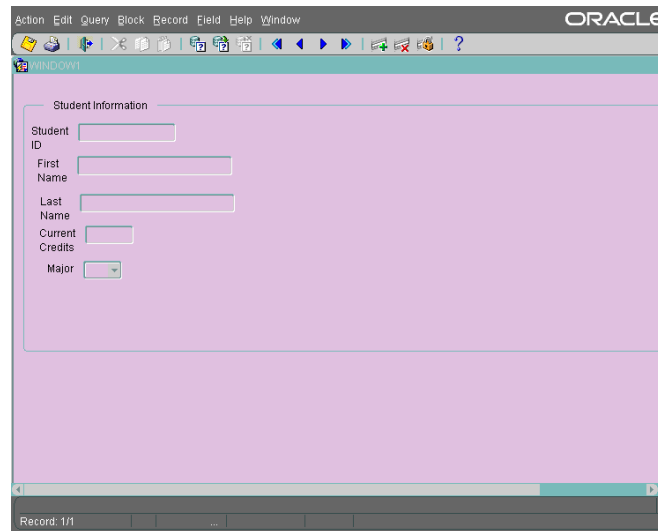
Click on the compile icon or program > CompileModule

If you see error FRM-10142, make sure the OC4J Instance is running.



Click on the run icon or program > RunForm

## The output



---

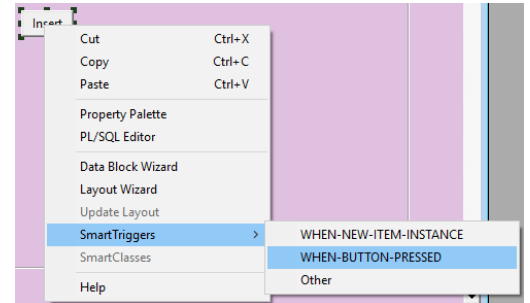
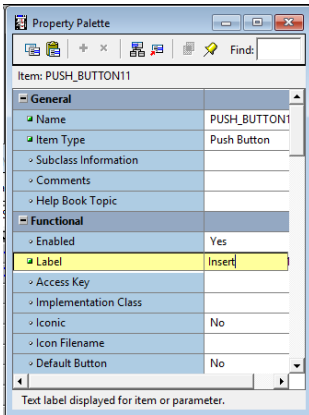
## Built-in Procedures

- You can customize your form by adding procedures. These procedures can be used in buttons
- COMMIT\_FORM: used to save the changes (insert)
- CLEAR\_FORM(parameter1, parameter2): used to clear contents in form
  - parameter1 can be: ASK\_COMMIT, DO\_COMMIT, OR NO\_COMMIT
  - parameter2 can be: TO\_SAVEPOINT, FULL\_ROLLBACK, OR NO\_ROLLBACK
  - Example:
    - CLEAR\_FORM;
    - CLEAR\_FORM(DO\_COMMIT);
    - CLEAR\_FORM(ASK\_COMMIT, TO\_SAVEPOINT);
- EXIT\_FORM(parameter1, parameter2): used to exit the form. The parameters of EXIT\_FORM procedure are similar to CLEAR\_FORM
- DELETE\_RECORD: to delete current record
- Check the help for built-in procedures

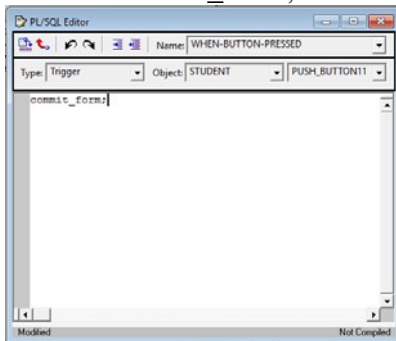
---

## Dealing with Buttons

- Click on the buttons icon
- Create a new button
- Change the name and label using property palette.



- Right click the button
- Choose smart triggers then choose WHEN-BUTTON-PRESSED
- Write “commit\_form;” built in procedure



#### 4. Laboratory Exercises

- Add two more buttons, one is to clear the form and the other to exit.
- You should have the resultant form like the one shown in the following figure

Student Information

Student ID:

First Name:

Last Name:

Major:

Current Credits:

Buttons: Insert, New, Exit

## Lab #9 – Adding Interactive Items to Oracle Form

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to Oracle Forms Builder.

### 2. Laboratory Learning Outcomes:

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

- Understanding form run time window
- Creating List of Values (LOV)
- Create check item
- Creating radio group
- Creating non-input item (read-only text box)

### 3. Introductory Concepts

#### Blocks

Master and Detail Blocks:

- You can define a data block as detail (child) of a master (parent) data block.
- In the Create relationships screen, click on create relationship → then choose the master data block
- The join condition will be created automatically (note that if you have the check box cleared you should choose the joined attributes manually)
- Examples of this relationship include:
  - A Customer Order with many Order Items.
  - A Department with many Employees.
  - An Employee with many Dependents.
  - A Company with many Branch Offices.


#### Lab Example:

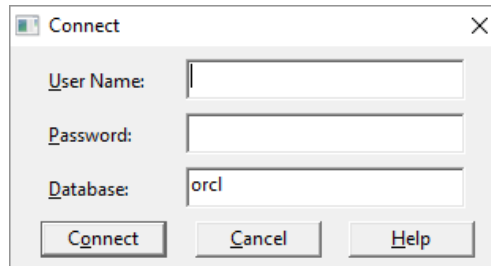
We are going to complete on the previous lab example. Suppose you want to add another data block to display student's registered classes. The result form should be as shown in the picture below.

The screenshot displays an Oracle Forms application window titled 'ORACLE'. The form is divided into two main sections. The top section, 'Student Information', is a master block containing several input fields: 'Student ID', 'First Name', 'Last Name', 'Current Credits', and 'Major'. To the right of these fields are four buttons: 'Insert', 'New', 'Delete', and 'Exit'. The bottom section, 'Student Classes', is a detail block represented as a table with three columns: 'Department', 'Course', and 'Grade'. The table is currently empty. At the bottom of the window, a status bar shows 'Record 1/1'.



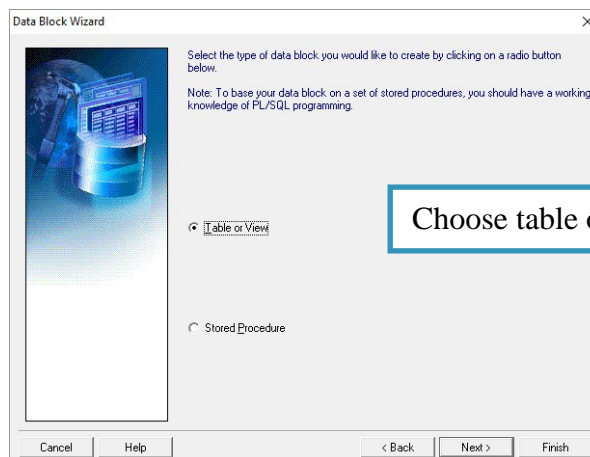
### Creating Another Data Block Steps

1. Open the form created in the previous lab
2. From the toolbar click Connect. 
3. Enter your username and password.



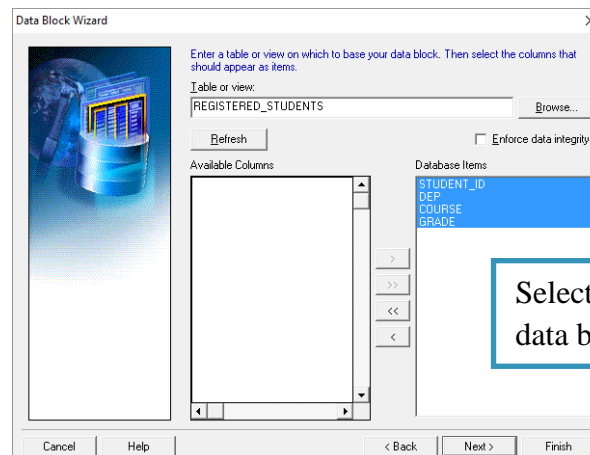
The 'Connect' dialog box has a title bar with a green icon and a close button. It contains three text input fields: 'User Name:', 'Password:', and 'Database:'. The 'Database' field contains the text 'orcl'. At the bottom are three buttons: 'Connect', 'Cancel', and 'Help'.

4. Go to tools → Data Block wizard. Create a new data block for registered students table.



The 'Data Block Wizard' dialog box, Step 1, has a title bar with a close button. It contains a large text area with instructions: 'Select the type of data block you would like to create by clicking on a radio button below.' and a note: 'Note: To base your data block on a set of stored procedures, you should have a working knowledge of PL/SQL programming.' There are two radio buttons: 'Table or View' (selected) and 'Stored Procedure'. At the bottom are buttons: 'Cancel', 'Help', '< Back', 'Next >', and 'Finish'.

