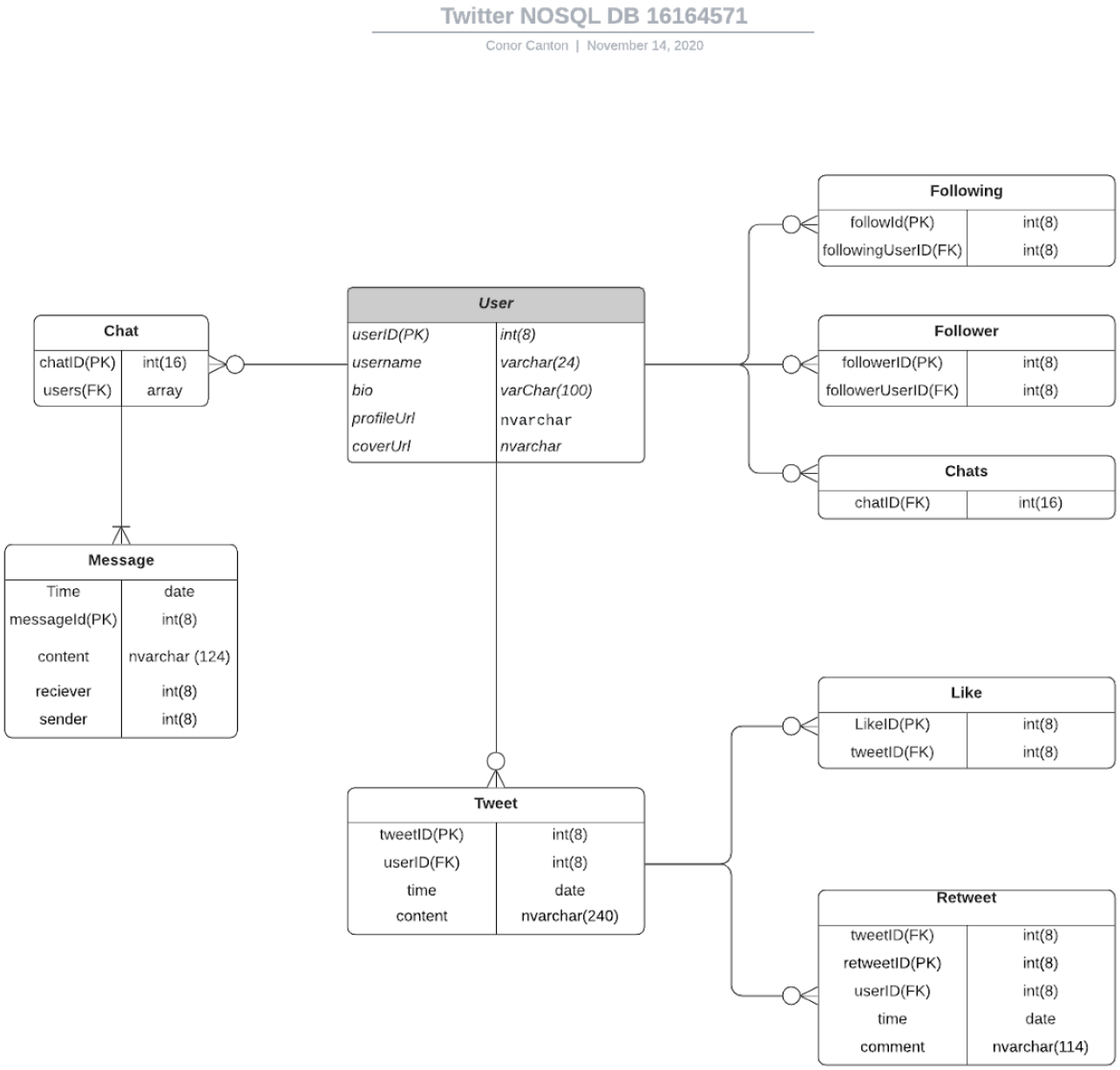
# 16164571 Twitter NoSQL Report

For this database model I choose to have 3 main documents. These documents are user,tweet and chats. Each document contains its own individual sub collection which can be seen in the ER Diagram.



# MongoDB JSON

## User

Chat

```
"chat": {
    "chatId": 1616457112345421,
    "users": ["16164571", "12345421"],

    "message": {
        "time": "04-08-2020 10:10:10",
        "messageId": 46745787,
        "content": "This is an example message",
        "reciever": 16164571,
        "sender": 12345421
    }
}
```

Tweet

```
"tweet": {
    "tweetId": 12340000,
    "userId": 16164571,
    "time": "04-08-2020 10:10:10",
    "content": "This is an example tweet",
    "isRetweet": false,

    "likes": {
        "like_id": "abc123be",
        "liked_user_id": "13432137"
    },

    "retweet": {
        "tweetId": 123404560,
        "userId": 16747288,
        "time": "04-08-2020 10:10:10",
        "comment": "This is an example retweet"
    }
```

I believe that a NOSQL database implementation would be well suited to twitter. NOSQL is well suited to fast paced, agile teams and this type of database would allow for fast changes to the database schema as scopes within twitter development teams may change. As twitter is constantly being developed and new features have been added all the time, NOSQL allows for the easy changing and adding of features, function and data types. Data consistency is also not so much of an issue with NOSQL and in relation to social media application. It isn't important for everyone to view your latest post at the exact same time.
NOSQL is also scaled very well horizontally meaning it can be spread out across hundreds of servers depending on demand as Twitter grows and becomes larger.