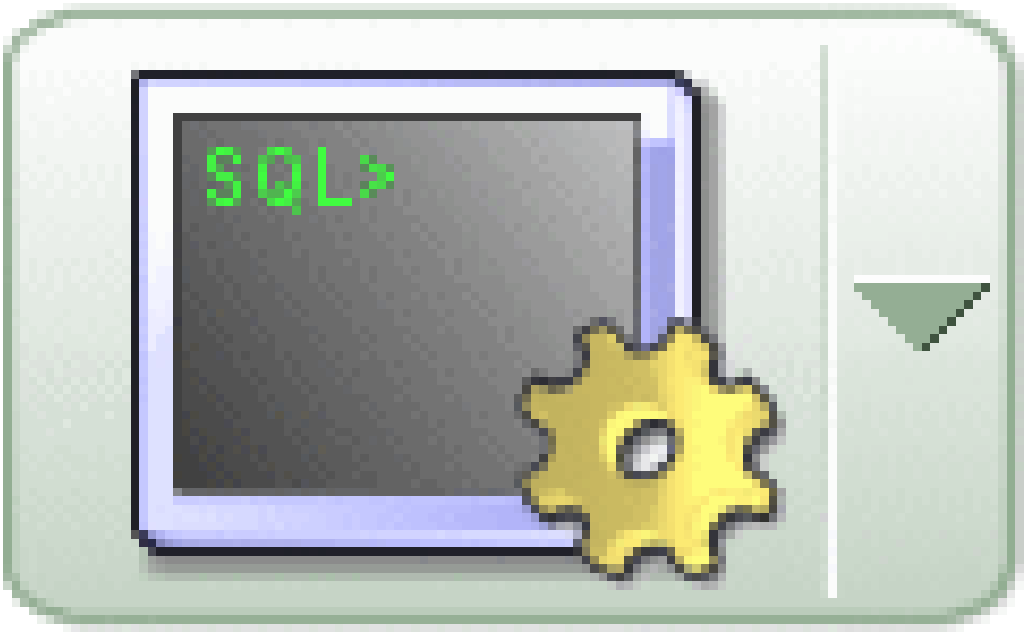

Exploring a Relational Database using Oracle SQL*Plus

Activity 2 – Objects Definition – UDTs and Object Tables

WORKBOOK OBJECTIVES

DEFINING OBJECT TYPES 1
USING OBJECT COLUMNS IN RELATIONAL TABLES 2
DEFINING OBJECT TABLES 3
REFERENCING OBJECT TABLES IN RELATIONAL TABLES 4
ALTERING RELATIONAL TABLES TO REFERENCE OBJECT TABLES 5
LOG OUT OF SQL USING THE EXIT COMMAND ☺ 6

DEFINING OBJECT TYPES

The main differences between RD and ER/ORD are objects and *complex [data]types*. Objects in oracle are created using PL/SQL, the *procedural 'wrapper'* for the declarative SQL. As PL/SQL has all the advantages of a full procedural language, many semi colons can appear in one PL/SQL block. To tell oracle that the block of code is finished, the *slash (/) symbol* is used.

- ✓ Logon to Oracle with your student login and student number as your password

- ➢ To display errors generated using PL/SQL the **SHOW ERRORS** command is used
- ➢ To create objects in our database use a *CREATE OR REPLACE TYPE* command
- ➢ To create an *object* ***type*** the following structure is used:

NOTE: The syntax looks similar to a standard table

```
CREATE OR REPLACE TYPE object_type_name AS OBJECT(
column_name1 DATATYPE,
column_name2 DATATYPE);
/
```

NOTE: *REPLACE* means it will write over an existing object with the same name. Therefore it doesn't need to be DROPped

- ✓ Create a new notepad file named '*your_login_object_definition.txt*'
- ✓ In the editor enter the command to create the ***object type*** outlined below
- ✓ Name the *object* ***type*** *invoice_address_type*

invoice_address_type (object type)

Column Name	Datatype
street	VARCHAR2(25)
city	VARCHAR2(25)
country	VARCHAR2(25)

NOTE: Try not to get mixed up between the *object* ***table*** and the *object* ***type***

- ✓ At the SQL> prompt and run your script file

- ➢ If you get *Warning: Type created with compilation errors* when you run your statement, it ***hasn't worked*** and you can ***not*** use that type. You will need to correct it and re run it.

