Table of Contents

[CHAPTER 18: OBJECT TYPE STORAGE 2](#_Toc416725986)

[Theory 2](#_Toc416725987)

[AIM 3](#_Toc416725988)

[Lab Exercise 18: OBJECT TYPE STORAGE 4](#_Toc416725989)

[1. Column Object 5](#_Toc416725990)

[2. Object Table 13](#_Toc416725991)

[3. Nested Object 19](#_Toc416725992)

[4. REF Object 22](#_Toc416725993)

[5. View Object structure 25](#_Toc416725994)

[SUMMARY 29](#_Toc416725995)

[REFERENCES 30](#_Toc416725996)

[INDEX 31](#_Toc416725997)

# CHAPTER 18: OBJECT TYPE STORAGE

## Theory

In the previous chapter, we see how to create an Object Type. The Object Type contains attributes and, optionally, methods. The second step is to save instances of this Object Type in the database, so we don't have to re-fill them again when the application closed. In other languages, this process may be called serialization.

You can save the object instance state by two main methods:

1. Column Object: create a table in which you add a column of Object Type.
2. Object Table: create a table of Object Type row.

After saving the Object Instance in table, you can SELECT, UPDATE, DELETE and INSERT from the table as usual but with some consideration. We will cover these issues in the lab.

You will also review Nested Object in which an Object Type has one or more Object Type attribute(s).

It is very important to know that Oracle will eventually convert your Object Table into regular table with some additional columns. You will learn how to view the Object Type structure in this lab but you will actually see its benefits in the next chapter.

## AIM

The AIM of the following exercise is to demonstrate the use Object Type.

The steps involved will include:

* Column Object
* Object Table
* Nested Object
* REF Object
* View Object structure

In general, lab exercises are done in sequential order. Thus, it is assumed that you successfully completed the previous labs. However, not all previous labs are required. Please be sure to run the following lab before proceeding:

* Installing Oracle Database 12c.
* Object Types

Estimated Completion Time:

25 minutes

# Lab Exercise 18: OBJECT TYPE STORAGE

|  |
| --- |
|  |

## Column Object

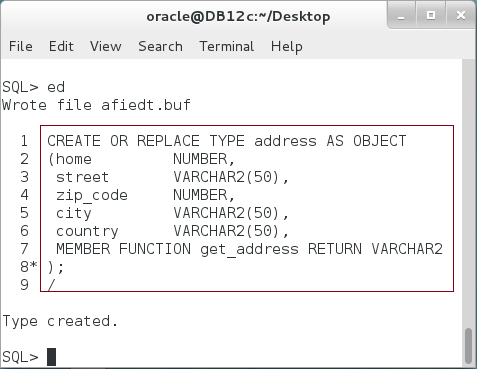
**Step 1:** Open the Terminal, open SQL\*Plus console and connect to hr schema.

|  |  |
| --- | --- |
| Command | Description |
| sqlplus | Open SQL\*Plus console. |
| hr/oracle | connect to **hr** schema. |

****

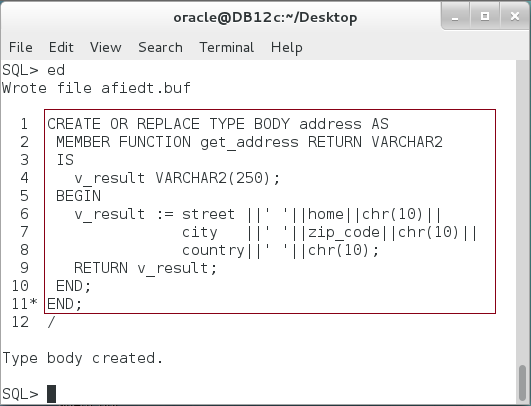
**Step 2:** Execute the following block:

|  |  |
| --- | --- |
| Command | Description |
| **CREATE OR REPLACE TYPE** address **AS OBJECT** | Create object specification. |
| (home NUMBER, |
| street VARCHAR2(50), |
| zip\_code NUMBER, |
| city VARCHAR2(50), |
| country VARCHAR2(50), |
| MEMBER FUNCTION get\_address RETURN VARCHAR2 |  |
| ); |  |
| / |  |

