perplexity

est ce que maintenant tu peux me faire un cours COMPLET en te basant sur toutes les infos que tu a, je veux etre le plus pret possible

Alright, combining the course outline with the image details, I can create a more structured and comprehensive review for your NoSQL exam.

NoSQL Exam Preparation: A Comprehensive Review

This review is structured to cover the main topics outlined in your course materials, focusing on key concepts and potential exam questions.

1. MongoDB Essentials

1.1. Installation and Database Design

- **Installation:** Familiarize yourself with the steps to install MongoDB. This may involve downloading the correct version for your operating system, configuring the installation, and verifying that the MongoDB server is running.
- **Database Design:**
  - **JSON (JavaScript Object Notation):** Understand the basic syntax and structure of JSON documents. MongoDB stores data in BSON (Binary JSON), which is a binary-encoded serialization of JSON-like documents.
  - **Data Modeling: Embedded vs. Normalized:**
    - **Embedded Data Model:** Store related data within a single document. Advantages: faster reads, fewer queries. Disadvantages: potential data redundancy, updates can be more complex. Use Cases: Data that is frequently accessed together, read-heavy applications.
    - **Normalized Data Model:** Store related data in separate collections and use references (links) to connect them. Advantages: data consistency, reduced redundancy. Disadvantages: slower reads, requires multiple queries to retrieve related data. Use Cases: Data that is frequently updated, write-heavy applications, scenarios where data consistency is critical.

1.2. CRUD Operations

- **Create:**
  - `insertOne()`: Inserts a single document.
  - `insertMany()`: Inserts multiple documents. The `ordered` option controls whether the operation stops if one insertion fails.
- **Read:**
  - `find()`: Retrieves documents. Uses query operators to filter results. Projection operators control which fields are returned.
- **Update:**
  - `updateOne()`: Updates a single document matching the query.
  - `updateMany()`: Updates multiple documents matching the query.
  - `replaceOne()`: Replaces the entire document with a new document.
- **Delete:**
  - `deleteOne()`: Deletes a single document.
  - `deleteMany()`: Deletes multiple documents.
- **CUD Operations Atomicity:** MongoDB provides atomicity at the document level. This means that a single write operation on a single document is atomic (either it succeeds completely or it fails completely).

1.3. Querying Documents

- **Query Operators:**
  - Comparison Operators: `$eq`, `$gt`, `$gte`, `$lt`, `$lte`, `$ne`
  - Logical Operators: `$and`, `$or`, `$not`, `$nor`
  - Element Operators: `$exists`, `$type`
  - Evaluation Operators: `$regex`, `$mod`
  - Array Operators: `$in`, `$nin`, `$all`, `$size`
- **Projection Operators:** Include or exclude specific fields in the results. Use `1` to include a field and `0` to exclude a field.

1.4. Cursor Methods

- `sort()`: Sorts the results.
- `limit()`: Limits the number of returned documents.
- `skip()`: Skips a specified number of documents.

1.5. Aggregation Framework

- **Aggregation Pipeline:** A sequence of stages that transform and analyze data.
- **Key Pipeline Stages:**
  - `$match`: Filters documents.
  - `$group`: Groups documents based on a specified key.
  - `$project`: Reshapes documents (add, remove, rename fields).
  - `$sort`: Sorts documents.
  - `$limit`: Limits the number of documents.
  - `$skip`: Skips a specified number of documents.
  - `$unwind`: Deconstructs an array field.
  - `$lookup`: Performs a "join" with another collection.
- **Aggregation Operators:**
  - `$sum`, `$avg`, `$min`, `$max`, `$first`, `$last`, `$push`, `$addToSet`

1.6. SQL to Aggregation Map Chart

Be able to translate simple SQL queries to MongoDB aggregation pipelines and vice-versa. This includes `SELECT`, `WHERE`, `GROUP BY`, `ORDER BY`, and `LIMIT` clauses.

2. Neo4j Essentials

2.1. Introduction and Database Design

- **Installation:** Download and install Neo4j Desktop. Create a new database.
- **Graph Modeling:**
  - Identify the entities (nodes) and the relationships between them.
  - Consider the properties that nodes and relationships will have.
  - Design the graph model based on the types of questions you want to answer.
  - Modelling: Relational to graph

2.2. Introduction to Cypher Query Language

- **Basic Queries:**
  - `MATCH`: Finds patterns in the graph.
  - `CREATE`: Creates new nodes and relationships.
  - `SET`: Updates node and relationship properties.
  - `DELETE`: Deletes nodes and relationships.
  - `RETURN`: Specifies what data to return.
- **Clauses:**

    - `WHERE`: Filters results.
    - `ORDER BY`: Sorts results.
    - `LIMIT`: Limits the number of results.
    - `SKIP`: Skips results.
- **Subqueries:** Perform more complex traversals.

2.3. Constraints

- **Node Property Uniqueness Constraint:** Ensures that a property value is unique across all nodes with a specific label. This is very common.
    - `CREATE CONSTRAINT FOR (n:LabelName) REQUIRE n.propertyName IS UNIQUE`

2.4. Loading CSV Data

- Use the `LOAD CSV` command to import data from CSV files into Neo4j. Specify the file path, column delimiters, and how to map the CSV columns to node/relationship properties.

2.5. Using SET to Update

- Use the `SET` clause to update the properties of nodes and relationships.
- Example: `MATCH (n:Person {name: "Alice"}) SET n.age = 30`

2.6. Cypher - SQL Comparison

- Understand the different approaches to querying data in Cypher (graph-based) vs. SQL (relational).

2.7. Visualizing Graph Data

- Use the Neo4j Browser to visualize graph data. Explore the graph, expand nodes, and see relationships.

3. Cassandra Essentials

3.1. Introduction and Database Design

- **Installation:** Download and install Cassandra.
- **Data Model:**
    - **Keyspace:** A container for tables (like a database in SQL).
    - **Table:** A collection of rows.
    - **Column Family:** (Conceptually) A container for columns within a row.
    - **Columns:** Name, value, and timestamp.
    - Note: Cassandra is a distributed database, designed for high availability and scalability.

3.2. Introduction to CQL (Cassandra Query Language)

- **Basic Statements:**
  - `CREATE KEYSPACE`: Creates a keyspace.
  - `CREATE TABLE`: Creates a table. Specify primary key (partition key and clustering columns).
  - `SELECT`: Retrieves data.
  - `INSERT`: Inserts data.
  - `UPDATE`: Updates data.
  - `DELETE`: Deletes data.

General Exam Tips

- **Know the key differences:** Be able to compare and contrast the different NoSQL databases (MongoDB, Neo4j, Cassandra). Consider their data models, strengths, and weaknesses.
- **Practice, Practice, Practice:** Write queries, design data models, and explore the different databases.
- **Review previous quizzes:** Focus on the areas where you had difficulty.

This comprehensive review should give you a strong foundation for your NoSQL exam. Good luck! Let me know if you have any further questions!

⁂