Oracle Object-Relational Database Example 3

(Object-Relational Database with Member Functions)

CS 5513

Member Functions (Methods)

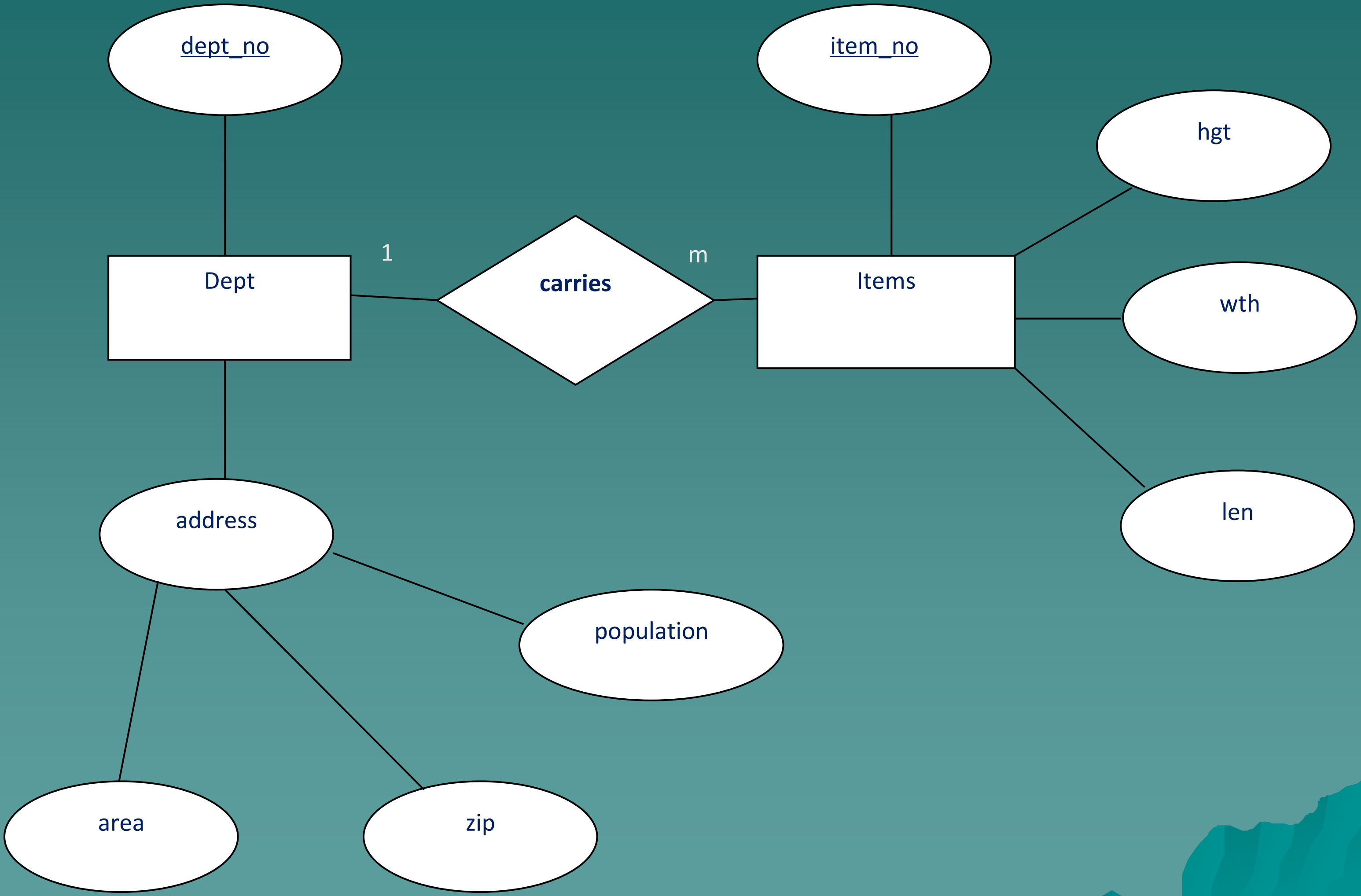
- Comparison functions: used for comparing instances of object types; without such functions Oracle has no idea of how to compare types. Only one comparison function is allowed in a type.
  - Map function: compares or sorts multiple objects of a given built-in type. Use the keyword MAP MEMBER FUNCTION to define a map function.
  - Order function: compares two objects of a given built-in type and returns a value that encodes the order of relationship. For example, it may return -1 if the first object is smaller, 0 if they are equal, and 1 if the first object is larger. Use the keyword ORDER MEMBER FUNCTION to define an order function
- Non-comparison functions: used for common data accessing purposes. Use the keyword MEMBER FUNCTION to define a non-comparison function

