IT214 Lab Report LAB-3

Date - September 9 ,2022
Name - Divya Kirtikumar Patel
Student ID - 202001420
Lab Group – 6
Section - 8

I. Create before trigger functions to avoid the common error message.

1. Create a trigger on the Table of your choice to check if the Primary key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.

```
CREATE OR REPLACE FUNCTION rail_db.primary_key_check()
RETURNS trigger
AS $$
DECLARE
temp_id integer;

BEGIN
SELECT tc_id into temp_id from rail_db.ticketcollector tc1 where tc1.tc_id=NEW.tc_id;
IF(temp_id=NEW.tc_id)
THEN
RAISE NOTICE 'Notice : This violates constraint of primary key.';
RETURN OLD;
ELSE
RAISE NOTICE 'Notice : Primary key does not exist in table.';
RETURN NEW;
END IF;
END;
$$ LANGUAGE 'plpgsql';

-- Creating trigger
CREATE OR REPLACE TRIGGER "trigger1"
BEFORE INSERT
ON "rail_db".ticketcollector
FOR EACH ROW
EXECUTE PROCEDURE "rail_db".primary_key_check()

-- Sample insertion for query verification
INSERT INTO ticketcollector VALUES(9,'darsh','mehta',29);
INSERT INTO ticketcollector VALUES(18,'raju','sheth',22);
```

provided tc_id (primary key) is already present in table, hence it will not be inserted.

```
25
26  -- trigger
27  CREATE OR REPLACE TRIGGER "trigger1"
28  BEFORE INSERT
29  ON "rail_db".ticketcollector
30  FOR EACH ROW
31  EXECUTE PROCEDURE "rail_db".primary_key_check()
32
33  -- verification
34  INSERT INTO ticketcollector VALUES(9,'darsh','mehta',29);
35  INSERT INTO ticketcollector VALUES(18,'raju','sheth',22);
36
37
```

Data output | Messages | Notifications

```
NOTICE:  Violating constraint of primary key.
INSERT 0 0

Query returned successfully in 58 msec.
```

provided tc_id (primary key) is already present in table, hence it will not be inserted.

```
25
26  -- trigger
27  CREATE OR REPLACE TRIGGER "trigger1"
28  BEFORE INSERT
29  ON "rail_db".ticketcollector
30  FOR EACH ROW
31  EXECUTE PROCEDURE "rail_db".primary_key_check()
32
33  -- verification
34  INSERT INTO ticketcollector VALUES(9,'darsh','mehta',29);
35  INSERT INTO ticketcollector VALUES(18,'raju','sheth',22);
36
37
```

Data output | Messages | Notifications

```
NOTICE:  Violating constraint of primary key.
INSERT 0 0

Query returned successfully in 45 msec.
```

provided tc_id (primary key) is not present in table, hence it will not be inserted.

```
25
26  -- trigger
27  CREATE OR REPLACE TRIGGER "trigger1"
28  BEFORE INSERT
29  ON "rail_db".ticketcollector
30  FOR EACH ROW
31  EXECUTE PROCEDURE "rail_db".primary_key_check()
32
33  -- verification
34  INSERT INTO ticketcollector VALUES(9,'darsh','mehta',29);
35  INSERT INTO ticketcollector VALUES(18,'raju','sheth',22);
36  INSERT INTO ticketcollector VALUES(34,'mohan','shah',22);
37
38
```

Data output | Messages | Notifications

```
NOTICE:  Notice : Primary key does not exist in table.
INSERT 0 1

Query returned successfully in 61 msec.
```

