Module 3

NoSQL Big Data Management, MongoDB and Cassandra: Introduction, NoSQL Data Store, NoSQL Data Architecture Patterns, NoSQL to Manage Big Data, Shared-Nothing Architecture for Big Data Tasks, MongoDB, Databases, Cassandra Databases.

Introduction:

Big Data uses distributed systems. A distributed system consists of multiple data nodes at clusters of machines and distributed software components. The tasks execute in parallel with data at nodes in clusters. The computing nodes communicate with the applications through a network.

Following are the features of distributed-computing architecture:

1. Increased reliability and fault tolerance: The important advantage of distributed is computing system reliability. If a segment of machines in a cluster fails then the rest of the machines continue work. When the datasets replicate at number of data nodes, the fault tolerance increases further. The dataset in remaining segments continue the same computations as being done at failed segment machines.
2. Flexibility makes it very easy to install, implement and debug new services in a distributed environment.
3. Sharding is storing the different parts of data onto different sets of data nodes, clusters or servers. For example, university students huge database, on sharding divides in databases, called shards. Each shard may correspond to a database for an individual course and year. Each shard stores at different nodes or servers.
4. Speed: Computing power increases in a distributed computing system as shards run parallelly on individual data nodes in clusters independently (no data sharing between shards).
5. Scalability: Consider sharding of a large database into a number of shards, distributed for computing in different systems. When the database expands further, then adding more machines and increasing the number of shards provides horizontal scalability.
6. Resources sharing: Shared resources of memory, machines and network architecture reduce the cost.
7. Open system makes the service accessible to all nodes.
8. Performance: The collection of processors in the centralized system provides higher performance than a centralized computer.

The demerits of distributed computing are:

i) issues in troubleshooting in a larger networking infrastructure
ii) additional software requirements
iii) security risks for data and resources.

Big Data solutions require a scalable distributed computing model with solution shared-nothing architecture. A solution is Big Data store in HDFS files. NoSQL data also store Big Data, and facilitate random read/write accesses. The accesses are sequential in HDFS data.

Following are selected key terms used in database systems:

Class refers to a template of program codes that is extendable. Class creates instances, called objects. A class consists of initial values for member fields, called state (of variables), and implementations of member functions and methods called behaviour.

Object is an instance of a class in Java, C++, and other object-oriented languages. Object can be an instance of another object (for example, in JavaScript).

Tupple is an ordered set of data which constitutes a record. For example, one row record in a table. A row in a relational database has column fields or attributes.

Transaction means execution of instructions in two interrelated entities, such as a query and the database.

Database transactional model refers to a model for transactions.

MySQL refers to a widely used open-source database, which excels as a content management server.

Oracle refers to a widely used object-relational DBMS, written in the C+ language that provides applications integration with service-oriented architectures and has high reliability. Oracle has also released the NoSQL database system.

DB2 refers to a family of database server products from IBM with built-in support to handle advanced Big Data analytics.

Sybase refers to database server based on relational model for businesses, primarily on UNIX. Sybase was the first enterprise-level DBMS in Linux.

MS SQL server refers to a Microsoft-developed RDBMS for enterprise-level databases that supports both SQL and NoSQL architectures.

PostgreSQL refers to an enterprise-level, object-relational DBMS. PostgreSQL uses procedural languages like Perl and Python, in addition to SQL.

SQL Data Store:

SQL is a programming language based on relational algebra. It is a declarative language and it defines the data schema. SQL creates databases and RDBMSs. RDBMS uses tabular data store with relational algebra, precisely defined operators with relations as the operands.

ACID Properties in SQL Transactions

- **Atomicity** of transaction means all operations in the transaction must complete, and if interrupted, then must be undone (rolled back).
- **Consistency** in transactions means that a transaction must maintain the integrity constraint, and follow the consistency principle.
- **Isolation** of transactions means two transactions of the database must be isolated from each other and done separately.
- **Durability** means a transaction must persist once completed.

Triggers, Views and Schedules in SQL Databases:

Trigger is a special stored procedure. Trigger executes when a specific action(s) occurs within a database. such as change in table data or actions such as UPDATE, INSERT and DELETE. For example, a Trigger store procedure inserts new columns in the columnar family data store.

View refers to a logical construct, used in query statements. A View saves a division of complex query instructions and that reduces the query complexity. Viewing of a division is similar to a view of a table. View does not save like data at the table. Query statement when uses references to a view, the statement executes the View. Query (processing) planner combines the information in View definition with the remaining actions on the query.

Join in SQL Databases

SQL databases facilitate combining rows from two or more tables, based on the related columns. Combining action uses Join function during a database transaction. Join refers to a clause which combines.

For example. consider an SQL statement:

Select KitKat Sales from TransactionsTbl INNER JOIN ACVMSalesTbl ON TransactionsTbl. KitKatSales = TransactionsTbl. KitKatSales;

The statement selects those records in a column named KitKatSales which match the values in two tables: one Transactions Tbl and other ACVMSalesTbl.

NoSQL:

A new category of data stores is NoSQL (means Not Only SQL) data stores. NoSQL is an altogether new approach of thinking about databases, such as schema flexibility, simple relationships, dynamic schemas, auto sharding, replication, integrated caching, horizontal scalability of shards, distributable tuples, semi structures data and flexibility in approach.

Issues with NoSQL data stores are lack of standardization in approaches, processing difficulties for Complex queries, dependence on eventually consistent results in place of consistency in all states.

Big Data NoSQL or Not-0nly SQL

NoSQL DB does not require specialized RDBMS like storage and hardware for processing. Storage can be on a cloud. NoSQL records are in non-relational data store systems.

NoSQL data stores are considered as semi-structured data. Big Data Store uses NoSQL.

NoSQL data store characteristics are ae follows:

1. NoSQL is a class of non-relational data storage system with flexible data model. Examples of NoSQL data-architecture patterns of datasets are key-value pairs, name/value pairs, Column family Big-data store, Tabular data store, Cassandra (used in Facebook/Apache), HBase, hash table [Dynama (Amazon S3)], unordered keys

using JSON (CouchDB), JSON (PNUTS), JSON (MongoDB), Graph Store, Object Store, ordered keys and semi-structured data storage systems.

2. NoSQL not necessarily has a fixed schema, such as table; do not use the concept of Joins in distributed data storage systems); Data written at one node can be replicated to multiple nodes. Data store is thus fault-tolerant. The store can be partitioned into unshared shards.

   Features in NoSQL Transactions NoSQL transactions have following features:

   (i) Relax one or more of the ACID properties.

   (ii) Characterize by two out of three properties (consistency, availability and partitions) of CAP theorem, two are at least present for the application/service/process.

   iii) Can be characterized by BASE properties

Big Data NoSQL Solutions

Big Data NoSQL Solutions NoSQL DBs are needed for Big Data solutions. They play an important role in handling Big Data challenges. The following table gives the examples of widely used NoSQL data stores:

Table 3.1 NoSQL data stores and their characteristic features

NoSQL Data store	Description
Apache's HBase	HDFS compatible, open-source and non-relational data store written in Java; A column-family based NoSQL data store, data store providing BigTable-like capabilities (Sections 2.6 and 3.3.3.2); scalability, strong consistency, versioning, configuring and maintaining data store characteristics
Apache's MongoDB	HDFS compatible; master-slave distribution model (Section 3.5.1.3); document-oriented data store with JSON-like documents and dynamic schemas; open-source, NoSQL, scalable and non-relational database; used by Websites Craigslist, eBay, Foursquare at the backend
Apache's Cassandra	HDFS compatible DBs; decentralized distribution peer-to-peer model (Section 3.5.1.4); open source; NoSQL; scalable, non-relational, column-family based, fault-tolerant and tuneable consistency (Section 3.7) used by Facebook and Instagram
Apache's CouchDB	A project of Apache which is also widely used database for the web. CouchDB consists of Document Store. It uses the JSON data exchange format to store its documents, JavaScript for indexing, combining and transforming documents, and HTTP APIs
Oracle NoSQL	Step towards NoSQL data store; distributed key-value data store; provides transactional semantics for data manipulation, horizontal scalability, simple administration and monitoring
Riak	An open-source key-value store; high availability (using replication concept), fault tolerance, operational simplicity, scalability and written in Erlang

CAP Theorem

Among C, A and P, two are at least present for the application/service/process.

- **Consistency** means all copies have the same value like in traditional DBs.
- **Availability** means at least one copy is available in case a partition becomes inactive or fails.
- **Partition** means parts which are active but may not cooperate (share) as in distributed DBs.

1. Consistency in distributed databases means that all nodes observe the same data at the same time. Therefore, the operations in one partition of the database should reflect in other related partitions in case of distributed database.
2. Availability means that during the transactions, the field values must be available in other partitions of the database so that each request receives a response on success as well as failure. Distributed databases require transparency between one another. Network failure may lead to data unavailability in a certain partition in case of no replication. Replication ensures availability.
3. Partition means division of a large database into different databases without affecting the operations on them by adopting specified procedures.

Partition tolerance: Refers to continuation of operations as a whole even in case of message loss, node failure or node not reachable.

Brewer's CAP (Consistency, Availability and Partition Tolerance) theorem:

Demonstrates that any distributed system cannot guarantee C, A and P together.

1. Consistency- All nodes observe the same data at the same time.

2. Availability- Each request receives a response on success/failure.

3. Partition Tolerance-The system continues to operate as a whole even in case of message loss, node failure or node not reachable.

Partition tolerance cannot be overlooked for achieving reliability in a distributed database system.

In case of any network failure, a choice can be:

- Database must answer, and that answer would be old or wrong data (AP).
- Database should not answer, unless it receives the latest copy of the data (CP).

