

NoSQL and Cloud Databases

Data Science Institute

Zia Babar, PhD
January 2023

Agenda

- Introduction
- Types of Data
- Cloud Computing
- Databases on the Cloud
- Non-Relational Databases

Speakers' Background

Zia is a practitioner and researcher in the space of enterprise information systems, with over two decades of professional experience in developing enterprise-grade, mission-critical software applications encompassing areas such as cloud computing, big data analytics, distributed systems, high-performance computing, and data engineering.

He has experience working in startups, scaleups and large global corporations in progressively senior positions, most recently at PwC Canada. He has a PhD from the University of Toronto where his research focused on data-centric systems design for fast moving enterprises.



Types of Data

Structured Data

- Structured data is comprised of clearly defined data types. It usually resides in relational databases (RDBMS).
- Data may be human- or machine-generated as long as the data is created within an RDBMS structure.
- This format is eminently searchable both with human generated queries and via algorithms using type of data and field names, such as alphabetical or numeric, currency or date.
- Examples of structured data are Phone numbers, Social Insurance Numbers, Postal Codes, Personal Names and Addresses.

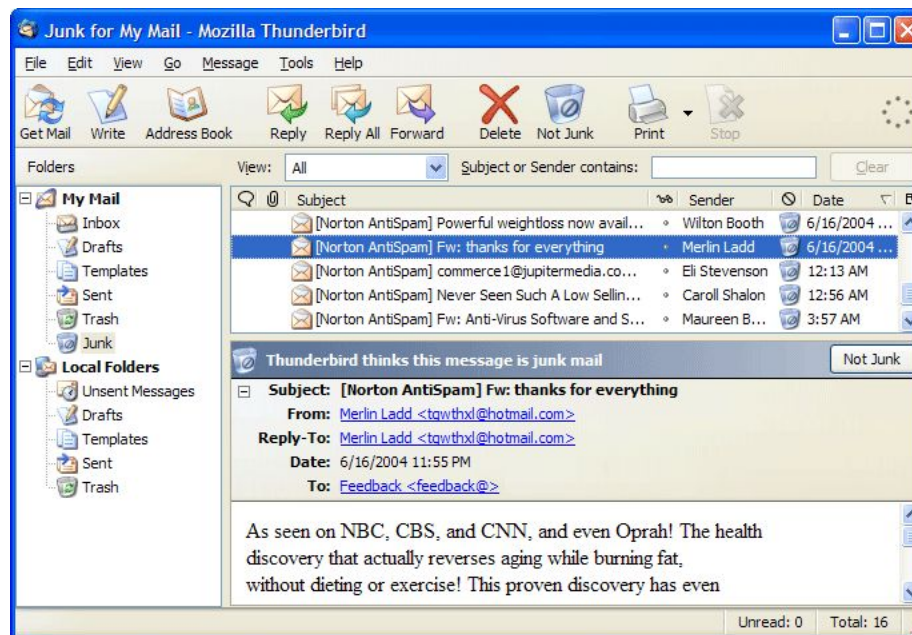
Unstructured Data

- Unstructured data is information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- Unstructured data may be in different formats.
- Unstructured data is difficult to search.
- Examples of "unstructured data" may include,
 - books, journals, documents, metadata, health records, audio, video, analog data, images, files
 - unstructured text such as the body of an e-mail message, Web page, or word-processor document.

Semi-Structured Data

- Semi-structured data maintains internal tags and markings that identify separate data elements, which enables information grouping and hierarchies.
- Some examples of semi-structured data in the enterprise include,
 - Email
 - Markup language like XML or JSON
 - NoSQL Semi-structured data
- NoSQL databases differ from RDBMS as they do not separate the organization (schema) from the data thus allowing storing of information that does not easily fit into the record and table format, such as text with varying lengths.

Semi-Structured Data



XML Data Format

- A lot of our information is not in the form of columns and rows.
 - This information can be organized as part of the XML document format.
- An XML document is a basic unit of XML information composed of elements and other markup in an orderly package.
 - An XML document can contains wide variety of data.
 - For example, database of numbers, numbers representing molecular structure or a mathematical equation.

XML Data Format

- Here is a sample of an XML music catalogue
 - Every tag that is opened also has to be closed
 - It is possible to nest tags inside each other:

```
<name>
  <first>John </first>
  <last>Smith</last>
</name>
```

```
<xml>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>

  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
</CATALOG>
</xml>
```

JSON Data Format

- JSON stands for JavaScript Object Notation and is a lightweight format for storing and transferring data.
 - It's self-describing and relatively easy to understand as the data in a JSON document is in name-value pairs.

```
{ "name": "John" }  
{ "firstName": "John", "lastName": "Smith" }
```

- JSON consists of name/value pairs and an ordered value list, such as arrays, sequences, lists, etc.
 - JSON excels at transmitting data between web applications and servers.

