DBMS ASSIGNMENT

S NIKHIL RAM
PES1201801972
4-B (56)

PROBLEM STATEMENT

Consider a **version control management system(like GitHub)** that is described as follows. A user is defined with a unique user name,email id, name.A user must be at least 18 years of age.A user can have a premium or normal account and might have an associated company. Each user can create/own a repository or can contribute to an existing repository.A repository is defined by a unique repository name,owner and is stored at a particular server that is represented by an IP address. The time stamp of the creation time and brief description of the repository are stored.Users can "star" repositories. Each repository contains a list of files, each file having a unique name and type. Where the type can be python code/ C code/ ASCII text/ SQL file/ image etc. and its associated size in bytes.A commit is a modification to a file in a repository by a user. A commit is defined by a unique commit ID, commit type(insertion,deletion) and time stamp, with a brief description about the commit.Github is run by a team of employees managing various servers. An employee has a name, email, job description and the server they manage. All employees in a given job get the same amount of salary.A server is defined by a unique IP address and a location. A server can contain multiple repositories.

FUNCTIONAL DEPENDENCIES

- A user is defined with a unique user name,email id, name.
- A user must be at least 18 years of age.
- A user can have a premium or normal account and might have an associated company

  User_name -> email_id,name,age

  User_name -> account_type, company
- Each user can create/own a repository or can contribute to an existing repository.
- A repository is defined by a unique repository name,owner and is stored at a particular server that is represented by an IP address. The time stamp of the creation time and brief description of the repository are stored.

  Repository_name -> owner_name, IP, Ctime
- Users can "star" repositories.

  Repository_name, User -> Rating

- Each repository contains a list of files, each file having a unique name and type. Where the type can be python code/ C code/ ASCII text/ SQL file/ image etc. and its associated size in bytes.

  File_extension->File_type
  Repository,File_name,File_extension -> file_size

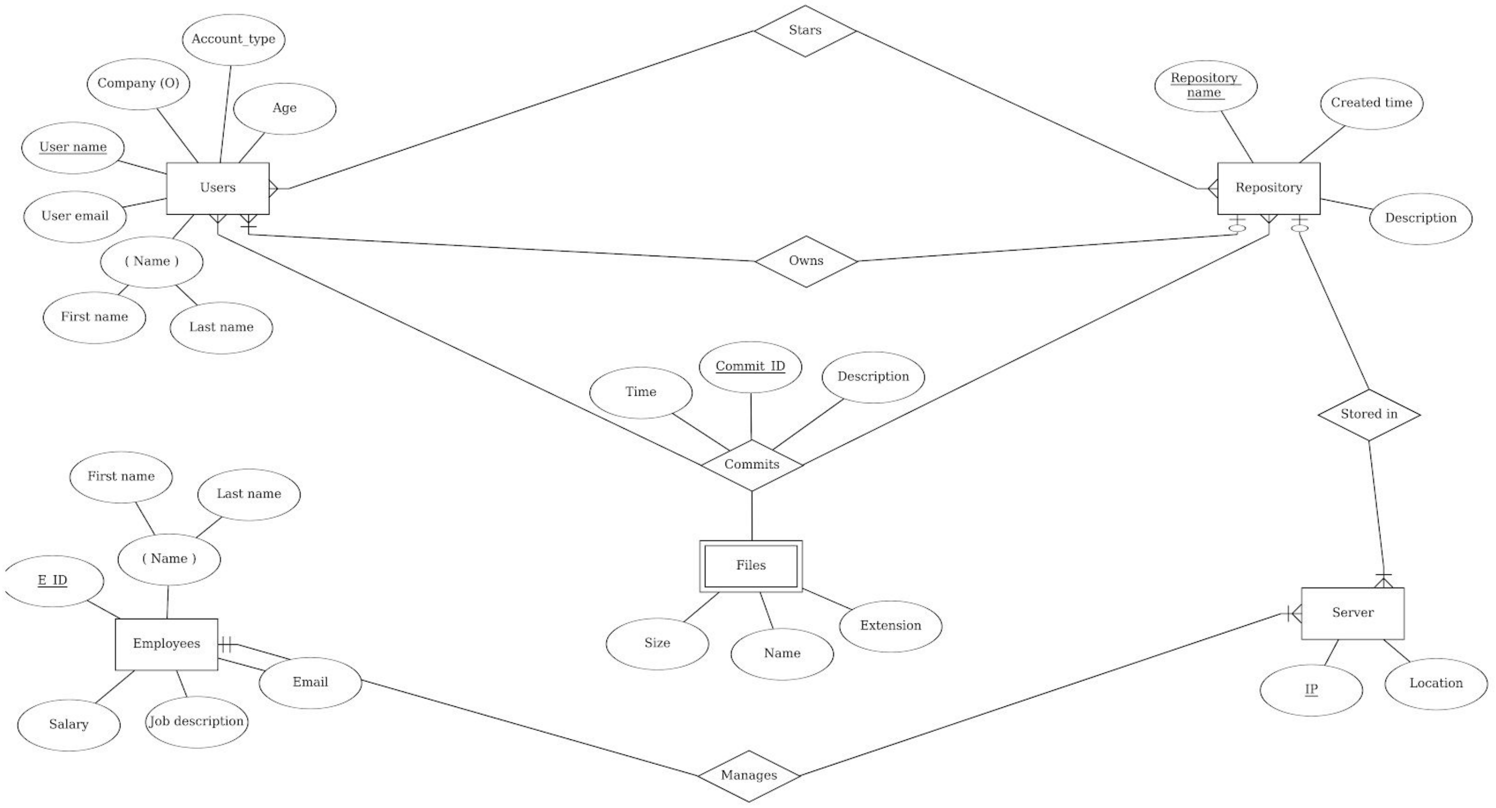
- A commit is a modification to a file in a repository by a user. A commit is defined by a unique commit ID, commit type(insertion,deletion) and time stamp, with a brief description about the commit.

