**Object Oriented Database**

Creating and demonstrating object types with attributes and methods can be done in Oracle using the object-relational model. In Oracle, you can define object types, create tables based on those types, and then perform insert, update, delete, and query operations. Below is a simplified example using Oracle's object types and SQL:

-- a simple object type 'Person' with attributes and methods

CREATE TYPE Person AS OBJECT (

person\_id NUMBER,

first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

birth\_date DATE,

Create -- Method to calculate age

MEMBER FUNCTION calculate\_age RETURN NUMBER

);

-- Create the type body to implement the method

CREATE TYPE BODY Person AS

MEMBER FUNCTION calculate\_age RETURN NUMBER IS

BEGIN

RETURN TRUNC(MONTHS\_BETWEEN(SYSDATE, birth\_date) / 12);

END;

END;

-- Create a table based on the 'Person' object type

CREATE TABLE People OF Person;

-- Insert data into the table

INSERT INTO People VALUES (1, 'John', 'Doe', TO\_DATE('1990-05-15', 'YYYY-MM-DD'));

INSERT INTO People VALUES (2, 'Jane', 'Smith', TO\_DATE('1985-08-21', 'YYYY-MM-DD'));

-- Query data and use the method

SELECT person\_id, first\_name, last\_name, birth\_date, p.calculate\_age() AS age

FROM People p;

-- Update data

UPDATE People SET first\_name = 'Jonathan' WHERE person\_id = 1;

-- Delete data

DELETE FROM People WHERE person\_id = 2;

create a nested table in Oracle and demonstrate how to insert data into it and perform queries. In this example, we'll create a nested table within a type to represent phone numbers for individuals.

-- Create a type for phone numbers

CREATE TYPE PhoneNumber AS OBJECT (

phone\_type VARCHAR2(20),

phone\_number VARCHAR2(15)

);

-- Create a nested table type

CREATE TYPE NestedTableType AS TABLE OF PhoneNumber;

-- Create a type for a person with a nested table of phone numbers

CREATE TYPE Person AS OBJECT (

person\_id NUMBER,

first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

phones NestedTableType

) NOT FINAL;

-- Create a table based on the Person object type

CREATE TABLE People OF Person;

-- Insert data into the table

INSERT INTO People VALUES (

1,

'John',

'Doe',

NestedTableType(PhoneNumber('Home', '123-456-7890'), PhoneNumber('Work', '987-654-3210'))

);

INSERT INTO People VALUES (

2,

'Jane',

'Smith',

NestedTableType(PhoneNumber('Mobile', '555-1234-5678'))

);

-- Query data from the table

SELECT person\_id, first\_name, last\_name,

COLUMN\_VALUE.phone\_type AS phone\_type,

COLUMN\_VALUE.phone\_number AS phone\_number

FROM People p, TABLE(p.phones) COLUMN\_VALUE;

-- Query data with nested table method

SELECT person\_id, first\_name, last\_name, phones.COUNT() AS num\_phones

FROM People;

In this example:

* We create a PhoneNumber object type to represent a phone number.
* We create a NestedTableType as a nested table type of PhoneNumber.
* We create a Person object type with a nested table of phone numbers.
* The People table is created based on the Person type.
* Data is inserted into the table using the nested table type.
* Two SELECT queries demonstrate how to query data from the nested table.

**Multimedia Database**

-- Create a table with a multimedia attribute (BLOB)

CREATE TABLE MultimediaTable (

item\_id NUMBER PRIMARY KEY,

item\_name VARCHAR2(100),

multimedia\_content BLOB

);

-- Insert data with multimedia content

INSERT INTO MultimediaTable VALUES (

1,

'Image1',

EMPTY\_BLOB()

);

-- Inserting actual multimedia content (replace 'your\_image\_path' with the actual path to your image file)

DECLARE

image\_blob BLOB;

BEGIN

DBMS\_LOB.createtemporary(image\_blob, FALSE);

DBMS\_LOB.fileopen(image\_blob, 'your\_image\_path', 'rb');

DBMS\_LOB.loadfromfile(multimedia\_content, image\_blob, DBMS\_LOB.getLength(image\_blob));

DBMS\_LOB.fileclose(image\_blob);

END;

/

-- Query to retrieve multimedia content

SELECT item\_id, item\_name, multimedia\_content

FROM MultimediaTable;

-- Update multimedia content (replace 'your\_updated\_image\_path' with the actual path to your updated image file)

DECLARE

updated\_image\_blob BLOB;

BEGIN

DBMS\_LOB.createtemporary(updated\_image\_blob, FALSE);

