Module 3:
Document Database

Text chapters 6, 7, & 8

University of Illinois
at Springfield

Document Database

- **In the context of document databases, documents are sets of key-value pairs.**
  - **In our text, documents are represented using JSON notation.**
- **Here are some core points about JSON:**
  - **Data is organized in key-value pairs, similar to key-value databases.**
  - **Documents consist of name-value pairs separated by commas. Documents start with a { and end with a }.**
  - **Names are strings, such as “customer_id” and “address”.**
  - **Values can be numbers, strings, Booleans (true or false), arrays, objects, or the NULL value.**
    - The values of arrays are listed within square brackets; that is, [and].
    - Values of objects are listed as key-value pairs within curly brackets; that is, { and }.

Document Database

- An advantage of documents over key-value databases is that related attributes are managed within a single object.

- This enables database developers to more easily implement common requirements, such as returning all attributes of an entity based on a filter applied to one of the attributes.

Document Database

- Documents are roughly analogous to rows in relational tables.
- Collections, or groups of documents, are roughly equivalent to tables in relational databases.
  - Documents in the same collection do not need to have identical structures, but they should share some common structures.
- This is a significant difference between document and relational databases.
  - Document databases are considered schemaless because of this feature.

Designing Documents and Collections

Here are tips for designing documents and collections:

- Avoid highly abstract entity types. They can lead to frequent filtering to find specific types of data.
  - Instead, use collections for more specific entity types.
- Watch for separate functions for manipulating different document types.
  - The application code that manipulates a collection should have substantial amounts of code that applies to all documents and some amount of code that accommodates specialized fields in some documents.
- Use document subtypes when entities are frequently aggregated or share substantial code.

Designing Documents and Collections

- **In general, avoid overly abstract document types.**
  - **If you find yourself writing separate code to process different document subtypes, you should consider separating the types into different collections.**
  - **Poor collection design can adversely affect performance and slow your application.**
    - There are cases where it makes sense to group somewhat dissimilar objects (for example, small kitchen appliances and books) if they are treated similarly (for example, they are all products) in your application.

Document Database Operations

- The basic operations on document databases are the same as other types of databases and include
  - inserting,
  - deleting,
  - updating,
  - retrieving.
- No standard data manipulation language, such as SQL, is used across document databases to perform these operations.

Document Database Design

- **Relational database design is based on rules of normalization. There are no such formalisms for document database design.**
  - **There are many cases where there is no single “correct” model.**
  - **Designers should make choices based on how the document database will be queried and updated.**

Document Database Terminology: DOCUMENT

- **Document: A group of ordered key-value pairs.**
  - **In some cases, the order of key-value pairs matters in determining the identity of a document.**
- **The same key values in different orders would constitute different documents.**
- **Keys are generally strings while values can be a variety of data types.**

Document Database Terminology: COLLECTION

- **Collection: A group of documents, frequently with similar structures.**
  - **Collections enable developers to efficiently operate on groups of related documents.**
  - **Collections support additional data structures, such as indexes, that make such operations more efficient.**

Document Database Terminology: EMBEDDED DOCUMENT

- **Embedded document: A document within a document.**
  - **This is used to avoid the need to perform joins.**
  - **An example of an embedded document is an order item document in an order document.**
- **Embedded documents are one method for modeling one-to-many relations in document databases.**

Document Database Terminology: SCHEMALESS

- **Schemaless: This refers to the fact that there is no need to define an explicit and relatively fixed structure to the data model.**
  - **Schemaless models are easy to work with from a developer's perspective because they can simply add attributes as needed.**
  - **This makes it more difficult to manage in large projects.**
- **Polymorphic schema: A document database is polymorphic because the documents that exist in collections can have many different forms.**

Document Database Terminology: PARTITIONING

- **Partitioning: The process of splitting a document database and distributing different parts of the database to different servers.**
- **Two common types of partitioning are:**
  - **vertical partitioning**
  - **horizontal partitioning.**

Document Database Terminology: VERTICAL PARTITIONING

- Vertical partitioning: A technique for improving database performance by separating columns of a relational table into multiple separate tables.
- This is used when the database has some columns that are frequently accessed and others that are not.
- Vertical partitioning is more frequently used with relational database management systems than with document management systems.

Document Database Terminology: HORIZONTAL PARTITIONING

- **Horizontal partitioning: A technique of dividing a database by documents in a document database or by rows in a relational database.**
  - **These parts of the database, known as shards, are stored on separate servers.**
  - **A single shard may be stored on multiple servers when a database is configured to replicate data.**
- **Shard key: One or more keys or fields that exist in all documents in a collection that is used to separate documents.**
  - **A shard key could be virtually any atomic field in a document.**

Document Database Terminology: PARTITIONING

- **Range partition: A technique that uses an ordered set of values for shard keys, such as dates and numbers.**
- **Hash partition: A hash partition uses a hash function to determine where to place a document.**
- **List partition: List-based partitioning uses a set of values to determine where to place data.**

Document Database Terminology

- **Normalization: Database normalization is the process of organizing data into tables in such a way as to reduce the potential for data anomalies.**
  - **An anomaly is an inconsistency in the data.**
- **Denormalization: A technique that undoes normalization; specifically, it introduces redundant data.**
- **Query processor: The query processor is an important part of a database management system. It takes as input queries and data about the documents and collections in the database and produces a sequence of operations that retrieve the selected data.**

Document Database Model Design

- Relational database designers start with rules of normalizations.
- Document database modelers depend more on heuristics, or rules of thumb, when designing databases.
- The rules are not formal, logical rules like normalization rules. One of the most important heuristics is to start with how users will query the document database.
- One of the goals of document data modeling is minimizing the number of read or write operations that must be performed when executing those queries.
  - This goal is balanced by another goal of not having more duplication than needed. This is a balancing act.

Document Database Model Design

- In relational data model design, one-to-many and many-to-many relations are modeled using multiple tables.
  - Queries on these tables require joins.
- In document data modeling, one tries to minimize joins.
  - This is typically done by embedding documents within other documents.
  - Embedded documents are a common way of denormalizing data.

Document Database Model Design: JOINS

- The three main ways of performing joins are to use:
  - nested loops,
  - sort merge,
  - hash joins.
- Nested loops work well when both tables are small.
- Sort merges work well with large tables that must be ordered.
- Hash joins work well when selecting a small subset of rows from large tables.

Document Database Model Design

- Use queries as a guide to help strike the right balance of normalization and denormalization.
- Too much of either can adversely affect performance.
  - Too much normalization leads to queries requiring joins.
  - Too much denormalization leads to large documents that will likely lead to unnecessary data read from persistent storage and other adverse effects.

Document Database Model Design

- Document data modelers also need to plan for growth in documents.
- Preallocating space for anticipated growth can help avoid performance problems in the future.
  - If a document outgrows its allocated space, it will need to be copied to another location in storage, introducing a delay in finishing an update operation.

Document Database Model Design

- **Use indexes as needed to improve read operations but consider the cost of maintaining indexes.**
  - **Each time an indexed attribute is inserted, updated, or deleted the corresponding index must be updated.**
    - This introduces additional work for the write operation.
- **Hierarchies can be modeled with parent-child, child-parent references.**
  - **Alternatively, one could include a list of all ancestors. This is called a materialized path.**