NOSQL and Data Lake Architecture

IBM
590 Madison Avenue (at 57th St.)
New York, NY

By Tom Haughey
InfoModel, LLC
Tom.Haughey@InfoModelUSA.com
201 755 3350
Agenda

• Describe NOSQL
• Propose relationship of NOSQL to the data lake
• Analyze the capabilities of NOSQL to support BI
• Propose a new overall reference data architecture
NOSQL

On the outside

On the inside – hundreds to thousands of servers

Server

© InfoModel, LLC. 2017
• Blog entry could also be absorbed into Blog as an array.
• All arrows are links.
• In many cases, the actual user name would be included rather than just the ID.
• This is an example of a NOSQL data model, not necessarily of notation
• [ ] indicates embedding.
Understanding NOSQL

• The following essential to proper understanding of NOSQL:
  – **Data distribution**: Data is spread horizontally over many servers
  – **Semi-structured**: Not “schema-less”; schema is embedded in the data or the application
  – **Composite**: A structure can consist of other structures
  – **Hierarchical**: 1:M ranking from parent to child (except graph DBMSs)
  – **Key-value structure**: Most NOSQL’s consist of a key and a value
    • Value can be a blob, string, or container of *other key : value pairs.*
    • *The exception to this is graph DBMS*
  – **Materialized queries**: Basically a structure for each query
  – **Application orientation**: Not enterprise-oriented, but query oriented.
  – **Relationships**: Unidirectional links, not joins.
  – **Data model**: A NOSQL data model is a physical, not logical, data model

• **Four main types**: key : value, wide column, document, graph
1 - Key Value

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
</table>

Whatever you want it to contain, such as a CSV, a blob, or other key:values, e.g., streaming stock market data:

```python
ticker : ..., 
rate_date : ..., 
price : ... , 
price_change : ... , 
percent_change : ...
```

**Use Cases**

- File storage
- Log records
- Session storage
- Content management
- Streaming data
  - (songs, albums, etc.)
- Product data management
- High volume data feeds
- User data management
- Hadoop

@ InfoModel, LLC. 2017
### 2 - Wide Column (Column Family)

- Sometimes called column-oriented but actually row-oriented
- Several variations but this is the Hbase version

#### The table is sorted based on row key

<table>
<thead>
<tr>
<th>Row Key</th>
<th>Name</th>
<th>Home Phone</th>
<th>Office Phone</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>Luke</td>
<td>212 555 5689</td>
<td>201 891 2536</td>
<td>21 McCoy Ave</td>
</tr>
<tr>
<td>00021</td>
<td>Bryce</td>
<td>212 232 6785</td>
<td>201 766 2091</td>
<td>26 McCoy Ave</td>
</tr>
<tr>
<td>00033</td>
<td>Matthew</td>
<td>201 234 5768</td>
<td>201 766 7381</td>
<td>26 McCoy Ave</td>
</tr>
<tr>
<td>00046</td>
<td>James</td>
<td>908 435 6242</td>
<td>908 657 5438</td>
<td>5 Hatfield St</td>
</tr>
<tr>
<td>00057</td>
<td>Jack</td>
<td>347 361 5429</td>
<td>973 376 8394</td>
<td>6 Wallace St</td>
</tr>
</tbody>
</table>

Each cell can have multiple versions individually timestamped.
2 – Wide Column (Cassandra)

Characteristics:
- Sparse table structure
- For each key, there can be:
  - Variable attributes
  - New columns without indicating they are new
  - Omission of columns

Use Cases
- Product Catalog/Playlist
- Recommendation/Personalization Engine
- Sensor Data/Internet of Things
- Messaging
- Fraud Detection
- Used by eBay, Twissandra ...

