Data Architecture and Database Technologies (I)

Dr. Karl-Heinrich Anders

NOSQL DATABASE SYSTEMS

DOCUMENT DATABASE

What is Document Database?

- A document database, also known as a document-oriented database, is a type of NoSQL database that stores data in the form of documents, rather than in tables with rows and columns like a traditional relational database.
- These documents can be in a variety of formats, such as JSON, BSON, or XML.
- They often include nested data structures, which can make it easier to store and query complex data.
- Document databases are well suited for storing semi-structured data and are often used in web and mobile applications.
- In a document database, data is stored in the form of documents, which can include nested data structures.
  - Each document can have a unique structure and can contain different fields.
  - This is in contrast to a relational database, where data is stored in tables with a fixed schema.

Document Database Design I

1. **Data modeling:**
    1. This involves deciding how to organize and structure the data in your database. In document databases, data is stored in the form of documents, which can include nested data structures. The structure of the documents should be designed to support the queries and use cases of your application.
2. **Indexing:**
    1. Indexing allows for fast searching and querying of the data. In document databases, you can index specific fields in the documents to improve query performance. It's important to carefully consider which fields to index, as too many indexes can lead to decreased performance.
3. **Sharding:**
    1. Sharding is the process of distributing the data across multiple servers to improve performance and scalability. In a document database, sharding can be done based on the value of a specific field in the documents, such as the user ID.

Document Database Design II

4. **Replication:**
   - Replication involves creating multiple copies of the data to improve reliability and availability. In document databases, replication can be done in a variety of ways, such as master-slave replication or peer-to-peer replication.

5. **Data validation:**
   - Data validation involves specifying rules for how data is stored in the database. Some document databases support data validation, which can help ensure data consistency and integrity.

6. **Security:**
   - Document databases should have a robust security mechanism to protect against unauthorized access to the data. It's important to consider how to secure data at rest and in transit, as well as how to authenticate and authorize users.

7. **Backup and Recovery:**
   - Document databases should have a robust backup and recovery plan in place to ensure that data is not lost in case of a disaster.

Document Database Design III

- Remember that the design of a document database will depend on the specific needs of your application and the features of the document database you are using!

Core Operations Document Database I

1. **CRUD (Create, Read, Update, and Delete) operations:**
   - These are the basic operations for creating, reading, updating, and deleting documents in the database.
2. **Indexing:**
   - Document databases often support indexing of documents, which allows for fast searching and querying of the data.
3. **Aggregation:**
   - The ability to group and summarize data based on certain criteria, similar to SQL's GROUP BY and aggregate functions.
4. **Sharding:**
   - Distributes the data across multiple servers to improve performance and scalability.

Core Operations Document Database II

5. **Replication:**
   - Allows for multiple copies of the data to be stored on different servers for improved reliability and availability.
6. **Data validation:**
   - Some document databases support data validation, which allows you to specify rules for how data is stored in the database.
7. **Transactions:**
   - Document databases usually support atomic transaction operations for multiple documents, ensuring data consistency.

Use Cases of Document Database I

1. **Content management systems:**
   - Document databases can be used to store and manage large amounts of content, such as articles, blog posts, and images.
2. **E-commerce and retail:**
   - Document databases can be used to store product information, customer data, and order history.
3. **Social media:**
   - Document databases can be used to store user profiles, comments, and other data associated with social media platforms.
4. **Gaming:**
   - Document databases can be used to store player data, game progress, and leaderboards.

Use Cases of Document Database II

5. **IoT:**
   - Document databases can be used to store sensor data and other information generated by Internet of Things (IoT) devices.
6. **Logging and monitoring:**
   - Document databases can be used to store and analyze log data for debugging and performance monitoring.
7. **User data:**
   - Document databases can be used to store user data, such as preferences and personal information, for web and mobile applications.
8. **Financial services:**
   - Document databases can be used to store financial data, such as transactions and account information.