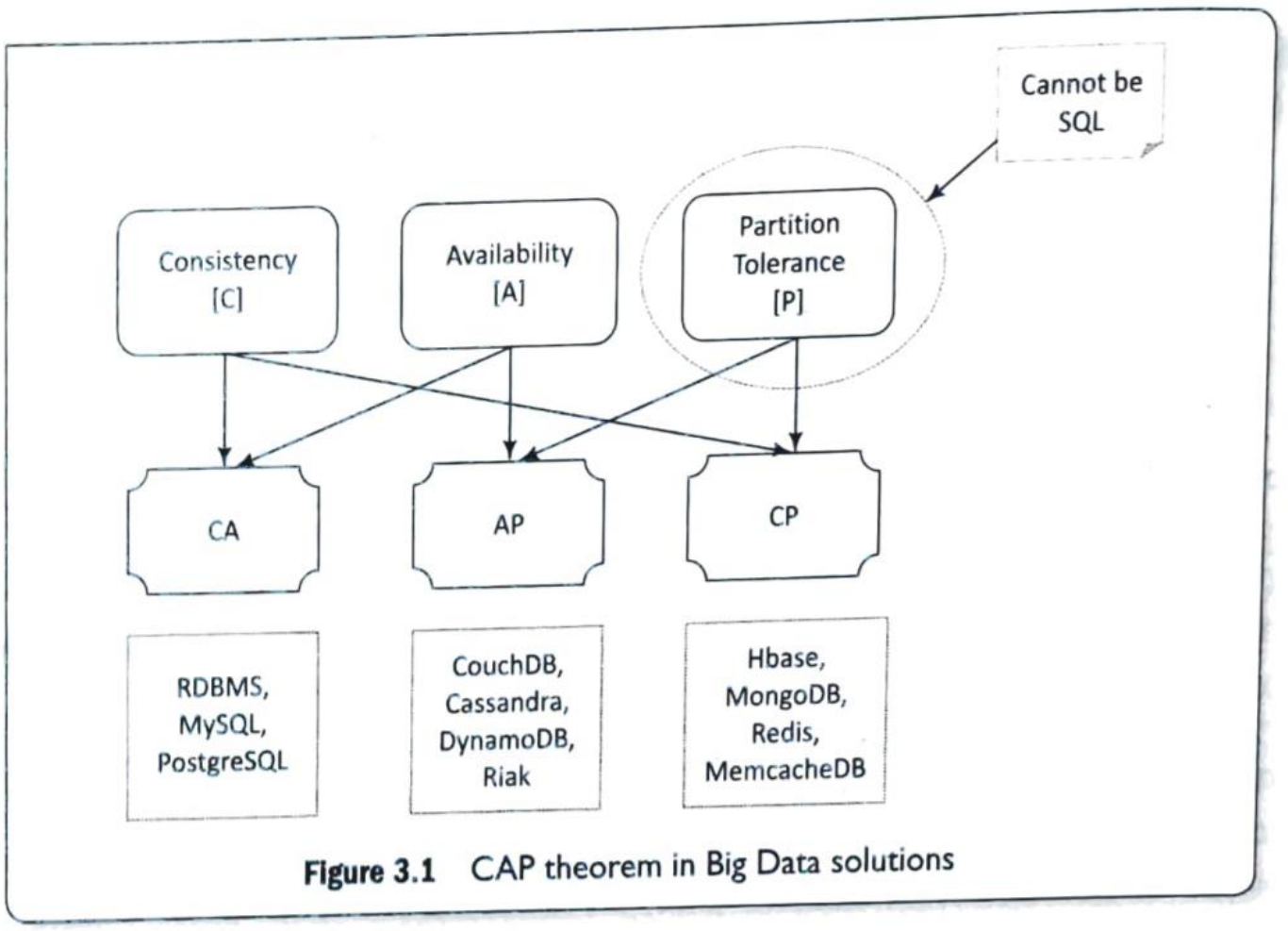


Figure 3.1 CAP theorem in Big Data solutions

The CAP theorem implies that for a network partition system, the choice of consistency and availability are mutually exclusive. CA means consistency and availability; AP means availability and partition tolerance and CP means consistency and partition tolerance. The above figure shows the CAP theorem usage in Big Data Solutions.

Schema-less Models:

Schema of a database system refers to designing of a structure for datasets and data structures for storing into the database. NoSQL data not necessarily have a fixed table schema. The systems do not use the concept of Join (between distributed datasets).

Data written at one node replicates to multiple nodes. Therefore, these are identical, fault-tolerant and partitioned into shards. Distributed databases can store and process a set of information on more than one computing nodes.

NoSQL data model offers relaxation in one or more of the ACID properties (Atomicity, consistence, isolation and durability) of the database. Distribution follows CAP theorem. CAP theorem states that out of the three properties, wo must at least be present for the application/service/process.

Figure 3.2 shows characteristics of Schema-less model for data stores. ER stands for entity-relation modelling.

Relations in a database build the connections between various tables of data. For example, a table of subjects offered in an academic programme can be connected to a table of programmes offered in the academic institution.

Metadata refers to data describing and specifying an object or objects. Metadata is a record with all the information about a particular dataset and the inter-linkages. Metadata helps in selecting an object, specifications of the data and, usages that design where and when.

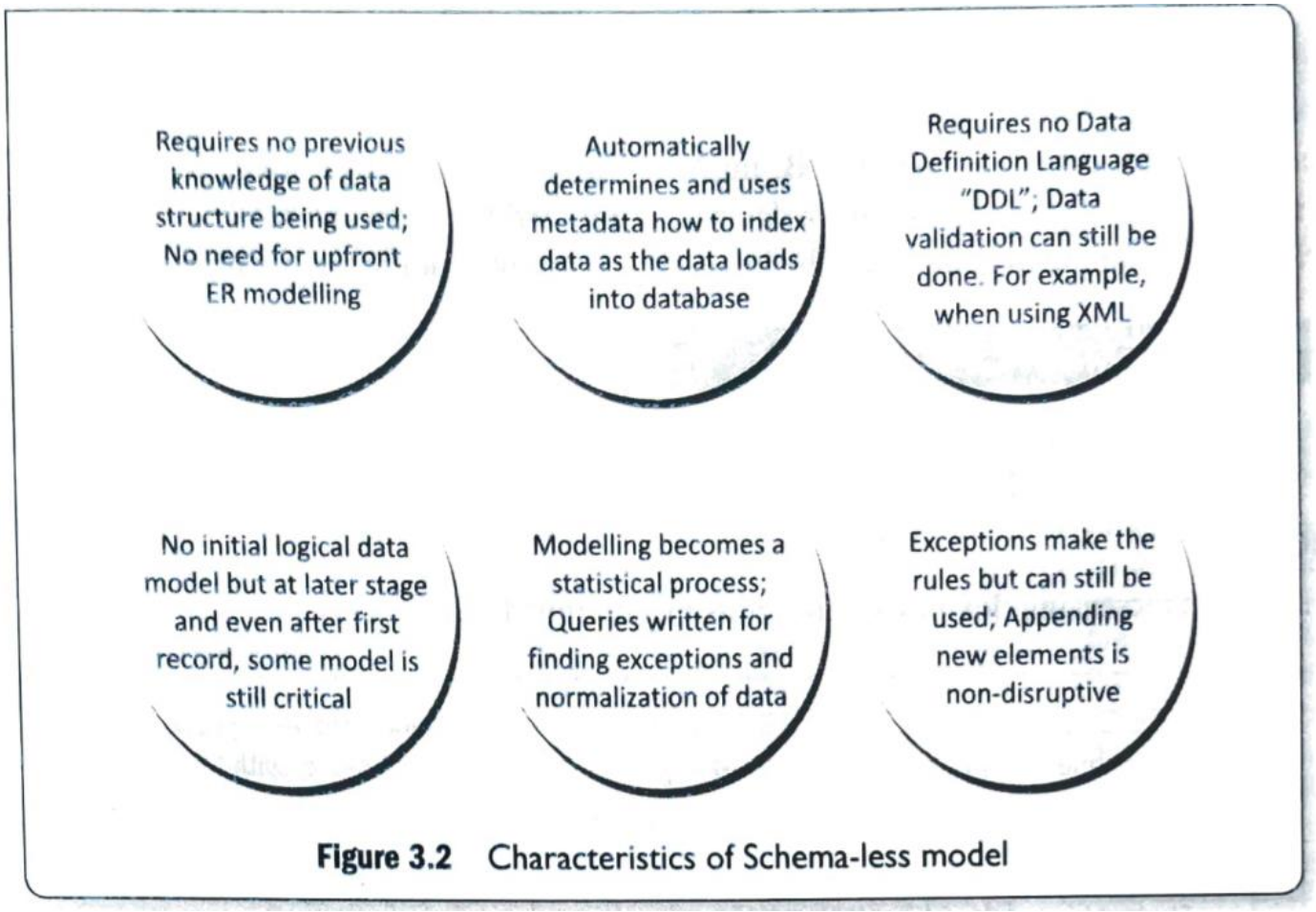


Figure 3.2 Characteristics of Schema-less model

Increasing Flexibility for Data Manipulation:

Consider database 'Contacts'. They follow a fixed schema. Now consider students' admission database That also follow a fixed schema. Later, additional data is added as the course progresses. NoSQL data store characteristics are schema-less.

NoSQL data store possess characteristic of increasing flexibility for data manipulation. The new attributes to database can be increasingly added. Late binding of them is also permitted. BASE is a flexible model for NoSQL data stores. Provisions of BASE increase flexibility.

BASE Properties:

BA stands for basic availability, S stands for soft state and E stands for eventual consistency.

1. Basic availability ensures by distribution of shards (many partitions of huge data store) across many data nodes with a high degree of replication. Then, a segment failure does not necessarily mean a complete data store unavailability.
2. Soft state ensures processing even in the presence of inconsistencies but achieving consistency eventually. A program suitably takes into account the inconsistency found during processing. NoSQL database design does not consider the need of consistency all along the processing time.
3. Eventual consistency means consistency requirement in NoSQL databases meeting at some point of time in future. Data converges eventually to a consistent state with no time-frame specification for achieving that. ACID rules require consistency all along the processing on completion of each transaction. BASE does not have that requirement and has the flexibility.

BASE model is not necessarily appropriate in all cases but it is flexible and is an alternative to SQL-lik adherence to ACID properties.

Following is an example to understand the increasing flexibility for data manipulation:

Example :

Use examples of database for the students in various university courses to demonstrate the concept of increasing flexibility in NoSQL DBs.

Given figure shows increasing flexibility concept using additional data models:

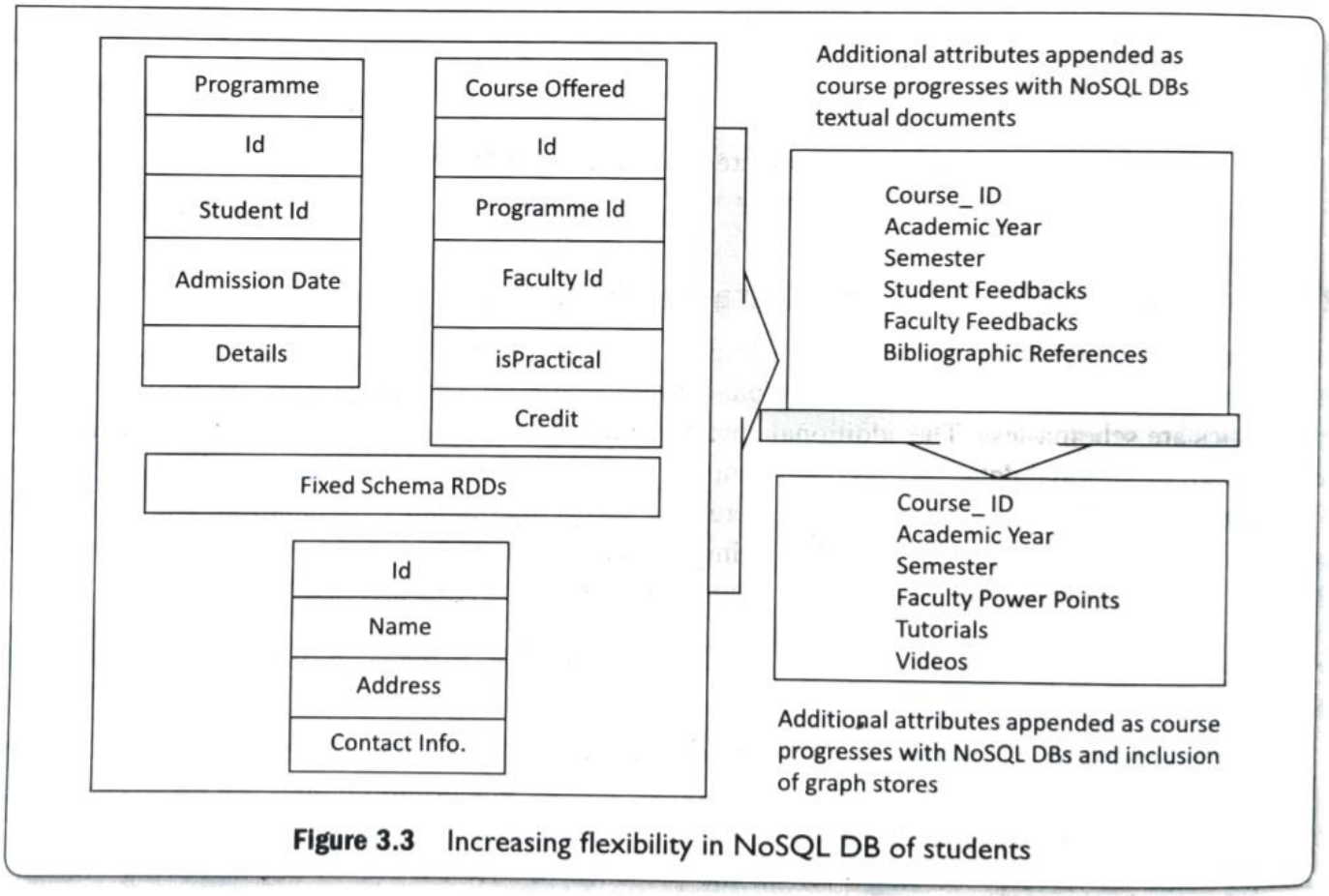


Figure 3.3 Increasing flexibility in NoSQL DB of students

NOSQL DATA ARCHITECTURE PATTERNS:

NoSQL data stores broadly categorize into architectural patterns described in the following subsections:

Key-Value Store:

The simplest way to implement a schema-less data store is to use key-value pairs. The data store characteristics are high performance, scalability and flexibility. Data retrieval is fast in key-value pairs data store. The concept is similar to a hash table where a unique key points to a particular item(s) of data. Figure 3.4 shows key-value pairs architectural pattern and example of students' database as key-value pairs.