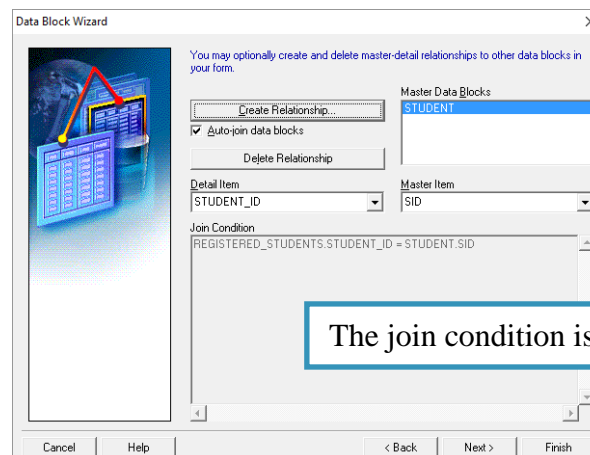
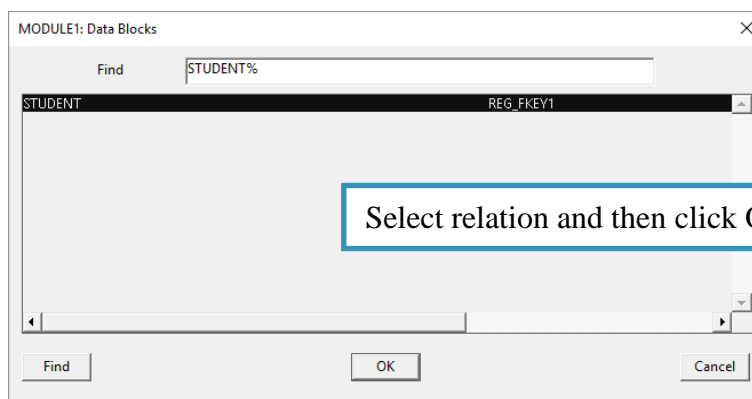
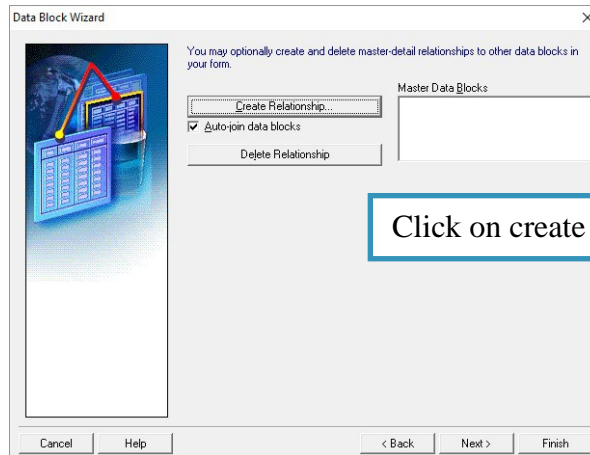
Choose table or view, and click next.



The 'Data Block Wizard' dialog box, Step 2, has a title bar with a close button. It contains a text area with instructions: 'Enter a table or view on which to base your data block. Then select the columns that should appear as items.' There is a text input field for 'Table or view:' containing 'REGISTERED\_STUDENTS' and a 'Browse...' button. Below this is a 'Refresh' button. There are two panes: 'Available Columns' (empty) and 'Database Items' (containing 'STUDENT\_ID', 'DEP', 'COURSE', and 'GRADE'). Between the panes are navigation buttons: '>', '>>', '<<', and '<'. At the bottom are buttons: 'Cancel', 'Help', '< Back', 'Next >', and 'Finish'.

Choose the  
REGISTERED\_STUDENTS table

Select all columns to be included in the  
data block. Click next.



Data Block Wizard

Enter a name for your data block.

Data Block Name:

Student\_Classes

Cancel Help < Back Next > Finish

Name the block as “Student\_Classes”, then click Next.

Data Block Wizard

**Congratulations!**

You have finished describing your data block.

Before you click Finish to create your new data block, click a radio button to tell the Data Block Wizard what you want to do next.

☒ Create the data block; then call the Layout Wizard

☐ Just create the data block.

You can also use the Data Block Wizard to modify your existing data blocks.

Simply select the data block in the Object Navigator and click the Data Block Wizard toolbar button, or choose 'Data Block Wizard' from the 'Tools' menu.

Cancel Help < Back Next > Finish

Select the “Create the data block, then call the Layout Wizard”

Layout Wizard

Select the canvas on which you wish to lay out the data block's items. If you are creating a new canvas, also be sure to select the appropriate canvas type.

Canvas: CANVAS4

Type: Content

If you select a tab canvas, then you also must select a tab page on which to lay out the data block's items.

Tab Page: (New Tab Page)

Cancel Help < Back Next > Finish

Select CANVAS4 then click Next

Layout Wizard

Select the data block you wish to lay out. Then select the items that should be displayed in the frame, and select an item type for each.

Data Block: STUDENT\_CLASSES

Available Items

STUDENT ID

Displayed Items

DEP  
COURSE  
GRADE

Item Type: Bean Area

View: Group

Cancel Help < Back Next > Finish

Choose the fields needed to be displayed, then click Next.

Layout Wizard

Enter a prompt, width, and height for each item. The units for item width and height are Points.

| Name   | Prompt     | Width | Height |
|--------|------------|-------|--------|
| DEP    | Department | 33    | 16     |
| COURSE | Course     | 41    | 16     |
| GRADE  | Grade      | 17    | 16     |

Cancel Help < Back Next > Finish

Change the width and name displayed fields if needed

Layout Wizard

Select a layout style for your frame by clicking a radio button below.

☐ Form

☒ Tabular

Cancel Help < Back Next > Finish

Select Tabular layout, then click Next.

Layout Wizard

Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

Frame Title:

Records Displayed:

Distance Between Records:

☐ Display Scrollbar

Cancel Help < Back Next > Finish

- Enter a title in the Frame Title field
- Enter the number of records that you want to display at run time in the Records Displayed field

5. You should have the form as shown below

Oracle Forms Designer Window

Student Information

Student ID:

First Name:

Last Name:

Current Credits:

Major:

Insert

New

Delete

Exit

Student Classes

| Department | Course | Grade |
|------------|--------|-------|
|            |        |       |
|            |        |       |
|            |        |       |

Record: 1/1

6. Now, press F7, then go to execute query and you will see the students with his classes shown.

Oracle Forms Designer Window

Student Information

Student ID:

First Name:

Last Name:

Current Credits:

Major:

Insert

New

Delete

Exit

Student Classes

| Department | Course | Grade |
|------------|--------|-------|
| CSG        | 102    | A     |
| HIS        | 101    | A     |
|            |        |       |
|            |        |       |

Print

Record: 1/2

### Form Runtime Window

Mode of operation

- Enter Query Mode
  - Normal Mode
- 

### Enter Query Mode

- Used to enter search criteria
  - It allows:
    - Retrieve all records
    - Retrieve records by using selection criteria
    - Retrieve records by using Query/Where Dialog Box.
    - Obtain the number of records from database
  - It does not allow:
    - navigate out the current block
    - exit from the run-time session
    - go to next record
    - Insert new records
    - update existing records
    - delete records
- 

### Normal Mode

- Used to insert and alter records in the database
- It allows:
  - retrieve all records
  - insert new records
  - update records
  - delete records
  - commit and rollback
  - navigate outside the current block
  - exit from run-time session
- It does not allow:
  - retrieve a restricted set of records
  - Invoke the query/where dialog box

Enter Query      Execute Query      Cancel

Student Information

Student ID: 1000

First Name: Sara

Last Name: Jassem

Current Credits: 120

Major: C...

Buttons: Insert, New, Delete, Exit

Student Classes

| Department | Course | Grade |
|------------|--------|-------|
| CSC        | 102    | A     |
| HIS        | 101    | A     |
|            |        |       |

Print  
Record: 1/2

○ To run a query:

- Click the Enter Query button
- Enter the condition you want in the related field
- Click Execute query

○ To Cancel a query in an enter query mode:

- Click on Cancel Query button

○ To insert a record:

- Enter the data you want (make sure you are not in enter query mode)
- Click insert button

○ To update a record:

- Search for the record you want using query enter mode
- Update the record

○ To delete a record:

- Search for the record you want using query enter mode
- Click delete button

○ Discard changes:

- Go to action → clear all

- Navigate between records:
  - previous block, previous record, next record and next block respectively



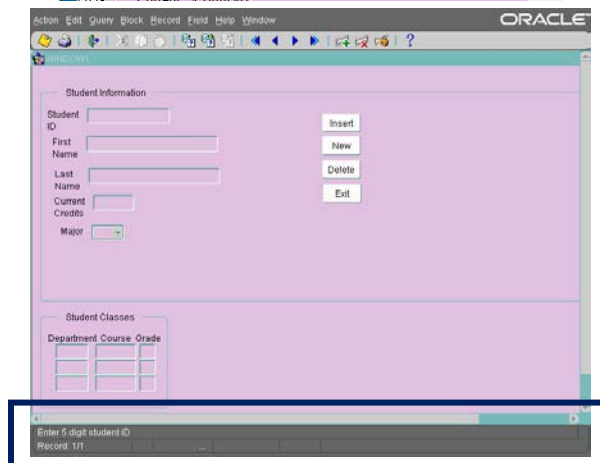
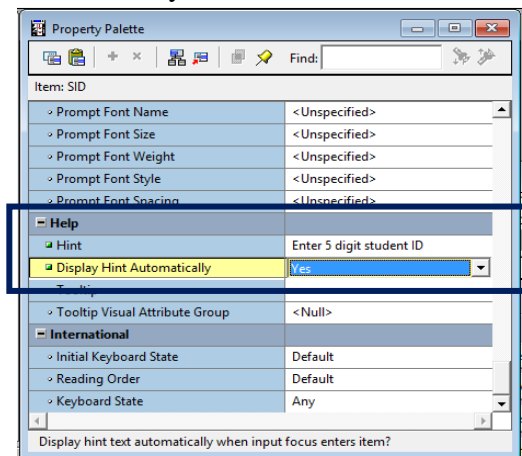
## Blocks

- Data Blocks:
  - Associated with a database table, view or a stored procedure.
  - Its items related to database columns
- Control Blocks:
  - Not associated with a database object
  - Its items called control items;
  - They control the functionalities of the form
  - You should create it manually NOT using the data block wizard

---

## Including Helpful Messages

- In the item properties window, go to help property that has the following features:
  - Hint: write your help message
  - Display Hint automatically: set it to YES.