  Commit ID -> description, time stamp
  Commit ID -> User ,Repository,File

- Github is run by a team of employees managing various servers. An employee has a name, email, job description and the server they manage. All employees in a given job get the same amount of salary.
- A server is defined by a unique IP address and a location. A server can contain multiple repositories.

  Employee_id->name,email,job_desciption,IP
  Job description -> salary
  Therefore, Employee_id -> salary
  IP -> location

ER DIAGRAM (BASED ON FDs')



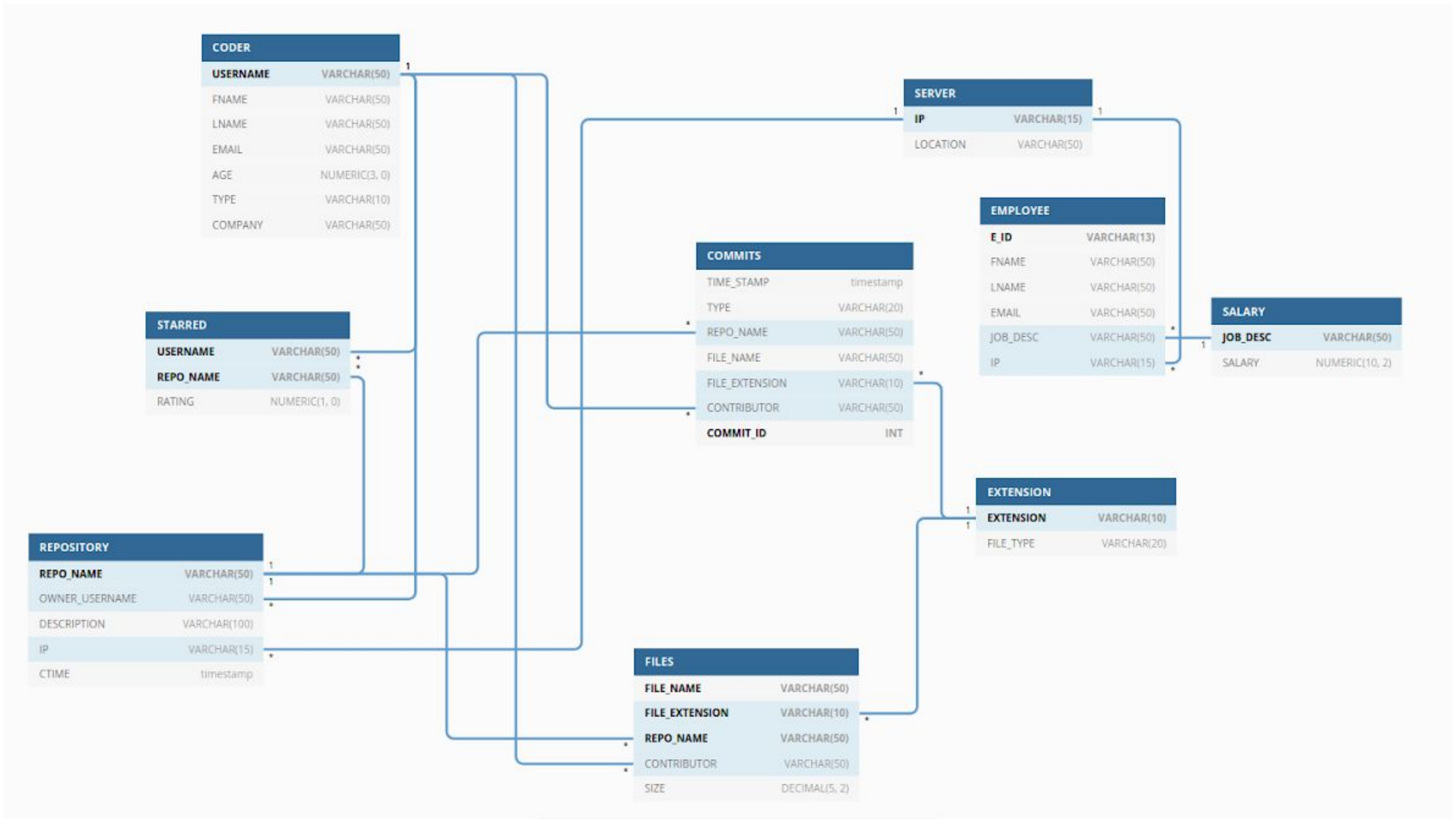
NORMALISATION

Normalisation is performed in order to reduce *data-redundancy* as well as increase *relational-consistency*. Hence these are the levels of protocols which are needed to be taken in steps to maintain a normalised database.