2. Create a trigger on the Table of your choice to check if the Foreign key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.

```
-- trigger function
CREATE OR REPLACE FUNCTION rail_db.foreign_key_check()
RETURNS trigger
AS $$
DECLARE
temp_id integer;
BEGIN
SELECT tc_id into temp_id from rail_db.train t1 where t1.tc_id=NEW.tc_id;
IF(temp_id=NEW.tc_id)
THEN
RAISE NOTICE 'Notice : Foreign key already exists in this table.';
RETURN OLD;
ELSE
RAISE NOTICE 'Notice : Foreign key is not present in this table.';
RETURN NEW;
END IF;
END;
$$ LANGUAGE 'plpgsql';

-- trigger
CREATE OR REPLACE TRIGGER "trigger2"
BEFORE INSERT
ON "rail_db".train
FOR EACH ROW
EXECUTE PROCEDURE "rail_db".foreign_key_check()

--verification
INSERT INTO train VALUES(201,'Rajdhani Express','10:21:00','02:30:00',12,'2022-08-
11',14);
INSERT INTO train VALUES(202,'Vikas Express','09:05:00','04:30:00',12,'2022-08-
11',31);
```

```
61  -- Creating trigger
62  CREATE OR REPLACE TRIGGER "trigger2"
63  BEFORE INSERT
64  ON "rail_db".train
65  FOR EACH ROW
66  EXECUTE PROCEDURE "rail_db".foreign_key_check()
67
68  -- Sample insertion for query verification
69  INSERT INTO train VALUES(201,'Rajdhani Express','10:21:00','02:30:00',12,'2022-08-11',14);
70  INSERT INTO train VALUES(202,'Vikas Express','09:05:00','04:30:00',12,'2022-08-11',31);
71
72
```

Data output | Messages | Notifications

```
NOTICE:  Notice : Foreign key already exists in this table.
INSERT 0 0

Query returned successfully in 41 msec.
```

```
61  -- Creating trigger
62  CREATE OR REPLACE TRIGGER "trigger2"
63  BEFORE INSERT
64  ON "rail_db".train
65  FOR EACH ROW
66  EXECUTE PROCEDURE "rail_db".foreign_key_check()
67
68  -- Sample insertion for query verification
69  INSERT INTO train VALUES(201,'Rajdhani Express','10:21:00','02:30:00',12,'2022-08-11',14);
70  INSERT INTO train VALUES(202,'Vikas Express','09:05:00','04:30:00',12,'2022-08-11',31);
71
```

Data output | Messages | Notifications

```
NOTICE:  Notice : Foreign key is not present in this table.
INSERT 0 1

Query returned successfully in 95 msec.
```

II. Create functions for a specific section.

1. Find the ticket id for an amount greater than 2000. Return a temp table with ticket id, amount and status in the result table. Make sure below both records are visible in results.

```
--tickets with id greater than 2000
SELECT ticket_id FROM ticket WHERE amount>2000 ORDER BY ticket_id;
--trigger function
CREATE OR REPLACE FUNCTION rail_db.func_temp1_Q2()
RETURNS TABLE(a integer, b integer , c character(30))
AS $$
DECLARE
TEMP_R_LIST record;
BEGIN
CREATE TEMP TABLE test (a1 integer, b1 integer, c1 character(30)) ON COMMIT DROP;
FOR TEMP_R_LIST IN (select ticket_id,amount,status from rail_db.ticket)
LOOP
IF(TEMP_R_LIST.amount>2000)
THEN
INSERT INTO test (a1, b1, c1) VALUES
(TEMP_R_LIST.ticket_id,TEMP_R_LIST.amount,TEMP_R_LIST.status);
END IF;
END LOOP;
RETURN QUERY TABLE test;
END;
$$ LANGUAGE 'plpgsql';
SELECT ALL rail_db.func_temp1_Q2()
```

```
77  SELECT ticket_id FROM ticket WHERE amount>2000 ORDER BY ticket_id;
78  --trigger function
79  CREATE OR REPLACE FUNCTION rail_db.func_temp1_Q2()
80  RETURNS TABLE(a integer, b integer , c character(30))
81  AS $$
```

Data output | Messages | Notifications

	ticket_id integer
1	1000001
2	1000002
3	1000003
4	1000004
5	1000006
6	1000007
7	1000008
8	1000009
9	1000010

Total rows: 123 of 123 | Query complete 00:00:00.163

```
$$ LANGUAGE 'plpgsql' ;

SELECT ALL rail_db.func_temp1_Q2()

--2. Create a new column, "total_amount" in the ticket table. Call the function to calculate the total amou
```

