Advanced Integration Services – Part 2

Managing Incremental Loads

Stacia Misner
blog.datainspirations.com
@StaciaMisner

pluralsight
hardcore dev and IT training

Overview

- T-SQL MERGE Statement
- Change Data Capture Tasks and Transforms
- Custom Component

T-SQL MERGE Statement

Data warehouse dimension table

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Staging table or SELECT statement

Foundational concepts for Type 1 and Type 2
slowly changing dimensions (SCDs) are covered in
my Advanced Integration Services course

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Define join between tables:

- *Business key*
- *Status indicator or date range*

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Create search condition comparing each Type 1 column in target to source column

Connect multiple conditions with AND operator

```
. . .
WHEN MATCHED
   AND NOT (Source.Name = ISNULL(target.EnglishProductName, '')
      AND source.Color = ISNULL(target.Color, ''))
THEN
   .  . .
```

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Update Type 1 columns from source

```
. . .
SET target.EnglishProductName = source.Name,
   target.Color = source.Color
   .  . .
```

Create a list for multiple columns to update

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Find records in source that are NOT in target

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Insert new records into the target

T-SQL MERGE Statement

Type 1 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.Name = ISNULL(target.EnglishProductName, ''))
THEN
   UPDATE
   SET target.EnglishProductName = source.Name
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

$ action = INSERT, UPDATE, or DELETE

To capture counts for auditing

T-SQL MERGE Statement

Type 2 SCD

```
MERGE dw.DimProduct AS target
USING (
  SELECT Name, . . .
  FROM tmp.scdProduct) AS source
ON target.ProductAlternateKey = source.ProductNumber
  AND target.Status = 'Current'
WHEN MATCHED AND NOT (Source.ListPrice = ISNULL(target.ListPrice, ''))
THEN
   UPDATE
   SET target.Status = NULL, target.EndDate = GETDATE()
WHEN NOT MATCHED BY target AND source.SellEndDate IS NULL
THEN
   INSERT (ProductAlternateKey, ProductSubcategoryKey, . . .)
   VALUES (source.ProductNumber, source.ProductSubcategoryKey, . . .)
OUTPUT $action as MergeAction, source.*
```

Expire changed existing records

To capture counts for auditing

Auditing and Type 2 Inserts

```
CREATE TABLE #DimProduct (MergeAction NVARCHAR(10),
   Name NVARCHAR(50), . . .)

INSERT INTO #DimProduct
SELECT * FROM (
   MERGE dw.DimProduct AS target . . . ) mergeOutput

INSERT INTO dw.DimProduct
SELECT Name, . . . FROM #DimProduct
   WHERE MergeAction = 'UPDATE'
```

Type 2 Inserts

Type 1 Auditing

```
SELECT
   SUM(CASE WHEN MergeAction = 'INSERT' THEN 1 ELSE 0 END)
      AS RowCountInsert,
   SUM(CASE WHEN MergeAction = 'UPDATE' THEN 1 ELSE 0 END)
      AS RowCountUpdate
FROM #DimProduct
```

Type 2 Auditing

```
SELECT
   SUM(CASE WHEN MergeAction IN ('INSERT', 'UPDATE')
      THEN 1 ELSE 0 END)
      AS RowCountInsert,
   SUM(CASE WHEN MergeAction = 'UPDATE' THEN 1 ELSE 0 END)
      AS RowCountUpdate
FROM #DimProduct
```