1. FIRST NORMAL FORM
    a. The current database follows the first normal form in all respects.
    b. All multivalued attributes are spread across each record
2. SECOND NORMAL FORM
    a. The database follows the second normal form in all respects.
    b. Its done by storing all partial dependency based attributes into separate records
    c. Eg:- The file_type can be found from file extension, hence its stores in a separate table.
    d. Eg:- The salary of the employee can be determined by his job_decription hence is stored in a separate table
3. THIRD NORMAL FORM

a. The database follows the third normal form in all respects.
b. It's done by storing transitively dependent records in a separate table with 2 foreign keys referring to the primary keys of the 2 main table.
c. Eg:- Starred table consists of rating which depends on both the user and the repository, hence there is a separate table used to maintain the ratings.

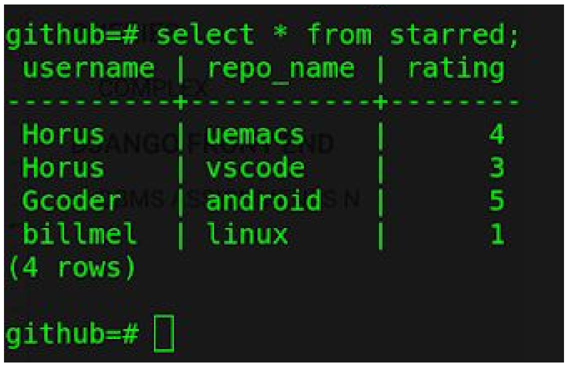
NORMALISED RDBMS (SCHEMA)



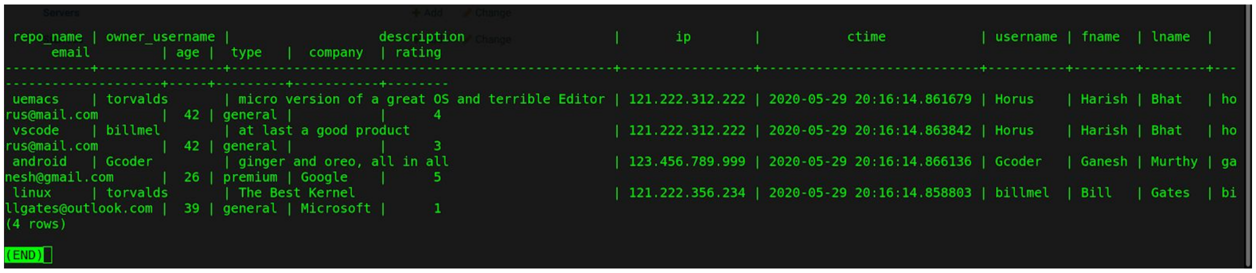
LOSSLESS-JOIN

As the entire Database has been built up in such a manner that normalisation upto 3NF is maintained in all the relations, hence all the Partial and the Transitive dependencies are eliminated bottom-up in the construction of the RDBMS itself. Hence the Database didn't require any decomposition of tables into smaller tables which require lossless join verification to proceed.
Anyhow, there can be some examples pointed out where lossless-join occurrence can be demonstrated.

- SELECT * FROM STARRED gives the same sub-tuples as SELECT * FROM REPOSITORY NATURAL JOIN (SELECT * FROM CODER NATURAL JOIN STARRED) AS CJ;



```
github=# select * from starred;
 username | repo_name | rating
----------+-----------+--------
 Horus    | uemacs    |      4
 Horus    | vscode    |      3
 Gcoder   | android   |      5
 billmel  | linux     |      1
(4 rows)

github=# 
```



```
 repo_name | owner_username |                  description                    |       ip        |           ctime            | username | fname  | lname  |
     email          | age |  type   |  company  | rating
-----------+----------------+-------------------------------------------------+-----------------+----------------------------+----------+--------+--------+---
-------------------+-----+---------+-----------+--------
 uemacs    | torvalds       | micro version of a great OS and terrible Editor | 121.222.312.222 | 2020-05-29 20:16:14.861679 | Horus    | Harish | Bhat   | ho
rus@mail.com        |  42 | general |           |      4
 vscode    | billmel        | at last a good product                          | 121.222.312.222 | 2020-05-29 20:16:14.863842 | Horus    | Harish | Bhat   | ho
rus@mail.com        |  42 | general |           |      3
 android   | Gcoder         | ginger and oreo, all in all                     | 123.456.789.999 | 2020-05-29 20:16:14.866136 | Gcoder   | Ganesh | Murthy | ga
nesh@gmail.com      |  26 | premium | Google    |      5
 linux     | torvalds       | The Best Kernel                                 | 121.222.356.234 | 2020-05-29 20:16:14.858803 | billmel  | Bill   | Gates  | bi
llgates@outlook.com |  39 | general | Microsoft |      1
(4 rows)

(END)
```

