Building Event Driven Architectures using OpenEdge CDC

Richard Banville, Fellow, OpenEdge Development
Dan Mitchell, Principal Sales Engineer
October 26, 2018
Agenda

- Change Data Capture (CDC) Overview
- Configuring data capture via policies
  - OpenEdge Explorer/Management
  - ABL APIs
- Data Capture
  - Change Tracking
  - Recording of data value changes
- CDC data as part of an event driven architecture
What is Change Data Capture?

- Ability to intercept data as it changes
- Change data storage
  - Relational tables
  - Flat change logs
Why Change Data Capture?

- **Data Warehousing**
  - Repository of data from various sources
  - Input to strategic business decisions

- **Avoid warehouse bulk reload**
  - Time consuming
  - Some data is irrelevant to business needs
  - Requires coordination of data sources

- **Business needs**
  - Capture changes to data of interest only
  - Extract, Transform, Load (ETL)
    - Tools / mechanisms process captured data
OpenEdge Change Data Capture

- Policy driven configuration
- Database triggers
  - Capture data within same DB
- Change Tracking
  - Event occurrence
  - Transaction and time stamp
  - Unique identification
- Change Data
  - Record changes of field data in “native” format

Diagram:
- OpenEdge Data Source
  - OLTP
  - Source Tables
  - Capture process
  - Changed Data
  - ETL Processing
  - Data Warehouse
  - Reporting
  - OLAP analysis
  - Data Mining
Why OpenEdge Change Data Capture?

- Alternative to trigger-based replication
- Eases deployment costs
  - Configuration
  - Maintenance
  - Handling of schema changes
- And it’s fast
  - Using CDC, Pro2 experienced 2.5x performance improvement
- ETL languages
  - ABL – write your own
  - SQL – yours or off the shelf
Change Data Capture in your Enterprise

- Data Files
- Governance?
- Alerting?
- Monitoring?
- Cleansing?

OLTP DB Source tables

CDC ETL Data

Data Warehouse

OLAP analysis

Data Mining

Reporting
Change Data Capture in your Enterprise

OLTP DB Source tables

OLTP DB Source tables

Data Files

Governance!

Alerting!

Event stream platform

Monitoring!

Cleansing!

Data Warehouse

Reporting

OLAP analysis

Data Mining
Change Data Capture in your Enterprise

- Greater insights drive growth
- Increased efficiency
- Enhances visibility into the Enterprise
- Improves operational productivity
First things first:
Policy Configuration
Enabling CDC for a Database

- Command line or via Database Admin Console in OEM/OEE
  - Enabled online or offline

```bash
proutil <db> -C enableCdc
area CDC_Tracking_Data
indexarea CDC_Tracking_Index
deactivateidx
```
Configuring CDC Policies in OpenEdge

OpenEdge Management & Explorer

Dump & Load Policy

Change Data Capture ABL API
Change Data Capture – Policies

- Define source table & data capture levels
- Maintained through
  - OpenEdge Explorer/Management - Database Management Console (GUI)
  - ABL APIs (programmatic)
  - “Special” dump and load
- CDC Policies:

<table>
<thead>
<tr>
<th>Policy Info</th>
<th>Policy Table Name</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDC source table policy</td>
<td>_Cdc-Table-Policy</td>
<td>One record per source table</td>
</tr>
<tr>
<td>CDC field info policy</td>
<td>_Cdc-Field-Policy</td>
<td>One record per field per source table</td>
</tr>
</tbody>
</table>
Policy Levels

- Levels build on each other
- Amount of data collected is flexible
  - Based on your business needs
- Levels 1 thru 3 require fields specification

- **Level 0:** Record in Change Tracking that a change occurred
- **Level 1:** Level 0 + Identifies which fields changed
- **Level 2:** Level 1 + records current value of fields
- **Level 3:** Level 2 + records previous value of fields

▪ Levels build on each other
▪ Amount of data collected is flexible
▪ Based on your business needs
▪ Levels 1 thru 3 require fields specification
Configuring Record Change Policy
Configuring Field Change Policy

Change Table Properties
CDC Field Policy - Level >0

- A field policy is required
- Unlimited field policy entries allowed
- Change data captured only for selected fields
- Identifying fields 1 to 15
  - Indexing optional

