Progress

OpenEdge Change Data Capture and the ETL Process

WHITEPAPER

AUTHOR : LAKSHMI PADMAJA

Introduction

Keeping the multitude of data sources housed within an organization updated is a cumbersome and time intensive process. Having accurate data at your fingertips helps drive business success, and the ability to precisely capture changes and efficiently export and synchronize those changes with data repositories and warehouses enhances visibility and operational productivity.

Progress OpenEdge Change Data Capture (CDC), a new OpenEdge 11.7 feature enhancement, acts as a tracking mechanism that enables applications to implement a process that automatically logs changes made to user tables in an OpenEdge database. It also supports the automatic copying of subsets to change tables. The data gathered through the change data capture process can be utilized by a category of tools commonly known as ETL (extract, transform, load) processes for analytics or reporting purposes. An example of how this data might be used is to identify business trends using data housed in a data warehouse, whereby the accuracy of that data has been provided by synching organizational data sources using data which has been populated from OpenEdge CDC change tables through an ETL process.

What is CDC?

Change data capture (CDC) is an industry term used to describe the duplication of subsets of OLTP data in an external data source with a relatively up to date version of relational data. Implementations of CDC can involve an ETL application that extracts individual data changes per table.

Using CDC in the OpenEdge database is a two-step process:

1. Enable CDC on OpenEdge database
2. Define CDC policies for user tables and fields

Enable CDC on OpenEdge Database

CDC can be enabled in the database in two ways. The first is through a database utility. The enablement installs the underlying product support for enabling CDC. The data captured is controlled by using policies. Policies define what tables and fields are tracked by the CDC. Policies can be created and managed using ABL APIs or tools like OpenEdge Management (OEM) or OpenEdge Explorer (OEE). The captured data is then stored in a source specific Change Table.

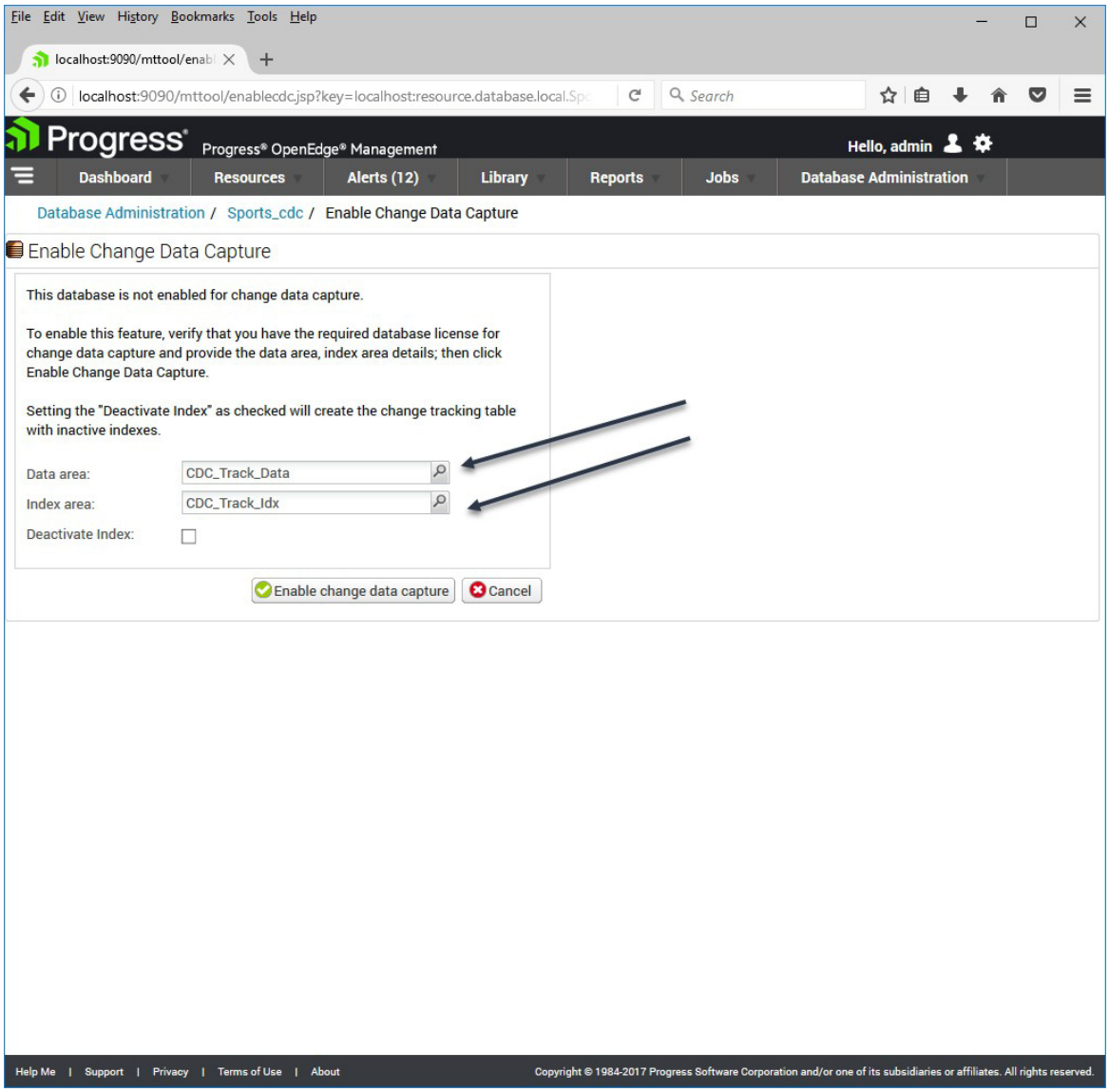
Steps to enable CDC on the database

proutil command is used to enable CDC on any given database(exampleCDCDB):

