System Design

Database

Key-value Databases

Key-value databases, also known as *key-value stores*, work by storing and managing *associative arrays*. An associative array, also known as a *dictionary* or *hash table*, consists of a collection of key-value pairs in which a key serves as a unique identifier to retrieve an associated value. Values can be anything from simple objects, like integers or strings, to more complex objects, like JSON structures.

Operational Database Model	Example DBMSs
Key-value store	Redis, MemcacheDB
Columnar database	Cassandra, Apache HBase
Document store	MongoDB, Couchbase
Graph database	OrientDB, Neo4j

Database	Description
Redis	An in-memory data store used as a database, cache, or message broker, Redis supports a variety of data structures, ranging from strings to bitmaps, streams, and spatial indexes.
Memcached	A general-purpose memory object caching system frequently used to speed up data-driven websites and applications by caching data and objects in memory.
Riak	A distributed key-value database with advanced local and multi-cluster replication.

Columnar Databases

Columnar databases, sometimes called *column-oriented databases*, are database systems that store data in columns. This may seem similar to traditional relational databases, but rather than grouping columns together into tables, each column is stored in a separate file or region in the system's storage.

Database	Description
Apache Cassandra	A column store designed to maximize scalability, availability, and performance.
Apache HBase	A distributed database that supports structured storage for large amounts of data and is designed to work with the Hadoop software library.
ClickHouse	A fault tolerant DBMS that supports real time generation of analytical data and SQL queries.

Document-oriented Databases

Document-oriented databases, or *document stores*, are NoSQL databases that store data in the form of documents. Document stores are a type of key-value store: each document has a unique identifier — its key — and the document itself serves as the value.

Database	Description
MongoDB	A general purpose, distributed document store, MongoDB is the world's most widely used document-oriented database at the time of this writing.
Couchbase	Originally known as Membase, a JSON-based, Memcached-compatible document-based data store. A *multi-model* database, Couchbase can also function as a key-value store.
Apache CouchDB	A project of the Apache Software Foundation, CouchDB stores data as JSON documents and uses JavaScript as its query language.

Graph Databases

Graph databases can be thought of as a subcategory of the document store model, in that they store data in documents and don't insist that data adhere to a predefined schema. The difference, though, is that graph databases add an extra layer to the document model by highlighting the relationships between individual documents.

To better grasp the concept of graph databases, it's important to understand the following terms:

- **Node**: A *node* is a representation of an individual entity tracked by a graph database. It is more or less equivalent to the concept of a *record* or *row* in a relational database or a *document* in a document store. For example, in a graph database of music recording artists, a node might represent a single performer or band.
- **Property**: A *property* is relevant information related to individual nodes. Building on our recording artist example, some properties might be "vocalist," "jazz," or "platinum-selling artist," depending on what information is relevant to the database.
- **Edge**: Also known as a *graph* or *relationship*, an *edge* is the representation of how two nodes are related, and is a key concept of graph databases that differentiates them from RDBMSs and document stores. Edges can be *directed* or *undirected*.
    - **Undirected**: In an undirected graph, the edges between nodes exist just to show a connection between them. In this case, edges can be thought of as "two-way" relationships — there's no implied difference between how one node relates to the other.
    - **Directed**: In a directed graph, edges can have different meanings based on which direction the relationship originates from. In this case, edges are "one-way" relationships. For example, a directed graph database might specify a relationship from Sammy to the Seaweeds showing that Sammy produced an album for the group, but might not show an equivalent relationship from The Seaweeds to Sammy.

Database	Description
Neo4j	An ACID-compliant DBMS with native graph storage and processing. As of this writing, Neo4j is the most popular graph database in the world.
ArangoDB	Not exclusively a graph database, ArangoDB is a multi-model database that unites the graph, document, and key-value data models in one DBMS. It features AQL (a native SQL-like query language), full-text search, and a ranking engine.
OrientDB	Another multi-model database, OrientDB supports the graph, document, key-value, and object models. It supports SQL queries and ACID transactions.

Relational database management systems (Oracle, MySQL, MS Server, PostgreSQL)

Relational databases store data sets as “relations”: tables with rows and columns where all information is stored as a value of a specific cell. Data in an RDBMS is managed using SQL. Though there are different implementations, SQL is standardized and provides a level of predictability and utility.