SQL SCHEMA

```
CREATE TABLE CODER
(USERNAME VARCHAR(50) PRIMARY KEY,
FNAME VARCHAR(50) NOT NULL,
LNAME VARCHAR(50) NOT NULL,
EMAIL VARCHAR(50) NOT NULL,
AGE NUMERIC(3,0) NOT NULL,
TYPE VARCHAR(10) NOT NULL,
COMPANY VARCHAR(50),
CHECK ( AGE>17 AND EMAIL LIKE '%@%.%' AND TYPE IN ('general','premium'))
);

CREATE TABLE SERVER
(
IP VARCHAR(15),
LOCATION VARCHAR(50) NOT NULL,
PRIMARY KEY(IP),
CHECK (IP LIKE '___.___.___.___' AND
```

```sql
LOCATION IN
('Bangalore','Albuquerque','Medellin','London','Kampala','Sydney'))
);

CREATE TABLE EXTENSION
(
EXTENSION VARCHAR(10),
FILE_TYPE VARCHAR(20) DEFAULT '.txt',
PRIMARY KEY(EXTENSION),
CHECK(EXTENSION LIKE '.%')
);

CREATE TABLE SALARY
(
JOB_DESC VARCHAR(50),
SALARY NUMERIC(10,2) NOT NULL,
PRIMARY KEY(JOB_DESC),
CHECK(SALARY>0)
);

CREATE TABLE REPOSITORY
(
REPO_NAME VARCHAR(50) PRIMARY KEY,
OWNER_USERNAME VARCHAR(50) NOT NULL,
DESCRIPTION VARCHAR(100),
IP VARCHAR(15) NOT NULL,
CTIME TIMESTAMP DEFAULT NOW(),
FOREIGN KEY (OWNER_USERNAME) REFERENCES CODER(USERNAME) ON DELETE CASCADE,
FOREIGN KEY (IP) REFERENCES SERVER(IP) ON DELETE CASCADE
);

CREATE TABLE STARRED
(
USERNAME VARCHAR(50) NOT NULL,
REPO_NAME VARCHAR(50) NOT NULL,
RATING NUMERIC(1,0) NOT NULL,
CHECK ( RATING>=0 AND RATING<=5),
FOREIGN KEY (USERNAME) REFERENCES CODER(USERNAME) ON DELETE CASCADE,
FOREIGN KEY (REPO_NAME) REFERENCES REPOSITORY(REPO_NAME) ON DELETE CASCADE,
PRIMARY KEY (USERNAME,REPO_NAME)
);
```

```sql
CREATE TABLE EMPLOYEE
(
E_ID VARCHAR(13) PRIMARY KEY,
FNAME VARCHAR(50) NOT NULL,
LNAME VARCHAR(50) NOT NULL,
EMAIL VARCHAR(50) NOT NULL,
JOB_DESC VARCHAR(50),
IP VARCHAR(15),
CHECK (E_ID LIKE 'HUB__________'),
CHECK (EMAIL LIKE '%@github.org'),
FOREIGN KEY (JOB_DESC) REFERENCES SALARY(JOB_DESC) ON DELETE SET NULL,
FOREIGN KEY (IP) REFERENCES SERVER(IP) ON DELETE SET NULL
);

CREATE TABLE COMMITS
(
TIME_STAMP TIMESTAMP,
TYPE VARCHAR(20),
REPO_NAME VARCHAR(50),
FILE_NAME VARCHAR(50),
FILE_EXTENSION VARCHAR(10),
CONTRIBUTOR VARCHAR(50),
COMMIT_ID INT GENERATED ALWAYS AS IDENTITY,
FOREIGN KEY (CONTRIBUTOR) REFERENCES CODER(USERNAME) ON DELETE SET NULL,
FOREIGN KEY (REPO_NAME) REFERENCES REPOSITORY(REPO_NAME) ON DELETE CASCADE
,
FOREIGN KEY (FILE_EXTENSION) REFERENCES EXTENSION(EXTENSION) ON DELETE SET
DEFAULT,
PRIMARY KEY(COMMIT_ID),
CHECK(TYPE IN ('INSERT','DELETE','UPDATE'))
);

CREATE TABLE FILES
(
FILE_NAME VARCHAR(50) NOT NULL,
FILE_EXTENSION VARCHAR(10),
REPO_NAME VARCHAR(50) NOT NULL,
CONTRIBUTOR VARCHAR(50),
SIZE DECIMAL(5,2) NOT NULL,
FOREIGN KEY (CONTRIBUTOR) REFERENCES CODER(USERNAME) ON DELETE SET NULL,
FOREIGN KEY (REPO_NAME) REFERENCES REPOSITORY(REPO_NAME) ON DELETE CASCADE
,
FOREIGN KEY (FILE_EXTENSION) REFERENCES EXTENSION(EXTENSION) ON DELETE SET
NULL,
PRIMARY KEY(FILE_NAME,REPO_NAME,FILE_EXTENSION),
CHECK(SIZE>0)
);
```

APPLICATION OF TRIGGERS

Considering that the creation, deletion and the modification of the files resulted in a Version Control System (like GitHub). There were 3 Triggers created in-order to add a record to the commit table based on changes in the diles table.

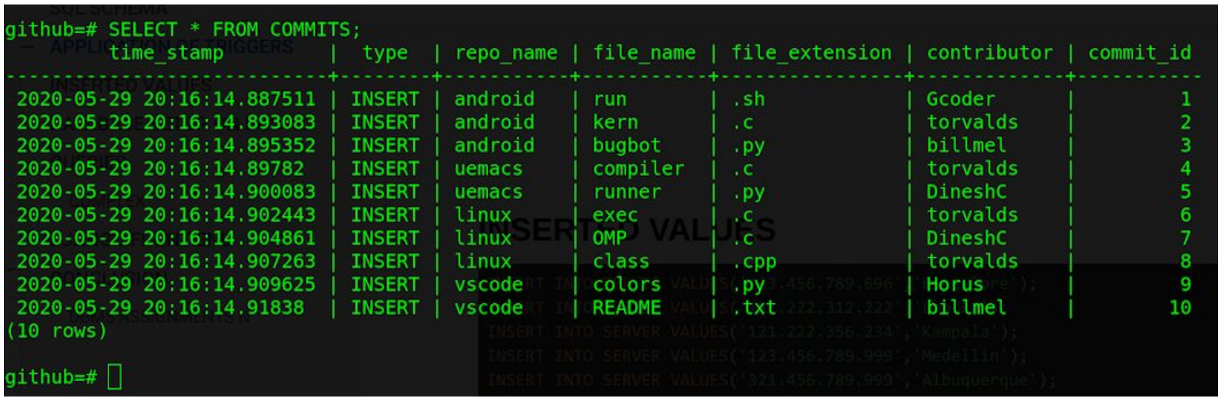
1. Insertion of a file into a repository results in an insertion of a commit of type INSERT
2. Modification of a file in a repository, which is understood by change in file size, results in an insertion of a commit of type UPDATE.
3. Deletion of a file into a repository results in an insertion of a commit of type DELETE

