Data Lakes, Data Hubs and AI

Dan McCreary
Distinguished Engineer in Artificial Intelligence
Optum Advanced Applied Technologies
Background for Dan McCreary

- Co-founder of "NoSQL Now!" conference
- Coauthor (with Ann Kelly) of "Making Sense of NoSQL"
  - Guide for managers and architects
  - Focus on NoSQL architectural **tradeoff** analysis
  - Basis for **40 hour course** on database architectures
  - How to pick the right database **architecture**
  - [http://manning.com/mccreary](http://manning.com/mccreary)
- Focus on metadata and IT strategy (capabilities)
- Currently focused on NLP and Artificial Intelligence Training Data Management
The Impact of Deep Learning

Predictive Precision

Deep Learning

Traditional Machine Learning (Linear Regression)

Training Set Size

Large datasets create competitive advantage
High Costs of Data Sourcing for Deep Learning

80% of time wasted

By data scientists just getting access to data and preparing data for analysis

60% of the cost

Of data warehouse projects is on ETL

$3.5 billion in spending

In 2016 on data integration software
Six Database Core Architecture Patterns

Relational

Analytical (read-mostly OLAP)

Key-Value

Column-Family

Graph

Document

Which architectures are best for data lakes, data hubs and AI?
Role of the Solution Architect

Sally Solutions
Title: Solution Architect

• Non-bias matching of business problem to the right data architecture before we begin looking at a specific products
Google Trends

Data Lake

Data Hub
Data Science and Deep Learning

Data Science

Deep Learning
Data Lake Definition

A storage repository that holds a vast amount of raw data in its native format until it is needed.

Examples of raw native format:
• Dump of data from an RDBMS in csv format
• Export data with many numeric codes
• Log files

~10 TB and up $350/10TB Drive

http://searchaws.techtarget.com/definition/data-lake
Data Lake Assumptions

• Scale-out architecture
  • Adding more data won't slow down reads
  • No "joins" are ever used to access data
• Consistency
  • Consistent read and write performance, even under heavy load
• High availability and tunable replication
  • Default replication factor = 3
• No secondary indexes
• Low cost
  • $500/TB/year (Amazon S3 is at under $360/TB/year)
# Amazon S3 Pricing (Nov. 2016)

<table>
<thead>
<tr>
<th>Region: US East (Ohio)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Storage</strong></td>
</tr>
<tr>
<td>First 50 TB / month</td>
</tr>
<tr>
<td>Next 450 TB / month</td>
</tr>
<tr>
<td>Over 500 TB / month</td>
</tr>
<tr>
<td><strong>Standard - Infrequent Access Storage</strong></td>
</tr>
<tr>
<td>First 50 TB / month</td>
</tr>
<tr>
<td>Next 450 TB / month</td>
</tr>
<tr>
<td>Over 500 TB / month</td>
</tr>
<tr>
<td><strong>Glacier Storage</strong></td>
</tr>
<tr>
<td>First 50 TB / month</td>
</tr>
<tr>
<td>Next 450 TB / month</td>
</tr>
<tr>
<td>Over 500 TB / month</td>
</tr>
</tbody>
</table>

- 23 cents/GB/year = $230/TB/year

Service Level Agreement (SLA) Design:
- 5 “9s” availability (you can read your data at any point in time)
- 11 “9s” durability (your data will not get lost)
- Price has **never** gone up (only down)
Implemented with a distributed fault-tolerant parallel file systems

• Hadoop
  • Hadoop Distributed File System – HDFS
  • Default replication level 3
  • 128MB block size designed for write once, read may
• Amazon S3
  • Cloud based object store – cost leader for High Availability
• Other Distributed File Systems:
  • Ceph (OpenStack)
  • GlusterFS (now owned by Red Hat)
  • GNU Cluster File System
  • Lustre

Data Hub Definition

A collection of data from multiple sources organized for distribution, sharing, and subsetting.

You can query it!

Generally this data distribution is in the form of a hub and spoke architecture.

A data hub differs from a data lake by homogenizing data and possibly serving data in multiple desired formats, rather than simply storing it in one place, and by adding other value to the data such as de-duplication, quality, security, and a standardized set of query services.

