Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS3041 Information Management II

SQL Project – Source Code Management Design

School of Computer Science and Statistics

(e-Report submission)

Table of Content

Find the project on my GitHub: [rohantaneja/git-scm-db]

Submitted by:

Rohan Taneja
19323238

Submission: 13/12/2020

Introduction

This project is based on the ideology of GitHub as a whole. I chose to model a modern source code management platform such as Git which is widely used seamless workflow management tool. SCM is an extensively used tool in the world of computer science to maintain, and track changes taking place in a repo (referred to a repository) which makes it easier to resolve conflicts when any new commit (reference for change) takes place from the users contributing to it. Git is something I have been accustomed to since I stepped into the world of open-source software. Initial release of Git dates to 2005, which was made with public license by Linus Torvalds – the author of Linux Kernel. During my time as an android developer – I have worked on many projects which required a lot of continuous changes (especially in kernel development) and multiple developers made them. Also, this modern SCM approach is based on distributed version control which makes peer-to-peer synchronization of version more convenient keeping a completely working local copy with localized tracked changes.

The SCM model approach

We are defining the database as **GitDB**. Consisting following entities with their respective attributes.

- **Organization** – unique identity in the system – name, description, email address.
- **User** – can be part of an organisation – unique identity in the system – consists of their avatar, name, email address, description, followers, and following.
- **Repo** – unique identity (referred when forked), name of the repository, username who owns it (can be an organization), date of initialization, visibility parameter (public/private), reference (branch, tags), contributors on the repo.
- **Branch** – unique identity of branch, name of branch, last commit on that branch, time of update.
- **Tag** – unique identity of tag, name of tag, last commit on that branch, create date.
- **Commit** – unique identity as SHA, author of the commit, committed by user, date of commit, title of the commit, parent commit, author of commit (user committing else's commit).
- **Patch** – unique index (based on the git diff), name of old blob (in case of rename), name of new file, changes made by additions and deletion of lines of code.
- **Pull Request** – if a commit is made on other user's repo – this request is generated – have a unique identity, name of the repo, author of the commit, status of pull request, date of creation, date of closure (when status is updated).
- **Issue** – if any user faces a problem in the working of a code – an issue can be raised in the repo – unique identity, author of the issue raised, status of issue, title, description of the issue, date of creation.
- **Comment** – corresponding to a commit, pull request or an issue – has a unique identity, comment content, date of creation, author of the comment.

Now that I have covered major functionalities of a SCM in the modern world use leaving additional functions like Exploration, Marketplace & 3rd party API integrations as such in GitHub and other service providers - let's move to our first step of modelling the database – **Entity-Relationship Diagram**.

ER Diagram

The following diagram represents relationship of entities as defined in the SCM Model approach.