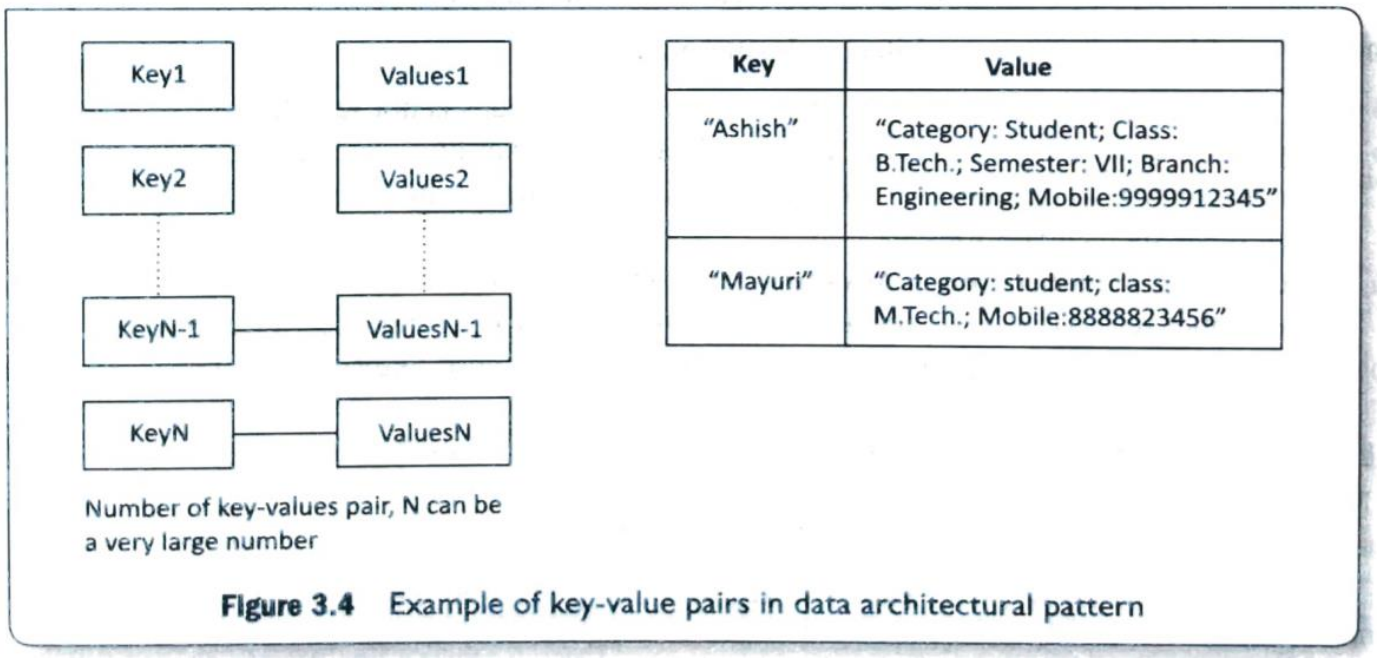


Figure 3.4 Example of key-value pairs in data architectural pattern

Advantages of a key-value store are as follows:

1. Data Store can store any data type in a value field. The key-value system stores the information as a BLOB of data (such as text, hypertext, images, video and audio) and return the same BLOB when the data is retrieved. Storage is like an English dictionary.
2. A query just requests the values and returns the values as a single item. Values can be of any data type.
3. Key-value store is eventually consistent.
4. Key-value data store may be hierarchical or may be ordered key-value store.
5. Returned values on queries can be used to convert into lists, table-columns, data-frame fields and columns.
6. Have (i) scalability, (ii) reliability, (ii) portability and (iv) low operational cost.
7. The key can be synthetic or auto-generated. The key is flexible and can be represented in many formats: () Artificially generated strings created from a hash of a value, (ii) Logical path names to images or files, (ii) REST web-service calls (request response cycles), and (iv) SQL queries.

   The key-value store provides client to read and write values using a key as follows:

   (i) Get (key), returns the value associated with the key.
   (ii) Put (key, value), associates the value with the key and updates a value if this key is already present.
   (iii) Multi-get (key1, key2, . ., keyN) , returns the list of values associated with the list of keys.
   (iv) Delete (key), removes a key and its value from the data store.

   Limitations of key-value store architectural pattern are:

   (i) No indexes are maintained on values, thus a subset of values is not searchable.
   (ii) Key-value store does not provide traditional database capabilities, such as atomicity of transactions, or consistency when multiple transactions are executed simultaneously. The application needs to implement such capabilities.
   (iii) Maintaining unique values as keys may become more difficult when the volume of data increases. One cannot retrieve a single result when a key-value pair is not uniquely identified.
   (iv) Queries cannot be performed on individual values. No clause like 'where' in a relational database usable that filters a result set.

A comparison between traditional relational data model with the key-value store mode:

Traditional relational model	Key-value store model
Result set based on row values	Queries return a single item
Values of rows for large datasets are indexed	No indexes on values
Same data type values in columns	Any data type values

Typical uses of key-value store are: i) Image store, (i) Document or file store, (iii) Lookup table. and iv) Query-cache.

Concept of Hash Key:

The following example explains the hash and key-value pairs associated with a hash in traditional data.

Example :

Consider an example. Assume that student name is key, *k*. Each student grade sheet entry has a number of values or set of (secondary) key-value pairs. For example, semester grade point average (SGPAs) values and cumulative grade point average (CGPA) value. How will the hash function be used?

SOLUTION

A hash function generates an index, I_k for k. I_k should ideally be unique and should uniquely map to k. I_k is a number with few digits only, compared to a number of characters (0-255 bytes) in the main key k used as input for the hash function. Assume that total 20 numbers of entries are present between slots indices between 00 to 99. Student name may consist of several characters, but index will be just two digits.

Document Store:

Characteristics of Document Data Store are high performance and flexibility. Scalability varies, depends on stored contents. Complexity is low compared to tabular, object and graph data stores.

Following are the features in Document Store:

1. Document stores unstructured data.

2. Storage has similarity with object store.

3. Data stores in nested hierarchies. For example, in JSON formats data model [Example 3.3(ii)],. XML document object model (DOM), or machine-readable data as one BLOB. Hierarchical information stores in a single unit called document tree. Logical data stores together in a unit.

4. Querying is easy. For example, using section number, sub-section number and figure caption and table headings to retrieve document partitions.

5. No object relational mapping enables easy search by following paths from the root of document tree.

6. Transactions on the document store exhibit ACID properties.

Typical uses of a document store are: (i) office documents, (ii) inventory store, (iii) forms data, (iv) document exchange and (v) document search.

The following example explains the CSV and JSON object concept and aspects of CSV and JSON file formats:

Example :

Assume Preeti gave examination in Semester 1 in 1995 in four subjects. She gave examination in five subjects in Semester 2 and so on in each subsequent semester. Another student, Kirti gave in Semester 1 examination in 2016 in three subjects, out of which one was theory and two were Presume practical subjects. the subject names and grades awarded to them.

(i) Write two CSV files for cumulative grade-sheets for both the students. Point the difficulty during processing of data in these two files.

(ii) Write a file in JSON format with each student grade-sheet as an object instance. How does the object-oriented and hierarchical data record in JSON make processing easier?

SOLUTION

(i) Two CSV file for cumulative grade-sheets are as follows:
CSV file for Preeti consists of the following nine lines each with four columns:
Semester, Subject Code, Subject Name, Grade

1, CS101, ""Theory of Computations"", 7.8.
1, CS102,1, ""Computer Architecture"", 7.8.
.....
2, CS204, ""Object Oriented Programming"", 7.2.
2, CS205, ""Data Analytics"", 8.1.

The CSV file for Kirti consist of following five lines each with five columns: Semester, Subject Type, Subject Code, Subject Name, Grade

1, Theory, EL101, ""Analog Electronics"", 7.6.
1, Theory, EL102,1, ""Principles of Analog Communication"", 7.5.
1, Theory, EL103, ""Digital Electronics"", 7.8.
1, Practical, CS104, ""Analog ICs"", 7.2
1, Practical, CS105, ""Digital ICs"", 8.4

A column head is a key. Number of key-value pairs are $(4 \times 9) = 36$ for preetiGradeSheet.csv and $(5 \times 5) = 25$ for kirtiGradeSheet.csv. Therefore, when processing student records, merger of both files into a single file will need a program to extract the key-value pairs separately, and then prepare a single file.

(ii) JSON gives an advantage of creating a single file with multiple instances and inheritances of an object. Consider a single JSON file, *studentGradeSheets.json* for cumulative grade-sheets of many students. *Student_Grades* object is top in the hierarchy. Each *student_name* object is next in the hierarchy with object consisting of student name, each with number of instances of subject codes, subject types, subject titles and grades awarded. Each student name object-instance extends in student grades object-instances.

The database stores and retrieves documents, such as XML, JSON, BSON (Binary-encoded Script Object Notation (for objects). The documents are self-describing, hierarchical tree-structured consisting of maps. collections and scalar values. The documents stored are similar to each other but do not have to be the same. Some of the popular document data stores are CouchDB, MongoDB, Terrastore, OrientDB and RavenDB.

Certain NoSQL DBs enable ACID rule-based transactions also. Examples of document data stores are MongoDB, Apache Couchbase and MarkLogic. CouchDB uses the JSON store data, HTTP APIs for connectivity, JavaScript for the query language and MapReduce for processing.

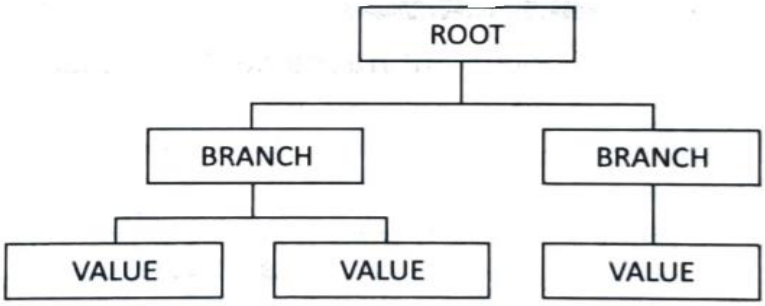
Document JSON Format CouchDB Database Apache CouchDB is an open-source database. Its features are:

1. CouchDB provides mapping functions during querying, combining and filtering of information.
2. CouchDB deploys JSON Data Store model for documents. Each document maintains separate data and metadata (schema).
3. CouchDB is a multi-master application. Write does not require field locking when controlling the concurrency during multi-master application.
4. CouchDB querying language is JavaScript. Java script is a language which documents use to transform.
5. CouchDB queries the indices using a web browser. CouchDB accesses the documents using HTTP API.
6. CouchDB data replication is the distribution model that results in fault tolerance and reliability.