****

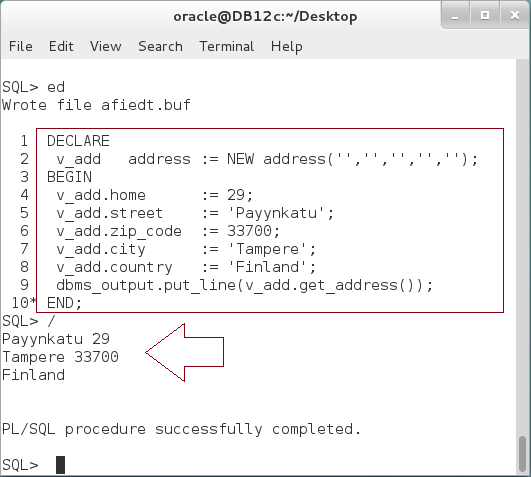
**Step 3:** Create a Type Body as shown below:

|  |  |
| --- | --- |
| Command | Description |
| CREATE OR REPLACE **TYPE BODY** address AS | Create type body |
| MEMBER FUNCTION get\_address RETURN VARCHAR2 |
| IS |
| v\_result VARCHAR2(250); |
| BEGIN |
| v\_result := street ||' '||home||chr(10)|| |
| city ||' '||zip\_code||chr(10)|| |
| country||' '||chr(10); |
| RETURN v\_result; |
| END; |
| END; |
| / |



**Step 4:** Of course, you can use the type as usual: declare variables, set attitudes' values, and call methods as show below:

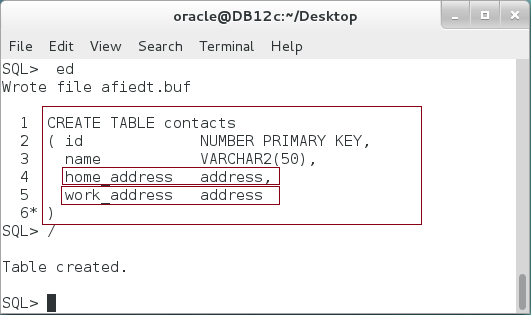
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_add address := NEW address('','','','',''); |  |
| BEGIN |  |
| v\_add.home := 29; |  |
| v\_add.street := 'Payynkatu'; | Use the type as usual. |
| v\_add.zip\_code := 33700; |
| v\_add.city := 'Tampere'; |
| v\_add.country := 'Finland'; |  |
| dbms\_output.put\_line(v\_add.get\_address()); |  |
| END; |  |
| / |  |



**Please note:** So far, there is nothing new. The question now, how can we save this object instance to be used later on?

**Step 5:** Create an Column Object as show below:

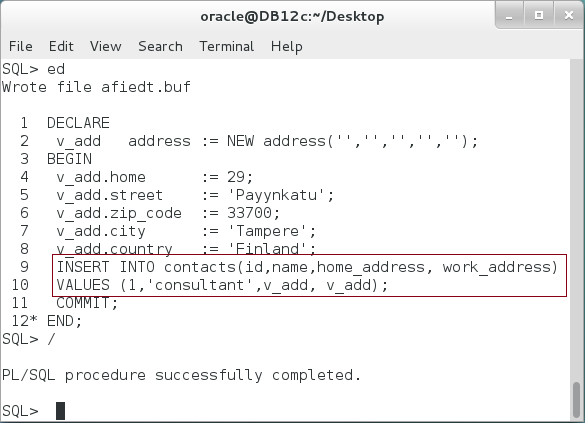
|  |  |
| --- | --- |
| Command | Description |
| CREATE TABLE contacts |  |
| ( id NUMBER PRIMARY KEY, |  |
| name VARCHAR2(50), |  |
| home\_address **address**, | You can define a table column of the type "address", which is an Object Type. |
| work\_address **address** |
| ) |
| / |  |