DBMS\_LOB.fileopen(updated\_image\_blob, 'your\_updated\_image\_path', 'rb');

DBMS\_LOB.loadfromfile((SELECT multimedia\_content FROM MultimediaTable WHERE item\_id = 1), updated\_image\_blob, DBMS\_LOB.getLength(updated\_image\_blob));

DBMS\_LOB.fileclose(updated\_image\_blob);

END;

/

-- Query to retrieve updated multimedia content

SELECT item\_id, item\_name, multimedia\_content

FROM MultimediaTable;

-- Inserting actual multimedia content (replace 'your\_image\_path' with the actual path to your image file)

DECLARE

image\_blob BLOB;

lob\_locator BLOB;

BEGIN

DBMS\_LOB.createtemporary(image\_blob, FALSE);

lob\_locator := DBMS\_LOB.fileopen(image\_blob, 'your\_image\_path', 'rb');

DBMS\_LOB.loadfromfile(multimedia\_content, lob\_locator, DBMS\_LOB.getLength(lob\_locator));

DBMS\_LOB.fileclose(lob\_locator);

END;

/

* I added a lob\_locator variable to store the LOB locator obtained from DBMS\_LOB.fileopen.
* I replaced DBMS\_LOB.fileopen(image\_blob, 'your\_image\_path', 'rb') with lob\_locator := DBMS\_LOB.fileopen(image\_blob, 'your\_image\_path', 'rb').
* I replaced DBMS\_LOB.fileclose(image\_blob) with DBMS\_LOB.fileclose(lob\_locator).

Please replace 'your\_image\_path' with the actual path to your image file in both instances.

**TEMPORAL DATABASE**

[A] Create a table tblEmp\_Appnt, which stores the account number, name, and valid time say,

recruitment date and retirement date. Insert 10 records in the table.

EXECUTION:

Creation of the table tblEmp\_Appnt:

create table tblEmp\_Appnt

(

account\_number int primary key,

name varchar(20),

recruitment\_date date,

retirement\_date date

);

Table created.

Inserting 10 records in the tblEmp\_Appnt table:

insert into tblEmp\_Appnt values(1,'Pranav',to\_date('2 3 2001','dd mm yyyy'), to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(2,'Prathamesh',to\_date('2 3 2001','dd mm yyyy'), to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(3,'Nikita',to\_date('2 3 2001','dd mm yyyy'),to\_date('12 12 2020','dd mm yyyy'));

> insert into tblEmp\_Appnt values(4,'Chaitali',to\_date('2 3 2001','dd mm yyyy'),to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(5,'Dimpy',to\_date('31 1 2004','dd mm yyyy'),to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(6,'Prem',to\_date('22 8 2006','dd mm yyyy'),to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(7,'Praful',to\_date('12 9 2008','dd mm yyyy'),to\_date('12 12 2020','dd mm yyyy'));

1 row created.

SQL> insert into tblEmp\_Appnt values(8,'Sreekesh',to\_date('31 1 2009','dd mm yyy'),to\_date('2 3 2001','dd mm yyyy'));

**(i)** **Find** **all the employees who join the company on 2/3/2001.**

SQL> select account\_number, name, recruitment\_date from tblEmp\_Appnt where recruitment\_date=to\_date('2 3 2001','dd mm yyyy');

**(i)** **Find all the employees who will retire on 2/3/2001.**

SQL> select account\_number, name, retirement\_date from tblEmp\_Appnt where retirement\_date=to\_date('2 3 2001','dd mm yyyy');

**XML DATABASE**

**Aim:** To create an XML application.

CJoireate a table employee having dept\_id as number datatype and employee\_spec as XML datatype (XMLType).The employee\_spec is a schema with attributes emp id, name, email, acc\_no, managerEmail, dateOfning.

**EXECUTION:**

* **Creation of the global conceptual schema employee :**

create table employee

(

dept\_id number,

employee\_spec xmltype

);

Table created.

* **Inserting 10 records in the employee table:**

insert into employee values(1,xmltype.createxml('

<emp>

<empid> 1 </empid>

<name> pranav </name>

<email> [pranav@gmail.com](mailto:pranav@gmail.com) </email>

<acc\_no> ac0001 </acc\_no>

[<managerEmail>leena@gmail.com</managerEmail](mailto:<managerEmail>leena@gmail.com</managerEmail)>

<dateofjoning> 1/1/2010 </dateofjoning>

</emp>'));

1 row created.

