COMS4111-project2

TEAM MEMBER: Chenhui Mao(uni: cm4054), Zihao Huang(uni: zh2481)

Abstract

In this project we will modify the schema we have in project 1 to accommodate the following features

- **Composite type**
- **Arrays**
- **Documents**
- **Trigger**

Database

Account

zh2481

Command to enter the database

psql -U zh2481 -h 35.196.73.133 -d proj1part2

New Schema

```
create type university AS ENUM ('Columbia University',
  'New York University',
  'Rochester University',
  'University of California, San Diego',
  'Massachusetts Institute of Technology',
  'Stanford University',
  'Harvard University',
  'University of California, Los Angeles',
  'University of Southern California',
  'University of California, Irvine');
```

```sql
create type location AS ENUM ('New York',
  'San Diego',
  'Massachusetts',
  'California',
  'Los Angeles'
);

CREATE TYPE new_user AS (
  name text[2],
  webpage_link varchar(150)
);

CREATE TYPE medals AS (
  gold_medal INTEGER,
  silver_medal INTEGER,
  bronze_medal INTEGER
);

CREATE TABLE DATASETS(
  name text NOT NULL,
  idx VARCHAR(20),
  provenance text NOT NULL,
  PRIMARY KEY (idx)
);

CREATE TABLE TASKS(
  t_id VARCHAR(20),
  name TEXT NOT NULL,
  description TEXT,
  d_id VARCHAR(20) NOT NULL,
  PRIMARY KEY (t_id),
  FOREIGN KEY (d_id) REFERENCES DATASETS(idx)
    ON DELETE CASCADE
);

CREATE TABLE UNIVERSITIES(
  name university,
  location location NOT NULL,
  PRIMARY KEY (name)
);

CREATE TABLE USERS(
  u_id VARCHAR(20),
  user_info new_user NOT NULL,
  medals medals,
  u_name university,
  PRIMARY KEY (u_id),
  FOREIGN KEY (u_name) REFERENCES UNIVERSITIES(name)
    ON DELETE SET NULL,
  CONSTRAINT minimum_medal
  CHECK ((medals).gold_medal>=0 AND (medals).silver_medal>=0 AND
(medals).bronze_medal>=0)
);

CREATE TABLE CODES(
  idx VARCHAR(20),
  code TEXT NOT NULL,
  output_file TEXT,
  submit_date DATE NOT NULL,
  stars INTEGER DEFAULT(0),
  t_id VARCHAR(20) NOT NULL,
  u_id VARCHAR(20) NOT NULL,
  PRIMARY KEY (idx),
  FOREIGN KEY (t_id) REFERENCES TASKS(t_id)
    ON DELETE CASCADE,
  FOREIGN KEY (u_id) REFERENCES USERS(u_id)
    ON DELETE CASCADE,
  CONSTRAINT minimum_stars
  CHECK (stars>=0)
);

CREATE TABLE MAINTAIN(
  u_id VARCHAR(20),
  d_id VARCHAR(20),
  PRIMARY KEY (u_id, d_id),
  FOREIGN KEY (u_id) REFERENCES USERS(u_id)
    ON DELETE NO ACTION,
  FOREIGN KEY (d_id) REFERENCES DATASETS(idx)
    ON DELETE CASCADE
);

--Follow yourself---
CREATE TABLE FOLLOW(
  follow_date DATE NOT NULL,
  followed_id VARCHAR(20),
  follower_id VARCHAR(20),
  PRIMARY KEY (followed_id, follower_id),
  FOREIGN KEY (followed_id) REFERENCES USERS(u_id)
    ON DELETE CASCADE,
  FOREIGN KEY (follower_id) REFERENCES USERS(u_id)
    ON DELETE CASCADE
);

CREATE TABLE COMPETITIONS(
  name TEXT,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  prize VARCHAR(10) NOT NULL,
  PRIMARY KEY (name)
);

CREATE TABLE USE(
  d_id VARCHAR(20),
  c_name TEXT,
  PRIMARY KEY (d_id, c_name),
  FOREIGN KEY (d_id) REFERENCES DATASETS(idx),
  FOREIGN KEY (c_name) REFERENCES COMPETITIONS(name)
    ON DELETE CASCADE
);

CREATE TABLE TAKE(
  score REAL,
  paticipant_date DATE,
  u_id VARCHAR(20),
  c_name TEXT,
  PRIMARY KEY (u_id, c_name),
  FOREIGN KEY (u_id) REFERENCES USERS(u_id)
    ON DELETE CASCADE,
  FOREIGN KEY (c_name) REFERENCES COMPETITIONS(name)
    ON DELETE CASCADE,
  CONSTRAINT minimum_score
  CHECK (score>=0)
);

CREATE TABLE COURSES(
  idx VARCHAR(20),
  name TEXT NOT NULL,
  description TEXT,
  PRIMARY KEY (idx)
);

CREATE TABLE EXPLORE(
  d_id VARCHAR(20),
  c_id VARCHAR(20),
  PRIMARY KEY (d_id,c_id),
  FOREIGN KEY (d_id) REFERENCES DATASETS(idx)
    ON DELETE CASCADE,
  FOREIGN KEY (c_id) REFERENCES COURSES(idx)
    ON DELETE CASCADE
);

CREATE TABLE TEACH(
  c_id VARCHAR(20),
  u_name university,
  PRIMARY KEY (c_id, u_name),
  FOREIGN KEY (c_id) REFERENCES COURSES(idx)
    ON DELETE CASCADE,
  FOREIGN KEY (u_name) REFERENCES UNIVERSITIES(name)
    ON DELETE CASCADE
);

CREATE TABLE DATASETS_AUDIT(
    operation         char(1)   NOT NULL,
    stamp             timestamp NOT NULL,
    d_id              text      NOT NULL,
);

CREATE OR REPLACE FUNCTION process_datasets_audit() RETURNS TRIGGER AS
$datasets_audit$
    BEGIN
        IF (TG_OP = 'DELETE') THEN
            INSERT INTO datasets_audit SELECT 'D', now(), OLD.idx;
        ELSIF (TG_OP = 'UPDATE') THEN
            INSERT INTO datasets_audit SELECT 'U', now(), NEW.idx;
        ELSIF (TG_OP = 'INSERT') THEN
            INSERT INTO datasets_audit SELECT 'I', now(), NEW.idx;
        END IF;
        RETURN NULL;
    END;
$datasets_audit$ LANGUAGE plpgsql;

CREATE TRIGGER datasets_audit
AFTER INSERT OR UPDATE OR DELETE ON datasets
    FOR EACH ROW EXECUTE FUNCTION process_datasets_audit();
```