Data Flow Comparison

**Data Lake**

- Applications
  - CSV Dumps
    - Distributed File System
      - Log Files
        - Batch Analysis

*Not* real-time

**Data Hub**

- Applications
  - Data Service
  - Data Service
  - Data Service

Sub-millisecond response times

Which one generates web pages?
Sample Document Data Hub Data Flow Diagram

Staging
- Bulk Loader
  - http put

Canonical Harmonized
- Transform
  - Raw Data
  - One doc per row
  - One doc per object
- Semantically Precise Document Data
  - validate

Egress APIs (REST web services)
- Extract
  - http get
  - XML
  - JSON
  - RDF
  - ODBC
- search
dashboard

XML Schemas (data quality)

Metadata and Reference Data (semantics)
A.I. Driven Strategies

Staging

- Raw Data
  - One doc per row

Canonical
Harmonized

- Semantically Precise Data
  - One doc per object

- validate

Machine Learning Services
(REST web services)

- Feature Scaling
- Mean Normalization

- Training Set
- Model Parameters
- Inference

Predictive Applications

XML Schemas (data quality)

Metadata and Reference Data (semantics)
GAFA 2016 R&D Spending Amounts

1. Google - $16B
2. Amazon - $16B
3. Facebook - $6B
4. Apple - $10B

Total - $48 billion in annual R&D spending

https://www.recode.net/2017/9/1/16236506/tech-amazon-apple-gdp-spending-productivity

Or “GAFAM” and “GAFAMI” if you include Microsoft and IBM
How do data lakes answer the question...

What is the answer to life, the universe and everything?

<answer>42</answer>
Seven Levels of Semantics

Typical CSV data

1
<INDGENCD>42</INDGENCD>

2
<Gender>42</Gender>

3
<PersonGenderCode>42</PersonGenderCode>

4
<PersonGenderCode>F</PersonGenderCode>

5
<PersonGenderCode>Female</PersonGenderCode>

6
<c:PersonGenderCode>Female</c:PersonGenderCode>

7

What does this mean?

Clear Meaning Search Friendly!

Low Semantics

Low Data Governance

Data Stewards

High Semantics

RDF (next page)
Resource Description Format

- RDF breaks each assertion into Node-Arc-NodeSubject-Predicate-Object Relationships
- Subject and Predicates are URLs. Objects are sometimes "literal" strings
- If all imported data used RDF than transformation and integration would be trivial (no ETL)
The Semantic Spectrum

Low Semantics

1. Mostly Numeric Codes
2. No harmonization
3. Write and read by the source application
4. No namespace and validation
5. No data quality

High Semantics

1. Numbers and labels
2. Incremental harmonization
3. Writes and read by everyone
4. Validation
5. Data quality for all egress documents

Note that high-semantics are not "free"

It requires a large framework of tools to convert numeric codes into useful labels
It's About Building Integrated Views (360 views)

**Integrated** views of customers - every touchpoint visible by call center

**Integrated** views of hardware devices - every desktop, server, firewall etc.

**Integrated** views of whatever....
100% Failure Rate

The Old Way: Comprehensive Enterprise Modeling

First a brief word on the old approach. People used to (and occasionally still) build a new enterprise data model comprising every field and value in their existing enterprise, across all silos, and then map every silo to this new model in a new data warehouse via ETL jobs.

ITBusinessEdge surveyed companies and found that this approach always fails. Survey respondents report that it goes over budget or fails 100% of the time.

The Challenge of Data Variability in RDBMS Systems

“Our ER modeling process is taking too long.”
“Every time we think we have our ER model finalized there is another change request.”
“It takes us a long time to update or models after we have 100M rows of test data loaded.”
“These hidden one-to-many relations have slowed down our teams progress to a crawl.”
“We have no way to predict future data variability.”
“Exceptions make the rules. Each new system load has 10% variability.”
“Our system will be obsolete the day after we go to production.”
“Relational databases are like concrete – ones they set they are difficult to change.”

Perfection is the Enemy of Progress
What Data Lakes and Data Hubs both have in common

1. They both are **NOT** relational (Yeah)!
2. No data modeling before you load your data! -> Agility, Flexibility (Schema Agnostic)
3. They both leverage low-cost shared-nothing commodity hardware
4. They both know how to reliably distribute computing loads over hundreds of processors
5. The both help organization understand the challenges of distributed computing
Low-Cost Scalability: Shared Nothing Architecture

Every node in the cluster has its own CPU, RAM and disk - but what about GPUs?
Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

L Peter Deutsch
Data Hub Philosophy

• Ingest everything
• Index everything
• Analyze everything from the indexes
• Track data quality
• Incrementally harmonize
• Promote strong data governance and data stewardship
• Make it easy to do transactions, search and analytics on harmonized data
Document-Centric Data Hubs

- Document databases are **ideal** for many egress tasks
- Graphs make it easy to link related documents together
What is an Enterprise Canonical Model?