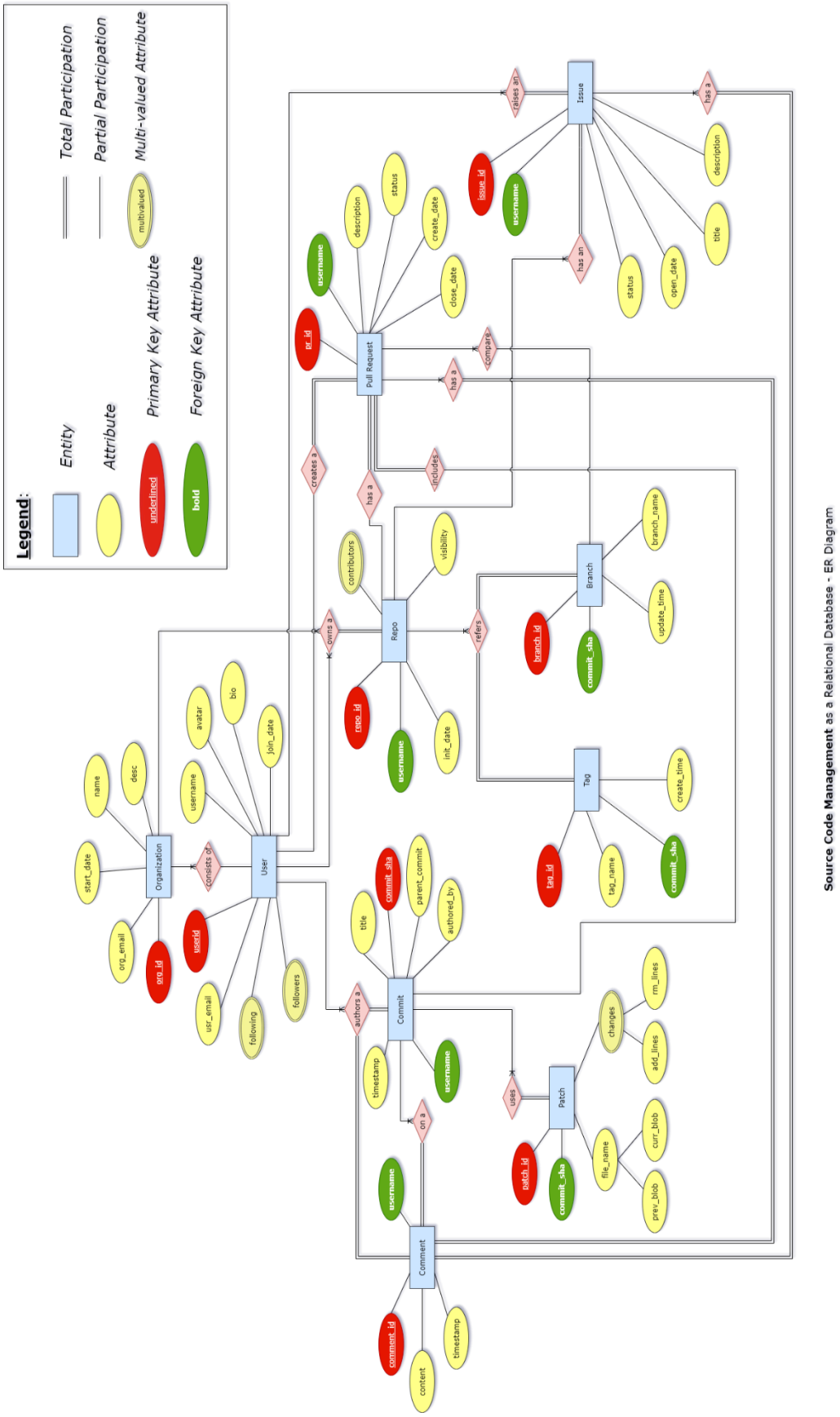

Fig 1. ER Diagram

Mapping to Relational Schema

Translating the ER Diagram to Relational Schema.