JSON Data Format

- Here is a sample of an JSON movie entry
 - Multiple name-value pairs exist.
 - The main characters has many actors present:

```
[
  "name": "value",
  "name": "value",
]
```

```
{
  "title": "The Godfather",
  "director": "Francis Ford Coppola",
  "yearFilmed": 1972,
  "novelWriter": "Mario Puzo",
  "mainCharacters": [
    {
      "characterName": "Don Vito Corleone",
      "actorName": "Marlon Brando"
    },
    {
      "characterName": "Michael Corleone",
      "actorName": "Al Pacino"
    },
    {
      "characterName": "Tom Hagen",
      "actorName": "Robert Duvall"
    },
    {
      "characterName": "Kay Adams",
      "actorName": "Diane Keaton"
    }
  ]
};
```

Differences between XML and JSON

XML	JSON
Extensible Markup Language	JavaScript Object Notation
Derived from SGML	Based on the JavaScript language
Supports comments	Doesn't support comments
Uses tag structure to represent items	Used to represent objects
Doesn't support arrays	Support arrays
Provides support for namespaces	Doesn't provide support for namespaces
Difficult to read and comprehend	Easy to read and comprehend
Uses start and end tags	Doesn't use end tags

Cloud Computing

What is Cloud Computing?

“Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider”

– Amazon AWS

What Cloud Providers Are There?








IBM Cloud



Google Cloud

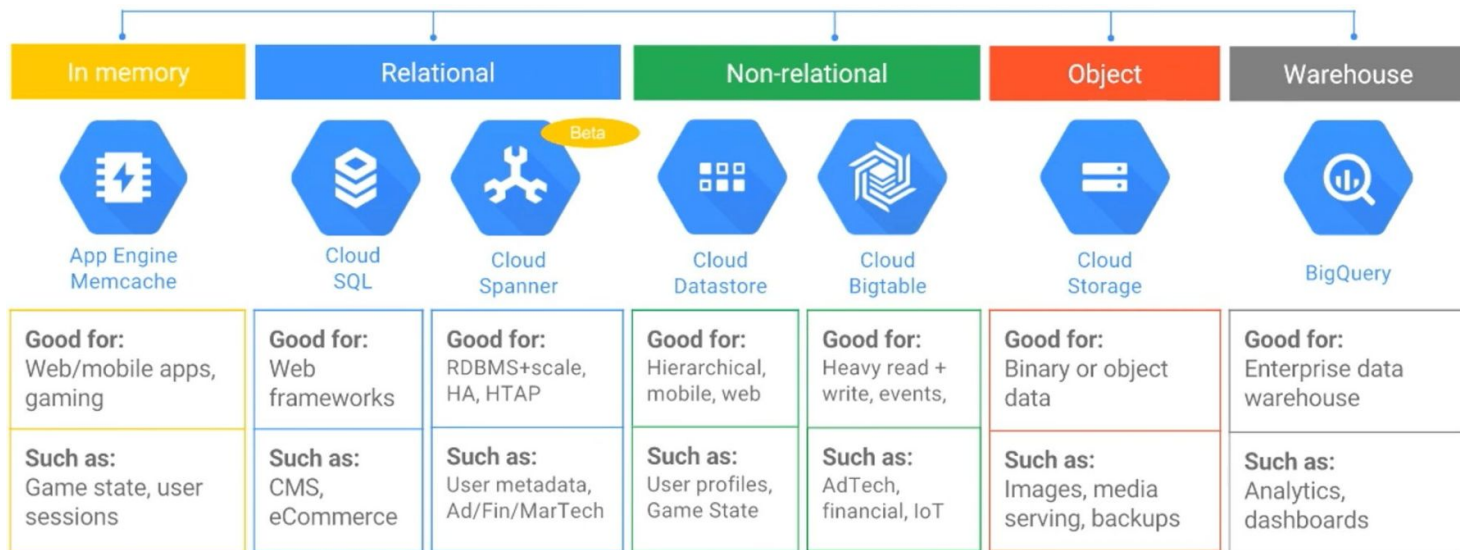
Databases on the Cloud

Microsoft Azure Databases

Relational	Key/Value	Column Family	Document	Graph
				
<ul style="list-style-type: none"> • Windows Azure SQL Database • SQL Server • Oracle • MySQL • SQL Compact • SQLite • Postgres 	<ul style="list-style-type: none"> • Windows Azure Blob Storage • Windows Azure Table Storage • Windows Azure Cache • Redis • Memcached • Riak 	<ul style="list-style-type: none"> • Cassandra • HBase 	<ul style="list-style-type: none"> • MongoDB • RavenDB • CouchDB 	<ul style="list-style-type: none"> • Neo4J

Source: Microsoft

Google Cloud Databases



Source: Google

IBM Cloud Database

Catalog

Search the catalog...

Services

Software

Private

All Categories

VPC Infrastructure

Compute

Containers

Networking

Storage

AI

Analytics

Databases >

Developer Tools

Integration

Internet of Things

Security and Identity

Starter Kits

Pg

Databases for PostgreSQL

IBM

PostgreSQL is a powerful, open source object-relational database that is highly customizable.

