2. Data Models and Query Languages

- Data models
  - 👍 are perhaps the most important part of developing software
    - how the software is written
    - how we think about the problem that we are solving
  - layering one data model on top of another
    - abstractions
  - data model embodies assumptions about how it is going to be used
- Relational Model vs Document Model
  - SQL
  - The Birth of NoSQL
  - The Object-Relational Mismatch
    - Whether you store an ID or a text string is a question of duplication
      - Anything that is meaningful to humans may need to change sometime in the future
  - Many-to-One and Many-to-Many Relationships
    - One-to-many
      - hierarchical model
    - Many-to-many
      - Need some other approach
  - Are Document Databases Repeating History?
    - Network model
      - CODASYL model
      - a record could have multiple parents
    - Relational model
    - Comparison to document databases
      - Hierarchical model
        - every record has exactly one parent
      - And relational model for many-to-many relations
  - Relational vs Document Databases Today
    - Which data model leads to simpler application code?
      - If app has document-like data structures
        - 👍 document model can be the best
        - shredding (splitting a document-like structure into multiple tables)
          - can lead to cumbersome schema
      - document model has limitations
        - 👎 cannot refer directly to a nested item within a document
      - for highly interconnected data (when many-to-many is common)
        - 👎 document model is awkward
        - ⚠️ relational model is acceptable
        - 👍 graph models are the most natural
    - Schema flexibility in the document model
      - Two types
        - schema-on-read
          - "schemaless" is incorrect term
        - schema-on-write
          - traditional approach of relational databases
      - It is similar to dynamic type checking vs static type checking
      - 👍 schema-on-read approach is advantageous when
        - There are many different types of objects, and it is not practical to put each type of object in its own table
        - The structure of the data is determined by external systems over which you have no control and which may change at any time
    - Data locality for queries
      - if app often needs to access the entire document
      - recommended to
        - keep documents fairly small
        - avoid writes that increase the size of a document
      - Some related databases support this feature
        - Google's Spanner db has tables nested within parent table
        - Oracle has multi-table index cluster tables
        - column-family concept in the Bigtable data model
          - Cassandra
          - HBase
    - Convergence of document and relational databases
- Query Languages for Data
  - imperative language
    - tells the computer to perform certain operations in a certain order
  - declarative
    - just specify
      - the pattern of the data you want (what conditions the results must meet)
      - how you want the data to be transformed (e.g., sorted, grouped, and aggregated)
      - but not how to achieve that goal
    - is attractive because
      - hides implementation details of the database engine
      - often lend themselves to parallel execution
  - Declarative Queries on the Web
    - Also good for web - CSS / XSL
  - MapReduce Querying
    - MapReduce
      - a programming model for processing large amounts of data in bulk across many machines
      - It is between declarative and imperative
        - logic of the query is expressed with snippets of code
        - Based on functions
          - map (also known as collect)
          - reduce (also known as fold or inject)
          - must be pure functions (can only use input, and must not have any side effects)
      - It has usability problem - harder to write two coordinated functions than single query
      - aggregation pipeline
        - Is more expressive
          - Just like SQL
            - 😊 So, NoSQL is reinventing SQL
- Graph-Like Data Models
  - two kinds of objects
    - vertices (also known as nodes or entities)
    - edges (also known as relationships or arcs)
  - Typical examples
    - Social graphs
    - The web graph
    - Road or rail networks
  - few ways to structure and query
    - property graph
      - implemented by Neo4j, Titan, and InfiniteGraph
    - triple-store
      - implemented by Datomic, AllegroGraph, and others
  - three declarative query languages for graphs
    - Cypher, SPARQL, and Datalog
  - imperative graph query
    - Gremlin
  - graph processing frameworks
    - Pregel
  - Property Graphs
    - can think of a graph store as consisting of two relational tables
      - vertices
      - edges
    - The Cypher Query Language
      - declarative query language for property graphs, created for the Neo4j graph database
      - -[:WITHIN*0..]->
    - Graph Queries in SQL
      - recursive common table expressions
        - WITH RECURSIVE syntax
  - Triple-Stores and SPARQL
    - subject, predicate, object
    - Turtle language
      - _:lucy a :Person; :name "Lucy"; :bornIn _:idaho.
    - The semantic web
      - Resource Description Framework (RDF)
        - machine-readable data on web sites
      - Triples can be a good internal data model for applications
    - The RDF data model
      - Turtle/N3 language
      - Apache Jena
        - can convert between different RDF formats
    - The SPARQL query language
      - a query language for triple-stores using the RDF data model
    - Graph Databases Compared to the Network Model
  - The Foundation: Datalog
    - predicate(subject, object)
    - is a subset of Prolog