DSCI 551 - Project Proposal

Basic design of the system

Dataset Description

Relational Model: For the SQL part of the application's design, we are working with a YouTube dataset, and the data has details on the specific videos on the platform and other analytical data related to the videos on the platform. We have the rank of the associated user and data on subscribers, video views, and other metrics on the YouTube data. There is information about the monetary benefits on the data and the earnings and some demographic information on the data and the viewers. Therefore, among the attributes of the data elements available to us, we have divided the data model into some specific categories and model them in an Entity Relationship model so that we can model the information into a relational model, thereby implementing the features of a database system.

Non-relational Model: Using a NoSQL database to efficiently manage and analyze Air Traffic Landing statistics at San Francisco International Airport (SFO), we've organized the data into distinct categories tailored for airlines, airport authorities, and aircraft manufacturers. Each category is represented as an entity in the data model. By structuring the data into these well-defined entities and relationships, we empower airlines, airport authorities, and aircraft manufacturers to perform in-depth analyses, draw actionable insights, and enhance their operations and designs based on the comprehensive flight data available in the NoSQL database.

1. Data model

1.1 SQL

Specifics of the Data Model:
URL: https://www.kaggle.com/datasets/nelgiriyewithana/global-youtube-statistics-2023
The entities available from the data we are working with can be split into separate entities and follow a strict schema which would help us run queries against the data more easily and find out the details as requested. Therefore, we work on a specific implementation of finding out the attributes and grouping them into entities to make relations between the attributes. For the youtube dataset we have the following Entities:

Creator , Video Statistics , Country Details , Subscribers , Monetary Statistics. We would be storing the creator statistics like the name , creator category etc. Some specific attributes associated with the creator would be the subscriber details and information about the videos the creator is creating. The Country contains information about the video and more details on the demographics of the creator and information. Apart from this we associate the information of the creator with the video and then link it to the Monetary Statics table.

Implementation Details:
The denormalized data from the dataset is filtered and then put into separate files each analogous to a table from a relational database. For doing the joins and projections the program is supposed to read specific tables and load only related information inside the memory making sure that the program works even if the data could not be loaded onto the memory but the current query works without using up the entire memory taken up by all the data stored in the database. This would be done using a python dictionary and a customized version of a user declared hashmap similar to a key value store which ensures that we have a fixed schema since user attributes are user added. When we need to do the joins and projections we would filter the tables using the keys which would be the primary key analogous to a relational database and for walking the inbound and outbound foreign keys. The file structure would allow for creating a database and would be crucial to the workflow ,therefore to create a new database would be analogous to creating a file with specified attributes i.e. the column of those file names. Additionally using this structure of the data allows for essentially duplicating the basic functionality of a relational database model. We would implement a python library which takes the inputs from the prompt to create the files. Allow for random access to the specific rows using keys of the hashmap we created. Additionally, we would also be implementing an algorithm which would generate UUID with each new row inserted so that we have an inbuilt mechanism to account for data duplicacy and fixes for cases where a hash conflict occurs during data ingests and retrieval. For this we would be leveraging the uuid generator for ensuring uniqueness.

1.2 NoSQL :
Specifics of data model:
https://www.kaggle.com/datasets/lostinworlds/sf-air-traffic-landing-statistics
This data model will be organized into collections that represent various entities related to landing statistics. This data model for Air Traffic Landing statistics at SFO airport in MongoDB organizes the data into three collections: Airlines, AirportRegions, and AircraftDetails. Each collection has a well-defined structure which can be adjusted on the fly to store and retrieve landing statistics efficiently based on airlines, geographic regions, and aircraft models. This model enables comprehensive analysis and insights

for airlines, airport authorities, and aircraft manufacturers to enhance their operations and decision-making processes.

Explanation :
In the "Airlines" collection, each document represents an airline. The "OperatingAirline" field stores the name of the operating airline. The "OperatingAirlineIATACode" field stores the IATA code of the operating airline. The "PublishedAirline" field stores the name of the published airline (if different from the operating airline). The "PublishedAirlineIATACode" field stores the IATA code of the published airline (if different from the operating airline). In the "AirportRegions" collection, each document represents an airport region. The "GEO Summary" field stores a summary description or category related to the airport region. The "GEO Region" field stores the name of the airport region.In the "AircraftDetails" collection, each document represents an aircraft. The "Landing Aircraft Type" field stores a description or category related to the aircraft type. The "Aircraft Body Type" field stores information about the body type of the aircraft. The "Aircraft Manufacturer" field stores the name of the aircraft manufacturer. The "Aircraft Model" field stores the name of the aircraft model. The "Aircraft Version" field stores the version or variant of the aircraft. Since this model is going to mimic the design on a NoSQL Database therefore we allow for adjusting the schema design on the fly and the collections are internally represented by files and data in format of JSON blob for management and storage purposes.