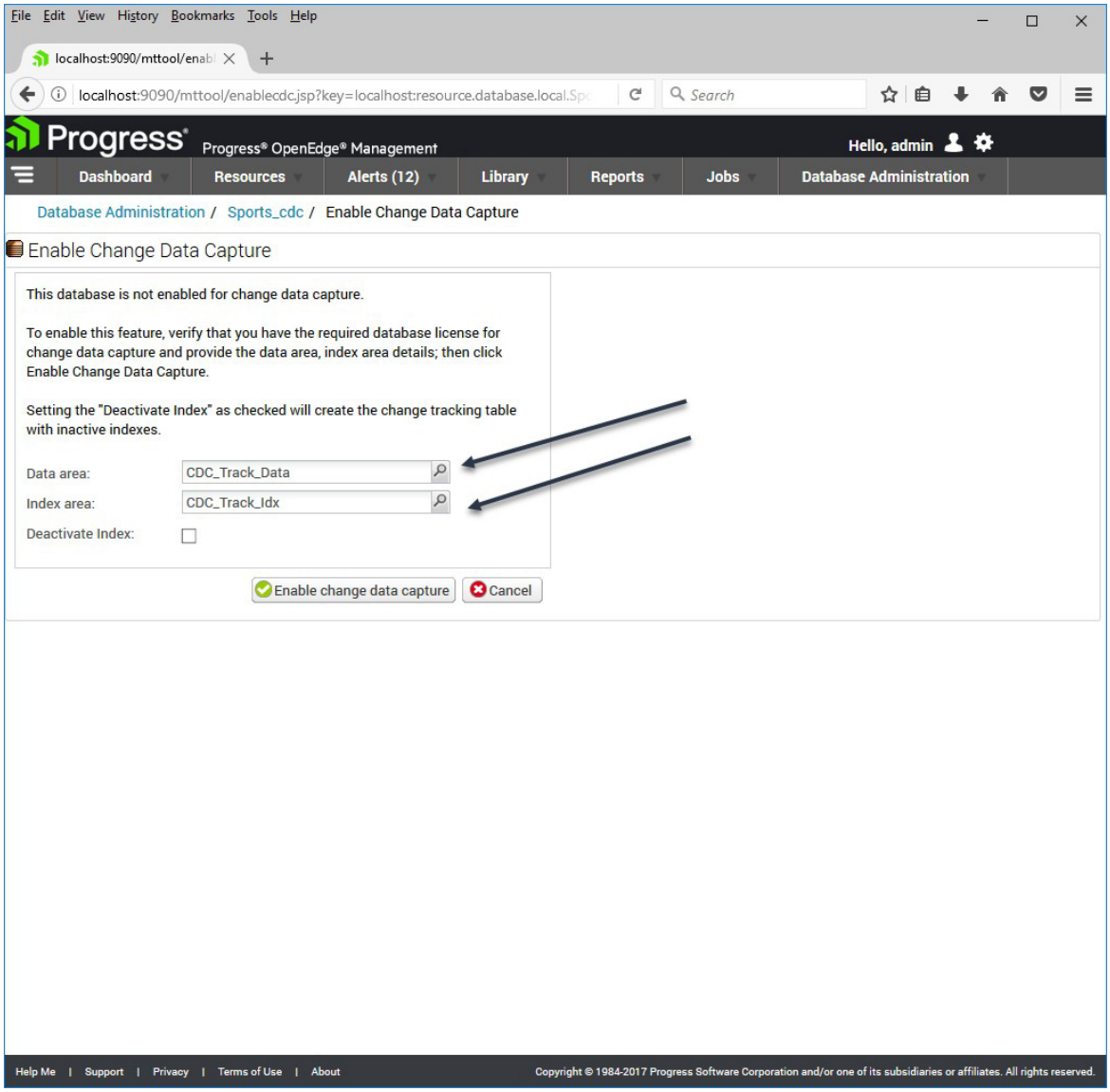
```
proutil exampleCDCDB -C enablecdc area “CDC Tracking Area” indexarea “CDC Index Area”
```

Above command enables CDC feature on database “exampleCDCDB” with area as “CDC Tracking Area” and indexarea as “CDC Index Area”. The area in this command specifies where the CDC tracking table will be located. The indexarea is where the CDC indexes will be located. Both must be specified, they should be type II areas. The second way to enable CDC against the database is through OpenEdge Explorer or OpenEdge Management.

This method results in the same solution as using the database utility but is graphically driven. This is done through the Database Administration tab by checking the “enable” link for Change Data Capture.



Then input the storage areas for each of the required areas.



To disable CDC on the database use:

```
proutil exampleCDCDB -C disablecdc
```

Create Policies for User Tables and Fields

After enabling the CDC on the database, capturing data with CDC requires you to create policies that define the tables and fields that are tracked, and the amount of change recorded. Create policies with the Database Administration Console of OpenEdge Management or OpenEdge Explorer or through ABL API. Once the new policy is defined, created, and committed, the policy becomes active and each time a Create, Update or Delete (CUD) record operation is executed, a record will automatically be written to the Change Tracking Table and the Change Table assigned to the policy.

Policy Levels:

When defining a CDC policy, the level you specify determines the amount of change tracking data collected. The following table gives a description of the data captured at different levels:

Level	Description	Contains change Fieldmap?	Change Level Allowed?
0	Stores data in tracking table only	Null	No
1	Stores data in tracking table only. Includes Fieldmap.	Reflects changed fields in update only	Yes
2	This level records the current (after) value of all CUD operations.	Reflects changed fields in update only	Yes
3	This level records both the previous (before) and current (after) values of all CUD operations.	Reflects changed fields in update only	Yes

CDC Field Policy Table definition allows the policy designer to indicate which source fields are of interest for data capture. There is one field policy per field value to be captured. The policy designer can request identifying fields specifically tied to a field value and an optional index based on those identifying fields.

