EX.NO: 01

Date: 16.08.2024

REFERENTIAL INTEGRITY

Aim:

To write a PL/SQL program for Referential Integrity.

Procedure:

Step1: Create two tables.

Step2: Insert values to the suitable table.

Step3: Use describe function to display the table.

Step4: While inserting the values into the table the error message will be displayed.

Coding:

Create table supply_vid55(supplier_id numeric(10) not null, supplier_name varchar(10) not null,contact_name varchar(10),constraint supply_vid55_pk primary key(supplier_id));

Table created

Create table product_vid55(product_id numeric(10) not null, supplier_id numeric(10) not null,constraint pk_supply_vid55_pk foreign key(supplier_id) references supply_vid55(supplier_id));

Table created

Desc supply_vid55;

Name	**Null?**	**Type**
SUPPLIER_ID	**NOTNULL**	**NUMBER**
SUPPLIER_NAME	**NOTNULL**	**VARCHAR(10)**
CONTACT_NAME		**VARCHAR(10)**

Insert into supply_vid55 values(01,'Abhinaya',2349875002);

1 row created

Insert into supply_vid55 values(02,'Bhaviya',9884455332);

1 row created

Insert into supply_vid55 values(03,'Dharani',9457321023);

1 row created

Select* from supply_vid55;

SUPPLIER_ID	SUPPLIER_NAME	CONTACT_NAME
1	Abhinaya	2349875002
2	Bhaviya	9884455332
3	Dharani	9457321023

Desc product_vid55;

Name	null?	Type
PRODUCT_ID	NOTNULL	NUMBER(10)
SUPPLIER_ID	NOTNULL	NUMBER(10)

Insert into product_vid55 values(100,1);

1 row created

Insert into product_vid55 values(100,1);

Error at line 1:

ORA-002291:integrity constraint(GENERAL PK_SUPPLY_VID55)violated-parent key not found

EX.NO: 02

Date: 22.08.2024

TRIGGERS

Aim:

To write a PL/SQL program for Trigger function.

Procedure:

Step1: Create a trigger before insert or update on a table.

Step2: Update the table data based on the necessary conditions.

Step3: When the trigger is violated raise an error.

Coding:

```
Create table emp03_vid(emp_id  varchar(10), emp_name varchar(10) ,salary number(7));
Table created
Insert into emp03_vid values(01,'meena',20000);
1 row created
Insert into emp03_vid values(02,'priya',25000);
1 row created
Insert into emp03_vid values(03,'naveen',30000);
1 row created
Insert into emp03_vid values(04,'lalitha',55000);
1 row created
Insert into emp03_vid values(05,'arun',80000);
1 row created
Select* from emp03_vid;
Create or replace trigger trigday
Before insert or update on emp03_vid
For each row
Declare
Empid varchar(10);
```

```
Begin
Empid: =:new.salary
If(empid-‘20000)’ then
Raise_application_error(-20011,’salary not allowed’);
End if;
End;
/
```

EMP_ID	EMP_NAME	SALARY
1	meena	20000
2	priya	25000
3	naveen	30000
4	lalitha	55000
5	arun	80000

Error report:

ORA-04095: trigger ‘TRIGDAY’ already exits on anther table, cannot replace it

Cause: Cannot replace a trigger which already exists on a different table than the one being replaced.

Action: Drop the trigger with the same name and re-create it.

EX.NO: 03

Date: 06.09.2024

IMPLICIT AND EXPLICIT CURSORS

A)Implicit Cursors

Aim:

To write a PL/SQL program for cursors.

Procedure:

Step1: Create rowtype variable for cursor.

Step2: Create a cursor to select the values from the table.

Step3: Open the cursor. Using a loop fetch the values into the cursor.

Step4: Display the output after the necessary manipulation.

Step5: Close the cursor and exit the program.

