ICOM 5016 Database Systems

Database Implementation Overview

PIPS Project Team:

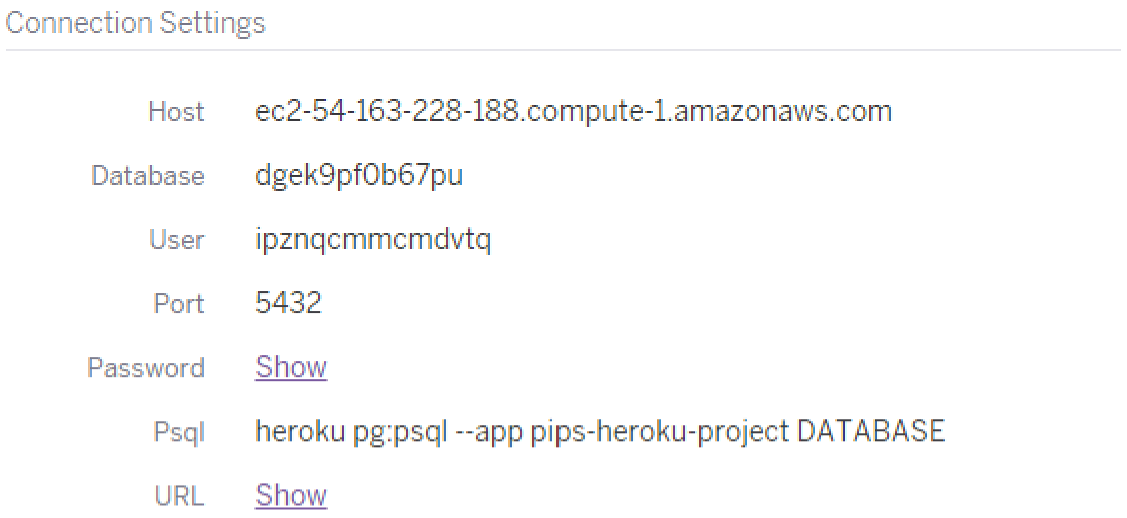
Sebastian Vissepo

Javier E. Colon

Fernando M. Mari

I. Selected DMBS Technology: PostgreSQL

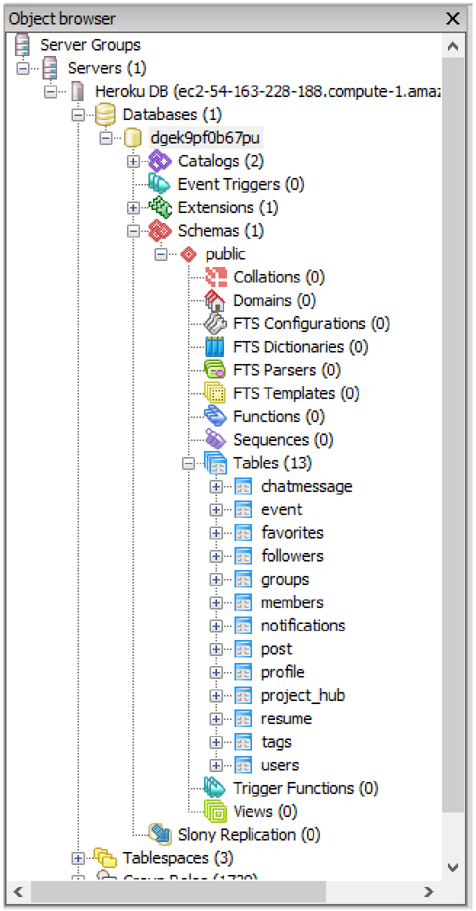
Heroku provides users with the ability to provision a PostgreSQL database. The following database was generated by Heroku:



Connection Settings

Host	ec2-54-163-228-188.compute-1.amazonaws.com
Database	dgek9pf0b67pu
User	ipznqcmmcmdvtq
Port	5432
Password	Show
Psql	heroku pg:psql --app pips-heroku-project DATABASE
URL	Show

Using PosgreSQL Tools PgAdmin, 13 tables were added to the schema:



