IMS to Big Data

Speaker:
Rick Weaver
Senior Customer Engineer
DataKinetics
Populating Big Data Repositories from IMS

• Address Practical Approach to Real-Time IMS Data Feeds
• Discuss Business Drivers / Considerations
• Outline Concepts
  • Popular Big Data Platforms → Strengths and Weaknesses
  • Bulk Loads (ETL) vs Changed Data Capture (CDC)
  • Data Types / Formats
• Walk through Various Streaming Scenarios
• Q & A
Populating Big Data Repositories from IMS

- Big Data Overview
- Outline Concepts
- Streaming Scenarios
- Q & A
Big Data Hype vs Reality

• What You May Have Heard...
  • The 'New Wave' of Technology
  • Exclusively Hadoop and/or NoSQL Based
  • Big Data 'Knows' What You are Doing...
Big Data Hype vs Reality

- **Reality** → A Large Collection of Data... in Existence for 50+ Years
- **Characteristics**
  - Significant Amount of Data
  - Advanced Analytics of Disparate Data
  - Many Different Formats → Structured, Semi-Structured, Un-Structured
  - High Rate of Change
- **Challenges**
  - Increasing Data Volumes → Stress Traditional RDBMS
  - Computing and Infrastructure Costs to Process / Analyze
  - Most Companies in Early Stages of Adoption
- **Exciting Times Ahead**
  - Large Open Source Communities
  - Rapid Evolution of Technology
You Have a Few Choices → More on the Way
Why Real-Time Streaming of IMS to Big Data?

Analytics...Analytics...Analytics

• Decisions based on Current Information vs 24+ Hour Old Data
• Quickly Detect Key Events / Trends
• Maintain a Competitive Advantage
• Provide Better Customer Service
• Increase Revenue / Profitability
<table>
<thead>
<tr>
<th>Industry</th>
<th>Use Case</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Services</td>
<td>New Account Risk Screens</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Trading Risk</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Insurance Underwriting</td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Telecom</td>
<td>Call Detail Records (CDR)</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Infrastructure Investment</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Real-time Bandwidth Allocation</td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Retail</td>
<td>360 View of the Customer</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Localized, Personalize Promotions</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Supply Chain and Logistics</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Assembly Line Quality Assurance</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Crowd-sourced Quality Assurance</td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Healthcare</td>
<td>Use Genomic Data in Medial Trials</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Monitor Patients Vitals in Real-Time</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Pharmaceuticals</td>
<td>Recruit and Retain Patients for Drug Trials</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Improve Prescription Adherence</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Oil &amp; Gas</td>
<td>Unify Exploration &amp; Production Data</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Monitor Rig Safety in Real-Time</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
<tr>
<td>Government</td>
<td>ETL Offloaded Response to Federal Budgetary Pressures</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Sentiment Analysis for Government Programs</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Geographic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clickstream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unstructured</td>
</tr>
</tbody>
</table>
Best Practices Summary

• Let the Business Drive the Effort
• Temper the Exuberance
• Keep the Fiefdoms at Arm's Length
• Keep an Open Mind with Regard to Technology
• Use an Iterative Approach for Implementation
• Limit costs on the mainframe by **streaming** rather than network bandwidth hungry **bulk unloads** every night or worse…
Key Considerations

- **Big Data Repository Selection**
  - Open Source Projects → the Larger the Community, the Better
  - Beware of Vendor Lock
  - Will Require Multiple Components

- **Data Delivery / Latency**
  - Business Driven
  - Full Extracts → Periodic
  - Near-Real-Time / Scheduled Updates

- **Workload Characteristics**
  - Read vs Update Ratio
  - Update Volume → Transaction Arrival Rate
  - Will Effect Big Data Repository Selection

- **Format**
  - Level of Normalization → Less is Usually Desirable
  - Common Across Multiple Applications / Languages
  - Level of Transformation Required
Today's Popular Big Data Components

- **Hadoop HDFS**
  - Most Commonly Used Big Data Store
  - Foundation Layer for other Technologies such as Spark
  - Highly Scalable

- **Spark**
  - High-Performance Processing Engine
  - Extremely Fast and Versatile → 100x Faster than MapReduce
  - Runs on HDFS or Standalone

- **Kafka**
  - Ultra-Fast Message Broker
  - Streams Data into Most Common Big Data Repositories
  - Multiple Producers / Consumers