Coding:

```
create table vid_friend(id number(10),name varchar(10), address varchar(20));
create table succeeded.
insert into vid friend values(1, 'barani', 'coimbatore');
1 rows inserted
insert into vid_friend values (2, 'malathi', 'tirupur');
1 rows inserted
insert into vid_friend values(3,'sana', 'kumbakonam');
1 rows inserted
insert into vid_friend values(4,'shoba','ooty');
1 rows inserted
insert into vid_friend values (5,'priyanka', 'palani');
1 rows inserted
declare
c_id vid_friend.id%type;
c_name vid_friend.name%type;
c_address vid_friend.address%type;
```

```
cursor c_vid_friend is
select id, name, address from vid_friend;
begin
open c_vid_friend;
loop
fetch c_vid_friend into c_id,c_name,c_address;
exit when c_vid_friend%not found;
dbms_output.put_line(c_id||' '||c_name||' '||c_address);
end loop;
close c_vid_friend;
end;
/
set serveroutput on;
```

Output:

anonymous block completed

1	barani	coimbatore
2	malathi	tirupur
3	sana	kumbakonam
4	shoba	ooty
5	priyanka	palani

B)Explicit Cursors

Aim:

To write a PL/SQL program for Explicit cursor.

Procedure:

Step1: Create a cursor and select the table values.

Step2: In a loop fetch the values check for PASS or FAIL Condition.

Step3: Insert PASS record into one table and other record into the other table.

Step4: Display the output table.

Step5: Close the cursor and exit the program.

Coding:

```
create table spilt_vid(regno number(10), name varchar(10), subj1 number(10), subj2
number(10), subj3 number(10), averagepercent number(10));
create table succeeded.
insert into spilt_vid values(001, 'barani',90,86,89,88);
1 rows inserted
insert into spilt_vid values(002,'malathi',87,82,94,88);
1 rows inserted
insert into spilt_vid values(003, sana',75,88,91,85);
1 rows inserted
insert into spilt_vid values(004, 'shoba', 79,95,80,85);
1 rows inserted
insert into spilt_vid values(005, 'priyanka',74,93,78,82);
1 rows inserted
insert into spilt_vid values(006,'vinisha',97,92,84,91);
1 rows inserted
select * from spilt_vid;
```

REGNO	NAME	SUBJ1	SUBJ2	SUBJ3	AVERAGEPERCENT
1	barani	90	86	89	88
2	malathi	87	82	94	88
3	sana	75	88	91	85
4	shoba	79	95	80	85
5	priyanka	74	93	78	82
6	vinisha	84	92	84	91

6 rows selected

```
create table spilt_pass(regno number(10), name varchar(10), subj1 number(10), subj2
number(10), subj3 number(10), averagepercent number(10));
create table succeeded.
create table spilt_fail(regno number(10), name varchar(10), subjl number(10), subj2 number(10),
subj3 number(10), averagepercent number(10));
create table succeeded.
declare
cursor ms is select * from spilt_vid
dos ms%rowtype;
begin
for dos in ms
loop
if(dos.subj 1<50)or(dos.subj2<50)or(dos.subj3<50)
then
insert into spilt_pass values(dos.regno,dos.name,dos.subj 1,dos.subj2,dos.subj3,dos.
averagepercent);
else
Insert into spilt_fail
values(dos.regno,dos.name,dos.subj1,dossubj2,dos.subj3,dos.averagepercent);
end if;
dbms_output.put_line(dos.regno "dos.name); end loop; end;
/
```

set serveroutput on;

Output:

anonymous block completed

1 barani

2 malathi

3 sana

4 shoba

5 priyanka

6 vinisha

EX.NO: 04

Date: 16.09.2024

EXCEPTION HANDLING

Aim:

To write a PL/SQL program for exception handling.

Procedure:

Step1: Declare the necessary variables.

Step2: Get the input from the user.

Step3: Compute answer:=a/b.

Step4: Display the answer else raise the necessary exception.

Step5: Display the output and terminate.

Coding:

```
declare
a int;
b int;
answer int;
begin
a:=&a;
b:=&b;
answer:-a/b;
dbms_output.put_line('The result after division is"] [answer); exception
when zero divide then
dbms_output.put_line('Dividing by zero please check the values again');
dbms_output.put_line('The value of a is' (a);
dbms_output.put_line("The value of b is'| [b);
end;
/
```

Output:

anonymous block completed

The result after division is 1

anonymous block completed

Dividing by zero please check the values again

The value of a is 6

The value of b is 0

EX.NO: 05

Date: 24.09.2024

PROCEDURES

Aim:

To write a PL/SQL program for procedures.