Error Handling

```
BEGIN TRY
BEGIN TRANSACTION
```

Place the MERGE and auditing operations in a transaction

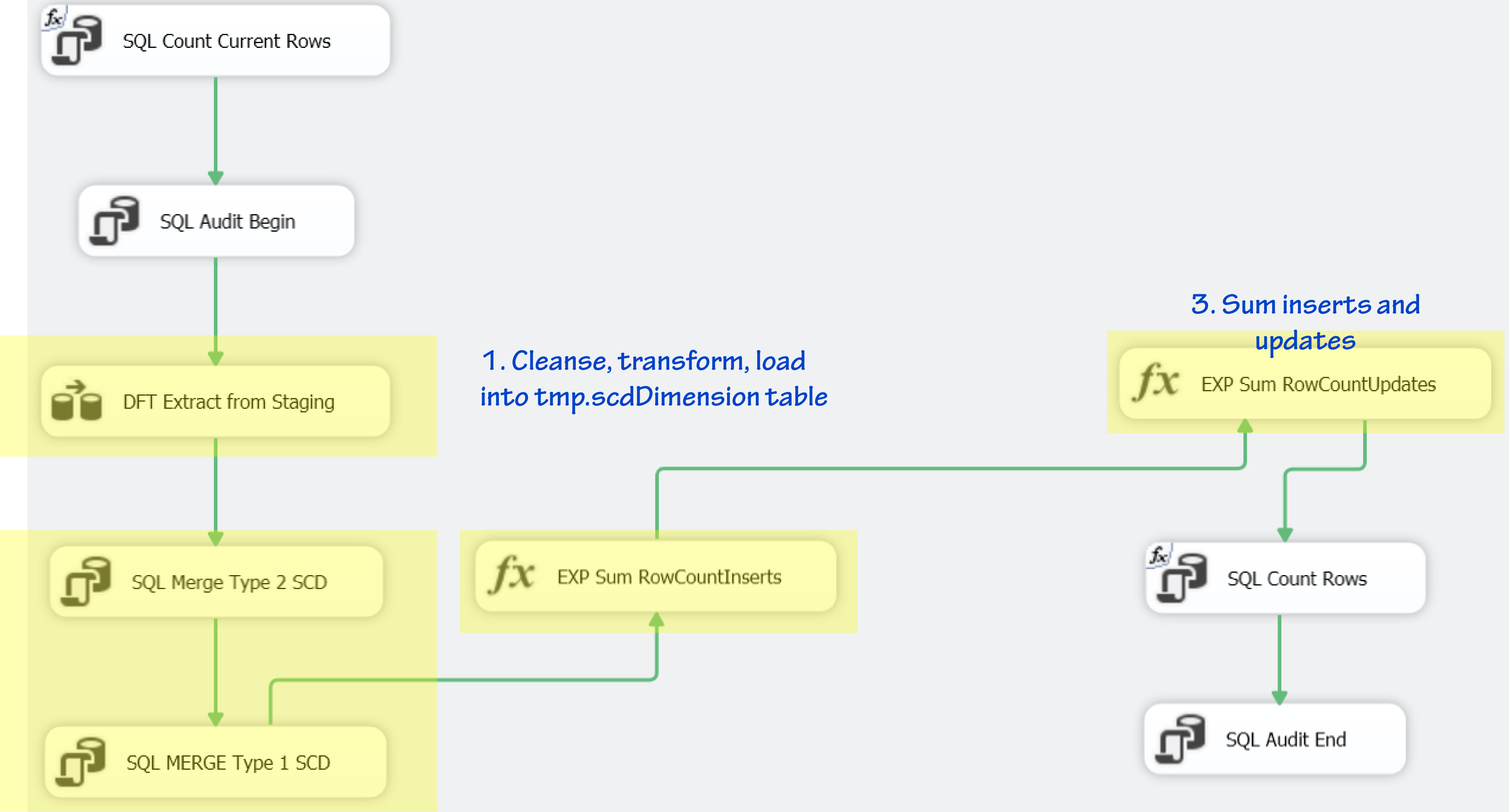
```
CREATE TABLE #DimProduct (MergeAction NVARCHAR(10),
   Name NVARCHAR(50), . . .)

INSERT INTO #DimProduct
SELECT * FROM (
   MERGE dw.DimProduct AS target . . . ) mergeOutput
SELECT
   SUM(CASE WHEN MergeAction = 'INSERT' THEN 1 ELSE 0 END)
      AS RowCountInsert,
   SUM(CASE WHEN MergeAction = 'UPDATE' THEN 1 ELSE 0 END)
      AS RowCountUpdate
FROM #DimProduct

COMMIT
END TRY
BEGIN CATCH
   . . . <Error Handling code> . . .
END CATCH
```

Use TRY/CATCH block to perform ROLLBACK and capture error for logging

Dimension Load Pattern with MERGE



1. Cleanse, transform, load into tmp.scdDimension table

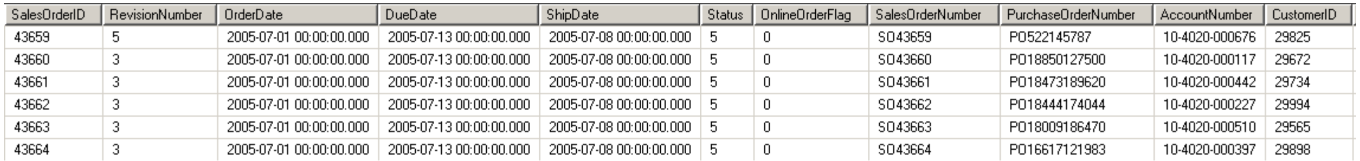
2. Execute MERGE statements for Type 2 then for Type 1
(or just one of those)

3. Sum inserts and updates

Change Data Capture

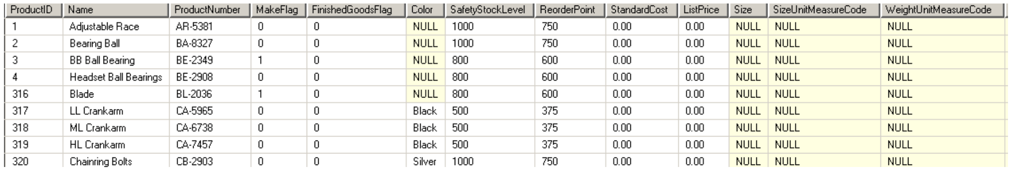
Enterprise Edition!!