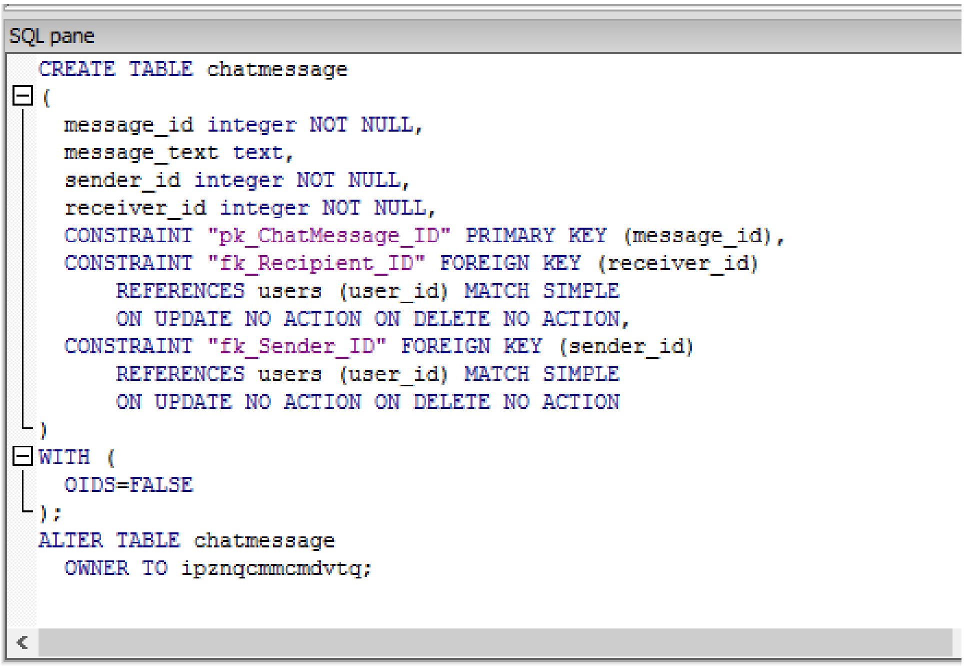
Object browser

```
Server Groups
  Servers (1)
    Heroku DB (ec2-54-163-228-188.compute-1.amaz
      Databases (1)
        dgek9pf0b67pu
          Catalogs (2)
          Event Triggers (0)
          Extensions (1)
          Schemas (1)
            public
              Collations (0)
              Domains (0)
              FTS Configurations (0)
              FTS Dictionaries (0)
              FTS Parsers (0)
              FTS Templates (0)
              Functions (0)
              Sequences (0)
              Tables (13)
                chatmessage
                event
                favorites
                followers
                groups
                members
                notifications
                post
                profile
                project_hub
                resume
                tags
                users
              Trigger Functions (0)
              Views (0)
          Slony Replication (0)
      Tablespaces (3)
      Group Roles (1720)
```

II. Queries generated by PgAdmin

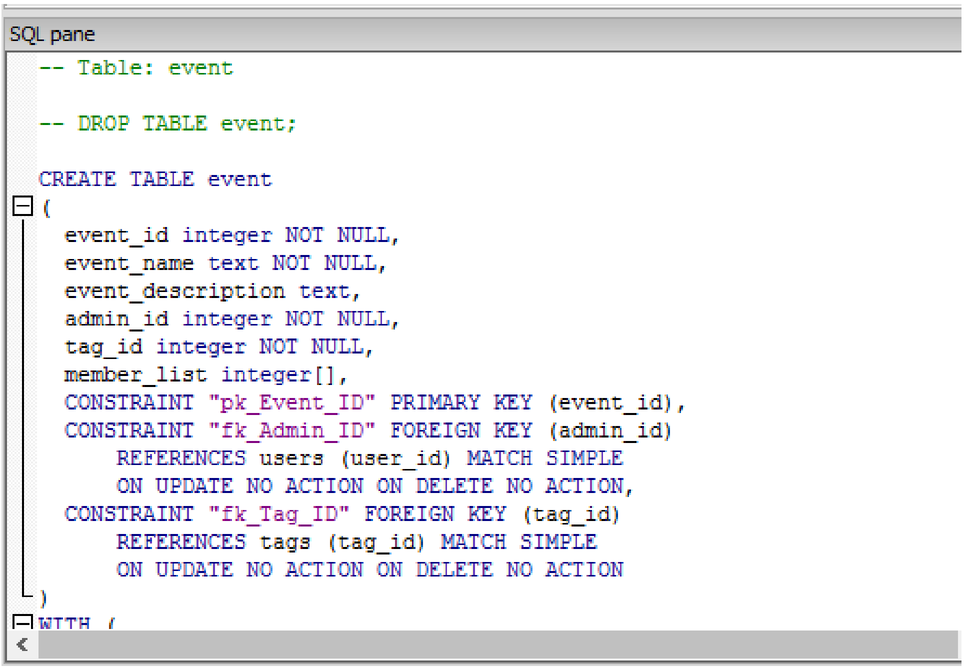
The UI was capable of generating tables, columns and constraints while at the same time generating the queries that result in the operation.

ChatMessage Table:



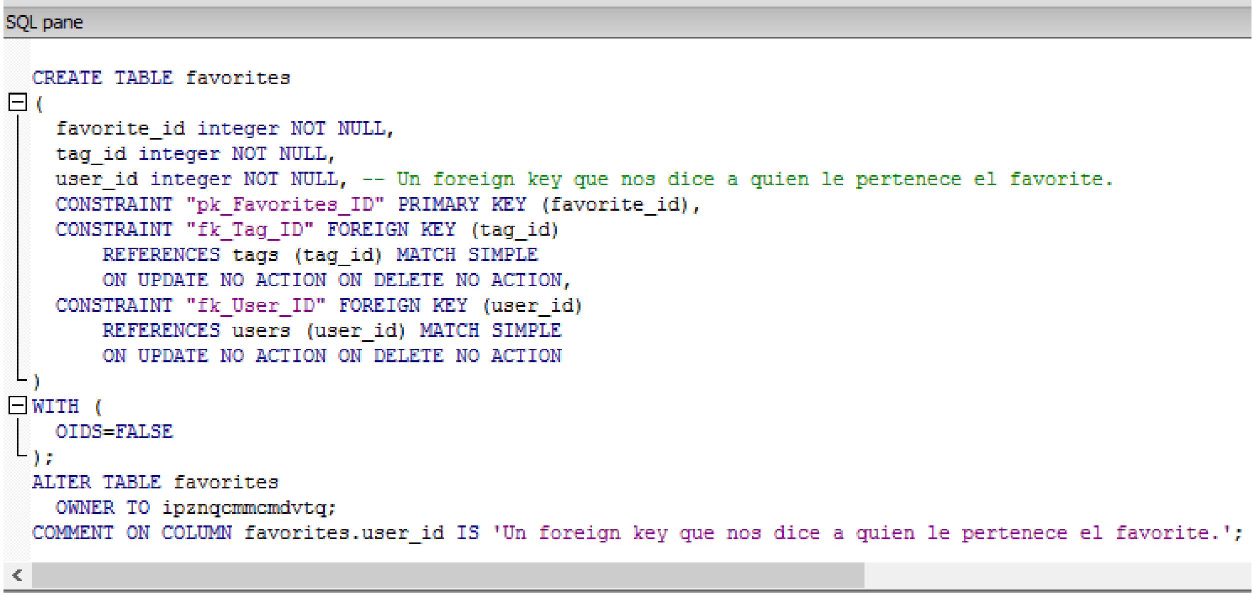
```
CREATE TABLE chatmessage
(
  message_id integer NOT NULL,
  message_text text,
  sender_id integer NOT NULL,
  receiver_id integer NOT NULL,
  CONSTRAINT "pk_ChatMessage_ID" PRIMARY KEY (message_id),
  CONSTRAINT "fk_Recipient_ID" FOREIGN KEY (receiver_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_Sender_ID" FOREIGN KEY (sender_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE chatmessage
  OWNER TO ipznqcmmcmdvtq;
```

