NoSQL Database & AWS Architecture

NoSQL Database:

When structuring the star schema in a specific NoSQL non-relational database like MongoDB or Neo4j, the structure of the database would be different compared to a relational database. NoSQL databases are designed to be flexible and schema-less, allowing for easy scalability and handling of unstructured or semi-structured data.

In the case of the disease star schema, the structure in a NoSQL database would be represented differently, taking advantage of the features and capabilities of the chosen NoSQL database. Here are some possible approaches for representing the star schema in different NoSQL databases:

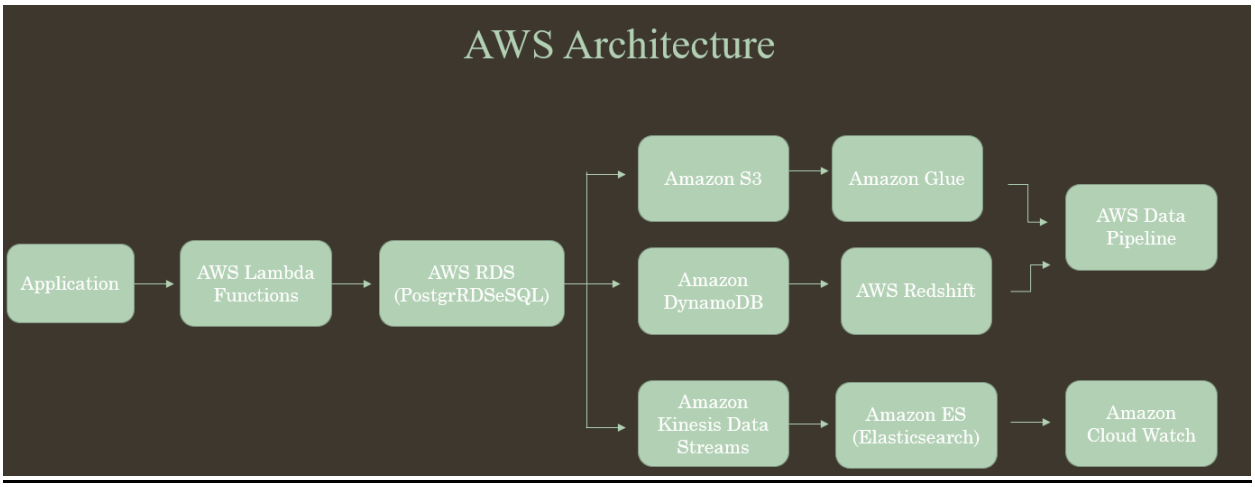
1. MongoDB:

   - The fact table, patient_fact, can be stored as a collection in MongoDB, with each document representing a patient's record. The attributes from the fact table, such as symptom_id, disease_id, hospital_id, insurance_id, medication_id, medical_history, severity_type, severity_score, length_of_hospital_stay, and diagnosis_month, can be stored as fields within the document.
   - The dimension tables, patient_dim, medication_dim, hospital_dim, insurance_dim, symptom_dim, and disease_dim, can be stored as separate collections in MongoDB. Each document within the dimension collections represents a unique entity and contains the corresponding attributes.
   - To establish relationships between documents, we can use embedded documents or references within the fact and dimension collections. For example, we can include the symptom_id directly within the patient_fact document or use references to link to the corresponding symptom document in the symptom_dim collection.

2. Neo4j:

   - In Neo4j, the star schema can be represented using nodes and relationships.
   - Each dimension table, such as patient_dim, medication_dim, hospital_dim, insurance_dim, symptom_dim, and disease_dim, can be represented as separate node labels. Each node represents a unique entity and contains the corresponding attributes as properties.
   - The fact table, patient_fact, can be represented as a node labeled "Patient" with its own unique properties, such as severity_type, severity_score, length_of_hospital_stay, and diagnosis_month. The relationships between the "Patient" node and the dimension nodes can be defined using relationships. For example, a relationship "HAS_SYMPTOM" can connect the "Patient" node to the corresponding "Symptom" node.

AWS Architecture:



In AWS, to create a resilient, high-performing, and secure architecture for the database and application, we can leverage various AWS services and components. One architectural pattern that aligns with these requirements is the Lambda Architecture, which combines batch processing and real-time streaming to handle data ingestion, processing, and querying.

AWS Database Services:

Amazon RDS (Relational Database Service): Use RDS to host the relational database, such as PostgreSQL, to store the data in a structured manner.

Amazon DynamoDB: Use DynamoDB as a NoSQL database for fast and scalable access to dimension tables or other data with flexible schemas.

AWS Analytics Services:

Amazon Kinesis Data Streams: Use Kinesis Data Streams to ingest and process real-time data streams from various sources, enabling real-time processing and analysis.

AWS Glue: Use Glue for data transformation, data cataloging, and ETL (Extract, Transform, Load) processes to prepare the data for analysis.

AWS Compute Services:

AWS Lambda: Use Lambda functions to implement serverless computing for both batch and real-time processing. For batch processing, trigger Lambda functions periodically using AWS CloudWatch Events or AWS Step Functions. For real-time processing, Lambda functions as an event-driven computer for processing data as it arrives in Kinesis Data Streams.

AWS Storage Services:

Amazon S3 (Simple Storage Service): Use S3 to store both raw data and processed data. Store the batch data in S3 for long-term storage and use S3 for intermediate storage during data processing.

Amazon Redshift: A data warehousing service used for large-scale data analytics & business intelligence.

Amazon ES: A fully managed search & analytics engine for real-time data exploration & visualization.

AWS Data Pipeline: To schedule & automate data movement & transformation workflows.

Amazon Cloud Watch: Used for batch processing, also trigger Lambda functions periodically.