Introduction

Meet CouchDB

CouchDB is often categorized as a “NoSQL” database, a term that became increasingly popular in late 2009, and early 2010. While this term is a rather generic characterization of a database, or data store, it does clearly define a break from traditional SQL-based databases. A CouchDB database lacks a schema, or rigid pre-defined data structures such as tables. Data stored in CouchDB is a JSON document(s). The structure of the data, or document(s), can change dynamically to accommodate evolving needs.

What it is Not

To better understand what CouchDB is, it may be helpful to understand a few things that CouchDB isn't:

- A relational database. These differences are articulated above in the Meet CouchDB section, and other portions of this Wiki.
- A replacement for all databases. When developing and designing a good information system you should select the best tool for the job. While CouchDB can be used in a wide variety of applications you may find that another data store is a better fit for your problem. If you are new to CouchDB, and aren't sure if it's a good fit for your data management problem, please ask others on the mailing list and the #couchdb IRC channel for advice.
- An object-oriented database. While CouchDB stores JSON objects, it isn't meant to function as a seamless persistence layer for an object-oriented programming language.

Key Characteristics

Let's review some of the basic elements of CouchDB.

Documents

A CouchDB document is a JSON object that consists of named fields. Field values may be strings, numbers, dates, or even ordered lists and associative maps. An example of a document would be a blog post:

```
{
    "Subject": "I like Plankton",
    "Author": "Rusty",
    "PostedDate": "5/23/2006",
    "Tags": ["plankton", "baseball", "decisions"],
    "Body": "I decided today that I don't like baseball. I like plankton."
}
```

In the above example document, Subject is a field that contains a single string value "I like plankton". Tags is a field containing the list of values "plankton", "baseball", and "decisions".

A CouchDB database is a flat collection of these documents. Each document is identified by a unique ID.

Views

To address this problem of adding structure back to semi-structured data, CouchDB integrates a view model using JavaScript for description. Views are the method of aggregating and reporting on the documents in a database, and are built on-demand to aggregate, join and report on database documents. Views are built dynamically and don’t affect the underlying document; you can have as many different view representations of the same data as you like. Incremental updates to documents do not require full re-indexing of views.

Schema-Free

Unlike SQL databases, which are designed to store and report on highly structured, interrelated data, CouchDB is designed to store and report on large amounts of semi-structured, document oriented data. CouchDB greatly simplifies the development of document oriented applications, such as collaborative web applications.

In an SQL database, the schema and storage of the existing data must be updated as needs evolve. With CouchDB, no schema is required, so new document types with new meaning can be safely added alongside the old. However, for applications requiring robust validation of new documents custom validation functions are possible. The view engine is designed to easily handle new document types and disparate but similar documents.

Distributed

CouchDB is a peer based distributed database system. Any number of CouchDB hosts (servers and offline-clients) can have independent "replica copies" of the same database, where applications have full database interactivity (query, add, edit, delete). When back online or on a schedule, database changes can be replicated bi-directionally.

CouchDB has built-in conflict detection and management and the replication process is incremental and fast, copying only documents changed since the previous replication. Most applications require no special planning to take advantage of distributed updates and replication.

Unlike cumbersome attempts to bolt distributed features on top of the same legacy models and databases, replication in CouchDB is the result of careful ground-up design, engineering and integration. This replication framework provides a comprehensive set of features:

- Master Slave replication
- Master Master replication
- Filtered Replication
- Incremental and bi-directional replication
- Conflict management

These replication features can be used in combination to create powerful solutions to many problems in the IT industry. In addition to the fantastic replication features, CouchDB's reliability and scalability is further enhanced by being implemented in the Erlang programming language. Erlang has built-in support for concurrency, distribution, fault tolerance, and has been used for years to build reliable systems in the telecommunications industry. By design, the Erlang language and runtime are able to take advantage of newer hardware with multiple CPU cores.

Next Steps

For installation instructions consult the installation instructions or dive right into the basics if CouchDB is already installed. Detailed information for the more curious can be found in the technical overview.