TRIGGERS

GROUP 5D

1.

```
set SEARCH_PATH to researchportal;

CREATE TRIGGER rsch_id_check
BEFORE INSERT OR UPDATE ON research_contribution
FOR EACH STATEMENT EXECUTE PROCEDURE author_id_check();

CREATE OR REPLACE FUNCTION author_id_check() RETURNS trigger AS $rsch_id_check$
   DECLARE
                studid character(15);
                profid character(15);
                auth_id integer;
   BEGIN
      --
      -- Create a row in emp_audit to reflect the operation performed on emp,
      -- make use of the special variable TG_OP to work out the operation.
      --
      IF (TG_OP = 'INSERT') THEN

                        FOR profid IN SELECT prof_id FROM researchportal.professor
                        LOOP
                                IF NEW.author_id = profid THEN
                                        RETURN NEW;
                                END IF;
                        END LOOP;

                        --auth_id := to_number(NEW.author_id, '999999');

                        FOR studid IN SELECT sid FROM researchportal.student
                        LOOP
                                IF NEW.author_id = studid THEN
                                        RETURN NEW;
                                END IF;
                        END LOOP;

      END IF;

      raise EXCEPTION 'author_id does not exist';
      RETURN NULL;
   END;
$rsch_id_check$ LANGUAGE plpgsql;
```

2.

```
set SEARCH_PATH to researchportal;
CREATE TRIGGER city_scope_check
BEFORE INSERT OR UPDATE ON conference
FOR EACH ROW EXECUTE PROCEDURE conference_check();

CREATE OR REPLACE FUNCTION conference_check() RETURNS trigger AS
$city_scope_check$
   DECLARE
                scope character varying(15);
   BEGIN
      --
      -- Create a row in emp_audit to reflect the operation performed on emp,
      -- make use of the special variable TG_OP to work out the operation.
      --
      IF (TG_OP = 'INSERT') THEN
                                NEW.city = INITCAP(NEW.city);
                                NEW.scope = INITCAP(NEW.scope);
                                IF NEW.scope = 'International' OR NEW.scope = 'National' THEN
                                        RETURN NEW;
                                END IF;
      END IF;

      raise EXCEPTION 'conference scope must be national or international';
      RETURN NULL;
   END;
$city_scope_check$ LANGUAGE plpgsql;
```

3.

```
set search_path to researchportal;
CREATE TRIGGER prof_delete_check
AFTER DELETE OR UPDATE ON professor
FOR EACH ROW
EXECUTE PROCEDURE prof_delete_check();

CREATE OR REPLACE FUNCTION researchportal.prof_delete_check()
  RETURNS trigger AS
$BODY$
   BEGIN
      IF (TG_OP = 'DELETE')THEN
                DELETE FROM research_contribution WHERE author_id = OLD.prof_id ;

         ELSIF (TG_OP = 'UPDATE') THEN
                UPDATE research_contribution SET author_id = NEW.prof_id WHERE author_id =
OLD.prof_id;
      END IF;
      RETURN NEW;
   END;
$BODY$
LANGUAGE plpgsql;
```

4.

```
CREATE TRIGGER check_project
BEFORE INSERT OR UPDATE ON project
FOR EACH ROW
EXECUTE PROCEDURE check_project_status();


CREATE OR REPLACE FUNCTION check_project_status() RETURNS trigger AS
$BODY$
DECLARE

BEGIN
	IF NEW.status = 'Completed' OR NEW.status = 'Upcoming' OR NEW.status = 'Stalled' OR
NEW.status = 'Ongoing'
	THEN
		RETURN NEW;
	ELSE
		RAISE EXCEPTION 'invalid status. Must be one of
Completed,Upcoming,Stalled,Ongoing';
	END IF;
END;
$BODY$
LANGUAGE plpgsql;
```

5.

```
CREATE OR REPLACE FUNCTION researchportal.student_delete_check()
  RETURNS trigger AS
$BODY$
   BEGIN
      IF (TG_OP = 'DELETE')THEN
		DELETE FROM research_contribution WHERE author_id = OLD.sid ;

	ELSIF (TG_OP = 'UPDATE') THEN
		UPDATE research_contribution SET author_id = NEW.sid WHERE author_id =
OLD.sid;
      END IF;
      RETURN NEW;
   END;
$BODY$
LANGUAGE plpgsql;

set search_path to researchportal;
CREATE TRIGGER student_delete_check
AFTER DELETE OR UPDATE ON student
FOR EACH ROW
EXECUTE PROCEDURE student_delete_check();
```