Fact Table

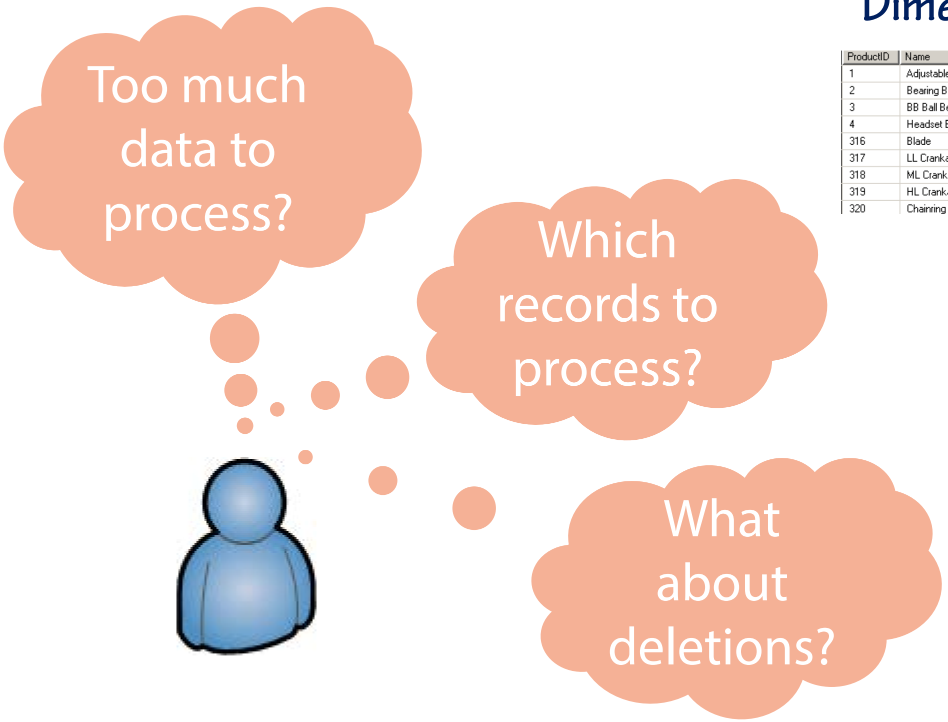


SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID
43659	5	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	29825
43660	3	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43660	PO18850127500	10-4020-000117	29672
43661	3	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43661	PO18473189620	10-4020-000442	29734
43662	3	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43662	PO18444174044	10-4020-000227	29994
43663	3	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43663	PO18009186470	10-4020-000510	29565
43664	3	2005-07-01 00:00:00.000	2005-07-13 00:00:00.000	2005-07-08 00:00:00.000	5	0	SO43664	PO16617121983	10-4020-000397	29898

Dimension Table



ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice	Size	SizeUnitMeasureCode	WeightUnitMeasureCode
1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0.00	0.00	NULL	NULL	NULL
2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0.00	0.00	NULL	NULL	NULL
3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0.00	0.00	NULL	NULL	NULL
4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0.00	0.00	NULL	NULL	NULL
316	Blade	BL-2036	1	0	NULL	800	600	0.00	0.00	NULL	NULL	NULL
317	LL Crankarm	CA-5965	0	0	Black	500	375	0.00	0.00	NULL	NULL	NULL
318	ML Crankarm	CA-6738	0	0	Black	500	375	0.00	0.00	NULL	NULL	NULL
319	HL Crankarm	CA-7457	0	0	Black	500	375	0.00	0.00	NULL	NULL	NULL
320	Chainring Bolts	CB-2903	0	0	Silver	1000	750	0.00	0.00	NULL	NULL	NULL



Too much data to process?

Which records to process?

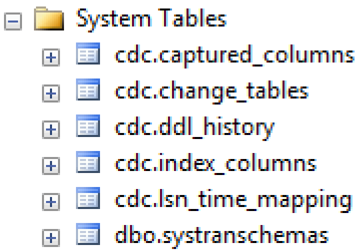
What about deletions?

CDC Setup

Step 1. Enable CDC for database

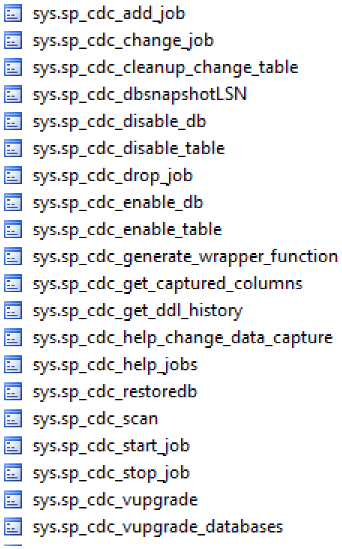
```
USE AdventureWorks2012
GO
EXEC sp_changedbowner 'sa'
GO
EXEC sys.sp_cdc_enable_db
GO
```

CDC Tables



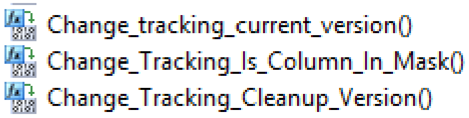
System Tables
- cdc.captured_columns
- cdc.change_tables
- cdc.ddl_history
- cdc.index_columns
- cdc.lsn_time_mapping
- dbo.systranschemas

CDC Stored Procedures



- sys.sp_cdc_add_job
- sys.sp_cdc_change_job
- sys.sp_cdc_cleanup_change_table
- sys.sp_cdc_dbsnapshotLSN
- sys.sp_cdc_disable_db
- sys.sp_cdc_disable_table
- sys.sp_cdc_drop_job
- sys.sp_cdc_enable_db
- sys.sp_cdc_enable_table
- sys.sp_cdc_generate_wrapper_function
- sys.sp_cdc_get_captured_columns
- sys.sp_cdc_get_ddl_history
- sys.sp_cdc_help_change_data_capture
- sys.sp_cdc_help_jobs
- sys.sp_cdc_restoredb
- sys.sp_cdc_scan
- sys.sp_cdc_start_job
- sys.sp_cdc_stop_job
- sys.sp_cdc_vupgrade
- sys.sp_cdc_vupgrade_databases

CDC Functions



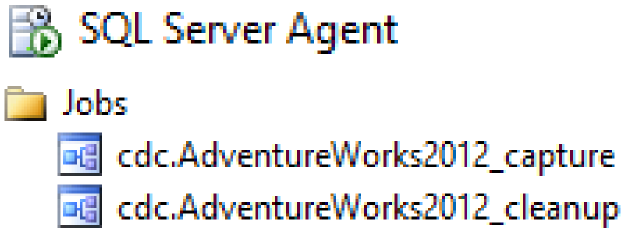
- Change_tracking_current_version()
- Change_Tracking_Is_Column_In_Mask()
- Change_Tracking_Cleanup_Version()

CDC Setup

Step 2. Enable CDC for table(s)