USING OBJECT COLUMNS IN RELATIONAL TABLES

- It is possible to include an *object column* straight into a standard *relational* table
- To include an object column in a *relational* table the following structure is used:

```
CREATE TABLE table_name(
column_name1 datatype,
column_name2 datatype,
column_name2 object_type);
```

- ✓ In the object_definition file write the command to create the *relational* table outlined below
- ✓ Name the table *customers*

customers

Column Name	Datatype
customer_id	NUMBER(6)
customer_name	VARCHAR2(25)
invoice_address	*object_type*

- ✓ At the SQL> prompt paste in your new table
- ✓ At the SQL> prompt enter the code to view the *structure* of your new table

Q *What [data]type is your invoice_address column?*

NOTE: The data is actually stored in the column

DEFINING OBJECT TABLES

- To create an *object **table*** the following structure is used:

NOTE: This can only be done after the type has been declared

CREATE TABLE *object_**table**_name* OF *object_**type**_name*
(column_name DEFAULT '*value*'); ***--this line is optional***

- *column_name* refers to the element of the type that will form the attribute of the table

- ✓ In your *object_definition* file create an *object **table*** named *bill_addresses*
- ✓ Base your *bill_addresses object **table*** on the *object **type*** you defined before

- The *object **table*** uses the attributes of the *object **type*** as the column definition

- ✓ In the box below make a note of the *object **type*** name and *object **table*** name

Use this as a reference throughout the activities

*object **type*** **name**	
*object **table*** **name**	

- ✓ In your object_definition script file enter the command to view the structure of your *object_**table***
- ✓ At the SQL> prompt paste in the commands

REFERENCING OBJECT TABLES IN RELATIONAL TABLES

- *Object* ***table*** and ***types*** do not have primary keys and therefore are not referenced using foreign keys. Consequently, they can either be included as an *object* ***column*** or can be stored separately and referenced. Confused? Objects are user-defined [data]types. For example instead of a student's address being VARCHAR2 it would be an *object* ***type*** which defines the storage of all address details. This is tricky but practise makes perfect.

- There are 2 steps to creating *object* ***tables***:
- Firstly create the *object* ***type***
- Secondly use the *object* ***type*** to create the *Object* ***table***.

- Both the *object* ***table*** and the *object* ***type*** can be used as [data]types in both *relational* and *object tables*.

- To reference *object* ***tables*** and ***types*** the REF and SCOPE IS commands are used
- To reference an *object* ***type*** in an *object* ***table*** the following structure is used, instead of a normal datatype:

REF *object_**type**_name* SCOPE IS *object_**table**_name*

- ✓ In the object_definition file write the command to drop your old customers table
- ✓ In the object_definition file write the command to create the *relational* ***table*** outlined below
- ✓ Name the table *customers*

Q *What will be the names of your object_**type*** and *object _**table**?*

---

customers

Column Name	Datatype
customer_id	NUMBER(6)
customer_name	VARCHAR2(25)
invoice_address	REF *object_**type*** SCOPE IS *object _**table***

- ✓ At the SQL> prompt paste in your new table
- ✓ At the SQL> prompt enter the code to view your new table

ALTERING RELATIONAL TABLES TO REFERENCE OBJECT TABLES

- You can not add columns to an existing *object **table***
- You can alter *relational* tables to reference *object **tables*** using the usual method:

```
ALTER TABLE table_name ADD
(new_coulumn_name REF object_type_name SCOPE IS object_table_name);
```

- ✓ In your *object_definition* file enter the command to create the object named *state_type* outlined below

state_type

Column Name	Datatype
state	VARCHAR2(25)
country	VARCHAR2(25)

- ✓ At the SQL> prompt run your script file

- ✓ In your object_definition script file enter the command to view your *state_type* object
- ✓ At the SQL> prompt paste in the command

- ✓ In your object_definition file create an *object **table*** named *states*
- ✓ Base your *states* table on the *object **type*** you defined before
- This uses the attributes of the *object **type*** as the column definition

- ✓ In your *object_definition* file alter the *sites table* by adding the column outlined below
- ✓ This should *reference objects* in the *states object **table***
- ✓ Limit the scope of the column only to the *states table*
- ✓ Use the DESCRIBE command to ensure the new column exists

NOTE: Scope means where it will look for the column

Column Name	Datatype
state_ref	REF ?? SCOPE IS states

Q *Use the desc command to display the columns in the user_objects table, what are the 2 most useful columns?*

- ✓ In your object_definition file write the SQL to query the *user_objects* table, only showing the 2 most useful columns
- ✓ At the SQL> prompt and run your script file

LOG OUT OF SQL USING THE EXIT COMMAND ☺

NOTES