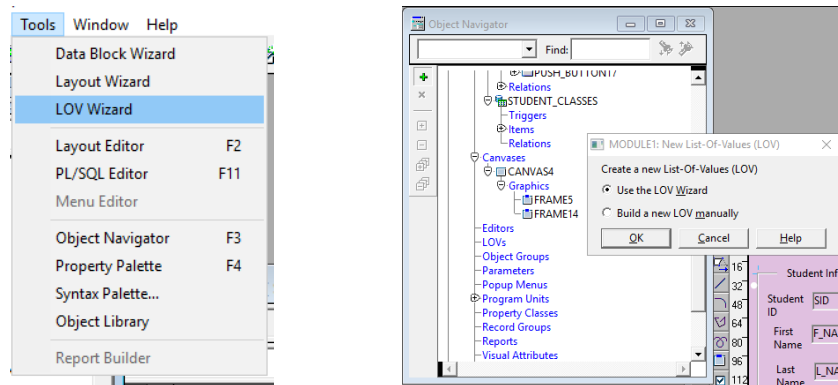


## Creating LOVs

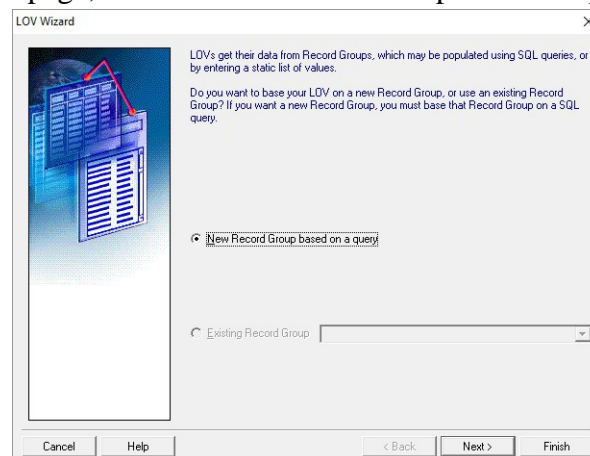
- LOVs (List Of Values) are objects in form modules that each open their own window when activated at run time.
- Defined at form level → Used to support text items in any block in form module
- Include data from database
- LOVs have the following qualities:
  - Dynamic: takes data from database
  - Flexible: used same LOV for many text items
  - Independent: Not dependent on specific text item
  - Efficient: reuse data loaded to form

## Creating an LOV with the LOV Wizard

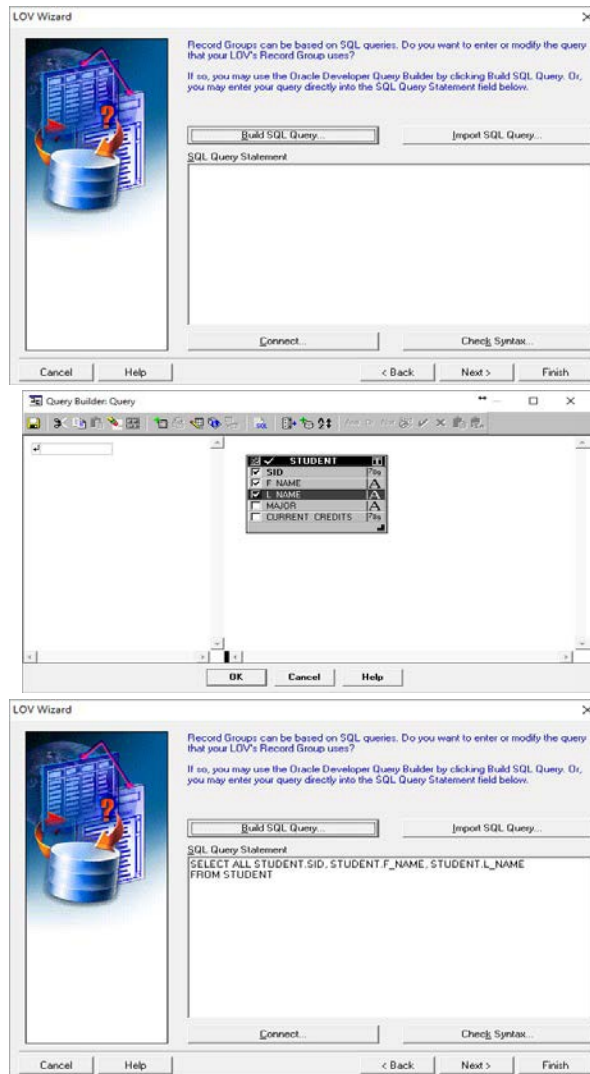
- Launch the LOV Wizard



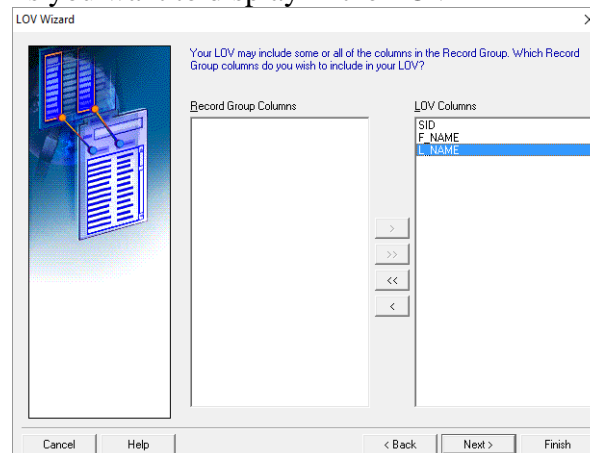
- If the welcome screen appears. Click Next
- In the LOV source page, choose New Record Group based on query. Click Next



- Click Build SQL Query to use Query Builder



## ○ Choose the columns you want to display in the LOV



## ○ Change the size and label of LOV columns if needed

LOV Wizard

If you wish to specify the LOV column properties, you may enter a title, width and return value for each LOV column. The units for the column width is Points.

| Column | Title      | Width | Return value |
|--------|------------|-------|--------------|
| SID    | Student Id | 83    |              |
| F_NAME | First Name | 132   |              |
| L_NAME | Last Name  | 132   |              |

☐ Automatically size columns

Look up return item...

Cancel Help < Back Next > Finish

Items and Parameters

Find STUDENT%

STUDENT.SID  
STUDENT.F\_NAME  
STUDENT.L\_NAME  
STUDENT.MAJOR  
STUDENT.CURRENT\_CREDITS  
STUDENT.PUSH\_BUTTON11  
STUDENT.ITEM15  
STUDENT.ITEM16  
STUDENT.PUSH\_BUTTON17  
STUDENT\_CLASSES.STUDENT\_ID  
STUDENT\_CLASSES.DEP  
STUDENT\_CLASSES.COURSE  
STUDENT\_CLASSES.GRADE

Find OK Cancel

LOV Wizard

If you wish to specify the LOV column properties, you may enter a title, width and return value for each LOV column. The units for the column width is Points.

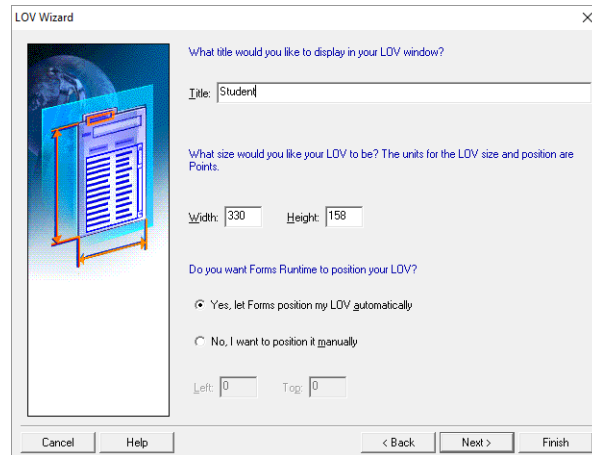
| Column | Title      | Width | Return value   |
|--------|------------|-------|----------------|
| SID    | Student Id | 83    | STUDENT.SID    |
| F_NAME | First Name | 132   | STUDENT.F_NAME |
| L_NAME | Last Name  | 132   |                |

☐ Automatically size columns

Look up return item...

Cancel Help < Back Next > Finish

- Specify the title, the width, and the height of the LOV window.
- You can choose to display it at a set position that manually define, or let Forms position it automatically. Click Next



LOV Wizard

What title would you like to display in your LOV window?

Title:

What size would you like your LOV to be? The units for the LOV size and position are Points.

Width:  Height:

Do you want Forms Runtime to position your LOV?

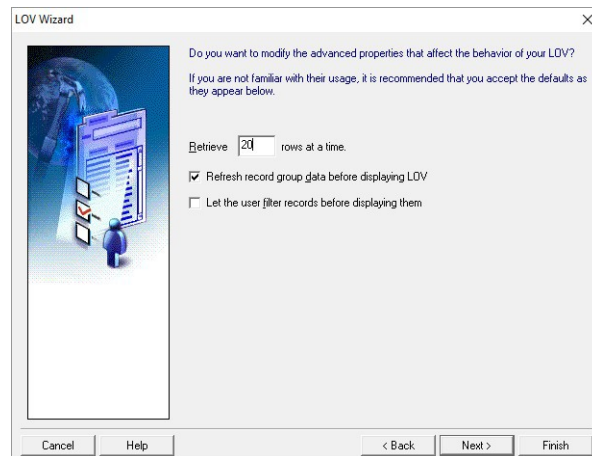
☒ Yes, let Forms position my LOV automatically

☐ No, I want to position it manually

Left:  Top:

Cancel Help < Back Next > Finish

- Specify the number of records at a time to be fetched from the database



LOV Wizard

Do you want to modify the advanced properties that affect the behavior of your LOV?

If you are not familiar with their usage, it is recommended that you accept the defaults as they appear below.