Popular Document Databases I

1. **MongoDB**:
   - One of the most widely used document databases, MongoDB is known for its scalability, performance, and ease of use.
2. **Couchbase**:
   - A document database that supports both SQL and NoSQL querying. It also provides a built-in caching layer for improved performance.
3. **RavenDB**:
   - A document database that focuses on performance and ease of use, with built-in support for full-text search and indexing.
4. **Amazon DocumentDB**:
   - A managed document database service provided by Amazon Web Services, it is compatible with MongoDB and can be used to easily scale and replicate data.

Popular Document Databases II

5. **Cloud Firestore**:
   - A document-oriented NoSQL database service provided by Google Cloud Platform, it can be used to easily store, sync, and query data for web, mobile, and IoT applications.
6. **CosmosDB**:
   - A multi-model, globally distributed database service provided by Microsoft Azure, it supports document, key-value, graph, and column-family data models.
7. **CouchDB**:
   - An open-source document-oriented database that uses a document-based data model, it also supports ACID semantics, map-reduce and incremental replication.

Use of Document Database I

1. **When you have semi-structured or unstructured data:** Document databases are well suited for storing data that does not have a fixed schema, such as JSON or XML documents.
2. **When you need to store hierarchical or nested data:** Document databases can store nested data structures, which can make it easier to store and query complex data.
3. **When you need a high level of scalability:** Document databases are often horizontally scalable, which means that they can handle a large amount of data and a high number of concurrent users.
4. **When you need fast read and write performance:** Document databases are designed for fast read and write performance, which can be especially important in high-traffic web and mobile applications.

Use of Document Database II

5. **When you need a flexible data model:**
   - Document databases allow you to store data in a flexible, schema-less format, which can make it easier to adapt to changing requirements.
6. **When you are working with polyglot persistence:**
   - Document databases can work well alongside other types of databases, such as key-value or graph databases, to support different use cases and workloads in your application.

- However, it's worth noting that document databases may not be the best choice for all types of applications.
- For example, if you need to perform complex, multi-table joins or if you need to enforce strict data integrity constraints, a relational database might be a better choice.

Document Databases and UPDATE or DELETE anomalies!

- It's always complicated to know how to design your data. An important consideration is how often your data changes versus how often you read it.
  - Normalization will provide an update efficient data representation.
  - Denormalization will make data reading efficient.

- Should you store accounts preferences inside each user's document?
  - **Generally yes. If ..**
    - it's only relevant to this user.
    - it probably won't change a lot either.

- It's probably a good idea to keep a user's address inside your document too.
  - If it will not be updated regularly, so you should not penalize every read for the relevance of this information.

Document Databases and UPDATE or DELETE anomalies!

- Here are some guidelines to help you when considering this issue:
  - **Embedding:**
    - You have small subdocuments
    - Your data does **not** need to change regularly
    - You don't need immediate consistency (not up-to-date)
    - Your documents grow by a small amount
    - You need this data to perform a second query
    - You want faster reads
  - **Referencing:**
    - You have large subdocuments
    - Your data changes frequently
    - You need your data to be up-to-date
    - Your documents grow by a large amount
    - Your data is often excluded from your results
    - You want faster writes

Advantages of Document Databases

1. **Flexible schema:** Document databases allow for a flexible, schema-less data model, which can make it easier to adapt to changing requirements and handle unstructured or semi-structured data.
2. **High performance:** Document databases are designed for fast read and write performance, which can be especially important in high-traffic web and mobile applications.
3. **Scalability:** Document databases are often horizontally scalable, which means that they can handle a large amount of data and a high number of concurrent users.
4. **Ease of use:** Many document databases have a simple and intuitive API, which can make them easy to use and integrate into existing applications.
5. **Nested data structures:** Document databases can store nested data structures, which can make it easier to store and query complex data.