Strengths

Relational databases excel at handling highly structured data and provide support for **ACID (Atomicity, Consistency, Isolation, and Durability)** transactions. Data is easily stored and retrieved using SQL queries. The structure can be scaled up quickly because adding data without modifying existing data is simple.

Creating limits on what certain user types can access or modify is built into the structure of an RDBMS. Because of this, relational databases are well-suited to applications that require tiered access. For example, customers could view their accounts while agents could both view and make necessary changes.

Weaknesses

The biggest weakness of relational databases is the mirror of their biggest strength. As good as they are at handling structured data, they have a hard time with unstructured data. Representing real world entities in context is difficult in the bounds of an RDBMS. “Sliced” data has to be reassembled from tables into something more readable, and speed can be negatively impacted. The fixed schema doesn’t react well to change, either.

Cost is a consideration with relational databases. They tend to be more expensive to set up and grow. Horizontal scaling, or scaling by adding more servers, is usually both faster and more economical than vertical scaling, which involves adding more resources to a server

Use a relational database for:

- Situations where data integrity is absolutely paramount (i.e., for financial applications, defense and security, and private health information)
- Highly structured data
- Automation of internal processes

Document store (MongoDB, Couchbase)

A document store is a nonrelational database that stores data in JSON, BSON, or XML documents. They feature a flexible schema. Unlike SQL databases, where users must declare a table's schema before inserting data, document stores don't enforce document structure. Documents can contain any data desired. They have key-value pairs but also embed attribute metadata to make querying easier.

Strengths

Document stores are very flexible. They handle semistructured and unstructured data well. Users don't need to know during set-up what types of data will be stored, so this is a good choice when it isn't clear in advance what sort of data will be incoming.

Users can create their desired structure in a particular document without affecting all documents. Schema can be modified without causing downtime, which leads to high availability. Write speed is generally fast, as well.

Besides flexibility, developers like document stores because they're easy to scale horizontally. The sharding necessary for horizontal scaling is much more intuitive than with relational databases, so document stores scale out fast and efficiently.

Weaknesses

Document databases sacrifice ACID compliance for flexibility. Also, while querying can be done in a document it's not possible across documents.

Use a document database for:

- Unstructured or semistructured data
- Content management
- In-depth data analysis
- Rapid prototyping

Key-value store (Redis, Memcached)

A key-value store is a type of nonrelational database where each value is associated with a specific key. It's also known as an associative array.

The "key" is a unique identifier associated only with the value. Keys can be anything allowed by the DBMS. In Redis, for example, keys man be any binary sequence up to 512MB.

"Values" are stored as blobs and don't need predefined schema. They can take nearly any form: numbers, strings, counters, JSON, XML, HTML, PHP, binaries, images, short videos, lists, and even another key-value pair encapsulated in an object. Some DBMSs allow for the data type to be specified, but it isn't mandatory.

Strengths

This style of database has a lot of positives. It's incredibly flexible, able to handle a very wide array of data types easily. Keys are used to go straight to the value with no index searching or joins, so performance is high. Portability is another benefit: key-value stores can be moved from one system to another without rewriting code. Finally, they're highly horizontally scalable and have lower operating costs overall.

Weaknesses

Flexibility comes at a price. It's impossible to query values, because they're stored as a blob and can only be returned as such. This makes it hard to do reporting or edit parts of values. Not all objects are easy to model as key-value pairs, either.

Use a key-value store for:

- Recommendations
- User profiles and settings
- Unstructured data such as product reviews or blog comments
- Session management at scale
- Data that will be accessed frequently but not often updated

Wide-column store (Cassandra, HBase)

Wide-column stores, also called column stores or extensible record stores, are dynamic column-oriented nonrelational databases. They're sometimes seen as a type of key-value store but have attributes of traditional relational databases as well.

Wide-column stores use the concept of a keyspace instead of schemas. A keyspace encompasses column families (similar to tables but more flexible in structure), each of which contains multiple rows with distinct columns. Each row doesn't need to have the same number or type of column. A timestamp determines the most recent version of data.