An additional IF-CLAUSE was present in each function called by the above triggers so that these insertions of records into the “commits” table do not occur during CASCADE (when a *repository* is deleted directly or indirectly).

```
CREATE OR REPLACE FUNCTION insert_file()
  RETURNS trigger AS
$BODY$
BEGIN
      INSERT INTO commits
values(now(),'INSERT',new.REPO_NAME,new.FILE_NAME,new.FILE_EXTENSION,new.CO
NTRIBUTOR);
      RETURN new;
END;
$BODY$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION update_file()
  RETURNS trigger AS
$BODY$
BEGIN
       IF old.REPO_NAME IN (SELECT REPO_NAME FROM REPOSITORY) THEN
      INSERT INTO commits
values(now(),'UPDATE',old.REPO_NAME,old.FILE_NAME,old.FILE_EXTENSION,old.CO
NTRIBUTOR);

      ELSE
```

```
             DELETE FROM commits WHERE REPO_NAME=old.REPO_NAME;
             END IF;
      RETURN new;
END;
$BODY$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION delete_file()
  RETURNS trigger AS
$BODY$
BEGIN
       IF old.REPO_NAME IN (SELECT REPO_NAME FROM REPOSITORY) THEN
      INSERT INTO commits
values(now(),'DELETE',old.REPO_NAME,old.FILE_NAME,old.FILE_EXTENSION,old.CO
NTRIBUTOR);

      ELSE
             DELETE FROM commits WHERE REPO_NAME=old.REPO_NAME;
             END IF;
      RETURN old;

END;
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER insert_file
  AFTER INSERT
  ON FILES
  FOR EACH ROW
  EXECUTE PROCEDURE insert_file();

CREATE TRIGGER update_file
  BEFORE UPDATE
  ON FILES
  FOR EACH ROW
  EXECUTE PROCEDURE update_file();

CREATE TRIGGER delete_file
  BEFORE DELETE
  ON FILES
  FOR EACH ROW
  EXECUTE PROCEDURE delete_file();
```

The following Commit Table records are filled only through Triggers



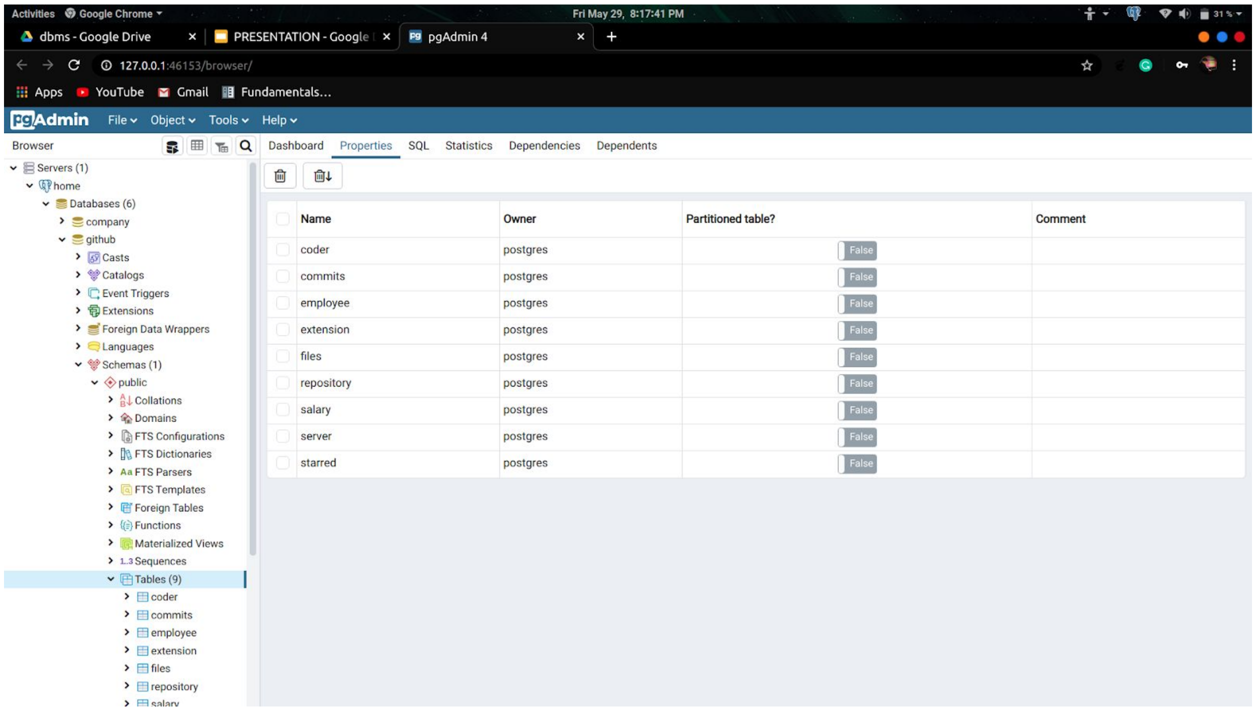
```
github=# SELECT * FROM COMMITS;
         time_stamp          |  type  | repo_name | file_name | file_extension | contributor | commit_id
-----------------------------+--------+-----------+-----------+----------------+-------------+-----------
 2020-05-29 20:16:14.887511 | INSERT | android   | run       | .sh            | Gcoder      |         1
 2020-05-29 20:16:14.893083 | INSERT | android   | kern      | .c             | torvalds    |         2
 2020-05-29 20:16:14.895352 | INSERT | android   | bugbot    | .py            | billmel     |         3
 2020-05-29 20:16:14.89782  | INSERT | uemacs    | compiler  | .c             | torvalds    |         4
 2020-05-29 20:16:14.900083 | INSERT | uemacs    | runner    | .py            | DineshC     |         5
 2020-05-29 20:16:14.902443 | INSERT | linux     | exec      | .c             | torvalds    |         6
 2020-05-29 20:16:14.904861 | INSERT | linux     | OMP       | .c             | DineshC     |         7
 2020-05-29 20:16:14.907263 | INSERT | linux     | class     | .cpp           | torvalds    |         8
 2020-05-29 20:16:14.909625 | INSERT | vscode    | colors    | .py            | Horus       |         9
 2020-05-29 20:16:14.91838  | INSERT | vscode    | README    | .txt           | billmel     |        10
(10 rows)

github=# 
```