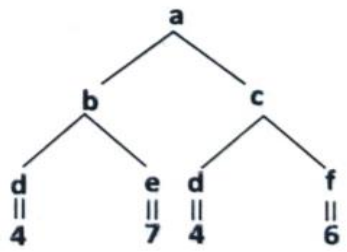
Document JSON Format-MongoDB Database MongoDB Document database provides a rich query language and constructs, such as database indexes allowing easier handling of Big Data.


```
{
    "id": "1001"
    "Student Name"
    {
     First": "Ashish",
    Middle": "Kumar",
     Last": "Rai"
    }
    Category": "Student",
     Class":"B. Tech.",
     Semester" "VII",
    "Branch": "Computer Engineering",
     "Mobile""12345"
}
```


Document Architecture Pattern and Discovering Hierarchical Structure Following is example of an XML document in which a hierarchical structure discovers later. Below figure shows an XML document architecture pattern in a document fragment and document tree structure.




```
<a>
  <b>
      <d> 4 </d>
      <e> 7 </e>
  </b>
  <c>
      <d> 4 </d>
      <f> 6 </f>
  </c>
</a>
```



(a) XML document fragment (b) Tree representation of fragment

Figure 3.5 XML document architecture pattern

XML and JSON both are designed to form a simple and standard way of describing different kinds of hierarchical data structures. They are popularly used for storing and exchanging data.

The following example explains the concept of Document Store in JSON and XML for hierarchical records.

Give the structures of XML and JSON document fragments for a student record.

SOLUTION

Following are the structures:

```
{
    "students": [
    {
        "name": "Ashish Jain",
        "rollNo": "12345"
    },
    {
        "name": "Sandeep Joshi",
        "rollNo": "12346"
    }
    ]
}
```

(a) JSON

```
<students>
    <student>
        <name>Ashish Jain</name>
        <rollNo>12345</rollNo>
    </student>
    <student>
        <name>Sandeep Joshi</name>
        <rollNo>12346</rollNo>
    </student>
</students>
```

(b) XML equivalent

When compared with XML, JSON has the following advantages:

- XML is easier to understand but XML is more verbose than JSON.
- XML is used to describe structured data and does not include arrays, whereas JSON includes arrays.

- ❖ JSON has basically key-value pairs and is easier to parse from JavaScript.
- ❖ The concise syntax of JSON for defining lists of elements makes it preferable for serialization of text format objects.

Document Collection:

A collection can be used in many ways for managing a large document store. Three uses of a document collection are:

1. Group the documents together, similar to a directory structure in a file-system. (A directory consists of grouping of file folders.)
2. Enables navigating through document hierarchies, logically grouping similar documents and storing business rules such as permissions, indexes and triggers (special procedure on some actions in a database).
3. A collection can contain other collections as well.

Tabular Data:

Tabular data stores use rows and columns. Row-head field may be used as a key which access and retrieves multiple values from the successive columns in that row. The OLTP is fast on in-memory row-format data.

Oracle DBs provide both options: columnar and row format storages. Generally, relational DB store is in-memory row-based data, in which a key in the first column of the row is at a memory address, and values in successive columns at successive memory addresses.

In-memory column-based data has the keys (row-head keys) in the first column of each row at successive memory addresses. The next column of each row after the key has the values at successive memory addresses. The values in the third column of each row are at the next memory addresses in succession, and so on up to N columns. The N can be a very large number. The column-based data makes the OLAP easier.

Following subsections describe NoSQL format data stores based on tabular formats:

Column Family Store:

Columnar Data Store:

A way to implement a schema is the divisions into columns. Storage of each column, successive values is at the successive memory addresses. Analytics processing (AP) In-memory uses columnar storage in memory. A pair of row-head and column-head is a key-pair.

Column-Family Data Store:

Column-family data-store has a group of columns as a column family. A combination of row-head, column-family head and table-column head can also be a key to access a field in a column of the table during querying. Combination of row head, column families head, column-family head and column head for values in column fields can also be a key to access fields of a column. A column-family head is also called a super-column head.

Examples of columnar family data stores are HBase, BigTable, HyperTable and Cassandra. The following example explains a column-family data store and why OLAP is fast in-memory column data store in memory:

Assume in-memory columnar storage. Data for a large number of ACVMs with an ACVM_ID each, store in column 1. Data for each day sales at each ACVM for KitKat, Milk, Fruit and Nuts, Nougat and Oreo store in Columns 2 to 6. Each row has six cells (ID +five sales data).

(i) How do the column key values store in memory?
(ii) How do the values store in the memory in columnar storage format?
(iii) How does analytics of each day's sales help?
(iv) Why do in-memory columnar storage result in fast computations during analytics?
(v) How are a column family and column-family head (key) specified?
(vi) How do a column-families group specify?
(vii) How do row groups form? What is the advantage of division into sub-groups?

SOLUTION

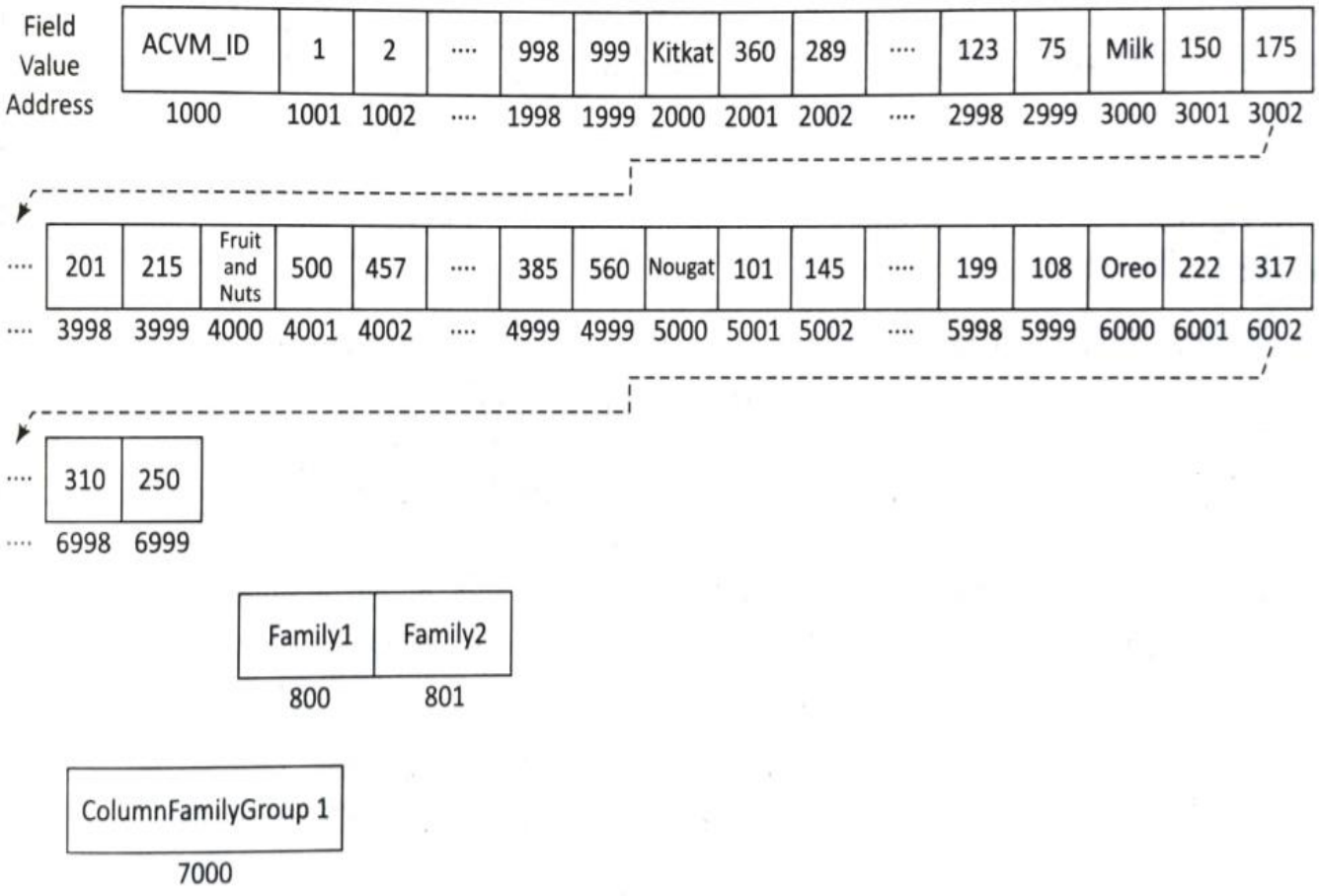
Assume the following columnar storage at memory:

(i) Column and row keys
Addresses 1000, 2000, 3000, …., 6000 save the column keys. Address 1000 stores string 'ACVM_ ID'. Then the adresses 1001, 1002, …., 1999 store the row keys, means ACVM_IDs.
Chocolate name of five flavours store at addresses 2000, 3000, 4000, 5000, and 6000.

(ii) Column field values
Column 1 ACVM_IDs store at address 1001 to 1999 for 999 ACVMs. Sales in a day for KitKat, Milk, Fruit and Nuts, Nougat and Oreo store at addresses 2001 to 2999, 3001 to 3999, 4001 to 4999, 5001 to 5999, and 6001 to 6999.
Table 3.3 gives sample values in the columns for a day's sales data. The table also gives the keys for row groups, rows (ACVM_IDs), a column family group, two column families and five column heads for 5 flavours of chocolates. The table gives a row group of just 100 rows, just for the sale of assumption.
Figure 3.6 shows fields in columnar storage and addresses in memory. The figure shows ACVM_ IDs as well as each day's sales of each flavour of chocolate at 999 ACVMs. Following are the addresses assigned to the values in fields of Table 3.3:

Table 3.3 Each day's sales of chocolates on 999 ACVMs

		Nestle Chocolate Flavours Group				
		Popular Flavours Family		Costly Flavours Family		
	ACVM_ID	KitKat	Milk	Fruit and Nuts	Nougat	Oreo
Row-group_1 for IDs 1 to 100	1	360	150	500	101	222
	2	289	175	457	145	317
	....	....	....		....	....
Row-group_m for IDs 901 to 999	....	....	....		....	....
	998	123	201	385	199	310
	999	75	215	560	108	250



Fields in columnar storage and addresses in memory.

Characteristics of Columnar Family Data Store:

Columnar family data store imbibes characteristics of very high performance and scalability, moderate level of flexibility and lower complexity when compared to the object and graph databases. Advantages of column stores are:

1. Scalability: The database uses row IDs and column names to locate a column and values at the column fields. The interface for the fields is simple. The back-end system can distribute queries over a large number of processing nodes without performing any Join operations.
2. Partitionability: For example, large data of ACVMs can be partitioned into datasets of size, say 1 MB in the number of row-groups. Values in columns of each row-group, process in-memory at a partition.
3. Availability: The cost of replication is lower since the system scales on distributed nodes efficiently. The lack of Join operations enables storing a part of a column- family matrix on remote computers.
4. Tree-like columnar structure consisting of column-family groups, column families and columns. 4. The columns group into families. The column families group into column groups (super columns).
5. Adding new data at ease: Permits new column Insert operations. Trigger operation creates new 5. columns on an Insert. The column-field values can add after the last address in memory if the column structure is known in advance.
6. Querying all the field values in a column in a family, all columns in the family or a group of column-families, is fast in in-memory column-family data store.
7. Replication of columns: HDFS-compatible column-family data stores replicate each data store with default replication factor = 3.
8. No optimization for Join: Column-family data stores are similar to sparse matrix data. The data do not optimize for Join operations.

Big Table Data Store:

Widely used column-family data store are Google's BigTable, HBase and Cassandra.

