Real-Time Streaming
IMS to Big Data

Prepared for the:
IMS Tech Symposium

8 March 2016
Briefing Objectives

- Address Practical Approach to Real-Time IMS Data Feeds
- Tool/Product Agnostic
- 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
- Address Any Questions that You May Have
About the Speaker

Scott Quillicy

- 35 Years Database Experience
- Database Software Development
- Performance & Availability

Founded SQData to Provide Customers with:

- A Better Way of Replicating Mainframe Data → Particularly IMS
- Solutions that Combine Expertise with Technology
- Technology Built Around Best Practices

Specialization

- Database Trends and Direction
- Data Replication
- IMS to Relational
- Big Data Streaming
- Continuous Availability
- Data Analytics
About SQData

➢ Enterprise Class Changed Data Capture (CDC) & Replication

➢ Specialization
  ✓ High-Performance Changed Data Capture (CDC)
  ✓ Non-Relational Data → IMS, VSAM, Flat Files
  ✓ Relational Databases → DB2, Oracle, SQL Server, etc.
  ✓ Deployment of Complex Data Integration Solutions
  ✓ Continuous Availability of Critical Applications
  ✓ Data Conversions / Migrations

➢ Customer Use Cases
  ✓ Real-Time Operational Data Stores / Big Data → Multiple Sources
  ✓ Continuous Availability → Active-Active, Active-Passive
  ✓ ETL (Bulk Data Extracts/Loads)
  ✓ Application Integration
  ✓ Business Event Publishing
  ✓ Data Warehouse Population
  ✓ Application Integration
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...

➢ 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 DB2 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
## Analytics → Use Cases by Industry

<table>
<thead>
<tr>
<th>INDUSTRY</th>
<th>USE CASE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sensor</td>
</tr>
<tr>
<td>Financial Services</td>
<td>New Account Risk Screens</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trading Risk</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insurance Underwriting</td>
<td></td>
</tr>
<tr>
<td>Telecom</td>
<td>Call Detail Records (CDR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Infrastructure Investment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Real-time Bandwidth Allocation</td>
<td></td>
</tr>
<tr>
<td>Retail</td>
<td>360° View of the Customer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Localized, Personalized Promotions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Website Optimization</td>
<td></td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Supply Chain and Logistics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assembly Line Quality Assurance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Crowd-sourced Quality Assurance</td>
<td></td>
</tr>
<tr>
<td>Healthcare</td>
<td>Use Genomic Data in Medial Trials</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monitor Patient Vitals in Real-Time</td>
<td></td>
</tr>
<tr>
<td>Pharmaceuticals</td>
<td>Recruit and Retain Patients for Drug Trials</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improve Prescription Adherence</td>
<td></td>
</tr>
<tr>
<td>Oil &amp; Gas</td>
<td>Unify Exploration &amp; Production Data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monitor Rig Safety in Real-Time</td>
<td></td>
</tr>
<tr>
<td>Government</td>
<td>ETL Offloaded Response to Federal Budgetary Pressures</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sentiment Analysis for Government Programs</td>
<td></td>
</tr>
</tbody>
</table>

Best Practices Summary

- **Let the Business Drive the Effort**
  - Ensures Business Goals are Met
  - Queries Drive the Data Model Design
  - Avoid I/T Initiated 'Build it and They will Come' (i.e. the EDW)

- **Temper the Exuberance**
  - Inevitable After Successful Implementation for a Given Application
  - Important to Refine Processes / Set Guidelines
  - It is More Expensive than the Hype Leads You to Believe

- **Keep the Fiefdoms at Arm's Length**
  - Departmental Groups Who are Working on Their Own Big Data Project
  - May Result in 'Mine is Better than Yours' Issues
  - I/T Circumvention is to be Expected

- **Keep an Open Mind with Regard to Technology**
  - Technology is Rapidly Evolving
  - What is OK Today may be Obsolete Tomorrow

- **Use an Iterative Approach for Implementation**
  - Set the Relational Mindset Aside
  - Allows for 'Adjustments' without Major Schedule Impact
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**
  - DB2AA / PureData Analytics (Netezza)
  - Cassandra
  - MongoDB
  - More Appearing each Day... 😊
Concepts
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 Analytics
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

```
A
   B
   D
   C
   E
   F
```

```
A B C C B C D E E F
```

or

```
A B C C B C
```

```
A D E E F
```

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

- **Frequency**
  - Near-Real-Time
    - Continuous Stream
    - Low Latency → Typically Sub-Second, but May be a Bit Higher for Larger Transactions
  - Batches
    - Triggered by # Records and/or Time Interval
    - Time Based
    - Latency Varies

