ORDB: Methods and Inheritance

Member Method

Functions or procedures are declared in an object type definition to implement the behavior of objects.

- Declared in a CREATE TYPE statement and Defined in a CREATE TYPE BODY statement.

Declare member methods

```
CREATE TYPE MenuType AS OBJECT (
        bar REF BarType,
        beer REF BeerType,
        price FLOAT,
        MEMBER FUNCTION priceInYen(rate IN FLOAT)
                RETURN FLOAT)
        /
```

Define member methods

```
CREATE TYPE BODY MenuType AS
MEMBER FUNCTION
priceInYen(rate FLOAT)
RETURN FLOAT IS
        BEGIN
                RETURN rate * SELF.price;
        END;
END;
/
```

```
CREATE TABLE Sells OF MenuType;
Commit ;
```

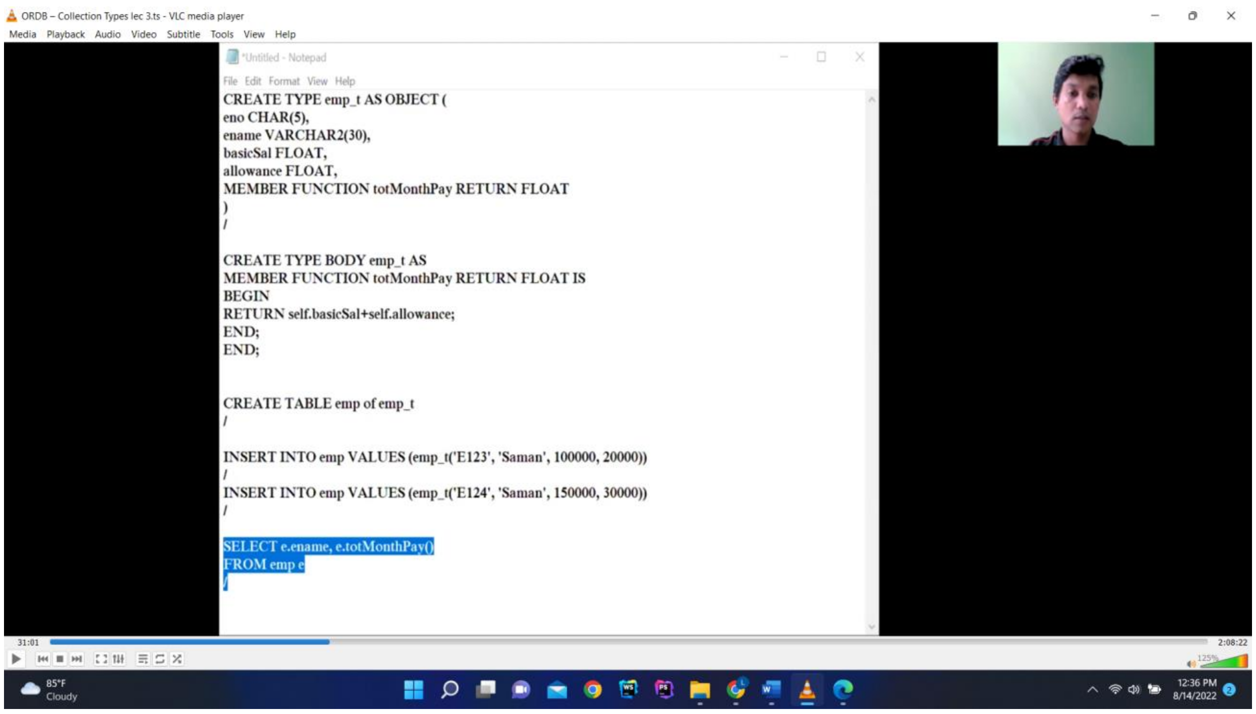
Call member method

```
SELECT s.beer.name, s.priceInYen(106.0)
FROM Sells s
WHERE s.bar.name = 'Joe''s Bar';
```

mehi method ekk create krddi use krn SELF key word ek java vala this key word eka vge. Mehi **priceInYen** method ekt parameter ekk pass krnn oni nattan apita nikmma **priceInYen** kiyl tiynn puluvn. Keep mind parameter pass krnne nattan () varahan danne na method ek declare karan tana. just name ek vitri tiyanne. But method ek call krddi api open close bracket dann oni

SHOW ERROR key word eken apita error ek mkkd kiyla balagan puluvn.
ED key word eken kalin liypu code ekk aran eka edit krnn puluvn.