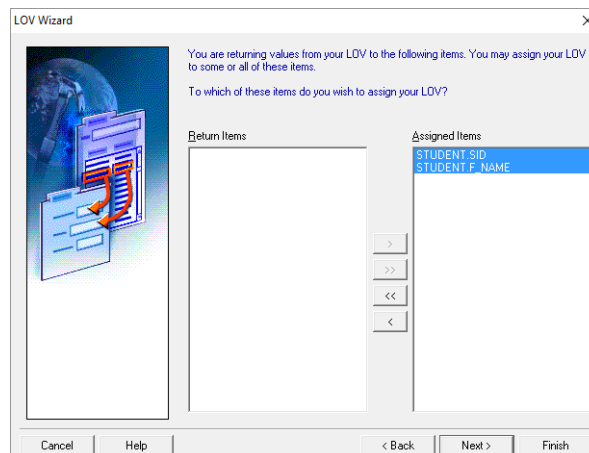
Retrieve  rows at a time.

☒ Refresh record group data before displaying LOV

☐ Let the user filter records before displaying them

Cancel Help < Back Next > Finish

- Assign the LOV with form items



LOV Wizard

You are returning values from your LOV to the following items. You may assign your LOV to some or all of these items.

To which of these items do you wish to assign your LOV?

Return Items

Assigned Items

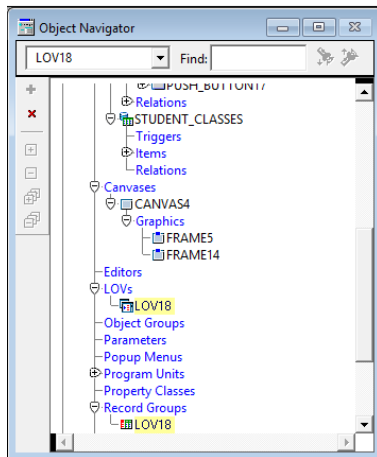
STUDENT SID  
STUDENT F NAME

> >> << <

Cancel Help < Back Next > Finish

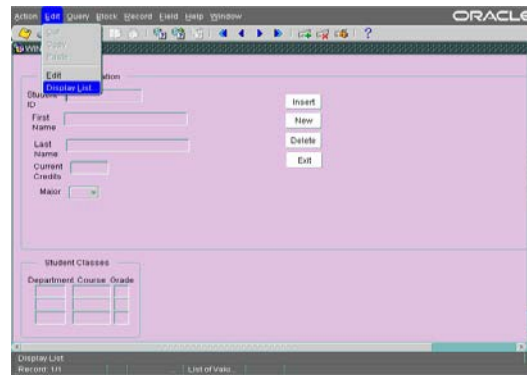
- Click Finish to complete the LOV creation process

In the figure below, the LOV has been created. A default name of LOV18 was given to both the LOV and to its associated record group. Depending on what other parts you have completed, the default name may be slightly different.

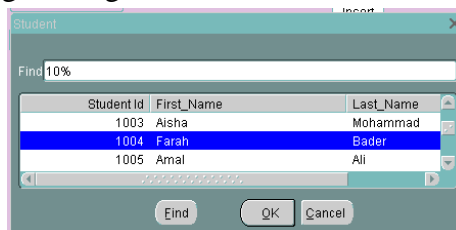


### How To Use LOV At Run Time

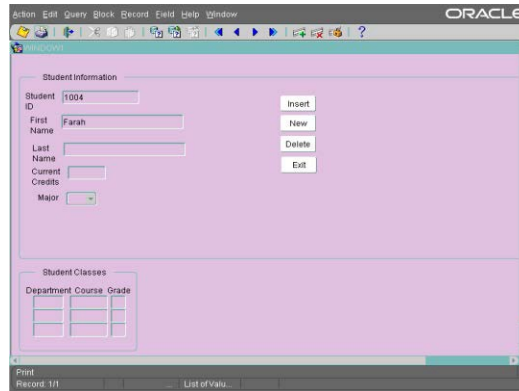
- Place your mouse cursor in the text item associated with the LOV
- Select Edit → Display list of values



- Select an entry from the displayed list, you can type a character to filter the list or you can search the list by typing a string in the find box.



- Click OK to retrieve the value




---

### Associating LOV With A More Text Items

- Select the text item from which the LOV is to be accessible
  - Set the list of values property in the property platter to the required LOV
- 

### Input Items

- Items types that accept user input. Include:
  - Check boxes
  - List items
  - Radio groups
- Enable insert, update, delete and query

### Check Boxes

- Is a two-state interface object that indicates whether a certain value is ON or OFF.
  - The display state of check box is always either checked or unchecked
  - Although it is limited to two states, it is not limited to just two values. You can specify the value to represent Checked, the value to represent Unchecked, and how other values are processed
  - Check box can be created in three ways:
    - Converting an existing item
    - Using the Check Box tool in the Layout Editor
    - Using the Create icon in the Object navigator
  - Important properties to be set:
    - Data Type
    - Label
    - Access Key
    - Initial value
    - Value when checked
    - Value when unchecked
    - Synchronize with item
-

### Radio Groups

- Set of mutually exclusive radio buttons, each representing a value
- Use
  - To display two or more static choices
  - As an alternative to a list item
  - As an alternative to a check box

---

### Creating a Radio Group

- Radio Group can be created in three ways:
  - Converting an existing item
  - Using the radio group item in the Layout Editor
  - Using the Create icon in the Object navigator
- Important properties to be set (for Radio group Property):
  - Data Type
  - Mapping of Other Values
- Important properties to be set (for Radio button Property):
  - Label
  - Radio Button Value

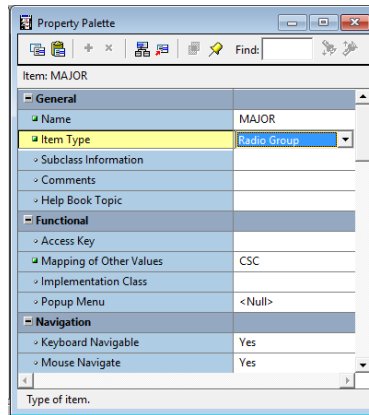
---

Suppose you want to convert Major list into radio group. Also create a frame labeled major containing the radio button for each major.

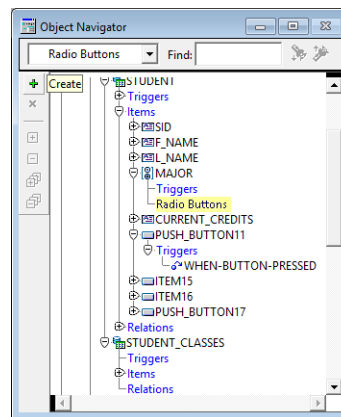
The screenshot shows an Oracle Forms application window. The main form is titled 'Student Information'. It contains several input fields: 'Student ID', 'First Name', 'Last Name', and 'Current Credits'. To the right of these fields are four buttons: 'Insert', 'New', 'Delete', and 'Exit'. Below the 'Current Credits' field is a section labeled 'Major' which contains a radio group. The radio group has five options: 'Nutrition', 'Music', 'Economics', 'History', and 'Computer Science'. The 'Computer Science' radio button is selected. Below the 'Major' section is a section labeled 'Student Classes' which contains a table with three columns: 'Department', 'Course', and 'Grade'. The status bar at the bottom of the window shows 'Enter 5 digit student ID' and 'Record: 1/1'.

### How to Convert an Existing Item into A Radio Group

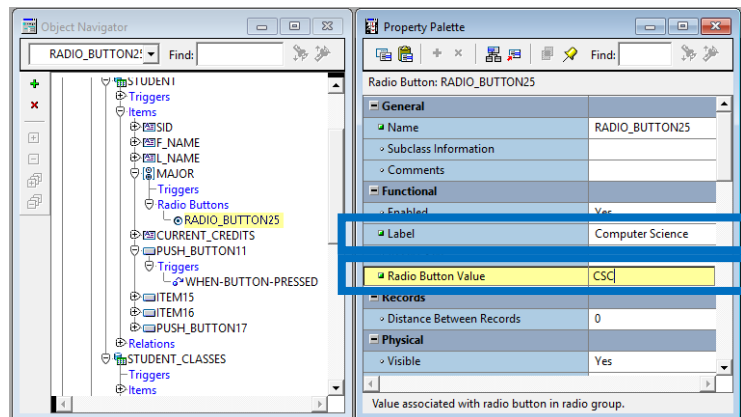
1. Invoke the property palette for the item that you want to convert.
2. Set the Item Type property to Radio Group.

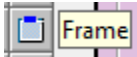


3. Set the Mapping of Other Values property to specify how the Radio Group should handle the other values
4. Expand the item node in the Object Navigator. The Radio Buttons node appears
5. Select the Radio Buttons node and click the Create icon.

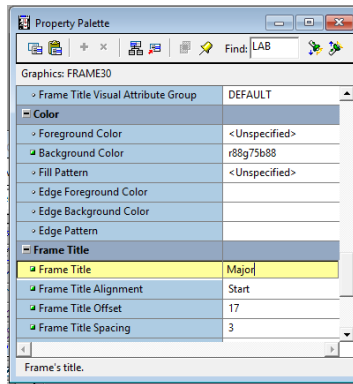


6. Enter a value, and a label for the radio button



7. Create additional radio buttons by repeating steps 4 to 6
8. Open the layout editor
9. Click on the Frame icon 
10. Create new frame
11. Change the label of the frame using property palette.





12. Move the radio button to the frame

13. The final form should be like the figure below

#### 4. Laboratory Exercises

Create a form as it shown in the figure below

The form should contain one data blocks that is used to display the information about a class. You should create the following:

- a LOV for professors Ids, first name, and last name and assign the ID text item with it.
- A radio button group to represent the department
- List of buttons:
  1. *New*: create a new empty record
  2. *Save*: Save the inserted data
  3. *Search*: enables user to enter into enter\_query mode
  4. *Show Results*: Show the result of the entered query
  5. *Exit*: To exit the form

## Lab #10 – Adding Non-Input Items to Oracle Forms

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to Oracle Forms Builder.

