Stored Procedures

&

console application code

Console application code:

#include <stdio.h>

#include <stdlib.h>

EXEC SQL BEGIN DECLARE SECTION;

// define as many double pointers as you want in your query projections! It will work only then!

const char \*stmt = "INSERT INTO team VALUES(?,?,?);"; // insert, update delete queries are working only for a particular table!

const char \*stmp = "DELETE FROM team where ownerid=?;";

const char \*stms = "UPDATE team SET location=? where ownerid=?;";

char name[30],query[100],dummy[100],buf[20],\*\*x1,\*\*x2,\*\*x3,\*\*x4,\*\*x5,\*\*x6,\*\*x7,\*\*x8, c;

int inp;

int no;

char y[100],x[100];

EXEC SQL END DECLARE SECTION;

void executeQuery(char \*query){

printf("%s\n",query);

EXEC SQL PREPARE query\_prep FROM :query;

EXEC SQL DECLARE emp\_cur CURSOR FOR query\_prep;

EXEC SQL OPEN emp\_cur;

//printf("%ld\n",sqlca.sqlcode);

//printf("%s\n",sqlca.sqlerrm.sqlerrmc);

if(no==1){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1 ; // add as many pointers as the projections

printf("%s\n",\*x1); // add as many pointers as the projections

}

}

else if(no==2){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2 ; // add as many pointers as the projections

printf("%s\t %s\n",\*x1,\*x2);

}

}

else if(no==3){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3 ; // add as many pointers as the projections

printf("%s\t %s\t %s\n",\*x1,\*x2,\*x3);

}

}

else if(no==4){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3,:x4 ; // add as many pointers as the projections

printf("%s\t %s\t %s\t %s\n",\*x1,\*x2,\*x3,\*x4);

}

}

else if(no==5){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3,:x4,:x5 ; // add as many pointers as the projections

printf("%s\t %s\t %s\t %s\t %s\n",\*x1,\*x2,\*x3,\*x4,\*x5);

}

}

else if(no==6){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3,:x4,:x5,:x6 ; // add as many pointers as the projections

printf("%s\t %s\t %s\t %s\t %s\t %s\n",\*x1,\*x2,\*x3,\*x4,\*x5,\*x6);

}

}

else if(no==7){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3,:x4,:x5,:x6,:x7 ; // add as many pointers as the projections

printf("%s\t %s\t %s\t %s\t %s\t %s\t %s\n",\*x1,\*x2,\*x3,\*x4,\*x5,\*x6,\*x7);

}

}

else if(no==8){

while ( sqlca.sqlcode == 0) {

EXEC SQL FETCH FROM emp\_cur INTO :x1,:x2,:x3,:x4,:x5,:x6,:x7,:x8 ; // add as many pointers as the projections

printf("%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\n",\*x1,\*x2,\*x3,\*x4,\*x5,\*x6,\*x7,\*x8);

}

}

//printf("%s\n",sqlca.sqlerrm.sqlerrmc);

//printf("%ld\n",sqlca.sqlcode);

EXEC SQL CLOSE emp\_cur;

}

void executeUpdate(char \* query){

printf("%s",query);

EXEC SQL PREPARE mystmt FROM :query;

EXEC SQL EXECUTE mystmt;

EXEC SQL COMMIT;

}

int main() {

EXEC SQL CONNECT TO "201401456@10.100.71.21" USER "201401456" USING "201401456";

EXEC SQL set search\_path to project;

printf("Enter 1 for insert update or delete \n insert 2 for running query with output \n Enter any other char for exit \n");

int option;

scanf("%d",&option);

c=getchar();

if(option==1){

printf("give your query\n");

scanf("%[^\n]\*c",query);

executeUpdate(query);

}

else if(option==2){

scanf("%d",&no);

char c=getchar();

printf("give your query\n");

scanf("%[^\n]\*c",query);

//printf("%s\n", query);

executeQuery(query);

