Relational vs Non-Relational Database Comparison

This semester we will be looking at how to use non-relational or NoSQL databases instead of traditional SQL style databases. As part of this, we need to understand some of the key differences between these databases and what this means for us when we use them.

Data Structure

SQL Databases	No-SQL
Relational databases organize data into structured tables with predefined columns and data types. Data is organized in rows and columns, making it suitable for well-defined and consistent data structures. Every record in a single table will always have the same fields and data types as the other records in the table. SQL databases are optimised with this in mind to allow for complex queries to be done efficiently due to a consistent structure.	No SQL databases come in a number of different data store types such as key : value pairs, document data stores, graphs, wide column stores and much more. Because of this, records can have varying structures, allowing for more dynamic and diverse data storage. Records can have structured, semi-structured or totally unstructured data stored in them as they do not rely on consistency of the data storage to improve performance and generally are able to reduce the overall storage capacity by doing this.

Database Schema

SQL Databases	No-SQL
Requires a predefined schema that outlines the structure of the data, specifying the tables, relationships between tables, and constraints. Changes to the schema can be challenging and may require careful planning.	Does not require a predefined schema; each record can have its own unique structure. Provides flexibility to adapt to changing data requirements without strict schema constraints. Fields can generally be added or removed without breaking the overall functionality of the database. This sometimes can mean some extra work to handle non-exiting fields in some queries. No relationships are enforced - but they still can be used.

Normalizing Data

SQL Databases	No-SQL
Emphasizes normalization to eliminate data redundancy and maintain data integrity. Helps in reducing data anomalies by breaking down tables into smaller, related entities. Values are normally only stored once and then referenced by other tables. Data is broken over multiple tables that reference each other through foreign keys. Faster to change data - normally only needed to change it in one place. Optimised more for space than speed (speed can still be improved with use of indexing)	Uses non-normalised data when it is stored in the system. Emphasizes denormalization to improve query performance by storing related data together. Allows for faster read operations for simple queries and insertions at the expense of some redundancy in the stored data. Less efficient for complex queries or when data from multiple record types are required simultaneously. Can be slower/harder to change data if replicated in multiple places

Query Language

SQL Databases	No-SQL
Utilizes SQL (Structured Query Language) for querying and manipulating data. SQL provides a standardized way to interact with relational databases regardless of vendor. Retrievals often require finding data across multiple tables for a single result. This generally means utilising joins and other more complex commands to retrieve data. The query syntax often focuses more on the output structure/format of the query rather then the steps for retrieving and filtering the data. Builds a record based upon a compilation of the related data.	Uses query languages specific to the database vendor type. For example, MongoDB's query language is a JavaScript based language which utilises a system of method calls rather than a new language. Some vendors have developed SQL-like query languages to help people transition to their systems. Queries are often more intuitive and closely aligned with the structure of the data. They focus more on how to identify and retrieve the required data before worrying about presentation. Many NoSQL languages feel more intuitive to programmers.

Performance

SQL Databases	No-SQL
High performance for complex queries and records with many relationships. Insertions take time due to heavy reliance on indexing and coordination of data. Simple insertion and retrievals still slower than NoSQL. Less suitable for high rates of data transfer to and from the system.	Allows lower processing power to still retrieve data efficiently. Faster to insert data when it is all going to the one collection. Generally faster to find all the details from a single event/record because it is all in one place. Greater efficiency when dealing with mass volumes of data where the records are simple, and relationships are not needed.

Scalability

SQL Databases	No-SQL
Generally, scales well vertically by adding more resources (CPU, RAM) to a single server. This can however hit limitations once the system needs source extremely high-performance components. Horizontal scaling is possible but can be challenging, especially for complex relational structures.	Typically scales horizontally by adding more servers to distribute the data load. Can also, scale individual server vertically for increased single machine performance. Well-suited for handling large amounts of data and high read and write loads.

Conclusion

Traditional relational databases are normally better suited moderate volume data where interactions and relationships between multiple entities are required.

NoSQL systems are better suited for high volume data where the structure is simple, and the schema needs the ability to change without disruption to the rest of the system.

Alternatively, organisations can implement a combination of both systems to allow for the complexity needs of some areas of the data but still allow for the high volume and flexibility of the data in other areas of the system.