- **Other Popular Stores**
  - IDAA / PureData Analytics (Netezza)
  - Cassandra
  - MongoDB
  - More Appearing each day…
Populating Big Data Repositories from IMS

• Big Data Overview
• Outline Concepts
• Streaming Scenarios
• Q & A
ACID vs BASE

ACID → Properties Guarantee DB Transactions are Processed Reliably
- Atomicity → All or Nothing...either the Transaction Commits or it Doesn't
- Consistency → Transaction brings DB from One Valid State to Another
- Isolation → Concurrency
- Durability → Once a Transaction Commits, it Remains Committed

BASE → Eventual Consistency
- Basically Available → Data is There...No Guarantees on Consistency
- Soft State → Data Changing Over Time...May Not Reflect Commit Scope
- Eventual Consistency → Data will Eventually become Consistent

More Info: Charles Rowe – Shifting pH of Database Transaction Processing
The Role of ETL and CDC

**ETL (Extract, Transform, Load):**
- Full Data Extract / Load
- Data Transformation Logic Defined in this Step → Reused by CDC
- Should be Run Against Live Data
- Should Minimize Data Landing

**CDC (Changed Data Capture):**
- Move Only Data that has Changed
- Re-Use Data Transformation Logic from ETL
- Near-Real-Time / Deferred Latency
- Allows for Time Series Deliver
ETL and Changed Data Capture (CDC)

**ETL**
- High Level of Control Over Level of De-Normalization
- Can Combine Many Segments in Target Row / Document
- Requires that ETL Tool can Handle Consolidation during Extract

**Changed Data Capture**
- May Dictate that Target not Fully De-normalized
- Capture Along One (1) Branch of IMS DB Record
- Path / Lookups may be Required
**Target Apply Concepts**

- **Frequency**
  - Near-Real-Time
  - Batches

- **Time Series**
  - Analyze Data Changes Over Time
  - All CDC Data is Inserted into Target
  - timeuuid type Key

- **Incremental Updates (Synchronized)**
  - Source Matches Target
  - Requires Query Adjustments for Insert-Only Targets (i.e. Hadoop HDFS)
Data Format(s)

- Common Formats → JSON, Avro, Delimited, XML, Relational
- JSON Recommended for CDC/ETL Data
  - Especially for Data Lakes
  - Records are Self-Described → Encapsulated Metadata
  - Payload Lighter than XML

```json
{"DEPT": {
  "database": "IMSDB01",
  "change_op": "U",
  "change_time": "2015-10-15 16:45:32.72543",
  "after_image": {
    "deptno": "A00",
    "deptname": "SPIFFY COMPUTER SERVICE DIV.",
    "mgrno": "000010",
    "admrdept": "A00",
    "location": "Chicago"
  },
  "before_image": {
    "deptno": "A00",
    "deptname": "SPIFFY COMPUTER SERVICE DIV.",
    "mgrno": "000010",
    "admrdept": "A00",
    "location": "Dallas"
  }
}}
```

Sample Update CDC Record in JSON Format
Data Types

In Addition to the Traditional Data Types (char, integer, decimal, etc.)

- **boolean** → True/False
- **counter** → Similar to Identity Columns
- **inet** → IP Address
- **timeuuid** → Unique Value based on Timestamp and Random
- **uuid** → Unique Value based on Random and Timestamp

**Complex Data Types**

- Lists
- Sets
- Maps
- Tuples
- Structures
- Arrays
Common IMS Data Challenges

- Code Page Translation
- Invalid Data
- Dates
- Repeating Groups
- Redefines
- Binary / 'Special' Fields
• Each Segment Maps to One (1) or More Tables
• Strong Target Data Types May Require Additional Transformation
• Tendency to Over Design / Over Normalize
• Still Required for Relational Type Targets (IDAA, Netezza, Teradata, etc.)
Design → IMS to Big Data

- De- Normalized / Minimal Normalization
- Still Requires Transformation (dates, binary values, etc.)
- Good News → IMS Structure Already Setup for Big Data

```
{ "order_no" : "12345",  
  "cust_no" : "20223",     
  "price" : 23.95,         
  "Lines" : { "item" : "Widget1",  
              "qty" : "6",   
              "cost" : "2.43"  
            },  
            { "item" : "Widge2y"  
            "qty" : "1",   
            "cost" : "9.37"  
            },  
  }  
```

```
{ "company_name" : "Acme",  
  "cust_no" : "20223",     
  "contact" : { "name" : "Jane Smith",  
                "address" : "123 Maple Street",  
                "city" : "Pretendville",  
                "state" : "NY",  
                "zip" : "12345" }  
}
```
Populating Big Data Repositories from IMS