Data output | Messages | Notifications

	func_temp1_q2 record
1	(1000001,4592,"confirm ")
2	(1000002,4733,"confirm ")
3	(1000003,2620,"confirm ")
4	(1000004,3946,"confirm ")
5	(1000006,3103,"waiting ")
6	(1000007,2979,"waiting ")
7	(1000008,3044,"waiting ")
8	(1000009,2176,"confirm ")
9	(1000010,3503,"waiting ")

Total rows: 123 of 123 | Query complete 00:00:00.067

2. Create a new column, "total_amount" in the ticket table. Call the function to calculate the total amount with the formula amount+0.12*amount. Display the updated table.

```
-- new attribute total_amount
ALTER TABLE ticket ADD total_amount decimal(7,2);

-- trigger function
CREATE OR REPLACE FUNCTION rail_db.func_temp2_Q2()
RETURNS void
AS $$
DECLARE
TEMP_R_LIST record;
BEGIN
FOR TEMP_R_LIST IN (select ticket_id,amount,total_amount FROM rail_db.ticket)
LOOP
UPDATE rail_db.ticket
SET total_amount= TEMP_R_LIST.amount + 0.12*TEMP_R_LIST.amount WHERE
TEMP_R_LIST.ticket_id=ticket_id;
END LOOP;
END;
$$ LANGUAGE 'plpgsql';

SELECT rail_db.func_temp2_Q2();
SELECT * FROM ticket ORDER BY ticket_id;
```

```
117
118 SELECT rail_db.func_temp2_Q2();
119 SELECT * FROM ticket ORDER BY ticket_id;
120
```

Data output | Messages | Notifications

	ticket_id integer	passenger_id integer	status character (30)	amount integer	train_id integer	total_amount numeric (7,2)
1	1000001	99	confirm	4592	101	5143.04
2	1000002	100	confirm	4733	102	5300.96
3	1000003	101	confirm	2620	103	2934.40
4	1000004	102	confirm	3946	104	4419.52
5	1000005	103	confirm	1546	105	1731.52
6	1000006	104	waiting	3103	106	3475.36
7	1000007	105	waiting	2979	107	3336.48
8	1000008	106	waiting	3044	108	3409.28
9	1000009	107	confirm	2176	109	2437.12

Total rows: 150 of 150 | Query complete 00:00:00.136

III. Create Trigger functions for a specific section.

1. Create a column “state” in the station table. Create a trigger to put the default value 1 for every new entry if nothing is given by the user. Insert new records and check the functionality of Triggers.

```
-- new attribute state
ALTER TABLE station ADD state INTEGER

-- trigger function
CREATE OR REPLACE FUNCTION rail_db.state_update()
RETURNS trigger
AS $$
DECLARE
temp_state int;
BEGIN
SELECT state INTO temp_state from rail_db.station s where
s.station_id=NEW.station_id and state is not null;
update station set state=1 where NEW.station_id=station_id;
if(temp_state<>1) then update station set state=temp_state WHERE
NEW.station_id=station_id;
end if;
return new;
END;
$$ LANGUAGE 'plpgsql';

-- trigger
CREATE OR REPLACE TRIGGER "t3"
AFTER INSERT
ON rail_db.station
FOR EACH ROW
EXECUTE PROCEDURE rail_db.state_update();

INSERT INTO station VALUES(103,'BM6 HGF','09:15:00',202,'Yes');
INSERT INTO station VALUES(101,'BHF H68','11:15:00',202,'Yes',5);

SELECT * FROM station
```

```
148
149 INSERT INTO station VALUES(103,'BM6 HGF','09:15:00',202,'Yes');
150 INSERT INTO station VALUES(101,'BHF H68','11:15:00',202,'Yes',5);
151
152 SELECT * FROM station
153
154 -- 2. Create a new table del_train with column tr_id. Create a new Trigger and call after deletion. After each trair
155 CREATE TABLE del_train(tr_id integer)
```

