thoughts

Choosing a database

I took my time in choosing the database, should it be sql or no sql?
the data is pretty structured and has a straight forward schema which hints to go for sql.on the other hand, sql databases don't scale well. in order to understand if I would need to scale or not I'm going to take some assumptions about the amount of data which is going to be in this database.

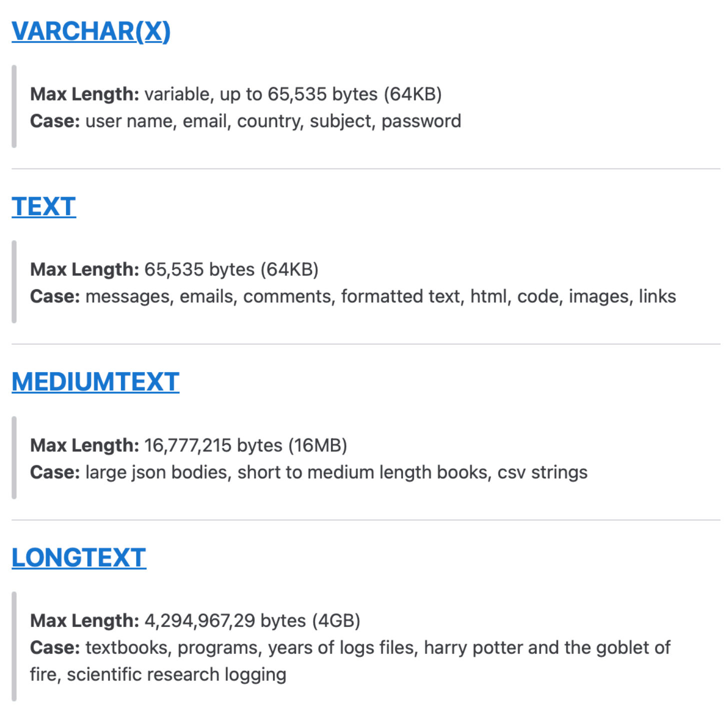
I researched the web using chatgpts aid to understand the traffic of NBC, one of the most widely used news website. Lets say there are 300 articles per day, in one year it amount to 100,000 and in 10 years to a million. NBC started its journey around the 2000's so lets assume there are roughly 3 million articles. About the users, lets say around 10,000 users are writing articles and each article gets around 100 comments, so 300 million comments.

	Size	Estimated amound	Total size
Article name	20 characters	3,000,000	60,000,000
Article content	5,000 words = 50,000 characters	3,000,000	150,000,000,000
User name	20 characters	10,000	200,000
Comment	50 words = 500 characters	300,000,000	150,000,000,000

Total amount of data in bytes is around 300 billion bytes which are 300gb. From what I understand, tables that are bigger than 1tb can encounter performance issues, so it is acceptable to put this data in sql tables even for a major news website like NBC. For the sake of this task I will use sqlite, but if this was a really system I would use postrage or MySql.

Article schema

Another thing I was taking into consideration is the schema of the article. More specifically, should I keep the content inside the table? Or keep the actual content in some object storage like s3 and keep only a url to it? A quick search in the web showed me what is the best way to keep large files in a sql database



VARCHAR(X)

Max Length: variable, up to 65,535 bytes (64KB)
Case: user name, email, country, subject, password

TEXT

Max Length: 65,535 bytes (64KB)
Case: messages, emails, comments, formatted text, html, code, images, links

MEDIUMTEXT

Max Length: 16,777,215 bytes (16MB)
Case: large json bodies, short to medium length books, csv strings