- **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)
    - Get Latest Image of Record by Key(s)
    - Filter Out Deletes
    - Merge into 'Master' File on Periodic Basis
## CDC / ETL Data Format(s)

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

### Sample Update CDC Record in JSON Format

```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"
  }
}}
```
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
  ✓ Non-Numeric Data in Numeric Fields
  ✓ Binary Zeros in Packed Fields (or Any Field)
  ✓ Invalid Data in Character Fields

➢ Dates
  ✓ Must be Decoded / Validated if Target Column is DATE or TIMESTAMP
  ✓ May Require Knowledge of Y2K Implementation
  ✓ Allow Extra Time for Date Intensive Applications

➢ Repeating Groups
  ✓ Sparse Arrays
  ✓ Number of Elements
  ✓ Will Probably be De-normalized

➢ Redefines

➢ Binary / 'Special' Fields
  ✓ Common in Older Applications Developed in 1970s / 80s
  ✓ Generally Requires Application Specific Translation
Design → Traditional IMS to Relational

- 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 (DB2AA, 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
<table>
<thead>
<tr>
<th>Key</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order#</td>
<td>Cust#</td>
</tr>
</tbody>
</table>

Line Item

```

{ "company_name" : "Acme",
  "cust_no" : "20223",
  "contact" : { "name" : "Jane Smith",
                  "address" : "123 Maple Street",
                  "city" : "Pretendville",
                  "state" : "NY",
                  "zip" : "12345" }
}
Streaming IMS to Big Data Stores
IMS Data Capture Methods

➢ Primary Methods of Capture
  ✓ Data Capture Exit Routines
  ✓ Log Based

➢ 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

➢ Log Based
  ✓ Near-Real-Time or Asynchronous
  ✓ CICS / DBCTL Environments
  ✓ Requires x'99' Log Records
  ✓ Scalability → Same as Database Exit Routines
IMS Streaming Illustration

**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
Hadoop HDFS

- Basic Distributed File System
- Append-Only Writes
- Eventually Consistent
- 1 Writer → Multiple Readers
- Ideal for Streams / Data Lakes
- Batch or Near-Real-Time Apply

Source: http://dailyhadoopsoup.blogspot.com/
HBase

- NoSQL on top of Hadoop HDFS
- Eventually Consistent
- Search Engines / Analyzing Logs
- Batch Apply Frequency
Streaming to Hadoop

- 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)
  - Requires Cross-Partition Multi-File Management

Capture/Publish

Apply

Apply

ODBC/JDBC

Native HDFS

HDFS

Hive

HBase

ODBC/JDBC

Hadoop

©Copyright SQData Corporation 2016 – All Rights Reserved
Kafka

- 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

```

db.xxxx.insert
db.xxxx.update
db.xxxx.remove
```

**Capture/Publish**

**Apply Engine**

**JSON**

```

kafka
```

**User Apply**

**MongoDB**

©Copyright SQData Corporation 2016 – All Rights Reserved
Performance: Cassandra vs HBase vs MongoDB

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

- Standalone Analytics Appliance
- Consistency, Partition tolerance
- Batch Apply Frequency

Receive / Transform / Acknowledge

Apply

Apply Thread
Apply Thread
Apply Thread

Controller

PureData Analytics

Capture

Publish
DB2 Analytics Accelerator (DB2AA)

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

Receive / Transform / Acknowledge

Apply Thread
Apply Thread
Apply Thread

Staging

DB2

Controller
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 Berklely
- Donated to Apache in 2013
- 100x Faster than MapReduce
- 10x Faster from Disk
- Highly Popular at the Moment

Standalone
Spark Streams

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

Source: http://www.databricks.com/
Summary

➢ Let the Business Drive the Effort
➢ Temper the Exuberance
➢ Keep Fiefdoms at Arm's Length
➢ Use an Iterative Approach for Implementation
 ➢ Keep an Open Mind with Regard to Technology
➢ For More Information:
✔ Visit the Infotel / Insoft Booths in the Expo Area
✔ www.infotel.com
Thank You!!
Real-Time Streaming

IMS to Big Data

Prepared for the:
IMS Tech Symposium

8 March 2016