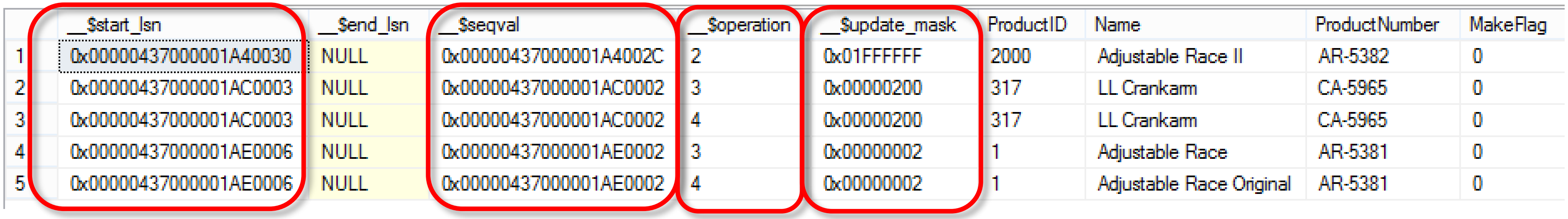
```
USE AdventureWorks2012
GO
EXEC sys.sp_cdc_enable_table
   @source_schema = N'Production'
   ,@source_name = N'Product'
   ,@role_name = N'cdc_Admin'
   ,@capture_instance = N'Production_Product'
   ,@supports_net_changes = 1
```

cdc.Production_Product_CT



SQL Server Agent
- Jobs
  - cdc.AdventureWorks2012_capture
  - cdc.AdventureWorks2012_cleanup

Anatomy of a CDC Table



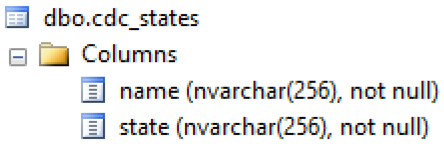
	__$start_lsn	__$end_lsn	__$seqval	__$operation	__$update_mask	ProductID	Name	ProductNumber	MakeFlag
1	0x00000437000001A40030	NULL	0x00000437000001A4002C	2	0x01FFFFFF	2000	Adjustable Race II	AR-5382	0
2	0x00000437000001AC0003	NULL	0x00000437000001AC0002	3	0x00000200	317	LL Crankarm	CA-5965	0
3	0x00000437000001AC0003	NULL	0x00000437000001AC0002	4	0x00000200	317	LL Crankarm	CA-5965	0
4	0x00000437000001AE0006	NULL	0x00000437000001AE0002	3	0x00000002	1	Adjustable Race	AR-5381	0
5	0x00000437000001AE0006	NULL	0x00000437000001AE0002	4	0x00000002	1	Adjustable Race Original	AR-5381	0

- **__$start_lsn and __$seqval**
  - Link record to a transaction
  - Specify order of operations
- **__$operation**
  - 1 = delete
  - 2 = insert
  - 3 = update (record data before change)
  - 4 = update (record data after change)
  - 5 = merge
- **__$update_mask**
  - Identify which columns changed
  - Use with Sys.fn_cdc_has_column_changed

CDC in Integration Services

Table Structures

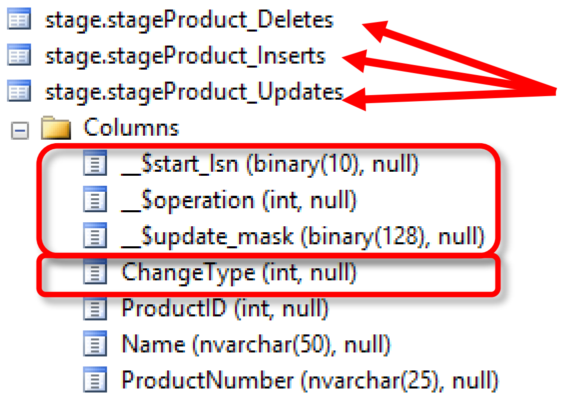
Source Database



```
dbo.cdc_states
⊟ Columns
    name (nvarchar(256), not null)
    state (nvarchar(256), not null)
```

