Practical 2: Analyze impact of storage format

Gahan Saraiya (18MCEC10), Rushi Trivedi (18MCEC08)

18mcec10@nirmauni.ac.in

I. Introduction

The data can not be always considered to be fit in one size and the goal of this article is to deduce the impact of various storage format.

Data needs to be stored in persistence storage, the focus is always to retrieve data efficiently. To achieve that we need to have optimal storage mechanism/schemes that allows to access the data efficiently.

II. Static data placement on disk pages

I. N-ary Storage Model (NSM)

Also known as Slotted Page.

- record are stored sequentially on data pages
- Accesses full record (all attributes of record)

✓ attributes of a same record are stored together

✗ Doesn't work well on modern memory hierarchies

Flow The whole page from disk is loaded in to main memory and then block(s) are fetched CPU cache

Impact

✓ Best results if full access to record is require

✗ Partial record access is slow. Due to Fixed page layout it wastes I/O bandwidth

✗ At CPU cache low spatial locality

II. Decomposition Storage Model (DSM)

Original Table is partitioned in to single attribute sub-tables. Each sub-table is stored in separate NSM page

- N-attribute table stored in to N sub-tables

✓ Saves I/O (eliminates unnecessary I/O)

✗ Hard to modify/construct record afterwards

Flow Only required NSM page(s) (single attributed sub-table) is/are loaded in to main memory and blocks are fetched from loaded page **Impact**

- ✓ Best results if partial access to record is require
- ✗ full record access require reconstruction which may lead to unnecessary I/O and performance degraded compare to NSM

III. Partition Attribute Across (PAX)

In single page it partitions data for spacial locality by attribute

Flow Likewise in NSM it loads whole PAX page from disk in to main memory. partitioned data are stored in to contiguous block and only require block(s) needs to be fetched **Impact**

- ✓ Optimizes communication/transfer between Main memory and CPU cache memory (Increases Hit Ratio)
- ✓ Optimal solution for both full and partial record access

III. Modern techniques of data storage

Modern Data Storage techniques can be classified in below categories:

1. Column-oriented DBMS

   stores data tables by column rather than by row
2. Key-value pair
3. Document oriented
4. NoSQL (Not Only SQL database)

   not required to follow an established relational schema
5. Graph Based

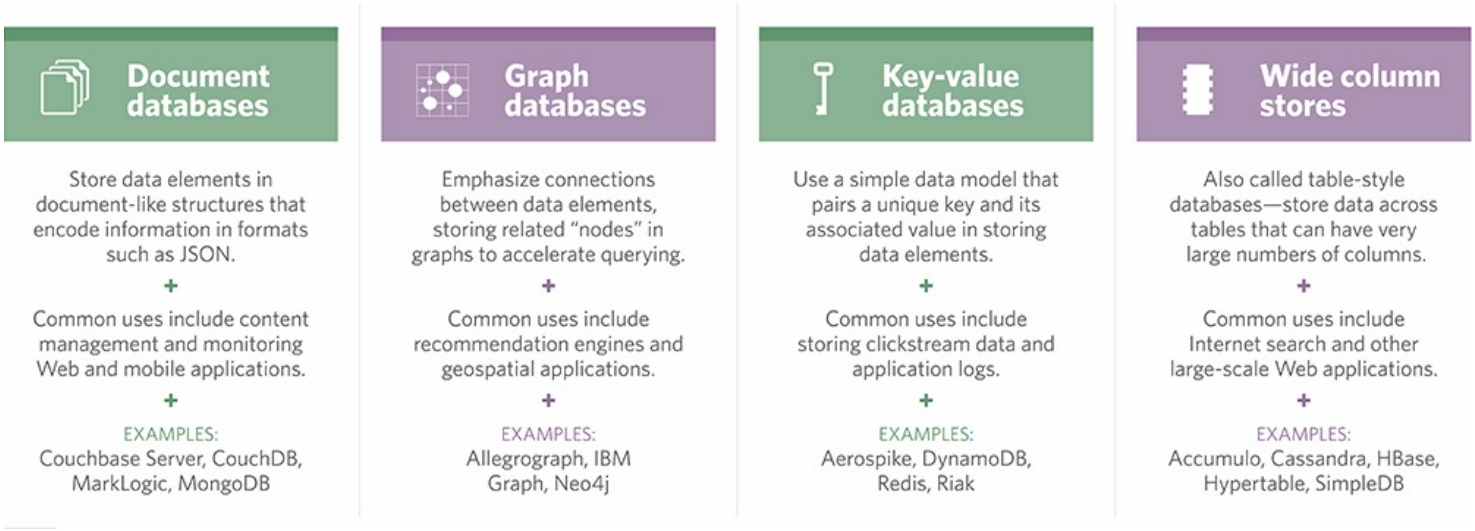

Figure 1: NoSQL family

I. Column-oriented DBMS

stores data tables by column rather than by row.

