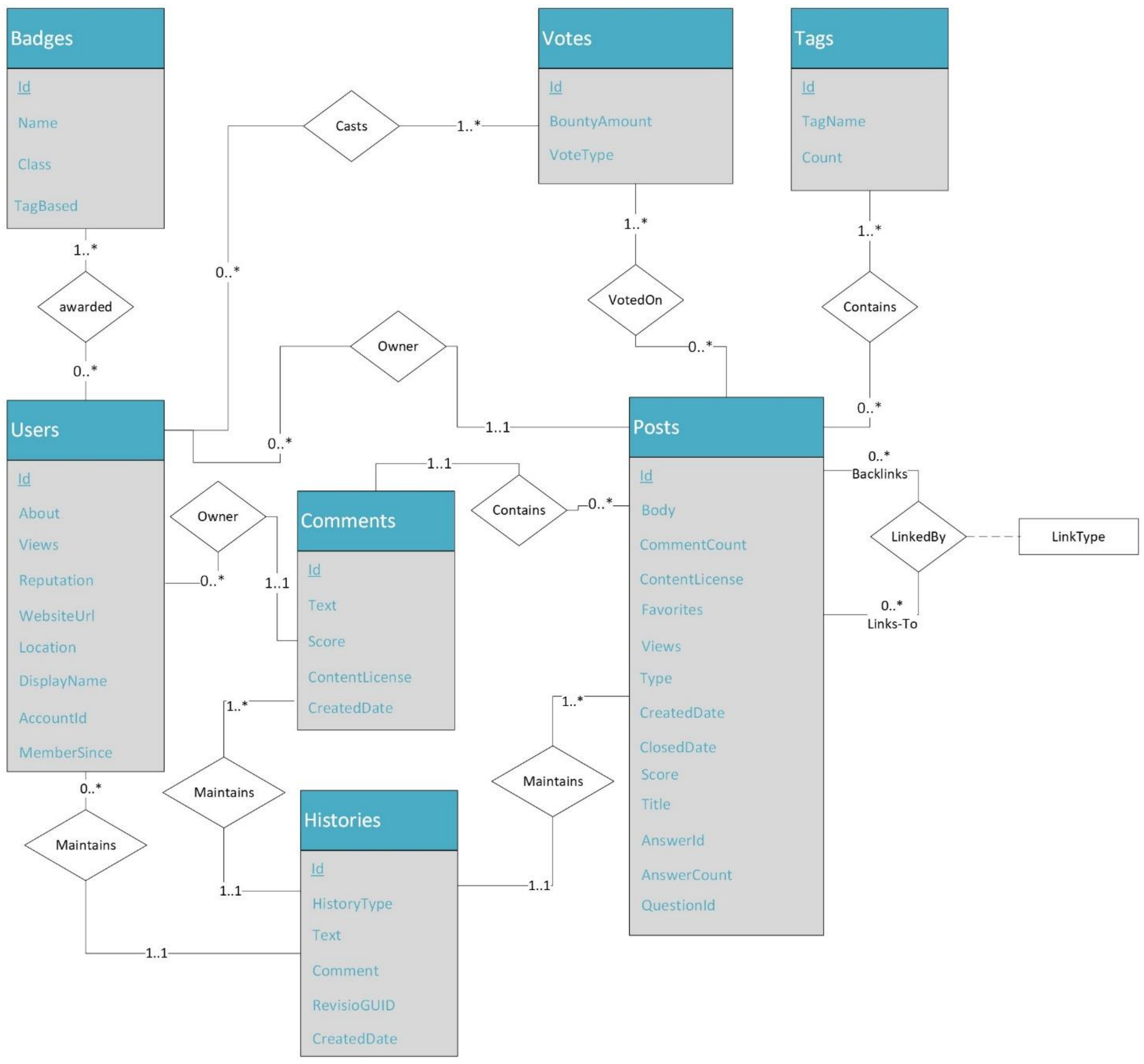
Report – Assignment 1 on Relational model

This report explains the process of modeling Ask Ubuntu's dataset from Stack Exchange. The datasets are available for download. The explanations in this report include dataset exploration, ER model, schema, data insertion, and managing transactions. The implementation for this assignment is carried out in PostgreSQL.

Question 1: Provide an ER diagram to represent the Stack Exchange data.