Disadvantages of Document Databases

1. **Limited querying capabilities:** Compared to relational databases, document databases may have limited querying capabilities, particularly when it comes to performing complex, multi-table joins.
2. **Data consistency:** In a schema-less model, it may be more difficult to enforce data consistency and integrity.
3. **Limited ACID support:** Some document databases may have limited support for ACID (Atomicity, Consistency, Isolation, Durability) transactions, which can make it difficult to ensure data consistency in certain situations.
4. **Data validation:** Some document databases may not have built-in support for data validation, which can make it difficult to ensure data consistency and integrity.
5. **Backup and Recovery:** Backup and Recovery process may be more complex than traditional relational databases.

KEY VALUE DATABASE

What is Key-Value Database?

- A key-value database, also known as a **key-value store**, is a type of NoSQL database that stores data as a collection of key-value pairs.
- Each key is a unique identifier that is used to retrieve the corresponding value. The value can be any type of data, such as a string, number, or object.
- Key-value databases are designed for high performance and scalability, and are often used in situations where the data does not require complex relationships or joins.
- They are well suited for storing data that can be easily partitioned, such as caching data or session data.
- Key-value databases are simple and easy to use, but they may not be as suitable for complex queries or data relationships as other types of databases such as document or relational databases.

Key-Value Database Use Cases I

1. **Caching:**
   - Key-value databases are often used as a caching layer in front of a more persistent data store to improve the read performance of an application.
2. **Session Management:**
   - Key-value databases can be used to store user session data, such as login status, shopping cart contents, or other temporary data.
3. **Real-time analytics:**
   - Key-value databases can be used to store and process large amounts of data in real-time, such as sensor data, social media feeds, and IoT data.
4. **Gaming leaderboards:**
   - Key-value databases can be used to store and retrieve high scores and rankings for online games.

Key-Value Database Use Cases II

5. **Distributed systems**:
   - Key-value databases can be used to store data that is distributed across multiple machines, such as distributed hash tables or distributed key-value stores.
6. **Content Management Systems:**
   - Key-value databases can be used to store and retrieve content such as images, videos, and audio.
7. **Product catalogs:**
   - Key-value databases can be used to store and retrieve product information, such as descriptions, prices, and inventory levels.

- The key-value databases are not suitable for all types of use cases.
- For example, if you need to perform complex queries or you need to enforce data integrity constraints, a relational database or a document database may be a better choice.

Popular Key Value Databases I

- **Redis:** An open-source, in-memory data structure store that can be used as a database, cache, and message broker.
- **Riak:** An open-source, distributed key-value database that is designed for high availability and scalability.
- **Aerospike:** An open-source, distributed key-value database that is designed for high performance and scalability, and is often used in real-time big data applications.
- **LevelDB:** An open-source, key-value storage library that is designed for high performance and low-level storage.
- **Berkeley DB:** A family of embedded key-value databases that are designed for high performance and low-level storage, and are often used in embedded systems and mobile devices.

Popular Key Value Databases II

- **Memcached:** An open-source, in-memory key-value cache that is often used to speed up dynamic web applications by reducing the number of times an external data source must be read.
- **RocksDB:** An open-source, persistent key-value store that is based on LevelDB and is optimized for storage on flash and hard disk drives.
- **Amazon DynamoDB:** A fully managed, highly available, key-value database service that is part of the Amazon Web Services (AWS) ecosystem.
- **Azure Cosmos DB:** A globally distributed, multi-model database service that supports key-value, document, graph, and column-family data models.
- **Google Cloud Bigtable:** A fully managed, high-performance, wide-column NoSQL key-value store that is part of the Google Cloud Platform (GCP) ecosystem.

When to use Key Value Databases

1. **You need high performance and scalability:** Key-value databases are designed for high performance and scalability and are well suited for situations where the data does not require complex relationships or joins.
2. **You are working with large amounts of unstructured or semi-structured data:** Key-value databases can store various types of data, from simple strings to complex objects.
3. **You are working with data that can be easily partitioned:** Key-value databases can handle data that can be easily partitioned, such as caching data or session data.
4. **You need to store data that is distributed across multiple machines:** Key-value databases can be used to store data that is distributed across multiple machines, such as distributed hash tables or distributed key-value stores.
5. **You need a simple and easy-to-use database:** Key-value databases have a simple and intuitive data model and are easy to use and integrate into existing applications.
6. **You need a database that can handle high-volume, high-velocity data:** Key-value databases are built to handle high-volume, high-velocity data and a high number of concurrent users.