SSIS manages current state of CDC processing here

Staging Database



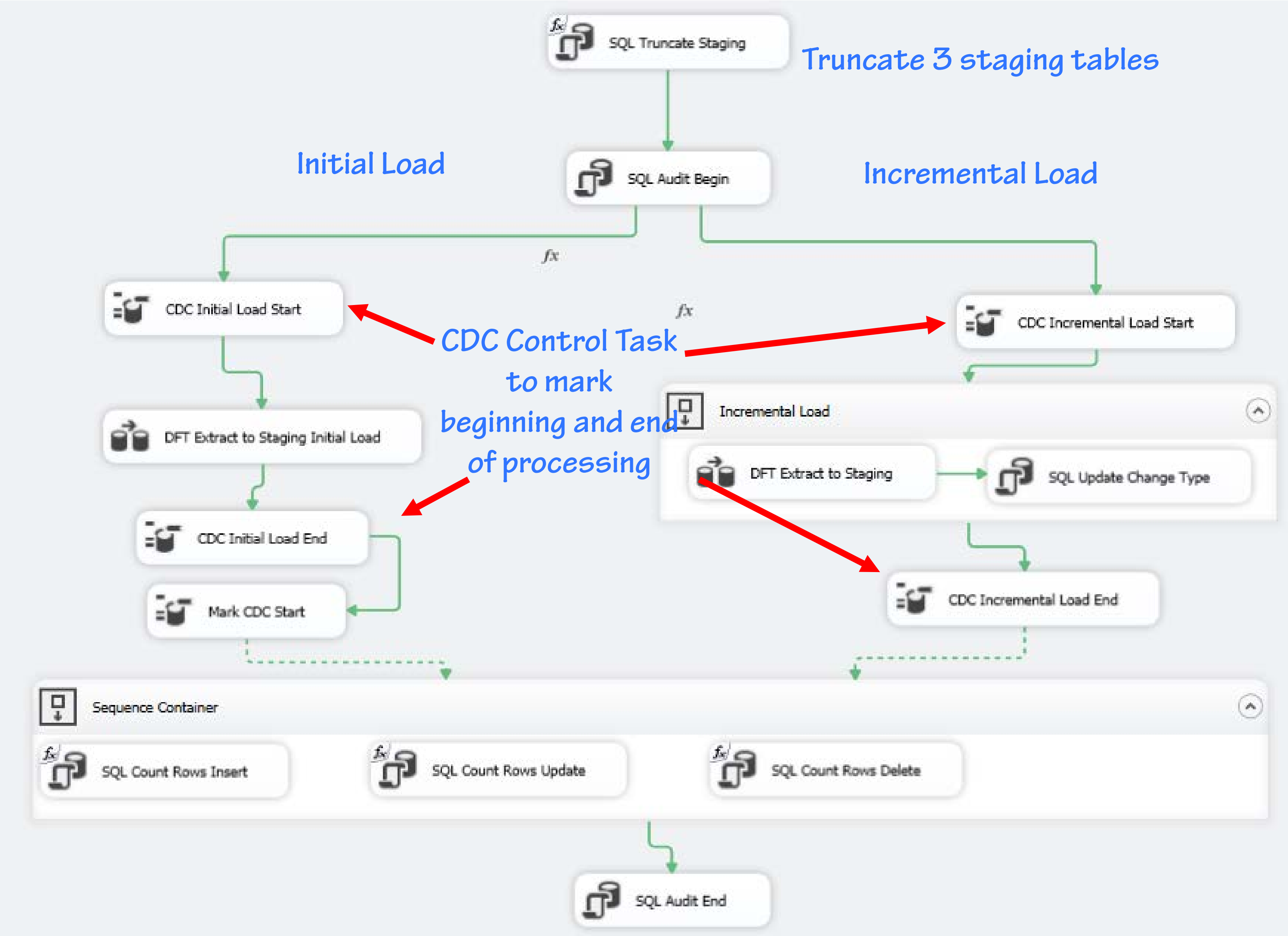
```
stage.stageProduct_Deletes
stage.stageProduct_Inserts
stage.stageProduct_Updates
⊟ Columns
    __$start_lsn (binary(10), null)
    __$operation (int, null)
    __$update_mask (binary(128), null)
    ChangeType (int, null)
    ProductID (int, null)
    Name (nvarchar(50), null)
    ProductNumber (nvarchar(25), null)
```

One staging table per type of change with source AND change columns

EXCEPT…
Updates table includes ChangeType when both Type 1 and Type2 processing required