As can be seen here we have made several changes to the original schema

Enumerate Type

Firstly, we add enumerate types to ensure that all universities in the database must be within certain range. It definitely can't be a random one. So as the location of all university. Another advantage of applying enumerate Type in database is to prevent typo error when inserting a new record into database.

```
CREATE TYPE university AS ENUM ('Columbia University',
  'New York University',
  'Rochester University',
  'University of California, San Diego',
  'Massachusetts Institute of Technology',
  'Stanford University',
  'Harvard University',
  'University of California, Los Angeles',
  'University of Southern California',
  'University of California, Irvine');

CREATE TYPE location AS ENUM ('New York',
  'San Diego',
  'Massachusetts',
  'California',
  'Los Angeles'
);

CREATE TABLE UNIVERSITIES(
  name university,
  location location NOT NULL,
  PRIMARY KEY (name)
);
```

Composite types & Array

We also add a composite type to users information and medals. Moreover, the first name and last name will be transformed into an array type, so we don't need two separated variable anymore. This kind of structure will make the declaration of the schema more succinct.

```
CREATE TYPE new_user AS (
  name text[2],
  webpage_link varchar(150)
);

CREATE TYPE medals AS (
  gold_medal INTEGER,
  silver_medal INTEGER,
  bronze_medal INTEGER
);

CREATE TABLE USERS(
  u_id VARCHAR(20),
  user_info new_user NOT NULL,
  medals medals,
  u_name university,
  PRIMARY KEY (u_id),
  FOREIGN KEY (u_name) REFERENCES UNIVERSITIES(name)
    ON DELETE SET NULL,
  CONSTRAINT minimum_medal
  CHECK ((medals).gold_medal>=0 AND (medals).silver_medal>=0 AND
(medals).bronze_medal>=0)
);
```

Text

We change some long text variable into text. With CHAR or VARCHAR, we need to define the maximum length for the variable, which is not always estimable beforehand That is not a problem to text. Moreover, with text we can make some complex text operations like parsing and matching easier. For example, we can find all courses that relate to *Machine Learn* (Will cover within 3 meaningful queries).

```
CREATE TABLE DATASETS(
  name TEXT NOT NULL,
  idx VARCHAR(20),
  provenance TEXT NOT NULL,
  PRIMARY KEY (idx)
);

CREATE TABLE COMPETITIONS(
  name TEXT,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  prize VARCHAR(10) NOT NULL,
  PRIMARY KEY (name)
);

CREATE TABLE COURSES(
  idx VARCHAR(20),
  name TEXT NOT NULL,
  description TEXT,
  PRIMARY KEY (idx)
);
```