Table’s sorted based on row key

Columns sorted based on column key

<table>
<thead>
<tr>
<th>Key 1</th>
<th>Key 2</th>
<th>Key 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 6</td>
<td>B = 42</td>
<td>A = 1</td>
</tr>
<tr>
<td>C = 3</td>
<td>D = 62</td>
<td>B = 5</td>
</tr>
<tr>
<td>D = 53</td>
<td>E = 84</td>
<td>F = 20</td>
</tr>
<tr>
<td>E = 10</td>
<td></td>
<td>H = 9</td>
</tr>
</tbody>
</table>
3 - Document

• A key : value store that understands the data

Posts =

{ _id: “A12345”
  author: “Mickey”,
  date: 22/6/2012,
  text: “The Adventures of Mickey Mouse”,
  keywords: [“drama”, “comic”, “adventure”] } 

comments : [ { author: “Nick Machiavelli”,
                date: 11/12/2012,
                rating: “Less filling, tastes great”,
                votes: 7000 } , … ]

}
Use Cases for Document Data Stores

- Content management
- High volume data feeds
  - Stock market data
  - Streaming music
- Operational intelligence
  - At detailed level
  - At aggregate level
- Product data management
- User data management (users, profiles, etc.)
- Hadoop
4 - Graph Databases

• Every element contains a direct pointer to its adjacent elements
Graph Databases

- Based on graph theory
- Addresses data complexity
- Direct path operations are easy
- Can be transactional
- FOAF (Friend Of A Friend)
- Usually paired with an index for search
Joins

• NOSQL structures are materialized queries with all or most of the query data embedded into the one structure
• Joins are not just “unnecessary” in NOSQL, they are not possible (through the DBMSs)
  – Remember data is highly distributed across many, many servers
  – Joins would be far-reaching and inefficient
  – NOSQL embedding implies redundancy
• Joining across servers would require (something like):
  – Use of a partitioning key that can hash (not necessarily the PK)
  – A hashing algorithm that consistently resolves a partitioning key to the same node
  – An ~even distribution of data
  – Ideally, collocation of data to be joined
Metadata in the Data Lake

- Some metadata, such as data type, length, domain, granularity, business/technical definition and others, must eventually be assigned to data lake for:
  - Data
  - Relationships and more

- Say Monthly Sales Revenue is ingested into the data lake from different orgs/countries (in which case these totals are raw data)
  - Are the grains the same? Do they agree?
  - They are both Sales Revenue but suppose one is “sales as of last day of the month” and the other is “sales as of the last Friday of the month”
  - Metadata is required to understand this data
Data Lake on NOSQL?

• A data lake can reside on Hadoop, NoSQL, Amazon Simple Storage Service, a relational database, or different combinations of them

• Fed by data streams

• Data lake has many types of data elements, data structures and metadata in HDFS without regard to importance, IDs, or summaries and aggregates

• Important to understand the variegated nature of the data lake data in relation to the metamodel of the perspective NOSQL database
  – Semi-structured
  – Key : value (mostly) with its hierarchical structure
  – The key and column name being essential parts of most NOSQL

• More often data lake is kept on Hadoop and fed to or from NOSQL
  – Some additionally used a graph database on top to keep track of the relationships
  – Once loaded onto NOSQL, the full power of NOSQL can be used

• Note: NOSQL is an operational, not an analytical, data store
BI on NOSQL

• NOSQL does not yet have commodity BI tools
• In all, there are several approaches* to BI on NOSQL, which divide into two major groups:
  – Those that use NoSQL only and strengthen the application with better UIs, better *adhoc* reporting and other custom features onto the NoSQL products they are currently using
  – Those that use NoSQL to run their applications, but then take that data out of the NoSQL system and put it into a RDBMS or traditional data warehouse for more “after the fact” analysis.
• Each approach has many success stories and the best approach for a particular company is based on their specific needs, budgets and skills

* Based on presentation by Nicholas Goodman, and articles by Charles Roe “BI/Analytics on NOSQL”
1 - Application on Top of NOSQL

• Reports on NOSQL
  – Have a developer build an application for reporting on top of NOSQL
  – Has the full richness of NoSQL but is expensive due to the need for a developer
2 - Enhanced NOSQL-Only

- Adds a dynamic query builder into the reporting app
- Needs a developer to build it is even more expensive
3 - NOSQL Extract to Relational

• Similar to traditional ETL but Hadoop or NOSQL are the source
  – Extract from NOSQL/Hadoop and insert into RDBMS
  – Allows the use of rich BI tools