**Please note:** Both, "home\_address" and "work\_address" are of the "address" Object Type.

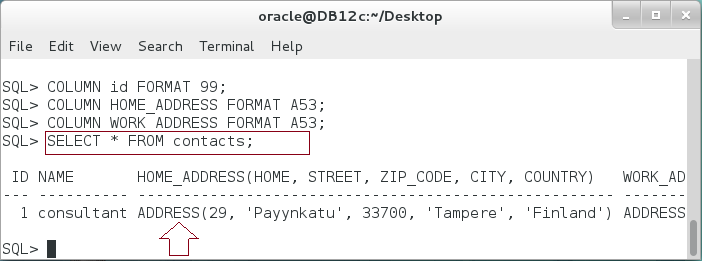
**Step 6:** Modify the previous block insert the Object Instance into a column in a table.

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_add address := NEW address('','','','',''); |  |
| BEGIN |  |
| v\_add.home := 29; |  |
| v\_add.street := 'Payynkatu'; |  |
| v\_add.zip\_code := 33700; |  |
| v\_add.city := 'Tampere'; |  |
| v\_add.country := 'Finland'; |  |
| **INSERT INTO contacts**  **(id,name,home\_address, work\_address)** | **Insert the Object Instance as a column in relational Table.** |
| **VALUES (1,'consultant',v\_add, v\_add);** |
| COMMIT; |
| END; |
| / |  |

****

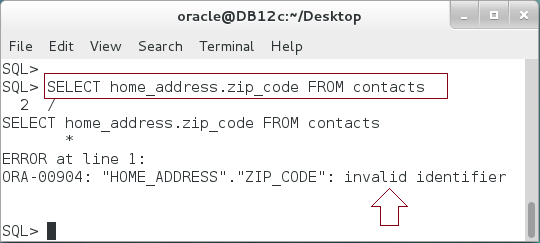
**Step 7:** Query the "**contacts**" table as shown below:

|  |  |
| --- | --- |
| Command | Description |
| COLUMN id FORMAT 99; |  |
| COLUMN HOME\_ADDRESS FORMAT A53; | **Query the table using "\*" notation.** |
| COLUMN WORK\_ADDRESS FORMAT A53; |
| **SELECT \* FROM contacts;** |

****

**Step 8:** Try to query the "**zip\_code**" attribute of the "**home\_address**" Column Object as show:

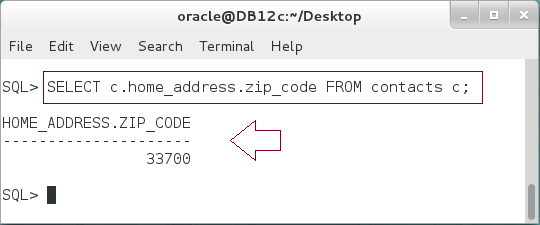
|  |  |
| --- | --- |
| Command | Description |
| SELECT home\_address.zip\_code FROM contacts |  |
| / |  |

****

**Please note:** The query failed. Even though, the "**zip\_code**" is an attribute of "**home\_address**" Column Object.

**Step 8:** Modify the previous query so you add a table alias as shown below:

|  |  |
| --- | --- |
| Command | Description |
| SELECT **c.**home\_address.zip\_code FROM contacts **c** | Add table alias "**c**". |
| / |

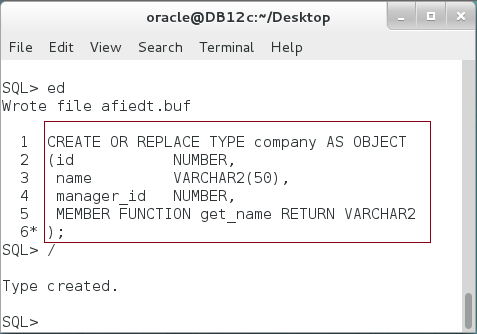
****

**Please note:** To avoid inner capture and similar problems resolving references, Oracle Database requires you to use a table alias to qualify any dot-notational reference to subprograms or attributes of objects.