### 2. Laboratory Learning Outcomes:

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

- Creating non-input item (read-only text box)
- Creating non-input item (image item, sound item)
- Create calculated items
- Creating a canvas (Stacked Canvas, Tool bar Canvas, Tab Canvas)
- Navigate between forms

### 3. Introductory Concepts (for this laboratory)

#### Create Noninput Items

- Used to enhance application by displaying additional data, often from a database table
- Item types that do not accept direct user input include:
  - Display item
  - Image item
  - Calculated items

---

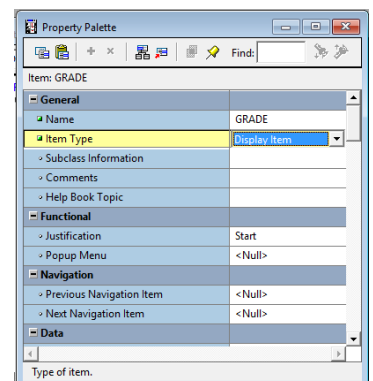
#### Display Items

- Are like a text item, except that they cannot:
  - Be edited
  - Be queried
  - Be navigated to
  - Accept user input
- Can display:
  - Display additional, nonbase table information
  - Display derived data values

---

#### Creating a Display Item

- Using the Item Type property to convert an existing item into a display item
  1. Double click on the item to display the Property Palette
  2. Change the Item Type to be Display Item

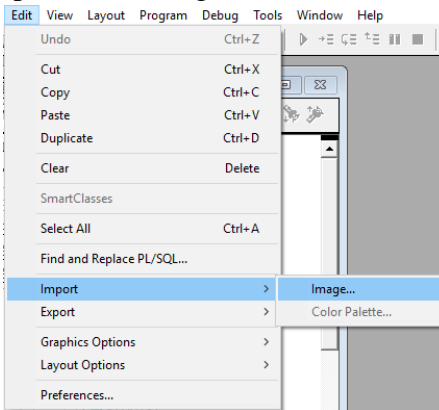


### Image Item

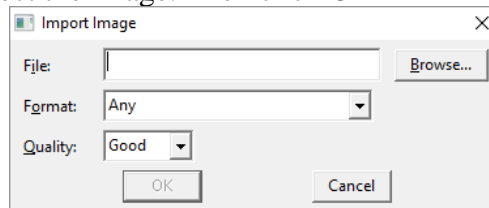
- Is a special interface control that can store and display vector or scanned images.
- Store and display images
- You can store images in
  - Database – long raw
  - File system – any supported file format

### Creating an Image Item

- Open Layout Editor
- From Edit Menu select Import, then Image



- Click Browse and select the image. Then click OK



---

### Calculated Items

- They accept item values that are based on calculations
- They are read-only
- They can be expressed as:
  - Formula: *calculated item is a result of horizontal calculation*
  - Summary: *calculated item is a vertical calculation involving items of single item over all the rows within a single block*
- Generally, Display Items are used as calculated items

---

### Setting Item Properties for the Calculated Item

- Formula
  - Calculation Mode
  - Formula
- Summary
  - Calculation Mode
  - Summary Function
  - Summarized Block
  - Summarized Item

### Rules for Summary Items

- Summary item must be resided in:
    - The same block as the summarized item
    - A control block with Single Record property set to Yes
  - Summarized item must be reside in:
    - A data block with Query All Records property or Precompute Summaries property set to Yes
    - A control block
  - Datatype of summary items must be Number, unless using MAX or MIN
- 

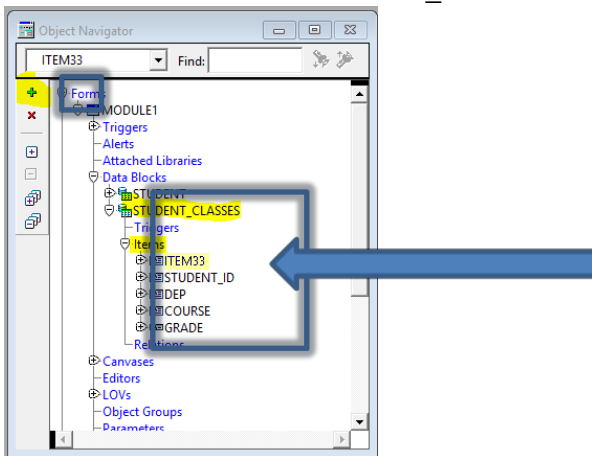
### Creating Calculated Item Based on a Summary

1. Create a new item in the Object Navigator
  2. Open property Palette of an item.
  3. Set the following properties:
    1. Canvas: *Select the canvas in which the item reside*
    2. Item Type: Display Item
    3. Data Type: Number (*\* only if it is not min/max*)
    4. Calculation Mode: Summary
    5. Summary Function: *select the required function*
    6. Summarized Block: *Select a block over which all rows will be summarized*
    7. Summarized Item: *Select item to be summarized*
  4. Open the property Palette for the data block and set Query All Record property to Yes
- 

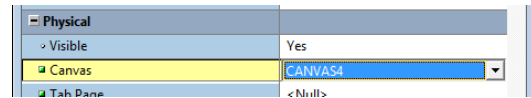
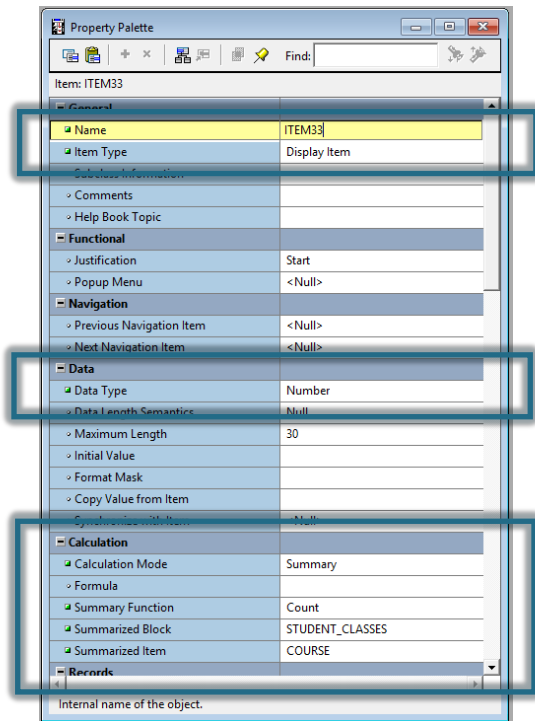
### Calculated Item: Example

Create a calculated item and count the number of courses for a student

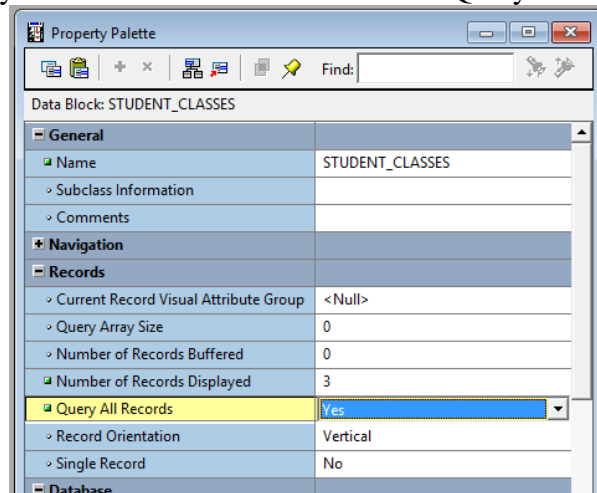
1. Create a new item in Student\_Classes data block



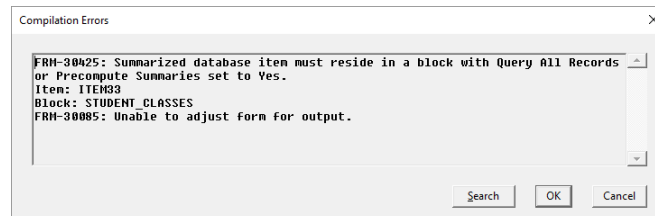
2. Set the required property as shown



3. Open the property Palette for the data block and set Query All Record property to Yes



Note: if you got the following error message, make sure to set Query All Record property to Yes as explained in step 3



Output:

### What Is a Canvas?

- Is a surface inside a window container on which you place visual objects such as interface items and graphics
- Each item in a form must refer to no more than one canvas
- Canvas Type:
  - Content Canvas
  - Stacked Canvas
  - Toolbar Canvas
  - Tab Canvas

---

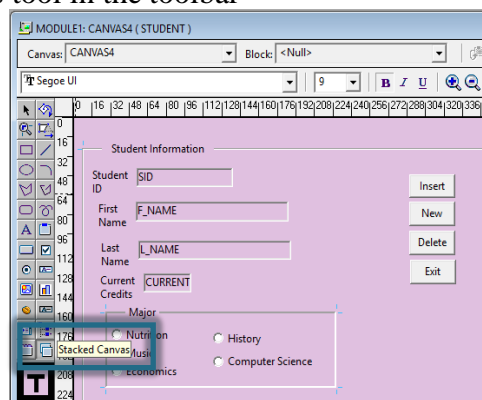
### The Stacked Canvas

- Display on top of a content canvas
- Shares a window with a content canvas
- Size:
  - Usually smaller than the content canvas
  - Determined by viewport size
- Created in:
  - Layout Editor
  - Object Navigator
- With stacked canvases you can achieve the following:
  - Scrolling views as generated by Oracle Designer
  - Creating an overlay effect within a single window
  - Displaying headers with constant information (ex. Company name)
  - Creating a cascading or a revealing effects within a single window
  - Displaying additional information
  - Displaying information conditionally
  - Displaying context-sensitive help
  - Hiding information