Procedure:

Step1: Create a procedure for inserting values into table.

Step2: In the procedure write the query to insert values.

Step3: The main program call the procedure to insert values into the table.

Step4: Display the necessary output and exit.

Coding:

```
create table user_monvi(id number(10) primary key, name varchar2(100));
create or replace procedure INSERTUSER (id IN NUMBER, name IN VARCHAR2)
IS
BEGIN
insert into user_monvi values(id,name);
end;
begin
insertuser(103, 'jerry');
dbms_output.put_line('record inserted successfully');
end;
select from user_monv;
```

Output:

create table succeeded.

procedure INSERTUSER Compiled.

anonymous block completed

record inserted successfully

ID	NAME
103	Jerry

1 rows selected

EX.NO: 06

Date: 01.10.2024

FUNCTIONS

Aim:

To write a PL/SQL program for functions on table.

Procedure:

Step1: Create a table.

Step2: Insert the necessary data.

Step3: Create a function to count the total number of records.

Step4: Execute the function.

Step5: In the main program call the function to display the result.

Coding:

```
CREATE TABLE TEACHERS_1( ID INT NOT NULL, NAME VARCHAR (20), AGE INT,
ADRESS CHAR (25), SALARY DECIMAL (18,2));
INSERT INTO TEACHERS_1 VALUES (1, 'RANJITHA', 34, 'BANGALORE', 2000.00);
INSERT INTO TEACHERS_1 VALUES (2, 'SAI', 30, 'HYDERABAD', 3000.00);
select * from TEACHERS_1;
CREATE OR REPLACE FUNCTION totalteachers
RETURN number IS
total number(2) := 0;
BEGIN
SELECT count(*) into total
FROM teachers_1;
RETURN total;
END;
```

Output:

CREATE TABLE succedded

1 row inserted

1 row inserted

ID	NAME	AGE	ADDRESS	SALARY
1	RANJITHA	34	BANGALORE	2000
2	SAI	30	HYDERABAD	3000

2 rows selected

FUNCTTION totalteachers Compiled

EX.NO: 07

Date: 17.10.2024

DATABASE CREATION, LISTING AND DROPPING USING MONGODB

Aim:

To demonstrate the basic Queries in MongoDB.

CODING:

Create a Database

```
> use meena
switched to db meena
```

Displaying the existing Databases

```
> show dbs
admin 0.000GB
local 0.000GB
```

Inserting Documents into the Collection

```
>db.meena.insert({"ID":"101","Name":"Meena","Address":"Covai"})
WriteResult({"nInserted": 1 })
> db.meena.insert({"ID":"102","Name":"Lalitha","Address":"Thirupur"))
WriteResult({"nInserted": 1 })
> db.meena.insert({"ID":"103","Name":"Subha","Address":"Sulur"))
WriteResult({"nInserted": 1 })
> db.meena.insert({"ID":"104","Name":"yuvasri","Address":"Avinashi"})
WriteResult({ "nInserted": 1 })
> db.meena.insert({"ID":"105","Name":"Pooja","Address":"Thudiyalur"})
WriteResult({ "nInserted": 1 })
```

Drop

```
> db.dropDatabase();
{ "dropped": "meena", "ok": 1 }
```

EX.NO: 08

Date: 24.10.2024

CRUD OPERATIONS

Aim:

To demonstrate CRUD operations using MongoDB.