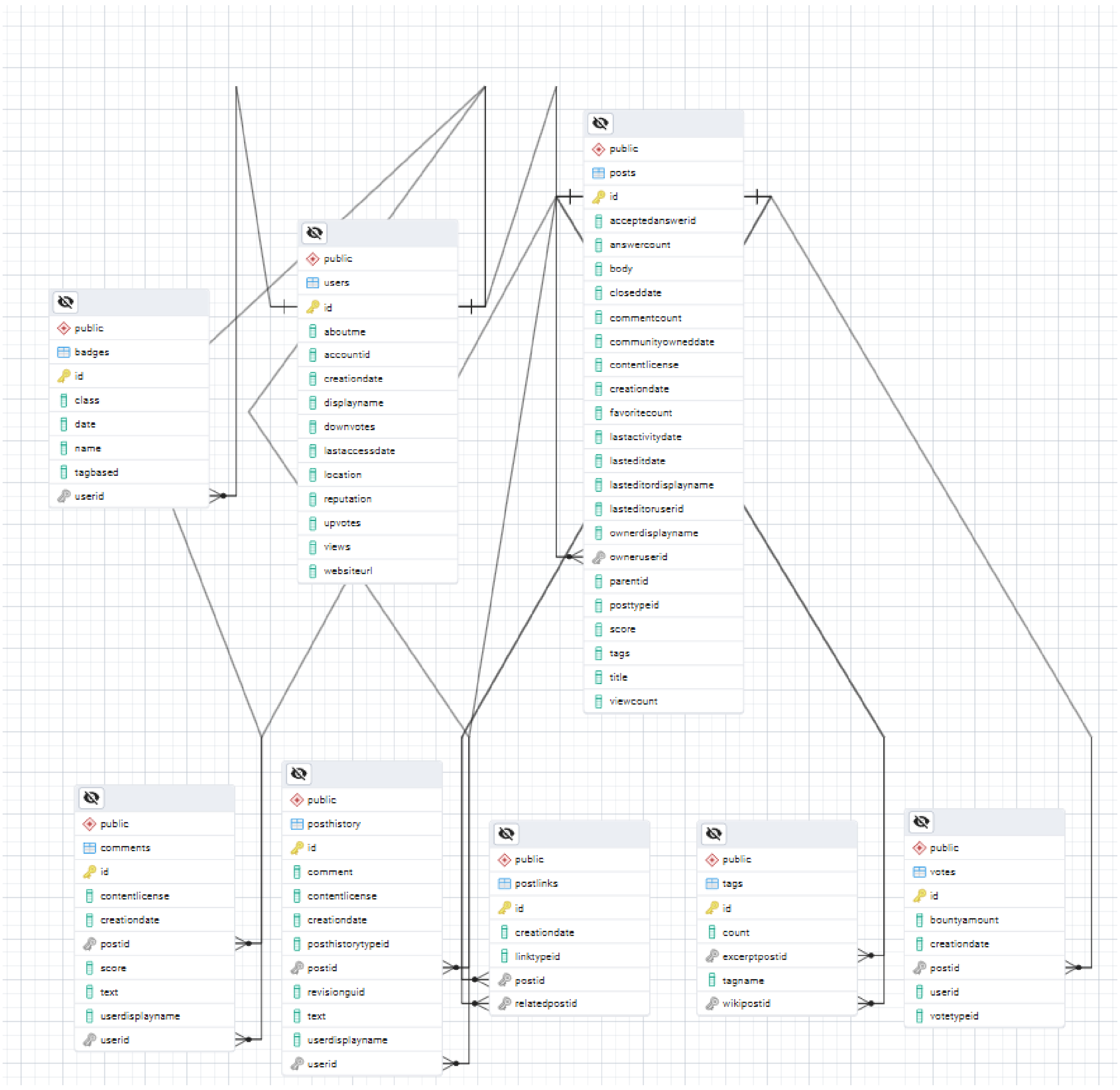


Question 2: Create a relational model to store Stack Exchange information based on your ER diagram.

The relational model is created based on the ER model and provided dataset. The SQL script to create all tables in the dataset including primary and foreign keys is provided in the "create_schema.sql" (in the root directory). The script is provided as an independent SQL file and can executed alongside the main code (from "app.py" in the root directory).

I explored the dataset to decide about the sizes of the attributes, appropriate data types, enforcement of null conditions, and relationship constraints. I evaluated each attribute for maximum length, and largest (or longest) value to discern the appropriate datatypes. The common types include Integer, Small Integer, Timestamp, Varchar, Text, and Boolean. I used the Small Integer (SMALLINT) for the attributes where the largest value was not higher than a few thousand. For other numeric values, the Integer data type was sufficient. For textual attributes, I used the TEXT type if it exceeded a few hundred characters (by length), otherwise, I used Varchar appropriate to the length of each attribute. For "not null" enforcement, I evaluated individual attributes for whether it is a foreign key or a key with no null values in the dataset. Foreign key constraints were not imposed on the attributes where there were higher null values. This a design decision for some cases where there is a large number of foreign key attributes that have null values. As shown in Table 1 below, I used this scheme to explain null enforcement (Auto=enforced by default, yes=enforced, empty=not enforced). Examples of unenforced foreign key constraints include AcceptedAnswerId (Posts), LastEditorUserId (Posts), and UserId (Votes) due to a very high null-to-total record ratio.

