



Bonds not included in the diagram:

1. member creates actions,
2. authorityid is also unique with respect to other ids.

init user privileges:

1. create leaders with passwords,
2. connect to the database.

app user privileges:

1. adding actions as particular members,
2. voting as members,
3. listing actions as leader member,
4. listing projects as leader member,
5. listing votes as leader member,
6. listing trolls.

Implementation of API functions draft:

1. open(database, login, password) - initialize connection to the database,
2. leader(password, member) - create or update proper row in Member table,
3. support(timestamp, member, password, action, project, [authority]) - add a proper row to Action
4. protest - virtually same as above
5. upvote(timestamp, member, password, action) - create row in Vote and update in Action (votes for)
6. downvote - almost the same as above
7. actions(timestamp, member, password, [type], [project / authority]) - proper select from Action
8. projects(timestamp, member, password, [authority]) - proper select from Project
9. votes(timestamp, member, password, [action / project]) - proper select from Vote
10. trolls() - proper select from Member (of course with some join with Vote and Action)

Powered By: Visual Paradigm Community Edition

Uruchamianie programu (przy zgodnym ze specyfikacją założeniu, że baza danych i użytkownik init już istnieją i pgcrypto jest dostępne):

```
python3 project.py --init
```

<teraz trzeba podać ciąg wywołań API z stdin>

```
python3 project.py
```

<teraz ciąg wywołań API z logowaniem jako app>