Following are features of a BigTable:

1. Massively scalable NoSQL. BigTable scales up to 10s of petabytes.
2. Integrates easily with Hadoop and Hadoop compatible systems.
3. Compatibility with MapReduce, HBase APIs which are open-source Big Data platforms.
4. Key for a field uses not only row_ID and Column_ID.
5. Handles million of operations per second.
6. Handle large workloads with low latency and high throughput.
7. Consistent low latency and high throughput.
8. APIs include security and permissions.
9. BigTable, being Google's cloud service, has global availability and its service is seamless.

The following example explains the use of rowID, ColumlD and Column attributes in BigTable format:

Consider Example 3.6. Consider column fields which have keys to access a field not only by row ID and Column ID but also include the timestamp and attributes in a row. Show the column-keys for accessing column fields of a column.

SOLUTION

Table 3.4 gives keys for each day's sales of KitKat chocolates at ACVMs. First row-headings are the column-keys.

Table 3.4 Each day's sales of KitKat chocolates at ACVMs

Column-keys →	ACVM_ID	KitKatSalesDate	Timestamp	KitKatSalesNumber

RC File Format:

Hive uses Record Columnar (RC) file-format records for querying. RC is the best choice for intermediate tables for fast column-family store in HDFS with Hive. Serializability of RC table column data is the advantage. RC file is DeSerializable into column data. (Serializability means query or transaction executable by series of instructions such that execution ensures correct results).

The following example explains the use of row groups in the RC file format for column of a row group:

Example:

Practically, row groups have millions of rows and in-memory between 10 MB and 1 GB. Assume two row groups of just two rows each.

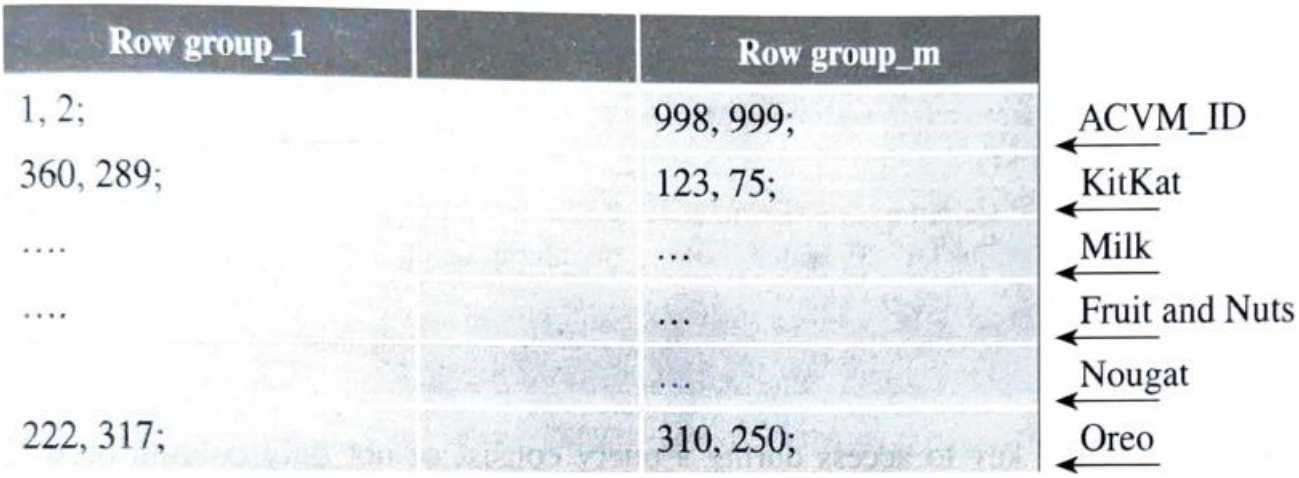
Row-group_1 for IDs 1 to 2					
1	360	150	500	101	222
2	289	175	457	145	317

Row-group_m for IDs 998 to 999					
998	123	201	385	199	310
999	75	215	560	108	250

Make a file in RC format.

SOLUTION:

The values in each column are the records in file for each row group. Each row-group data is like a column of records which stores in the RC file.

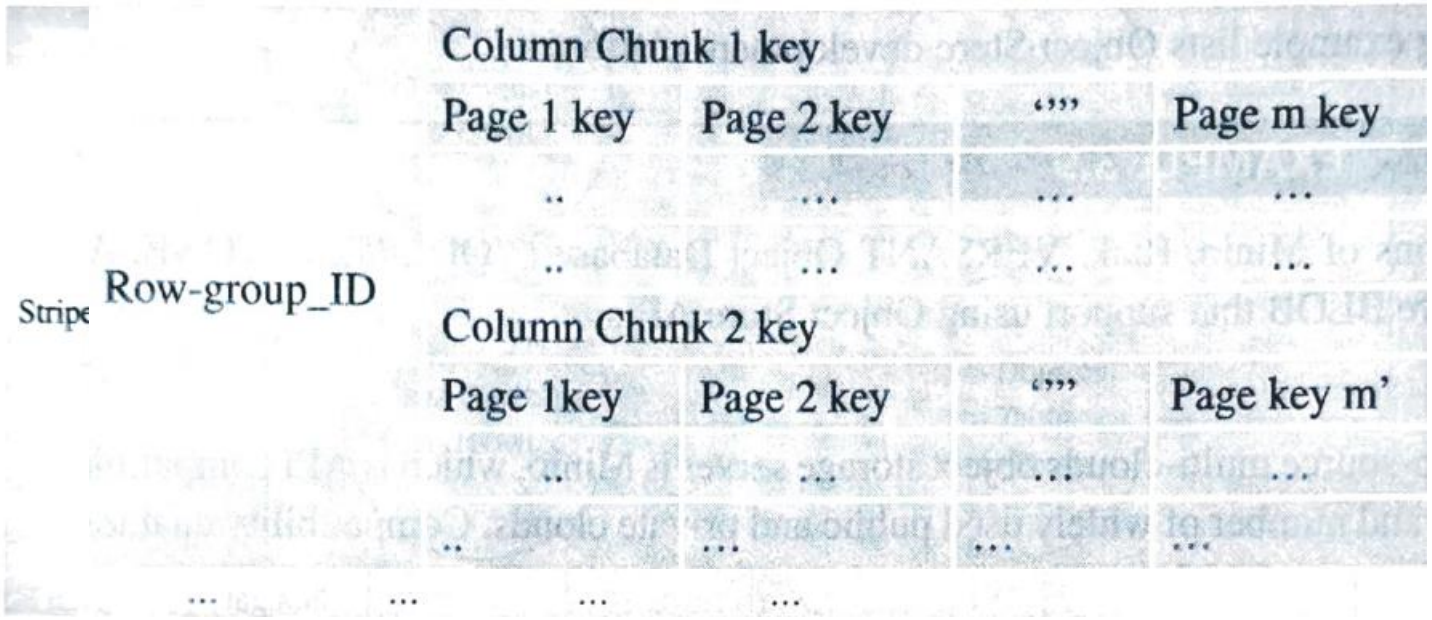


RC file for row group_I will consists of records 1, 2;360, 289; ..., 222,317; on serialization of column records. RC file for row group_m will consists of 998, 999; 123, 75; ..., 310, 250;

ORC File Format:

- An ORC (Optimized Row Columnar) file consists of row-group data called stripes. ORC enables concurrent reads of the same file using separate RecordReaders.
- ORC is an intelligent Big Data file format for HDFS and Hive. An ORC file stores a collections of rows as a row-group. Each row-group data store in columnar format. This enables parallel processing of multiple row-groups in an HDFS cluster.
- An ORC has two sets of columns data instead of one column data in RC. One column is for each map or list size and other values which enable a query to decide skipping or reading of the mapped columns. A mapped column has contents required by the query.
- The columnar layout in each ORC fle thus, optimizes for compression and enables skipping of data in columns. This reduces read and decompression load.
- Lighiweight indexing is an ORC feature. Those blocks of rows which do not match a query skip as they do not map on using indices data at metadata. Each index includes the aggregated values of minimum, maximum, sum and count using aggregation functions on the content columns.
  The keys used to access or skip a contents column during querying. The keys are Stripe_ID, Index-column key, and content column name, min, max and count.

Combination of keys for content page in the Parquet file format



These values do not need to compute again and again using aggregation functions, such as min, max and count.

An ORC thus, optimizes for reading serially the column fields in HDFS environment. The throughput increases due to skipping and reading of the required fields at contents-column key. Reading less number of ORC file content-columns reduces the workload on the NameNode.

Parquet File Formats:

- Parquet is nested hierarchical columnar-storage concept. Nesting sequence is the table, row group, column chunk and chunk page. Apache Parquet file is columnar-family store file.
- processing queries on Big Data. The file compulsorily consists of metadata, though the file need not consist of data.
- The Parquet file consists of row groups. A row-group columns data process in-memory after data cache and buffer at the memory from the disk.
- Each row group has a number of columns. A row group has N columns, and row group consists of N Column chunks. This means each column chunk consists of values saved in each column of each row group.
- An ORC array has two columns, one for array size and the other for contents. Parquet format file does not consist of extra column per nesting level.
  Given table shows the keys used to access or skip the contents page. Three keys are: (i) row-group_ID, i) column-chunk key and (ii) page key.

Object Data Store:

An object store refers to a repository which stores the:

1. Objects (such as files, images, documents, folders, and business reports)
2. System metadata which provides information such as filename, creation_date, last_modified, language_used (such as Java, C, C#, C++, Smalltalk, Python), access_permissions, supported query languages)
3. Custom metadata which provides information, such as subject, category, sharing permissions.

- Metadata enables the gathering of metrics of objects, searches, finds the contents and specifies the objects in an object data-store tree. Metadata finds the relationships among the objects, maps the object relations and trends. Object Store metadata interfaces with the Big Data.
- Document content can be stored in either the object store database storage area or in a file storage area. A single file domain may contain multiple Object Stores.
- Data definition and manipulation, DB schema design, database browsing, DB administration, application compilation and debugging use a programming language.
- Eleven Functions Supporting APls An Object data store consists of functions supporting APls for:

i) scalability

(i) indexing

(iii) large collections

(iv) querying language, processing and optimization

(V) Transactions

(vi) data replication for high availability, data distribution model, data integration (such as with relational dalabase, XMIL, custom code)

(vii) schema evolution

(vii) persistency

(ix) persistent object life cycle

(x) adding modules

(xi) locking and caching strategy

- Object Store may support versioning for collaboration. Amazon S3 and Microsoft Azure RI support the Object Store.
- Amazon S3 (Simple Storage Service) S3 refers to Amazon web service on the cloud named S3. The S3 provides the Object Store. The Object Store differs from the block and file-based cloud storage.

The following example lists Object Store development platforms:

Example :

List the functions of Minio, Riak, VERSANT Object Database (VOD), GEMSTONE, Amazon S3 and Microsoft Azure BLOB that support using Object Store APIs.

SOLUTION

1. An open-source multi-clouds object storage server is Minio, which is API compatible with Amazon S3 API and number of widely used public and private clouds. Compatibility enables data export to S3 and usages of APIs.

2. Riak CS (Cloud Storage) is object storage management software on top of Riak. It models on open-source distributed-database which is Amazon-compliant. This means database exports to S3 and use S3 APls.

3.VOD consists of 11 functions supporting APIs listed above. VOD enables use by multiple concurrent users. VOD supports cross-platform operating systems (OSs), such as Linux, Windows NT, AIX, HP-UX and Solaris (both 32 and 64 bits for all platforms).

4. GEMSTONE Object DB APIs development language is SmallTalk. The platform supports inmemory DBs, object-oriented processing and distributed caches. GEMSTONE provides cross platform support, OSs AIX, Linux, MacOS and Solaris.

Object Relational Mapping:

Example : How does an HTML object and XML based web service relate with tabular data stores?

The object relational mapping of HTML document and XML web services store with a tabular data store.