}

else{

printf("Wrong input");

}

return 0;

}

Stored Procedures :

(i) get league matches for a given team name and his ownerid

CREATE OR REPLACE FUNCTION project.get\_league\_matches\_for\_team(oid integer, tname character varying)

RETURNS SETOF type1 AS

$BODY$declare

rec type1;

curs1 cursor is (select leaguename,leaguecommissionerid,matchid from project.participations natural join project.leaguematches where ownerid=oid and teamname=tname);

begin

open curs1;

loop

fetch curs1 into rec.leaguename,rec.leaguecommissoinerid,rec.matchid;

exit when not found;

return next rec;

end loop;

close curs1;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_league\_matches\_for\_team(oid integer, tname character varying) OWNER TO "201401456";

(ii) get league score board given leaguecommissionerid and league name

CREATE OR REPLACE FUNCTION project.get\_league\_score\_board(comid integer, lname character varying)

RETURNS SETOF my\_type4 AS

$BODY$declare

oid my\_type4;

c cursor is (select ownerid,Score from

(select \* from

(select leaguename,leaguecommissionerid,ownerid,sum(score)as

Score from project.teamleaguematchperfomance

group by leaguename,leaguecommissionerid,ownerid)

as R where R.leaguename=lname and R.leaguecommissionerid=comid)

as R2 order by score);

begin

open c;

loop

fetch c into oid.f1,oid.f2;

exit when not found;

return next oid;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_league\_score\_board(comid integer, lname character varying) OWNER TO "201401456";

(iii) get league winner team given leaguecommissionerid and his commissionerid

CREATE OR REPLACE FUNCTION project.get\_league\_winner\_team(comid integer, lname character varying)

RETURNS SETOF project.team AS

$BODY$declare

rec project.team%rowtype;

begin

for rec in (select project.team.\* from (select teamname,ownerid from

(select R2.\* from (select teamname,leaguename,leaguecommissionerid,ownerid,sum(score) as LeagueScore from project.teamleaguematchperfomance group by teamname,leaguename,leaguecommissionerid,ownerid) as R2

join

(select leaguename,leaguecommissionerid,max(leaguescore)as MaxLeagueScore from(

select teamname,leaguename,leaguecommissionerid,ownerid,sum(score) as LeagueScore from project.teamleaguematchperfomance group by teamname,leaguename,leaguecommissionerid,ownerid) as R group by leaguename,leaguecommissionerid) as R3 on

(R3.MaxLeagueScore=R2.LeagueScore and R2.leaguename=R3.leaguename and R2.leaguecommissionerid=R3.leaguecommissionerid)) as R5 where R5.leaguename=lname and R5.leaguecommissionerid=comid) as Rf natural join project.team)

loop

return next rec;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_league\_winner\_team(comid integer, lname character varying) OWNER TO "201401456";

(iv) get number of league wins from

CREATE OR REPLACE FUNCTION project.get\_no\_of\_league\_wins(oid integer)

RETURNS bigint AS

$BODY$declare

result bigint;

begin

select no\_of\_leagues\_won into result from

(select ownerid,no\_of\_leagues\_won from (

select ownerid,count((leaguename,leaguecommissionerid)) as no\_of\_leagues\_won from (

select R2.\* from (select teamname,leaguename,leaguecommissionerid,ownerid,sum(score) as LeagueScore from teamleaguematchperfomance group by teamname,leaguename,leaguecommissionerid,ownerid) as R2

join

(select leaguename,leaguecommissionerid,max(leaguescore)as MaxLeagueScore from(

select teamname,leaguename,leaguecommissionerid,ownerid,sum(score) as LeagueScore from teamleaguematchperfomance group by teamname,leaguename,leaguecommissionerid,ownerid) as R group by leaguename,leaguecommissionerid) as R3

on (R3.MaxLeagueScore=R2.LeagueScore and R2.leaguename=R3.leaguename and R2.leaguecommissionerid=R3.leaguecommissionerid)) as R4 group by ownerid)as R6) as Rf where Rf.ownerid=oid;

return result;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_no\_of\_league\_wins(oid integer) OWNER TO "201401456";