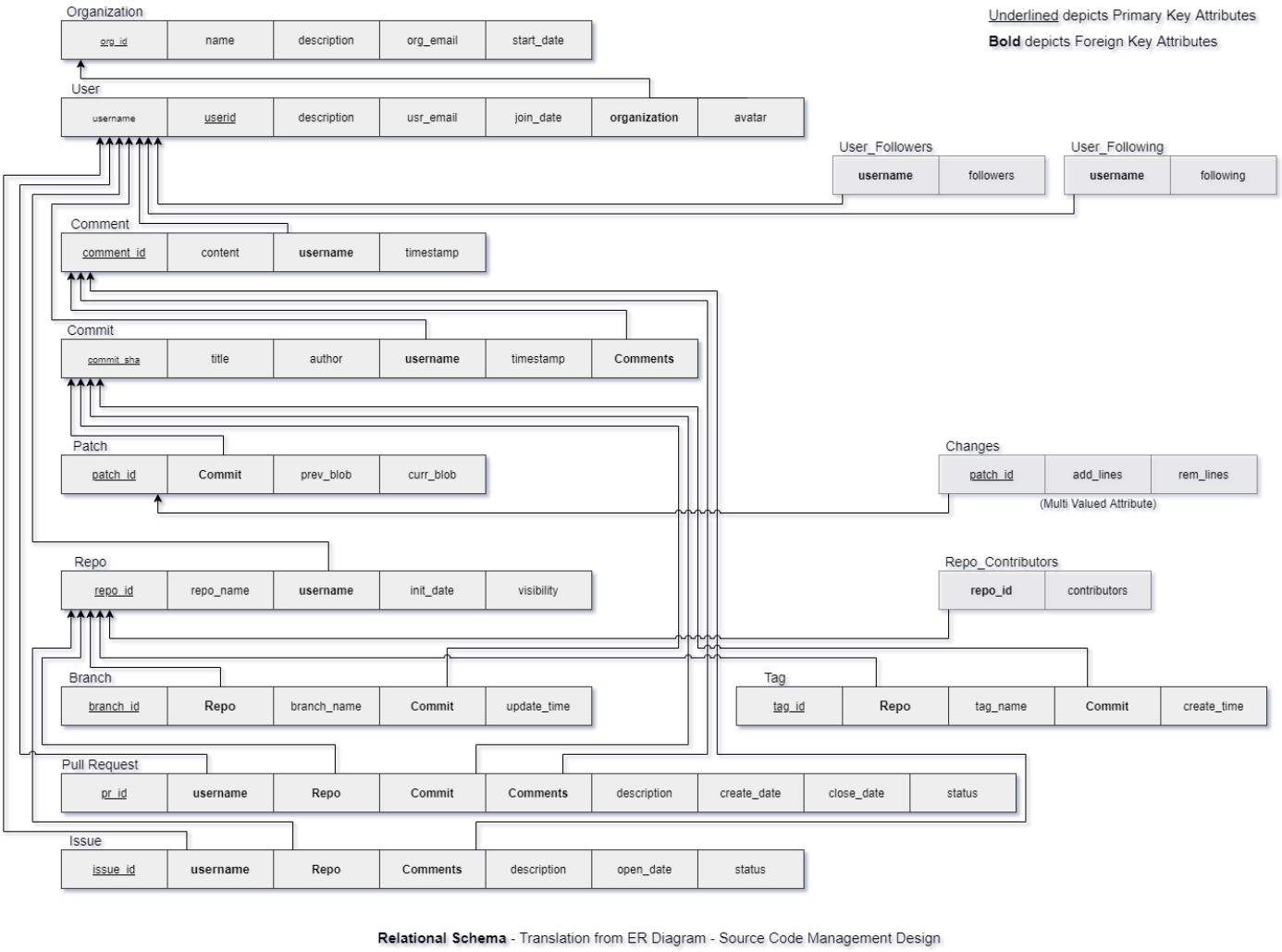

Fig 2. Relational Schema Diagram

Functional Dependency

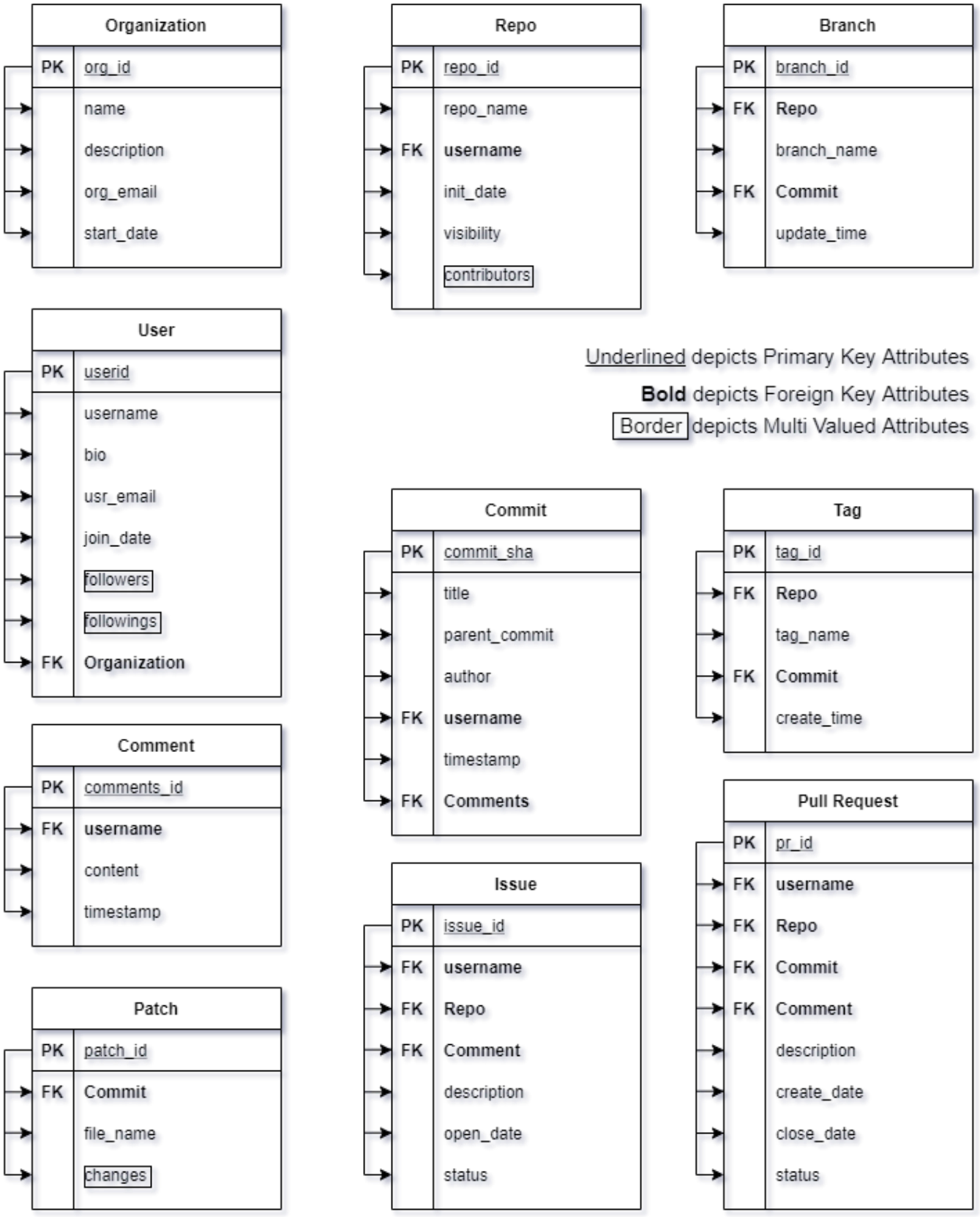

Fig 3. Functional Dependency Diagram

Normalisation

The schema defined above is compliant with Boyce Codd Normal Form. This includes satisfaction all initial 3 forms of normalisation. Additionally, for multivalued changes table required composite key of both patch_id and **commit_sha** to satisfy many-to-many relationship. Other than this all other redundancy has been removed based on the functional dependency as in the relational schema in Fig. 2 and functional dependency in Fig. 3.

Semantic Constraints

While definition of the tables – primary key constraint was set to respective attributes of the table. The parameters have defined constraints for NULL, NOT NULL and Unique values set to the respective attribute. To explain here's an example:

A user when exists – it can have an organization id or can set it to NULL if he doesn't blog. The org_id in question would be unique and must exist in the database to make user be part of one.

Additional constraints are defined as follows:

```
CONSTRAINT check_commit CHECK(LENGTH(commit_sha) = 40);
```