Use cases for Key Value Databases

1. **Caching:** Key-value databases are often used as a caching layer in front of a more persistent data store to improve the read performance of an application.
2. **Session Management:** Key-value databases can be used to store user session data, such as login status, shopping cart contents, or other temporary data.
3. **Real-time analytics:** Key-value databases can be used to store and process large amounts of data in real-time, such as sensor data, social media feeds, and IoT data.
4. **Gaming leaderboards:** Key-value databases can be used to store and retrieve high scores and rankings for online games.
5. **Distributed systems:** Key-value databases can be used to store data that is distributed across multiple machines, such as distributed hash tables or distributed key-value stores.
6. **Content Management Systems:** Key-value databases can be used to store and retrieve content such as images, videos, and audio.
7. **Product catalogs:** Key-value databases can be used to store and retrieve product information, such as descriptions, prices, and inventory levels.

Key-Value Store

- What are the advantages of key-value store?
  - The major advantages of key-value stores are **scalability, speed, and flexibility**.
  - Key-value stores handle size well and are good at processing a constant stream of read/write operations.
  - This property makes it highly scalable. Key-value stores scale out by implementing partitions, replication, and auto-recovery.
- What is a major drawback to a key-value store? Limitations of Key-Value Stores
  - These are general limitations: Key-value stores are not optimized and require a parser for multiple values.
  - They are only optimized for a single key and value.
  - These kinds of stores cannot filter out the value fields

Advantages of Key Value Databases

1. **High performance:** Key-value databases are designed for fast read and write performance, which can be especially important in high-traffic web and mobile applications.
2. **Simplicity:** Key-value databases have a simple and intuitive data model, which can make them easy to use and integrate into existing applications.
3. **Scalability:** Key-value databases are often horizontally scalable, which means that they can handle a large amount of data and a high number of concurrent users.
4. **Flexibility:** Key-value databases can store various types of data, from simple strings to complex objects.
5. **Distributed Systems:** Key-value databases can be used to store data that is distributed across multiple machines, such as distributed hash tables or distributed key-value stores.

Disadvantages of Key Value Databases

1. **Limited querying capabilities:** Key-value databases may have limited querying capabilities, particularly when it comes to performing complex queries or joins.
2. **Data modeling:** Key-value databases have a simple data model that is based on key-value pairs and may not support complex data structures or relationships.
3. **Data validation:** Some key-value databases may not have built-in support for data validation, which can make it difficult to ensure data consistency and integrity.
4. **Limited ACID support:** Some key-value databases may have limited support for ACID (Atomicity, Consistency, Isolation, Durability) transactions, which can make it difficult to ensure data consistency in certain situations.
5. **Lack of support for advanced features:** Some key-value databases may lack support for advanced features such as full-text search or geospatial indexing.

WIDE COLUMN DATABASE

What is a Wide Column Database?

- A wide-column database (also known as a column-family database) is a type of NoSQL database that stores data in a column-family format.
  - A column family is a collection of rows and columns, where each row has a unique key and each column has a name, value, and timestamp.
  - A wide-column database organizes data in a way that allows for a more efficient storage of data and faster query performance.
- In a wide-column database, data is organized into column families, where each column family represents a group of related data.
  - This allows for a more flexible and efficient data model, as each column family can have its own set of columns and can be optimized for different types of queries.
- Wide-column databases are well suited for data warehousing and business intelligence applications, where large amounts of data need to be analyzed and aggregated.
  - They are often used for analytical queries, such as aggregation and data mining, and are highly optimized for handling large datasets.
- Some examples of wide-column databases include Apache Cassandra and Apache Hbase.