Event Table:



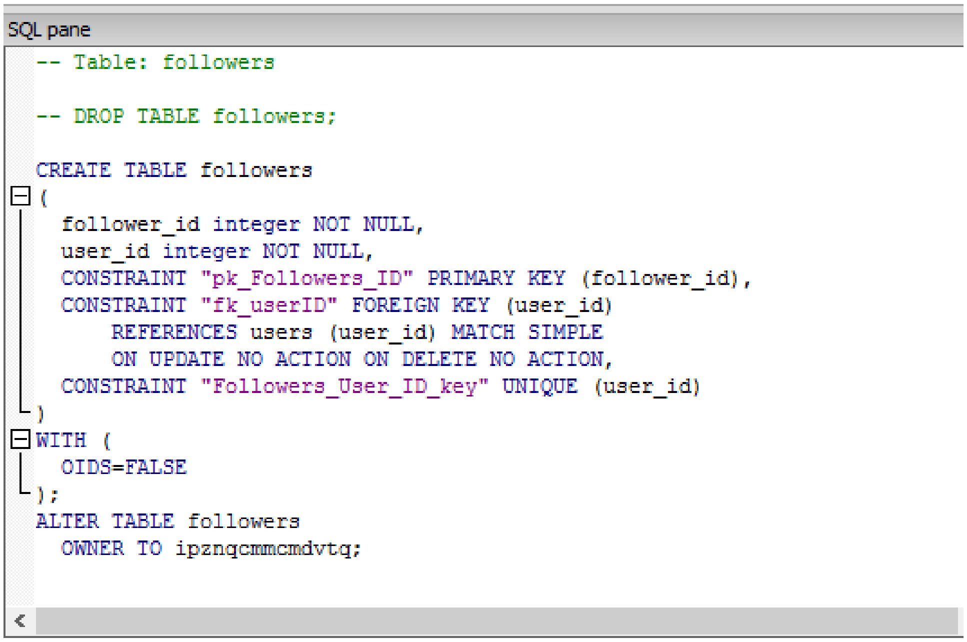
```
-- Table: event

-- DROP TABLE event;

CREATE TABLE event
(
  event_id integer NOT NULL,
  event_name text NOT NULL,
  event_description text,
  admin_id integer NOT NULL,
  tag_id integer NOT NULL,
  member_list integer[],
  CONSTRAINT "pk_Event_ID" PRIMARY KEY (event_id),
  CONSTRAINT "fk_Admin_ID" FOREIGN KEY (admin_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_Tag_ID" FOREIGN KEY (tag_id)
      REFERENCES tags (tag_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
```

Favorites Table:



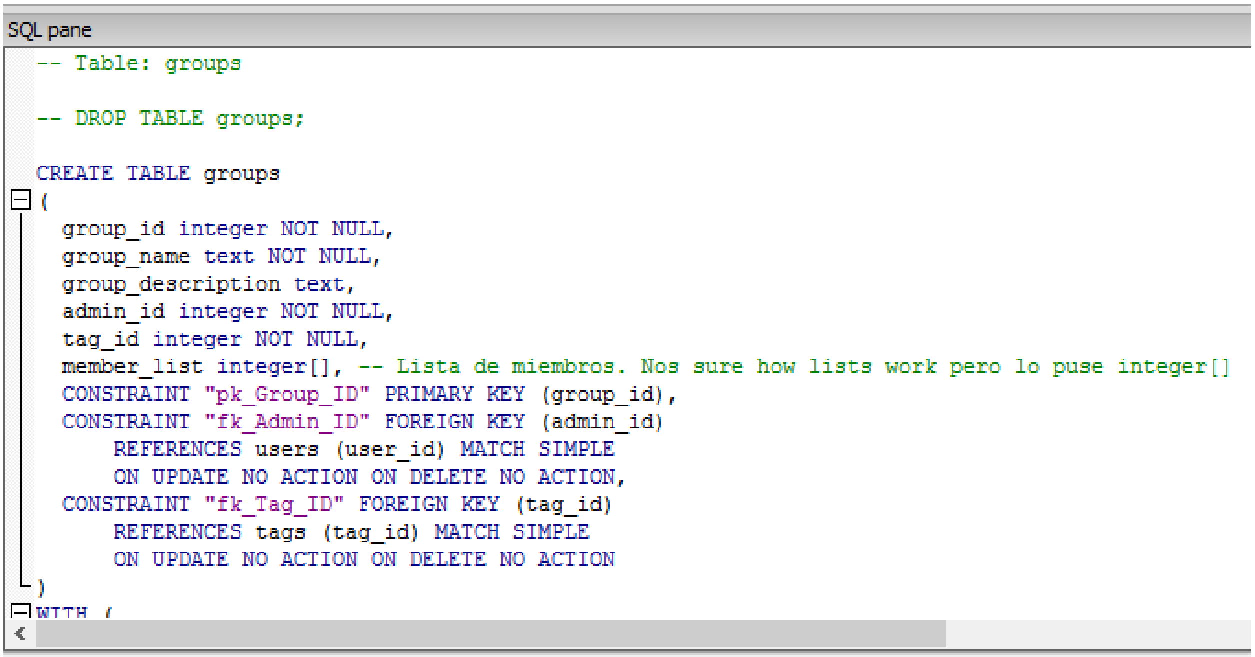
```
CREATE TABLE favorites
(
  favorite_id integer NOT NULL,
  tag_id integer NOT NULL,
  user_id integer NOT NULL, -- Un foreign key que nos dice a quien le pertenece el favorite.
  CONSTRAINT "pk_Favorites_ID" PRIMARY KEY (favorite_id),
  CONSTRAINT "fk_Tag_ID" FOREIGN KEY (tag_id)
      REFERENCES tags (tag_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_User_ID" FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE favorites
  OWNER TO ipznqcmmcmdvtq;
COMMENT ON COLUMN favorites.user_id IS 'Un foreign key que nos dice a quien le pertenece el favorite.';
```

Followers Table:



```
-- Table: followers

-- DROP TABLE followers;

CREATE TABLE followers
(
  follower_id integer NOT NULL,
  user_id integer NOT NULL,
  CONSTRAINT "pk_Followers_ID" PRIMARY KEY (follower_id),
  CONSTRAINT "fk_userID" FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "Followers_User_ID_key" UNIQUE (user_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE followers
  OWNER TO ipznqcmmcmdvtq;
```

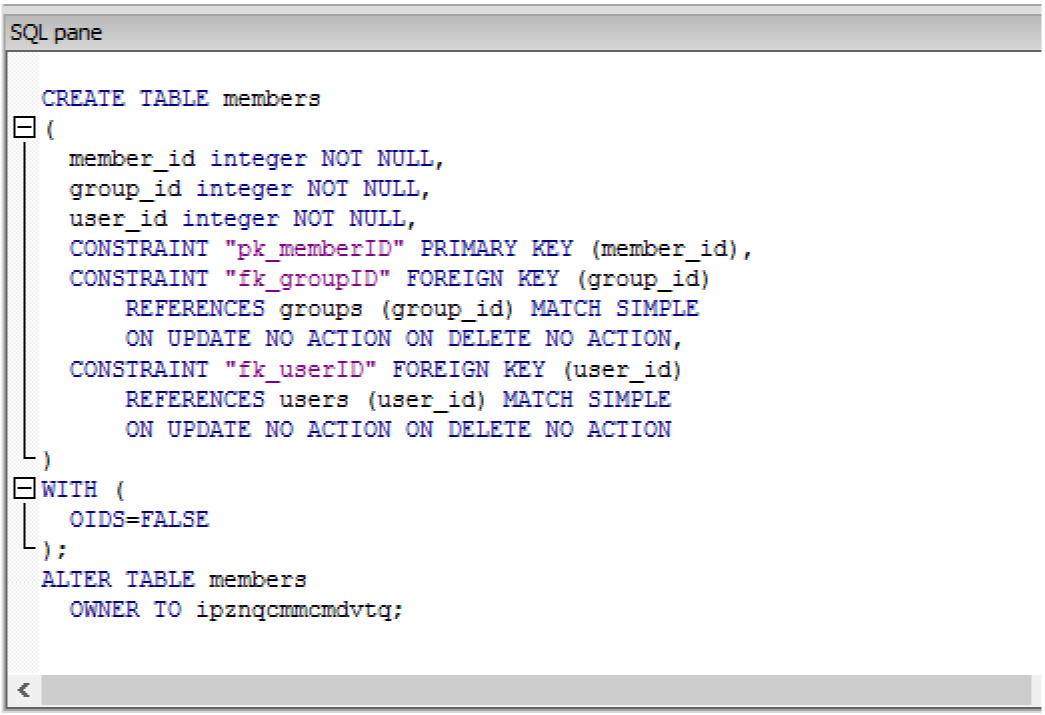
Groups Table:



SQL pane

```sql
-- Table: groups

-- DROP TABLE groups;

CREATE TABLE groups
(
  group_id integer NOT NULL,
  group_name text NOT NULL,
  group_description text,
  admin_id integer NOT NULL,
  tag_id integer NOT NULL,
  member_list integer[], -- Lista de miembros. Nos sure how lists work pero lo puse integer[]
  CONSTRAINT "pk_Group_ID" PRIMARY KEY (group_id),
  CONSTRAINT "fk_Admin_ID" FOREIGN KEY (admin_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_Tag_ID" FOREIGN KEY (tag_id)
      REFERENCES tags (tag_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
```