(v) get owner teams from ownerid

CREATE OR REPLACE FUNCTION project.get\_owner\_teams(oid integer)

RETURNS SETOF character varying AS

$BODY$declare

rec project.team.teamname%type;

begin

for rec in (select teamname from project.team where ownerid=oid)

loop

return next rec;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_owner\_teams(oid integer) OWNER TO "201401456";

(vi) get players for a uefa match

CREATE OR REPLACE FUNCTION project.get\_players\_for\_match(matchid integer)

RETURNS SETOF project.players AS

$BODY$declare

rec project.players%rowtype;

begin

for rec in ( select project.players.\* from (project.playermatchperformance natural join project.players )where project.playermatchperformance.matchid=matchid)

loop

return next rec;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_players\_for\_match(matchid integer) OWNER TO "201401456";

(vii) get team players for given team name, matchid and ownerid

CREATE OR REPLACE FUNCTION project.get\_team\_players\_for\_team(oid integer, name character varying, mid integer)

RETURNS SETOF project.players AS

$BODY$declare

rec project.players%rowtype;

begin

for rec in (select project.players.\* from project.teamplayers natural join project.players where ownerid=oid and teamname=name and matchid=mid)

loop

return next rec;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_team\_players\_for\_team(oid integer, name character varying, mid integer) OWNER TO "201401456";

(viii)get top scorer player for a match

CREATE OR REPLACE FUNCTION project.get\_top\_scorer\_for\_match(mid integer)

RETURNS SETOF project.players AS

$BODY$declare

rec project.players%rowtype;

begin

for rec

in (select \* from

(select playerid from

(select playerid,matchid from project.playermatchscore where

(matchid,score)

in

(select matchid,max(score)as MS from project.playermatchscore group by matchid)) as R where R.matchid=mid) as R2 natural join project.players)

loop

return next rec;

end loop;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_top\_scorer\_for\_match(mid integer) OWNER TO "201401456";

(ix) get uefa score board

CREATE OR REPLACE FUNCTION project.get\_uefa\_score\_board()

RETURNS SETOF my\_type AS

$BODY$declare

rec my\_type;

curs2 CURSOR is

(select teamname,wins from project.uefa\_teams

left join

(select uefateamname,count(matchid)as wins from

(select R1.uefateamname,R1.matchid from

(select uefateamname,matchid,sum(goals)as no\_of\_goals from project.playermatchperformance natural join project.players group by uefateamname,matchid)

as R1

join

(select uefateamname,matchid,sum(goals)as no\_of\_goals from project.playermatchperformance natural join project.players group by uefateamname,matchid) as R2

on(R1.matchid=R2.matchid and R1.no\_of\_goals>R2.no\_of\_goals)) as R group by uefateamname)as R on teamname=R.uefateamname);

begin

open curs2;

LOOP

FETCH curs2 INTO rec.f1,rec.f2;

EXIT WHEN NOT FOUND;

return next rec;

END LOOP;

close curs2;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.get\_uefa\_score\_board() OWNER TO "201401456";

(x) goals by uefateam

CREATE OR REPLACE FUNCTION project.goals\_by\_uefateam(tname character varying, mid integer)

RETURNS bigint AS

$BODY$declare

result bigint :=-1;

rec record;

begin

for rec in

(select uefateamname,matchid,sum(goals)as no\_of\_goals from playermatchperformance natural join players group by uefateamname,matchid)

loop

if (rec.uefateamname=tname and rec.matchid=mid )then

result=rec.no\_of\_goals;

end if;

end loop;

if result <> -1 then

return result;

else raise exception 'team % does not participate in this match %',tname,mid;

end if;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.goals\_by\_uefateam(tname character varying, mid integer) OWNER TO "201401456";