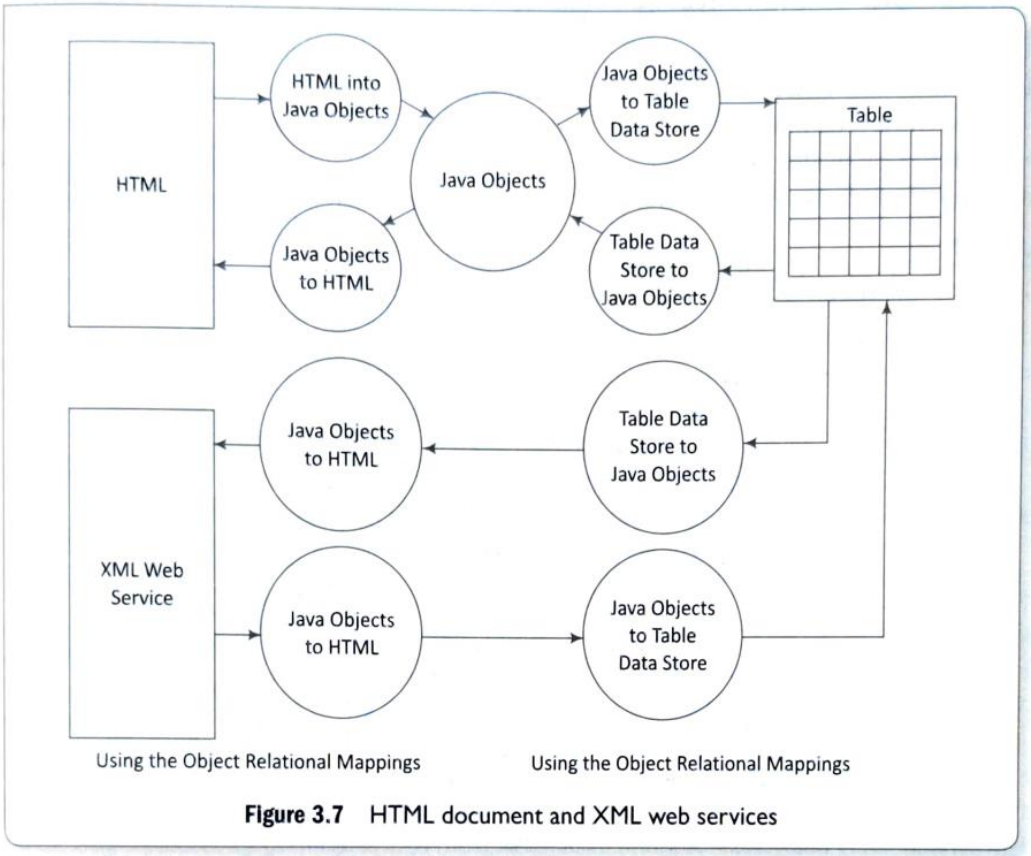


Figure 3.7 HTML document and XML web services

Graph Database:

- One way to implement a data store is to use graph database. A characteristic of graph is high flexibility. Any number of nodes and any number of edges can be added to expand a graph.
- The complexity is high and the performance is variable with scalability. Data store as series of interconnected nodes.
- Graph with data nodes interconnected provides one of the best database system when relationships and relationship types have critical values.
- Data Store focuses on modeling interconnected structure of data. Data stores based on graph theory relation S = (E, V), where E is set of edges e1, e2….. en and V is set of vertices, V1, V2 ., Vn.
- The following example explains the graph database application in describing entities relationships and relationship types.

Example :

Let us asume a car company represents a node entity, which has two connected nodes comprising two model entities, namely Hexa and Zest. Draw graph with directed lines, joining the car company wilth two entities.

(i) How do four directed lines relate to four weeks and two directed lines? One directed line corresponds to a car model. Only directed line corresponds to weekly total sales.

(ii) How will the yearly sales compute?

(iii) Show the path traversals for computations exhibit BASE properties.

SOLUTION:

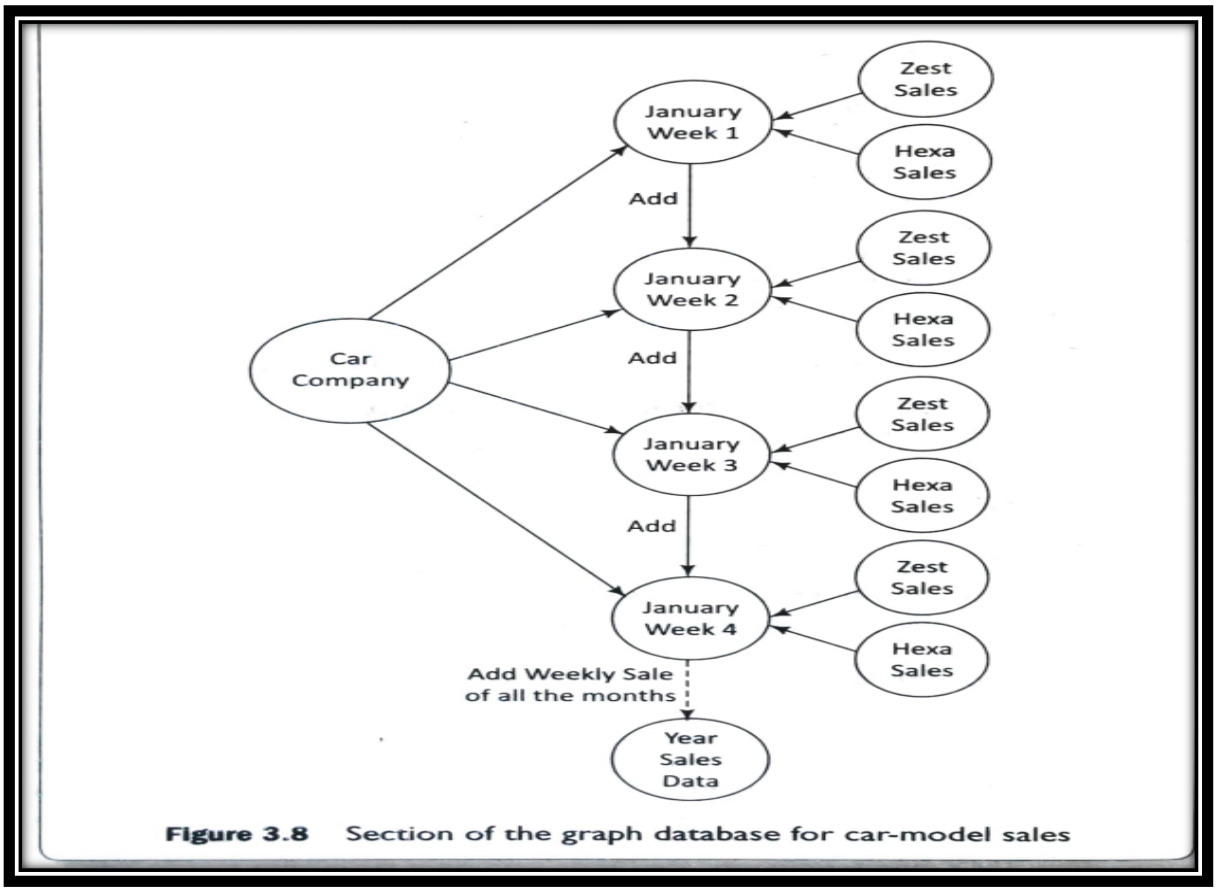


Figure 3.8 Section of the graph database for car-model sales

(i) Figure 3.8 shows section of a graph database for the sales of two car models.

(ii) The yearly sales compute by path traversals from nodes for weekly sales to yearly sales data.

(iii) The path traversals exhibit BASE properties because during the intermediate paths, consistency is not maintained. Eventually when all the path traversals complete, the data becomes consistent.

Characteristics of graph databases are:

1. Use specialized query languages, such as RDF uses SPARQL
2. Create a database system which models the data in a completely different way than the key-values, document, columnar and object data store models.
3. Can have hyper-edges. A hyper-edge is a set of vertices of a hypergraph. A hypergraph 1s a generalization of a graph in which an edge can join any number of vertices (not only the neighbouring 1. 3 vertices).
4. Consists of a collection of small data size records, which have complex interactions between graph-nodes and hypergraph nodes. Nodes represent the entities or objects. Nodes use Joins. Node identification can use URI or other tree-based structure. The edge encodes a relationship between the nodes.

Typical uses of graph databases are:

i) link analysis, (ii) friend of friend queries, (iii) Rules and inference, (iv) rule induction and (v) Pattern matching.

- Examples of graph DBs are Neo4J, AllegroGraph, HyperGraph, Infinite Graph. Titan and FlockDB. Neo4J graph database enable easy usages by Java developers.

Variations of NoSQL Architectural Patterns :

- Six data architectures are SQL-able, key-value pairs, in-memory column-family, document, graph and object. Selected architecture may need variations due to business requirements.
- Business requirements are ease of using an architecture and long-term competitive advantage.
- The following example explains the requirements for the database of students of a University that offers multiple courses in their various academic programmes for several years:

  Example :

  List the selection requirements for the database of University students in successive years. The University runs various Under Graduate and Post Graduate programmes. Students are registered to Multiple courses in a programme.

  SOLUTION:

  Following are the selection requirements:

  1. Scalability: Since the University archives the data for several years, data store should be scalable.
  2. Search ability: Search of required information needs to be fast.
  3. Quarrying ability: All applications need to query the data. Query retrieves the required data among the Big Data of several years.

4. Security: Database needs security and fault tolerance.
5. Affordability: Open source is a requirement. Interoperability: Needs ease in search from different platforms. Search from any computer operating system, such as Windows, Mac, Linux, Android and iOS should be feasible.
6. Importability: Database needs to import data from other platforms, such as import of slides, video lectures, tutorials, e-books, webinars should be facilitated in store.
7. Transformability: Queries may be written in one language and may require transformation to another language, such as HTML.

NoSQL TO MANAGE BIG DATA:

The following subsections describe how to use a NoSQL data store to manage Big Data:

Using NoSsQL to Manage Big Data:

NoSQL,

(i) limits the support for Join queries, supports sparse matrix like columnar-family

(ii) characteristics of easy creation and high processing speed, scalability and storability of much higher magnitude of data (terabytes and petabytes).

- NoSQL sacrifices the support of ACID properties, and instead supports CAP and BASE properties.
- NoSQL data processing scales horizontally as well vertically.

❖ NoSQL Solutions for Big Data:

Big Data solution needs scalable storage of terabytes and petabytes, dropping of support for database Joins. and storing data differently on several distributed servers (data nodes) together as a cluster. A solution, such as CouchDB, DynamoDB, MongoDB or Cassandra follow CAP theorem.

Characteristics of Big Data NoSQL solution are:

1. High and easy scalability: NoSQL data stores are designed to expand horizontally. Horizontal scaling means that scaling out by adding more machines as data nodes (servers) into the pool of resources (processing, memory, network connections). The design scales out using multi-utility cloud services.
2. Support to replication: Multiple copies of data store across multiple nodes of a cluster. This ensures high availability, partition, reliability and fault tolerance.
3. Distributable: Big Data solutions permit sharding and distributing of shards on multiple clusters which enhances performance and throughput.
4. Usages of NoSQL servers which are less expensive. NoSQL data stores require less management efforts. It supports many features like automatic repair, easier data distribution and simpler data models that makes database administrator (DBA) and tuning requirements less stringent.
5. Usages of open-source tools: NoSQL data stores are cheap and open source. Database implementation is easy and typically uses cheap servers to manage the exploding data.

6. Support to schema-less data model.
7. Support to integrated caching
8. No inflexibility unlike the SQL/RDBMS, NoSQL DBs are flexible (not rigid).

Types of Big Data Problems

Big Data problems arise due to limitations of NoSQL and other DBs. The following types of problems a faced using Big Data solutions:

1. Big Data need the scalable storage and use of distributed servers together as a cluster. The solutions must drop support for the database Joins.
2. NoSQL database is open source and that is its greatest strength but at the same time its greatest weakness also because there are not many defined standards for NoSQL data stores.