How can we minimize dependencies when integrating applications that use different data formats?

Design a *Canonical Data Model* that is independent from any specific application. Require each application to produce and consume messages in this common format.

http://www.enterpriseintegrationpatterns.com/CanonicalDataModel.html
Many-Component Specialized DB Cost Model

Many highly specialized databases
Each with their own strengths and weakness
Expensive ETL (batch and real-time) to keep databases in sync
Chargebacks based on CPUs and disk storage (not I/O)
Little understanding of the costs of moving data between systems

Total cost = RDBMS + ETL + DW + ETL + Search
Where do I store my metadata?

How can we eliminate the ETL?
Traditional EDW (and ODS) Pain Points

Non-trivial delivery time and effort

Dependency on Extract, Transform and Load (ETL) and movement of a lot of data

Very brittle with respect to change

Not suited for unstructured data

Legacy RDBMS technology does not scale flexibly or economically
Challenges with ETL

- Designed for table-to-table transformations
- Written in SQL with memory intensive "joins"
- Difficult to scale
- Difficult to re-use centralized business rules (no document validation)
- Batch transforms typically run over night
- Limited time windows
- Little chance for recovery on errors
Imagine a single database for:
- All transactions
- All Search
- All Analytics

Use a scale out architecture with ACID transactions
Index **everything** for fast queries, search and deep analytics

Avoid moving data around
Total costs can be much lower!
Why Use a Document Store?

Document stores are ideal when you have highly-variable data

- Example: clinical healthcare data

Document stores can be designed to have a "scale-out" horizontal-scalable architecture

- Unlike relational models, there are limited "join" operations
- Simply add new nodes to a "cluster" and the system can "rebalance" and support higher volumes of data
How are **Document-Centric** Data Hubs Different?

Handles complex data
- Does not fit well into a single table

Handles highly variable:
- Example: healthcare.gov
  - 37 states – 37 variations

Diverse users:
- Examples: Clinical, Claims, Analytics, Search, Research, Pharma

High security and audit (PHI)
- Requires role-based access control

Volume
- Requires a true scale-out architecture
Three Functions – One Cluster – One Set of APIs

Combines a transaction safe application server, a database server and a search engine in the same server. Minimize data movement and ETL costs.
Non-scalable systems reach a plateau of performance where adding new processors does not add incremental performance (Relational and Graph).

Linear scalable architectures provide a constant rate of additional performance as the number of processors increases.
Data Storage Cost Models

• Measured in dollars per TB per year ($/TB/Y)
  TB = Terabyte

• 1 TB hard drives cost = $40 (qty 1)
Comparison of Normal vs. Denormal Forms

- **Normalized**
  - Person
    - Contacts
    - Address
    - GenderCodes
    - EthnicityCodes
    - RaceCodes
    - CitizenshipCodes
    - Purchases
    - Ratings
  - One join per child table
  - 17 tables = 16 joins

- **Denormalized Document**
  - One document
  - Single line of code: `get-person('123')`
Sample Data Hub Data Flow Diagram

Staging

- http put
- mlcp

Bulk load

Canonical

- XQuery
- One doc per row
- XML Schemas (data quality)

Egress APIs (REST web services)

- http get
- JSON
- RDF
- ODBC

- search
- dashboard

validate

One doc per object

XML

Metadata and Reference Data (semantics)
Definitions

Staging
- Raw data
- One document per row from RDBMS
- Simple flat data

Canonical
- De-normalized and enriched data
- Semantically clear
- Validated by XML Schemas

Egress
- REST web services of high-quality data
- May include the use of pre-calculated aggregates (materialized views) to speed analytical reports
- Multiple formats
- Many options
- Strong SLAs (read times, high availability)
Reference Data

Data that is not associated with each new transaction

Data elements that use the "Code" suffix

- ISO-11179 Representation Term

Based on standards

- 80% of the tables in CDB is reference data

Goal:

- Consistent usage over time and between projects

Examples:

- Gender Code
- Ethnicity Code
- US State Code
# Sample Reference Codes

## Reference Data Codes

<table>
<thead>
<tr>
<th>Code Name</th>
<th>File Name</th>
<th>Record Count</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>citizenship-status-type</td>
<td>citizenship-status.xml</td>
<td>7</td>
<td>Thu, Apr 30 '15 15:21:31</td>
</tr>
<tr>
<td>gender-code</td>
<td>gender.xml</td>
<td>3</td>
<td>Thu, Apr 30 '15 13:35:57</td>
</tr>
<tr>
<td>marital-status</td>
<td>marital-status.xml</td>
<td>9</td>
<td>Thu, Apr 30 '15 11:45:06</td>
</tr>
<tr>
<td>race-type</td>
<td>race-type.xml</td>
<td>16</td>
<td>Thu, Apr 30 '15 13:52:57</td>
</tr>
<tr>
<td>state-abbr</td>
<td>state-abbr.xml</td>
<td>50</td>
<td>Mon, May 04 '15 20:55:59</td>
</tr>
</tbody>
</table>

## Reference Code Mapping

<table>
<thead>
<tr>
<th>Mapping Name</th>
<th>File Name</th>
<th>Record Count</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>marital-status-mapping</td>
<td>marital-status-mapping.xml</td>
<td>7</td>
<td>Thu, May 07 '15 10:35:57</td>
</tr>
<tr>
<td>race-type-mapping</td>
<td>race-type-mapping.xml</td>
<td>20</td>
<td>Thu, May 07 '15 10:36:12</td>
</tr>
</tbody>
</table>

Execution Time: 0.153703 seconds.
How to build semantically useful systems?

• Muddy Data Lake

• Clear Data Hub

Build Continuous Enrichment Pipelines

Raw Data → Index → Enrich → Validate → Score
Content Enrichment

Continuous process of enriching content using resources from your data hub
Reference Data Enrichment Services

- Conversion of low-semantics elements to high-semantics and search friendly forms
- Example:
  - Input: street, city, state
  - Output: add longitude and latitude
Data "Opaqueness"

• **Muddy**
  • No precise definitions and validation for data elements and code values
  • Expensive to integrate into other systems
  • Difficult to generate consistent reports over time and across projects
  • No data stewardship and change control
  • No "enrichment" processes

• **Clear**
  • Precise definitions for each data element and code values
  • Multiple sources continuously harmonized into canonical forms
  • Low cost to integrate and share with other systems
  • Designed for multiple purposes
  • Strong data stewardship and robust change control
  • Continual data enrichment

*How clear are the semantics of our integration hub?*
Data Analyzer / Data Profiler
Avoid Data Archaeology

• Time consuming task of converting numeric representations to symbolic representations
• Focus on strong metadata management and metadata services

What does '42' mean?

XML Schemas (data quality)

Metadata and Reference Data (Semantics)
"Semantics" vs. "semantics"

• **Semantics** with an uppercase "S" usually refers to the Semantic Web technology stack (RDF, SPARQL, inference)

• **semantics** (with a lower case) usually refers to the process of creating shared meaning across multiple business units. This refers to the processes of Data Stewardship, Data Governance and metadata registries

• Our recommendation is to use 90% documents and RDF (Semantics) to link documents

• We do not recommend storing canonical documents in pure RDF
The Role of Search

- Many systems don't integrate search well into their data
- Calculating keyword density is hard
- Calculating concept density is even harder
- Great data hubs must come up with ways to make it easy to find the data you need
The root cause of many integration problems is that the application is tightly coupled to the database. Raw data dumped about of a typical RDBMS is almost useless without using the logic within and application.
Metcalf's Law

The value of a [system] is proportional to the square of the number of connected [systems].

https://en.wikipedia.org/wiki/Metcalfe%27s_law
Lower Incremental Costs

Each new outbound data service can leverage prior data loaded in the data hub.
Architecture Tradeoff and Analysis Method (ATAM)

Business Drivers → Quality Attributes → User Stories
Architectural Plan →

Architecture Plan → Architectural Decisions

Risk Themes → Distilled info → Tradeoffs, Sensitivity Points, Non-Risks, Risks

See Chapter 12 of *Making Sense of NoSQL*
Quality Attribute Tree

- How important
- How difficult in any given architecture
Summary

- Deep Learning needs **lots** of data – typically millions of records
- Both Data Lakes and Data Hubs are great examples of **distributed** computing
- Both have **lower cost/TB/year** than RDBMS and are far more **flexible** than an RDBMS
- Be cautious about doing integration on a RDBMS unless you know you have homogeneous today and forever
- Use Data Lakes for ways to store log files or some forms of **simple** tabular data
- Use document and graph stores for building integration Data Hubs
- Use Data Hubs to power your Deep Learning models
Further Reading and Questions

Thank You!

Dan McCreary
dan.mccreary@optum.com
twitter @dmccreary
LinkedIn http://www.linkedin.com/in/danmccreary

http://manning.com/mccreary