Coding:

Create a Database

```
> use student
switched to db student
```

To Insert Document

```
>db.student.insert
        ({
                RegNo:"1001", Name:"MARIA",
                course:
                {
                        CourseName:"MIT", Duration:"2 Years"
                },
                address:
                {
                        City:"BANGALORE", State:"Karanataka", Country: "India"
                }
        })
```

Result:

```
WriteResult(("ninserted": 1))
```

To Retrieve the Documents in the Collections

```
> db.student.find()
```

Result:

```
{"_id" : ObjectId("59f3535828b78ec6e158529b"), "RegNo": "1001", "Name": "MARIA",
"course": { "CourseName": "MIT", "Duration": "2 Years" ), "Address": { "City":
"BANGALORE", "Sate": "Karanataka", "Country": "India")}
```

{"_id" : ObjectId("59f362bc11779e9977328556"), "RegNo": "1001", "Name": "KEERTHI",
"Course": "MIT", "Address": "Coimbatore")
{"_id" : ObjectId("59f362bc11779e9977328557"), "RegNo": "1002", "Name": "POORNI",
"Course": "MIT", "Address": "Coimbatore")
{"_id" : ObjectId("59f362bc11779e9977328558"), "RegNo": "1003", "Name": "PRINCY",
"Course": "MIT", "Address": "Coimbatore"}
{"_id" : ObjectId("59f362bc11779e9977328559"), "RegNo": "1004", "Name": "RENATA",
"Course": "MIT", "Address": "Coimbatore"}
{"_id" : ObjectId("59f362bc11779e997732855a"), "RegNo": "1005", "Name": "MARIA",
"Course": "MIT", "Address": "Coimbatore"}
To Update Document
> db.student.update(("RegNo":"1001"), ($set:{"Name":"KEERTHANA"}})
Result:
WriteResult({"nMatched": 1, "nUpserted": 0, "nModified": 1))
To Remove a Document
> db.student.remove(("Name":"MARIA"))
Result:
WriteResult({"nRemoved": 1))
> db.student.find()
Result:
Document MARIA has been removed
{"_id" : ObjectId("59f3535828b78ec6e158529b"), "RegNo": "1001", "Name": "KEERTHANA",
"course" ("CourseName": "MIT", "Duration": "2 Years"), "Address": {
"City": "BANGALORE", "Sate": "Karanataka", "Country": "India"}}
{"_id" : Objectid("59f362bc11779e9977328556"), "RegNo": "1001", "Name": "KEERTHI",
"Course": "MIT", "Address": "Coimbatore"}
("_id" : ObjectId("59f362bc11779e9977328557"), "RegNo": "1002", "Name": "POORNI",
"Course": "MIT", "Address": "Coimbatore")
{"_id" : ObjectId("59f362bc11779e9977328558"), "RegNo": "1003", "Name": "PRINCY",
"Course": "MIT", "Address": "Coimbatore"}
{"_id" : ObjectId("59f362bc11779e9977328559"), "RegNo": "1004", "Name": "RENATA",
"Course": "MIT", "Address": "Coimbatore")

EX.NO: 09

Date: 28.10.2024

FUNCTIONS IN MONGODB

Aim:

To demonstate Count, Limit, Sort and Skip in MongoDB.

Coding:

Create database javatpointdb

>use javatpointdb

switched to db javatpointdb

> show dbs

abidb 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

mydb 0.000GB

Create Collection

>db.createCollection("javatpoint")

o/p: ("ok": 1)

To check the created collection, use the command "show collections".

> show collections

o/p:

javatpoint

Now listing the currently used database

> show dbs

abidb

0.000GB

admin

0.000GB

config

0.000GB
javatpointdb 0.000GB
local
0.000GB
mydb
0.000GB
INSERT DOCUMENTS
> db.javatpoint.insert({ course: "java", details: { duration:"6 months", Trainer: "Sonoo jaiswal"}})
o/p:
WriteResult({"ninserted" : 1))
>db.javatpoint.insert((course:"net", details: (duration:"6 months", Trainer: "Prashant Verma")))
o/p: WriteResult(("ninserted": 1))
>db.javatpoint.insert((course:"web designing", details: (duration:"3 months", Trainer: "Rashmi")))
o/p: WriteResult(("ninserted": 1))
TO CHECK THE INSERTED DOCUMENTS:
find()
>db.javatpoint.find()
op:
{"_id" : ObjectId("60542a6d75bd7dc893d9b3a3"), "course": "java", "details": ["duration": "6 months", "Trainer": "Sonoo jaiswal" } } {"_id": ObjectId("60542bc575bd7dc893d9b3a4"), "course": "net", "details": { "duration": "6
months", "Trainer": "Prashant Verma")) {"id": ObjectId("60542bfd75bd7dc893d9b3a5"), "course": "web designing", "details": ("duration": "3 months", "Trainer": "Rashmi"}}
limit() method
>db.javatpoint.find().limit(1)
o/p: {"_id" : ObjectId("60542a6d75bd7dc893d9b3a3"), "course": "java", "details": { "duration": "6 months", "Trainer": "Sonoo jaiswal" } }
>db.javatpoint.find().limit(2)