Comparison of NoSQL with SQL/RDBMS:

Features	NoSQL Data store	SQL/RDBMS
Model	Schema-less model	Relational
Schema	Dynamic schema	Predefined
Types of data architecture patterns	Key/value based, column-family based, document based, graph based, object based	Table based
Scalable	Horizontally scalable	Vertically scalable
Use of SQL	No	Yes
Dataset size preference	Prefers large datasets	Large dataset not preferred
Consistency	Variable	Strong
Vendor support	Open source	Strong
ACID properties	May not support, instead follows Brewer's CAP theorem or BASE properties	Strictly follows

SHARED-NOTHING ARCHITECTURE FOR BIG DATA TASKS:

- Shared nothing (SN) is a cluster architecture. A node does not share data with any other node.
- Big Data store consists of SN architecture. Big Data store, therefore, easily partitions into shards. A partition processes the different queries on data of the different users at each node independently.
- An SN architecture optimizes massive parallel data processing.

The features of SN architecture are as follows:

1. Independence: Each node with no memory sharing; thus possesses computational self-sufficiency
2. Self-Healing: A link failure causes creation of another link
3. Each node functioning as a shard: Each node stores a shard (a partition of large DBs)
4. No network contention.

Choosing the Distribution Models:

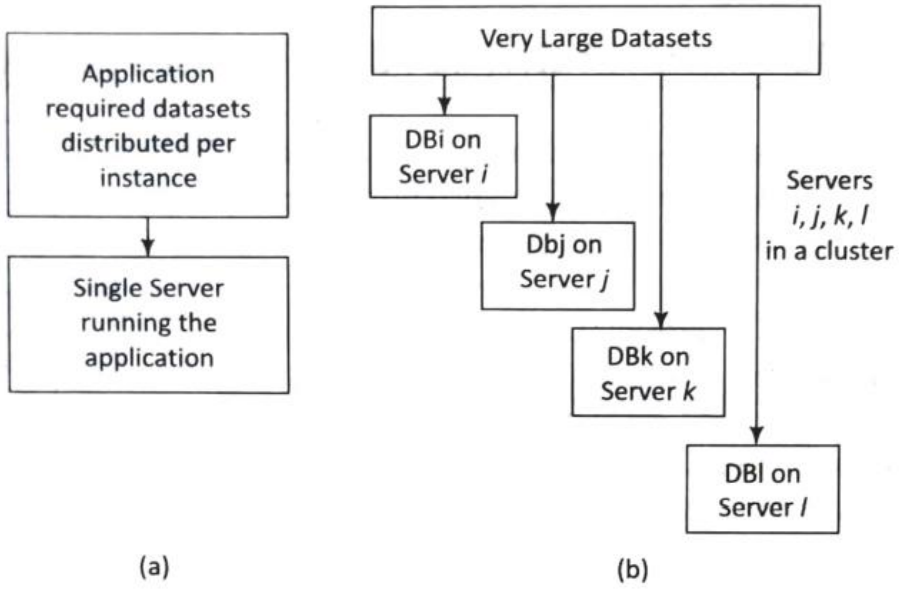
- Big Data requires distribution on multiple data nodes at clusters. Distributed software components give advantage of parallel processing; thus providing horizontal scalability.
- Distribution gives
  (i) ability to handle large-sized data
  (ii) processing of many read and write operations simultaneously in an application.
- A resource manager manages, allocates, and schedules the resources of each processor, memory and network connection. Distribution increases the availability when a network slows or link fails.
- Four models for distribution of the data store are given below:

  Single Server Model :

  Simplest distribution option for NosQL data store and access is Single Server Distribution (SSD) of an application. A graph database processes the relationships between nodes at a server. The SSD model suits well for graph DBs.

  Sharding Very Large Databases:

  Given figure shows sharding of very large datasets into four divisions, each running the application on four I,j, k and l different servers at the cluster. DBi,. DBj, DBk, and DBl, are four shards.

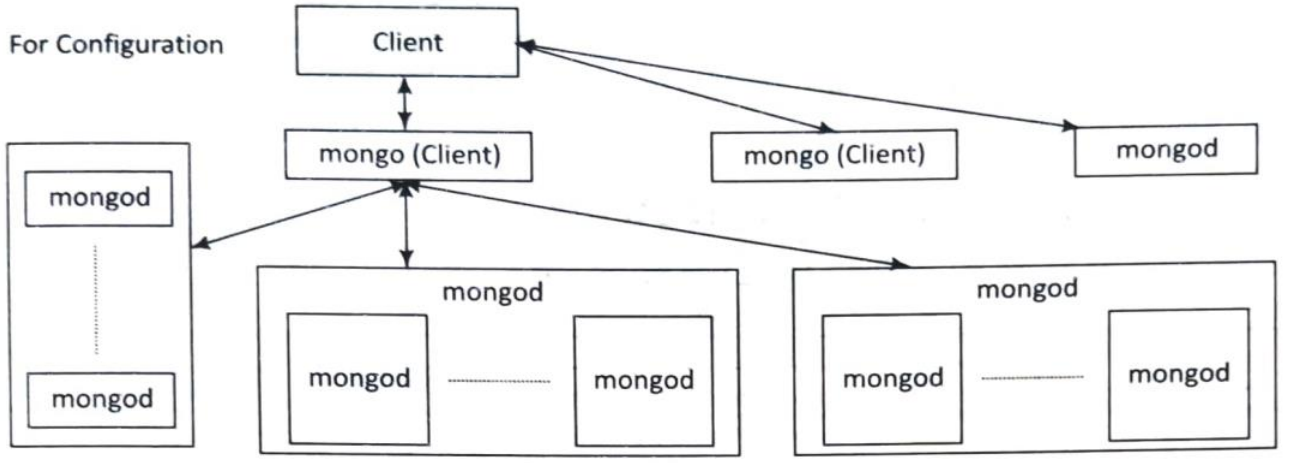


(a) Single server model (b) Shards distributed on four servers in a cluster.

The application programming model in SN architecture is such that an application process runs on multiple shards in parallel.

Master-Slave Distribution Model:

- A node serves as a master or primary node and the other nodes are slave nodes. Master directs the slaves. Slave nodes data replicate on multiple slave servers in Master Slave Distribution (MSD) model.
- MongoDB database server is mongod and the client is mongo.
- Master-Slave Replíication Processing performance decreases due to replication in MSD distribution model.
- Complexity Cluster-based processing has greater complexity than the other architectures.



Master-slave distribution model. Mongo is a client and mongod is the server

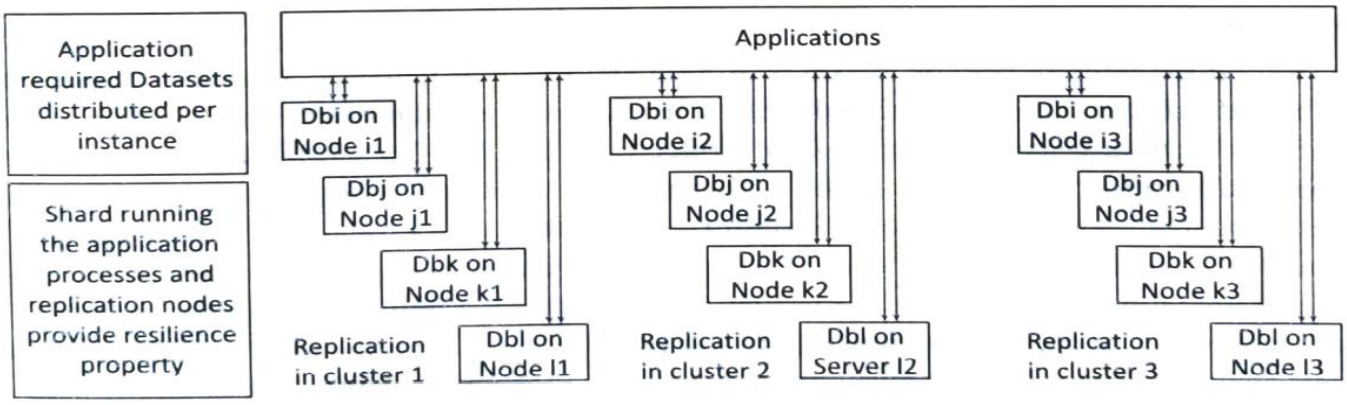
Peer-to-Peer Distribution Model:

Peer-to-Peer distribution (PPD) model and replication show the following characteristics:

(1) All replication nodes accept read request and send the responses.

(2) All replicas function equally. (3) Node failures do not cause loss of write capability, as other replicated node responds.

- ❖ Cassandra adopts the PPD model. The data distributes among all the nodes in a cluster.
- ❖ Following figure shows the PPD model:

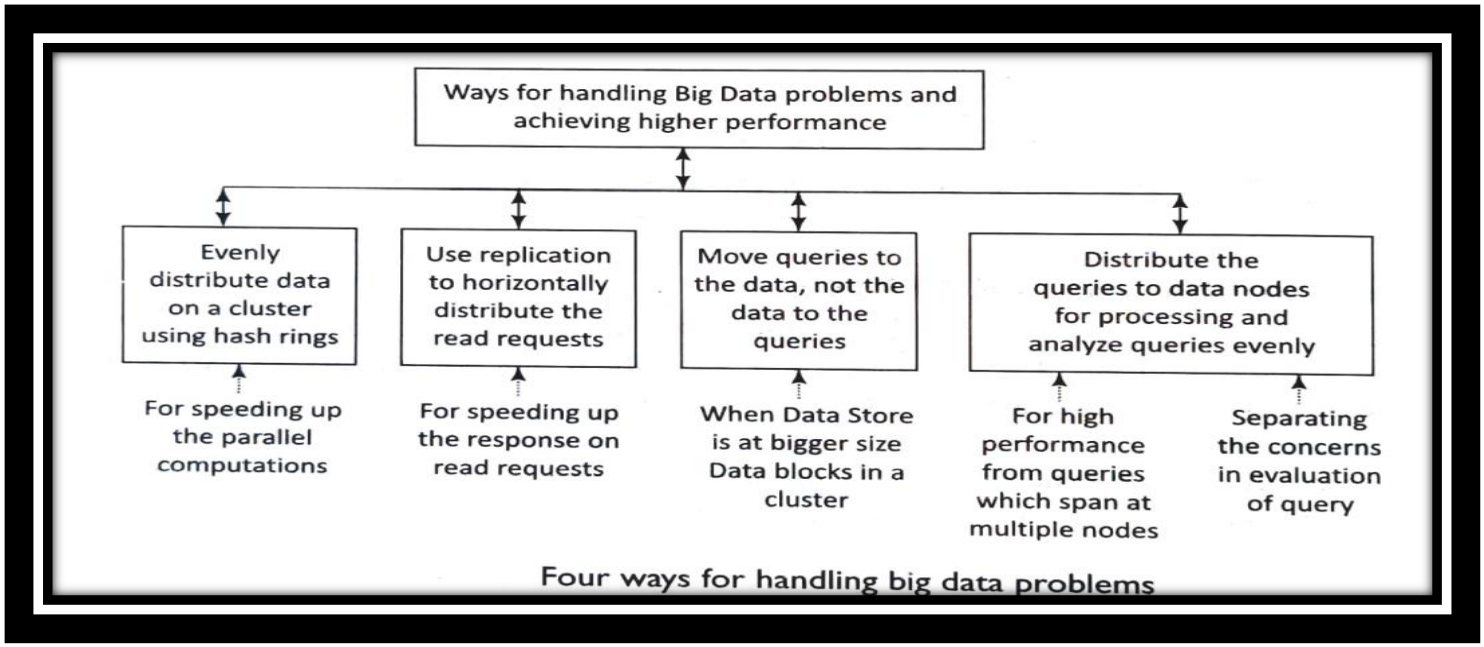


Shards replicating on the nodes, which does read and write operations both

Choosing Master-Slave versus Peer-to-Peer:

Master-slave replication provides greater scalability for read operations. Peer-to-peer replication provides resilience for read and writes both.

