DBMS ASSIGNMENT 1
5TH SEM SECTION I

PES1UG19CS579 VISHWAS R

PES1UG19CS548 UTHPAL P

PES1UG19CS534 T R SUDHARSHAN

PROJECT TITILE

ITHUB

IT

Problem statement:

To implement how FILES , VERSIONS , BRANCHES , REPOSITORY , PROJECTS , DEVELOPER DETAILS are stored in database.

GIT works in local machines and contains local repositories and versions.
GITHUB works in one server and contains remote repository.

Our DBMS can store details of company and their developers. Company assigns developers a project.

Developer create a project and project entity contains group of repository (one remote and many local repositories)

Each member will have a local repository.

Repositories have versions of files .

ENTITIES

- Company
- Developers
- Project
- Repository
- Branch
- Version
- Files

Operations allowed:

- Merge two branches: This operation is allowed only for branches which are “ancestor” related. If the 2 branches are parent-child related, the latest version of the 2 branches are chosen, then unified. If there are 2 files of the same name, the larger file is chosen and the smaller file is discarded. For grandfather-child relation and more, we shall have a recursive merge. The newly created file belongs to the “older” branch.

- Push: Because of the fork policy, the branches in the remote repo is the union of all branches in all local repos within a project. We have to perform the “union” of 2 branches.

- Pull: The operations are very similar to the push op but branches which are empty and not present in the target repo are not added to the target repo.

- Fork: When a branch is requested to be created in a local repo, the same branch is also created in the remote repo with common specs (branch id, timestamp).This operation can also be called as a light-weight push.

- Commit: Create new version within a branch.

- Drop repository: Deletes a local repository (not allowed to delete a remote repository).

Detailed information about our projectThere are users/ developers working on projects.

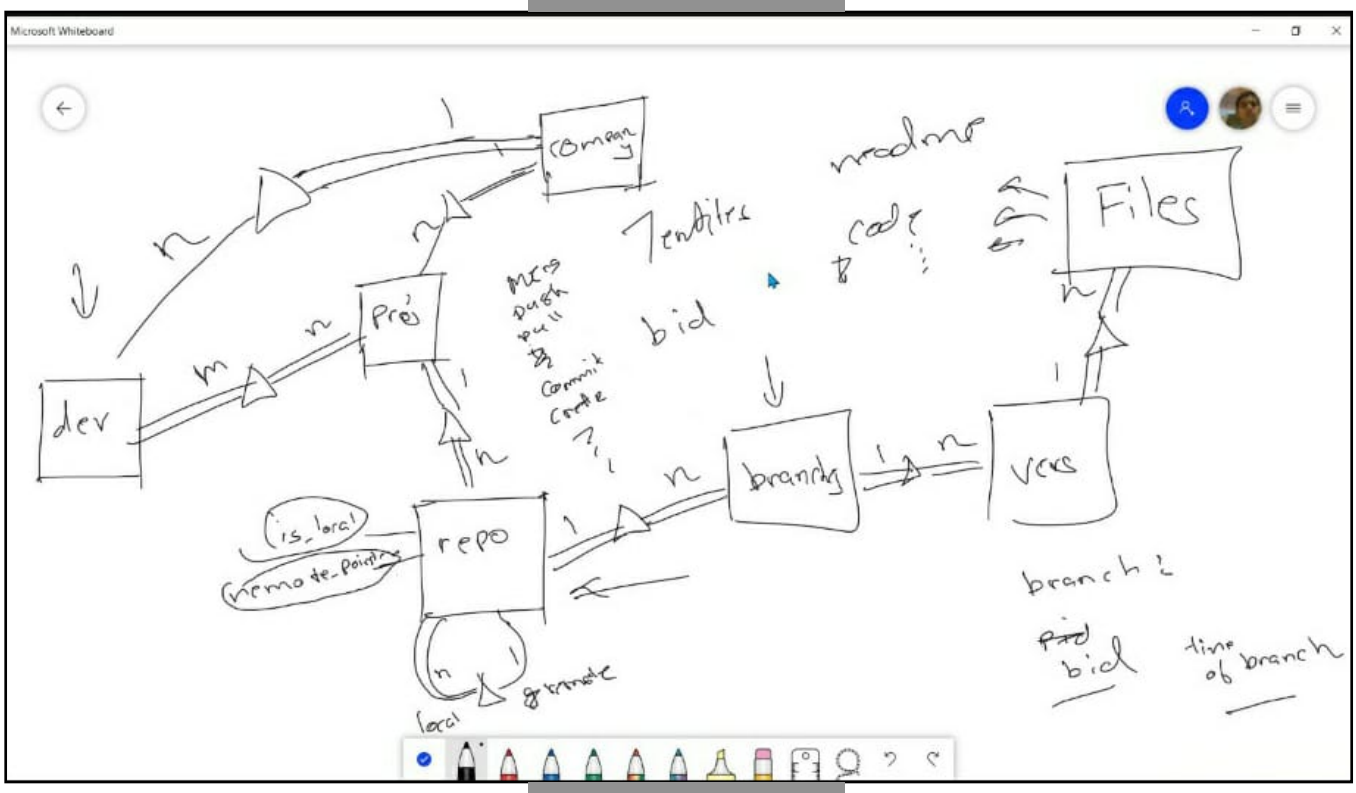
- Every user has their own username/userID(unique), password.
- A company can employ multiple developers.
- A company has company_id.
- A company can handle multiple projects.
- A repository can be local or remote.
- A local repository is personal to only one user.
- A project is implemented as a set of repositories consisting of one remote repository and multiple local repositories based on the number of users.
- A project is a weak entity identified by remote repo id.
- A project contains description, progress bar(percentage of completion)
- Every repository has a unique id(rid).