The following table contains information on basic CDC schema and related tables:

CDC/Schema table name	Description
CDC Table Policy	Defines a database table for CDC and declares which changes the OpenEdge database will capture.
CDC Field Policy	Defines specific fields for which changes should be captured. CDC Field Policy is optional.
_File table	This is a basic schema table for table definitions
Source table	This is the application table for which changes are tracked by CDC.
CDC Change Tracking table	This refers to many individual Change tables implicitly, using references to the table Policy, and the table Policy's source table, for individual changes.
CDC Change table	• All table Policies for a given source table use a single Change table, which contains changes for the given source table only • A Change table has a well-known, default name – the source table name prefixed by "CDC_" and a table pub.customer would have a Change table pub.CDC_customer • It is possible to have a custom Change table name, defined by the DBA creating the CDC table Policy for the source table. • The name of any Change table is contained in its corresponding table Policy definition row.

Steps to create the policy through ABL API

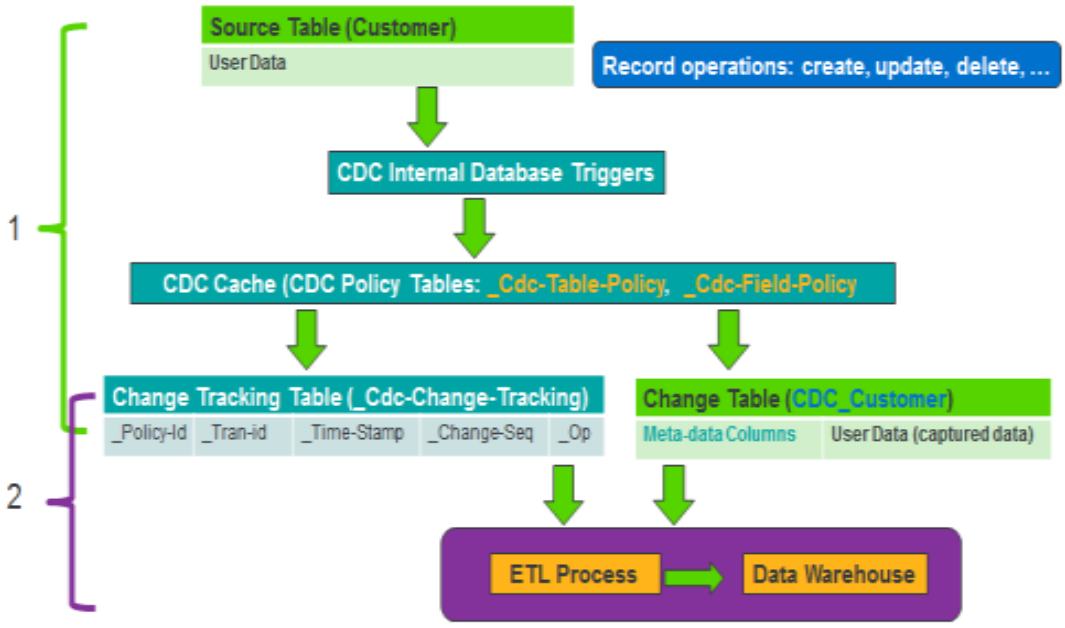
1. Create a policy using ABL command

```
//Create object for
cdcPol = NEW CDCPolicy().
cdcPol:CreateCDCPolicy( db, policyName, tableName, policyDesc ,policyState, policyLevel,
identifyingField,  encryptPolicy,dataAreaName, indexAreaName,flds).
```

Db	Name of the database
policyName	Name of the policy
tableName	Table name (ex customerorder)
policyLevel	CdcTablePolicyLevelEnum Maximum (stating policy Level 3 which records both previous and current values)
identifyingFields	<YES/NO> (to mark the CDC fields as Identifying fields and create an index on the change table)
encryptionPolicy	<YES/NO> (to enable encryption on the change table using the same encryption policy that is in effect for the source table)
dataAreaName	Name of a storage area where the change table is to be created (same name which is part of the structured file during DB creation)
IndexAreaName	Name of a storage area where the change table indices are to be created (same name which is part of the structured file during DB creation)
Flds	Fields for which data has to be captured (more than one field can be specified)

CDC with policies created on the database now enables applications to determine the changes made to user tables in a database. The changes that result from INSERT, UPDATE, and DELETE operations in a user table are tracked, captured and stored in relational tables called change tables. These change tables provide a view of historical data that has been changed over time. The ETL process is a user application program. To understand ETL applications using change data for analytics, we will explore the end to end flow of data from the CDC process through ETL.

How the ETL Process Uses Change Data from a CDC Enabled Database?