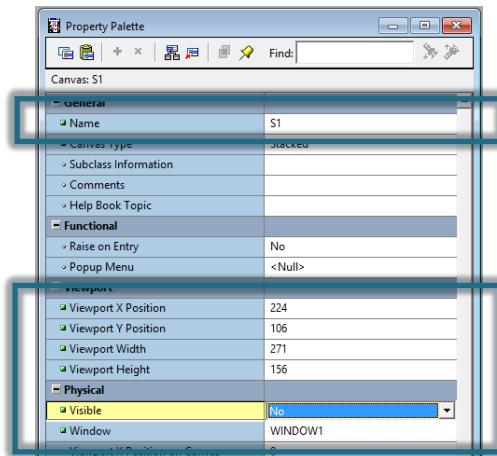
---

### Creating a Stacked Canvas in the Layout Editor

1. Display the Layout Editor for the content canvas on which you wish to create a stacked canvas
2. Click the Stacked Canvas tool in the toolbar



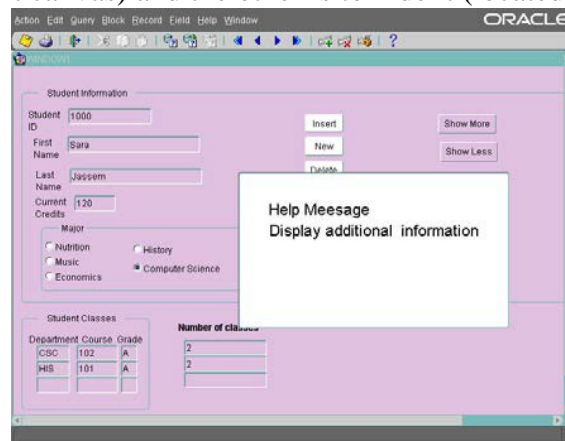
3. Click and drag the mouse in the canvas where you want to position the stacked canvas
4. Open the Property Palette of the stacked canvas. Set the following properties:
  1. Raise on Entry
  2. Viewport X/Y
  3. Viewport width/height
  4. Visible
  5. Window
  6. Show horizontal/vertical scroll bar



5. Add a button to show canvas. → `SHOW_VIEW('S1');`
  6. Add a button to hide the canvas. → `HIDE_VIEW('S1');`
- To show/hide the stack canvas at run time, use `SHOW_VIEW('canvas_name')`/`HIDE_VIEW('canvas_name')` built in procedures.
  - Make sure to set the visible property for the canvas to NO.
  - If you want to view stacked canvas in layout editor, go to View → Stacked View, and select/unselect the canvases you want them to be shown/hidden.
  - NOTE: [Control]+ Click to clear the selection in menu.

### Stacked Canvas Example

- create a help message stack canvas and two buttons; one is for showing the canvas (located on content canvas) and the other is to hide it (located also on the content canvas)



### The Toolbar Canvas

- Special type of canvas for tool items
  - Two types:
    - Vertical toolbar
    - Horizontal toolbar
  - Provide:
    - Standard look and feel
    - Alternative to menu or function key operation
  - Creating Tool bar Canvas:
    - In the object navigator select the canvases icon then click on create button
    - Change the type to your suggested type.
    - Change the attributes you want from the property platter
- 

### The Tab Canvas

- Enables you to organize and display related information on separate tabs
  - Creating Tab Canvas:
    - In layout editor, click on the Tab canvas button then drag the mouse to place the tab canvas in the place you want
    - If you want to view tab canvas in layout editor, go to View → Stacked View, and select/unselect the canvases you want them to be shown/hidden.
    - NOTE: [Control]+ Click to clear the selection in menu.
- 

### Navigate Between Forms

- Use forms built in procedures to navigate between forms
  - CALL\_FORM('form\_path\form\_name');
  - OPEN\_FORM('form\_path\form\_name');



## Lab #11 – Introduction to PL/SQL Programming

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to the basic of PL/SQL programming

### 2. Laboratory Learning Outcomes:

- After completing this lesson, you should be able to do the following:
  - PL/SQL blocks
  - PL/SQL procedures and functions
  - Triggers

### 3. Introductory Concepts (for this laboratory)

#### PL/SQL

- PL/SQL is an Oracle Procedural Language extension to SQL
- The following can be constructed with the PL/SQL language:
  - Anonymous block
  - Stored or standalone procedure and function
  - Package
  - Trigger
- Use PL/SQL to add business logic to a database application

---

#### PL/SQL Block Structure

##### **DECLARE**

/\* Declarative section – PL/SQL variables \*/

##### **BEGIN**

/\* Execution section – SQL statements go here ... only section that is REQUIRED \*/

##### **EXCEPTION**

/\* Exception handling section – error-handling statements go here \*/

##### **END;**

---

#### PL/SQL Declaring Variables

- Syntax:  
**DECLAE**  
**Variable\_name type [CONSTANT] [NOT NULL] [:=value];**
  - Variables can have any SQL datatype, such as VARCHAR2, DATE, or NUMBER, or a PL/SQL-only datatype, such as a BOOLEAN or PLS\_INTEGER.
  - Declaring a constant is similar to declaring a variable except that you must add the CONSTANT keyword and immediately assign a value to the constant.
  - Variable\_name consists of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs  
**Variable\_name      table.column%TYPE;**
  - Creates a variable that has the same type as the column type
-

### Example in Declaring Variables in PL/SQL

```
DECLARE -- declare the variables in this section
 v_last_name VARCHAR2(30);
 v_first_name VARCHAR2(25);
 n_employee_id NUMBER(6);
 b_active_employee BOOLEAN;
 n_monthly_salary NUMBER(6);
 number_of_days_worked NUMBER(2);
 pay_per_day NUMBER(6,2);
 -- a constant variable
 avg_days_worked_month CONSTANT NUMBER(2) := 21;
BEGIN
 NULL; -- NULL statement does nothing, allows this block to be executed and tested
END;
```

---

### To Show Output

- Set the serveroutput variable on:  
Syntax:  
SQL> SET serveroutput ON;
  - Use DBMS\_OUT.PUT\_LINE command:  
Syntax:  
SQL> DBMS\_OUTPUT.PUT\_LINE (variable\_name);
- 

### Example in PL/SQL With Output

```
DECLARE
text1 varchar2(50);
text2 varchar2(50) := 'var2 initially initialized --> ';
var1 number;
var2 number := 5;
BEGIN
 text1 := 'var1 just initialized --> ';
 var1 := 10;
 DBMS_OUTPUT.PUT_LINE ('... ' || text1 || var1);
 DBMS_OUTPUT.PUT_LINE ('... ' || text2 || var2);
END;
```

---

### Interacting with Tables

```
DECLARE
var1 number;
BEGIN
 SELECT number_seats
 INTO var1
 FROM rooms
 WHERE room_id = 99999;
 DBMS_OUTPUT.PUT_LINE (' Number of seats for room number 99999 is ' || var1);
END;
```

---

### PL/SQL Control Structures

- Conditional Control With IF-THEN
  - The forms of the statement can be IF-THEN, IF-THEN-ELSE, or IF-THEN-ELSEIF-ELSE.
  - The IF clause checks a condition
  - The THEN clause defines what to do if the condition is true
  - The ELSE clause defines what to do if the condition is false or null.
- Iterative Control With LOOPS.

---

#### IF-THEN-ELSE: Syntax

```
IF boolean_expresion1 THEN
 sequence_of_statemnts1;
[ELSIF boolean_expreasion2 THEN
 sequence_of_statemnets2;]
...
[ELSE
 sequence_of_statements3;]
END IF;
```

---

#### IF-THEN-ELSE: Example

```
DECLARE
 v_NumberSeats rooms.number_seats%TYPE;
 v_Comment VARCHAR2(35);
BEGIN
 SELECT number_seats
 INTO v_NumberSeats
 FROM rooms
 WHERE room_id = &input;
 IF v_NumberSeats <= 50 THEN
 v_Comment := 'Fairly small';
 ELSIF v_NumberSeats <= 100 THEN
 v_Comment := 'A little bigger';
 ELSE
 v_Comment := 'Lots of room';
 END IF;
 DBMS_OUTPUT.PUT_LINE (v_Comment);
END;
/
```

---

#### While Loop

- Syntax:  
WHILE condition LOOP  
 sequence\_of\_statements;  
END LOOP;
- Example:  
V\_counter := 0;

```
WHILE v_counter < 6 LOOP
 v_counter := v_counter + 1;
END LOOP;
```

---

### For Loop

- Syntax:  
FOR loop\_counter IN [REVERSE] low\_bound .. High\_bound LOOP  
 sequence\_of\_statements;  
END LOOP;
  - Example 1:  
FOR v\_counter2 IN 1..5 LOOP  
 DBMS\_OUTPUT.PUT\_LINE(v\_counter2);  
END LOOP;
  - Example 2:  
FOR v\_counter2 IN REVERSE 1..5 LOOP  
 DBMS\_OUTPUT.PUT\_LINE(v\_counter2);  
END LOOP;
- 

### Procedures

```
CREATE [OR REPLACE] PROCEDURE procedure_name (parameter1 mode type,
parameter2 mode type, ...) AS
/*Declarative section is here*/
BEGIN
/*Executable section is here*/
EXCEPTION
/* Exception section is here*/
END [procedure name];
```

---

### Procedures: Example

```
create or replace procedure update_room_capacity(
 r_room_id in rooms.room_id%TYPE,
 r_factor in NUMBER)
AS
 r_room_count INTEGER;
BEGIN
select count(*)
into r_room_count
from rooms
where room_id=r_room_id;
```

---

### Calling a Procedure

- insert into rooms  
values(111,'Building 7',100,100);
- call update\_room\_capacity(111,1.5);  
Call completed.
- select \* from rooms;

### Procedure

- Dropping a procedure  
DROP procedure update\_room\_capacity;
  - Show errors in a procedure  
Show errors
- 

### Functions

```
CREATE OR REPLACE FUNCTION function_name (parameter1 mode type,
parameter2 mode type,)
RETURN return_type AS
/*Declarative section is here*/
BEGIN
/*Executable section is here*/
EXCEPTION
/* Exception section is here*/
END [function name];
```

---

### Functions: Example

```
create function circle_area (
 radius in number
) return number as
 v_pi number := 3.1416;
 v_area number;
begin
 v_area := v_pi * radius * radius;
 return v_area;
end circle_area;
/
```

---

### Functions

- Calling a function  
SQL> select circle\_area(2)  
2 from dual;  
CIRCLE\_AREA(2)  
-----  
12.5664
  - Dropping a function  
SQL> drop function circle\_area;  
Function dropped.
- 

### Trigger

- Is a procedure that is run (or fired) automatically by the database when a specified DML (INSERT, DELETE, UPDATE) statement is run against a certain database table.
- Is useful for doing things like advanced auditing of changes made to column values in a table.
- May fire before or after DML statement runs.