INSERTED VALUES

```
INSERT INTO SERVER VALUES('123.456.789.696','Bangalore');
INSERT INTO SERVER VALUES('121.222.312.222','London');
INSERT INTO SERVER VALUES('121.222.356.234','Kampala');
INSERT INTO SERVER VALUES('123.456.789.999','Medellin');
INSERT INTO SERVER VALUES('321.456.789.999','Albuquerque');
INSERT INTO SERVER VALUES('321.389.789.999','Sydney');
INSERT INTO CODER
VALUES('Horus','Harish','Bhat','horus@mail.com',42,'general');
INSERT INTO CODER
VALUES('Gcoder','Ganesh','Murthy','ganesh@gmail.com',26,'premium','Google')
;
INSERT INTO CODER
VALUES('DineshC','Dinesh','Chakraborty','dinesh@yahoo.com',30,'general');
INSERT INTO CODER
VALUES('torvalds','Linus','Torvalds','linus@torvalds.com',60,'premium');
INSERT INTO CODER
VALUES('billmel','Bill','Gates','billgates@outlook.com',39,'general','Micro
soft');
INSERT INTO EXTENSION VALUES('.py','python');
INSERT INTO EXTENSION VALUES('.txt','ASCII text');
INSERT INTO EXTENSION VALUES('.c','C');
INSERT INTO EXTENSION VALUES('.cpp','C++');
INSERT INTO EXTENSION VALUES('.sh','bash');
INSERT INTO SALARY VALUES('CEO',300000);
INSERT INTO SALARY VALUES('Manager',200000);
INSERT INTO SALARY VALUES('Superintendent',250000);
INSERT INTO REPOSITORY VALUES('linux','torvalds','The Best
Kernel','121.222.356.234');
```

```sql
INSERT INTO REPOSITORY VALUES('uemacs','torvalds','micro version of a great
OS and terrible Editor','121.222.312.222');
INSERT INTO REPOSITORY VALUES('vscode','billmel','at last a good
product','121.222.312.222');
INSERT INTO REPOSITORY VALUES('android','Gcoder','ginger and oreo, all in
all','123.456.789.999');
INSERT INTO STARRED VALUES('Horus','uemacs',4);
INSERT INTO STARRED VALUES('Horus','vscode',3);
INSERT INTO STARRED VALUES('Gcoder','android',5);
INSERT INTO STARRED VALUES('billmel','linux',1);
INSERT INTO EMPLOYEE
VALUES('HUB0000000001','Nat','Friedman','natfreid@github.org','CEO','123.45
6.789.696');
INSERT INTO EMPLOYEE
VALUES('HUB0000000002','Ramesh','Ranganathan','rr@github.org','Manager','12
1.222.312.222');
INSERT INTO EMPLOYEE
VALUES('HUB0000000003','Sheela','Ramani','sheela@github.org','Superintenden
t','121.222.312.222');
INSERT INTO FILES VALUES('run','.sh','android','Gcoder',10);
INSERT INTO FILES VALUES('kern','.c','android','torvalds',20);
INSERT INTO FILES VALUES('bugbot','.py','android','billmel',30);
INSERT INTO FILES VALUES('compiler','.c','uemacs','torvalds',40);
INSERT INTO FILES VALUES('runner','.py','uemacs','DineshC',50);
INSERT INTO FILES VALUES('exec','.c','linux','torvalds',15);
INSERT INTO FILES VALUES('OMP','.c','linux','DineshC',25);
INSERT INTO FILES VALUES('class','.cpp','linux','torvalds',35);
INSERT INTO FILES VALUES('colors','.py','vscode','Horus',45);
INSERT INTO FILES VALUES('README','.txt','vscode','billmel',45);
```