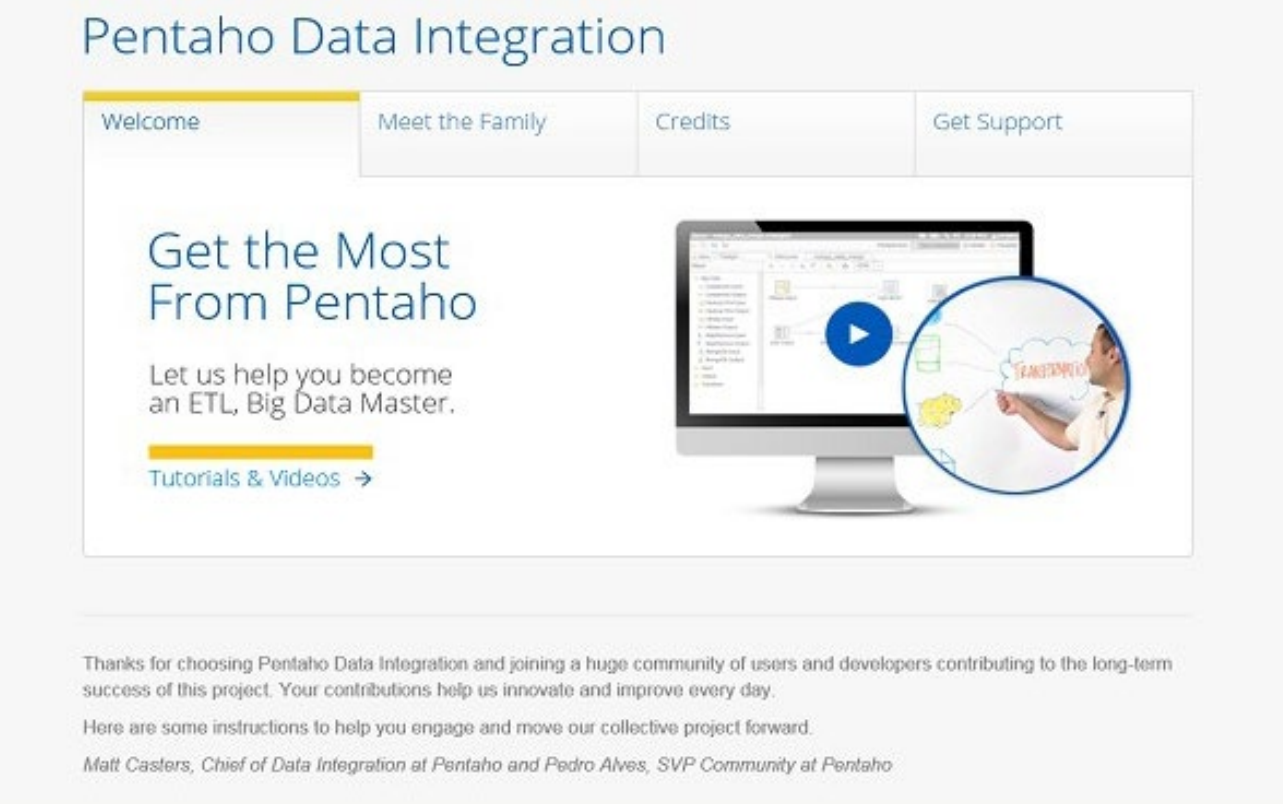


In Figure 1 we provide an abstract look at the end to end flow of the change data capture process through to the ETL process. The brackets, labeled 1 and 2 on the left, represent the two high-level processes. Item 1 represents CDC. Item 2 represents a 3rd party ETL process. Notice that there is overlap in the resources used by these two processes. This overlap is in the CDC Change Tracking Table and the CDC change tables.

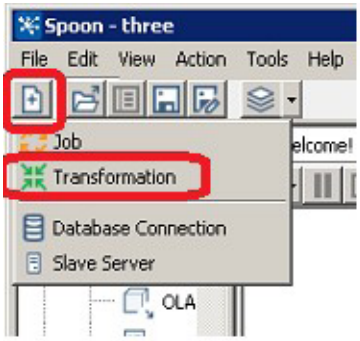
As an example of how the ETL uses change data from CDC, we will use PDI spoon Editor to build and monitor the PDI ETL processing.

Let's create a transformation.