Trigger: Syntax

```
CREATE OR REPLACE TRIGGER trigger_name
{ BEFORE|AFTER } trigger_event { INSERT | DELETE | UPDATE } ON table_reference
[FOR EACH ROW[WHEN trigger_condition]]
DECLARE
/*Declarative section is here*/
BEGIN
/*Executable section is here*/
EXCEPTION
/* Exception section is here*/
END;
```

## Lab #12 – Applying PL/SQL On Database

### 1. Laboratory Objective:

The objective of this laboratory is to introduce students to how to apply PL/SQL on Database

### 2. Laboratory Learning Outcomes:

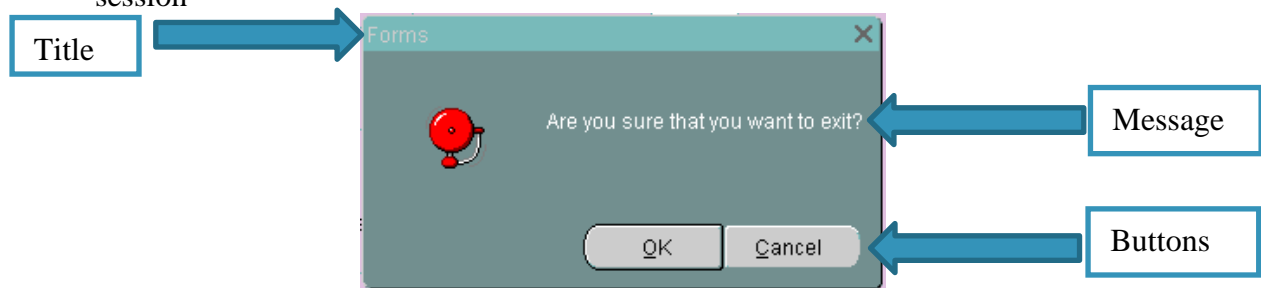
After completing this lesson, you should be able to do the following:

- Creating Alerts
- Applying PL/SQL On Database

### 3. Introductory Concepts

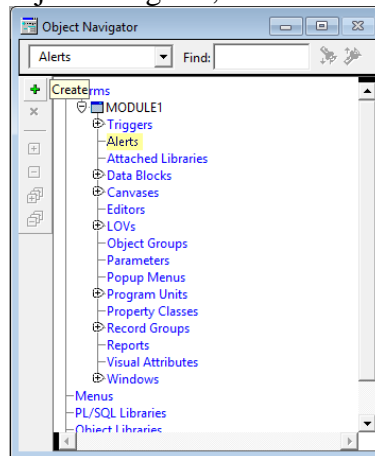
#### Run-Time Messages and Alert Overview

- Forms display messages at run time to inform the operator of events that occur in the session

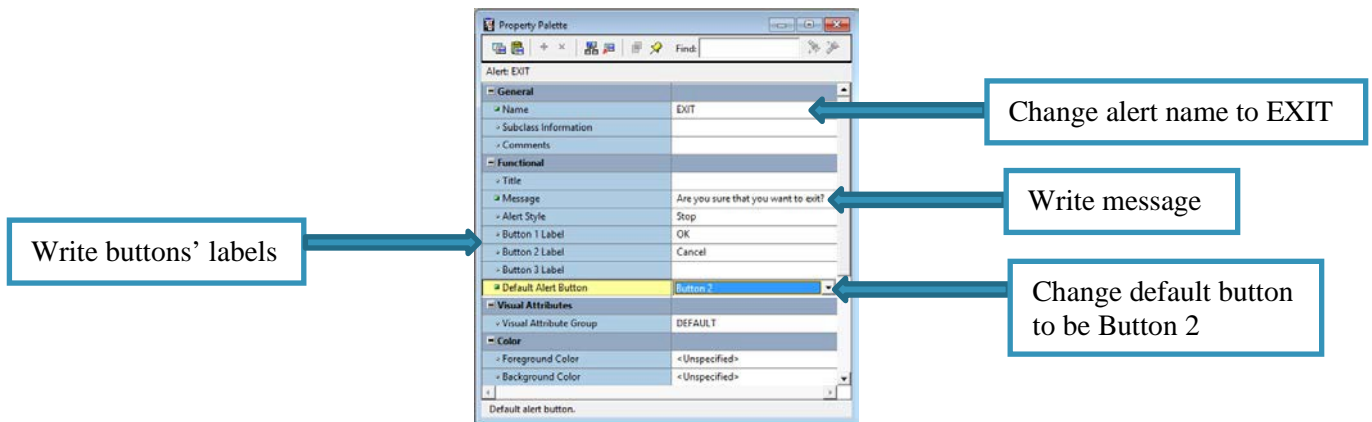


#### Creating Alerts

- Click on the alters icon in the object navigator, and then click on the create button



- In the Property Platter, you can define the following properties
  - Name
  - Title
  - Alert style
  - Button 1, Button 2, Button3 Labels
  - Default Alert Button
  - Message



### Controlling Alerts at Run Time

- Use Function Show\_alert to call your alert
- **Example:** create an exit\_form button and an alert that asks the user if he really wants to exit the form or not, define three buttons, and add the calling of the alert to the exit\_form button

Declare

num number;

Begin

num := show\_alert('EXIT');

if num = alert\_button1 then exit\_form;

end if;

End;

### How to create procedure in Form Builder?

1. To add a procedure, go to the Object Navigator and double click on Program Units. Immediately the window for the new Program unit will appear.
2. In the window for the new Program Unit, type in the name as update\_pro. Make sure that procedure is selected as the type of Program Unit and click OK.
3. Once you click OK, you will be automatically taken to the PL/SQL Editor window.
4. In the PL/SQL Editor, type in the following code for activating the alert and delete a record.

```
PROCEDURE update_pro IS
```

```
BEGIN
```

```
UPDATE student
```

```
SET f_name=student.f_name,
```

```
l_name=student.l_name,
```

```
major=student.major,
```

```
current_credits=student.current_credits
```

```
WHERE sid=student.sid;
```

```
commit;
```

```
END;
```



5. In update button, call the procedure

### Passing Parameters – Between Forms

#### ○ **In the Caller Form:**

1. Declare parameter list ( variable paramlist)
2. Create Parameter list using built in function (create\_parameter\_list('name'));
3. Add parameter value(s) using built in procedure (add\_parameter(parameter list,'name',text\_parameter,value); )
4. Call the form using call\_form built in and pass the parameter list to it.
5. Destroy the parameter list using built in procedure (destroy\_parameter\_list('name'));

#### ○ **In the Called Form:**

1. Create a parameter that has the same name as used in add\_parameter
2. Assign appropriate variable to this parameter using the prefix (:PARAMETER.name)

#### ○ **Example:**

Pass the student ID as a parameter from the main form you have to the other and display its information in the next form

#### ○ **In the main form (Caller Form):**

```
declare
param_test paramlist;
begin
param_test:= create_parameter_list('test');
add_parameter(param_test,'val1',text_parameter,:student.sid);
call_form('C:\Review2',no_hide,no_replace,no_query_only, 'test');
destroy_parameter_list('test');
end;
```

#### ○ **In the called form:**

- Create a parameter that has name: “val1”
- Create a two triggers on the form level:
- pre-query that has the following code:  
:REGISTERED\_STUDENTS.STUDENT\_ID := :PARAMETER.VAL1;
- when new-form-instance that has the following code: (  
execute\_query;

### IMPORTANT HINTS:

- If you want to click on a button to show a LOV, you can use show\_LOV built in function, it returns a Boolean.
- Some useful procedure:
  - Delete\_record
  - Next\_record
  - Previous\_record
  - Enter\_query;
  - Execute\_query;

**Example:**

1. Use the following code to create the user\_pass table

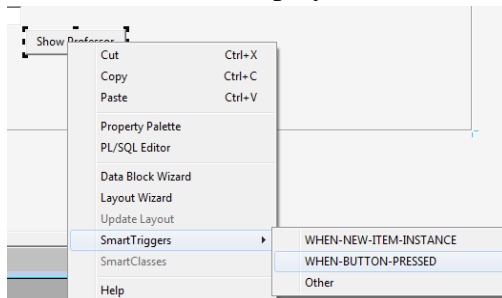
```
create table user_pass(
 username number(5) constraint up_pk primary key,
 pass varchar2(10),
 roless number(1)
);
insert into user_pass values(1111,'abc',1);
insert into user_pass values(2222,'a123',2);
commit;
```

2. Create the following forms:

**a. Form1.fmb**

This form contains one data block that is used to display the information about a class. It has a List of value LOV that contains professor names and IDs called 'PROFLOV'.