o/p: {"id": ObjectId("60542a6d75bd7dc893d9b3a3"), "course": "java", "details": { "duration": "6 months", "Trainer": "Sonoo jaiswal"}} {"_id" : ObjectId("60542bc575bd7dc893d9b3a4"), "course": "net", "details": ("duration": "6
months", "Trainer": "Prashant Verma"}}

> db.javatpoint.find().limit(3)

o/p:

{"_id" : ObjectId("60542a6d75bd7dc893d9b3a3"), "course": "java", "details": ("duration": "6 months", "Trainer": "Sonoo jaiswal"}}
("_id" : ObjectId("60542bc575bd7dc893d9b3a4"), "course": "net", "details": { "duration": "6 months", "Trainer": "Prashant Verma"}} {"_id" : ObjectId("60542bfd75bd7dc893d9b3a5"), "course": "web designing", "details": { "duration": "3 months", "Trainer": "Rashmi" } }

sort() method

db.javatpoint.find().sort(("course":-1})

op:

"_id" : ObjectId("564dbced8e2c097d15fbb603"), "Course": "Web Designing", "det
ails": { "Duration": "3 months", "Trainer": "Rashmi Desai"), "Batch": [{" size": "Small", "qty": 5 }, ["size": "Large", "qty": 10 }], "category":"
Programming Language") {"_id" : ObjectId("564dbced8e2c097d15fbb601"), "Course": "Java", "details" : ("Duration": "6 months", "Trainer": "Sonoo Jaiswal"), "Batch": [{"size": "Medium", "qty": 25)], "category": "Programming Language"} {"_id" : ObjectId("564dbccd8e2c097d15fbb602"), "Course": ".Net", "details": {
"Duration": "6 months", "Trainer": "Prashant Verma"), "Batch": [{"size": "Small", "qty": 5 }, { "size": "Medium", "qty": 10 }], "category": "Progra mming Language"}

skip() method

db.javatpoint.find().limit(1).skip(2)

o/p:

{"_id" : ObjectId("60542bfd75bd7dc893d9b3a5"), "course": "web designing", "details": {
"duration": "3 months", "Trainer": "Rashmi"}}

count command

> db.runCommand((count:'javatpoint'})

o/p:

{"n":3,"ok":1}

EX.NO: 10

Date: 05.11.2024

AGGREGATE FUNCTION

Aim:

To demonstrate count, average, sum, min, max in MongoDb.

Coding:

CREATE DATABASE:

> use employeetb

switched to db employeetb

INSERTING DOCUMENTS INTO THE COLLECTION:

>db.employeetb.insert({"empid":"EMP001","empname":"Aakash","Designation":"Associate ""Depatment id":"D001", "Salary":25000,"DOJ":"1999-12- 04", "NatureofJob":"Mechanical"})

WriteResult({"nInserted": 1))

>db.employeetb.insert({"empid":"EMP002","empname":"Anand","Designation":"Senior Consultant", "Depatment_id":"D002", "Salary":50000,"DOJ":"new Date()", "NatureofJob":"Technical"))

WriteResult({"nInserted": 1 })

>db.employeetb.insert({"empid":"EMP003","empname":"suresh","Designation":"Developer" "Depatment_id":"D003","Salary":30000,"DOJ":"2000-04-12", "NatureofJob":"Technical"})

WriteResult({ "nInserted": 1 })

USING AGGREGATE FUNCTIONS

COUNT():