## Object Table

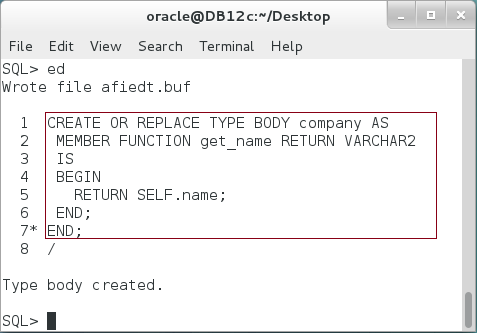
**Step 1:** Create an Object "company" as shown below:

|  |  |
| --- | --- |
| Command | Description |
| **CREATE OR REPLACE TYPE** company **AS OBJECT** |  |
| (id NUMBER, |  |
| name VARCHAR2(50), |  |
| manager\_id NUMBER, |  |
| MEMBER FUNCTION get\_name RETURN VARCHAR2 |  |
| ); |  |
| / |  |

****

**Step 2:** Implement the Objet Type Body as shown below:

|  |  |
| --- | --- |
| Line | Description |
| CREATE OR REPLACE **TYPE BODY** company AS | Type body |
| MEMBER FUNCTION get\_name RETURN VARCHAR2 |
| IS |
| BEGIN |
| RETURN SELF.name; |
| END; |
| END; |
| / |  |



**Step 3:** Create Object Table as shown below:

|  |  |
| --- | --- |
| Line |  |
| CREATE TABLE companies OF company; | Create a table of Objects. |



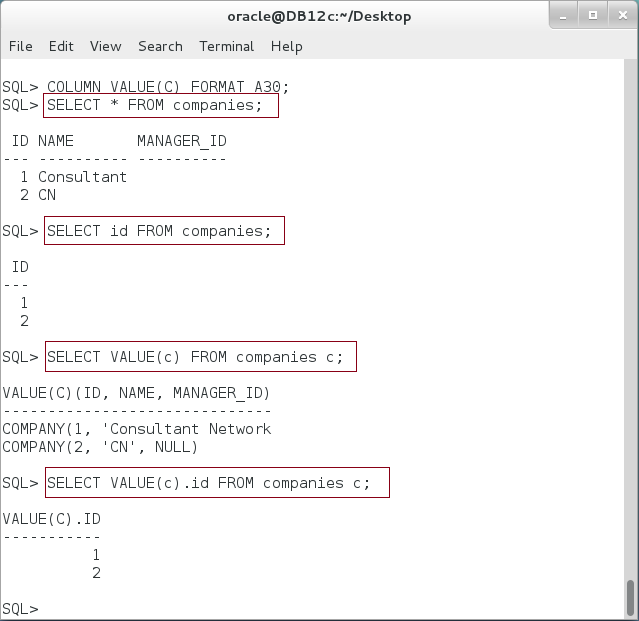
**Step 4:** Insert Object Instances in the Object Table as shown below:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company := NEW company(1,'Consultant Network',''); |  |
| BEGIN |  |
| **INSERT INTO** companies |  |
| **VALUES** (**v\_com**); |  |
| INSERT INTO companies |  |
| VALUES (**company(2,'CN','')**); |  |
| COMMIT; |
| END; |
| / |  |



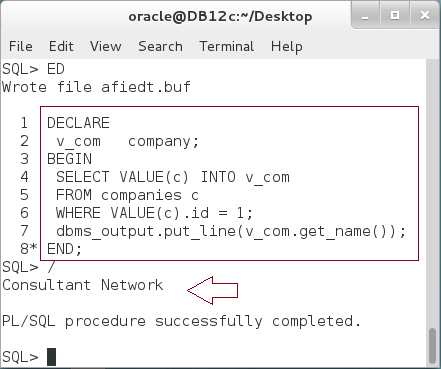
**Step 5:** Query the Object Table as shown below:

|  |  |
| --- | --- |
| Line | Description |
| SELECT \* FROM companies; | All attributes as columns |
| SELECT id FROM companies; | Object Attribute as column. |
| SELECT VALUE(c) FROM companies c; | Return Object. |
| SELECT VALUE(c).id FROM companies c; | Return Object attribute. |