Function Restrictions

- FUNCTION RESTRICTIONS
  - Use PRAGMA RESTRICT_REFERENCES(*func_name*, *restrictions*) to assign restrictions to a function.
  - More than one restriction can be assigned to a function.
  - Restriction “WNDS” means the *function can not modify database tables, i.e. no DDL*; Restriction “RNDS” means the *function can not query database tables, i.e. no DML.*
  - There are more restrictions. Please check the Oracle official documents to see all restrictions:

  http://docs.oracle.com/cd/B13789_01/appdev.101/b10807/13_elems039.htm

Example: ER Diagram



Type Creations with functions inside – address_type

- CREATE TYPE address_type AS object
  (zip number(5),
  population number(7),
  area number(5),
  MAP MEMBER FUNCTION rank RETURN INTEGER,
  PRAGMA RESTRICTION_REFERENCES(rank, WNDS, RNDS));

Type Body Creations with functions inside – address_type

- *CREATE OR REPLACE TYPE BODY address_type AS*

```
 MAP MEMBER FUNCTION rank RETURN INTEGER IS
  BEGIN
    RETURN population/area;
       --compare objects based on the population of unit
       --area.
  END;
END; --this end is for body creation not for begin
```

Type Creations with functions inside – items_type

```
CREATE TYPE items_type AS object
(item_no number(5),
hgt number(5),
wth number(5),
len number(5),
dept_no REF dept_type,
MEMBER FUNCTION volume RETURN INTEGER,
MAP MEMBER FUNCTION surface RETURN INTEGER);
```

Type Body Creations with functions inside – items_type

- *CREATE OR REPLACE TYPE BODY items_type AS*

```
      MEMBER FUNCTION volume RETURN INTEGER IS
       BEGIN
        RETURN len * wth * hgt;
       END;
      MAP MEMBER FUNCTION surface RETURN INTEGER IS
       BEGIN
        RETURN 2 * (len * wth + len * hgt + wth * hgt);
       END;
END; --this end is for body creation not for begin
```

Type Creations – dept_type

- CREATE TYPE dept_type AS object
  (dept_no number(5),
  address address_type,
  );

Table Creations

- CREATE TABLE dept_tab of dept_type
  (dept_no primary key)
  object id primary key;

- CREATE TABLE items_tab of items_type
  (primary key(item_no),
  foreign key(dept_no)
  references dept_tab)
  object id primary key;

Insertion

- A normal insertion:

```
INSERT INTO items_tab VALUES
(1001, 10, 5, 5);
```

- An insertion to a table that has complex attributes:

```
INSERT INTO dept_tab
VALUES(100,
address_type(73019, 115562, 189));
```

Selection with functions inside

- *Non-comparison function*

  SELECT a.volume() FROM items_tab a WHERE a.len = 5;

  Note: the volume function is explicitly called when we want to use it in a select query.

- MAP comparison function

  SELECT * FROM dept_tab a ORDER BY a.address DESC;

  Note: the rank function is implicitly called when we want to sort address_type objects.

One more example for MAP function – using simple PL-SQL

```
DECLARE
  obj_1 items_type;
  obj_2 items_type;
BEGIN
  obj_1  := NEW items_type(100,10,5,5,null);
  obj_2  := NEW items_type(101,5,2,2,null);
  IF (obj_1  > obj_2) THEN
--or obj_1.surface() > obj_2.surface()
--compare two items_type objects based on their surface
DBMS_OUTPUT.PUT_LINE('obj_1  is bigger');
  ELSE
DBMS_OUTPUT.PUT_LINE('obj_1  is not bigger');
  END IF;
END;
```

Note: when comparison is performed the surface function is called implicitly.