Data Models and Query Languages

Arnav Gupta

January 17, 2025

Contents

1 Introduction

Most applications are built by layering data models, each layer hiding the complexity of the layers below.

Every data model embodies assumptions about how it will be used.

2 Relational Model vs Document Model

Data organized into **relations**, which are unordered collections of **tuples**.

Hides implementation detail behind a clean interface.

2.1 Birth of NoSQL

Refers to *Not Only SQL*.

NoSQL is adopted due to:

- need for greater scalability than relational DBs
- preference for FOSS over commercial DBs
- specialized query operations not well supported by relational model
- frustration with restriction of relational schemas

2.2 Object-Relational Mismatch

Impedance mismatch: disconnect between object-oriented model of application code and relational model of DB

ORM frameworks help hide mismatch.

JSON can reduce impedance mismatch, with better locality (all relevant info in one place). This is the **document** model.

2.3 Many-to-One and Many-to-Many Relationships

Using standardized lists rather than strings can enforce consistent spelling, avoid ambiguity, make updates easier, localize support, and allow for better search.

Using strings causes duplication. Removing duplication is the idea behind DB **normalization**. This is easier to do with the relational model than the document model.

2.4 Are Document Databases Repeating History?

2.4.1 Network Model

Generalization of the hierarchical model where:

- each record can have multiple parents
- links between records are not foreign keys but instead pointers that can be followed through an **access path**
  - can have multiple paths to the same record

Can change access paths, but then must rewrite code to handle new access paths. Difficult to change application's data model.

2.4.2 Relational Model

Simpler since no nested structure.

Query optimizer decides which parts of query to execute in which order, and which indexes are used.

2.4.3 Comparison to Document Databases

Document DBs are similar to hierarchical DBs, except for many-to-one and many-to-many relationships where they are like relational DBs.

2.5 Relational vs Document Databases Today

Document data model benefits:

- schema flexibility
- better performance due to locality
- closer to data structures used by the application

Relational data model benefits:

- better support for joins, many-to-one, and many-to-many relationships

2.5.1 Simpler Application Code

If data is a tree of one-to-many relationships, document model is better.

If application uses many-to-many relationships, relational mode is better.

2.5.2 Schema Flexibility

Document DBs do not enforce any schema on the data in documents.

Document DBs have **schema-on-read** (structure interpreted when data is read).

Relational DBs have **schema-on-write** (DB ensures all written data conforms to schema).

Schema-on-read can be helpful if items in the collection don't all have the same structure.

2.5.3 Data Locality

Documents are usually stored as a string, providing **storage locality**.

Locality helps if large parts of document are needed at once.

Writes replace the entire document.

Some relational DBs allow for locality.

2.5.4 Convergence of Document and Relational DBs

Most relational DBs support XML, and some support JSON.

Some document DBs support relational-like joins.

3 Query Languages for Data

Imperative languages specify operations to perform.

Declarative languages specify only the pattern of the data you want.

Declarative languages allow any implementations and lend themselves to parallel execution.

3.1 Declarative Queries on the Web

HTML and CSS are declarative languages used on the web.

3.2 MapReduce Querying

Programming model for processing large amounts of data in bulk across machines.

Neither fully declarative or imperative.

Steps in MapReduce:

1. Filter is specified declaratively.
2. `map` is called once for every document that matches the query, where `this` is the document object
3. `map` emits a key and a value
4. Key-value pairs emitted by `map` are grouped by the key, and for all key-value pairs with the same key, `reduce` is called once
5. `reduce` adds up the number of animals from all observations in a particular month
6. Final output is written to collection specified.

`map` and `reduce` only use the data passed as input, cannot perform additional DB queries, and cannot have any side effects.

4 Graph-Like Data Models

Graphs can be used as a data model, especially for many-to-many relations.

4.1 Property Graphs

Each vertex consists of:

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

Each edge consists of:

- unique identifier
- **tail vertex**: vertex at which edge starts
- **head vertex**: vertex at which edge ends
- label to describe relationship between two vertices
- collection of properties (key-value pairs)

Any vertex can have an edge connecting it with any other vertex. No schema restricts which kinds of things can or cannot be associated.

Given any vertex, can traverse the graph through incoming and outgoing edges.

By using different labels for different kinds of relationships, store several kinds of info in a single graph while still maintaining a clean data model.

Graphs are good for evolvability: as features are added to an application, a graph can easily be extended to accommodate changes in the application's data structures.

4.2 The Cypher Query Language

Declarative query language for property graphs.

Query optimizer automatically chooses most efficient strategy.

4.3 Graph Queries in SQL

Graph data in a relational structure can be queried with SQL, though with some difficulty.

Variable-length traversal paths in a query can be expressed using **recursive common table expressions**.

4.4 Triple-Stores and SPARQL

In a **triple-store**, all info is stored in the form of simple 3-part statements: (subject, predicate, object).

Subject of the triple is a vertex. Object can be a value or a vertex. Predicate is an edge.

4.4.1 Semantic Web

Idea that websites should publish info as machine-readable data for computers to read.

4.4.2 RDF Data Model

Used for internet-wide data exchange.

Subject, object, and predicate are URIs.

4.4.3 SPARQL Query Language

Query language for triple-stores using RDF data model.

4.5 The Foundation: Datalog

Older language used in few data systems.

Define rules that tell that DB about new predicates.

Predicates aren't triples stored in the DB, but are derived from data or rules.

A rule applies if the system can find a match for all predicates on the right hand side of the `:-` operator.