- A remote repository is shared between users working on the same project.
- A repository is a tree which consists of multiple branches of which there is only 1 master branch which is the ancestor of all other branches of the same tree.
- A repository if local has user_id(identifying the user who works on the repo). If remote, the user_id holds NULL.
- Every repository must have a master branch.

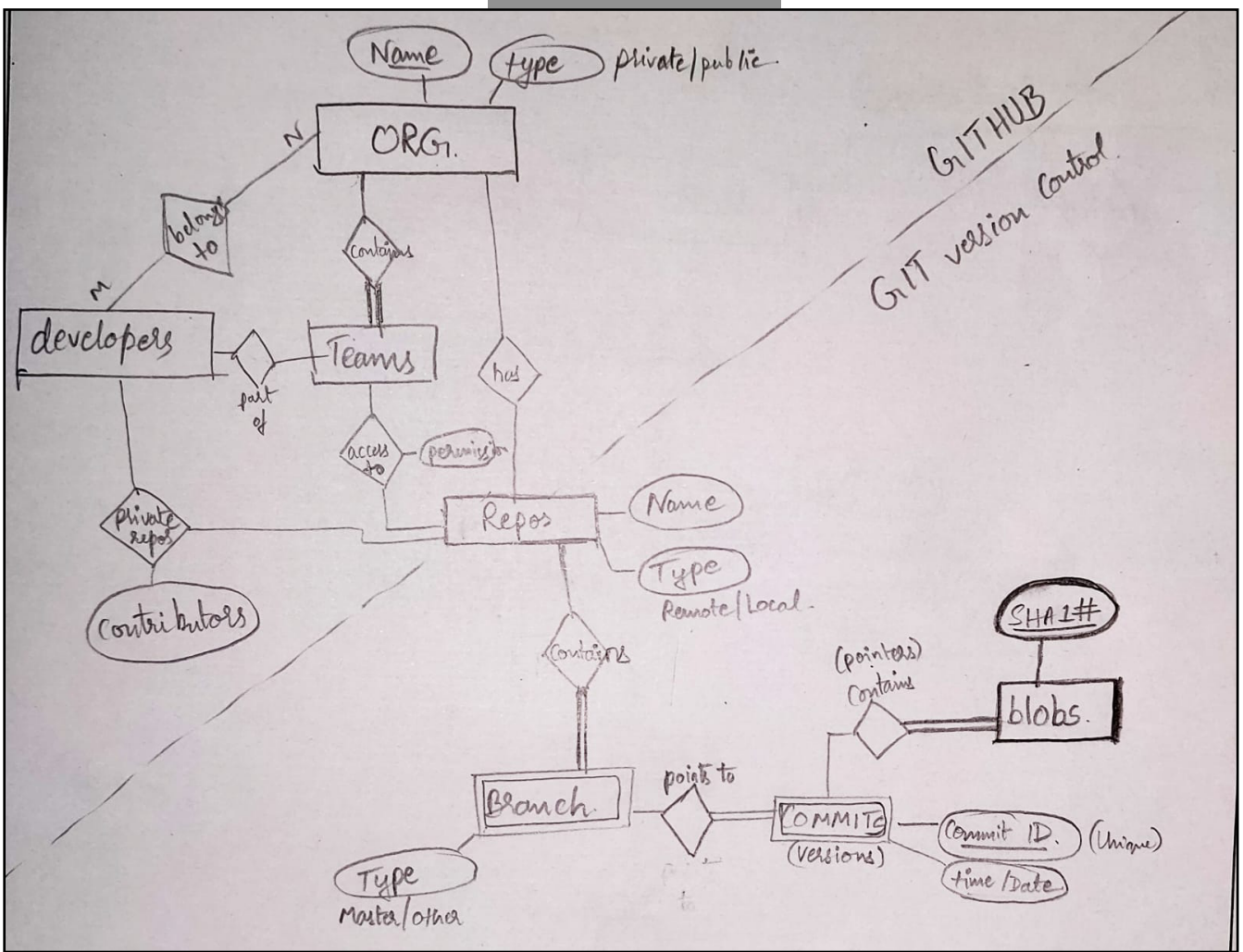
- Every branch has a parent branch except the master branch(this can be used to identify the master branch).
- Every branch has a branch id which is unique within the group of repositories(1 remote,n local) working towards a single project.
- Every branch has latest_version_id(signifying the latest version).
- Every has a Branch Created by (B_user_ID) which tells which user created the branch.
- Every branch also has timestamp
- The branch id is also used for telling whether 2 branches are same or not in different repos.
- A branch belongs to only one repository.
- Every branch has repo_id(the repository it belongs to).
- Every branch also has it's parent branch id(pbid), it also has child_num(used to distinguish b/w multiple children branches of the same parent).
- Every branch has multiple versions.
- A version has to belong to 1 branch.
- A version has created_by(user_id) which tells which user created the version.
- A version has version id(vid, not unique, which is useful for differentiating the "evolution" of versions in the same branch).
- A version is a weak entity.
- A version has branch id(the branch it belongs to).
- A version has multiple files of different types(.md,.doc,.c,.ipynb etc)
- A file belongs can belong to multiple versions (as to create a new version, we need to edit only 1 file while not touching others).
- A file has it's own unique id(fid).
- A file also has it's on name(it must be unique in the versions it belongs to).