Table showing selected change data capture fields:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Enable identifying field</th>
<th>Field order</th>
</tr>
</thead>
<tbody>
<tr>
<td>BillToID</td>
<td>integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Carrier</td>
<td>character</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Creditcard</td>
<td>character</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>CustNum</td>
<td>integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Instructions</td>
<td>character</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>OrderDate</td>
<td>date</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Ordemnum</td>
<td>integer</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>
Change Data Capture Policy Information

- Schema changes to source tables

<table>
<thead>
<tr>
<th>Schema Change</th>
<th>Affect / rules</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a new field</td>
<td>CDC will not be affected</td>
<td>Needs field policy for capture</td>
</tr>
<tr>
<td>Renaming a field</td>
<td>Cascades to Change Table fields</td>
<td>This is automatic</td>
</tr>
<tr>
<td>Deleting a field</td>
<td>CDC no longer captures data for it</td>
<td>Existing data remains</td>
</tr>
<tr>
<td>Renaming the table</td>
<td>No change for CDC</td>
<td></td>
</tr>
<tr>
<td>Deleting the table</td>
<td>No active CDC policy can exist</td>
<td>Policy must already be deleted</td>
</tr>
</tbody>
</table>

- Policies can be dumped (to .cd file) and loaded
  - Via Database Administration or Database Admin Console
  - For load, the associated schema must match the original database
Data Capture
OpenEdge Change Data Capture – Table Relations

**Change Tracking Table**
- Record specific change occurrences
- One per source database
- Name: Cdc-Change-Tracking

**Data Change Tables**
- Records actual data that changed
- Multiple change data fields
- One table per source table
- Name: CDC_\(<source name>\)
  - CDC_Customer
### Change Tracking Table – Record Format

<table>
<thead>
<tr>
<th>_Policy-Id</th>
<th>_Tran-Id</th>
<th>_Time-Stamp</th>
<th>_Change-Sequence</th>
<th>_Operation</th>
<th>_Change-Field-Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Recid</td>
<td>_Source-Table -#</td>
<td>Partition-Id</td>
<td>_Tenant-Id</td>
<td>_User-Misc</td>
<td>_User-Name</td>
</tr>
</tbody>
</table>

- **Ordering:** `_Source-Table`, Time Stamp, `_Change-Sequence`
  - unique per policy
- **Operation:** 1: Create, 2: Delete, 3: After Update, 4: Before Update, 5-11: FIELD deleted in some way
- **Change Field Map:** Identifies which fields changed
- **_User-Misc:** User/application updateable
  - Intended use: identify a change has been processed (i.e. loaded into the warehouse)
Change Tables – Record Format (field level data)

- No change table for CDC policy level 0
- **One** record per Create*, Delete operation
- **Two** records for update operation if policy level is 3

**Change Table – CDC_customer**

<table>
<thead>
<tr>
<th>_Tran_id</th>
<th>_Time-Stamp</th>
<th>_Change-Sequence</th>
<th>_Continuation-Position</th>
<th>_ArrayIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Fragment</td>
<td>Capture Col1</td>
<td>Capture Col2</td>
<td>Capture Col3</td>
<td>...</td>
</tr>
</tbody>
</table>

- Ordering of changes by Change-Sequence or Time-Stamp
- Child reference of Change Tracking table
- Also contains identifying field information
Change Tables – Record Format

- No change table for CDC policy level 0
- One record per Create*, Delete operation
- Two records for update operation if policy level is 3

Field level data requested
- One column for each field

<table>
<thead>
<tr>
<th>Change Table – CDC_customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Tran_id</td>
</tr>
<tr>
<td>_Fragment</td>
</tr>
</tbody>
</table>

*Create operations for CDC policy levels 1 and 2 create a single record.
Change Tables – Record Format

- No change table for CDC policy level 0
- **One** record per Create*, Delete operation
- **Two** records for update operation if policy level is 3

(change table example)

- **_Fragment**: Identifies split record order (> 32,000 bytes)
- **_Continuation-Position**: Which field was split
- ABL built-in methods help deal with this
Processing Captured Changes

Using an Event Driven Architecture
But first: How to access CDC Records?