- Big Data Overview
- Outline Concepts
- Streaming Scenarios
- Q & A
IMS Data Capture Methods

Primary Methods of Capture

- Data Capture Exit Routines
- Log Based

1. Database Capture Exit Routines
   - Near-Real-Time for IMS TM/DB
   - Extremely Fast and Efficient
   - Scalability → Capture / Apply by FP Area, HALDB Partition, PSB, Database
   - Does Not Require x'99' Log Records

2. Log Based
   - Near-Real-Time or Asynchronous
   - CICS / DBCTL Environments
   - Requires x'99' Log Records
   - Scalability → Same as Database Exit Routines
**Optimal Solution:**

- Sub-Second Latency → Capture to Apply
- Must be able to Handle High-Transaction Volume
- Multi-Purpose is a Major Plus
- Publish Should *Not* Require any Extra Parts
  - No Staging Tables
  - No Queues
- Must be Resilient / Fault Tolerant
- Basic Distributed File System
- Append-Only Writes
- Eventually Consistent
- 1 Writer → Multiple Readers
- Ideal for Streams / Data Lakes
- Batch or Near-Real-Time Apply
HBase

- NoSQL on top of Hadoop HDFS
- Eventually Consistent
- Search Engines / Analyzing Logs
- Batch Apply Frequency
• **HDFS Format → CSV, JSON, XML, Custom**
• **Typical Use → Multiple Files for Same Content**
  - File Size Based on # Records / Time Interval
  - Requires Multi-File Management
• **Partitioning → Based on Source Value(s)**
  - Not Native in HDFS
  - Based on Source Data Value(s)
• High-Throughput, Low-Latency Message Broker
• Open Sourced by LinkedIn 2011 / Apache 2012
• Supports a Variety of Targets → More on the Way
• Leverage JSON Message Format for CDC

Use Cases:
• Basic Messaging → Similar to MQ
• Website Activity Tracking
• Metrics Collection / Monitoring
• Log Aggregation
• Streaming
Cassandra

- NoSQL – Unique Keys
- Eventually Consistent
- Highly Scalable
- Great Read / Write Performance
- No Joins
- Data Typically Denormalized
mongoDB

- NoSQL – Document Store (JSON/BSON)
- Eventually Consistent
- Keys Not Required to be Unique
- Great for Dynamic Queries
- Not Extremely Scalable
Performance: Cassandra vs HBase vs MongoDB

Read/Write Mix Workload

http://planetcassandra.org/nosql-performance-benchmarks/
DB2 PureData Analytics (Netezza)

- Standalone Analytics Appliance
- Consistency, Partition tolerance
- Batch Apply Frequency
Integrated DB2 Analytics Accelerator (IDAA)

- Coupled with DB2 z
- Consistency, Partition tolerance
- Apply through DB2 → AOTs
- Batch Apply Frequency
- Requires IDAA PTF 5
DB2AA Replication Considerations

- Accelerator Must Know About Apply Processes
- Required: PTF 5
- Supports User Written Apply
- Accelerator Only Tables (AOTs)
  - Allows Update DML against Tables in Accelerator
  - Apply Process can Perform Inserts/Deletes via DB2
  - Decent Throughput Today → Will Only Get Better in the Future

- AOT Restrictions
  - Currently only Supported in DB2 V10
  - Single Row Inserts – Multi-Row Inserts in Development
  - Transient in Nature
  - Cannot be Enabled for Incremental Update
  - Cannot Backup/Recover via Utilities
Spark

- Super Fast Engine for Data Processing
- Supports Multiple BD Stores
- Started 2009 → UC Berkley
- Donated to Apache in 2013
- 100x Faster than MapReduce
- 10x Faster from Disk
- Highly Popular at the Moment
Spark Streams

- Real-Time Feeds into Spark
- Batching Apply Method → Short Bursts
- Each Batch is a Resilient Distributed Dataset (RDD)
Summary

• Let the Business Drive the Effort
• Temper the Exuberance
• Keep the Fiefdoms at Arm's Length
• Use an Iterative Approach for Implementation
• Keep an Open Mind with Regard to Technology
Thank you for your time.

Rick Weaver
Senior Customer Engineer
+1.613.523.5500
info@dkl.com