NoSQL Databases and CouchDB

Outline

- Traditional RDBMS
- The NoSQL paradigm
- CouchDB
- CouchDB tutorial and examples

Traditional RDBMS

- Relational databases are still the standard in higher education teaching
  - Strong theoretical foundation
  - Helps in the development of modeling skills
- The main principle is that of tables with rows and keys that link between rows across tables

The strong points of RDBMS

- Highly structured
- RDBMS can enforce application logic
- It has its own query language
- Query logic is embedded into the application code

The weak points of RDBMS

- Highly structured
  - It may not work well with semi-structured data
- RDBMS can enforce application logic
  - What if the entire logic is not apparent?
- It has its own query language
  - Not an intuitive language
- Query logic is embedded into the application code
  - Similar queries need to be enforced across multiple projects

The CAP theorem

- The CAP theorem states that a database can only choose between two of the following three traits:
  - Consistency
  - Availability
  - Partition tolerance

- RDBMS traditionally went for consistency and partition tolerance, which are the two most conservative

- New use-cases prompted the need for availability to play a greater role

- Today, these lines are blurred

The advent of NoSQL

- RDBMS had other competitors, such as object databases, but remained the standard for many years
- From 2005 and on, the IT world experienced an incredible increase in
  - Scale
  - Complexity
  - Entrepreneurship in online services

NoSQL for startups

- Not everyone was a SQL wizard
- RDBMS did not scale well and were difficult to manage as large-scale clusters
- The application model was not fully established
- Complex XML or relational schemas seemed unintuitive
- The advent of APIs and developers wanted a simpler data format

NoSQL's proposed benefits

- Data is stored as JSON-like documents or key/value pairs
- More suitable to store "objects"
- Allows semi-structured data that evolves over time
- Single operation updates
- Designed for horizontal scaling
- Can provide higher performance if they sacrifice consistency for availability

NoSQL design choices

- Query languages vary, except that they abandon the SQL language
- Focus is on simplicity in code and use
- Most are designed for easy adoption, often for the sake of security
- Scale, easy replication and clustering are often promoted as strong points
- Some offer a crossover into APIs

NoSQL examples

mongoDB

redis

cassandra

CouchDB

Apache CouchDB

Apache CouchDB

- A NoSQL database designed for easy adoption, yet with powerful features such as replication and clustering
- No client library, everything goes via HTTP
- Data is stored as native JSON
- A graphical management system, Fauxton (futon before)