MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT5137 – Advanced Database Technology

Week 12 – Guidelines for Selecting a Database

Semester 2, 2021

Developed by:
Dr. Agnes Haryanto
Agnes.Haryanto@monash.edu

Agenda

1. Guidelines for Selecting a Database
   - ❖ Choosing a NoSQL Database
   - ❖ Using NoSQL and Relational Databases Together

Which one to Choose?

- **Relational databases**
  - PostgreSQL, MySQL, and Microsoft SQL Server
- **Key-value databases**
  - Redis, Riak, and Oracle BerkeleyDB
- **Document databases**
  - MongoDB, CouchDB, and CouchBase
- **Column family databases**
  - Cassandra and HBase
- **Graph databases**
  - Neo4j and Titan

Choosing a NoSQL Database

Choosing a NoSQL Database

Relational Database	NoSQL Database
1. In relational database design, the structure and relations of entities drives design.	1. In NoSQL database design, performance is more important than preserving the relational model.
2. The relational model emerged for pragmatic reasons, that is, data anomalies and difficulty reusing existing databases for new applications.	2. NoSQL databases emerged for pragmatic reasons, specifically, the inability to scale to meet growing demands for high volumes of read and write operations. In exchange for improved read and write performance, you may lose other features of relational databases, such as immediate consistency and ACID transactions.

Choosing a NoSQL Database

- Queries have driven the design of data models.
  - Because queries describe how data will be used.
  - Queries are also a good starting point for understanding how well various NoSQL databases will meet your needs.
- You will also need to understand other factors, such as
  - The volume of reads and writes
  - Tolerance for inconsistent data in replicas
  - The nature of relations between entities and how that affects query patterns
  - Availability and disaster recovery requirements
  - The need for flexibility in data models
  - Latency requirements

Use Cases and Criteria for Selecting Document-Oriented Databases

- Document databases are designed for flexibility.
- If an application requires the ability to store varying attributes along with large amounts of data, then document databases are a good option.
  - For example, to represent products in a relational database, a modeler may use a table for common attributes and additional tables for each subtype of product to store attributes used only in the subtype of product.
  - Document databases can handle this situation easily.

Use Cases and Criteria for Selecting Document-Oriented Databases

文档数据库提供嵌入式文档，这对于非规范化很有用。

- Document databases provide for embedded documents, which are useful for denormalizing.
- Instead of storing data in different tables, data that is frequently queried together is stored together in the same document.
- Document databases improve on the query capabilities of key-value databases with indexing and the ability to filter documents based on attributes in the document.
- Document databases are probably the most popular of the NoSQL databases because of their flexibility, performance, and ease of use.

Use Cases and Criteria for Selecting Document-Oriented Databases

- These databases are well suited to a number of use cases, including:
  - Back-end support for websites with high volumes of reads and writes
  - Managing data types with variable attributes, such as products
  - Tracking variable types of metadata
  - Applications that use JSON data structures
  - Applications benefiting from denormalization by embedding structures within structures

Use Cases and Criteria for Selecting Column-Oriented Databases

- Column family databases are designed for large volumes of data, read and write performance, and high availability.
  - Google introduced BigTable to address the needs of its services.
  - Facebook developed Cassandra to back its Inbox Search service.
- These database management systems run on clusters of multiple servers.
- If your data is small enough to run with a single server, then a column family database is probably more than you need—consider a document or key-value database instead.

Use Cases and Criteria for Selecting Column-Oriented Databases

- Column family databases are well suited for use with:
  - Applications that require the ability to always write to the database
  - Applications that are geographically distributed over multiple data centers
  - Applications that can tolerate some short-term inconsistency in replicas
  - Applications with dynamic fields
  - Applications with the potential for truly large volumes of data, such as hundreds of terabytes

Use Cases and Criteria for Selecting Column-Oriented Databases

- Several areas can use this kind of Big Data processing capability, such as:
  - Security analytics using network traffic and log data mode
  - Big Science, such as bioinformatics using genetic and proteomic data
  - Stock market analysis using trade data
  - Web-scale applications such as search
  - Social network services

Use Cases and Criteria for Selecting Graph Databases

- Problem domains that lend themselves to representations as networks of connected entities are well suited for graph databases.
- One way to assess the usefulness of a graph database is to determine if instances of entities have relations to other instances of entities.
  - Now consider examples mentioned in the discussion of graph databases, such as highways connecting cities, proteins interacting with other proteins, and employees working with other employees. In all of these cases, there is some type of connection, link, or direct relationship between two instances of entities.

适合表示为连接实体网络的问题域非常适合图数据库。

评估图形数据库有用性的一种方法是确定实体实例是否与其他实体实例有关系。

Use Cases and Criteria for Selecting Graph Databases

- Two orders in an e-commerce application probably have no connection to each other.
- They might be ordered by the same customer, but that is a shared attribute, not a connection.

no

Use Cases and Criteria for Selecting Graph Databases

- Similarly to the previous case, a game player's configuration and game state have little to do with other game players' configurations.
- Entities like these are readily modeled with key-value, document, or relational databases.

no

Use Cases and Criteria for Selecting Graph Databases

- Highways connecting cities, proteins interacting with other proteins, and employees working with other employees.
- In all of these cases, there is some type of connection, link, or direct relationship between two instances of entities.

Use Cases and Criteria for Selecting Graph Databases

- Some examples of problem domains that are well suited to graph databases, include:
  - Network and IT infrastructure management
  - Identity and access management
  - Business process management
  - Recommending products and services
  - Social networking
- From these examples, it is clear that when there is a need to model explicit relations between entities and rapidly traverse paths between entities, then graph databases are a good database option.

Using NoSQL and Relational Databases Together

- NoSQL and relational databases are complementary.
- Relational databases offer many features that protect the integrity of data and reduce the risk of data anomalies.
  - Relational databases incur operational overhead providing these features.
- In some use cases, performance is more important than ensuring immediate consistency or supporting ACID transactions. In these cases, NoSQL databases may be the better solution.
- Choosing a database is a process of choosing the right tool for the job. The more varied your set of jobs, the more varied your toolkit.

Using NoSQL and Relational Databases Together

- Relational databases will continue to support transaction processing systems and business intelligence applications.
  - Decades of work with transaction processing systems and data warehouses has led to best practices and design principles that continue to meet the needs of businesses, governments, and other organizations.
  - At the same time, these organizations are adapting to technologies that did not exist when the relational model was first formulated.
  - Customer-facing web applications, mobile services, and Big Data analytics might work well with relational databases, but in some cases they do not.
- The current technology landscape requires a variety of database technologies.

Using NoSQL and Relational Databases Together

- Just as there is no best programming language, there is no best database management system.

- There are database systems better suited to some problems than others, and the job of developers and designers is to find the best database for the requirements at hand.

Summary

- Application developers have choices about which programming language they use, which development environments they work in, and which web frameworks they deploy. They also have choices when it comes to database management systems.
  - The different types of database management systems were all developed to solve real-world problems that could not be solved as well with other types of databases.
- One of the jobs of developers and designers is to choose an appropriate database system for their applications.
- You do this by understanding your problem domain and your user requirements. Often you will have options.
- The choice of database should be driven by your needs.

References

[1] D. Sullivan, NoSQL for Mere Mortals. Michigan, USA: Addison-Wesley Professional, 1st ed., 2015.

[2] W. Lemahieu, S. V. Broucke and B. Baesens, Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data. New York, NY, USA: Cambridge University Press, 1st ed., 2018.