NAME:INAMANG MONDAY EGWU

REG NO. 2020/HND/31041/CS

There are five practical differences between SQL and NoSQL:

Language

Scalability

Structure

Properties

Support and communities

1. Language

SQL has been around for over 40 years, so it is recognizable, documented, and widely-used. Safe and versatile, it’s particularly well suited for complex queries. However, SQL restricts the user to working within a predefined tabular schema, and more care must be taken to organize and understand the data before it is used.

The dynamic schemata of NoSQL databases allow representation of alternative structures, often alongside each other, encouraging greater flexibility. There is less emphasis on planning, greater freedom when adding new attributes or fields, and the possibility of varied syntax across databases. As a group, however, NoSQL languages lack the standard interface which SQL provides, so more complex queries can be difficult to execute.

Though there are many dialects of SQL, all share a common syntax and almost-identical grammar. When querying relational databases, fluency in one language translates to proficiency in most others. On the other hand, there is very little consistency between NoSQL languages, as they concern a diverse set of unrelated technologies. Many NoSQL databases have a unique data manipulation language constrained by particular structures and capabilities.

1. Scalability

Most SQL databases can be scaled vertically, by increasing the processing power of existing hardware. NoSQL databases use a master-slave architecture which scales better horizontally, with additional servers or nodes. These are useful generalizations, but it’s important to note:

SQL databases can be scaled horizontally as well, though sharding or partitioning logic is often the user’s onus and not well supported.

NoSQL technologies are diverse and while many rely on the master-slave architecture, options for scaling vertically also exist.

Savings made using more efficient data structures can overwhelm differences in scalability; most important is to understand the use case and plan accordingly.

1. Structure

SQL database schemata always represent relational, tabular data, with rules about consistency and integrity. They contain tables with columns (attributes) and rows (records), and keys have constrained logical relationships.

NoSQL databases need not stick to this format, but generally fit into one of four broad categories:

Column-oriented databases transpose row-oriented RDBMSs, allowing efficient storage of high-dimensional data and individual records with varying attributes.

Key-Value stores are dictionaries which access diverse objects with a key unique to each.

Document stores hold semi-structured data: objects which contain all of their own relevant information, and which can be completely different from each other.

Graph databases add the concept of relationships (direct links between objects) to documents, allowing rapid traversal of greatly connected data sets.

1. Properties

At a high level, SQL and NoSQL comply with separate rules for resolving transactions. RDBMSs must exhibit four “ACID” properties:

Atomicity means all transactions must succeed or fail completely. They cannot be partially-complete, even in the case of system failure.

Consistency means that at each step the database follows invariants: rules which validate and prevent corruption.

Isolation prevents concurrent transactions from affecting each other. Transactions must result in the same final state as if they were run sequentially, even if they were run in parallel.

Durability makes transactions final. Even system failure cannot roll-back the effects of a successful transaction.

NoSQL technologies adhere to the “CAP” theorem, which says that in any distributed database, only two of the following properties can be guaranteed at once:

Consistency: Every request receives the most recent result, or an error. (Note this is different than in ACID)

Availability: Every request has a non-error result, regardless of how recent that result is.

Partition tolerance: Any delays or losses between nodes will not interrupt the system’s operation.

1. Support and communities

SQL databases represent massive communities, stable codebases, and proven standards. Multitudes of examples are posted online and experts are available to support those new to programming relational data.

NoSQL technologies are being adopted quickly, but communities remain smaller and more fractured. However, many SQL languages are proprietary or associated with large single-vendors, while NoSQL communities benefit from open systems and concerted commitment to onboarding users.

SQL is available to most major platforms, from operating systems to architectures and programming languages. Compatibility varies more widely for NoSQL, and dependencies need to be investigated more carefully.