**Quiz 1:** general high-level NoSQL concepts, data models, scalability issues

Some true/false, multiple choice, and short answer

Read Harrison chapter 1 and the assigned paper Vogel2008 (on blackboard)

**Questions:**

**Why NoSQL emerged as an alternative to Relational?**

* Traditional relational databases do not scale well
* Offer many different models: Key/Value, Document Stores, Graphs, Columns as opposed to relational models which are all tables
* More cost efficient, hardware costs scale linearly with NoSQL. Licensing costs for relational structures can become prohibitive when scaling vertically.
* Stringent relational schema not ideal for unstructured data. Non relational models are schema-less allowing for fields to be added on the fly without re-definition
  + Ideal for non-uniform data and custom fields
* Open source, but have many commercial versions with add-ons and support
* Designed for distributed storage / cluster environments
* NoSQL supports trend towards polyglot persistence (when storing data, it is best to use multiple technologies, no “one size fits all”): using different data stores in different circumstances as part of a complete data-engineering architecture

**How does NoSQL differ from Relational approaches (keep in mind - there are always exceptions)**

|  |  |  |
| --- | --- | --- |
| **Relational** | **Characteristic** | **NoSQL** |
| Tables | **Data Models** | Key Value, Document, Graphs, Columns |
| Centralized, rate limiting | **Computing Models** | Distributed and scalable |
| Pre-declared | **Schema** | Flexible |
| Fixed | **Data Types** | Flexible |
| No | **Hierarchical** | Yes |
| Licensing can become prohibitive as you scale vertically | **Costs** | Hardware costs scale linearly |

* Relational: Centralized, strict consistency, transactional, single point of failure
* NoSQL: Distributed, highly available, highly resilient, Weak / Eventual consistency

**Partitioning vs. Replication - main ideas, what is the difference?**

* Partitioning: creating divisions of database into distinct independent parts based on data cohesion
  + Done for manageability, performance or availability purposes
  + Ex: Splitting an ERP database into sales database, accounts database, materials database
* Replication: Adding secondary database servers to keep up with demand. Copying tables / databases onto multiple servers, used to improve speed of access.

**What's the difference between a Key Value Store and a Document Store?**

* Key Value Store: access to data is only available via a key, but the value can be anything (simple value, document, list or database)
  + Often used for very fast yet simple operations
  + Data is simple hash tables
  + Ex: managing online shopping carts
* Document Store: stores data as structured documents, usually XML or JSON
  + Simple, flexible, scalable
  + Data is self describing, supports ad hoc query by value, not available in Key/Value Stores
  + Aligned with web-service based programming models, using XML/JSON as transport layer
  + Suitable for horizontal scaling
  + Query by content

**What is a graph database?**

* Graph Database: made up of a collection of nodes and relationships
  + Nodes have one or more categorical labels and represent things
  + Relationships have a single type and are between two nodes
  + Properties are key value pairs that are assigned to a node or relationship

**NoSQL: What is it good for? What are its limitations? What are some applications?**

**What is "Polyglot persistence?"**

* Polyglot persistence: the idea that there is no “one size fits all” way to store data (as assumed by relational models). Use different technologies and structures to best fit unique data sets, creates complete data engineering architecture.

**What is "Data Engineering"**

* Developing, constructing, testing and maintaining data architectures
* Includes data modeling, API implementation, data warehousing
* CPUs are no longer the only limiting factor. Also size (volume), flow rate (velocity) and complexity (variety) of the data at hand.
* Many applications are data intensive as opposed to only compute intensive

**What is Big Data all about?**

* Concerned with the acquisition, storage and analysis of extremely large volumes of data
* What we consider big data today, might not be the same as tomorrow
* 2018: Big data would be thought of as between 100 – 1,000 terabytes of data
* Sources of Big Data:
  + Activity data (online transactions, credit card data, stock data)
  + Sensor networks
  + Biological databases, including genomic data
  + Social Networking

**Approaches / Strategies to Scaling Relational Databases**

* Vertical Scaling: add more resources to your database management system – more RAM, more cores, faster disks, more $
  + Advantages: leveraging a platform you already know, easier software migration, easier administration, lower cooling overhead, lower networking hardware overhead
  + Disadvantages: exponential server costs and licensing costs, vendor lock-in, single point of failure, limits on available hardware expansion
* Horizontal Scaling: adding more machines to your pool of resources
  + Advantages: uses commodity hardware, greater reliability
  + Disadvantages: Incompatible with existing relational storage models, high licensing costs, computing models are more complicated, security issues might be more complicated or less mature
* Replication: adding secondary servers to keep up with demand. Copying tables / databases onto multiple servers, used to improve speed of access.
* Sharding: type of partitioning that separates large databases into smaller, faster, more easily manageable parts called shards.
  + Relational databases could also be run as separate servers for different shards
  + While this separates the load, all the sharding has to be controlled by the application which has to keep track of which database server to talk to for which data
* Memcaching: trading consistency for speed
* Denormalization: trading write performance for speed
  + Complex joins are simplified or eliminated making queries faster

**When is caching useful in theory?  (Hit rates, memory vs disk I/O performance)**

* Caching: remembering results of expensive queries to speed up reads

**Client-side vs. Server side consistency (See the assigned paper - Vogels2008 - in Course Materials >> Papers)**

**What is an "Inconsistency Window?" (also in Vogels2008).**

**I give you a SQL query and some processing elements (FILTER, JOIN, PARTITION\_BY\_KEY, etc). Show me conceptually how you would parallelize the query across a compute cluster.**