Boolean type is declared for checking status. So, before updating or adding changes to Table Issue or Pull Request we declare another constraint.

```
CONSTRAINT check_status CHECK(status IN(0,1));
```

Views

a) For visibility of repo we create a view for viewer with no access to private repo.

```
CREATE VIEW Public_Repo (username, repo_name, init_date) AS
SELECT username, repo_name, init_date
FROM Repo Where (visibility=0);
```

b) View recent 5 commits on a Branch

```
CREATE VIEW Recent_Commits AS
SELECT Commit.commit_sha, Commit.timestamp
FROM Commit, Branch
WHERE Commit. commit_sha = Branch.Commit
ORDER BY Commit.timestamp DESC LIMIT 5;
```

Triggers

```
CREATE TRIGGER add_ParentCommit
AFTER INSERT ON Branch FOR EACH ROW
        UPDATE Commit
        SET Commit.parent_commit = OLD.commit_sha
        WHERE Branch.Commit = NEW.Commit;
```

This trigger updates the Parent_Commit in case not defined as it can be NULL for initial commit. Now, we move onto Security parameters.

AUTO_INCREMENT

When a NULL id is passed in these parameters the trigger automatically insert value + 1 of the older available in the table. Helpful when multiple patches are being passed in the commit. Automatically this constraint is toggled.