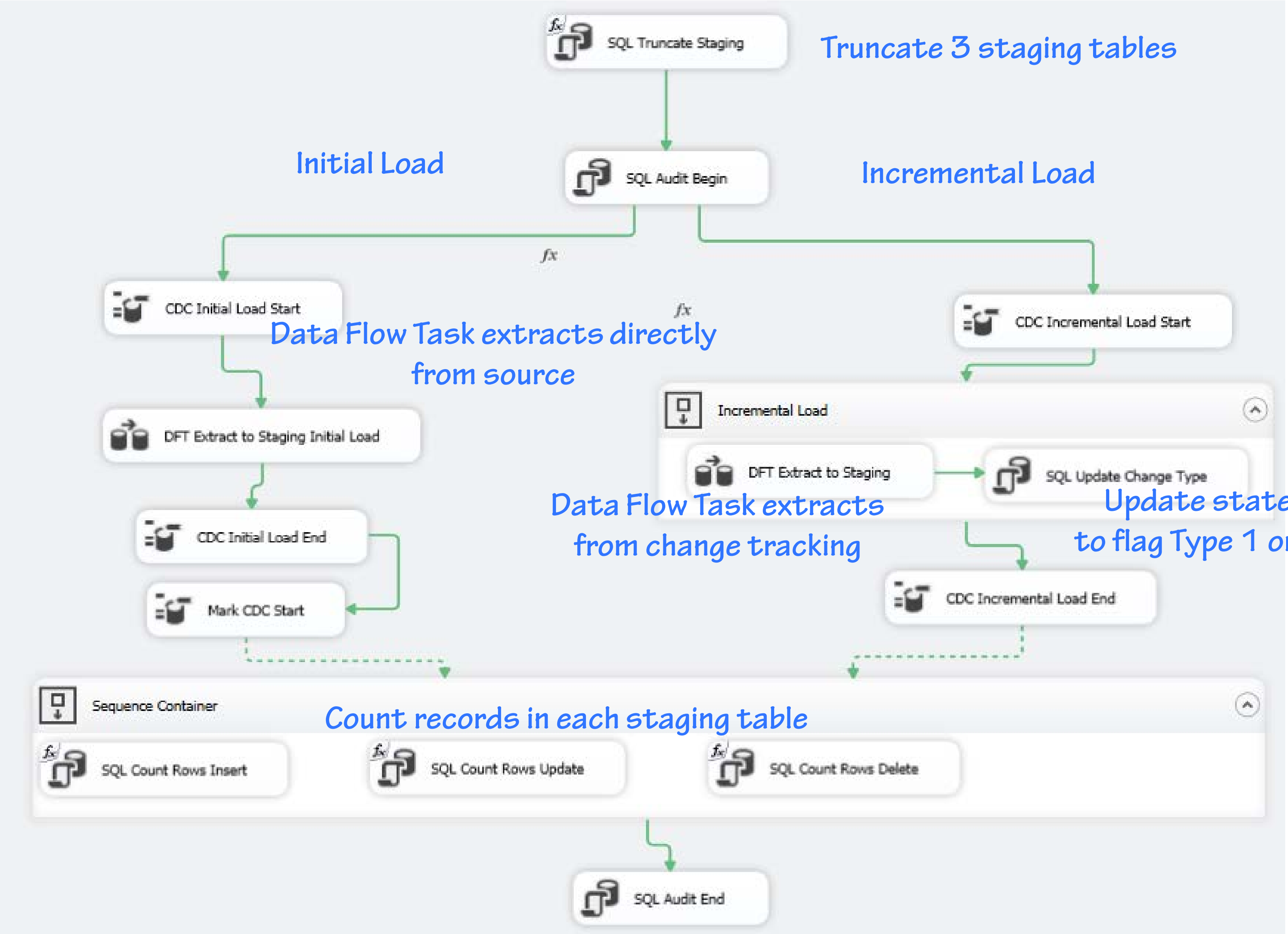
CDC in Integration Services

Control Flow - Extraction



CDC in Integration Services

Control Flow - Extraction



Truncate 3 staging tables

Initial Load

Incremental Load

Data Flow Task extracts directly from source

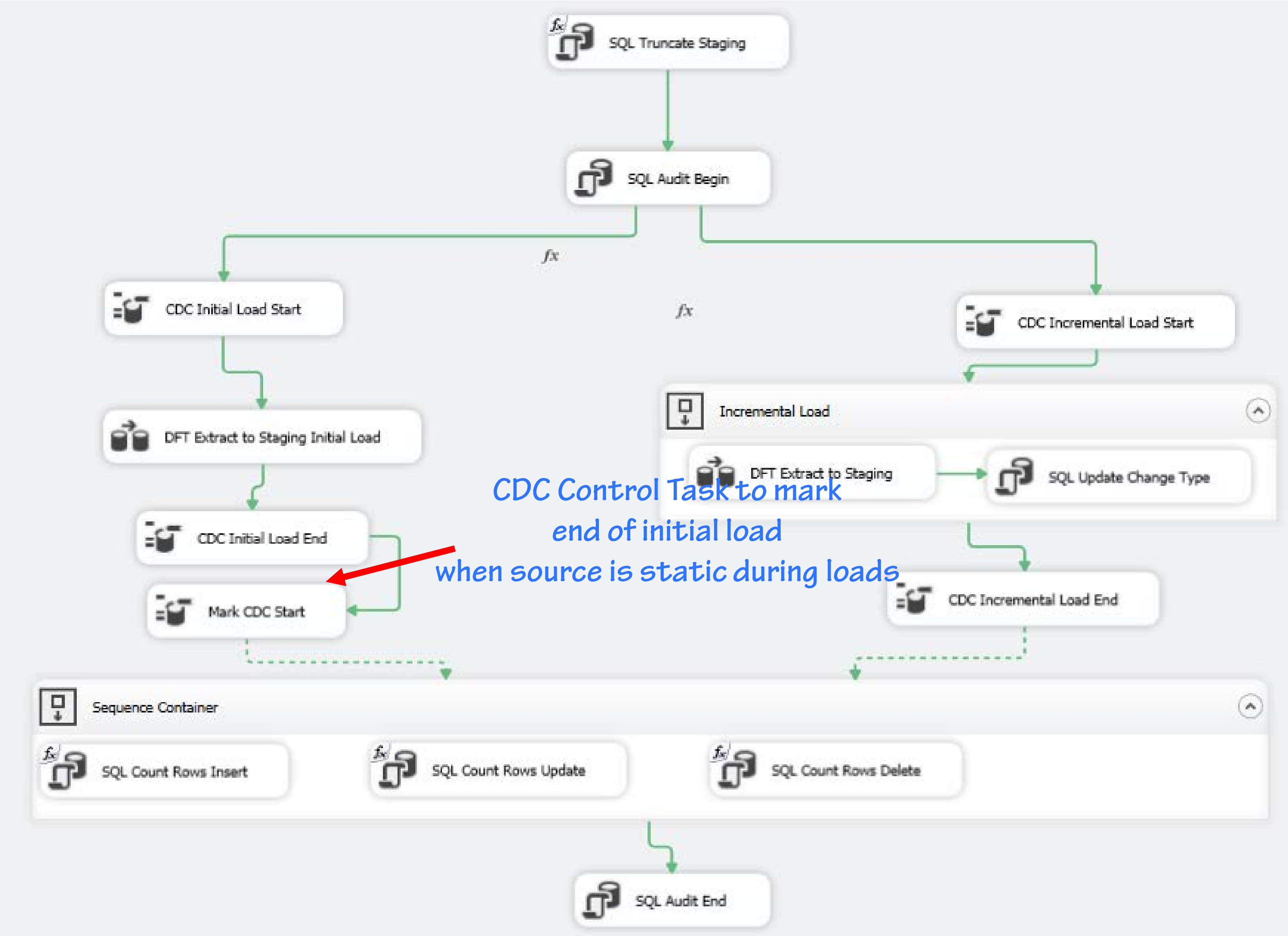
Data Flow Task extracts from change tracking