LONGTEXT

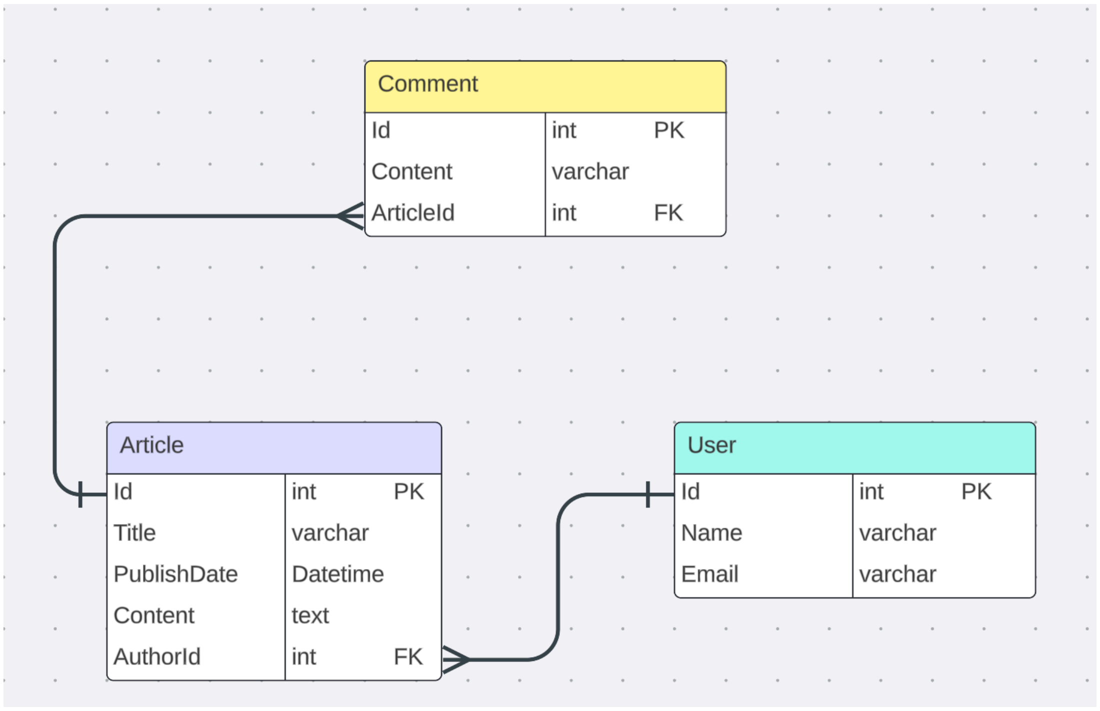
Max Length: 4,294,967,29 bytes (4GB)
Case: textbooks, programs, years of logs files, harry potter and the goblet of fire, scientific research logging

For our usage, 5,000 words are translated to 50,000 characters which are 50kb. So It seems reasonable to use TEXT or MEDIUMTEXT. These types are stored off the table while keeping some pointer to it, in contrast to varchar which keeps the data inside the table.

I think for our usage this solution suits us perfectly well, but I am making an assumption that the article does not contain any videos, images etc…

Of course these types of files are much larger and should be kept in an object storage.

Schema



API hierarchy

Because we have this kind of chained relation, a user has many articles and an article has many comments, we could design the hierarchy of the api in a manner that shows the relation in the endpoints, like so:

http method	Endpoint	Meaning
Post	/v1/users/	Create a new user
Get	/v1/users/	Read all users
Get	/v1/users/{user_id}	Read a user with user_id
Post	/v1/users/{user_id}/articles/	Create a new article which is written by user_id
Get	/v1/users/{user_id}/articles/	Read all articles written by user_id
Get	/v1/users/{user_id}/articles/{article_id}	Read a specific article
Post	/v1/users/{user_id}/articles/{article_id}/comments/	Create a new comment for article_id
Get	/v1/users/{user_id}/articles/{article_id}/comments/	Read all comments of article_id
Get	/v1/users/{user_id}/articles/{article_id}/comments/{comment_id}	Read a specific comment

On the other hand, we might want to have a root endpoint for articles and comments which are unrelated to a user, and the relation to a user can be sent through query params or inside the request body. This approach might be more flexible and focus more on the objects themselves instead of the relations between them.

http method	Endpoint	Meaning
Post	/v1/users/	Create a new user
Get	/v1/users/	Read all users
Get	/v1/users/{user_id}	Read a user with user_id
Post	/v1/articles/	Create a new article which is written by user_id (user_id is sent in the body)
Get	/v1/articles/	Read all articles written by user_id (user_id is an optional parameter, if it is empty all articles would be retrieved)
Get	/v1/articles/{article_id}	Read a specific article
Post	/v1/comments/	Create a new comment for article_id (article_id is sent in the body)
Get	/v1/comments/	Read all comments of article_id (article_id is an optional parameter, if it is empty all comments would be retrieved)
Get	/v1/comments/{comment_id}	Read a specific comment