Databases

Rd

Databases for Redis

IBM

Redis is a blazingly fast, in-memory data structure store.

Databases

Es

Databases for Elasticsearch

IBM

Elasticsearch combines the power of a full text search engine with the indexing strengths of a JSON document database.

Databases

Mg

Databases for MongoDB

IBM


















MongoDB is a JSON document store with a rich query and aggregation framework.

Databases

Source: IBM

20

Amazon AWS Databases

								
	Relational	Key-value	Document	In-memory	Graph	Time-series	Ledger	Wide Column
	Referential integrity, ACID transactions, schema-on-write	High throughput, Low latency reads and writes, endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Collect, store, and process data sequenced by time	Complete, immutable, and verifiable history of all changes to application data	Scalable, highly available, and managed Apache Cassandra-compatible service
AWS Service(s)	  Aurora RDS	 DynamoDB	 DocumentDB	 ElastiCache	 Neptune	 Timestream	 QLDB	 Keyspaces Managed Cassandra
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	IoT applications, event tracking	Systems of record, supply chain, health care, registrations, financial	Build low-latency applications, leverage open source, migrate Cassandra to the cloud

Source: Amazon

Non-Relational Databases

Relational Databases

- SQL databases (also known as relational databases) have one type of structure, i.e. relational.
- Developed in the 1970s to deal with first wave of data storage applications.
 - Structure and data types are fixed in advance.
- Uses SQL based querying through keywords such as SELECT, INSERT, UPDATE etc.
- Examples:
 - MySQL, Postgres, Oracle, SQL Server, etc.

Non-Relational Databases

- NoSQL (or non-relational) databases refer to many different types of databases
 - key-value, document, wide-column, and graph
- Developed in 2000s to deal with limitations of SQL DBs concerning scale, replication, unstructured data.
 - Dynamic schemas. Records add new information on the fly.
 - Access is through object-oriented APIs.
- Examples
 - MongoDB, Cassandra, Neo4j, Redis Cache, etc.

Non-Relational Databases

- Despite the advantages of SQL or relational databases, there exist certain disadvantages as well.
 - Records must have a fixed set of columns. It's challenging to support records that do not have values for all the columns.
 - The column structure needs to be defined beforehand, and new columns on-demand cannot be done without alternating the database schema.
 - The rigidly defined schema used in relationship databases is a poor fit for unstructured and semi-structured data.

Non-Relational Databases

- NoSQL databases have dynamic schemas and records can be added with new information on the fly. Records are accessed through object-oriented APIs.
- Many different types of non-relational databases
 - Key-value
 - Document
 - Wide-column
 - Graph

Relational DBs and Non-Relational DBs

Relational DBs	Non-Relational DBs
<ul style="list-style-type: none">• Stored data in tables	<ul style="list-style-type: none">• Stores data in different forms
<ul style="list-style-type: none">• Great for structure data	<ul style="list-style-type: none">• Great for unstructured data
<ul style="list-style-type: none">• Strict database schema	<ul style="list-style-type: none">• Flexible schemas
<ul style="list-style-type: none">• Schema alteration required periodically	<ul style="list-style-type: none">• New properties easily accommodated
<ul style="list-style-type: none">• Scales well vertically	<ul style="list-style-type: none">• Scales well horizontally
<ul style="list-style-type: none">• Relationships are captured through JOINS	<ul style="list-style-type: none">• Relationships can be captured through denormalization

Key-Value Database

Key-Value Database

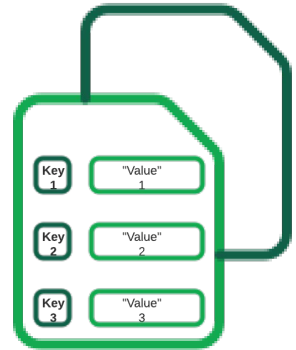
- A key-value database is a type of non-relational database that uses a simple key-value method to store data. Also known as key value stores.
- Data is stored in a “key-value” format and optimized for reading and writing that data.
 - Keys are used to retrieve the associated value with each key.
 - Values are simple data types like strings and numbers or complex objects.
 - Hold a single serialized object for each key value.
- Both keys and values can be anything, ranging from simple objects to complex compound objects.

Key-Value Database

- Key-value databases are highly partitionable and allow horizontal scaling at scales that other types of databases cannot achieve.
- Good for storing large volumes of data where you want to get one item for a given key value
- No need to query based on other properties of the item.
- Data is written (inserted, updated, and deleted) and queried based on the key to store/retrieve its value.