**Triggers :**

(i) check no of players in the team is less than or equal to 10 before insert on teamplayers

CREATE OR REPLACE FUNCTION project.check\_no\_of\_players()

RETURNS trigger AS

$BODY$declare

num bigint;

begin

select No\_of\_players into num from

(select matchid,teamname,ownerid,count(playerid)as No\_of\_players from project.teamplayers group by matchid,teamname,ownerid) as R where R.matchid=NEW.matchid and R.teamname=NEW.teamname and R.ownerid=NEW.ownerid;

if num >10 then

raise exception 'The team already has 11 players';

end if;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.check\_no\_of\_players() OWNER TO "201401456";

CREATE TRIGGER check\_no\_of\_players

BEFORE INSERT

ON project.teamplayers

FOR EACH ROW

EXECUTE PROCEDURE project.check\_no\_of\_players();

(ii) insert player match score in the table player match score after insert into player match performance table

CREATE OR REPLACE FUNCTION project.set\_player\_match\_score()

RETURNS trigger AS

$BODY$declare

scoretemp numeric;

begin

IF (TG\_OP = 'DELETE') THEN

DELETE FROM project.playermatchscore WHERE playerid = OLD.playerid and matchid=OLD.matchid;

IF NOT FOUND THEN RETURN NULL; END IF;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

scoretemp=((NEW.saves\*2) -NEW.overtimelosses -NEW.goalsallowed + (2.5\*NEW.goalssaved) - NEW.blockedshots + (1.5\*NEW.goals) + NEW.shots + (1.5\*NEW.penaltykickattemptsdefended) - NEW.penaltykickgoalsdefended + (0.75\*NEW.steals) + (3\*NEW.gamewinninggoals) );

UPDATE project.playermatchscore SET score = scoretemp WHERE playerid = OLD.playerid and matchid=OLD.matchid;

IF NOT FOUND THEN RETURN NULL; END IF;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

scoretemp=((NEW.saves\*2) -NEW.overtimelosses -NEW.goalsallowed + (2.5\*NEW.goalssaved) - NEW.blockedshots + (1.5\*NEW.goals) + NEW.shots + (1.5\*NEW.penaltykickattemptsdefended) - NEW.penaltykickgoalsdefended + (0.75\*NEW.steals) + (3\*NEW.gamewinninggoals) );

INSERT INTO playermatchscore VALUES(NEW.playerid, NEW.matchid,scoretemp);

RETURN NEW;

END IF;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.set\_player\_match\_score() OWNER TO "201401456";

CREATE TRIGGER insert\_into\_playermatchscore

AFTER INSERT OR UPDATE OR DELETE

ON project.playermatchperformance

FOR EACH ROW

EXECUTE PROCEDURE project.set\_player\_match\_score();

(iii) set team league match score after insert into player match score

CREATE OR REPLACE FUNCTION project.set\_team\_league\_match\_score()

RETURNS trigger AS

$BODY$begin

delete from project.teamleaguematchperfomance;

insert into project.teamleaguematchperfomance (matchid,teamname,ownerid,leaguename,leaguecommissionerid,score)

select matchid,teamname,ownerid,leaguename,leaguecommissionerid,sum(score)as perfomance from (select \* from (select \* from (select \* from project.participations natural join project.leaguematches) as R natural join project.teamplayers)as R2 natural join project.playermatchscore)as R3 group by R3.matchid,R3.teamname,R3.ownerid,R3.leaguename,R3.leaguecommissionerid;

return new;

end;$BODY$

LANGUAGE 'plpgsql' VOLATILE;

ALTER FUNCTION project.set\_team\_league\_match\_score() OWNER TO "201401456";

CREATE TRIGGER set\_team\_league\_match\_score

AFTER INSERT OR UPDATE OR DELETE

ON project.playermatchperformance

FOR EACH ROW

EXECUTE PROCEDURE project.set\_team\_league\_match\_score();