• Adds to first approach, creating a dynamic query builder into the reporting system
  – Guided *adhoc*
  – Data freshness issue due to day-old data
4 - NOSQL as ETL Source

• NOSQLs are just part of the DW sourcing (ETL)
• Data extracted from NOSQL/Hadoop and inserted into DW and integrated with other DW data

• Pros and Cons
  – Can use standard BI tools, which are costly
  – Rich flexibility/scalability on NOSQL gone
  – ETL development cost
  – Note: Hadoop is a batch environment, not real-time
5 - NoSQL Added to BI Tools

- Developer intensive
  - Adds a service to a standard commodity BI tool
  - Flattens the NoSQL data and outputs it into a report
  - No need for SQL or *adhoc* web-based access tools
  - No need for loads of expensive reports written in NOSQL

- The program is written one time
  - It is an application with substantial developer costs
  - Can use M:R for up-to-date access to 100% of the dataset
  - Aggregations might be slower
6 – Interfacing NOSQL and BI

- Third party Enterprise Information Integration (EII) between the commodity BI and the NoSQL or data lake
  - The EII tool can speak to both NOSQL and BI
  - Integration with other data, gives live up-to-date access
  - ETL is simple with INSERT/MERGEs done nightly and has *adhoc* access to live, cached data.
  - “Best of both”: NoSQL on the back, BI on the front
  - The cost and complications of introducing a third system
  - There is still some loss of the richness of NoSQL language
Transforming Data

• To use data, it must be put into a useable state
• You cannot put raw data into any repository (including the data lake) and pretend that *ipso facto* it eliminates the need for transformation (as many claim!)
• It’s a case of “Pay me now or pay me later”
• Data is either transformed by ETL/ELT *before* it is stored or it is transformed *after* by the query
  – Schema on write - a batch process of ETL or ELT in a relational environment
  – Schema on read by the query manager – whether in RDBMS, NOSQL, or Map:Reduce
• One main reason for storing aggregates in the DW is to provide consistent numbers
  – The data lake too should ensure consistent numbers
Sample Strategic Query

• Strategic queries using many dimensions and aggregations could pose a problem to NOSQL
  – “Give me a breakdown of total commissions paid and transaction count by account, summarized by product type and product class, and ordered by the organization that owns the product and the organization that sold the product.”

• Such a query would be difficult to do in NOSQL without materializing the query and using Map/Reduce functions

• A DW on a robust platform could efficiently support this without having to materialize the aggregate and still support queries with other mixes of dimensions
Mining Data Lake Data

• Mining is the use of mathematical algorithms to find hidden relationships in the data
• Just as the popularity of new tools is exploding, so are the capabilities in data mining
• NOSQL is well suited for data mining
• Data-mining techniques fall into four major categories:
  – Classification – such as targeted marketing
  – Association – such as market basket analysis
  – Sequencing – those who bought this bought that
  – Clustering – developing conclusions using space and distance
• NOTE: In Hadoop, querying and mining can be done through Hive, Mahout and Pig
Data Lake Reference Architecture

Raw Data
Data Lakes

Data Streams

Big Data
RDBMSs Appliance NOSQL HADOOP

Data Integration

Trusted Data
EDW Mart

Real-time Analytics
RYO data

Consumers

Streams Sensors Events Docs XML/JSON Files Cloud Tables OLAP Web Logs
A DW is a generalized environment in which data is gathered from many sources, transformed, stored and then delivered with business meaning
- Feeds any type of query, including ad hoc queries
- Application neutral

Data marts are a specialized collection of related data for a particular user community and for a limited time
- Are user- and application-specific
- Will answer very well those questions within data marts scope

Four types of NOSQL
- Key : value, wide column, document, graph
- Most NOSQL databases are hierarchical and operational
- This means materializing almost every query requiring any data stores

Data Lake
- A storage repository that holds a vast amount of raw data in its native format until it is needed
Questions?

The term NOSQL was first used by Carlo Strozzi in 1998 to describe a file-based database he was building