Evolution of our ER diagram

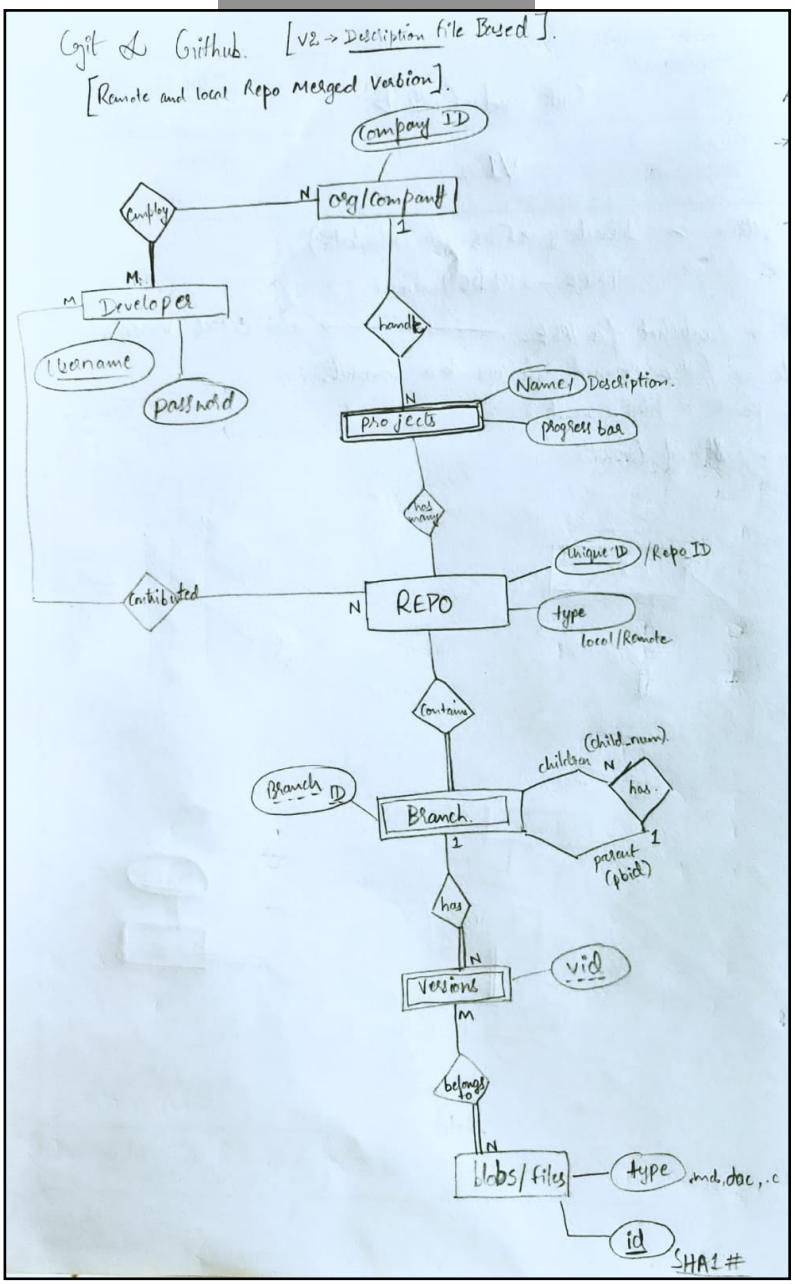
Version 1



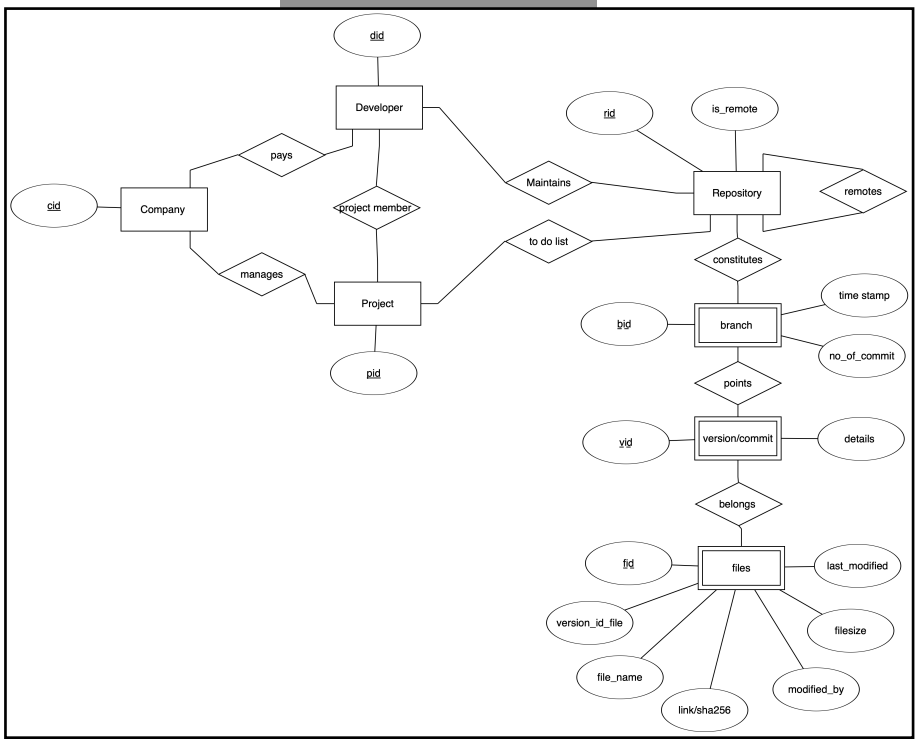
Version 2



Version 3

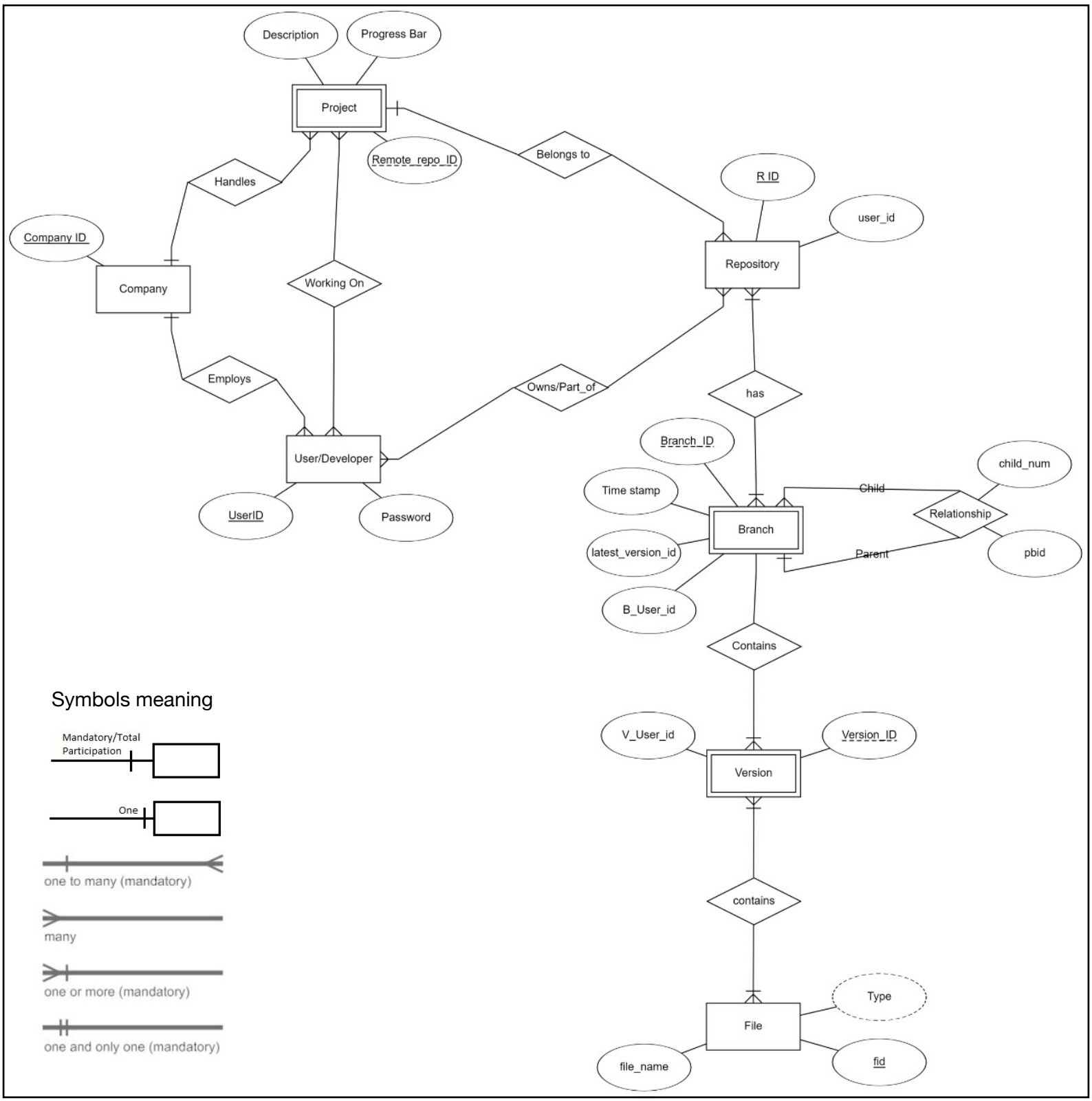


Version 4



ER diagram

Final Version



ER tool used :

Link - https://erdplus.com

Features -

- ERD Plus is a basic database modelling tool for creating Entity Relationship Diagrams, Relational Schemas, Star Schemas, and SQL DDL statements.
- Easy to use due to drag and drop style GUI
- It's Free and saves diagrams safely on their server
- Automatically converts ER Diagrams into Relational Schemas thus helping us in better understanding of our Model.
- Can export diagrams as a PNG and can be imported into another system.
- Its a website so no installation problem