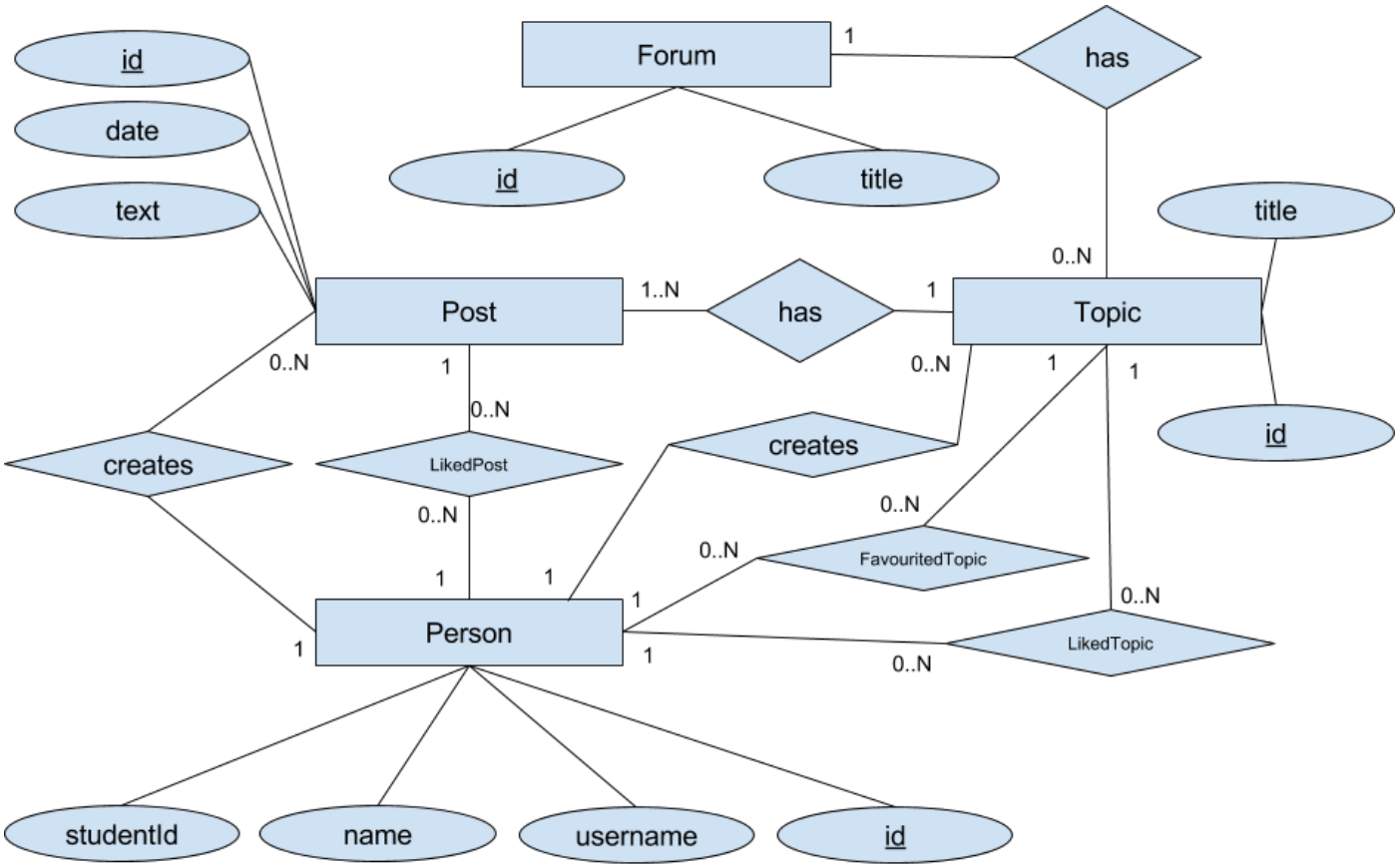
NAMES & STUDENTID

Alexander Lorimer: al15273
James Birch: jb15339
Phan Nguyen: pn15644

ENTITY RELATIONSHIP DIAGRAM



Person		
id	INTEGER	PRIMARY KEY
username	TEXT	UNIQUE, NN
name	TEXT	NN
studentId	TEXT	

Topic		
id	INTEGER	PRIMARY KEY
title	TEXT	NN
postId	INTEGER	FK

Forum		
id	INTEGER	PRIMARY KEY
title	TEXT	UNIQUE, NN
topicId	INTEGER	FK

NOTE: the Topic-has-Post relationship is not N-N, so I think we need to delete topicId here and just keep Topic's PostId..?

Post		
id	INTEGER	PRIMARY KEY
date	DATE	NN
text	TEXT	NN
personId	INTEGER	FK
topicId	INTEGER	FK

LikedPost		
PostId	INTEGER	FK
PersonId	INTEGER	FK

LikedTopic		
TopicId	INTEGER	FK
PersonId	INTEGER	FK

FavouritedTopic		
TopicId	INTEGER	FK
PersonId	INTEGER	FK

DESIGN CHOICES

/* What I've done here is to just specify which API requirement drove each design choice as regards entities and attributes, but I haven't made any mention of the use-case scenarios that necessitating the existence of each attribute, so it's a bit tautological. I've also not commented on the multiplicity of the relationships. - Jamie */

We identified the main entities as described in the API to be Forum, Topic, Post, and Person. Each of these has an integer primary key, ‘id’, which the foreign keys of other entities may refer to. Other attributes and relationships are justified as follows:

Forum
- Unique, non-null **title** as specified in APIProvider’s createForum method’s documentation.

Topic
- Non-null **title**, as specified in APIProvider’s createTopic method’s documentation.

// just noticed that our capital letters standardisation isn’t adhered to in certain attributes within entities like Post.

Post
- Non-null **date**, so that APIProvider’s getLatestPost method has a date by which to sort Posts of a Topic.
- Non-null **text**, as required by APIProvider’s createPost method’s documentation.
- A foreign key to the **id of Person**; every Post may be liked by 0..N Persons, as specified in APIProvider’s likePost method’s documentation.
- A foreign key to the id of Topic; every Post // see my comment on the tables.

// I’m reconsidering whether to mention all methods that require a username.
Person
- Unique, non-null **username**; mandatory for the addNewPerson method. to be displayed as the person liking a post or topic (likePost, likeTopic), favouriting a topic (favouriteTopic) to search the userbase by (getUsers, getPersonView and getAdvancedPersonView), and to be displayed upon creating a post or topic (createPost, createTopic).
- A **studentId**; may be null (some users aren’t students), and thus needn’t be unique. // note: needs to be ‘non-empty’ nonetheless.
- A non-null **name**; as required by APIProvider’s addNewPerson method’s documentation.

There are additionally some N-N relationships between these entities: Each Person may favourite and/or like several Topics, while each Topic may be favourited and/or liked by several Persons; and each Person may like several Posts while each Post may be liked by several Persons. These relationships necessitated three join tables:

LikedPost, LikedTopic, and FavouritedTopic, which each consist only of foreign keys to the two entities they join together.

/*
Questions to answer:
Why we chose to normalise certain elements or not…
How this makes the api easier or harder to implement &or to keep the database consistent...
*/