insert into employee values(1,xmltype.createxml(

'<emp>

<empid> 2 </empid>

<name>prathamesh</name>

<email> [prathamesh@gmail.com](mailto:prathamesh@gmail.com) </email>

<acc\_no> ac0002 </acc\_no>

<managerEmail> [leena@gmail.com](mailto:leena@gmail.com) </managerEmail>

<dateofjoning> 1/1/2010 </dateofjoning>

</emp>'));

1 row created.

insert into employee values(1,xmltype.createxml(

'<emp>

<empid> 3 </empid>

<name> nikita </name>

<email> [nikita@gmail.com](mailto:nikita@gmail.com)</email>

<acc\_no> ac0003 </acc\_no>

<managerEmail> [leena@gmail.com](mailto:leena@gmail.com) </managerEmail>

<dateofjoning> 1/1/2010 </dateofjoning>

</emp>'));

* **Fire the following queries:**

**Find the names of all employees.**

select e.employee\_spec.extract('/emp/name/text()').getStringVal()

as Name from employee e ;

**Retrieve the acc\_no of the employees.**

select e.employee\_spec.extract('/emp/acc\_no/text()').getStringVal() as Account\_Number from employee e;

**Retrieve the names,acc\_no, email of employees.**

select e.employee\_spec.extract('/emp/name/text()').getStringval() as Name, e.employee\_spec.extract('/emp/acc\_no/text()').getStringval() as Acc\_No,

e.employee\_spec.extract('/emp/email/text()').getStringVal() as Email from employee e;

**Update the 3rd record from the table and display the name of an employee.**

update employee e

set employee\_spec = xmltype.createxml(

'<emp> <empid> 3 </empid> <name> nikita </name> <email> [nikita@gmail.com](mailto:nikita@gmail.com) </email> <acc\_no> ac0003 </acc\_no> <managerEmail> [nilesh@gmail.com](mailto:nilesh@gmail.com) </managerEmail> <dateofjoning> 1/1/2010 </dateofjoning> </emp>')

where e.employee\_spec.extract('/emp/empid/text()').getStringVal()=3;

**Delete 4th record from the table.**

delete from employee e

where e.employee\_spec.extract('/emp/empid/text()').getStringVal() = 3;

**ACTIVE DATABASE**

**Aim:** To implement an Active database using PL/SQL.

Create a table emp (eno, ename, hrs, pno, super\_no) and project (pname, pno, thrs, head\_no) where thrs is the total hours and is the derived attribute. Its value is the sum of hrs of all employees working on that project. eno and pno are primary keys, head\_no is foreign key to emp relation. Insert 10 tuples into each table.

**EXECUTION:**

* **Creation of the proj table in the database:**

SQL>create table proj

2 (

3 pno int not null,

4 pname varchar(20),

5 thrs int,

6 primary key (pno)

7 );

Table created.

* **Creation of the emp table in the database:**

SQL>create table emp

2 (

3 eno int not null,

4 ename varchar(20),

5 hrs int,

6 pno int,

7 primary key (eno),

8 foreign key (pno) references proj

9 );

Table created.

SQL>insert into proj values(01,'Blood Bank',20);

1 row created.

SQL>insert into proj values(02,'ERP System',20);

1 row created.

SQL>insert into proj values(03,'Construction',20);

1 row created.

SQL>insert into proj values(04,'Cop Portal',20);

1 row created.

SQL>insert into proj values(05,'Inventory',20);

* **Inserting 10 records in the emp table:**

SQL>insert into emp values(01,'Nikita',20,01);

1 row created.

SQL>insert into emp values(02,'Chaitali',20,02);

1 row created.

SQL>insert into emp values(03,'Aakanksha',20,03);

1 row created.

SQL>insert into emp values(04,'Pranav',20,04);

1 row created.

SQL>insert into emp values(05,'Bhaagu',20,05);

1 row created.

* **View records from the proj table:**

SQL>select \* from proj;

* **View records from the emp table:**

SQL>select \* from emp;

* **Creating triggers to do the following:**

1. **Creating a trigger to insert a new employee tuple and display the new total hours from project table.**

SQL>create trigger thrsupdate

2 after insert on emp

3 for each row

4 begin

5 update proj

6 set thrs=thrs+:new.hrs

7 where pno=:new.pno;

8 end;

/

SQL> insert into emp values(11,'Niki',10,01);

1 row created.

SQL> select \* from proj;

1. **Creating a trigger to change the hrs of an existing employee and display the new total hours from project table.**

SQL>create trigger hrsupdate

2 after update of hrs on emp

3 for each row

4 begin

5 update proj

6 set thrs=thrs+:new.hrs-:old.hrs

7 where pno=:new.pno;

8 end;

/

SQL>update emp

2 set hrs=15

3 where eno=10;

SQL>select \* from proj;