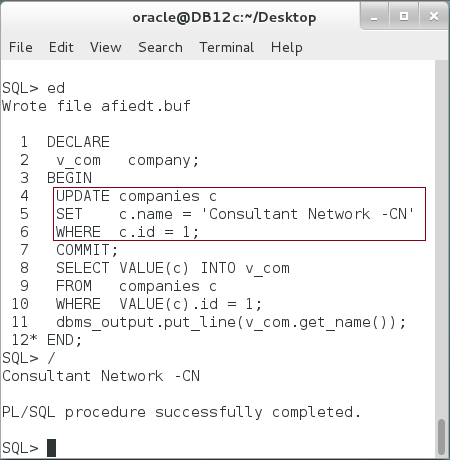
**Step 6:** Please note how use "VALUE" function to return an Object Instance from the Object Table in a PL/SQL block as shown below:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company; |  |
| BEGIN |  |
| SELECT **VALUE(c)** INTO **v\_com** |  |
| FROM companies c |  |
| WHERE **VALUE(c).id** = 1; |  |
| dbms\_output.put\_line(v\_com.get\_name()); |  |
| END; |  |
| / |  |



**Step 7:** Update the Object Instance using UPDATE statement as shown below:

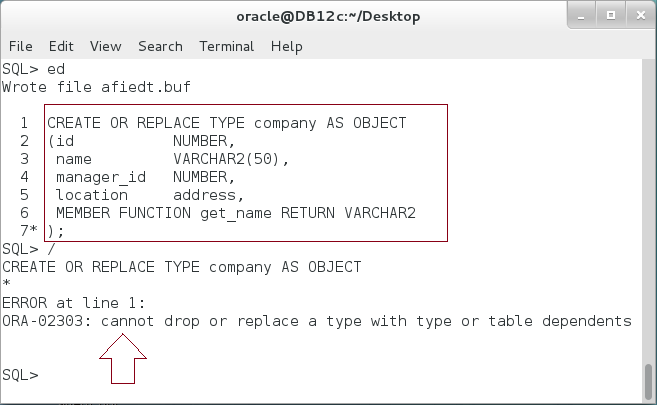
|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company; |  |
| BEGIN |  |
| **UPDATE companies c** | Update Object Table as shown. |
| **SET c.name = 'Consultant Network -CN'** |
| **WHERE c.id = 1;** |
| COMMIT; |
| SELECT VALUE(c) INTO v\_com |
| FROM companies c |  |
| WHERE VALUE(c).id = 1; |  |
| dbms\_output.put\_line(v\_com.get\_name()); |  |
| END; |  |
| / |  |



## Nested Object

**Step 1:** First, we will try to modify "company" Object Type and adding one Object Type attribute "address" using "CREATE OR REPLACE" syntax as shown below:

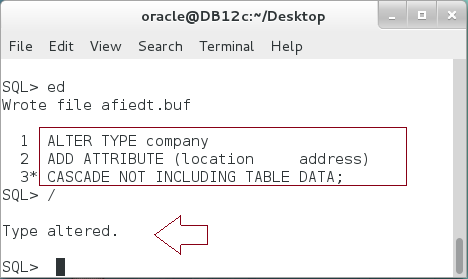
|  |  |
| --- | --- |
| Line | Description |
| CREATE OR REPLACE TYPE company AS OBJECT |  |
| (id NUMBER, |  |
| name VARCHAR2(50), |  |
| manager\_id NUMBER, | Add "location" attribute of "address" type using "CREATE OR REPLACE" syntax. |
| **location address**, |
| MEMBER FUNCTION get\_name RETURN VARCHAR2 |
| ); |
| / |



**Please note:** The operation failed because there is a Type or Table which uses "**company**" Object Type. In our case, it is "**companies**" table.