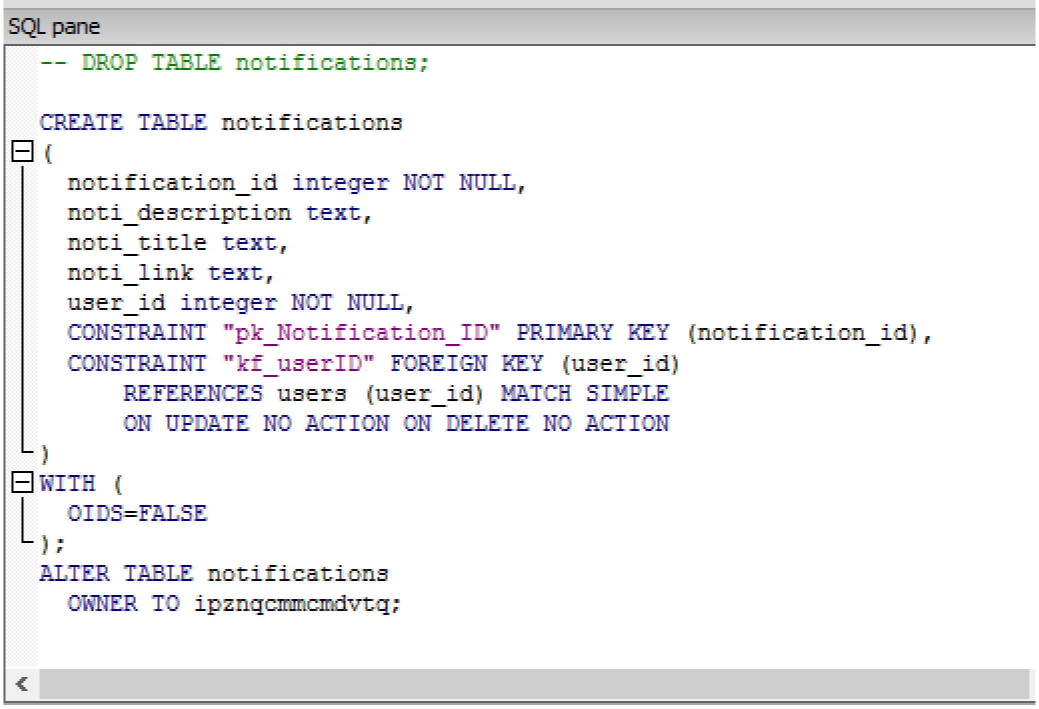
Members Table:



SQL pane

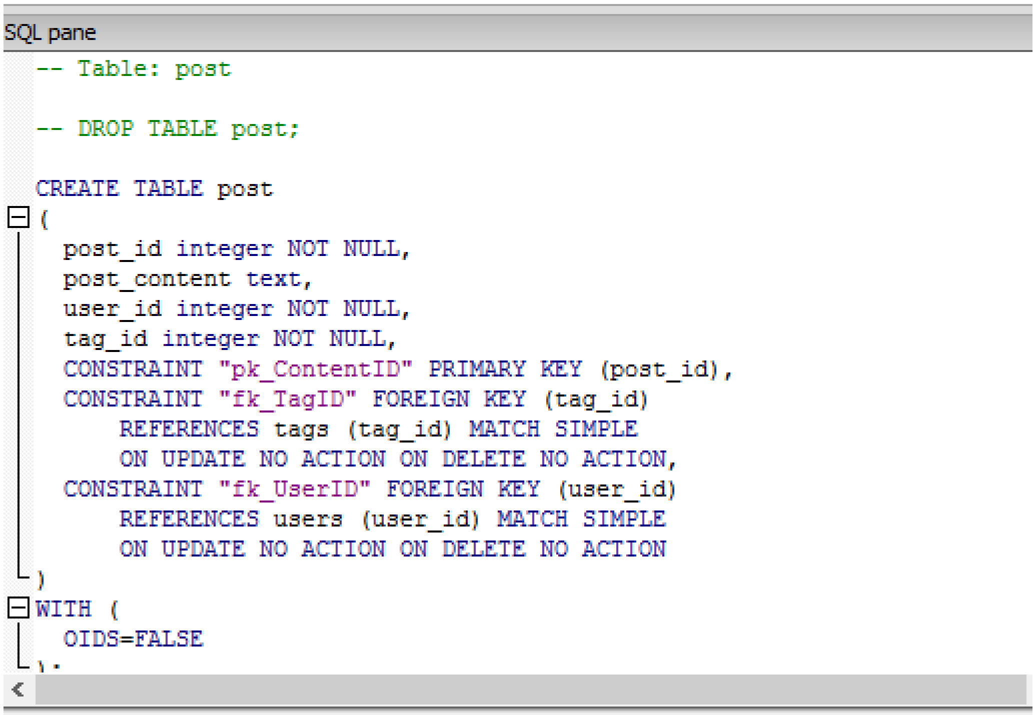
```sql
CREATE TABLE members
(
  member_id integer NOT NULL,
  group_id integer NOT NULL,
  user_id integer NOT NULL,
  CONSTRAINT "pk_memberID" PRIMARY KEY (member_id),
  CONSTRAINT "fk_groupID" FOREIGN KEY (group_id)
      REFERENCES groups (group_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_userID" FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE members
  OWNER TO ipznqcmmcmdvtq;
```