Background

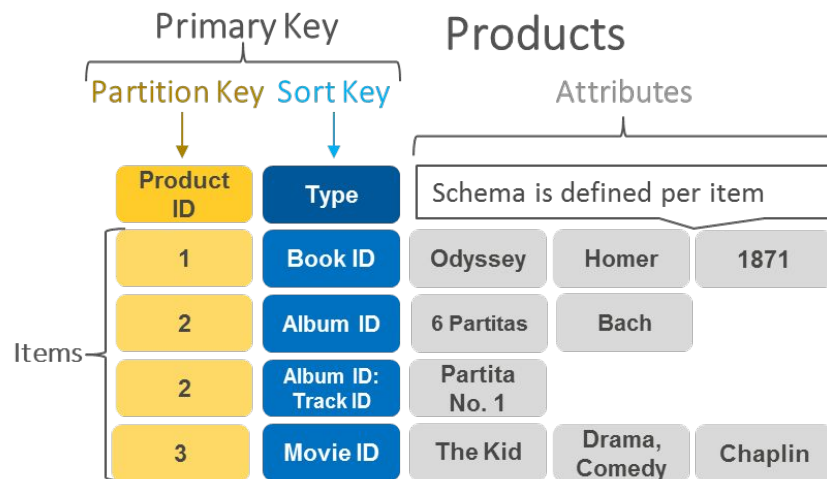
- Key-value database are not new concept and has been around for the last few decades.
 - Early example is that of the Windows Registry allowing the system/applications to store data in a “key-value” structure. Here, key is a unique identifier or a unique path to the value.



Working of Key-Value Databases

- A key-value store associates a value with a key that is used to keep track of the object.
 - This is like a dictionary/array/map object that is stored in a persistent way and managed by a Database Management System (DBMS).
- Here, compact, efficient index structures are used to quickly and reliably locate a value by its key
 - Ideal for systems that need to be able to find and retrieve data in constant time.

Examples of Key-Value Pair



Source: Amazon

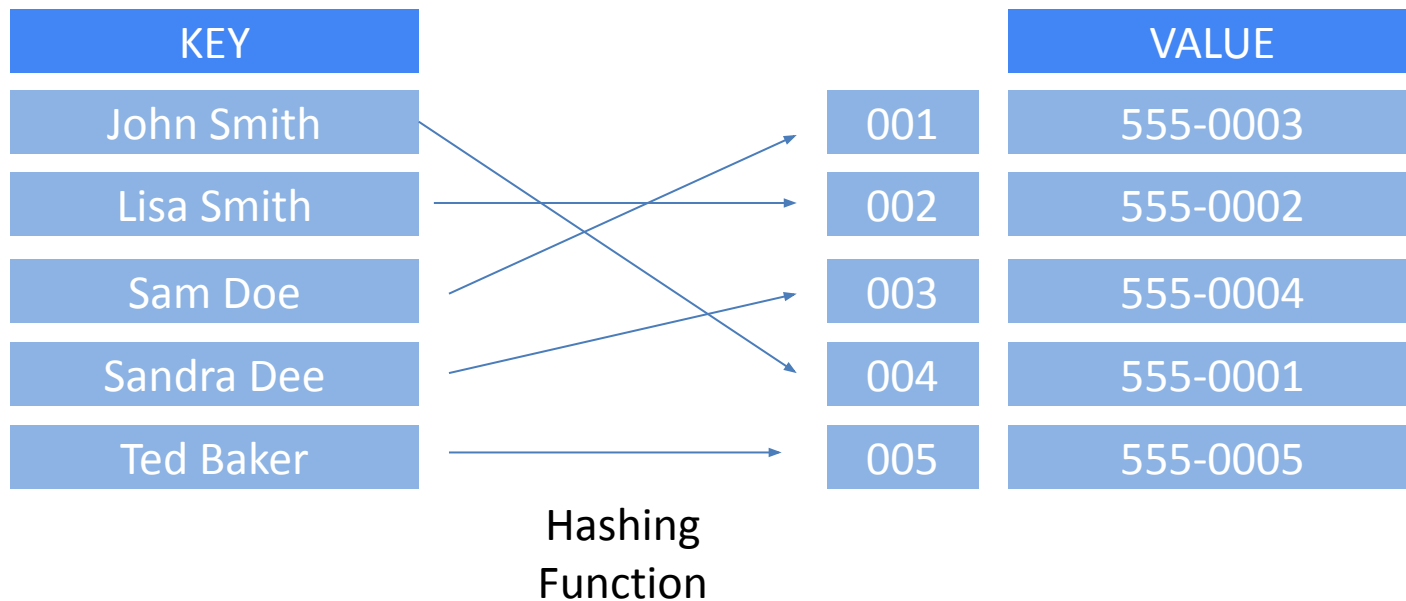
Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Source: Wikipedia

Key-Value Pair

KEY	VALUE
John Smith	555-0001
Lisa Smith	555-0002
Sam Doe	555-0003
Sandra Dee	555-0004
Ted Baker	555-0005

Key-Value Pair



Key-Value Database Features

- Allows programs or users of programs to retrieve data by keys that point to some stored value.
- Their simplicity allows for them to be extended and optimized in numerous ways.
 - Retrieving a value (if there is one) stored and associated with a given key
 - Deleting the value (if there is one) stored and associated with a given key
 - Setting, updating, and replacing the value (if there is one) associated with a given key

Key-Value Database Advantages

- Efficient and compact data structure to access data in a simple form of a key-value fetch/update/remove.
- Extensible and Optimizable in numerous ways.
- Creation of in-memory data stores for fast storing and retrieval of data.