- serializes all of the values of a column together

✓ well suited for sparse data sets

✗ retrieve all the data for a given record (entire row) requires more disk operations to collect data from multiple columns

I.1 Cassandra

Cassandra is a distributed database management system designed for handling a high volume of structured data across commodity servers.

- Apache Cassandra is highly scalable, distributed and high-performance NoSQL database.
- Cassandra is designed to handle a huge amount of data.
- Cassandra handles the huge amount of data with its distributed architecture.
- Data is placed on different machines with more than one replication factor that provides high availability and no single point of failure.

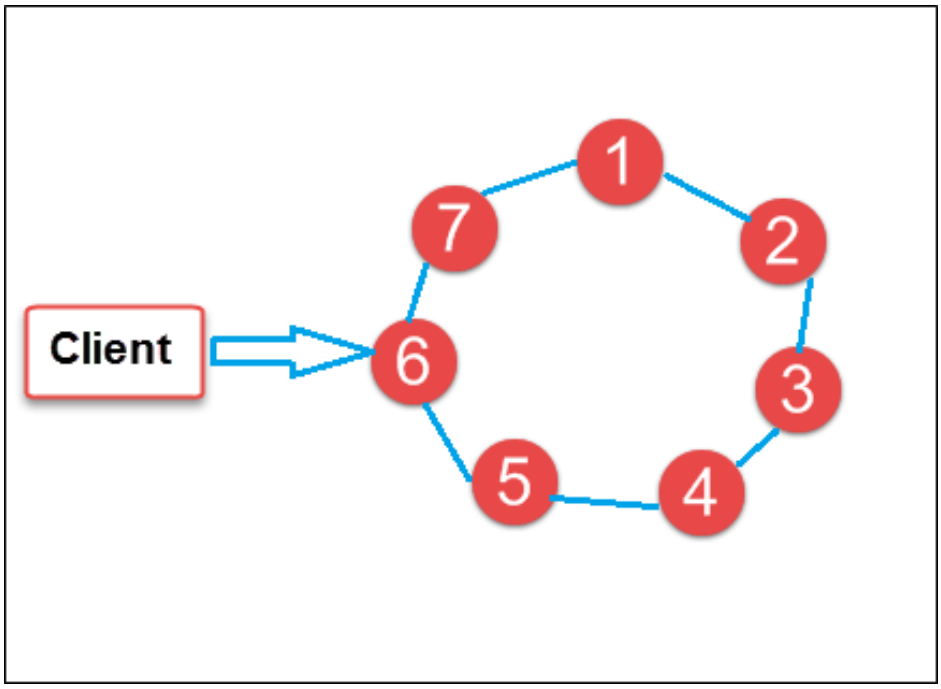

Figure 2: Visualizing Cassandra structure

In Fig. 2 circles are Cassandra nodes and lines between the circles shows distributed architecture, while the client is sending data to the node.

I.2 HBase

HBase is a column-oriented database management system that runs on top of Hadoop Distributed File System (HDFS). It is well suited for sparse data sets, which are common in many big data use cases.

Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical Apache™ MapReduce application. HBase does support writing applications in Apache™ Avro™, REST, and Thrift.

An HBase system comprises a set of tables. Each table contains rows and columns, much like a traditional database. Each table must have an element defined as a Primary Key, and all access attempts to HBase tables must use this Primary Key. Avro, as a component, supports a rich set of primitive data types including: numeric, binary data and strings; and a number of complex types including arrays, maps, enumerations and records. A sort order can also be defined for the data.

II. Graph based

A graph database is a database designed to treat the relationships between data as equally important to the data itself. It is intended to hold data without constricting it to a pre-defined model. Instead, the data is stored like we first draw it out – showing how each individual entity connects with or is related to others.

