Logical Concept PolSim

Amin Haidar, Noel Monnerat,
Samuel Pearce, Alban Selimi

Table of Contents:

1. Table Structure

Table 1. Group

Key	Field	Datatype	Options
PK	ID_group	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
	groupname	VARCHAR(50)	NOT NULL, UNIQUE
FK	leader_person_ID	INT	NOT NULL
	power	INT	NOT NULL
FK	parent_group_ID	INT	NOT NULL

Table 2. Person

Key	Field	Datatype	Options
PK	ID_person	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
	username	VARCHAR(50)	NOT NULL, UNIQUE
	passwordhash	VARCHAR(255)	NOT NULL
FK	member_group_ID	INT	NOT NULL
	power	INT	NOT NULL

Table 3. Vote

Key	Field	Datatype	Options
PK	ID_vote	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
FK	voter_person_ID	INT	
FK	voter_group_ID	INT	
FK	applied_decision_ID	INT	NOT NULL
	agreement	BOOLEAN	NOT NULL

Table 4. Decision

Key	Field	Datatype	Options
PK	ID_decision	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
	title	VARCHAR(100)	NOT NULL
FK	creator_user_ID	INT	NOT NULL
	close_time	TIMESTAMP	NOT NULL
FK	voting_group_ID	INT	NOT NULL
	creation_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP()
	last_vote	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP()

Table 5. Action

Key	Field	Datatype	Options
PK	ID_action	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
FK	approval_decision_ID	INT	NOT NULL
	description	VARCHAR(255)	NOT NULL
	exec_time	TIMESTAMP	NOT NULL

Table 6. Command

Key	Field	Datatype	Options
PK	ID_command	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
FK	executing_command_ID	INT	NOT NULL
	sql_command	VARCHAR(255)	NOT NULL

Table 7. Argument

Key	Field	Datatype	Options
PK	ID_argument	INT	NOT NULL, UNIQUE, AUTO_INCREMENT
FK	applying_command_ID	INT	NOT NULL
	value	INT	NOT NULL
	position	INT	NOT NULL

2. Indices

There were two columns we decided to put indices on in order to be able to reference them more quickly. Namely the username column and the groupname column. We wanted these names to be able to be used as indexes and therefore had to be unique. This meant if a user wants to log in, they will send their username and not the user ID. This is why it is important that the column be fast when fetching data.

The group column was a similar story to the username column. Group names need to be unique, so they can be used in the user interface. So we also made an index for the group column.

3. Constraints

All the foreign keys listed in the above tables reference one another in the way described previously and shown in the ERM and ERD. Most of the columns are limited to not being NULL so that we can guarantee some of the answers will be at least set. There are only a few columns allowed to be NULL. These are voter_person_ID and voter_group_ID in the vote table. These columns need to be flexible so that both groups and people can cast votes and have them attributed correctly. At least one must be set though when a row is inserted.

4. Triggers

Trigger 1: `trg_last_vote`
For statistics and as a source of more information, the decision table has a column called "last vote" which stores a timestamp of the time the last new vote was

cast. This will be triggered by an insertion into the vote table. If the insertion is a success, then the last_vote column is updated to the current time.

Trigger 2: `trg_unowned_vote`
As mentioned previously, there is the possibility, that a vote could be inserted, that has no owner, and can therefore not be validated. These votes should automatically be removed. This will be triggered by an insertion into the vote table. If the new vote has neither a person owner, nor a group owner, it is deleted from the table.

5. Stored Procedures

Stored Procedure 1 `stp_compile_command(INT command_ID)`
This procedure will return a command's SQL statement with all the values filled in from the argument table. This means that most of the work interpreting the actions is done by the database itself. This saves time later on with the program and ensures efficiency.

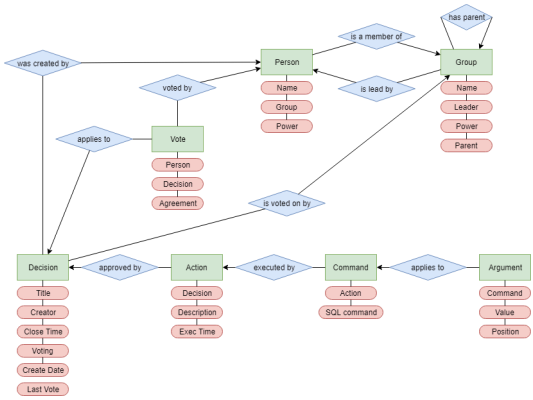
Stored Procedure 2 `stp_even_power(INT group_ID)`
When a new group is created, a revolution is succesful or a new member is added, all the political power is shared equally amongst the group's members. This procedure automates this process of equalizing the power. It takes the total population of the current group and divides 100 power points to each member group or person.

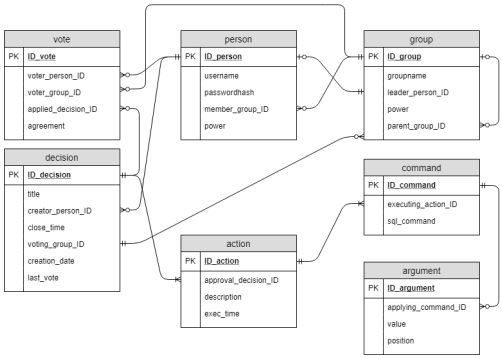
6. Views

View 1 `v_user_groups`
This view makes it easier to get an overview of the people's connections. Each person's username is listed out beside the name of the group they're in.

View 2 `v_decision_results`
This view was supposed to display the number of political power points of support have been given in favour of or against any decision in total. Although, as of writing this it doesn't appear to be working correctly as on of the internal commands can return a NULL, which propagates forward, ruining the result numbers.

7. ERM / ERD





8. Naming Convention

Table Names: `<tblname>, lowercase`
Primary Key: `ID_<tblname>, lowercase`
Foreign Key: `<name>_<reftable>_ID, lowercase`
Column Names: `<colname>, lowercase`
View Names: `v_<name>, lowercase`
Index Names: `idx_<name>, lowercase`
Trigger Names: `trg_<name>, lowercase`
Stored Procedure Names: `stp_<name>, lowercase`