The actual relational model looks like this;



Question 3: Describe the contents and purpose of the files in the Stack Exchange dataset.

The dataset consists of a total of 8 files provided. These files are “Users.xml”, “Badges.xml”, “Posts.xml”, “PostLinks.xml”, “Comments.xml”, “PostHistory.xml”, “Votes.xml”, “and Tags.xml”. The name of each file is self-explanatory to discern the data it contains. A detailed description of the contents, total records, their data types, and completeness is provided in Table 1 below. Here’s a short description of each file in the dataset:

1. **Users.xml:** It contains information about users such as display names, reputations, views, votes, and other basic content.

2. **Badges.xml:** It contains information about the badges (class, tag, etc.) awarded to associated users for various achievements.
3. **Posts.xml:** It contains post information, type of post (post, question, link, answer, etc.), views, scores, and relevant users' information. It also includes additional information such as answer counts, favorites, tags, and associated activities on a post.
4. **PostLinks.xml:** It contains post links; for instance the link to an associated post in the posts or whether a post is a duplicate, a related question, or has some other connection.
5. **Comments.xml:** It contains information about comments made by users on specific posts.
6. **PostHistory.xml:** It tracks the post history over time, for instance, what type of edits and who (users) have made edits to the posts.
7. **Votes.xml:** It contains information on voting activity by users related to posts.
8. **Tags.xml:** It just stores the tags and where those are used/originated. Tags are keywords or labels that help categorize posts.

Table 1. Description of data files and contents for total records, attributes, sample values, probable data types, null and unique counts. Based on the null counts, I decided not to enforce "not null" where a majority of values were null for an attribute.

Sr.	Attribute	Sample Value (long)	Data Type	Null Count	Unique Count	Not Null?
		Users (Total Records: 1450497)				
1.	Id	1000000	Integer	0	50+	Auto (PK)
2.	AboutMe	Long text (7126)	Text	966039	50+	
3.	AccountId	10000119	Integer	48	50+	
4.	CreationDate	2010-07-28T16:38:27.683	Timestamp	0	50+	Yes
5.	DisplayName	Belle Oasis Hydrating Cream	Varchar (50)	0	50+	Yes
6.	DownVotes	219459	Integer	0	50+	Yes
7.	LastAccessDate	2010-07-28T16:38:27.683	Timestamp	0	50+	Yes
8.	Location	Text (100)	Varchar (110)	1112729	50+	
9.	Reputation	291283	Integer	0	50+	Yes
10.	UpVotes	25482	Integer	0	50+	Yes
11.	Views	16456	Integer	0	50+	Yes
12.	WebsiteUrl	Text (200)	Varchar (210)	1039502	50+	
		Badges (Total Records: 1977518)				
1.	Id	1000000	Integer	0	50+	Auto
2.	Class	3	Small Integer	0	3	Yes
3.	Date	2010-07-28T19:09:00.243	Timestamp	0	50+	Yes
4.	Name	windows-subsystem-for-linux	Varchar (30)	0	50+	Yes
5.	TagBased	False	Boolean	0	2	Yes
6.	UserId	1000006	Integer	0	50+	Yes (FK)
		Posts (Total Records: 939104)				
1.	Id	1000000	Integer	0	50+	Auto (PK)
2.	AcceptedAnswerId	1320018	Integer	803789	50+	
3.	AnswerCount	20	Small Integer	526871	44	
4.	Body	Long Text (38146)	Text	1667	50+	
5.	ClosedDate	2013-03-14T16:21:49.323	Timestamp	890344	50+	
6.	CommentCount	14	Small Integer	0	50+	Yes
7.	CommunityOwnedDate	2010-07-28T19:34:40.093	Timestamp	934804	50+	
8.	ContentLicense	CC BY-SA 3.0	Varchar (20)	0	3	Yes
9.	CreationDate	2010-07-28T19:04:21.300	Timestamp	0	50+	Yes
10.	FavoriteCount	1	Small Integer	927948	2	
11.	LastActivityDate	2018-10-05T23:56:48.997	Timestamp	0	50+	Yes
12.	LastEditDate	2014-12-16T01:47:45.980	Timestamp	481343	50+	
13.	LastEditorDisplayName	Dario Delpiano	Varchar (50)	929836	50+	
14.	LastEditorUserId	1460940	Integer	489601	50+	
15.	OwnerDisplayName	BEEDELL ROKE JULIAN LOCKHART	Varchar (50)	921039	50+	
16.	OwnerUserId	1065077	Integer	16690	50+	Yes (FK)
17.	ParentId	1000012	Integer	419156	50+	
18.	PostTypeId	1	Small Integer	0	7	Yes