Trigger

We add an auditing table in the database with Trigger. Originally we need a table called *Maintain* to record all update done to the datasets. The procedure to make it work is quite complex, Some judgements must be done by the backend codes. However, with trigger we can make the operation easier. All things can be done by executing the aduit function each time when changes have been made to the datasets.

```
CREATE TABLE datasets_audit(
    operation         char(1)   NOT NULL,
    stamp             timestamp NOT NULL,
    d_id              text      NOT NULL,
);

CREATE OR REPLACE FUNCTION process_datasets_audit() RETURNS TRIGGER AS
$datasets_audit$
    BEGIN
        IF (TG_OP = 'DELETE') THEN
            INSERT INTO datasets_audit SELECT 'D', now(), OLD.idx;
        ELSIF (TG_OP = 'UPDATE') THEN
            INSERT INTO datasets_audit SELECT 'U', now(), NEW.idx;
        ELSIF (TG_OP = 'INSERT') THEN
            INSERT INTO datasets_audit SELECT 'I', now(), NEW.idx;
        END IF;
        RETURN NULL;
    END;
$datasets_audit$ LANGUAGE plpgsql;

CREATE TRIGGER datasets_audit
AFTER INSERT OR UPDATE OR DELETE ON datasets
```

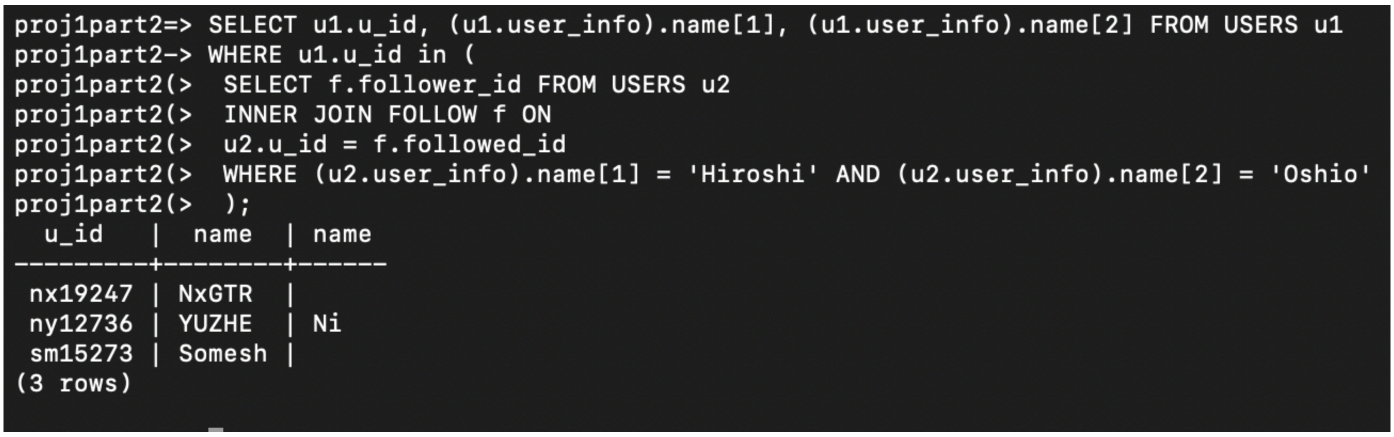
```
    FOR EACH ROW EXECUTE FUNCTION process_datasets_audit();
```

Three meaningful queries

Query 1

We want to find out who follows users whose first name is Hiroshi and las name is Oshio. This is one of the interesting query we made in project 1 part 2. And we made changes to this query to accommodate all changes have beed made to the USERS table. This query shows how we handle the composite type.

```
SELECT u1.u_id, (u1.user_info).name[1], (u1.user_info).name[2] FROM USERS u1
WHERE u1.u_id in (
  SELECT f.follower_id FROM USERS u2
  INNER JOIN FOLLOW f ON
  u2.u_id = f.followed_id
  WHERE (u2.user_info).name[1] = 'Hiroshi' AND (u2.user_info).name[2] = 'Oshio'
  );
```



