Document Databases and MongoDB Fundamentals
Briefing Document: Document Databases and MongoDB
Date: October 26, 2023**Source:** Excerpts from "07 - Document DBs and Mongo.pdf" by Mark Fontenot, PhD, Northeastern University
Overview:
This document provides an overview of document databases, focusing on MongoDB. It outlines the fundamental concepts of document databases, the structure and features of MongoDB, its relationship to relational databases, and basic interaction methods.
Main Themes and Important Ideas/Facts:
1. Document Databases as a Non-Relational Alternative:
•
A document database is defined as "a non-relational database that stores data as structured documents, usually in JSON."
•
The key characteristics highlighted are that they are "*designed to be simple, flexible, and scalable.*"
•
The document emphasizes that document databases address the "*impedance mismatch problem between object persistence in OO systems and how relational DBs structure data.*" In object-oriented programming, data is often represented through inheritance and composition, which can be complex to map directly to the tabular structure of relational databases. Document databases, with their flexible and self-describing document structure, offer a more natural alignment for such applications.
2. JSON as the Core Data Format:
•
JSON (JavaScript Object Notation) is presented as the primary format for storing data in document databases.
•
Key attributes of JSON are:
◦
"*a lightweight data-interchange format*"
◦
"*It is easy for humans to read and write.*"
◦
"*It is easy for machines to parse and generate.*"
•
JSON is built on two fundamental data structures:
◦
"A collection of name/value pairs" (objects, dictionaries).
◦
"An ordered list of values" (arrays, lists).
•

The document underscores the universality of these structures, stating, "*These are two universal data structures supported by virtually all modern programming languages*," making JSON an excellent data interchange format.

3. BSON (Binary JSON): The Internal Representation:

- BSON (Binary JSON) is introduced as the "*binary-encoded serialization of a JSON-like document structure*" used internally by MongoDB.
- Its advantages include:
  - Support for "*extended types not part of basic JSON (e.g., Date, BinaryData, etc.)*"
  - "*Lightweight - keep space overhead to a minimum*"
  - "*Traversable - designed to be easily traversed, which is vitally important to a document DB*"
  - "*Efficient - encoding and decoding must be efficient*"

4. Comparison with XML:

- XML (eXtensible Markup Language) is mentioned as a "*Precursor to JSON as data exchange format*."
- While structurally similar to HTML and extensible, JSON has become the preferred format due to its simplicity and ease of parsing.
- The document briefly lists XML-related technologies like XPath, XQuery, DTD, and XSLT.

5. Motivation Behind Document Databases (and MongoDB):

- Document databases emerged to solve the challenges of persisting complex objects from object-oriented systems into relational databases, avoiding the need to "*deconstruct it*."
- The self-describing nature of document structures makes them "*well-aligned with apps that use JSON/XML as a transport layer*."
- MongoDB's origin is traced back to 2007 when three veterans from Doubleclick, after its acquisition by Google, recognized "*the limitations of relational databases for serving > 400,000 ads per second*."
- "*MongoDB was short for Humongous Database*."
- MongoDB Atlas, a "*documentdb as a service*," was released in 2016.

6. MongoDB Structure and Key Concepts:

- MongoDB organizes data into:

◦
Databases: Logical groupings of collections.
◦
Collections: Analogous to tables in relational databases, but without a fixed schema.
◦
Documents: The fundamental unit of data storage, represented in BSON (a binary form of JSON).
•
A crucial characteristic of MongoDB is that "*No predefined schema for documents is needed*." This means "*Every document in a collection could have different data/schema*," providing flexibility.
•
The document provides a table comparing relational database concepts to MongoDB equivalents: | RDBMS | MongoDB | | ---------- | ---------- | | Database | Database | | Table/View | Collection | | Row | Document | | Column | Field | | Index | Index | | Join | Embedded Document | | Foreign Key | Reference |
7. MongoDB Features:
•
Rich Query Support: Offers "*robust support for all CRUD ops* (Create, Read, Update, Delete).
•
Indexing: Supports "*primary and secondary indices on document fields*" for efficient data retrieval.
•
Replication: Features "*replica sets with automatic failover*" for high availability and data redundancy.
•
Load balancing built in: Enables horizontal scaling by distributing data across multiple servers.
8. MongoDB Versions:
•
MongoDB Atlas: "*Fully managed MongoDB service in the cloud (DBaaS)*."
•
MongoDB Enterprise: "*Subscription-based, self-managed version of MongoDB.*"
•
MongoDB Community: "*source-available, free-to-use, self-managed.*"
9. Interacting with MongoDB:
•
mongosh: "*MongoDB Shell - CLI tool for interacting with a MongoDB instance.*"
•
MongoDB Compass: "*free, open-source GUI to work with a MongoDB database.*"
•
DataGrip and other 3rd Party Tools: Various tools offer MongoDB integration.
•
Language Drivers: "*Every major language has a library to interface with MongoDB*" such as "*PyMongo (Python), Mongoose (JavaScript/node), …*"

10. Basic MongoDB Operations using `mongosh` and PyMongo:

- The document provides examples of basic `mongosh` commands:
  - Selecting all documents in a collection (`db.users.find()`).
  - Filtering documents based on specific criteria (`db.users.find({"name": "Davos Seaworth"})`).
  - Using comparison operators (`$in`, `$gte`).
  - Combining conditions with logical operators (`$or`).
  - Counting documents (`db.movies.countDocuments(...)`).
  - Projecting specific fields (`db.movies.countDocuments(..., {"name": 1, "_id": 0})`).
- Basic PyMongo usage is demonstrated:
  - Connecting to a MongoDB instance (`MongoClient(...)`).
  - Getting a database and collection (`client['ds4300']`, `db['myCollection']`).
  - Inserting a single document (`collection.insert_one(post)`).
  - Counting documents in a collection (`demodb.collection.count_documents({})`).

Conclusion:

The provided excerpts offer a foundational understanding of document databases, highlighting their flexibility and scalability compared to relational databases. MongoDB is presented as a leading document database, emphasizing its JSON/BSON document structure, rich features, and various tools for interaction. The document effectively introduces the core concepts and provides a glimpse into basic query and manipulation operations using both the MongoDB shell and the PyMongo library.