1. Start PDI Spoon editor

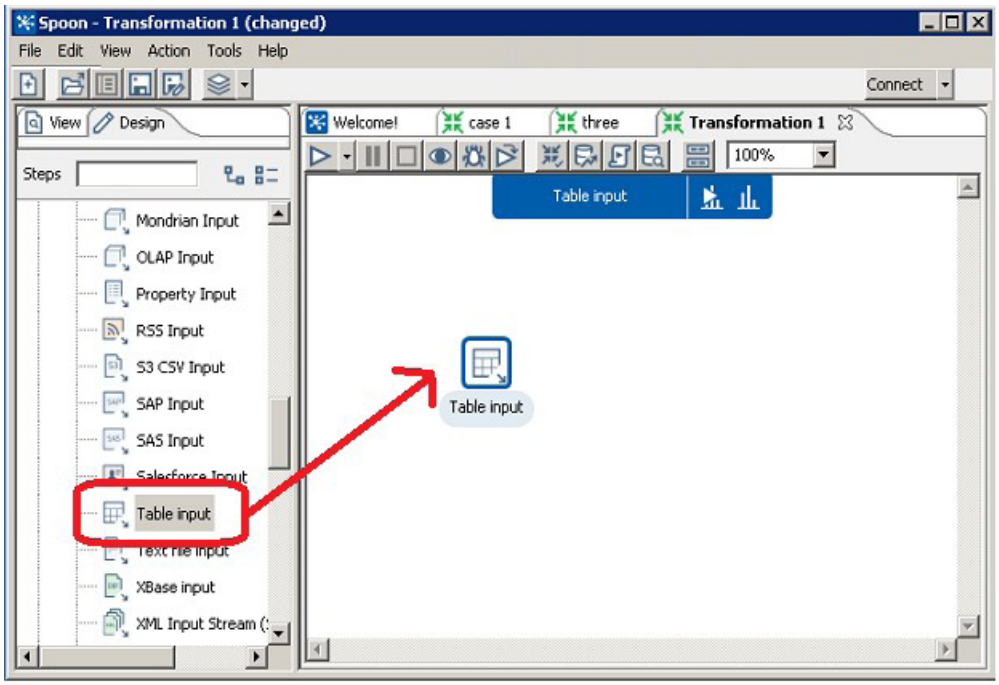


2. Create a new **Transformation either from the File menu or using the tool bar**

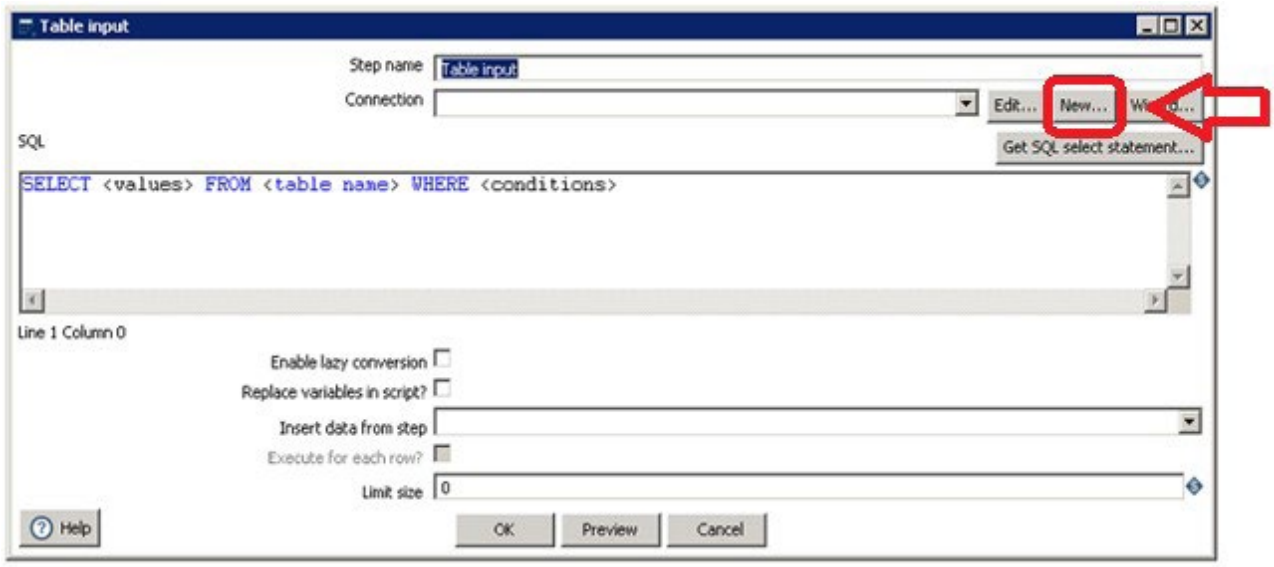


3. Drag **Input->Table input** to the canvas from the left side design panel, using Table input user can write sql queries on the connected database