Key-Value Database Use Cases

- Real time random data access, e.g., user session attributes in an online application such as gaming or finance.
- Caching mechanism for frequently accessed data or configuration based on keys.
- Application is designed on simple key-based queries.

Example Use Cases

- Session store
 - A session-oriented application starts a session when a user logs in and is active until the user logs out or the session times out.
 - During this period, the application stores all session-related data in a key-value database.
- Shopping cart
 - An e-commerce website receives billions of orders in seconds.
 - Website relies on key-value database to handle the scaling of large amounts of data and extremely high volumes of state changes.

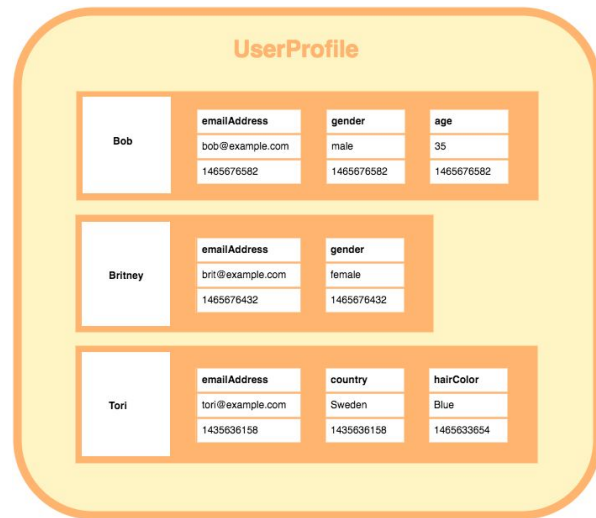
Cloud Key-Value Databases

- Google Cloud
 - Cloud Datastore
 - Cloud Bigtable.
- Amazon AWS
 - DynamoDB
- Microsoft Azure
 - Cosmos DB
 - Table Storage

Columnar Database

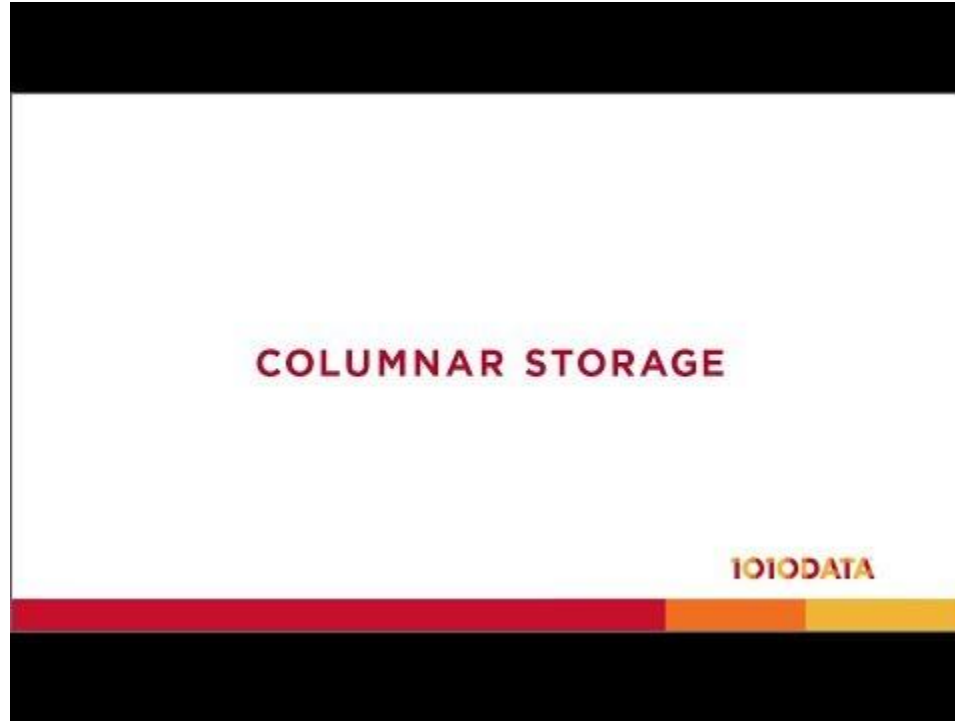
Columnar Database

- Key/value data stores that structure data storage into collections of related columns called column families.
- Store each column family in a separate partition, with same key.
- An application can read a single column family without reading through all of the data for an entity.