Notifications Table:



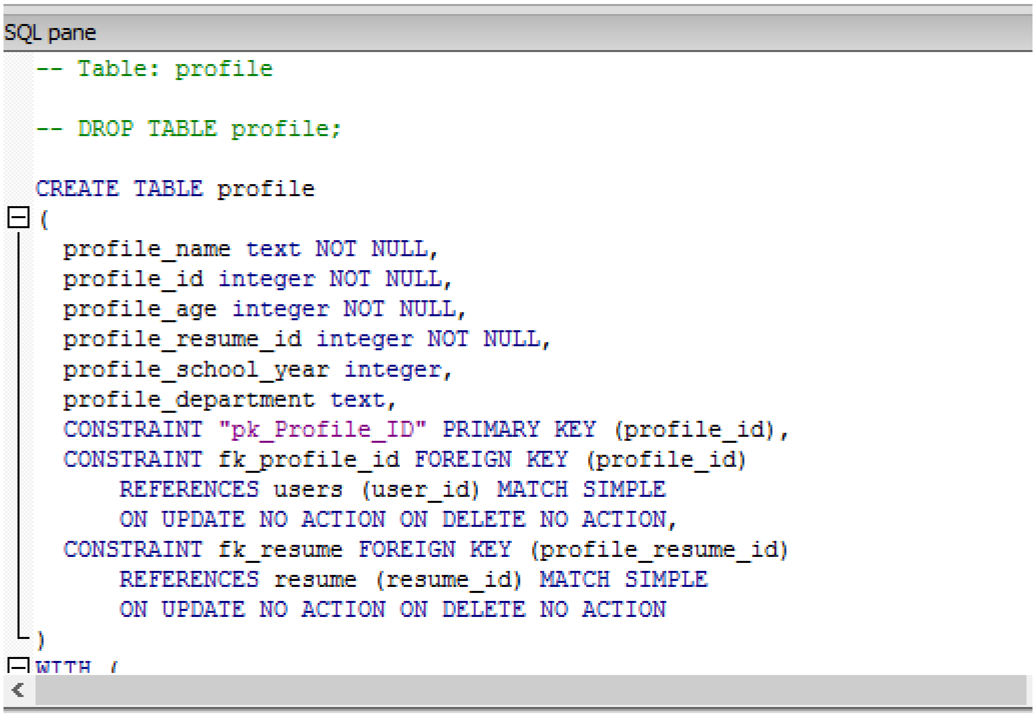
```sql
-- DROP TABLE notifications;

CREATE TABLE notifications
(
  notification_id integer NOT NULL,
  noti_description text,
  noti_title text,
  noti_link text,
  user_id integer NOT NULL,
  CONSTRAINT "pk_Notification_ID" PRIMARY KEY (notification_id),
  CONSTRAINT "kf_userID" FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE notifications
  OWNER TO ipznqcmmcmdvtq;
```

Post Table:



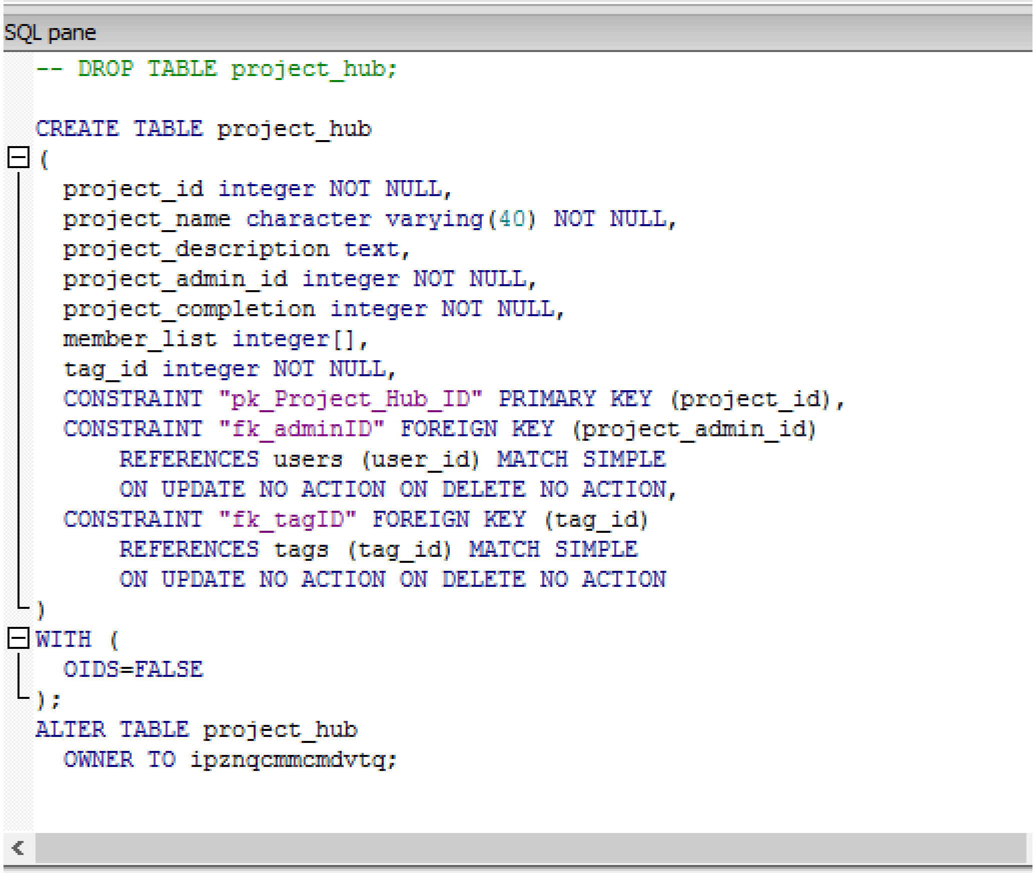
```sql
-- Table: post

-- DROP TABLE post;

CREATE TABLE post
(
  post_id integer NOT NULL,
  post_content text,
  user_id integer NOT NULL,
  tag_id integer NOT NULL,
  CONSTRAINT "pk_ContentID" PRIMARY KEY (post_id),
  CONSTRAINT "fk_TagID" FOREIGN KEY (tag_id)
      REFERENCES tags (tag_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_UserID" FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Profile Table:



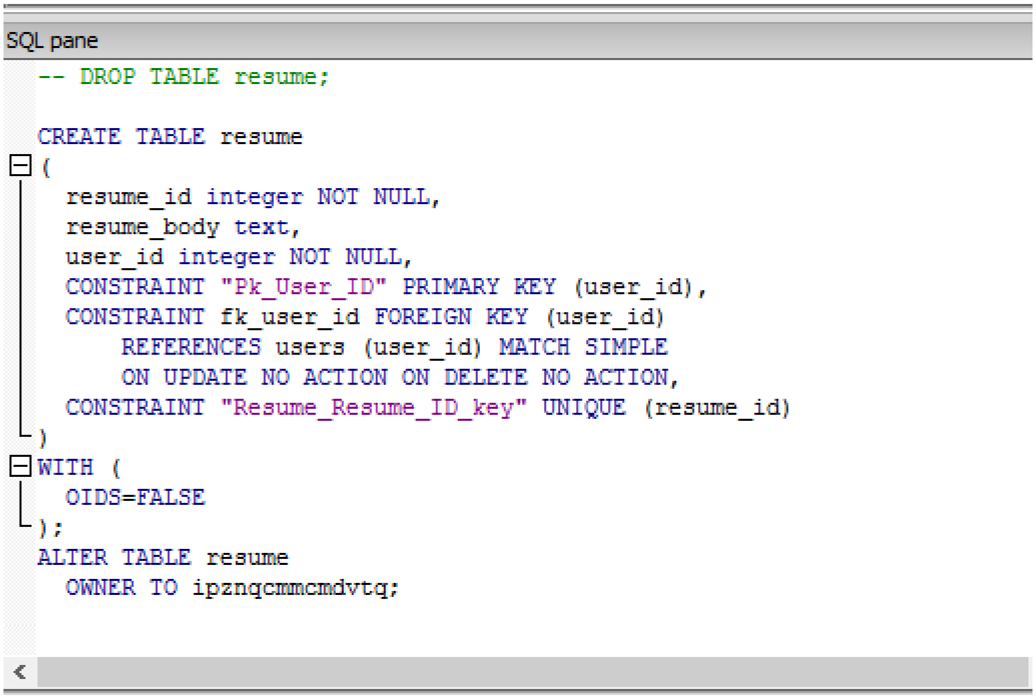
```
-- Table: profile

-- DROP TABLE profile;

CREATE TABLE profile
(
  profile_name text NOT NULL,
  profile_id integer NOT NULL,
  profile_age integer NOT NULL,
  profile_resume_id integer NOT NULL,
  profile_school_year integer,
  profile_department text,
  CONSTRAINT "pk_Profile_ID" PRIMARY KEY (profile_id),
  CONSTRAINT fk_profile_id FOREIGN KEY (profile_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_resume FOREIGN KEY (profile_resume_id)
      REFERENCES resume (resume_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
```

Project Hub Table:



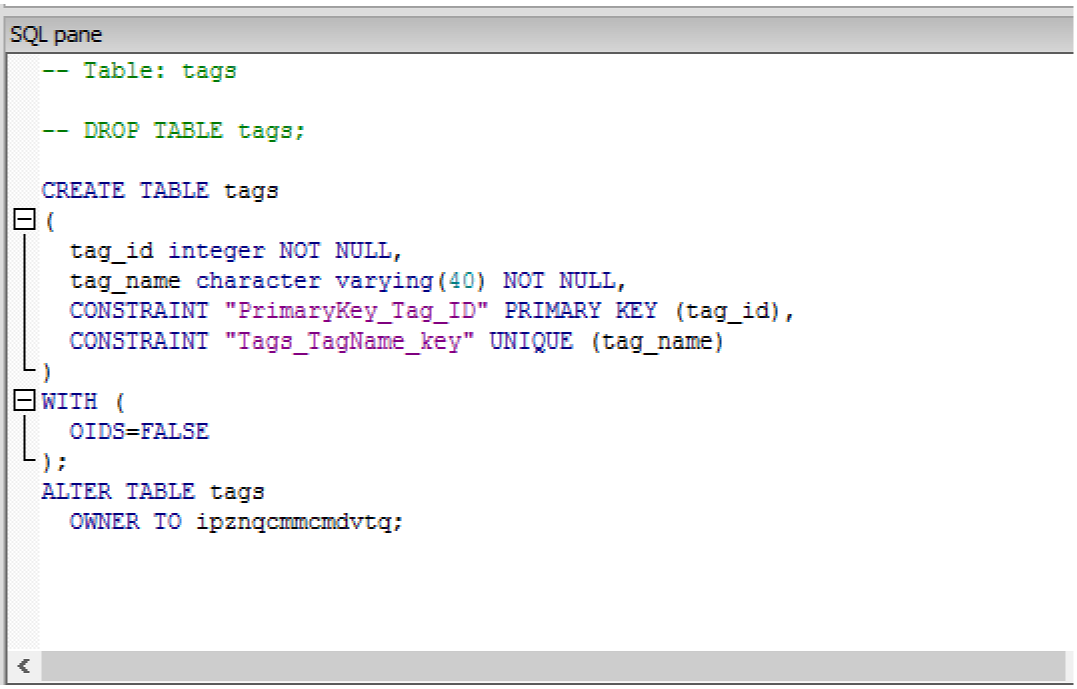
```
-- DROP TABLE project_hub;