4. Right click on **Table input** on canvas and click on edit to specify the connection details for the database

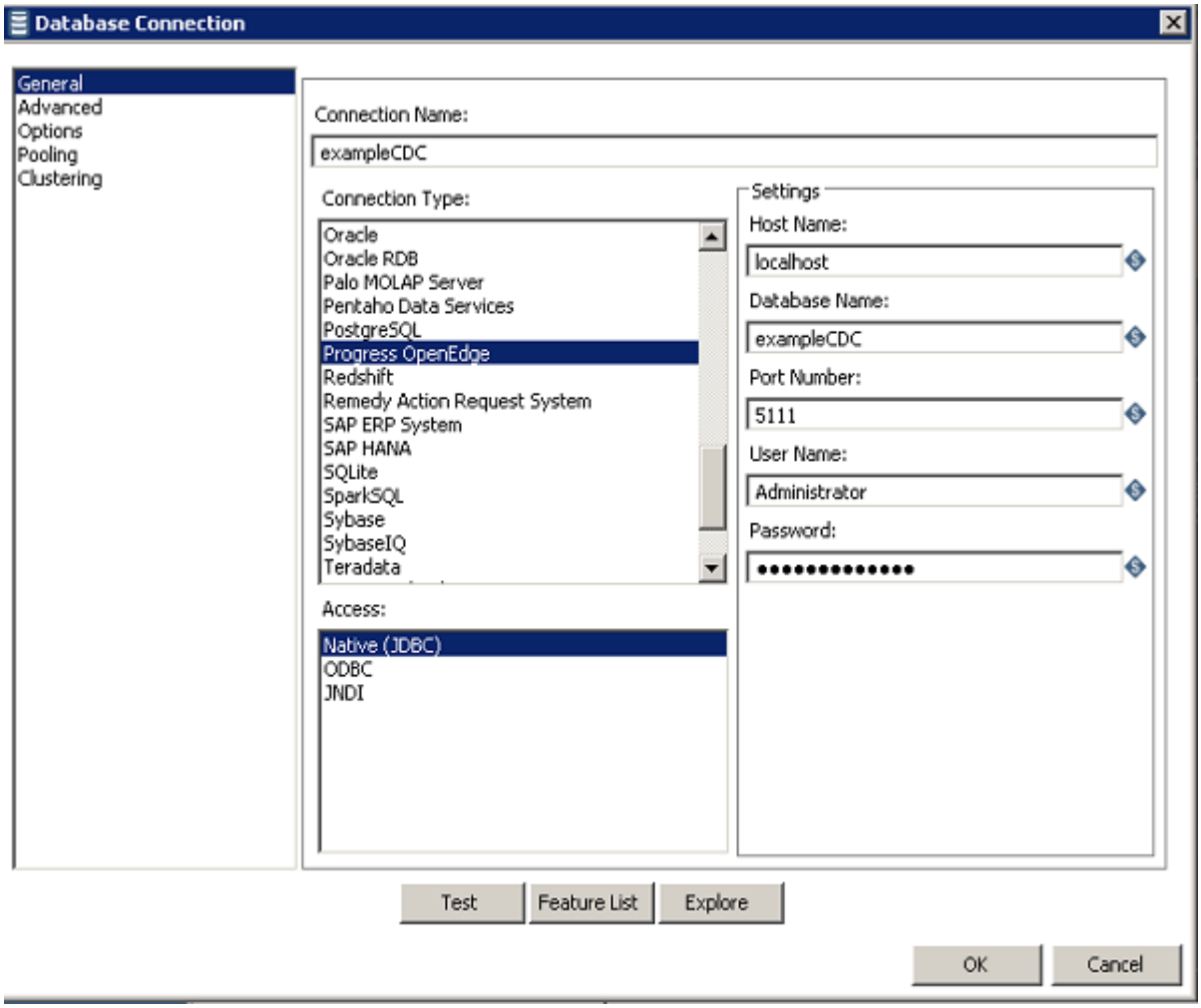


5. Click on **New...** button. Database connection wizard will open

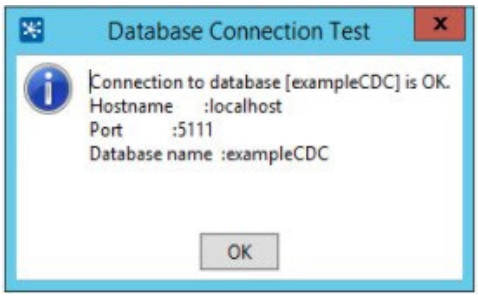


6. Provide input as given below:

- **a.** Connection Name: **exampleCDC**
- **b.** Connection Type: **Progress OpenEdge**
- **c.** Access: **Native (JDBC)**
- **d.** Host Name: **localhost**
- **e.** Database Name: **exampleCDC**
- **f.** Port Number: **5111**
- **g.** User Name: **Administrator**
- **h.** Password: **<Password>**
- **i.** Click on **Test** to validate connection



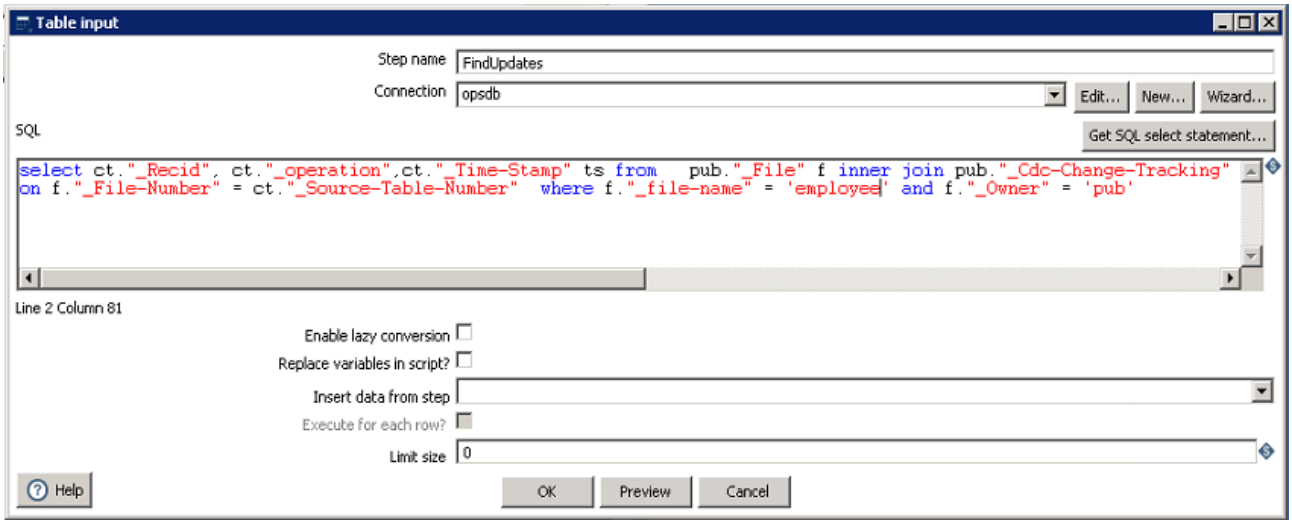
7. Click **OK**



8. Click **OK** to close **Database Connection** wizard

9. Change **Step name** to **FindUpdates**

10. Type the SQL command as shown below which extracts all Change operations on **Employee** table. In this example we are trying to query the change tracking table to get all operations performed (update/insert/delete) on the Employee table.