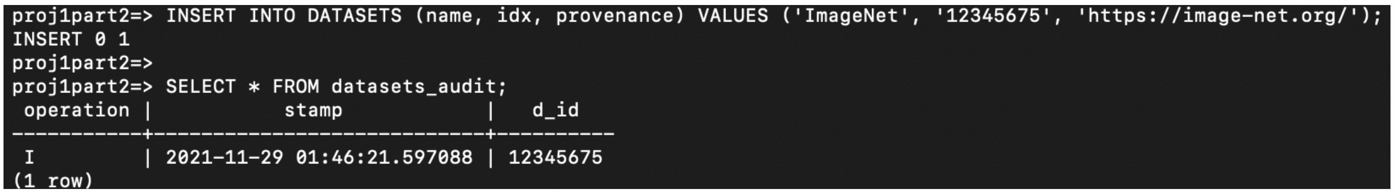
```
proj1part2=> SELECT u1.u_id, (u1.user_info).name[1], (u1.user_info).name[2] FROM USERS u1
proj1part2-> WHERE u1.u_id in (
proj1part2(>  SELECT f.follower_id FROM USERS u2
proj1part2(>  INNER JOIN FOLLOW f ON
proj1part2(>  u2.u_id = f.followed_id
proj1part2(>  WHERE (u2.user_info).name[1] = 'Hiroshi' AND (u2.user_info).name[2] = 'Oshio'
proj1part2(>  );
  u_id   |  name   | name
---------+---------+------
 nx19247 | NxGTR   |
 ny12736 | YUZHE   | Ni
 sm15273 | Somesh |
(3 rows)
```

Query 2

This query shows how the trigger functions work. Suppose we insert a new record into the DATASETS. The Trigger function will be executed automatically after the inseration have done. The detail is that we will add (The operation, timestamp, dataset id) into the datasets_audit Table.

```
INSERT INTO DATASETS (name, idx, provenance) VALUES ('ImageNet', '12345675',
'https://image-net.org/');

SELECT * FROM datasets_audit;
```

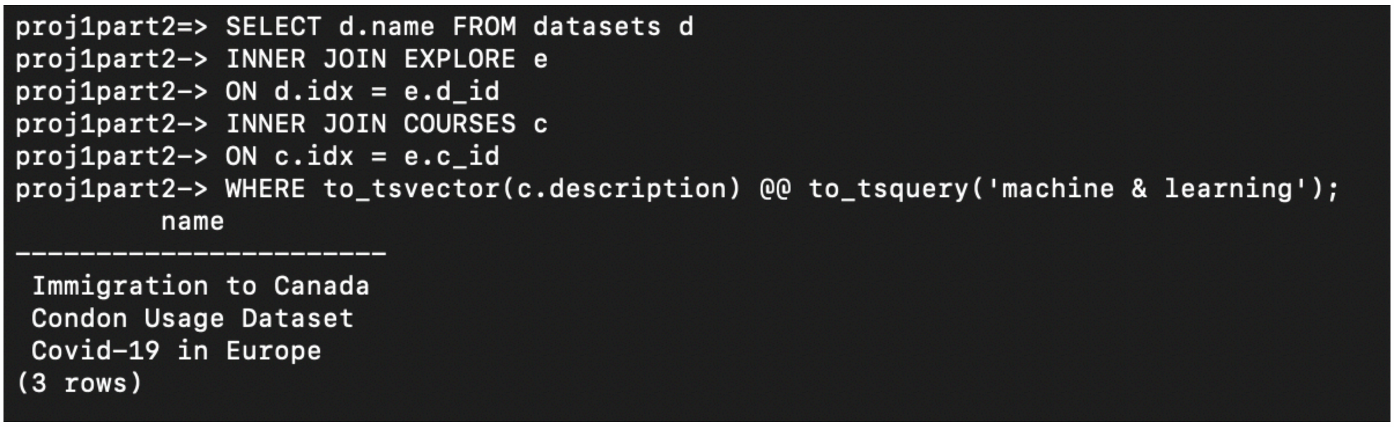


```
proj1part2=> INSERT INTO DATASETS (name, idx, provenance) VALUES ('ImageNet', '12345675', 'https://image-net.org/');
INSERT 0 1
proj1part2=>
proj1part2=> SELECT * FROM datasets_audit;
 operation |           stamp            |   d_id
-----------+----------------------------+----------
 I         | 2021-11-29 01:46:21.597088 | 12345675
(1 row)
```

Query 3

The third query is to find out which datasets have been explored in the course related to machine learning.

```
SELECT d.name FROM datasets d
INNER JOIN EXPLORE e
ON d.idx = e.d_id
INNER JOIN COURSES c
ON c.idx = e.c_id
WHERE to_tsvector(c.description) @@ to_tsquery('machine & learning');
```



```
proj1part2=> SELECT d.name FROM datasets d
proj1part2-> INNER JOIN EXPLORE e
proj1part2-> ON d.idx = e.d_id
proj1part2-> INNER JOIN COURSES c
proj1part2-> ON c.idx = e.c_id
proj1part2-> WHERE to_tsvector(c.description) @@ to_tsquery('machine & learning');
         name
-----------------------
 Immigration to Canada
 Condon Usage Dataset
 Covid-19 in Europe
(3 rows)
```