Source:
<https://database.guide/what-is-a-column-store-database/>

Column Database



Source: <https://www.youtube.com/watch?v=YQR2RLW3R5s>

Column Database

Relational tables are optimized for reading in records from disk drives

RowId	EmpId	LastName	FirstName	Phone
001	1111	Smith	John	555-0001
002	2222	Lisa	Smith	555-0002
003	3333	Doe	Sam	555-0003
004	4444	Dee	Sandra	555-0004
005	5555	Baker	Ted	555-0005

Column Database

Relational tables are optimized for reading in records from disk drives

RowId	EmpId	LastName	FirstName	Phone
001	1111	Smith	John	555-0001
002	2222	Lisa	Smith	555-0002
003	3333	Doe	Sam	555-0003
004	4444	Dee	Sandra	555-0004
005	5555	Baker	Ted	555-0005

```
1111:001,2222:002,3333:003,4444:004,5555:005;  
Smith:001,Lisa:002,Doe:003,Dee:004,Baker:005;  
John:001,Smith:002,Sam:003,Sandra:005,Ted:005;  
555-0001:001,555-0002:002,555-0003:003,555-0004:004,555-0005:005;
```

Column DBs are optimized for reading in column data from disk drives for performing analysis

Columnar Database for Analytics

- While a relational database stores data in rows and reads data row by row, a column store is organized as a set of columns.
 - For analytics on a small number of columns, columns can be read directly without consuming memory with the unwanted data.
- Columns are often of the same type and benefit from more efficient compression, making reads even faster.
 - Columnar databases can quickly aggregate the value of a given column

Columnar Database Disadvantages

- While columnar databases are great for analytics, however,
 - the structure makes it very difficult for them to be strongly consistent as writes of all the columns require multiple write events on disk.
- Relational databases don't suffer from this problem as row data is written contiguously to disk.

Cloud Columnar Databases

- Google Cloud
 - BigQuery
- Amazon AWS
 - AWS Redshift
- Microsoft Azure
 - Azure Synapse

Document Database

Document Database

- A document database is a database that stores information in documents.
 - Also known as a document-oriented database or a document store.
- Document databases use flexible documents instead of storing data in fixed rows and columns.
- Document databases are the most popular alternative to tabular, relational databases.

Document Database

- Key/value DBs in which the values are documents.
- Can query on non-key fields and define secondary indexes for faster querying.
- Suitable for applications that retrieve data based on complex criteria (beyond the value of the document key).

Document 1

```
{
  "id": "1",
  "name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
}
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

```
{
  "id": "3",
  "fullName": {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

Source:

<https://lennilobel.wordpress.com/2015/06/01/relational-databases-vs-nosql-document-databases/>

Documents

- A document is a record in a document database.
 - A document typically stores information about one object and any of its related metadata.
- Documents store data in key-value pairs.
 - Values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects.
- Documents can be stored in formats like JSON, BSON, and XML.

```
{
  "_id": 1,
  "first_name": "Tom",
  "email": "tom@example.com",
  "cell": "765-555-5555",
  "likes": [
    "fashion",
    "spas",
    "shopping"
  ],
  "businesses": [
    {
      "name": "Entertainment 1080",
      "partner": "Jean",
      "status": "Bankrupt",
      "date_founded": {
        "$date": "2012-05-19T04:00:00Z"
      }
    },
    {
      "name": "Swag for Tweens",
      "date_founded": {
        "$date": "2012-11-01T04:00:00Z"
      }
    }
  ]
}
```

Collections

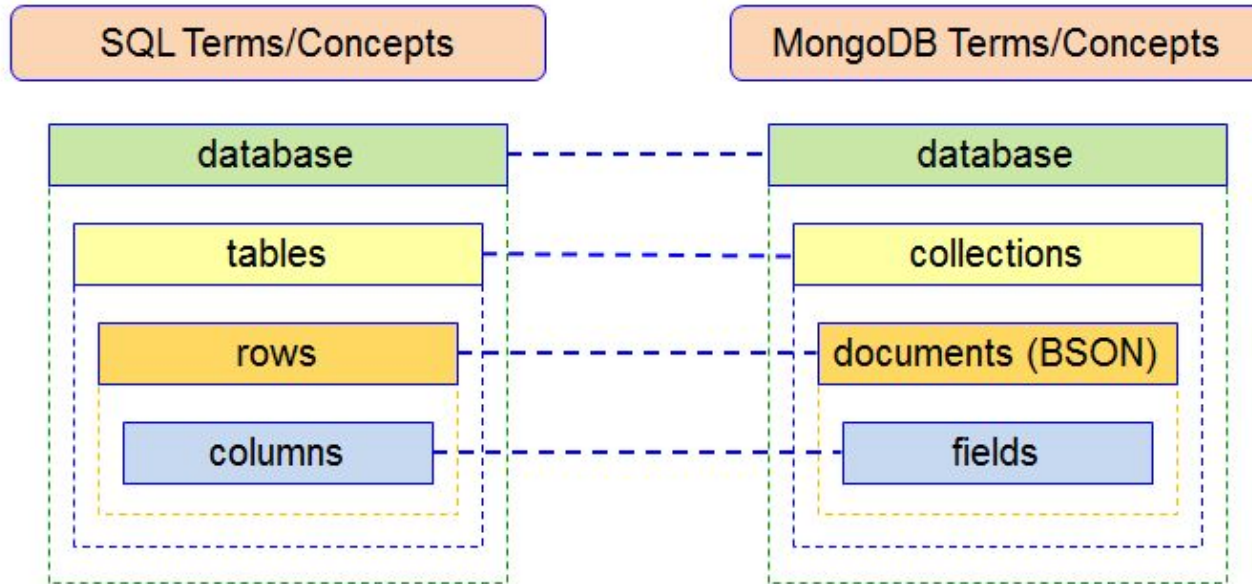
- A collection is a group of documents.
 - Collections typically store documents that have similar contents.
- Not all documents in a collection are required to have the same fields, as document databases have a flexible schema.
- Some document databases provide schema validation, so the schema can optionally be locked down when needed.