**Step 2:** Use "ALTER TYPE" command to modify an Object Type with dependences as shown below:

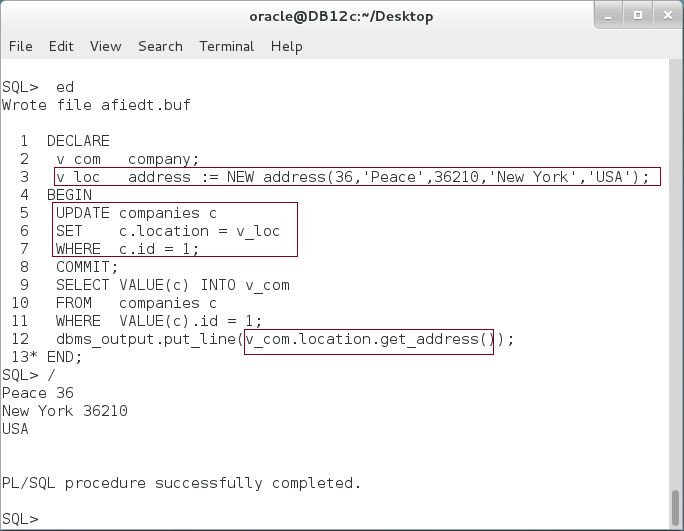
|  |  |
| --- | --- |
| Line | Description |
| ALTER TYPE company |  |
| ADD ATTRIBUTE (location address) |  |
| CASCADE NOT INCLUDING TABLE DATA; |  |

****

**Please note:** Now, a "company" type has an attribute of "address" Object Type.

**Step 3:** Access and update the inner Object Type attribute:

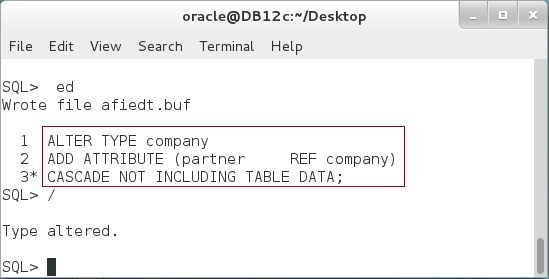
|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company; |  |
| v\_loc address := NEW address(36,'Peace',36210,'New York','USA'); |  |
| BEGIN | Assign the inner Object attribute value. |
| UPDATE companies c |
| SET **c.location = v\_loc** |
| WHERE c.id = 1; |
| COMMIT; |
| SELECT VALUE(c) INTO v\_com |  |
| FROM companies c |  |
| WHERE VALUE(c).id = 1; |  |
| dbms\_output.put\_line(**v\_com.location.get\_address()**); |  |
| END; |  |
| / |  |

****

## REF Object

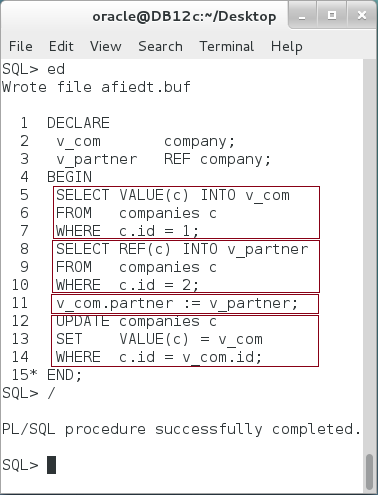
**Step 1:** Alter "company" object type to add a pointer attribute to "company" Object Type [like a self join in table-relational database] as shown below:

|  |  |
| --- | --- |
| Line | Description |
| ALTER TYPE company |  |
| ADD ATTRIBUTE (**partner REF company**) |  |
| CASCADE NOT INCLUDING TABLE DATA; |  |
| / |  |