Wide Column Database Use Cases I

- **Data Warehousing**
  - Wide-column databases are optimized for data warehousing and business intelligence applications, where large amounts of data need to be analyzed and aggregated.
  - They are often used for analytical queries, such as aggregation and data mining.
- **OLAP (Online Analytical Processing)**
  - Wide-column databases can handle complex and large queries and calculations which is required in OLAP systems, they are good for multi-dimensional analysis and reporting.
- **Real-time analytics**
  - Wide-column databases can handle high-volume, high-velocity data and a high number of concurrent users, which makes them a choice for real-time analytics applications.
- **Big data**
  - Wide-column databases can handle large datasets and provide efficient storage and retrieval of data, therefore, can be used for big data applications.

Wide Column Database Use Cases II

- **Cloud-based analytics**
  - Wide-column databases can be easily scaled to handle large amounts of data and are designed for high availability, this makes them suitable for cloud-based analytics applications.
- **IoT**
  - Wide-column databases can handle a high number of writes and reads and can be used for storing and processing IoT data.
- **Handling high write throughput**
  - Wide-column databases are optimized to handle high write throughput and low latency and are suitable for use cases where there is a high amount of writes such as gaming and e-commerce.
- **Wide-column databases are not suitable for all types of use cases.**
  - For example, if you need to perform complex data modeling or you need to enforce data integrity constraints, a relational database may be a better choice.

Wide Column Database vs Key-Value I

1. **Data Modeling:**
   - Key-value databases store data as simple key-value pairs, while wide-column databases store data in a column-family format. This allows for a more flexible and efficient data model in wide-column databases, as each column family can have its own set of columns and can be optimized for different types of queries.
2. **Query Capabilities:**
   - Key-value databases have simple querying capabilities and are optimized for simple key-value lookups. Wide-column databases have more advanced querying capabilities and are optimized for analytical queries, such as aggregation and data mining.
3. **Data Retrieval:**
   - Key-value databases are optimized for key-based data retrieval and are typically faster at retrieving data based on a specific key. Wide-column databases are optimized for column-based data retrieval and are typically faster at retrieving large amounts of data based on multiple criteria.

Wide Column Database vs Key-Value II

4. **Scalability:**
   - Both key-value and wide-column databases are horizontally scalable, but wide-column databases are often better for handling large amounts of data and a high number of concurrent users.

5. **Use Cases:**
   - The use case of key-value databases are caching, session management and real-time data retrieval.
   - While wide-column databases use cases are data warehousing, business intelligence, real-time analytics and big data.

- ✓ Both key-value and wide-column databases are NoSQL databases, they have different data models, query capabilities, and use cases.
- ✓ Key-value databases are good for simple key-value lookups, while wide-column databases are good for analytical queries and handling large amounts of data.

Wide Column Database vs Document Database I

1. **Data Modeling:**
   - Document databases store data in a semi-structured format, usually in JSON or BSON format, allowing for a flexible and dynamic data model.
   - While wide-column databases store data in a column-family format, which allows for a more efficient storage of data and faster query performance.
2. **Query Capabilities:**
   - Both wide-column and document databases have advanced querying capabilities, but document databases often provide more powerful and expressive query languages, that allow for complex queries and data manipulation.
3. **Data Retrieval:**
   - Document databases are optimized for document-based data retrieval, allowing for fast and efficient retrieval of a single document or a collection of documents.
   - Wide-column databases are optimized for column-based data retrieval, and are typically faster at retrieving large amounts of data based on multiple criteria.

Wide Column Database vs Document Database II

1. **Scalability:**
   - Both wide-column and document databases are horizontally scalable, but the scalability model may differ between the two types of databases.
2. **Use Cases:**
   - Document databases are well suited for use cases such as content management, real-time analytics and e-commerce.
   - While wide-column databases are well suited for use cases such as data warehousing, business intelligence and big data.

- ✓ Both wide-column and document databases are NoSQL databases, they have different data models, query capabilities and use cases.
- ✓ Document databases are good for flexible and dynamic data modeling, while wide-column databases are good for analytical queries and handling large amounts of data.