19.	Score	1612	Small Integer	0	50+	Yes
20.	Tags	Text (103)	Varchar (110)	526871	50+	
21.	Title	Text (150)	Varchar (160)	526871	50+	
22.	ViewCount	1940878	Integer	526871	50+	
Post Links (Total Records: 335807)						
1.	Id	10000037	Integer	0	50+	Auto (PK)
2.	CreationDate	2010-08-03T14:28:34.573	Timestamp	0	50+	Yes
3.	LinkTypeId	1	Small Integer	0	2	Yes
4.	PostId	1000014	Integer	0	50+	Yes (FK)
5.	RelatedPostId	11710552	Integer	0	50+	Yes (FK)
Comments (Total Records: 1517942)						
6.	Id	1000000	Integer	0	50+	Auto (PK)
7.	ContentLicense	CC BY-SA 2.5	Varchar (20)	0	3	Yes
8.	CreationDate	2010-07-28T19:36:59.773	Timestamp	0	50+	Yes
9.	PostId	1000001	Integer	0	50+	Yes (FK)
10.	Score	146	Small Integer	0	50+	Yes
11.	Text	Long Text (600)	Text	1	50+	
12.	UserDisplayName	Eugene Mayevski 'EldoS Corp	Varchar (50)	1483493	50+	
13.	UserId	1089473	Integer	33684	50+	Yes (FK)
PostHistory (Total Records: 3069982)						
1.	Id	1000001	Integer	0	50+	Auto (PK)
2.	Comment	Long Text (329)	Text	1853626	50+	
3.	ContentLicense	CC BY-SA 2.5	Varchar (20)	202799	3	
4.	CreationDate	2010-07-28T19:04:21.300	Timestamp	0	50+	Yes
5.	PostHistoryTypeId	12	Small Integer	0	31	Yes
6.	PostId	1000000	Integer	0	50+	Yes (FK)
7.	RevisionGUID	Text (36)	Varchar (40)	0	50+	Yes
8.	Text	Long Text (30011)	Text	240634	50+	
9.	UserDisplayName	Jigar Joshi - org.life.java	Varchar (40)	3014787	50+	
10.	UserId	1089473	Integer	199642	50+	Yes (FK)
Votes (Total Records: 4873241)						
1.	Id	1000000	Integer	0	50+	Auto (PK)
2.	BountyAmount	100	Small Integer	4857378	16	
3.	CreationDate	2010-07-28T00:00:00.000	Timestamp	0	50+	Yes
4.	PostId	1000003	Integer	0	50+	Yes (FK)
5.	UserId	1000564	Integer	4862143	50+	
6.	VoteTypeId	11	Small Integer	0	50+	Yes
Tags (Total Records: 3155)						
1.	Id	10003	Integer	0	50+	Auto (PK)
2.	Count	16370	Integer	0	50+	Yes
3.	ExcerptPostId	1435100	Integer	604	50+	Yes (FK)
4.	TagName	windows-subsystem-for-linux	Varchar (40)	0	50+	Yes
5.	WikiPostId	1435099	Integer	604	50+	Yes (FK)

Question 4: Provide a program to load the Stack Exchange data into the database. Provide explanations on handling associated issues.

The program code is provided in the "app.py" to load the data into the database. The program can be executed by following the README.md instructions. The "app.py" (in the root directory), contains method calls from the other files (db_methods.py, db_utils.py) in the "code" subfolder to make the code more manageable and readable. The insert SQL statements are provided in "globals.py" inside the "code" subfolder.

I read data from files in chunks and inserted each chunk into the database. To ensure the consistency of the database (i.e., to ensure insertion does not violate the foreign key constraints), I took the foreign key values from the chunk and queried those values to ensure those references existed in the database. If all the values matched with existing referenced primary keys in the referenced table; I safely inserted all of the chunks. In case, if some values were not present in the reference table; I remove rows from the chunk for non-existing referenced keys.

On my machine, the code took about 1451 seconds (about 24 minutes in repeated runs) in total to create a schema, insert data, and test transactions.

Question 5: Provide a program that connects to the previous database and creates a transaction to insert three rows of data. Force an error in row #2 so that the transaction aborts. Ensure that the database is in the same state as before.

The transaction test is provided in the “app.py” and it runs after the data insertion process. The transaction test method is provided in “db_methods.py” in the “code” subfolder. In this case, I try to insert three “comments” to test the transaction. The first one is a valid comment having all associated attributes. The comment gets inserted. For the second row, I try to insert a comment with a non-existing user (violating the constraint). This fails the transaction. The transaction rollbacks ensure that the database is consistent and no partial record updates are processed. That means the first inserted comment is not committed to the database.

Note: Instructions to run the code are provided in the README.md file in the root folder.