**Step 2:** Execute the following block:

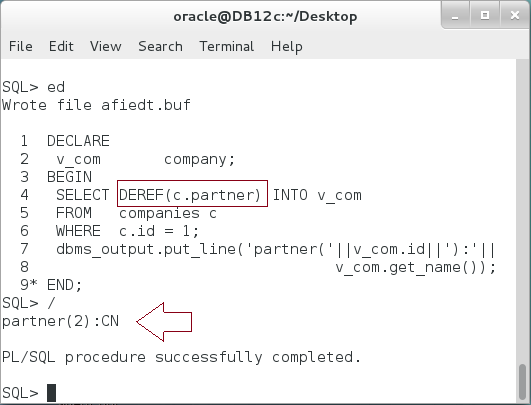
|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company; |  |
| v\_partner REF company; |  |
| BEGIN |  |
| **SELECT VALUE(c) INTO v\_com** | Get a "**company**" object instance from "**companies**" object tables. |
| FROM companies c |
| WHERE c.id = 1; |
| **SELECT REF(c) INTO v\_partner** | Get a **REF** to a "**company**" object instance of company id=2 |
| FROM companies c |
| WHERE c.id = 2; |
| v\_com.partner := v\_partner; | Assign a REF to company id=2 as company id=1 partner. |
| **UPDATE companies c** |
| **SET VALUE(c) = v\_com** |
| WHERE c.id = v\_com.id; |  |
| END; |  |
| / |  |

****

**Please note:** A partner of the first company (id=1) is now referring to the second company (id=2).

**Step 3:** To get the object in which the REF attribute is referring to, use "**DEREF**" function as shown:

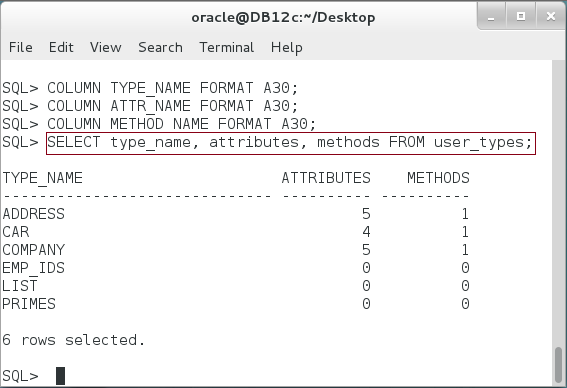
|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_com company; |  |
| BEGIN |  |
| SELECT **DEREF(**c.partner**)** INTO v\_com | Get the object in which a pointer "**partner**" is referring to. |
| FROM companies c |
| WHERE c.id = 1; |
| dbms\_output.put\_line('partner('||v\_com.id||'):'|| |
| v\_com.get\_name()); |
| END; |  |
| / |  |

****

## View Object structure

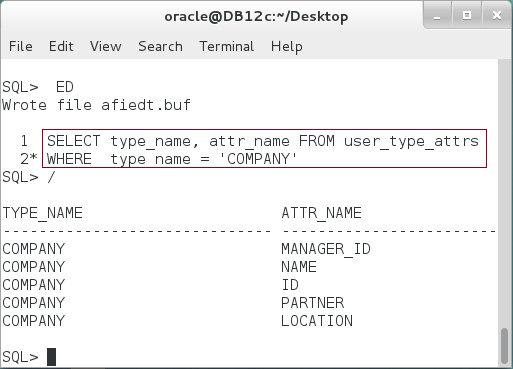
**Step 1:** To know about the types you have, execute the following query:

|  |  |
| --- | --- |
| Line | Description |
| COLUMN TYPE\_NAME FORMAT A30; |  |
| COLUMN ATTR\_NAME FORMAT A30; |  |
| COLUMN METHOD\_NAME FORMAT A30; |  |
| **SELECT type\_name, attributes, methods FROM user\_types;** |  |



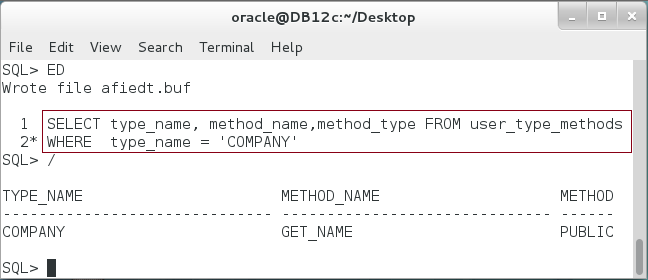
**Step 2:** To list all attributes of the Object Type, use "**USER\_TYPE\_ATTRS**" view:

|  |  |
| --- | --- |
| Line | Description |
| SELECT type\_name, attr\_name FROM **user\_type\_attrs** |  |
| WHERE type\_name = **'COMPANY'**; |  |