Collections

- Additional documents are added to the users collection in order to store information about other users.
 - In the figure on the right, a document provides information about Donna to be added to the users collection.

```
{
  "_id": 2,
  "first_name": "Donna",
  "email": "donna@example.com",
  "spouse": "Joe",
  "likes": [
    "spas",
    "shopping",
    "live tweeting"
  ],
  "businesses": [
    {
      "name": "Castle Realty",
      "status": "Thriving",
      "date_founded": {
        "$date": "2013-11-21T04:00:00Z"
      }
    }
  ]
}
```

Index compare b/w MongoDB and SQL



Source: <http://sql-vs-nosql.blogspot.com/2013/11/indexes-comparison-mongodb-vs-mssqlserver.html>

Document Database Features

- Document model
 - Data is stored in documents.
 - Documents map to objects in most popular programming languages, which allows developers to rapidly develop their applications.
- Flexible schema
 - Document databases have a flexible schema
 - Not all documents in a collection need to have the same fields.

Document Database Features

- Distributed and resilient
 - Document databases are distributed, which allows for horizontal scaling and data distribution.
 - Document databases provide resiliency through replication.
- Querying through an API or query language
 - Document databases have an API or query language that allows developers to execute the CRUD operations.
 - Developers have the ability to query for documents based on unique identifiers or field values.

Difference from Relational Databases

- Data Model Intuitiveness
 - Documents map to the objects in code, so they are much more natural to work with.
 - There is no need to decompose data across tables or run expensive joins.
 - Data that is accessed together is stored together, so developers have less code to write and end users get higher performance.
- JSON Document Ubiquity
 - JSON documents are lightweight, language-independent, and human-readable.
 - Documents are a superset of all other data models so developers can structure data in the way their applications need — rich objects, key-value pairs, tables, geospatial and time-series data, or the nodes and edges of a graph.

Difference from Relational Databases

- Schema Flexibility

- A document schema is dynamic and self-describing, so developers don't need to first pre-define it in the database.
- Fields can vary from document to document.
- Developers can modify the structure at any time, avoiding disruptive schema migrations.

- Document Simplicity

- Working with data in documents is easier and more intuitive than working with data in tables.
- Documents map to data structures in most popular programming languages.
- Developers don't have to worry about manually splitting related data across multiple tables when storing it or joining it back together when retrieving it.

Difference from Relational Databases

```
{
  "_id": 1,
  "first_name": "Tom",
  "email": "tom@example.com",
  "cell": "765-555-5555",
  "likes": [
    "fashion",
    "spas",
    "shopping"
  ],
  "businesses": [
    {
      "name": "Entertainment 1080",
      "partner": "Jean",
      "status": "Bankrupt",
      "date_founded": {
        "$date": "2012-05-19T04:00:00Z"
      }
    },
    {
      "name": "Swag for Tweens",
      "date_founded": {
        "$date": "2012-11-01T04:00:00Z"
      }
    }
  ]
}
```

Users

ID	first_name	email	cell
1	Tom	tom@example.com	765-555-5555

Likes

ID	user_id	like
10	1	fashion
11	1	spas
12	1	shopping

Businesses

ID	user_id	name	partner	status	date_founded
20	1	Entertainment 1080	Jean	Bankrupt	2011-05-19
21	1	Swag for Tweens	NULL	NULL	2012-11-01

Document Database Strengths

- The document model is ubiquitous, intuitive, and enables rapid software development.
- The flexible schema allows for the data model to change as an application's requirements change.
- Document databases have rich APIs and query languages that allow developers to easily interact with their data.
- Document databases are distributed (allowing for horizontal scaling as well as global data distribution) and resilient.

Document Database Weaknesses

- There is a difficult transition from relational database to document databases.
- Transactional processing may not be as efficient at relational databases.
- Document databases do not support multi-document ACID transactions.

Document Database Use Cases

- Single view or data hub
- Customer data management and personalization
- Product catalogs and content management
- Payment processing
- Mobile apps
- Operational analytics and Real-time analytics

Extending Relational Databases with Documents?

- Proprietary Extensions
 - Working with documents means using custom, vendor-specific SQL functions which will not be familiar to most developers.
- Primitive Data Handling
 - Presenting JSON data as simple strings and numbers rather than the rich data types makes computing, comparing, and sorting data complex and error prone.

Why not just use JSON in a relational database?

- Poor Data Quality & Rigid Tables
 - Relational databases offer little to validate the schema of documents which means no way to apply quality controls against JSON data.
 - Still need to define a schema for regular tabular data with the overhead that comes when tables are altered as application features evolve.
- Low Performance
 - Most relational databases do not maintain statistics on JSON data, preventing the query planner from optimizing queries against documents, and tuning queries.