2. Query language:

Since we are abstracting the files as tables the query language would allow for inserting data into the tables by creating files. So once the application is launched and run we create an environment which steps up the database and starts taking inputs from the user for finding out the specific results. The data model uses the same query language for both the data models in SQL and No-SQL.

Therefore, inside the interactive shell to interact with the database we have first to create the database

2.1 CREATE

Query: CREATE A TABLE ${TABLE_NAME} (ATTRIBUTE DATA_TYPE ATTRIBUTE_2 DATA_TYPE_2 ….)
This query creates a table whose name is table name and inside the brackets expects the command which takes the Attribute_Name_1 and Data_Type_1 and so on which

would be the name of the attribute and the corresponding data type and it would be very similar.
This operation returns the name of the database on success and if an error occurs returns the error message.

2.2 INSERT

Query: INSERT INTO ${TABLE_NAME} ({VALUE_ATTRIBUTE_1 , VALUE_ATTRIBUTE_2})

Here, the table name is the name of the table created already and inside the brackets we have an ordered list which lists out the values which need to be inserted inside the attributes.

(NOTE: We have implemented the functionality for the database to verify types inside the data being inserted into the tables).
Returns the database table and the attributes that have been inserted.
Query: UPDATE INTO ${TABLE_NAME} ${UUID} ({VALUE_ATTRIBUTE_1 , VALUE_ATTRIBUTE_2})

Here, the update attribute would be using the name of the table and find out the specific record to be updated using the UUID and therefore use that as a filter into the database and would find out the record and the value inside the curly braces would be the ones with which the database would be updated.
Returns the database name and the updated records.

2.3 DELETE

DELETE FROM ${TABLE_NAME} ${UUID}

Deletes the row from the table using UUID.

2.4 PROJECTION

FIND ALL FROM $TABLE_NAME -> For finding all records

Or

FIND {ATTRIBUTE_1} , {ATTRIBUTE_2} FROM ${TABLE_NAME} Project to select some records.

2.6 FILTER:

FIND {ATTRIBUTE_1} , {ATTRIBUTE_2} FROM TABLE $TABLE_NAME WHOSE {ATTRIBUTE} > $VALUE

Finds the list of records based on a filter.

2.7 JOIN:

FIND {ATTRIBUTE_1} , {ATTRIBUTE_2} , {ATTRIBUTE_3} FROM ${TABLE_1}.{ATTRIBUTE_1} = ${TABLE_2}.{$ATTRIBUTE_2} FROM WHERE ${CONDN_1} AND ${CONDN_2}

Joins Table 1 and Table 2 using the attributes specified.

2.8 AGGREGATE:

AGGREGATE COUNT{ATTRIBUTE_1} , {ATTRIBUTE_2} FROM ${TABLE} GROUP BY {ATTRIBUTE_1}

Aggregates based on the characteristics mentioned.

2.9 GROUP

SELECT {ATTRIBUTE_1} , {ATTRIBUTE_2} FROM ${table_name} WHERE condition
GROUP BY {ATTRIBUTE_1} ORDER BY {ATTRIBUTE_1}...;

Finds and groups records together and then orders them.

2.10 ORDER

SELECT {ATTRIBUTE_1} , {ATTRIBUTE_2} FROM ${table_name} ORDER BY {ATTRIBUTE_1} , {ATTRIBUTE_2}...

Shows selected attributes by arranging the table in the order by specified attributes.

Here ,we specify the name of the table from which we need to delete and provide the unique ID(UUID / PK) to the record that needs to be deleted.

3. Project Group member

1. Karan Manishkumar Shah
   Degree - MS in Computer Science
   Skills: Proficient in NoSQL database systems, data modeling, and data integration.
   Responsibilities:
   - Design and implement the NoSQL part of the hybrid database system.
   - Choose an appropriate NoSQL database system based on project requirements.
   - Define data models and collections/documents for unstructured or semi-structured data.
   - Develop data access and retrieval methods using NoSQL queries and APIs.
   - Ensure data consistency and scalability in the NoSQL database

2. Aniket Kumar
   Degree - MS in Computer Science
   Skills: Proficient in SQL, database design, and query optimization.
   Responsibilities:
   - Design and implement the SQL part of the hybrid database system.
   - Create and manage the relational schema, tables, and indexes.
   - Develop SQL queries and stored procedures for structured data.
   - Optimize SQL queries for performance.