****

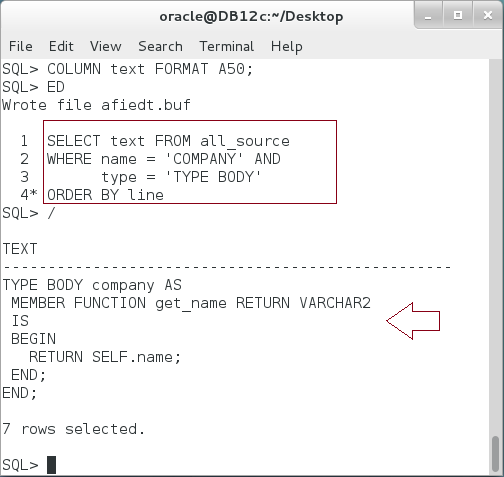
**Step 3:** List all methods of the type using "**USER\_TYPE\_METHODS**" view as shown below:

|  |  |
| --- | --- |
| Line | Description |
| SELECT type\_name, method\_name,method\_type FROM **user\_type\_methods** |  |
| WHERE type\_name = **'COMPANY'** |

****

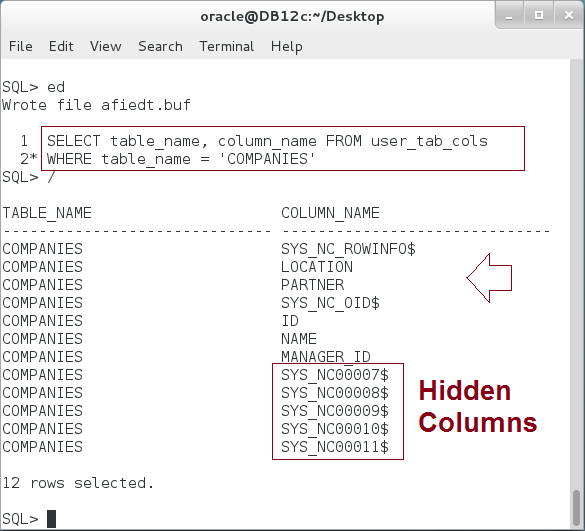
**Step 4:** To view of source of the Object Type Body, execute the following:

|  |  |
| --- | --- |
| Line | Description |
| COLUMN text FORMAT A50; |  |
| SELECT text FROM **all\_source** |  |
| WHERE name = 'COMPANY' AND |  |
| type = 'TYPE BODY' |  |
| ORDER BY line |
| / |  |

****

**Step 5:** To view the columns of the Object Table, execute the following:

|  |  |
| --- | --- |
| Line | Description |
| SELECT table\_name, column\_name FROM **user\_tab\_cols** |  |
| WHERE table\_name = 'COMPANIES' |  |
| / |  |

****

**Please note:** Oracle automatically creates hidden columns to store the columns of the inner Object, "address". However, using REF column will not create columns of the object. Instead, it stores the reference to the object only.

# SUMMARY

Object Type can be stored in a database either as a column in a relational table, thus is called Column Object, or as whole row of the table, thus is called Object Table. Oracle stores the object's attributes as column in a relational table. Therefore, it creates a hidden columns to store the attributes of the inner object. You can view the Object type and table from the dictionary tables. To retrieve a row from Object Table as object instance, use VALUE function. To retrieve the object instance reference stored in the Object Table, use REF function. To get the object in which the pointer REF is referring to, use DEREF function. You can use REF as a column data type. In such case, Oracle stores the reference to the existing object instance stored in an Object Table.

After completing this lab exercise, you should be able to store the Object Type either in Column Object or Object Table.

# REFERENCES

* https://docs.oracle.com/database/121/ADOBJ/adobjint.htm#ADOBJ001
* https://docs.oracle.com/database/121/ADOBJ/adobjbas.htm#ADOBJ7095

# INDEX

Column Object 2, 3, 5, 8, 11, 27

Object Type 3, 27