Cloud Document Databases

- Google Cloud
 - Firestore
- Amazon AWS
 - AWS DocumentDB
- Microsoft Azure
 - Azure Cosmos DB
- IBM Cloud
 - Cloudant

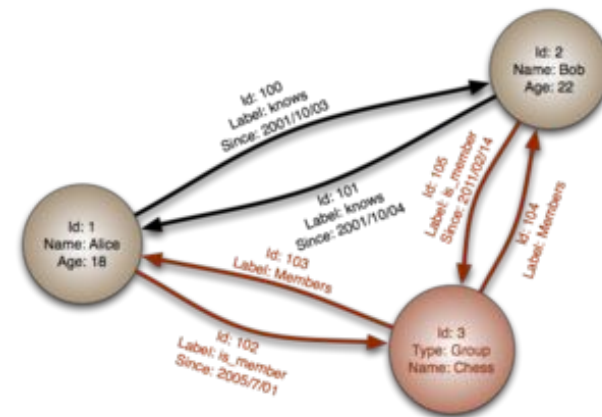
Graph Database

Graph Databases

- A graph database focuses on the relationship between data elements.
 - Each element is stored as a node.
 - The connections between elements are called links or relationships.
- A graph database is optimized to capture and search the connections between data elements
 - Overcomes the overhead associated with JOINing multiple tables in SQL.

Graph Databases

- Data schema consists of nodes, edges and properties to model the relationship between objects.
- Can efficiently perform queries that traverse the network of objects and the relationships between them.

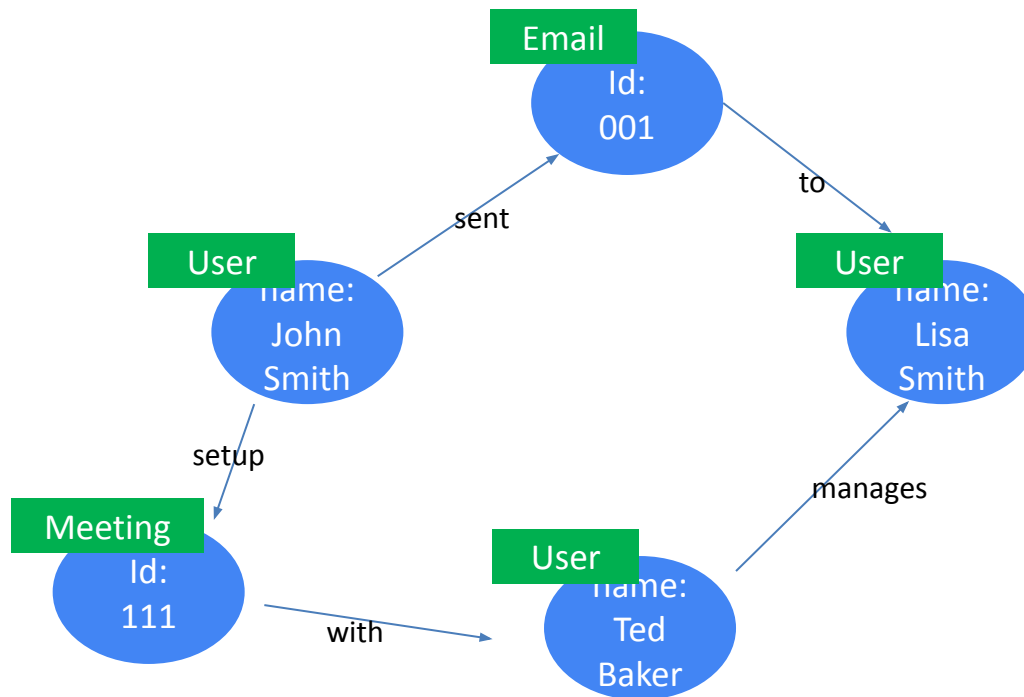


Source: Wikipedia

Difference from Relational Databases

- In a graph database, connections are first-class elements of the database, stored directly.
 - In relational databases, links are implied, using data to express the relationships.

Graph Databases



Graph Database Advantages

- To solve many-to-many relationship problems.
- To generate insight from the existing data and not just manage larger volumes of data.
- When the relationships between data elements are important.
- When application needs low latency with large-scale datasets.

Graph Database Disadvantages

- Very few real-world business systems can survive solely on graph queries.
- As a result graph databases are usually run alongside other more traditional databases.

Example Uses Cases

- Social networks, logistics, and spatial data.
- Fraud detection and analytics.
- Product-recommendation engines.
- Identity and access management.

Cloud Document Databases

- Google Cloud
 - Neo4j
- Amazon AWS
 - AWS Neptune
- Microsoft Azure
 - Azure Cosmos DB
- IBM Cloud
 - Db2 Graph

Recap



Source: <https://www.youtube.com/watch?v=W2Z7fbCLSTw>

Thank You!