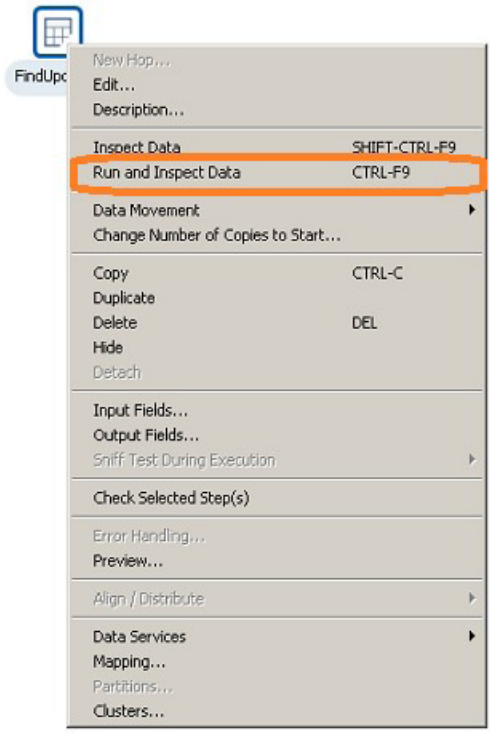


11. Click **OK**

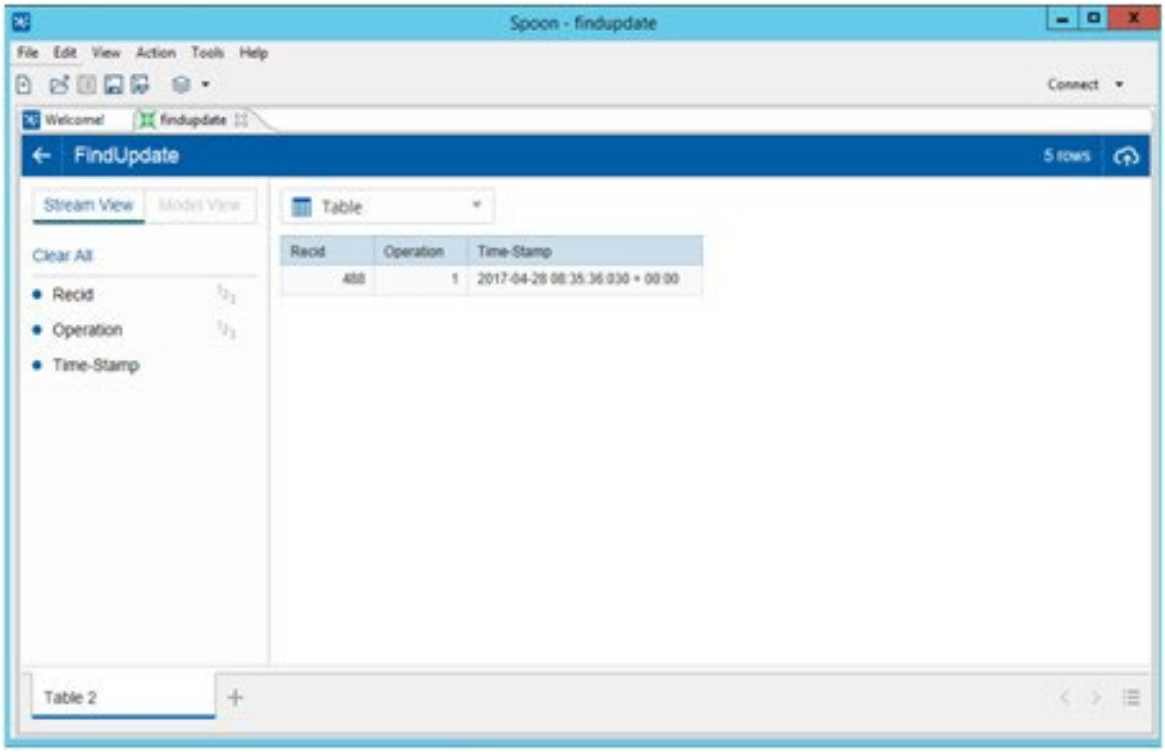
12. Save transformation with name as **FindUpdates**

13. To test, right click on **FindUpdates** in designer canvas

14. Click on **Run and Inspect Data**



Check that you can see output for your table like the one below:



15. Close the preview window.

With this step completed, you have successfully extracted data from change table. Likewise, we can have use cases for extracting the data based on different conditions and below are few use cases:

Sample use cases which can help you in extracting changed data:

1. Find all changes known for a specific Source table, in change order:

When the Table Policy defines Level 0 or Level 1 change tracking, there is no Change table involved. Rather, the Change Tracking table identifies the type of Change and what Source table row changed, among other information. This query will find all changes known for a specific Source table, in change order. The client application sets 2 parameters for the query – table name and owner name of the table. This query will find all changes known for a specific Source table, in change order. The client application sets 2 parameters for the query – table name and owner name of the table.

```
select ct.*                              -- specify change tracking fields here.
from   pub."_File" f
inner join
       pub."_CDC-Table-Policy" tp
        on  f.rowid = tp."_Source-File-Recid"
        and tp."_Policy-Instance"  = 0      -- Policy is current.
inner join
       pub."_Cdc-Change-Tracking" ct
          on tp."_Policy-Id" = ct."_Policy-Id"
where f."_file-name" = ?                    -- table name parameter from ETL app.
and   f."_Owner"     = ?                    -- table name parameter from ETL app.
order by tp."_Policy-Id", ct."_Sequence-Id";
```

Output for sample query:

```
select left(ct."_Policy-Id",20), ct."_Tran-Id", ct."_Time-Stamp", ct."_Operation" from
pub."_File" f inner join pub."_CDC-Table-Policy" tp on  f.rowid = tp."_Source-File-Recid"
and tp."_Policy-Instance"  = 0 inner join pub."_Cdc-Change-Tracking" ct on tp."_Policy-Id"
= ct."_Policy-Id" where f."_file-name" = 'tab1' and   f."_Owner"     = 'pub' order by
tp."_Policy-Id", ct."_Change-Sequence";
left(_Policy-Id,20)     _Tran-Id _Time-Stamp                 _Operation
-------------------- ---------- ---------------------------- ----------------------------
paj8RqV9kqYtFOc2AIJr     27996 2017-09-07 08:03:29:327 - 04:00        1
paj8RqV9kqYtFOc2AIJr     27996 2017-09-07 08:03:29:328 - 04:00        1
paj8RqV9kqYtFOc2AIJr     27996 2017-09-07 08:03:29:509 - 04:00        1
paj8RqV9kqYtFOc2AIJr     27996 2017-09-07 08:03:29:517 - 04:00        1
paj8RqV9kqYtFOc2AIJr     27996 2017-09-07 08:03:29:525 - 04:00        1
```