Security

We can GRANT a role to organization admin to manage the users.

```
CREATE ROLE Org_Admin
GRANT SELECT, UPDATE ON User TO Org_ Admin
```

Additionally, we can GRANT access to user to make changes to commit. Such as reverting, removing or merging. Similarly, for controlling on Pull Requests and Issues on their respective repo.

```
CREATE ROLE User_Manager
GRANT SELECT, UPDATE, DELETE ON Commit TO User
GRANT UPDATE (status), DELETE ON Issue TO User
GRANT UPDATE (status), DELETE ON Pull_Request TO User
```

Appendix

A. Create Table

Following Queries are to Generate the Tables –

```
# Initialize a Database
Create Database GitDB;

# Start Creating Tables
CREATE TABLE Organization(
    org_id Integer not null primary key,
    org_email varchar(64) null,
    start_date datetime not null DEFAULT NOW(),
    name varchar(32) not null,
    description varchar(160)
);
```

```
Create Table User(
userid Integer primary key,
username varchar(32) not null unique,
usr_email varchar(64) not null,
join_date datetime not null DEFAULT NOW(),
organization Integer,
avatar varchar(128),
bio varchar(160),

FOREIGN KEY (oragnization) REFERENCES Organization(org_id) ON DELETE RESTRICT ON UPDATE
CASCADE
);
```

```
Create Table User_Followers(
username varchar(32) not null,
followers varchar(32) null,

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE
);

Create Table User_Following(
username varchar(32) not null,
following varchar(32) null,

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

```
Create Table Comment(
comment_id Integer primary key AUTO_INCREMENT,
username varchar(32) not null,
content varchar(512) not null,
timestamp datetime not null DEFAULT NOW(),

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT
);
```

```
Create Table Commit(
commit_sha varchar(40) primary key,
parent_commit varchar(40) DEFAULT NULL,
title varchar(128) null,
author varchar(64) not null,
username varchar(64),
timestamp datetime not null DEFAULT NOW(),
Comment Integer,

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Comment) REFERENCES Comment(comment_id) ON DELETE RESTRICT ON UPDATE
CASCADE
);

Create Table Patch(
patch_id Integer primary key AUTO_INCREMENT,
Commit varchar(40),
prev_blob varchar(255),
curr_blob varchar(255),

FOREIGN KEY (Commit) REFERENCES Commit(commit_sha) ON DELETE RESTRICT ON UPDATE CASCADE
);

Create Table Changes(
Patch Integer,
Commit varchar(40),
add_lines varchar(512),
rm_lines varchar(512),

FOREIGN KEY (Patch) REFERENCES Patch(patch_id) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Commit) REFERENCES Commit(commit_sha) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

```
Create Table Repo(
repo_id Integer primary key,
repo_name varchar(64) null,
username varchar(32) not null,
init_date datetime not null DEFAULT NOW(),
visibility bool not null DEFAULT 0, -- 0: public; 1: private

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE
);

Create Table Repo_Contributors(
repo_id Integer not null,
contributor_id varchar(32) null,

FOREIGN KEY (repo_id) REFERENCES Repo(repo_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

```
Create Table Branch(
branch_id Integer primary key AUTO_INCREMENT,
Repo Integer,
branch_name varchar(64) not null,
Commit varchar(40),
update_time datetime not null DEFAULT NOW(),

FOREIGN KEY (Repo) REFERENCES Repo(repo_id) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Commit) REFERENCES Commit(commit_sha) ON DELETE RESTRICT
);

Create Table Tag(
tag_id Integer primary key,
Repo Integer,
tag_name varchar(32) not null,
Commit varchar(40),
create_time datetime not null DEFAULT NOW(),

FOREIGN KEY (Repo) REFERENCES Repo(repo_id) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Commit) REFERENCES Commit(commit_sha) ON DELETE RESTRICT
);
```

```
Create Table Pull_Request(
pr_id Integer primary key AUTO_INCREMENT,
username varchar(30) null,
Repo Integer not null,
Commit varchar(40) not null,
Comment Integer,
description varchar(160),
create_date TIMESTAMP not null,
close_date TIMESTAMP,
status bool DEFAULT 0, -- 0: OPEN; 1: CLOSE

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Repo) REFERENCES Repo(repo_id) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Commit) REFERENCES Commit(commit_sha) ON DELETE RESTRICT ON UPDATE
CASCADE,
FOREIGN KEY (Comment) REFERENCES Comment(comment_id) ON DELETE RESTRICT ON UPDATE
CASCADE
);