**ABL Batch Client**  e. g. reading new entries on a scheduled basis.
- Find new entries by _Source-Table-Number and _Change-Sequence or _Time-Stamp
- Build the “piece of information” to publish
- Update field „_cdc-Change-Tracking._User-Misc“ for tagging processing status.

**SQL-Client**  e. g. reading a filtered set of records.
- Add Views to provide de-normalized data, including aggregates
- Access the CDC tables in DB Schema „PUB“

**PASOE**  Provide an API for processing CDC logic
- e. g. Dynamic fetches
- Expose Operations via REST
How to build the Entity that gets published

Chg.-Tracking Table
- Identifies Change & Operation
- Fields that changed
- Time-Stamp and Sequence
- Update _User-Misc

Change Table
- Source and Operation
- field values
- Identifying fields
- Difference (numeric & level 3)

Source Table
- Add “static fields“ values
- Foreign Keys to related tables

Related Tables
- Create de-normalized records, meaning full entities of business information

Relation by _Source-Table-Number, _Change-Sequence
Relation by Primary Key or _cdc-Change-Tracking, _Source-Table-Number & _Recid

Publish this piece of information = Event
An Event-Streaming Architecture

Producers

- Kafka JDBC Connect

Consumers

- E-Mail Wrapper
- Spark Client
- Redshift Client

Any Messaging System

Change Data turned into Business Events

CDC Enabled

Alert on Order Status

Analyze in real-time and enable data science

Warehouse for audit and analytics
Apache Kafka in a Nutshell

- Kafka is a Messaging System, handling **Key-Value Entries**. Originally developed by LinkedIn.
- Kafka runs on a cluster of one or more servers (called brokers), distributes and replicates partitions for performance and fault-tolerance.
- Arbitrarily many processes called **Producers** feed messages into different "partitions" within different "topics".
- Other processes called **Consumers** can read messages from Topics.
- Another Server process called **Zookeeper** coordinates concurrent consumer access to Kafka.
## Demo Producers and Consumers

<table>
<thead>
<tr>
<th>Our Producers</th>
<th>Our Consumers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• are “Kafka JDBC Connect“ instances</td>
<td>• 3 custom programs</td>
</tr>
<tr>
<td>• reading records from Change Tables via SQL &amp; JDBC</td>
<td>• reading new entries from topics on a scheduled</td>
</tr>
<tr>
<td>• and passing it into a Kafka Topic named by the OE Source Table Name</td>
<td>bases</td>
</tr>
<tr>
<td>• executed each 5 seconds implementing Polling</td>
<td>• passing these into Target Applications using their</td>
</tr>
<tr>
<td></td>
<td>APIs</td>
</tr>
</tbody>
</table>
Our “Target Applications”

An E-Mail Client
• Receiving Order Information in a human-readable format.

An Apache Spark Database providing Real-Time Analytics
• Batching the stream of records into 20 sec batches, applying analysis instantly.

An Amazon Redshift Data Warehouse
• Which gets kept in synch with the Source Database by the CDC track
The full picture

Producers

- Kafka
- JDBC Connect

Consumers

- E-Mail Wrapper
- Spark Client
- Redshift Client

Alert on Order Status

- Analyze in real-time and enable data science
- Warehouse for audit and analytics

Any Messaging System

- Change Data
- SQL-Editor

CDC Enabled
The full picture

Producers

- Kafka
- JDBC Connect

Event stream

Consumers

- E-Mail Wrapper
- Spark Client
- Redshift Client

Alert on Order Status:

- Analyze in real-time and enable data science

Warehouse for audit and analytics:

- Spark Streaming

Any Messaging System:

- Any Messaging System

SQL-Editor:

- SQL-Editor

Change Data

CDC Enabled

Progress OpenEdge'
For more information

- Blog on Event Architecture with Kafka by Saikrishna Teja Bobba

- Documentation OpenEdge CDC

- How-To Videos introducing OpenEdge CDC
Change Data Capture (CDC) in OpenEdge 11.7

Easy to use
- Policy driven with GUI configuration
- Multiple configuration levels
- Code can be generated for you

Performant
- Recorded by internal database triggers (similar to auditing)
- Data stored in multiple "change" tables

Flexible
- Data stored in "native format"
- Schema change aware
- ETL helper function for ABL and SQL

A component for delivering event driven architectures in your enterprise
Thank You!

Richard Banville
Dan Mitchell