DATABASE SETUP ON POSTGRESQL

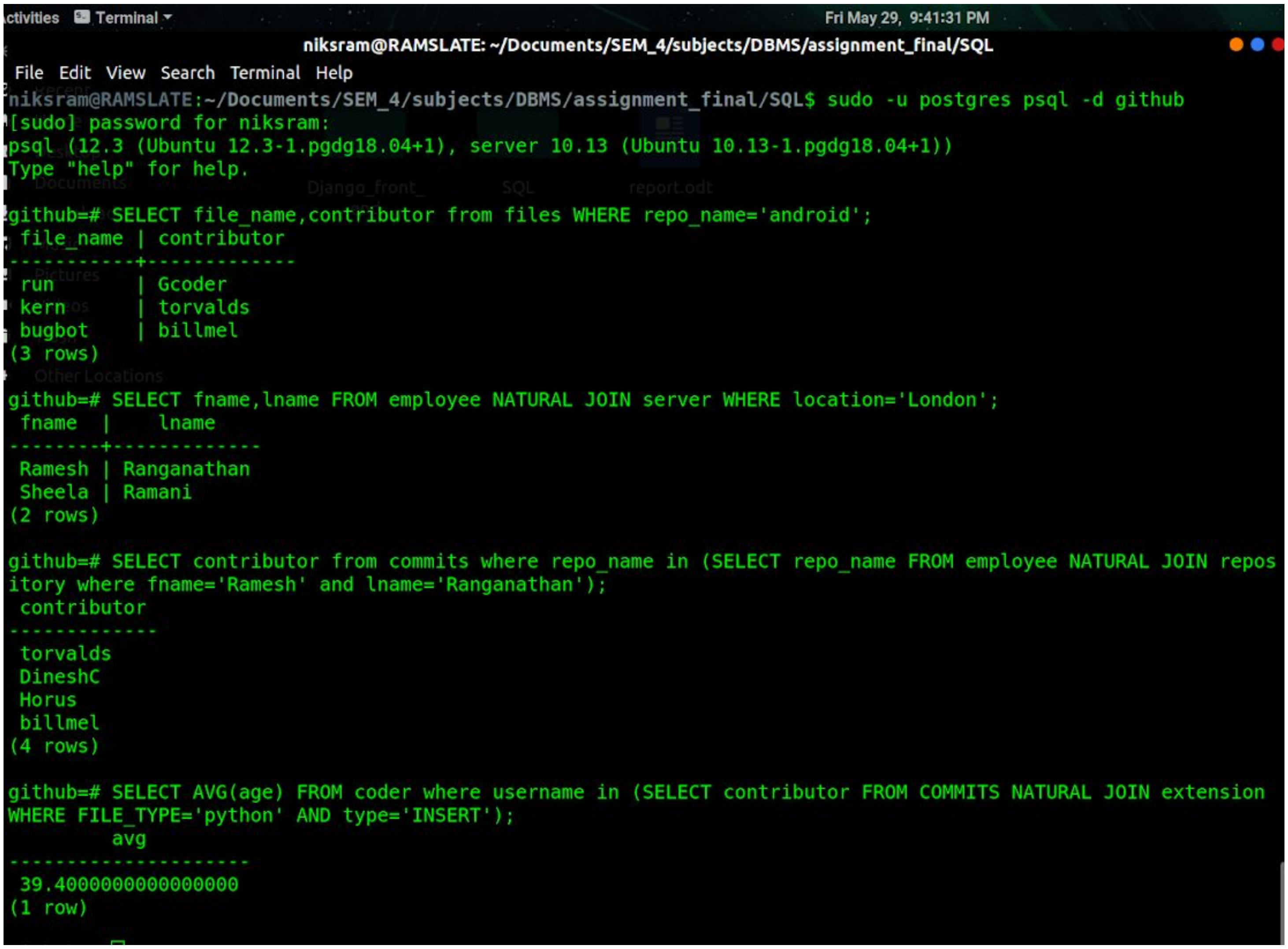


- Servers (1)
  - home
    - Databases (6)
      - company
      - github
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Schemas (1)
          - public
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Sequences
            - Tables (9)
              - coder
              - commits
              - employee
              - extension
              - files
              - repository
              - salary

Name	Owner	Partitioned table?	Comment
coder	postgres	False	
commits	postgres	False	
employee	postgres	False	
extension	postgres	False	
files	postgres	False	
repository	postgres	False	
salary	postgres	False	
server	postgres	False	
starred	postgres	False	