1. Write the code to display LOV in Show professor name button

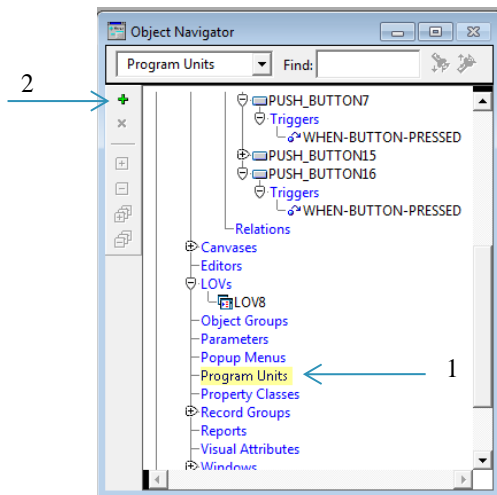


Right click on show professor button >  
Smart Triggers > When-button-pressed

add the following code:

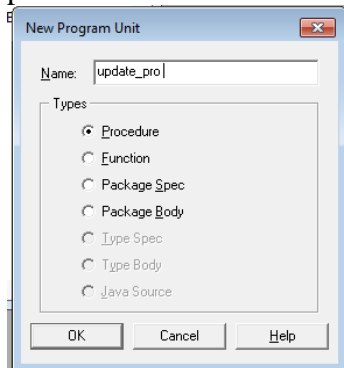
```
Declare
 return_LOV boolean;
begin
 return_LOV := show_LOV('PROFLOV');
end;
```

2. Write update\_pro procedure to update the record.



1. Click on the Program Units icon in the object navigator
2. Click on the create button

- a. In the window for the new Program Unit, type in the name as update\_pro. Make sure that procedure is selected as the type of Program Unit and click OK.



- b. Once you click OK, you will be automatically taken to the PL/SQL Editor window.
- c. In the PL/SQL Editor, type in the following code updating a record.

```
PROCEDURE update_pro IS
BEGIN
 UPDATE classes
 SET current_student = classes. current_student,
 Num_credits = classes. Num_credits,
 Room_id = classes. Room_id,
 Prof_id = classes. Prof_id
 WHERE dep = classes. dep and course = classes. course;
 commit;
END;
```

3. Write the code to call update\_pro procedure in the Update button

*Right click on Update button > Smart Triggers > When-button-pressed and add the following code*

```
update_pro;
```

4. Write the code to call the form Form2.fmb with room\_id passed as parameter. Do the necessarily changes in Form2.fmb

*Right click on Show Room Info button > Smart Triggers > When-button-pressed and add the following code*

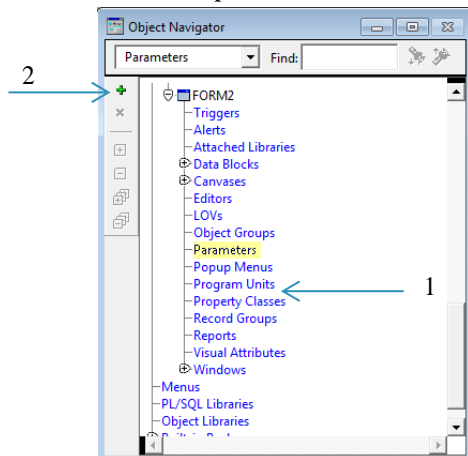
```
declare
 param_test paramlist;
begin
 param_test:= create_parameter_list('test');
 add_parameter(param_test,'val1',text_parameter,:classes.room_id);
 call_form('D:\Form2',no_hide, no_replace, no_query_only, 'test');
 destroy_parameter_list('test');
end;
```

#### **b. Form2.fmb**

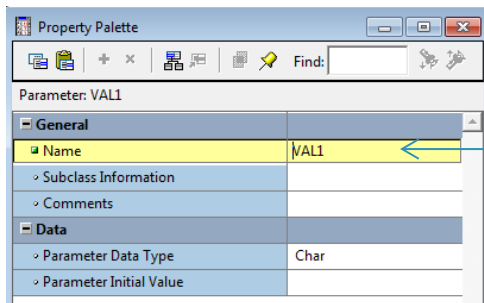
The screenshot shows a form titled 'Form2.fmb' with four input fields: 'Room Id' with value 'ROOM\_ID', 'Room Number' with value 'ROOM\_N', 'Building' with value 'BUILDING', and 'Number Seats' with value 'NUMBER\_SEA'. Below these fields is a 'Back' button.

This form contains room information from room table

1. Create a parameter that has name: "val1"

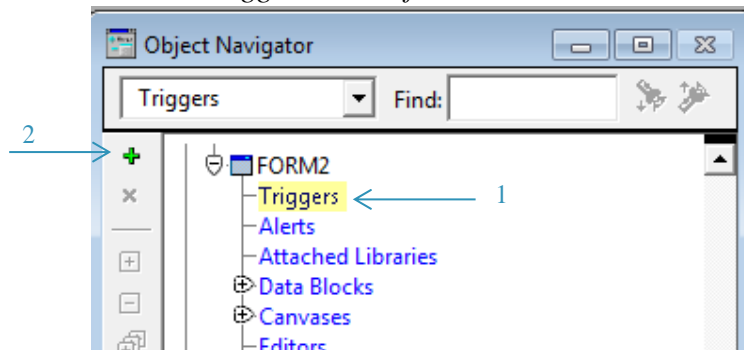


1. Click on the Parameters icon in the object navigator
2. Click on the create button



3. Change the parameter name to "VAL1"

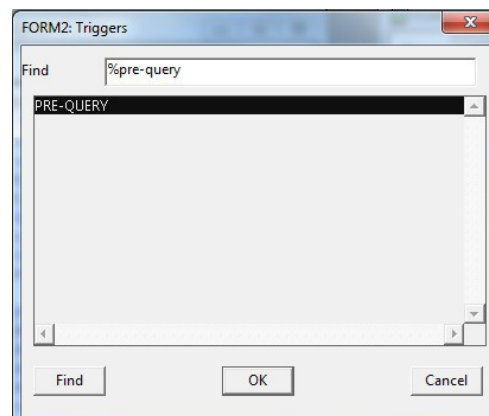
2. Create a triggers on the form level



1. Click on the Triggers icon in the object navigator

2. Click on the create button

3. Search for *pre-query* then click OK

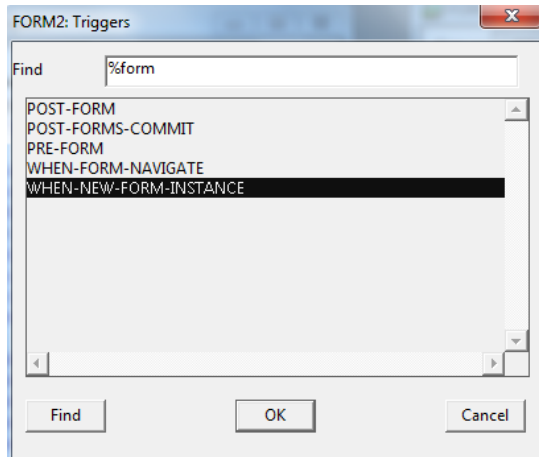


4. Add the following code

```
:rooms.room_id := :PARAMETER.VAL1;
```

5. Create another triggers on the form level

6. Search for *when-new-form-instance* then click OK



7. Add the following code:

```
execute_query;
```

8. Right click on Back button and select Smart Triggers > When-button-pressed and add the following code:

```
call_form('D:\Form1');
```

c. **Form3.fmb**

Just display welcome message

d. **Check\_login.fmb**

This form has only one data block for the user\_pass table (table that has user login information) as follow

- Create Alert “NO” that display error message for entering wrong username/password
- Add code for login such that if the role is 1 it calls form1 else if role is 2 itcalls form3

```

declare
 u_count number(1);
 u_role number(1);
 num number;
BEGIN
select count(*)
into u_count
from user_pass
where USER_PASS.username = :username and USER_PASS.pass = :pass;

IF u_count = 1 THEN
 select roless
into u_role
from user_pass
where USER_PASS.username = :username and USER_PASS.pass = :pass;
 if u_role = 1 then
 call_form('D:\Form1');
 else
 call_form('D:\Form3');
 end if;
else
 num := show_alert('NO');
 if num = alert_button3 then exit_form(no_commit);
 end if;
END IF;
end;

```

## Appendix A: Rules to follow by Computer Lab Users

- The loud conversations / discussion that disturbing the other users is prohibited.
- Audio CDs or applications with audio output may only be used with headphones with minimum volume that it should not be disturb other users.
- All cell phones are to be turned off or set to silent while in the lab. If you receive a phone call, you should exit the lab before answering your cell phone.
- Do not bring food or beverages inside the lab.
- Any file saved on the computer hard drive will be deleted without notice. Students should save their work onto an external storage device such as USB drive or CD.
- Changing hardware and software configurations in the computer labs is prohibited. This includes modifications of the settings, modification of system software, unplugging equipment, etc.
- Open labs are reserved for academic use only. Use of lab computers for other purposes, such as personal email, non-academic printing, instant messaging, playing games, and listening to music is not permitted.
- Please leave the computer bench ready for the next patron. Leave the monitor on the login screen, and do not forget to take goods related to you. While leaving computer bench please push the chair inside the computer bench.
- Users are responsible for their own personal belongings and equipment. Do not leave anything in the Computer Lab unattended for any length of time. The Computer Labs staffs are not responsible for lost or stolen items.
- Users are not allowed to clear paper jams in the printer by themselves.
- Operate the lab equipments with care.
- After using white-board the user must clean for other user.

Thanks for your cooperation.

Information Science Department





## Appendix B: Endorsement

### LABORATORY MANUAL FOR COURSE ISC 321 (Database Systems I)

| # | Instructor name | Remarks | Signature | Date |
|---|-----------------|---------|-----------|------|
| 1 |                 |         |           |      |
| 2 |                 |         |           |      |
| 3 |                 |         |           |      |