Popular Wide Column Databases

- **Cassandra**
- **Hbase**
- **Microsoft Azure Cosmos DB**
- **ScyllaDB**
- **Microsoft Azure Table Storage**
- **Accumulo**
- **BigTable**
- **Amazon Keyspaces**
- **Azure Table Storage**

Advantages of Wide Column Databases I

1. **High performance:**
   - Wide-column databases are optimized for analytical queries and are designed for fast query performance, which can be especially important in data warehousing and business intelligence applications.
2. **Flexible and efficient data model:**
   - Wide-column databases store data in a column-family format, which allows for a more flexible and efficient data model, as each column family can have its own set of columns and can be optimized for different types of queries.
3. **Scalability:**
   - Wide-column databases are often horizontally scalable, which means that they can handle large amounts of data and a high number of concurrent users.

Advantages of Wide Column Databases II

4. **Distributed Systems:**
   - Wide-column databases can be distributed across multiple machines, which allows for high availability and scalability.
5. **Handling high write throughput:**
   - Wide-column databases are optimized to handle high write throughput and low latency and are suitable for use cases where there is a high amount of writes such as gaming and e-commerce.

Disadvantages of Wide Column Databases I

1. **High performance:**
   - Wide-column databases are optimized for analytical queries and are designed for fast query performance, which can be especially important in data warehousing and business intelligence applications.
2. **Flexible and efficient data model:**
   - Wide-column databases store data in a column-family format, which allows for a more flexible and efficient data model, as each column family can have its own set of columns and can be optimized for different types of queries.
3. **Scalability:**
   - Wide-column databases are often horizontally scalable, which means that they can handle large amounts of data and a high number of concurrent users.

Disadvantages of Wide-column databases II

4. **Distributed Systems:**
   - Wide-column databases can be distributed across multiple machines, which allows for high availability and scalability.
5. **Handling high write throughput:**
   - Wide-column databases are optimized to handle high write throughput and low latency and are suitable for use cases where there is a high amount of writes such as gaming and e-commerce.

GRAPH DATABASES

What is a Graph Database?

- In a graph database, the data is stored as a set of vertices and edges, with each vertex representing an entity (such as a person or a business) and each edge representing a relationship between two vertices (such as a friendship or a business partnership).
- The graph structure allows for flexible and efficient querying, as it allows for traversing relationships between entities in various ways.
- Graph databases are particularly useful for storing and querying data that has complex relationships, such as social networks, recommendation engines, and fraud detection systems.
- They are also often used in areas such as bioinformatics and supply chain management, where the data has a complex, interconnected structure.

Graph Database Use Cases I

- **Social networks**
  - Graph databases can be used to store and query data about relationships between people, such as friendships, family relationships, and professional connections.
  - This can be used to build social networking platforms, recommendation engines, and other applications.
- **Fraud detection**
  - Graph databases can be used to identify patterns of fraudulent activity by analyzing the relationships between entities such as individuals, businesses, and transactions.
- **Recommendation engines**
  - Graph databases can be used to store and query data about users and their interactions with products or content.
  - This can be used to build recommendation engines that suggest products or content to users based on their interests and past behavior.

Graph Database Use Cases II

- **Supply chain management**
  - Graph databases can be used to store and query data about the relationships between different entities in a supply chain, such as suppliers, manufacturers, and retailers.
  - This can be used to optimize logistics and supply chain management.
- **Bioinformatics**
  - Graph databases can be used to store and query data about the relationships between different biological entities, such as genes, proteins, and diseases.
  - This can be used to study the relationships between different biological processes and to develop new drugs and treatments.

