Assignment - 2

Goal: create an ontology or semantic model of a domain of their choice. (Consider the same domain from Assignment 1 so that data can be reused.) They need to develop the semantic model using the description logic SROIQ.

We have selected Social Media Networking Domain in Assignment 1.

Here is an ontology or semantic model of the social networking website domain using the description logic SROIQ:

Important pieces of knowledge from the domain that are captured in the ontology :

User : Represents a registered member of the social networking website. Has a unique ID, a username, a password, an email, a first name, a last name, a gender, a birthdate, and a profile picture.

Profile : Contains information about a user's profile, including their location, education, and occupation.

Bio : A textual description of a user's profile.

Friends : A set of users who are friends with a given user.

Post : A piece of content created by a user on the website. Contains text, an optional image, an optional video, and a set of comments.

Text : The textual content of a post.

Image : An image file associated with a post. Can have an optional caption.

Video : A video file associated with a post. Can have an optional caption.

Comments : A set of comments on a post.

Comment : A textual comment on a post. Has a unique ID and a timestamp.

Like : represents a like made by a user on a post

DL ontology (TBox) :

ObjectProperties : hasProfile, hasFriend, hasPost, hasComment, hasImage, hasVideo, likedBy, memberOf

DataProperties : hasUsername, hasPassword, hasEmail, hasFirstName, hasLastName hasGender, hasBirthdate, hasProfilePicture, hasBio, hasLocation hasEducation, hasOccupation, hasCaption, hasText, timestamp privacySetting

Classes : User, Profile, FriendList, Post, TextPost, ImagePost, VideoPost, Comment, Like Group

Axioms:

User ⊑ ∃hasProfile.Profile

User ⊑ ∃hasFriend.FriendList

User ⊑ ∃hasPost.Post

Post ⊑ ∃hasText.TextPost

Post ⊑ ∃hasImage.ImagePost

Post ⊑ ∃hasVideo.VideoPost

Post ⊑ ∃hasComment.Comment

Post ⊑ ∃timestamp.xsd:dateTime

TextPost ⊑ ∃hasText.xsd:string

ImagePost ⊑ ∃hasImage.xsd:anyURI

ImagePost ⊑ ∃hasCaption.xsd:string

VideoPost ⊑ ∃hasVideo.xsd:anyURI

VideoPost ⊑ ∃hasCaption.xsd:string

Comment ⊑ ∃hasText.xsd:string

Comment ⊑ ∃timestamp.xsd:dateTime

Like ⊑ ∃likedBy.User

Like ⊑ (∃timestamp.xsd:dateTime) ⊓ (∃privacySetting.xsd:string)

Group ⊑ ∀memberOf.User

Design choices and explanation :

The ontology is designed using the Description Logic SROIQ. It is based on the given XML schema of the social networking website domain. The ontology is structured into several classes, properties, and axioms.

Classes

- SocialNetworkingWebsite: This is the main class representing the entire social networking website domain.
- User: This class represents a user of the social networking website. It has several attributes such as id, username, password, email, first_name, last_name, gender, birthdate, and profile_picture.
- Profile: This class represents a user's profile. It has several attributes such as location, education, and occupation.
- Bio: This class represents the user's bio, which is a short description of the user's background or interests.
- Friends: This class represents the list of a user's friends.
- Posts: This class represents a list of posts made by the user.
- Post: This class represents a single post made by the user. It has several attributes such as id and timestamp.
- Text: This class represents the text content of a post.
- Image: This class represents the image content of a post. It has an optional caption attribute.
- Video: This class represents the video content of a post. It has an optional caption attribute.
- Comments: This class represents the list of comments on a post.
- Comment: This class represents a single comment on a post. It has several attributes such as id and timestamp.

- Like: represents a like of a post on the social media network
- Group: represents a group of persons on the social media network

Properties

- hasProfile: This object property connects a user to their profile.
- hasBio: This object property connects a profile to the user's bio.
- hasFriends: This object property connects a user to their friends.
- hasPosts: This object property connects a user to their posts.
- hasText: This object property connects a post to its text content.
- hasImage: This object property connects a post to its image content.
- hasVideo: This object property connects a post to its video content.
- hasComments: This object property connects a post to its comments.
- hasComment: This object property connects a comment to its parent post.
- likedBy: relates a Like to the Person who made it
- timestamp: assigns a timestamp to a Post, Comment, or Like
- privacySetting: assigns a privacy setting to a Profile, Post
- memberOf: relates a Person to the Group they belong to

Axioms

- User and Profile are disjoint classes. This axiom ensures that a user cannot also be a profile, and vice versa.
- Bio and String are disjoint classes. This axiom ensures that a bio can only be a string.
- Friends and User are disjoint classes. This axiom ensures that a user cannot also be their own friend.

- Posts and User are disjoint classes. This axiom ensures that a user cannot also be a post.

- Post and Comments are disjoint classes. This axiom ensures that a post cannot also be a comment.

- Comment and String are disjoint classes. This axiom ensures that a comment can only be a string.

- hasProfile, hasBio, hasFriends, hasPosts, hasText, hasImage, hasVideo, hasComments, and hasComment are all object properties.

- hasProfile, hasBio, hasFriends, hasPosts, hasText, hasImage, hasVideo, and hasComments are functional object properties. This axiom ensures that each user has only one profile, one bio, one list of friends, one list of posts, one text content per post, one image content per post, one video content per post, and one list of comments per post.

Motivating Examples

- To find all the users who have a profile in the social networking website, we can use the following DL query:

  hasProfile some Profile

  This query returns all users who have at least one profile.

- To find all the posts that have an image witha caption containing the word "beach", we can use the following DL query:

  hasImage some (hasCaption some "beach") and rdf:type Post

  This query returns all posts that have an image with a caption containing the word "beach".