QUERIES

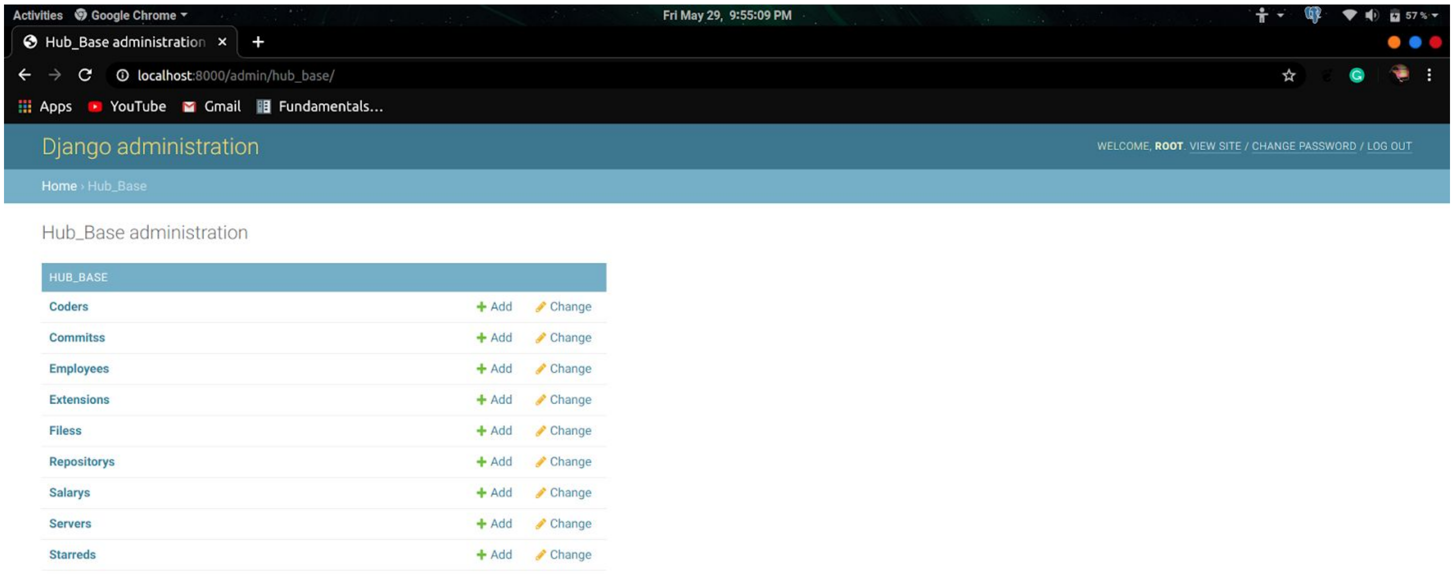
- **SIMPLE**
  - **all contributors in android repository**
    - `SELECT file_name,contributor from files WHERE repo_name='android';`
  - **all employees working on the london server**
    - `SELECT fname,lname FROM employee NATURAL JOIN server WHERE location='London';`
- COMPLEX
  - **all contributors to the server managed by Ramesh Ranganathan**
    - `SELECT contributor from commits where repo_name in (SELECT repo_name FROM employee NATURAL JOIN repository where fname='Ramesh' and lname='Ranganathan');`
  - **average age of python coders**
    - `SELECT AVG(age) FROM coder where username in (SELECT contributor FROM COMMITS NATURAL JOIN extension WHERE FILE_TYPE='python' AND type='INSERT');`


Activities
Terminal
Fri May 29, 9:41:31 PM
niksram@RAMSLATE: ~/Documents/SEM_4/subjects/DBMS/assignment_final/SQL
File Edit View Search Terminal Help

```
niksram@RAMSLATE:~/Documents/SEM_4/subjects/DBMS/assignment_final/SQL$ sudo -u postgres psql -d github
[sudo] password for niksram:
psql (12.3 (Ubuntu 12.3-1.pgdg18.04+1), server 10.13 (Ubuntu 10.13-1.pgdg18.04+1))
Type "help" for help.

github=# SELECT file_name,contributor from files WHERE repo_name='android';
 file_name | contributor
-----------+-------------
 run       | Gcoder
 kern      | torvalds
 bugbot    | billmel
(3 rows)

github=# SELECT fname,lname FROM employee NATURAL JOIN server WHERE location='London';
 fname  |    lname
--------+-------------
 Ramesh | Ranganathan
 Sheela | Ramani
(2 rows)

github=# SELECT contributor from commits where repo_name in (SELECT repo_name FROM employee NATURAL JOIN repos
itory where fname='Ramesh' and lname='Ranganathan');
 contributor
-------------
 torvalds
 DineshC
 Horus
 billmel
(4 rows)

github=# SELECT AVG(age) FROM coder where username in (SELECT contributor FROM COMMITS NATURAL JOIN extension
WHERE FILE_TYPE='python' AND type='INSERT');
         avg
---------------------
 39.4000000000000000
(1 row)
```

DJANGO FRONT END

A simple front-end of the database is created with the help of the ORM model generated by DJANGO. The ORM classes were generated by *"inspectdb"* which converts our SQL schema into PYTHON CLASSES.



Hub_Base administration

HUB_BASE		
Coders	+ Add	Change
Commitss	+ Add	Change
Employees	+ Add	Change
Extensions	+ Add	Change
Filess	+ Add	Change
Repositorys	+ Add	Change
Salarys	+ Add	Change
Servers	+ Add	Change
Starreds	+ Add	Change

CONCLUSION

A Good Sequel Database is constructed by following procedures of constructing

- ER-Diagram with constraints as well as Relationship types.
- RDBMS constructed by maintaining Normalisation
- Incorporating Check and Not-Null constraints for the attributes
- Cascades so that normalisation does not distort on updation and deletion
- Appropriate triggers if necessary
- Verification of lossless join if decomposition of tables were performed at the schema level

Such a Database shall have a very less memory-footprint but at the same time shall maintain a Rich-Consistency in maintaining the Integrity between Relations.