Chapter 2 - Design Scalable systems

Type	Designing Data-Intensive Applicaitons
Content	Principles
Created time	@September 29, 2022 11:08 AM
Last edited time	@October 4, 2022 4:27 PM
Resources	
💼 Projects	

Data Models

💡 This chapter focuses on Data Models and Query languages. Data models are the most important part of developing software as they impact not only how it is written but also about how we think to solve problems.

✏️ **Data Model:** An abstract model that organises elements of data and standardizes how they relate to one another and to the properties of real-world entities.

- Most applications are built on top of other data models, as a hierarchy, the question that is raised is: How is it represented in terms of the next-lower layer?
- Example. Showing the layers of Data Models:
    - High level apps: An app developer would look at the real world and try and model it in terms of objects or data structures and APIs that manipulate those data structures. Those are specific to your application.

    - Storage of data structures: When you store those data structures that model the real world, you express them in terms of a general-purpose data model, such as JSON or XML, tables in a relational database or graph models.
    - Representing the models: The engineers who built your database software decided on a way of representing that JSON/XML in terms of bytes in memory, on disk, or on a network. This representation allows the data to be searched, queried and manipulated in various ways
    - Lower level: Hardware engineers have figured out how to represent bytes in terms of electrical currents, pulses of lights and more.

The idea is that each layer hides the complexity of the layer below it by providing a clean data model.

General-purpose data models for storage and querying:

- Relational model
- Document model
- Graph-based models

Relational Model versus Document Model

Relational Model

- The most well known relational model is SQL: Data is organised into relations (tables in SQL), where each relation is an unordered collection of tuples (rows in SQL).
- The goal of the relational model is to hide the implementation of representing complex relationships behind a cleaner interface.

A major issue with relational models, is that an awkward translation layer is required between the objects in the application code and the database model of tables, rows and columns. This disconnect between the models is called impedance mistmatch.

✏️ **Object-relational impedance mismatch:** Is a set of conceptual and technical difficulties that are often encountered when a RDBMS is being by served by an object oriented application program. Objects or class definitions must be mapped to database tables defined a relational schema.

📺 https://www.youtube.com/watch?v=wg-NCF5KXNk&ab_channel=MavenEdge

Video notes
Object and relational models don't work well together because:
- Object models are more granular
- Identity mismatch
- Inheritance cannot be represented in relational models
- Associations are non-directional in relational models
- Data navigation

- Developers believe the JSON model reduces the impedance mismatch between app code and the storage layer
    - JSON representation has better locality than the multi-table schema.
    - [x] ~~What is data locality?~~
    - [x] ~~What is Normalization and De-Normalization~~

Document Model

The hierarchical model is used by document databases and is very similar to the JSON model. It represents all data as a tree or records nested within records.

- [x] ~~What are query optimizers?~~

The main arguments of the document data model are schema flexibility, better performance due to locality and that for some apps it is closer to the data structure used by the apps. The relational model on the other hand, provides better support for joins, and many-to-one, and many-to-many relationships.

- The lack of a schema for document databases, mean that when reading, clients have no guarantees as to what fields the document may contain.

> ⚠️ Document databases are usually called scemaless, which is misleading, as the data usually assumes some kind of strucutre. A more accurate term is schema-on-read, the structure is implicit and interpreted when the data is read). Schema-on-write is what is used for relational models, the schema is explicit and the database ensure conformity.
>
> - Schema on read is similar to dynamic type checking in programming
> - Schema on write is similar to static type checking

- The schema-on-read approach is advantageous if the item structure are not homogeneous.
- A document is usually stored as a single string, encoded as JSON, XML or binary variant.
- The locality advantage is only applicable if a large part of the document is needed at the same time.

Relational and Document based models are becoming more similar over time which is a great thing. If a database can handle document like data and perform relational queries on it, this could best fit the needs of a company.

NoSQL

- A more popular data model today, it's growth has been driven through a need for greater scalability than relational databases can easily achieve, frustration with the restrictiveness of relational schemas, and a desire for a more dynamic and expressive data model.
  - ☐ What does dynamic and expressive mean?

Query languages for Data

Imperative vs Declarative language

💡 When relational model was introduced, it included a new way of querying. SQL is a declarative query language, whereas other models used imperative language.

Imperative language: Tells the computer to perform certain operations in a certain order. Step through the code line by line.

Declarative language: Speficify the pattern of data you want, what conditions must be met, and how you want the data to be transformed. It will then optimize the best way to achieve that goal.

- Declarative language hides the implementation details of the database engine, making it possible for the database system to improve performance without impacting queries.
    - No re-training involved; Under the hood performance improvements. Abstracted away complexity.
- Imperative language which shows you how the query should be performed, relies on ordering. That limits its functionality.
- Imperative code is also hard to implement across several cores/nodes because it specifies instructions in a particular order.
- Declarative only specifies the pattern of the results, not the algorithm that is implemented to find them. CSS and XSL are declarative languages for specifying styling.

MapReduce

💡 MapReduce is a programming model for processing large amounts of data in bulk across many machines. It is neither declarative nor imperative, but somewhere in between. The logic of the query is expressed with snippets of code, which are called by the processing framework.

Based on the map(collect) and reduce(fold/aggregate) functions that exist in many functional programming languages.

- MapReduce is a fairly low-level programming model for distributed execution on a cluster of machines.
- The map and reduce functions are restricted in what they are allowed to do. They must be pure functions: they can only use the data passed to them as input, cannot perform additional database queries, and they must not have side effects.
    - These restrictions allow the database to run the functions anywhere, in any order, and rerun on failure. ~ Hence it can be used for multicore processing.

