DBMS UNIT V

No SQL Database

Why NoSQL?

- Most features of RDBMS are not required in internet/web-based applications.
  - Features like transaction support, structured data storage, powerful query language, data consistency.
- Instead, other features are more important for internet/web-based application.
  - Semi-structured data, high performance, availability, scalability, replication.

What do these applications need?

- Google/Bing/Yahoo — Index of documents on the web
- Google/Yahoo/Hotmail — emails
- Amazon — Products, Reviews, Ratings, Customers, Addresses
- Facebook — Posts(Videos, Images, Text,), Likes, Shares
- Twitter — Tweets, Replies, Retweets, Likes.
- Flickr, Google — Photos
- Youtube/Netflix — Videos
- News Sites — Articles, comments and replies

Scenario - 1

- You uploaded a picture to facebook.
  - Some of your friends see the picture and some do not see it. Is it OK?
- You withdrew money from ATM.
  - Is it OK to see two different balances like your friends are seeing two different views of your “facebook wall”?
- Is it better to let some friends view the post or wait till everyone can see the same post?

Scenario - 2

- You are browsing amazon for buying some item.
  - Are there more reads or writes?
  - Are there only text or text and images and videos of advertisements, product usage and user reviews and ratings as well?
  - Are all the products (say a camera and a mobile phone) having the same attributes?
  - Are there thousands or millions of – users, products, reviews, ratings, pictures, videos, etc?
  - Is it OK to shutdown the servers for maintenance?
  - Some time there are thousands of users and some other time there are millions of users.

NoSQL Database

- NoSQL databases emerged when SQL databases could not meet the requirements of new web applications like the ones mentioned earlier.
- NoSQL (non-relational) databases have existed since 1960s.
  - The name NoSQL was given in 1998 (wikipedia).
- Emergence of Internet (Increase in volume of data and the need for a better way of managing the data):
- Search-indexing, Web based Email, Amazon shopping, Social Network: Facebook, Twitter, Flikr, NetFlix, Youtube, ...

What is NoSQL?

- NoSQL stands for not only **SQL**.
  - **SQL** stands for **relational databases** and NOT **SQL the language.**
- Different applications require different (other than RDBMS) database approaches.
- Some NoSQL databases
  - BigTable by Google (Column-based or Wide Column, Apache Hbase is based on BigTable)
  - DynamoDB by Amazon (Key-Value store, Voldemort is key-value store)
  - Cassandra by Facebook (Based on both key-value and column-based. Available as Apache Cassandra)
  - MongoDB, CouchDB (Dcument based)
  - Neo4J Graph database.

Characteristics

- Scalability
  - From 100s to millions of users/usage/objects.
  - Adding more servers and databases handles increased users/usage/objects.
- Availability
  - Always on. Imagine life(for customers and companies) without Google / Amazon / FB...
- Replication Models
  - How and Where to store a copy (in case things go wrong, read from the copy)?
- Sharding of files (Horizontal partitioning of files)
  - Dividing rows into many parts and storing in many servers (for faster access).
- High Performance Data Access
  - The dreaded hourglass/spinning wheel… users will run away from slow sites.

More Characteristics

- Not requiring a Schema
  - Mostly accessed by keys.
  - Semi-structured data described by JSON or XML
- Less powerful Query Language
  - No support for SQL (or limited SQL support)
  - Access for CRUD through API
- Versioning
  - Storing multiple version of data with time stamps.

Four categories of NoSQL DBMS

- Document based NoSQL systems
  - MongoDB, …
- NoSQL Key-Value stores
  - Riak, Redis, memcachd, ...
- Column based or Wide column NoSQL systems
  - Cassandra, ...
- Graph based NoSQL systems
  - Neo4J

Document based NoSQL systems

- Also called: Document stores.
  - Examples: MongoDB, CouchDB
- Data stored as collections of similar documents. Each document has a unique ID with name _**id**.
- “Documents” are self describing. Attributes are stored as “attribute_name”=”value” format.
- No requirement that documents be similar.
- JSON-most popular, XML/other format also used.

MongoDB Data Model

- A server has many databases
- A database has many collections
- **db.createCollection(“name”, {capped : true, autoIndexID : true, size : 6142800, max : 10000})**
  - capped: there is a limit on size-true/false
  - autoIndexID: Whether to create index on _id
  - size: size for capped collection
  - max: number of documents for capped collection
- No need to create a collection explicitly. A collection is created:
  - When a document is created using db.collection_name.insert() if the collection_name does not exist in the database.
- **show collections** lists the collections in a database.

Data Model – 2 – CRUD Operations

- Creating documents
  - db.collection_name.insert({})
- Retrieving documents
  - db.collection_name.find(<condition>)
  - db.collection_name.find(“attribute_name”:”value”)
- Updating Documents
  - db.collection_name.update({query},{<new document>})
  - db.collection_name.update({query},$set:{field1:”new_value”})
- Deleting Documents
  - db.collection_name.remove({<condition>})

MongoDB commands - Create

- show databases
  - Lists the databases
- use database_name
  - Connect to a database or create a new database
- db.createCollection("books")
  - Creates a new collection with a name "books"
- db.books.insert({"_id":"10", "title": "Fundamentals of Database Systems", "edition": "7", })
  - Insert a record in the collection

MongoDB commands - Retrive

- db.books.find()
  - Lists the documents in the collection
- db.books.find({_id:"10"})
- db.books.find({"edition":"7"})
  - Find books by attribute name and value
- db.books.find({"edition":{$gt:"6"}})
- db.books.find({"edition":{$lt:"8"}})
- db.books.find({"edition":{$gt:"6",$lt:"8"}})

MongoDB commands - Update

- db.books.update({"_id":"10"},{$set:{"author":"Ramez Elmasri"}})
  - Adds the author attribute and value.
- db.books.update({"_id":"10"},{$set:{"author":"Shamkant Navathe"}})
  - Does not add a second author but changes the author name.
- db.books.update({"_id":"11"},{$set:{"author":["Ramez Elmasri"]}})
  - Make the Authors attribute a list (by enclosing in [ ])
- db.books.update({"_id":"11"},{$set:{"author.1":"Shamkant Navathe"}})
  - Note the author.1 to say that we are using index 1 to store second author name,
- db.books.find({"_id":"11"})
  - { "_id" : "11", "title" : "Fundamentals of Database Systems", "edition" : "8", "author" : ["Ramez Elmasri", "Shamkant Navathe"] }

MongoDB commands - Delete

- db.books.remove({"_id":"10"});
  - Delete the document with the “_id” of “10”
- db.books.save({"_id":"10", "name":"Some name of a book", "publisher":"Pearson"})
  - This command will update the document (with _id of 10) if it finds a document, else (if no document was found) it inserts a new document into the books collection.
  - This is called the UPSERT command

Summary

- Note
  - The commands are like function calls (API).
  - What works for one database(MongoDB) will not work on another database(CouchDB) (no standardization).
  - Schema less(No need to define a schema before inserting/updating. Each document can have different attributes and different number of attributes)
  - Mostly accessed by keys or some attribute which identifies a document.
  - No data independence. Programs have to be written for the application and the database model.

Questions