Example code :



```
CREATE TYPE emp_t AS OBJECT (
eno CHAR(5),
ename VARCHAR2(30),
basicSal FLOAT,
allowance FLOAT,
MEMBER FUNCTION totMonthPay RETURN FLOAT
)
/

CREATE TYPE BODY emp_t AS
MEMBER FUNCTION totMonthPay RETURN FLOAT IS
BEGIN
RETURN self.basicSal+self.allowance;
END;
END;


CREATE TABLE emp of emp_t
/

INSERT INTO emp VALUES (emp_t('E123', 'Saman', 100000, 20000))
/
INSERT INTO emp VALUES (emp_t('E124', 'Saman', 150000, 30000))
/

SELECT e.ename, e.totMonthPay()
FROM emp e
/
```

Declare a new member method to an existing object

```
ALTER TYPE MenuType
        ADD MEMBER FUNCTION
        priceInUSD(rate FLOAT)
        RETURN FLOAT
        CASCADE;
```

(Mehi cascade keyword ek anivaryai. Bcz existing ekkt add krn nisa)

Define member methods to existing one

```
CREATE OR REPLACE TYPE BODY
        MenuType AS
        MEMBER FUNCTION
        priceInYen(rate FLOAT)
        RETURN FLOAT IS
        BEGIN
                RETURN rate * SELF.price;
        END priceInYen;

        MEMBER FUNCTION
        priceInUSD(rate FLOAT)
        RETURN FLOAT IS
        BEGIN
                RETURN rate * SELF.price;
        END priceInUSD;

END;
```

(Me vidiyata tavat function ekak define krddi kalin tibb function ekat api aaai liynn oni. nattan error ekk env)

Object Comparisons

Saralavam qwot object compression kiynne object compare krl eva same da, mkkd greater than eka, mkkd less than ek kiyl hoyn ekata. Mekat ORDB vala Map method ek ha Order method ek use krnva

- The values of scalar data types such as CHAR or REAL have a predefined order.
- But instances of an object type have no predefined order.
- To compare two items of a user-defined type, define an order relationship using a map or an order method

1. Map Object

Apita map key word ek use kirimen,

1. Comparisons krnn puluvn, such as obj_1 > obj_2 and
2. DISTINCT, GROUP BY, ORDER BY keyword use krnnt puluvn. (Map keyword eka natuva apita me keyword use krnn ba. Bcz object valata compression krnn reason ekk hadagann ba.)

Map method ekkt parameter tiyenna be

```
CREATE TYPE Rectangle_type AS OBJECT (
            length NUMBER,
            width NUMBER,
            MAP MEMBER FUNCTION area RETURN NUMBER
 );

CREATE TYPE BODY Rectangle_type AS MAP MEMBER
            FUNCTION area RETURN NUMBER IS
            BEGIN
                        RETURN length * width;
            END area;
END;

CREATE TABLE rectangles OF Rectangle_type;
```

(Mekat kalin ek vgemai. Venasa tama function ek declare krn tanai define krn tanai function ekt kalin **MAP** keyword ek use krn eka)

incase above code create a compilation error while creating body

```
CREATE TYPE BODY Rectangle_type AS MAP MEMBER
  2  FUNCTION area RETURN NUMBER IS
  3  BEGIN
  4  RETURN length * width;
  5   END area;
  6  END;
  7  /
```

Example: - (Ihata create krpu table ekt data insert krl api example ek krmu)

```
INSERT INTO rectangles VALUES (1,2);
INSERT INTO rectangles VALUES (2,1);
INSERT INTO rectangles VALUES (2,2);
```

1. **SELECT DISTINCT VALUE(r) FROM rectangles r;**

```
Output -> RECTANGLE_TYP (1, 2)
          RECTANGLE_TYP (2, 2)
```

(Attatama meke comparison ek siddavenne area valat. Api samanyayen parameter pass krnne natuva method ekk haduvama ek automa call venva. So, Meka baluvama tereneva api data 3k insert krt print vela tiyenne 2i. bcz mehi first dekema area ek 2i. So Distinct keyword ek nisa duplicated code ain vela tiynva. Itin me vge Distinct, group by, order by keyword use krnn oni nm map function create krnn venva.)

2. **SELECT VALUE(r)**
   FROM rectangles r
   ORDER BY VALUE(r) desc
   /

Value(r) eken venne object ek e vidiytm print karan eka.