Create Table Issue(
issue_id Integer not null primary key,
username varchar(30) null,
Repo Integer not null,
Comment Integer,
description varchar(160),
open_date TIMESTAMP,
status bool,

FOREIGN KEY (username) REFERENCES User(username) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Repo) REFERENCES Repo(repo_id) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Comment) REFERENCES Comment(comment_id) ON UPDATE CASCADE
);
```

```
# End Creating Tables
```

B. Populate Table

```
# Populating the Database

# Inserting into Table: Organization
INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (39131176,'team-dolan@outlook.com', '2018-05-09 15:52:47','Team-Dolan',"check
our dolan deeds!");

ALTER TABLE Organization MODIFY COLUMN org_email VARCHAR(64) NULL;

INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (18412755, NULL, '2016-04-12 04:53:30','Bashmug',"innovation is in the cup ,
hard you work brighter your idea gets");

INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (32689599, NULL, '2017-10-10 23:00:21','android', NULL);

INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (1342004, "opensource@google.com", '2012-01-18 01:30:18','google',"Google ♡
Open Source");

INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (9919, NULL, '2008-05-11 04:37:31','GitHub',"How people build software.");

INSERT INTO Organization(org_id, org_email, start_date, name, description)
Values (6615685, NULL, '2014-02-07 13:01:43','EpicGames', NULL);

DELETE FROM Organization WHERE org_id=18412755;
```

```
# Inserting into Table: User
ALTER TABLE User MODIFY COLUMN usr_email VARCHAR(64) NULL;
ALTER TABLE User RENAME COLUMN oragnization TO organization;
ALTER TABLE User ADD COLUMN bio VARCHAR(160) NULL;

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (7235275, 'rohantaneja', NULL, '2014-04-09 07:03:21', 39131176, NULL, "You have
reached 127.0.0.1 of my mind space..");

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (13597645, 'sukratkashyap', NULL, '2015-08-01 06:31:21', NULL, NULL, NULL);

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (13597646, 'torvalds', NULL, '2015-08-01 06:31:21', NULL, NULL, NULL);

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (51951179, 'dodoshrey', NULL, '2019-06-18 09:29:49', NULL, NULL, NULL);

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (32036579, 'Dyneteve', NULL, '2017-09-17 14:05:58', 6615685, NULL, NULL);

INSERT INTO User(userid , username, usr_email, join_date, organization, avatar, bio)
Values (3029663, 'danielhk', NULL, '2012-12-13 00:23:47', NULL, NULL, NULL);
```

```
# Inserting into Table: User_Followers
INSERT INTO User_Followers (username, followers) Values
('rohantaneja','sukratkashyap'),
('rohantaneja','deepansht'),
('rohantaneja','dodoshrey'),
('rohantaneja','rajatenzyme'),
('rohantaneja','lbaweja1999');
```

Inserting into Table: User_Following

```
INSERT INTO User_Following (username, following) Values
('rohantaneja','sukratkashyap'),
('rohantaneja','coolharsh55'),
('rohantaneja','lbaweja1999'),
('rohantaneja','Dyneteve'),
('rohantaneja','MadhavBahl');
```

Inserting into Table: Comment

```
INSERT INTO Comment(comment_id, username, content, timestamp) Values
(1, 'HostZero', 'PR 12: Fixing mount issue', '2020-09-09 18:03:51');
```

Inserting into Table: Commit; Patch; Changes

```
INSERT INTO Commit (commit_sha, title, author, username, timestamp, Comment)
Values ('6d63ce1d67ab9c8aea7067614492614d1256982c', NULL,'mt6582: fix mounting
partitions on few devices (#12)', 'HostZero', 'rohantaneja', '2020-09-09 18:03:51', 1);
INSERT INTO Patch Values (1, '6d63ce1d67ab9c8aea7067614492614d1256982c',
'a/rootdir/factory_init.rc', 'b/rootdir/factory_init.rc');
INSERT INTO Changes Values (1, '6d63ce1d67ab9c8aea7067614492614d1256982c','mount_all
/fstab.sprout','mount_all /fstab.mt6582');

