Types of NoSQL databases

This document summarises the difference between the types of NoSQL databases (key value stores, document databases, column family stores and graph databases) and provides an example of a possible application in each case.

1. Key Value stores

A key value store is a simple hash table, primarily used when all access to the database is via the primary key. The Key/value model is the simplest and easiest to implement. However, it is inefficient when you are only interested in querying or updating part of a value.

Examples:	Riak, Redis, Voldemort
Typical applications:	Storing session information, user profiles/preferences, shopping cart data etc.
Data model:	Collection of key-value pairs
Strengths:	Fast lookups
Weaknesses:	Stored data has no schema

Comparison of terminology (Oracle vs. Riak)

Oracle	Riak
Database instance	Riak cluster
Table	Bucket
Row	Key-value
Row ID	Key

Example application: *Storing session information*

Generally, every web session is unique and is assigned a unique sessionid value. Applications that store the sessionid on disk or in an RDBMS will greatly benefit from moving to a key-value store, since everything about the session can be stored by a single PUT request or retrieved using GET. This single-request operation makes it very fast, as everything about the session is stored in a single object.

2. Document databases

In the document model, the database stores and retrieves documents, which can be written in languages like XML (eXtensible Markup Language) or JSON (JavaScript Object Notation). These documents are self-describing, hierarchical tree structures which allow nested values associated with each key. Document databases support querying more efficiently.

Sources: http://www.thoughtworks.com/insights/articles/nosql-comparison,
NoSQL Distilled (Sadalage & Fowler, 2013)

Examples:	CouchDB, MongoDb
Typical applications:	Event logging, content management systems, web analytics or e-commerce applications
Data model:	Documents (with multiple values in a document)
Strengths:	Tolerant of incomplete data
Weaknesses:	Query performance, no standard query syntax

Comparison of terminology (Oracle vs. MongoDB)

Oracle	MongoDB
Database instance	MongoDB instance
Schema	Database
Table	Collection
Row	Document
Row ID	ID

Example application: *Web analytics*
Document databases can store data for real-time analytics; since parts of the document can be updated, it's very easy to store page views or unique visitors, and new metrics can be added easily without schema changes.

3. Column Family Stores

Column family stores allow you to store data with keys mapped to values and the value grouped into multiple column families, each column family being a map of data.

Examples:	Cassandra, HBase, Amazon SimpleDB
Typical applications:	Event logging, content management systems, blogging platforms, counters
Data model:	Columns – column families
Strengths:	Fast lookups, good distributed storage of data
Weaknesses:	Performing different types of queries can be complex

Comparison of terminology (RDBMS vs. Cassandra)

RDBMS	Cassandra
Database instance	Cluster
Database	Keyspace
Table	Column family
Row	Row
Column (same for all rows)	Column (can be different per row)

Sources: http://www.thoughtworks.com/insights/articles/nosql-comparison,
NoSQL Distilled (Sadalage & Fowler, 2013)

Example application: *Blogging platforms*
Using column families, you can store blog entries with tags, categories, links and trackbacks in different columns. Comments can either be stored in the same row or moved to a different keyspace; similarly, blog users and the actual blogs can be put into different column families.

4. Graph Databases

Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties. Think of a node as an instance of an object in the application. Relations are known as edges that can also have properties. Nodes are organised by relationships which allow you to find interesting patterns between the nodes. The organisation of the graph lets the data be stored just once and then interpreted in different ways based on relationships.

Examples:	Neo4J, InfoGrid, Infinite Graph
Typical applications:	Connected data, routing, dispatch and location-based services, recommendation engines
Data model:	Graph (nodes and edges)
Strengths:	Allows for the type of analysis that aggregate-oriented databases find very difficult
Weaknesses:	Has to traverse the entire graph to achieve a definitive answer.

Example application: *Recommendation engines*
As nodes and relationships are created in the system, they can be used to make recommendations like "your fields also brought this product" or "when invoicing this item, these other items are usually invoiced". Or, it can be used to make recommendations to travellers mentioning that when other visitors come to Barcelona they usually visit Antonio Gaudi's creations.

An interesting side effect of using the graph databases for recommendations is that as the data size grows the number of nodes and relationships available to make the recommendations quickly increases. The same data can also be used to mine information – for example, which products are always bought together, or which items are always invoiced together; alerts can be raised when these conditions are not met. Like other recommendation engines, graph databases can be used to search for patterns in relationships to detect fraud in transactions.

Sources: http://www.thoughtworks.com/insights/articles/nosql-comparison,
NoSQL Distilled (Sadalage & Fowler, 2013)