2. *Order methods*

Called automatically whenever two objects need to be compared

Saralavama qwot **ORDER** method eke ha **MAP** method eke tiyena vens nam MAP method ek okkoma object eke compare krl blnva but order method ek object 2k vitri compare krnne. E qwe assume apita oredre by eken object tika descending order ekt hadann oni. Ehidi internally apita object okkoma ekinekata compare krl tama eke order ek hadann oni. So api eyat use krnne **MAP** ek. But assume apita oni object dekak compare krl e deken vishalam ek ganna. Ehidi internally okkoma object apita call krnn avashya na. just e object deka vitrk compare karama ati. E vge velavat api use krnne **ORDER** method ek

Example: -

```
CREATE TYPE Customer_typ AS OBJECT (
id NUMBER,
name VARCHAR2(20),
addr VARCHAR2(30),
ORDER MEMBER FUNCTION match (c Customer_typ) RETURN INTEGER
); /
```

```
CREATE TYPE BODY Customer_typ AS
        ORDER MEMBER FUNCTION match (c Customer_typ)
        RETURN INTEGER IS
        BEGIN
                IF id < c.id
                        THEN RETURN -1;
                ELSIF id > c.id
                        THEN RETURN 1;
                ELSE
                        RETURN 0;
                END IF;
        END;
END; /
```

Methods on nested tables

Meke krnne saralavama methods nested table ekak atule implement krn vidiya balana eka.

Example: -

```
CREATE TYPE proj_t AS OBJECT (
	projno number,
	Projname varchar(15)
);
/

CREATE TYPE proj_list AS TABLE OF proj_t;




CREATE TYPE emp_t AS OBJECT (
	eno number,
	projects proj_list,
	MEMBER FUNCTION projcnt RETURN INTEGER
);
/
```

Meke tiyna aINTO key word ek "=" ekt samana venva. E qwe p.projno count eka pcount valat assign venva

```
CREATE OR REPLACE TYPE BODY emp_t AS MEMBER
	FUNCTION projcnt RETURN INTEGER IS
	pcount INTEGER;
	BEGIN
		SELECT count(p.projno) INTO pcount
		FROM TABLE(self.projects) p;
		RETURN pcount;
	END;
END;
 /
```

```
CREATE TABLE emptab OF emp_t
        (Eno PRIMARY KEY)
        NESTED TABLE projects STORE AS emp_proj_tab;
/
```

```
SELECT e.eno, e.projcnt() projcount
FROM emptab e;
/
```

INHERITANCE

ORDB vala object type 4i.

Final or Not Final

instantiable or not instantiable (pahala kata krnva meka gana**)**

By default, ek **Final** ha **instantiable** ve. Apita oni nm eya Not **Final** lesa ho **Not instantiable** define krnn puluvn. To permit subtypes (sub type valat avasara dimata), Inheritance vala all parent objects, not final viya yutui. Sub type final ve.

EX: - **CREATE TYPE Person_type AS OBJECT (pid NUMBER, name VARCHAR2(30), address VARCHAR2(100)) NOT FINAL.**

Apita final or not final lesa switch venn oni nm **ALTER TYPE** keyword ek use krnn puluvn.

Ex: - ALTER TYPE Person_type FINAL;

Creating object

CREATE TYPE Person_type AS OBJECT (pid NUMBER, name VARCHAR2(30), address VARCHAR2(100)) NOT FINAL/

Creating Sub Type object

CREATE TYPE Student_type UNDER Person_type (deptid NUMBER, major VARCHAR2(30)) NOT FINAL; /

CREATE TYPE Employee_type UNDER Person_type (empid NUMBER, mgr VARCHAR2(30)); /

(Mehi api person type ek inherit krl child type dekak hadan tiynva. But mehi student_type not final krl employee_type final krl. Eyata hetuva student_type ek part_time_student type eken aai inherit krn nisa. E qwe student_type kiynnr part_time_student eke parent nisa)

CREATE TYPE PartTimeStudent_type UNDER Student_type (numhours NUMBER);

Creating Table

Table hadaddi apita akara dekkt krnn puluvn. Root level ekata ha leaf level ekt table hadann puluvn. Root level ekt table ekk hadan hati api blmu,

Create table person_tab of person_type (pid primary key).
Create table employee_tab of person_type (pid primary key).