CREATE TABLE project_hub
(
  project_id integer NOT NULL,
  project_name character varying(40) NOT NULL,
  project_description text,
  project_admin_id integer NOT NULL,
  project_completion integer NOT NULL,
  member_list integer[],
  tag_id integer NOT NULL,
  CONSTRAINT "pk_Project_Hub_ID" PRIMARY KEY (project_id),
  CONSTRAINT "fk_adminID" FOREIGN KEY (project_admin_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fk_tagID" FOREIGN KEY (tag_id)
      REFERENCES tags (tag_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE project_hub
  OWNER TO ipznqcmmcmdvtq;
```

Resume Table:



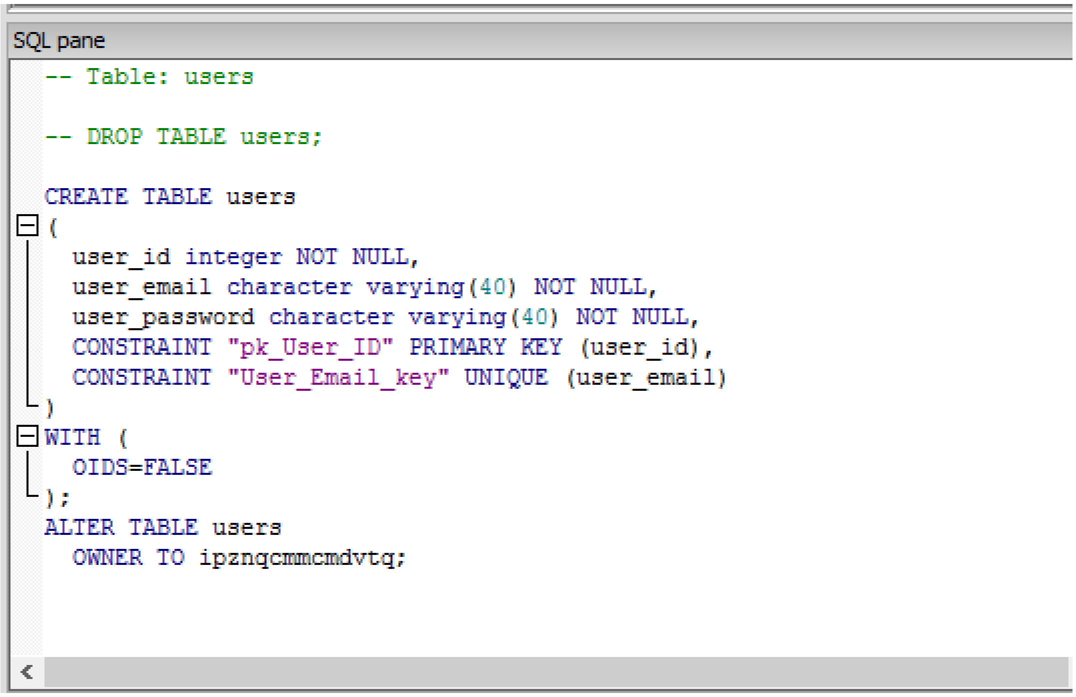
```
-- DROP TABLE resume;

CREATE TABLE resume
(
  resume_id integer NOT NULL,
  resume_body text,
  user_id integer NOT NULL,
  CONSTRAINT "Pk_User_ID" PRIMARY KEY (user_id),
  CONSTRAINT fk_user_id FOREIGN KEY (user_id)
      REFERENCES users (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "Resume_Resume_ID_key" UNIQUE (resume_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE resume
  OWNER TO ipznqcmmcmdvtq;
```

Tag Table:



```
-- Table: tags

-- DROP TABLE tags;

CREATE TABLE tags
(
  tag_id integer NOT NULL,
  tag_name character varying(40) NOT NULL,
  CONSTRAINT "PrimaryKey_Tag_ID" PRIMARY KEY (tag_id),
  CONSTRAINT "Tags_TagName_key" UNIQUE (tag_name)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE tags
  OWNER TO ipznqcmmcmdvtq;
```

User Table:



```
-- Table: users

-- DROP TABLE users;

CREATE TABLE users
(
  user_id integer NOT NULL,
  user_email character varying(40) NOT NULL,
  user_password character varying(40) NOT NULL,
  CONSTRAINT "pk_User_ID" PRIMARY KEY (user_id),
  CONSTRAINT "User_Email_key" UNIQUE (user_email)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE users
  OWNER TO ipznqcmmcmdvtq;
```