- [ ] What is a side effect?

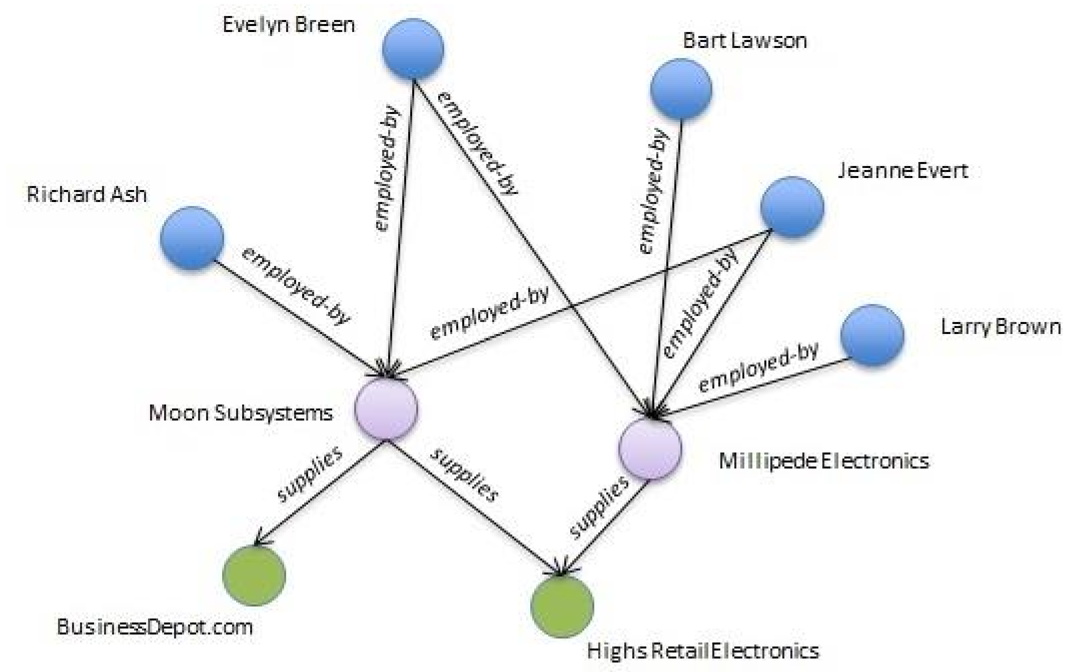
Graph-Like Data Models

💡 If many-to-many relationships are common and complex, this might not be well suited for relational models. In this case modelling your data more as a graph will be better and representing the complex relationships.

✏️ A graph consists of two kinds of objects: vertices (aka nodes, or entities) and egdes (aka relationships or arcs).

Many kinds of data can be modelled as a graph:

- Social graphs
    - nodes are people and relationships indicate which people know eachother
- Web graph
    - nodes are web pages, and relationships indicate HTML links to other pages
- Road or rail networks
    - nodes are junctions and relationships represent the roads or rail-way lines between them



Example of graph structured data

There are several ways to structure data in a graph:

- Property graphs (neo4j)
- Triple-store model

Property graphs

💡 In a property graph. Each vertex (node) consists of:

- A unique identifier
- A set of outgoing edges (relationships)
- A set of incoming edges (relationships)
- A collection of properties (key-value pairs)

Each edge (relationship) consists of:

- A unique identifier
- The vertex (node) in which the edge starts (tail vertex)
- The vertex (node) in which the edge ends (head vertex)
- A label that describes the relationship between the two vertices
- A collection of properties (key-value pairs)

- You can think of a property graph as two relational tables, one for edges, one for vertices.
- Some important aspects are:
  - Any vertex can have an edge connecting it to any other vertex
  - Given any vertex you can efficiently find it’s incoming and outgoing edges, you can traverse the graph
  - By using different labels for different types of relationships you can store several different kinds of information in a single graph, while still mainting a clean data model
- Graphs are good for evolvability, as you add more features to your application, a graph can easily be extended to accomodate changes in your application’s data structure

Cypher query language

> ✏️ Cyper is a declarative query language for property graphs, created for the Neo4j graph database

- You could use SQL queries to query graph data that is represented in a relational model, but it is very difficult as you will not know in advance which joins to use.

- [ ] What is recursive common table expressions

Triple-Stores

> 💡 The triple store is mostly the same as property graph model, using different words representing the same ideas.
>
> All information in a triple store is stored in the form of very simple three-part statements: (subject, predicate, object).

The subject in a triple is equivalent to a vertex in a graph. The object is one of two things:

1. A value in a primitive datatype, such as string or number. In that case the predicate and object of the triple are eqivalent to the key-value pair
2. Another vertex in the graph. In that case the predicate is an edge in the graph, the subject is the tail-verteix and the object is the head vertex

- [ ] Resource description framework and the semantic web
- [ ] What is turtle language

SPARQL

💡 SPARQL is a query language for triple store using the RDF data model

- [ ] CODASYL’s network model

Summary

💡 Historicall, data started out being represented as one big tree (hierarchical model), but that wasn’t good for representing many-to-many relationships, so the relational model was invented to solve that problem.

More recently some developers for that some use case don’t fit well in the relational model, so new NoSQL datastores have divered in two ways:

1. Document database targets use cases where the data comes in self contained documents and relationships between one document and another are rare
2. Graph databases go in the opposite direction where anything is potentially related to everything