Popular Graph Databases I

- **Neo4j**
  - Neo4j is a widely used open-source graph database that is optimized for storing and querying large amounts of data.
  - It is written in Java and supports ACID transactions, making it suitable for use in enterprise applications.
- **JanusGraph**
  - JanusGraph is an open-source, distributed graph database that is built on top of Apache Cassandra and Elasticsearch.
  - It is designed to handle large-scale graph data and is optimized for high performance and scalability.
- **Amazon Neptune**
  - Amazon Neptune is a fully managed graph database service that is optimized for storing and querying graph data.
  - It is designed to be easy to use and is backed by the reliability and security of the Amazon Web Services (AWS) cloud platform.

Popular Graph Databases II

- **OrientDB**
  - OrientDB is an open-source, multi-model database that supports graph, document, key-value, and object data models.
  - It is designed to be scalable and efficient, and it supports ACID transactions and SQL-like query language.
- **ArangoDB**
  - ArangoDB is an open-source, multi-model database that supports graph, document, and key-value data models.
  - It is designed to be flexible and easy to use, and it supports ACID transactions and a powerful query language.

Graph Database Properties

- **ACID transactions**
  - Many graph databases support ACID (Atomicity, Consistency, Isolation, Durability) transactions, which ensure that data is stored and accessed in a consistent and reliable manner.
  - This is important for applications that require high levels of data integrity and reliability.
- **High performance**
  - Graph databases are optimized for high performance and can handle a large number of queries and updates in real-time.
  - This makes them suitable for use in high-traffic applications.

Graph Database Properties

- **Flexible data model**
  - Graph databases use a flexible data model that allows for the representation of complex relationships between entities.
  - This makes it easy to store and query data that has many different types of relationships and connections.
- **Efficient querying**
  - Graph databases are optimized for efficient querying of data, particularly when it comes to traversing relationships between entities.
  - This makes it easy to find and retrieve data about specific entities and their relationships with other entities.
- **Scalability**
  - Graph databases are designed to scale well as the size of the data increases.
  - This makes them suitable for storing and querying large amounts of data.

How do queries work in a Graph Database?

- There is a wide range of query possibilities that can be exploited when using a graph database. The main reason for this is that there is no uniform query language.
- Unlike traditional models, graph databases count on **special algorithms** to fulfill their essential function: simplifying and speeding up complicated data queries.
- Two of the most important algorithms are the **depth-first search** and the **breadth-first search**: The depth-first searches for the next node below in each case, while the breadth-first search moves from layer to layer.
- The algorithms make it possible to find graph patterns as well as direct and indirect adjacent nodes.
- Other algorithms make it possible to calculate the shortest path between two nodes, and to identify cliques (subsets of nodes) and hotspots (information that is particularly highly interconnected).
- One of the strengths of the graph database is that **relationships are stored in the database itself**, so they don’t need to be calculated in the query. This results in a high performance speed, even for complicated queries.

The advantages and disadvantages of Graph Databases I

- The strength of a database can be measured using four principal factors: **Integrity, performance, efficiency** and **scalability**.
  - The data query ought to become quicker and simpler – the main purpose of graph databases can be roughly summarized in this way.
  - Where relational databases reach their capacity limits, the graph-based model is particularly agile, because complexity and the quantity of data don't negatively influence the query process in this model.

- Also, with the graph database model, **real facts can be stored in a natural way**.
  - The structure used is very similar to human thinking, and this is why the links are so clear.
  - Graph databases are not a complete solution, though.
    - They are **limited**, for example, where **scalability** is concerned.
    - As they are principally designed for one-tier architecture, growth represents a (mathematical) challenge.
    - Plus, there is still no uniform query language.

The advantages and disadvantages of Graph Databases II

- Graph databases should not be considered generally to be an absolute better replacement for conventional databases.
  - Relational structures remain entirely reasonable standard models, guaranteeing high data integrity and stability, and permitting flexible scalability.
  - As so often, the same applies here: It all depends on the intended purpose!

Advantages	**Disadvantages**
Query speed only dependent on the number of concrete relationships, and not on the amount of data	Difficult to scale, as designed as one-tier architecture
Results in real time	No uniform query language
Clear and manageable representation of relationships	
Flexible and agile structures	