>db.employeedb.aggregate([{$match: ("Designation": (Seq:"Developer"}}}, {$count:"TotalCount"}])

{"TotalCount":1}

SUM():

>db.employeetb.aggregate([{$group: {_id: "$Designation", Salary: {$sum: "ŚSalary"}}}])

{"_id": "Developer", "Salary": 30000}

{"_id": "Senior Consultant", "Salary" : 50000}

{"_id": "Associate", "Salary": 25000}

AVG():

>db.employeetb.aggregate([{$group: {_id:"$Designation", Salary: ($avg:"$Salary"}}}])

{"_id": "Developer", "Salary": 30000} {"_id": "Senior Consultant", "Salary" : 50000)

{"_id": "Associate", "Salary": 25000}

MIN():

>db.employeetb.aggregate([{$group: {_id:"$Designation", MinSalary: {$min:"$Salary"}}}])

{"_id": "Developer", "MinSalary": 30000}

{"_id": "Senior Consultant", "MinSalary" : 50000}

{"_id": "Associate", "MinSalary" : 25000 }

MAX():

>db.employeetb.aggregate([{$group: {_id:"$Designation", MaxSalary: {$max:"$Salary"}}}])

{"_id": "Developer", "MaxSalary": 30000)

{"id": "Senior Consultant", "MaxSalary" : 50000}

{"_id": "Associate", "MaxSalary": 25000}

EX.NO: 11

Date: 09.11.2024

PRODUCT CATALOGUE USING AGGREGATE FUNCTION

Aim:

To demonstrate Product catalogue using Aggregate function in MongoDB.

Coding:

```
{
   "_id": ObjectId("..."),
   "name": "Laptop",
   "category": "Electronics",
   "price": 1500,
   "quantity": 25,
   "brand": "Brand A",
   "rating": 4.5,
   "tags": ["technology", "portable", "gadgets"]
}
db.products.aggregate([
   {
      $group: {
         _id: "$category",
         totalProducts: { $sum: 1 }
      }
   },
   { $sort: { totalProducts: -1 } }
])
o/p:
   { "_id": "Electronics", "totalProducts": 50 },
   { "_id": "Furniture", "totalProducts": 30 },
   { "_id": "Clothing", "totalProducts": 20 }
db.products.aggregate([
```

```
  {
    $group: {
      _id: "$category",
      averagePrice: { $avg: "$price" }
    }
  },
  { $sort: { averagePrice: -1 } }
])
o/p:
  { "_id": "Electronics", "averagePrice": 1200 },
  { "_id": "Furniture", "averagePrice": 800 },
  { "_id": "Clothing", "averagePrice": 50 }
db.products.aggregate([
  {
    $group: {
      _id: "$brand",
      minPrice: { $min: "$price" },
      maxPrice: { $max: "$price" }
    }
  }
])
o/p:
  { "_id": "Brand A", "minPrice": 500, "maxPrice": 2000 },
  { "_id": "Brand B", "minPrice": 100, "maxPrice": 1500 },
  { "_id": "Brand C", "minPrice": 80, "maxPrice": 700 }
db.products.aggregate([
  {
    $sort: { quantity: -1 }
  },
  { $limit: 5 }
])
o/p:
```

```
  { "_id": ObjectId("..."), "name": "Smartphone", "quantity": 150 },
  { "_id": ObjectId("..."), "name": "Laptop", "quantity": 120 },
  { "_id": ObjectId("..."), "name": "Tablet", "quantity": 100 },
  { "_id": ObjectId("..."), "name": "Desk Chair", "quantity": 90 },
  { "_id": ObjectId("..."), "name": "Headphones", "quantity": 85 }
db.products.aggregate([
  {
    $group: {
      _id: "$category",
      totalInventoryValue: {
        $sum: { $multiply: ["$price", "$quantity"] }
      }
    }
  },
  { $sort: { totalInventoryValue: -1 } }
])
o/p:
  { "_id": "Electronics", "totalInventoryValue": 200000 },
  { "_id": "Furniture", "totalInventoryValue": 75000 },
  { "_id": "Clothing", "totalInventoryValue": 10000 }
```