Ways of Handling Big Data Problems:



Four ways for handling big data problems

1. Evenly distribute the data on a cluster using the hash rings: Consistent hashing refers to a process where the datasets in a collection distribute using a hashing algorithm which generates the pointer for a collection.
2. Use replication to horizontally distribute the client read-requests: Replication means creating backup copies of data in real time.
3. Moving queries to the data, not the data to the queries: Most NoSQL data stores use cloud ul services (Large graph databases may use enterprise servers). Moving client node queries to the data is efficient as well as a requirement in Big Data solutions.
4. Queries distribution to multiple nodes: Client queries for the DBs analyze at the analyzers, which evenly distribute the queries to data nodes/ replica nodes. High performance query processing requires usages of multiple nodes.

MongoDB DATABASE:

- MongoDB is an open-source DBMS. MongoDB programs create and manage databases. MongoDB manages the collection and document data store.
- MongoDB functions do querying and accessing the required information. The functions include viewing, querying, changing, visualizing and running the transactions.
- MongoDB is (i) non-relational, (ii) NoSQL, (iii) distributed, (iv) open source, (v) document based, (vi) cross-platform, (vii) Scalable, (vii) flexible data model, (ix) Indexed, (x) multi-master (xi) fault tolerant.

  The features of MongoDB:

  1. MongoDB data store is a physical container for collections. Each DB gets its own set of files on the file system. A number of DBs can run on a single MongoDB server. DB is default DB in MongoDB that stores within a data folder.
  2. Collection stores a number of MongoDB documents. It is analogous to a table of RDBMS. A collection exists within a single DB to achieve a single purpose. Collections may store documents that do not have the same fields. Thus, documents of the collection are schema-less.
  3. Document model is well defined. Structure of document is clear, Document is the unit of storing data in a MongoDB database. Documents are analogous to the records of RDBMS table. Insert, update and delete operations can be performed on a collection. Document use JSON (JavaScript Object Notation) approach for storing data.
  4. MongoDB is a document data store in which one collection holds different documents. Data store in the form of JSON-style documents. Number of fields, content and size of the document can differ from one document to another.
  5. Storing of data is flexible, and data store consists of JSON-like documents. This implies that the fields can vary from document to document and data structure can be changed over time.
  6. Storing of documents on disk is in BSON serialization format. BSON is a binary representation of JSON documents. The mongo JavaScript shell and MongoDB language drivers perform translation between BSON and language-specific document representation.

7. Querying. indexing, and real time aggregation allows accessing and analyzing the data efficiently.
8. Deep query-ability supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
9. No complex Joins.
10. Distributed DB makes availability high, and provides horizontal scalability.
11. Indexes on any field in a collection of documents: Users can create indexes on any field in a document.
12. Atomic operations on a single document can be performed even though support of multi-document transactions is not present.
13. Fast-in-place updates: The DB does not have to allocate new memory location and write a full new copy of the object in case of data updates.
14. No configurable cache: MongoDB uses all free memory on the system automatically by way of memory-mapped files (The operating systems use the similar approach with their file system caches).
15. Conversion/mapping of application objects to data store objects not needed.

Comparison of RDBMS and MongoDB databases

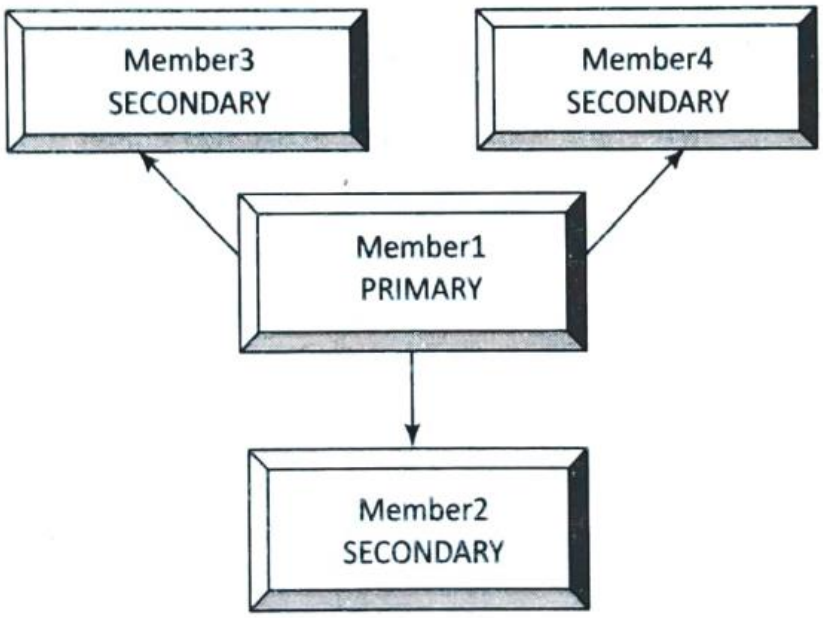
RDBMS	MongoDB
Database	Data store
Table	Collection
Column	Key
Value	Value
Records / Rows / Tuple	Document / Object
Joins	Embedded Documents
Index	Index
Primary key	Primary key (_id) is default key provided by MongoDB itself

Following table gives the commands used for replication (Recoverability means even on occurrences of failures; the transactions ensure consistency):

MongoDB Client commands related to replica set

Commands	Description
rs.initiate ()	To initiate a new replica set
rs.conf ()	To check the replica set configuration
rs.status ()	To check the status of a replica set
rs.add ()	To add members to a replica set

Following figure shows a replicated dataset after creating three secondary members from a primary member:



Replicated set on creating secondary members

Auto-sharding is a method for distributing data across multiple machines in a distributed application environment. MongoDB uses sharding to provide services to Big Data applications. Sharding automatically balances the data and load across various servers. Sharding provides additional write capability by distributing the write load over a number of mongod (MongoDB Server) instances.

Comparison of features MongoDB with respect to RDBMS

Features	RDBMS	MongoDB
Rich Data Model	No	Yes
Dynamic Schema	No	Yes
Typed Data	Yes	Yes
Data Locality	No	Yes
Field Updates	Yes	Yes
Complex Transactions	Yes	No
Auditing	Yes	Yes
Horizontal Scaling	No	Yes

Data types which MongoDB documents support:

Type	Description
Double	Represents a float value.
String	UTF-8 format string.
Object	Represents an embedded document.
Array	Sets or lists of values.
Binary data	String of arbitrary bytes to store images, binaries.
Object id	ObjectIds (MongoDB document identifier, equivalent to a primary key) are: small, likely unique, fast to generate, and ordered. The value consists of 12-bytes, where the first four bytes are for timestamp that reflects the instance when ObjectId creates.
Boolean	Represents logical true or false value.
Date	BSON Date is a 64-bit integer that represents the number of milliseconds since the Unix epoch (Jan 1, 1970).
Null	Represents a null value. A value which is missing or unknown is Null.
Regular Expression	RegExp maps directly to a JavaScript RegExp
32-bit integer	Numbers without decimal points save and return as 32-bit integers.
Timestamp	A special timestamp type for internal MongoDB use and is not associated with the regular date type. Timestamp values are a 64-bit value, where first 32 bits are time, t (seconds since the Unix epoch), and next 32 bits are an incrementing ordinal for operations within a given second.
64-bit integer	Number without a decimal point save and return as 64-bit integer.
Min key	MinKey compare less than all other possible BSON element values, respectively, and exist primarily for internal use.
Max key	MaxKey compares greater than all other possible BSON element values, respectively, and exist primarily for internal use.

MongoDB querying commands

Command	Functionality
Mongo	Starts MongoDB; (*mongo is MongoDB client). The default database in MongoDB is test.
db.help ()	Runs help. This displays the list of all the commands.
db.stats ()	Gets statistics about MongoDB server.
Use <database name)	Creates database
Db	Outputs the names of existing database, if created earlier
Dbs	Gets list of all the databases
db.dropDatabase ()	Drops a database
db.database name.insert ()	Creates a collection using insert ()
db.<database name>. find ()	Views all documents in a collection
db.<database name>.update ()	Updates a document
db.<database name>.remove ()	Deletes a document

CASSANDRA DATABASES:

- Cassandra is basically a column family database that stores and handles massive data of any format including structured, semi-structured and unstructured data.
- Cassandra was developed by Facebook and released by Apache. Cassandra was named after Trojan mythological prophet Cassandra, who had classical allusions to a curse on oracle.
- Apache Cassandra DBMS contains a set of programs. They create and manage databases. Cassandra provides functions (commands) for querying the data and accessing the required information.
- Apache Cassandra has the distributed design of Dynamo. Cassandra is written in Java. Big organizations such as Facebook, IBM, Twitter, Cisco, Rackspace, eBay, Twitter and Netflix have adopted Cassandra.

Characteristics of Cassandra:

(i) open source, (ii) scalable (ii) non-relational (v) NoSQL (iv) Distributed (Vi) column based, (vii) decentralized, (vii) fault tolerant and (ix) tuneable consistency.

Features of Cassandra are as follows:

1. Maximizes the number of writes- writes are not very costly (time consuming)

2. Maximizes data duplication

3. Does not support Joins, group by, OR clause and aggregations

4.Uses Classes consisting of ordered keys and semi-structured data storage systems

5. Is fast and easily scalable with write operations spread across the cluster. The cluster does not have a master-node, so any read and write can be handled by any node in the cluster.

6. Is a distributed DBMS designed for handling a high volume of structured data across multiple cloud Servers.

7. Has peer-to-peer distribution in the system across its nodes, and the data is distributed among all the nodes in a cluster.

Components of cassandra

Component	Description
Node	Place where data stores for processing
Data Center	Collection of many related nodes
Cluster	Collection of many data centers
Commit log	Used for crash recovery; each write operation written to commit log
Mem-table	Memory resident data structure, after data written in commit log, data write in mem-table temporarily
SSTable	When mem-table reaches a certain threshold, data flush into an SSTable disk file
Bloom filter	Fast and memory-efficient, probabilistic-data structure to find whether an element is present in a set, Bloom filters are accessed after every query.

Data types built into Cassandra, their usage and description

CQL Type	Description
ascii	US-ASCII character string
bigint	64-bit signed long integer
blob	Arbitrary bytes (no validation), BLOB expressed in hexadecimal
boolean	True or false
counter	Distributed counter value (64-bit long)
decimal	Variable-precision decimal integer, float

double	64-bit IEEE-754 *double precession* floating point integer, float
float	32-bit IEEE-754 *single precession* floating point integer, float
inet	IP address string in IPv4 or IPv6 format, used by the python-cql driver and CQL native protocols
int	32-bit signed integer
list	A collection of one or more ordered elements
map	A JSON-style array of literals: {literal: literal, literal: literal ...}
set	A collection of one or more elements
text	UTF-8 encoded string
timestamp	Date plus time, encoded as 8 bytes since epoch integers, strings
varchar	UTF-8 encoded string
varint	Arbitrary-precision integer

Cassadra Data Model:

Cassandra Data model is based on Google's BigTable. Each value maps with two strings (row key. column key) and timestamp, similar to HBase .

The database can be considered as a sparse distributed multi-dimensional sorted map. Google file system splits the table into multiple tablets (segments of the table) along a row. Each tablet, called METAI tablet, maximum size is 200 MB, above which a compression algorithm used.

Cassandra Data Model consists of four main components:

(i) Cluster: Made up of multiple nodes and keyspaces,

(ii) Keyspace: a namespace to group multiple column families, especially one per partition.

(iii) Column: consists of a column name, value and timestamp.

(iv) Column-family: multiple columns with row key reference.