Strengths

This type of database has some benefits of both relational and nonrelational databases. It deals better with both structured and semistructured data than other nonrelational databases, and it's easier to update. Compared to relational databases, it's more horizontally scalable and faster at scale.

Columnar databases compress better than row-based systems. Also, large data sets are simple to explore. Wide-column stores are particularly good at aggregation queries, for example.

Weaknesses

Writes are expensive in the small. While updating is easy to do in bulk, uploading and updating individual records is hard. Plus, wide-column stores are slower than relational databases when handling transactions.

Use a wide-column store for:

- Big data analytics where speed is important
- Data warehousing on big data
- Large scale projects (this database style is not a good tool for average transactional applications)

Search engine (Elasticsearch)

It may seem strange to include search engines in an article about database types. However, Elasticsearch has seen increased popularity in this sphere as developers look for innovative ways to cut down search lag. Elastisearch is a nonrelational, document-based data storage and retrieval solution specifically arranged and optimized for the storage and rapid retrieval of data.

Strengths

Elastisearch is very scalable. It features flexible schema and fast retrieval of records, with advanced search options including full text search, suggestions, and complex search expressions.

One of the most interesting search features is stemming. Stemming analyzes the root form of a word to find relevant records even when another form is used. For example, a user searching an employment database for “paying jobs” would also find positions tagged as “paid” and “pay.”

Weaknesses

Elastisearch is used more as an intermediary or supplementary store than a primary database. It has low durability and poor security. There’s no innate authentication or access control. Also, Elastisearch doesn’t support transactions.

Use a search engine like Elastisearch for:

- Improving user experience with faster search results
- Logging

DB

Key Value	Redis
	Memcached
	Riak
Columnar	Cassandra
	HBase
	ClickHouse
Graph	Neo4j
	Arango
	Orient
RRDBMS	SQL
	MySQL
	Oracle
	Postgrase
Document	Mango
	CouchBase
Search Engine	ElasticSearch

Key Value

Strength		Flexible
		Wide Array
		High Perfomrmance
		No index
		Portability
		Horizantally Scalable
Weakness		Store as Blob
		Report and Edit is hard
		Not all objects can be model as key value
Usecase		recommendation
		User profile setting
		Unstructured data such as Product review
		Session management at scale
		Data accessed frequently but not often update

Columnar

Strength	Benfits of relational an non-relational Structured and Semi Structured data Compared to RDBMS more horizontally scalable Compress Better Simple explore for large Data set Good for aggregation query update is easy un bulk
Weakness	Write are expensive Upload and update undividuak record is hard Slow for transaction compared to RDBMS
Usecase	Big Data Analytics where speed is important Data ware housing in Big Data Large Scale projects Not good for average Transactional App

Graph

Strength

- Flexibility: Change and extend for additional attributes and objects
- Search: Fast queries when you are looking for relationships between nodes
- Indexing: Graph databases are naturally indexed by relationships
- Fast to traverse nodes
- Can represent multiple dimensions

Weakness

- Inappropriate for transactional information
- Harder to get support
- Not optimized for large-volume analytics queries typical of data warehousing.

Usecase

- Machine Learning
- Fraud detection
- Real-time recommendation engines
- **Regulatory Compliance**
- **Identity and Access Management**
- **Supply Chain Transparency**

RDB

Strength

- Structured Data
- ACID Transaction
- Scale up structure, because adding data is easy
- Limit User Access

Weakness

- Un Structured Data
- Fixed Schema
- Expensive to setup and grow
- Not Vertical Scale

Usecase

- Integrity is important (financial ,defense, private health)
- Highly structured data
- Automation of internal process

Document

Strength	UnStructured and Semi Structured Data
	No nead to know type of data
	Create desired structure
	Modify Schema with out DownTme
	Write Speed is Fast
	Easy to scale Horizantaly
Weakness	NO ACID
	Query on 1 Document not across all Document
Usecase	UnStructured and Semi Structured Data
	Content Management
	In Depth Data Analysis
	Rapid Prototyping

Search Engine

Strength	Very Scalable
	Full text Search & Suggestion
	Stemming (Root of the word and related words)
	Used for Supplemtary not Primary DB
Weakness	Poor Security
	Low Durablity
	No Access Control
	No Transaction
Usecase	Improve User Experince with Faster Search
	Logging