- Compared to RDBMS, it eliminates keys in to relation between nodes

- ✓ easy to model and store relationship
- ✓ performance of relationship traversal remains constant with growth in data size
- ✓ queries are shortened and more readable
- ✓ adding additional properties and relationship can be done on the fly - no migrations

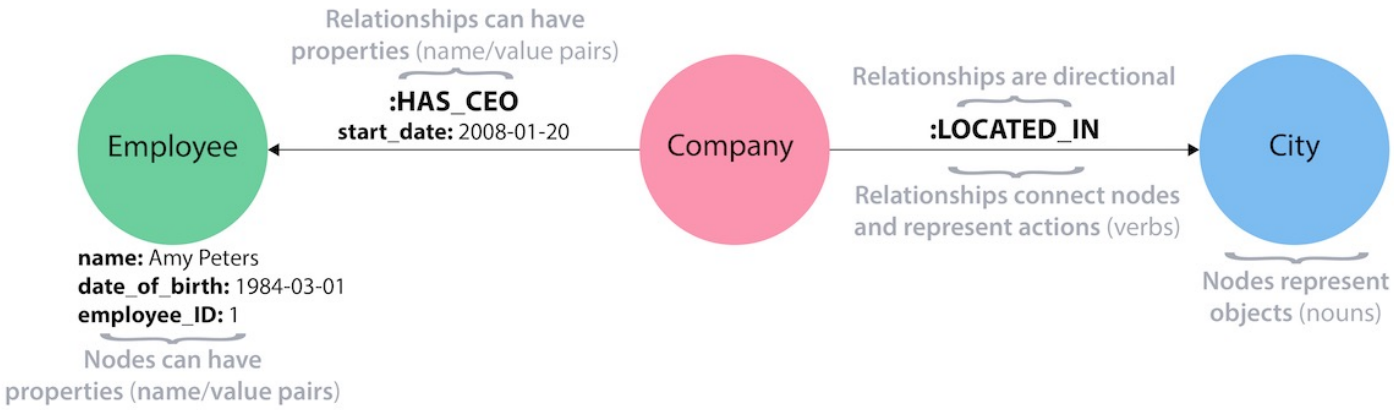

Figure 3: Building blocks of Property Graph Model

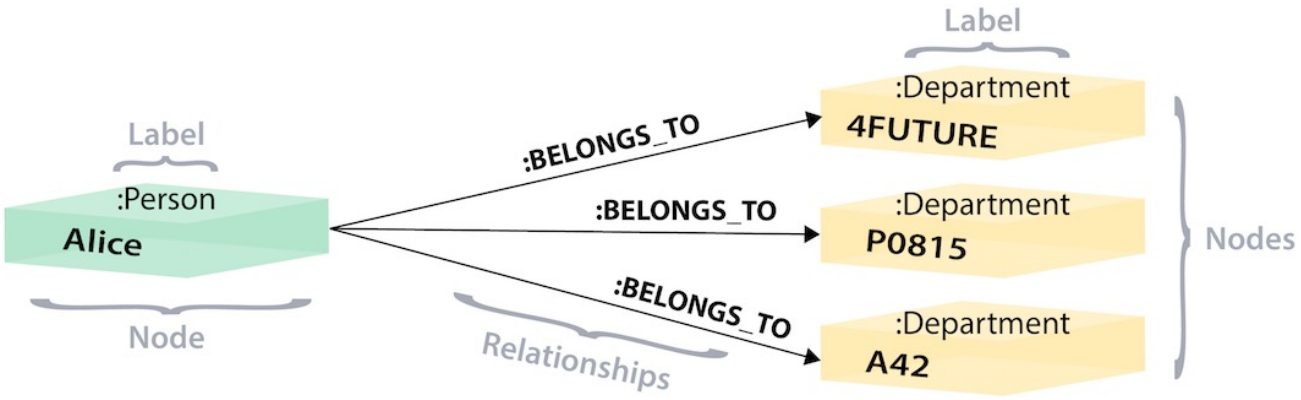

Figure 4: Graph – Alice and 3 Departments as nodes

IV. Summary

Table 1: Comparison of NSM, DSM and PAX

Performance	**Cache↔Memory**		**Memory↔Disk**	
Page Layout	**full record access**	**partial record access**	**full-record access**	**partial record access**
NSM	✔	✗	✔	✗
DSM	✗	✔	✗	✔
PAX	✔	✔	✔	✗