Inserting Data

Insert into person_tab values (student_type(4, 'Edward Learn', '65 Marina Blvd, Ocean Surf, WA, 6725', 40, 'CS'))/

Insert into employee_tab values (Employee_type (4, 'Edward Learn', '65 Marina Blvd, 6725', 40,’CS’))/

Retrieve Data

1. **SELECT VALUE(p) FROM person_tab p;**

```
Person_type(21937, ‘Fred’, ‘4 Ambrose Street’)
Student_type(27362, `Peter’, … , 21, ‘Oragami’)
PartTimeStudent_type(2134, ‘Jack’,…, 13, ‘Physics’, 5)

Person_type(21362, ‘Mary’, …)
Student_type(18437, `Susan’, … , 13, ‘Maths’)
PartTimeStudent_type(4318, ‘Jill’,…, 21, ‘Pottery’, 2)

Person_type(39374, ‘George’, …)
```

(Using the **VALUE ()** function to select all instances of a supertype)

2. **SELECT VALUE(s) FROM person_tab s WHERE VALUE(s) IS OF (Student_type);**

```
Student_typ(27362, `Peter’, … , 21, ‘Oragami’)
PartTimeStudent_type(2134,’Jack’,…,13,‘Physics’, 5)

Student_typ(18437, `Susan’, … , 13, ‘Maths’)
PartTimeStudent_type(4318,’Jill’,…,21,‘Pottery’, 2)
```

(**IS OF** key word eken krnne boundary ekk dana ek. Meke student type ek ha ehi subtype vala vitrk data retrieve krnna kyai. (From student type and its subtypes))

3. **SELECT VALUE(s) FROM person_tab s WHERE VALUE(s) IS OF (ONLY student_type);**

```
Student_typ(27362, `Peter', … , 21, 'Oragami')
Student_typ(18437, `Susan', … , 13, 'Maths')
```

(**ONLY** key word ek specific table eken vitrk data retrieve karai, (From student type but not from subtypes))

4. **SELECT Name, TREAT(VALUE(p) AS PartTimeStudent_type).numhours hours FROM person_tab p WHERE VALUE(p) IS OF (ONLY PartTimeStudent_type);**

```
NAME   Hour
-----------------------------------------------
Jack       5
Jill         2
```

(TREAT () function to make the system treat each person as a part-time student to access the subtype attribute numhours:,)
(apita part-time studentsl 'sub type ekak' vitrk consider krl data retrieve krnn oni nm mema Treat () function ek use kri, saralavam qwot sub type attribute access krnn oni nm api treat function ek use krnva ema nattan treat function ek avashya na)

apita kisima widiyak na sub type ekaka attribute access krnna meka nathuwa. mokda
select numHours from person_tab r where value(r) is of (only partTime_student)
me widiyata access krnna puluwan super type eke ewa withrai. sub type wala one nam treat eka one

1. NOT INSTANTIABLE Types

This is similar to the abstract class in OO.

CREATE TYPE Address_typ AS OBJECT(...) NOT INSTANTIABLE NOT FINAL; *
CREATE TYPE AusAddress_typ UNDER Address_typ(...);
CREATE TYPE IntlAddress_typ UNDER Address_typ(...);

You cannot create instances of the Address_typ only (similar to "abstract classes" in OO). Saralavam qwot Instantiable kiynne we can create an object. Not Instantiable kiynne We can't create an object. Mehi object kiynne create type object ek nevei. Ek hriyata class ekk vge. Api e class eken hadagann object tama table eke save krnne. Itin ema save krnna object create krnn ba kiyn ek tama mehi kiynne. Asamanyayen Not Instatntaible object ekk Not final viya yutui. Otherwise ek inherit krnn ba.

> not instantiable eken apita child la hadla e childlagen object hdnna puluwan. kelinma ara not instatiable eken object hadnna ba

2. NOT INSTANTIABLE Methods

Abstract class ekk abstract method tiynva vge not instantiable type ekk, not instantiable methods tiynva. Mema methods vala body ekk na. subtype eken tama eva override krnne. So not instantiable method ekk not final viya ytui. Otherwise, eva inherit krnn ba. Types vala vge methods vala by default type final venne na. apita final krnn oni nm explicitly ek mention krnn oni. Saralavam qwot method override krnne not instantiable method haraha

Method overriding

```
EX: - CREATE TYPE MyType AS OBJECT (
        MEMBER PROCEDURE Print,
        FINAL MEMBER FUNCTION foo (x NUMBER))
      NOT FINAL; /

      CREATE TYPE MySubType UNDER MyType (
        OVERRIDING MEMBER PROCEDURE Print
       ); /
```

(Me example ekt anuva print ek, not instantiable method ekki. So ek override krl tiynva mysubtype type ekedi)

Method overloading

```
Ex:- CREATE TYPE MyType AS OBJECT ( ...,
        MEMBER FUNCTION fun (x NUMBER)
     ) NOT FINAL; /

     CREATE TYPE MySubType UNDER MyType (
        MEMBER FUNCTION fun (x DATE)
      ); /
```

Same function name, different signature