NoSQL Databases

What are NoSQL Databases?

What are NoSQL Databases?

- "NoSQL" is a bit of a misnomer

What are NoSQL Databases?

- "NoSQL" is a bit of a misnomer
  - Post-hoc, techies claim they mean "not only SQL"

What are NoSQL Databases?

- "NoSQL" is a bit of a misnomer
  - Post-hoc, techies claim they mean "not only SQL"
- Non-tabular: Not organized in tables

What are NoSQL Databases?

- "NoSQL" is a bit of a misnomer
  - Post-hoc, techies claim they mean "not only SQL"
- Non-tabular: Not organized in tables
- Non-relational: Does not focus on relationships between data

History

- Relational databases goal:
  - accurately and efficiently store data *in the least amount of storage space possible*
- This is part of the reason we avoid storing duplicate data as much as possible
- Servers used to be very costly to operate

- Storage is much less costly than it used to be
- Volume of data to process has increased dramatically
    - Way more points of data collected, processed, and archived by application / software

- Modern data is unstructured and can change frequently
- Modern data is often stored in the "cloud"
    - aka centralized servers, rather than local / institutional level servers

Types of NoSQL Databases

Types of NoSQL Databases

- NoSQL Databases come in different types

Types of NoSQL Databases

- NoSQL Databases come in different types
- Each of these types is best suited for particular contexts

Types of NoSQL Databases

- NoSQL Databases come in different types
- Each of these types is best suited for particular contexts
- Need to understand the requirements, shape, and use of your data to choose the best fit

Document Databases

- Stores data in the form of standardized document formats
  - JSON, BSON, XML
- Documents can be nested
  - A JSON object can have an attribute that is another, nested, JSON object
- Typically, one document will contain all relevant data for an entity

Document Databases

- Document Database structure works around the same format of data used by many webapps
- Less translation needed to use DB data in application
- Structure is very flexible* / can be changed easily
    - while this may be true in very early development... less practical on a live application with multiple moving pieces

Document Databases Uses

- Content management
- Monitoring web / mobile apps

Key-value Stores

- Every element is stored as a key-value pair with an attribute name ("key") and value
- Keys are the unique identifier for each value
- Values can be anything: number, string, or another nested key-value pair

Key-value Stores

- Does this structure remind of us anything we've interacted with before in programming?

Key-value Stores

- Does this structure remind of us anything we've interacted with before in programming?
  - Dictionaries

Key-value Stores

- Does this structure remind of us anything we've interacted with before in programming?
  - Dictionaries
  - Or, relational table where a record is associated with a key

Key-value Stores

- Best suited for looking up by *key value*
- Not as good at filtering / querying by values that are not the key
    - WHERE clauses don't really work in this context

Columnar Databases

- Relational DBs store data in rows and process data row-by-row
- Treats sets of columns as one entity
- Define which combinations of columns we care about
    - "Column family"
- Basically re-orients how a relational database is handled

Columnar Databases

- Denormalized storage of data
- Groups of data are focused on subsets of columns
  - not normalized tables with FKs
- Rows do not need to have a value in every column

Columnar Databases

- Lends to analytics
- Makes aggregation queries much faster
- Good at compression, efficient storage
- Easy to expand; just build out more columns and define relevant column subsets

Graph Databases

- Data is stored as it is in graph theory
- Each element is a node
- Connections between elements are considered the edges, and referred to as "links" / "relationships"

Graph Databases

- Useful for modeling
  - Networked data, recommendation engines
- Built around relationships between individual entities
  - Relationships between entities are the same importance as the entities themselves
  - (In Relational DB, they are secondary)

Graph Databases

- Querying can be complex, but often because the data being modeled is complex
- Queries best traverse across relationships
  - "Find all of the x's that have a k relationship with y"

Goals of NoSQL

Flexible Data Modeling

- NoSQL DBs have flexible schemas that are easy to change
  - Unlike relational DBs, where a ERM must be constantly maintained and improved

Horizontal Scaling

- Can store across nodes in a server, allowing for incremental growth
- Data is partitioned and placed on multiple machines (commodity servers)
- This is referred to as **sharding** data '
    - (creating shards that can later be pieced back together)

In a Relational DB, because of the normalization and compartmentalizing of data, you don't want to partition data because it could affect certain relationships between tables

Simple Querying

- Because NoSQL databases consider storage to be cheap, we can just store related data next to each other
- Don't need to perform complicated joins to bring data related data together

Comparing NoSQL and Relational

NoSQL is Best For:

NoSQL is Best For:

- Data that is:
  - Large
  - Unrelated (or at least loosely related)
  - Rapidly changing

NoSQL is Best For:

- Data that is:
  - Large
  - Unrelated (or at least loosely related)
  - Rapidly changing
- Schemas:
  - The data is not strictly bound to a strong schema
  - The software / application architecture provides a schema

NoSQL Applications

- Apps where fast access to data is prioritized
- Always-on applications
  - Mobile apps
  - Real-time analytics
  - Content management
  - IoT

Relational is Best For:

Relational is Best For:

- Data that:
  - Is very relational
  - Can be robustly defined in advance
    - (distinct, separable, contiguous)

Relational is Best For:

- Data that:
  - Is very relational
  - Can be robustly defined in advance
    - (distinct, separable, contiguous)
- Schemas:
  - Well maintained, thoroughly defined
  - Consistency is prioritized

Relational Applications

- Legacy applications
  - Existing applications in which an older dataset must be maintained
    - Accounting / Finance / Banking
- Applications that require complex queries / filtering, or need to be able to operate on multiple rows
  - Inventory management
  - Transaction management

References

- "*NoSQL Database - What is NoSQL?*" - Microsoft Azure
- "*What is NoSQL?*" - MongoDB
- *"Introduction to NoSQL"* - GeeksforGeeks