2. Find all changes known for a specific Source table for a specific time range: This use Case gets all the changes that exist in the Change Tracking table for a given Source table, for a specific time range. 3. Find all changes known for a specific Source table using CDC Scalar functions

```
select ct."_Policy-Id", ct."_Partition-Id", ct."_Recid",
                                         -- Source table recid, partn id
from   pub."_File" f
inner join
       pub."_CDC-Table-Policy" tp
        on  f.rowid = tp."_Source-File-Recid"
        and tp."_Policy-Instance"  = 0      -- Policy is current.
inner join
       pub."_Cdc-Change-Tracking" ct
          on tp."_Policy-Id" = ct."_Policy-Id"
where f."_file-name" = ?                    -- table name parameter from ETL app.
and   f."_Owner"     = ?                    -- table name parameter from ETL app.
and   ct."_Time-Stamp" between ? and ?  -- time range parameters from ETL
group by ct."_Policy-Id",  ct."_Partition-Id",  ct."_Recid" ;
```

The 2nd step uses a query which gives a Result Set of the set of changes for each given recid from 1st query, in change sequence order.

```
select ct.*                          -- specify change tracking fields here.
from   pub."_Cdc-Change-Tracking" ct
where  ct."_Policy-Id"    = ?          -- policy id    from 1st query.
and    ct."_Partition-Id" = ?          -- partition id from 1st query.
and    ct."_Recid"        = ?          -- recid        from 1st query.
order by ct1."_Change-Sequence"      -- should map to index _Part-Rec-Id
                               -- since leading keys are const parameters
```

Output for sample query:

```
select left(ct."_Policy-Id",20),ct. "Partition-Id", ct."_Recid" from  pub."_File" f inner join
pub."_CDC-Table-Policy" tp on  f.rowid = tp."_Source-File-Recid" and tp."_Policy-Instance"
= 0  inner join pub."_Cdc-Change-Tracking" ct on tp."_Policy-Id" = ct."_Policy-Id" where
f."_file-name" = 'tab1' and   f."_Owner"    = 'pub'  and ct."_Time-Stamp" between '2016-09-
07' and '2017-09-09' group by ct."_Policy-Id",  ct." Partition-Id",  ct."_Recid" ;
left(_Policy-Id,20) _Partition-Id          _Recid
-------------------- ------------ -------------------------
paj8RqV9kqYtFOc2AIJr         0          13784
paj8RqV9kqYtFOc2AIJr         0          13785
paj8RqV9kqYtFOc2AIJr         0          13786
paj8RqV9kqYtFOc2AIJr         0          13787
paj8RqV9kqYtFOc2AIJr         0          13788
```

3. Find all changes known for a specific Source table using CDC Scalar functions

This use case also shows how the SQL CDC function CDC_get_changed_columns() can be used to understand and process change data and propagating change details to the data warehouse, and by using SQL Scalar function

```
select fld1, fld2, fld3, fld4, fld5, fld6, fld7, fld9, fld10, fld11,
fld12,
       ID_f1, ID_f2, ID_f3,
       "_Operation",
       CDC_get_changed_columns (pub.CDC_Item, _Change-Fieldmap) as gcc
from   pub."_Cdc-Change-Tracking" ct
inner join
       pub.CDC_Item c                -- assumed Change table name.
          on ct."_Change-Sequence" = c."_Change-Sequence"
where  ct."_Policy-Id"    = ?          -- policy id    from 1st case.
and    ct."_Partition-Id" = ?          -- partition id from 1st case.
and    c."_Operation"    != 3    -- optional - exclude Before Update
```

Output for sample query:

```
select col1, col2, col3, ct."_Operation", left(CDC_get_changed_columns (pub.CDC_tab1,
"_Change-Fieldmap"),10) as gcc from  pub."_Cdc-Change-Tracking" ct inner join
pub.CDC_tab1 c on ct."_Change-Sequence" = c."_Change-Sequence" where  ct."_Policy-Id"
= 'paj8RqV9kqYtFOc2AIJrDg' and    ct."_Partition-Id" = 0 and    c."_Operation"    != 3;
```

COL1	COL2	COL3	_Operation	GCC
1	t1	1	1	
2	t2	1	1	
3	t2	1	1	
4	t2	1	1	
5	t2	1	1	

Conclusion:

Drive data accuracy across the business with OpenEdge Change Data Capture. As a new component of the Advanced Enterprise Edition, OpenEdge CDC allows you to quickly identify, track and save all changes within the OpenEdge database while increasing efficiency for ETL synchronization with other data sources, repositories or data warehouses. It guarantees accurate tracking regardless of where data changes occur, with configuration flexibility to track the changes that matter most without having to make changes to the application.

For more information on how OpenEdge CDC works check out: our ondemand webinar, these videos, or test drive OpenEdge 11.7 for 60-days

Contact Us

About Progress

Progress (NASDAQ: PRGS) offers the leading platform for developing and deploying mission-critical business applications. Progress empowers enterprises and ISVs to build and deliver cognitive-first applications, that harness big data to derive business insights and competitive advantage. Progress offers leading technologies for easily building powerful user interfaces across any type of device, a reliable, scalable and secure backend platform to deploy modern applications, leading data connectivity to all sources, and award-winning predictive analytics that brings the power of machine learning to any organization. Over 1700 independent software vendors, 80,000 enterprise customers, and 2 million developers rely on Progress to power their applications.
Learn about Progress at www.progress.com or +1-800-477-6473

To learn more about the offerings available by members of the OpenEdge community, please visit and register for our forum dedicated to driving networking, sharing and learning opportunities: Progress Community
If you have questions or would like more information, please Contact Us

Worldwide Headquarters

Progress, 14 Oak Park, Bedford, MA 01730 USA
Tel: +1 781 280-4000 Fax: +1 781 280-4095
On the Web at: www.progress.com
Find us on facebook.com/progresssw twitter.com/progresssw youtube.com/progresssw
For regional international office locations and contact information, please go to www.progress.com/worldwide

Rev 2017/10 | 171010-0064

Progress®