Rusu Wu
11694764
CptS415
Assignment 4

1.

a) NoSQL is a non-relational database management system. NoSQL database technology stores information in JSON documents instead of columns and rows used by relational databases.

b) **Key-value stores:** Key-value stores group associated data in collections with records that are identified with unique keys for easy retrieval. Key-value stores have just enough structure to mirror the value of relational databases while still preserving the benefits of NoSQL. Example: Riak, and Oracle NoSQL database

Column store: Column store databases use the tabular format of relational databases yet allow a wide variance in how data is named and formatted in each row, even in the same table. Like key-value stores, wide-column databases have some basic structure while also preserving a lot of flexibility. Example: Scylla and Microsoft Azure Cosmos DB

Document-oriented: Document-oriented databases are primarily built for storing information as documents, including, but not limited to, JSON documents. These systems can also be used for storing XML documents, for example. Example: Apache CouchDB, IBM Cloudant, Couchbase Server

Graph (beyond noSQL): Graph databases uses graph structures to define the relationships between stored data points. Graph databases are useful for identifying patterns in unstructured and semi-structured information. Example: Amazon Neptune, AnzoGraph DB, DataStax Enterprise Graph

c) Social networking applications such as Facebook can make best use of the **Graph (beyond noSQL)** system. Graph databases allow developers to focus more on relations between objects rather than on the objects themselves. In this context, the system is beneficial to the establishment of a scalable social network that contain a scalable relation information. Social networks are about connections between people, so basically, they have graph structures. Needless to say, graph databases like Neo4j are perfectly tailored to social networks. They speed up the development of social network applications, enhance an app's overall performance, and allow you to better understand your data.

On the contrary, Content management systems may not be well-suited for the **Graph (beyond noSQL)** system.Suppose you want to create a social networking app which can save users' profile information and connections. If you use Graph-Based NoSQL database your application would be a champ in fetching user connections like friends, followers, etc, but when it comes to fetching user profile information or posting texts, the database fails miserably. However, using Document Database like MongoDB would greatly help in storing profile and post documents in such a case.

2.

a) Challenges are follows.

Fault Tolerance: replication; handle machine failures without losing data and without degradation in Performance.

Scalability: serve get ()'s in parallel; replicate/cache hot tuples; Need to scale to thousands of machines; Need to allow easy addition of new machines.

Consistency: quorum consensus to improve put/get performance; maintain data consistency in face of node failures and message losses.

Heterogeneity (if deployed as peer-to-peer systems): Latency challenge.

b) Two basic operations in a KV store

put (key, value)

get(key)

3.

a) **Data storage:**

Column store data are stored in the disk column by column.

In a column-store, all values of one column are stored sequentially on a database page.

I/O efficient for read-only queries as they read, only those attributes which are accessed by a query.

Most of the queries does not process all the attributes of a particular relation.

Storage key:

To answer queries, projections are joined using Storage keys and join indexes.

Within a segment, every data value of every column is associated with a unique Skey.

Values from different columns with matching Skey belong to the same logical row.

b) Compression: Data stored in columns is more compressible than data stored in rows. Compression algorithms perform better on data with low information entropy (i.e., with high data value locality), and values from the same column tend to have more value locality than values from different columns.

Late Materialization: Most applications read results tuple-at-a-time not column-at-a-time, so delaying Tuple Construction, and avoid constructing it altogether. In this case, it formed only relevant tuples and that too much later during execution. No CPU was wasted on forming tuples upfront without knowing if a particular tuple might eventually be a part of result-set or not, which is more efficient utilization of CPU-memory bandwidth.

Block iteration: Operators operate on blocks of tuples at once and iterate over blocks rather than tuples. If column is fixed size, it can be operated as an array, which can minimize per-tuple overhead and exploits potential for parallelism