Data output | Messages | Notifications

	station_id integer	name character (50)	arrival_time time without time zone	train_id integer	hault character (30)	state integer
94	94	E3L 5A8	07:31:00	194	Yes	[null]
95	95	X1U 6C6	07:16:00	195	No	[null]
96	96	O2M 2Q9	11:23:00	196	Yes	[null]
97	97	D9B 2T9	12:03:00	197	Yes	[null]
98	98	P0S 2P9	20:45:00	198	No	[null]
99	99	W6P 5U8	16:53:00	199	Yes	[null]
100	100	J5C 4K7	05:23:00	200	No	[null]
101	103	BM6 HGF	09:15:00	202	Yes	1
102	101	BHF H68	11:15:00	202	Yes	5

Total rows: 102 of 102 | Query complete 00:00:00.132

2. Create a new table del_train with column tr_id. Create a new Trigger and call after deletion. After each train deletion it should enter the tr_id to the del_train table. Check for at least 3 records and print the del_train table.

```
CREATE TABLE del_train(tr_id integer);

-- trigger function
CREATE OR REPLACE FUNCTION rail_db.train_backup()
RETURNS TRIGGER
AS $$
BEGIN
INSERT INTO del_train VALUES(OLD.train_id);
RETURN OLD;
END;
$$ LANGUAGE 'plpgsql';

-- trigger
CREATE OR REPLACE TRIGGER "trigger_train_backup"
AFTER DELETE
ON train
FOR EACH ROW
EXECUTE PROCEDURE rail_db.train_backup();

--Inserting some extra data
INSERT INTO train(train_id,tc_id) VALUES(201,4),(202,2),(203,8),(204,9),(205,11);


--Deleting data from train
DELETE FROM train WHERE train_id=201;
SELECT * FROM rail_db.del_train;

DELETE FROM train WHERE train_id=202;
SELECT * FROM rail_db.del_train;
```

```
DELETE FROM train WHERE train_id=204;
DELETE FROM train WHERE train_id=205;
SELECT * FROM rail_db.del_train;
```

```
173
174  --Inserting some extra data of train in tain
175  INSERT INTO train(train_id,tc_id) VALUES(201,10),(202,12),(203,30),(204,14),(205,15)
176
177  --Deleting data from train
178  DELETE FROM train WHERE train_id=201;
179  DELETE FROM train WHERE train_id=202;
180  DELETE FROM train WHERE train_id=204;
181  DELETE FROM train WHERE train_id=205;
182
```

Data output | Messages | Notifications

```
NOTICE:  Notice : Foreign key already exists in this table.
NOTICE:  Notice : Foreign key already exists in this table.
NOTICE:  Notice : Foreign key already exists in this table.
NOTICE:  Notice : Foreign key already exists in this table.
NOTICE:  Notice : Foreign key already exists in this table.
INSERT 0 0

Query returned successfully in 60 msec.
```

Delete train with train_id 201 :-

```
178  --Deleting data from train
179  DELETE FROM train WHERE train_id=201;
180  SELECT * FROM rail_db.del_train;
181
```

Data output | Messages | Notifications

```
DELETE 0

Query returned successfully in 152 msec.
```

Query | Query History

```
1  SELECT * FROM railway_db.del_train
2
```

Data output | Messages | Notifications

	tr_id integer
1	201

Delete train with train_id 202 :-

```
182  DELETE FROM train WHERE train_id=202;
183  SELECT * FROM rail_db.del_train;
184
185  DELETE FROM train WHERE train_id=204;
186  DELETE FROM train WHERE train_id=205;
187  SELECT * FROM rail_db.del_train;
188
```

Data output | Messages | Notifications

```
DELETE 0

Query returned successfully in 88 msec.
```

Query | Query History

```
1  SELECT * FROM railway_db.del_train
2
```

Data output | Messages | Notifications

	tr_id integer
1	201
2	202

Delete train with train_id 204 and 205 :-

```
184
185 DELETE FROM train WHERE train_id=204;
186 DELETE FROM train WHERE train_id=205;
187 SELECT * FROM rail_db.del_train;
188
```

Data output | Messages | Notifications

```
DELETE 0

Query returned successfully in 161 msec.
```

Data output | Messages | Notifications

	tr_id integer
1	201
2	202
3	204
4	205