INSERT INTO Commit (commit_sha, title, author, username, timestamp, Comment) Values
('b583e35b9ec7d718129e5dad572279f8ea3181ec ', 'update device parameters',
'rohantaneja', 'rohantaneja', '2017-09-09 18:03:51', 1),
('8723ce1d67ab9c8aea7067614492614d1256982c', NULL,'mt6582: fix mounting partitions on
few devices (#12)', 'HostZero', 'rohantaneja', '2020-09-09 18:03:51', 1),
('2d63ce1d81ab9c8aea7067614492614d1256982c', NULL,'Test Commit', 'rohantaneja',
'rohantaneja', '2018-09-09 18:03:51', 1);
INSERT INTO Patch Values (1, 'b583e35b9ec7d718129e5dad572279f8ea3181ec ',
'a/BoardConfig.mk', 'b/BoardConfig.mk');
INSERT INTO Changes Values (1, 'b583e35b9ec7d718129e5dad572279f8ea3181ec ',
NULL,'$(shell mkdir -p $(OUT)/obj/KERNEL_OBJ/usr)');

INSERT INTO Patch Values (1, '8723ce1d67ab9c8aea7067614492614d1256982c',
'a/rootdir/factory_init.rc', 'b/rootdir/factory_init.rc');
INSERT INTO Changes Values (1, '8723ce1d67ab9c8aea7067614492614d1256982c','mount_all
/fstab.sprout','mount_all /fstab.mt6582');

INSERT INTO Patch Values (DEFAULT, '2d63ce1d81ab9c8aea7067614492614d1256982c',
'a/init.rc', 'b/init.rc');
INSERT INTO Changes Values (1,
'2d63ce1d81ab9c8aea7067614492614d1256982c','wow!','how?');
```

Inserting into Table: Repo

```
INSERT INTO Repo Values
(52148045, 'android_device_mediatek_mt6582', 'rohantaneja', '2016-02-20 10:25:33', 0),
(238089109, 'College', 'rohantaneja', '2020-02-04 00:15:16', 0),
(2325298, 'linux', 'torvalds', '2011-09-04 22:48:12', 0),
(88264756, 'android_device_lenovo_aio_row', 'rohantaneja', '2017-04-14 12:12:07', 0);
```

Inserting into Table: Branch

```
INSERT INTO Branch Values
(1, 88264756, 'cm-14.1', '2d63ce1d81ab9c8aea7067614492614d1256982c', '2020-05-27
23:54:52'),
(2, 52148045, 'cm-13.0', '6d63ce1d67ab9c8aea7067614492614d1256982c', '2020-09-09
18:03:51'),
(3, 238089109, 'master', '8723ce1d67ab9c8aea7067614492614d1256982c', '2020-05-27
23:54:52'),
(4, 2325298, 'master', '6bff9bb8a292668e7da3e740394b061e5201f683', '2020-12-12
21:02:42');
```

```
# Inserting into Table: Tag
INSERT INTO Tag Values
(1, 88264756, 'release', '2d63ce1d81ab9c8aea7067614492614d1256982c', '2020-05-27
23:54:52'),
(2, 52148045, 'cm-13.0', '6d63ce1d67ab9c8aea7067614492614d1256982c', '2020-09-09
18:03:51'),
(3, 2325298, 'v5.10-rc7', '0477e92881850d44910a7e94fc2c46f96faa131f', '2020-12-12
21:02:42');
```

```
# Inserting into Table: Pull_Request
INSERT INTO Pull_Request Values
(1, 'rohantaneja', 2325298, '2d63ce1d81ab9c8aea7067614492614d1256982c', 1, 'Fix for
packets being rejected.', '2020-03-02 12:16:22', '2020-09-28 12:42:31', 0),
(2, 'rohantaneja', 52148045, '2d63ce1d81ab9c8aea7067614492614d1256982c', 1, 'Update
REPO.', '2017-03-02 12:16:22', '2018-09-28 12:42:31', 0);
```

```
# Inserting into Table: Issue
INSERT INTO Issue Values
(1, 'rohantaneja', 2325298, 1, 'Help with issue.', '2020-03-02 12:16:22', 0),
(2, 'rohantaneja', 88264756, 1, 'Help with another issue.', '2020-08-02 12:16:22', 0);
```