Update statement to flag Type 1 or Type 2

Count records in each staging table

CDC in Integration Services

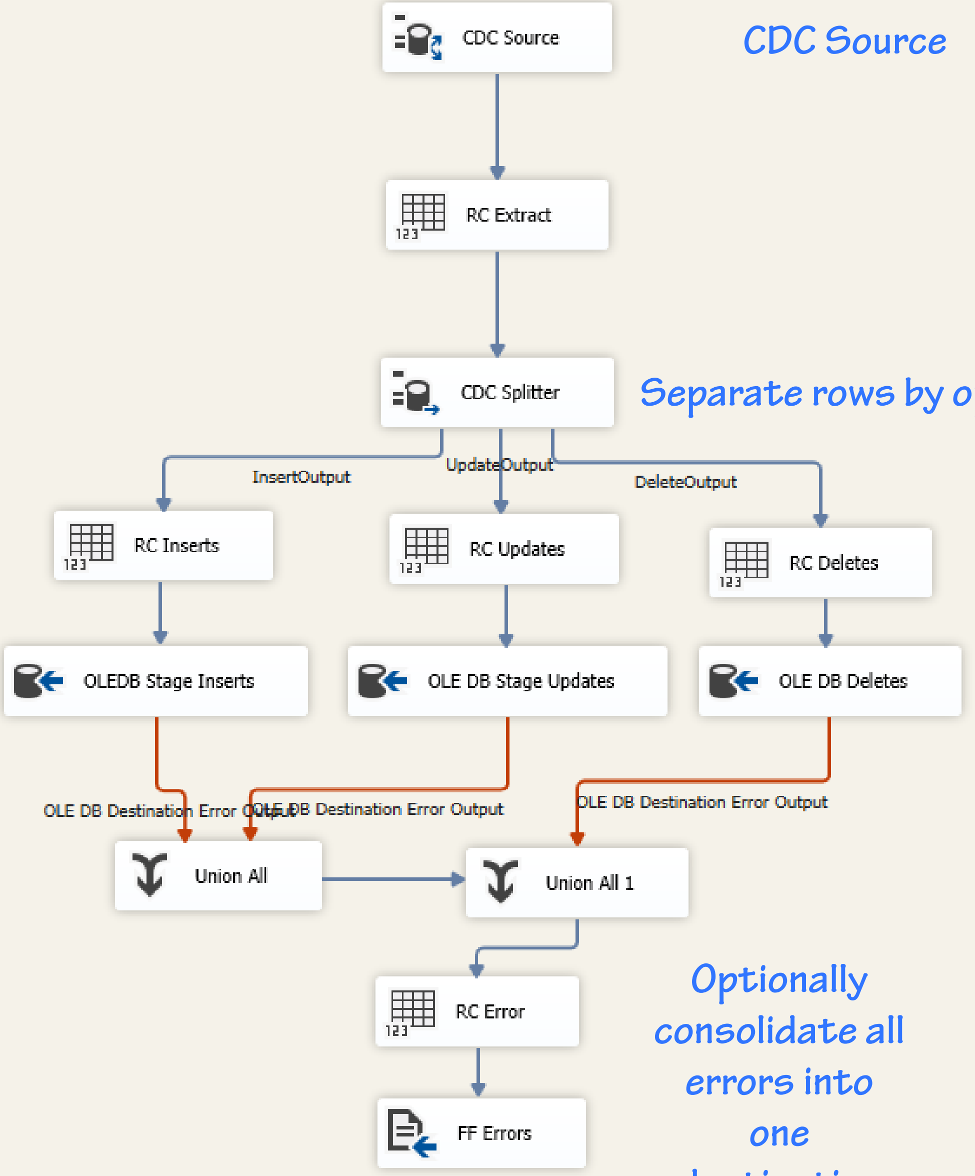
Control Flow - Extraction



CDC Control Task to mark
end of initial load
when source is static during loads

CDC in Integration Services

Data Flow Task – Extraction Incremental Load Only



CDC Source

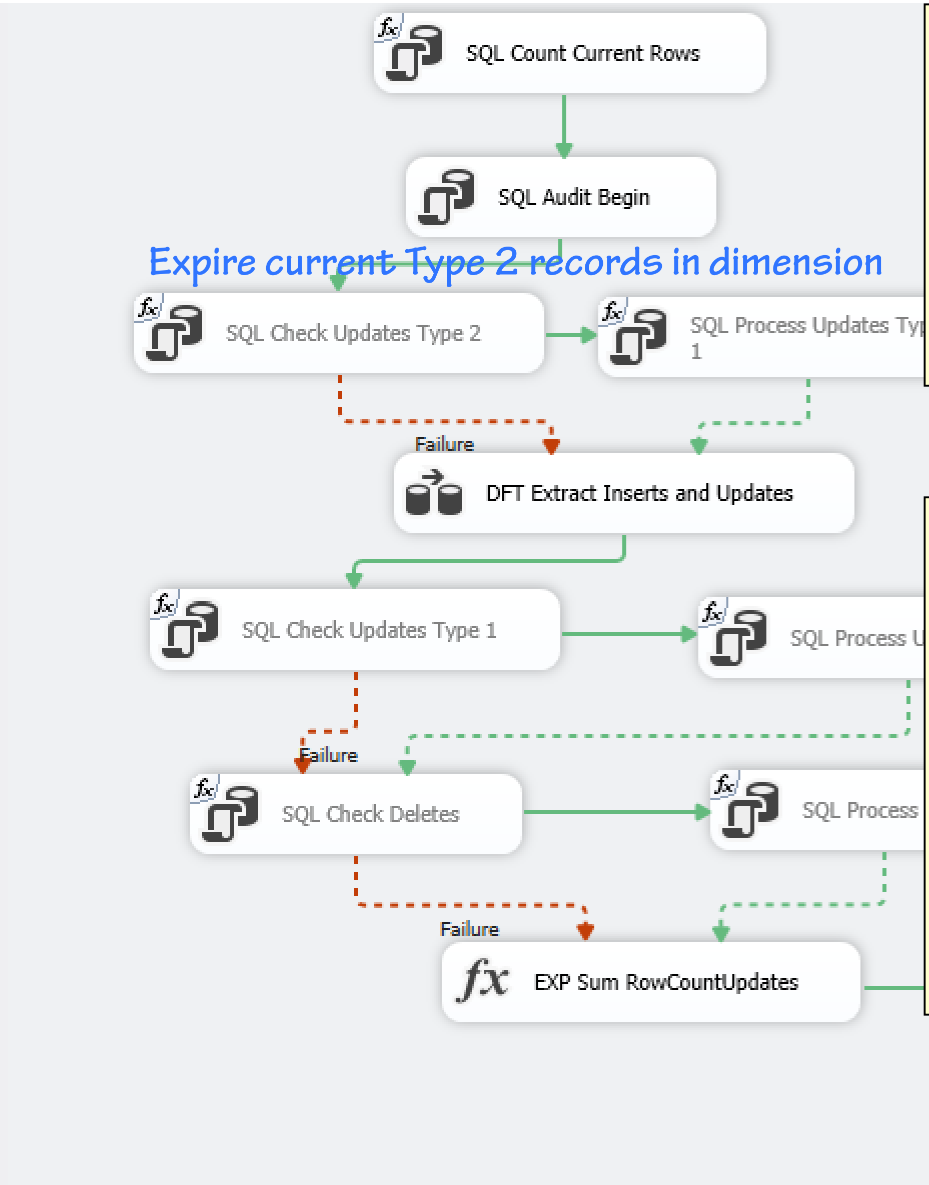
Separate rows by operation type

Send data to applicable staging table

Optionally consolidate all errors into one

CDC in Integration Services

Control Flow – Transform and Load



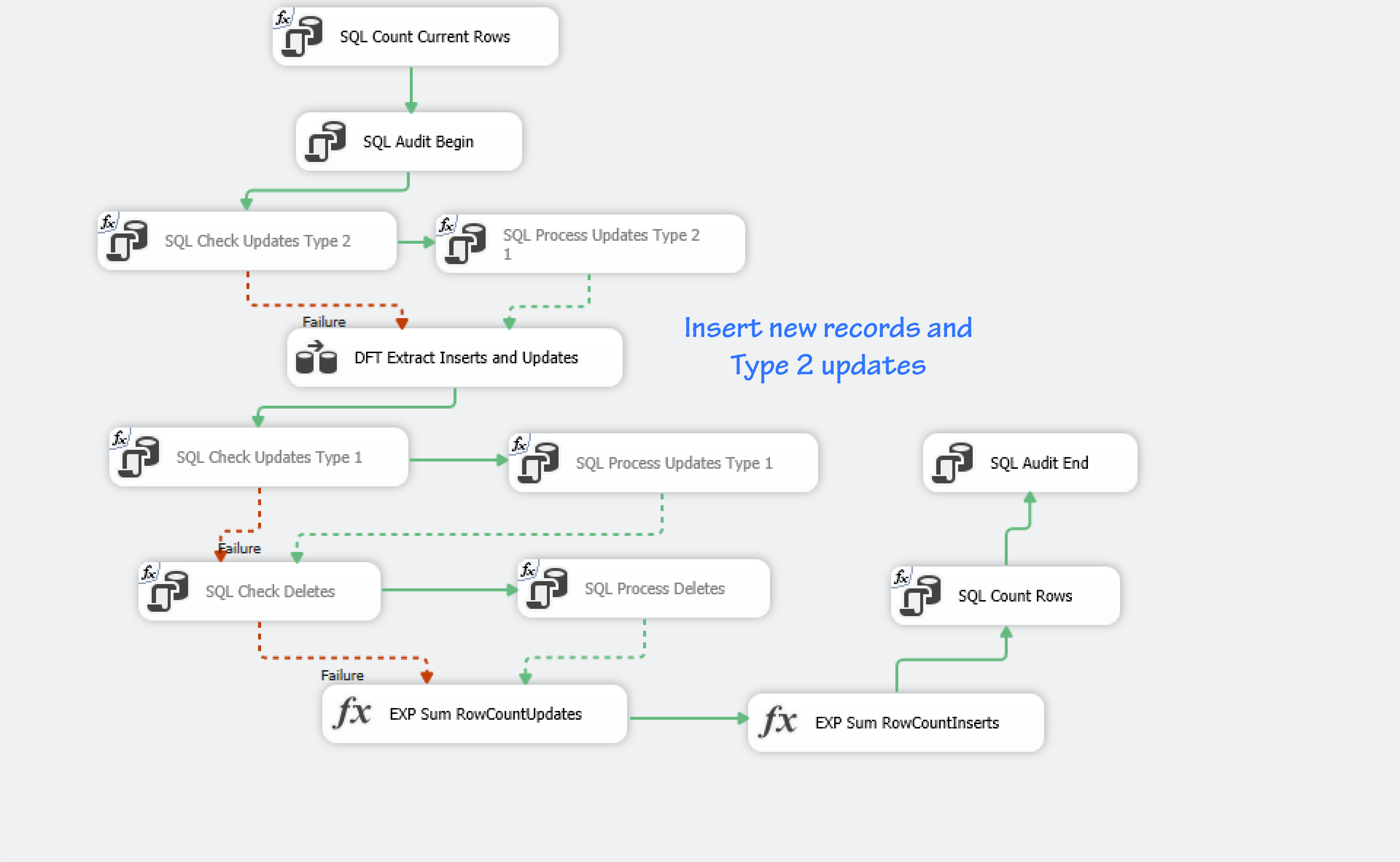
Expire current Type 2 records in dimension

```
SELECT COUNT(*)
FROM
[AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Updates] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND stg.ChangeType = 2
  AND dim.Status = 'Current'
```

```
UPDATE dim
SET
  dim.Status = NULL,
  dim.EndDate = GETDATE()
FROM
  [AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Updates] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND stg.ChangeType = 2
  AND dim.Status = 'Current'
```

CDC in Integration Services

Control Flow – Transform and Load



SQL Count Current Rows

SQL Audit Begin

SQL Check Updates Type 2

SQL Process Updates Type 2 1

Failure

DFT Extract Inserts and Updates

SQL Check Updates Type 1

SQL Process Updates Type 1

SQL Audit End

Failure

SQL Check Deletes

SQL Process Deletes

SQL Count Rows

Failure

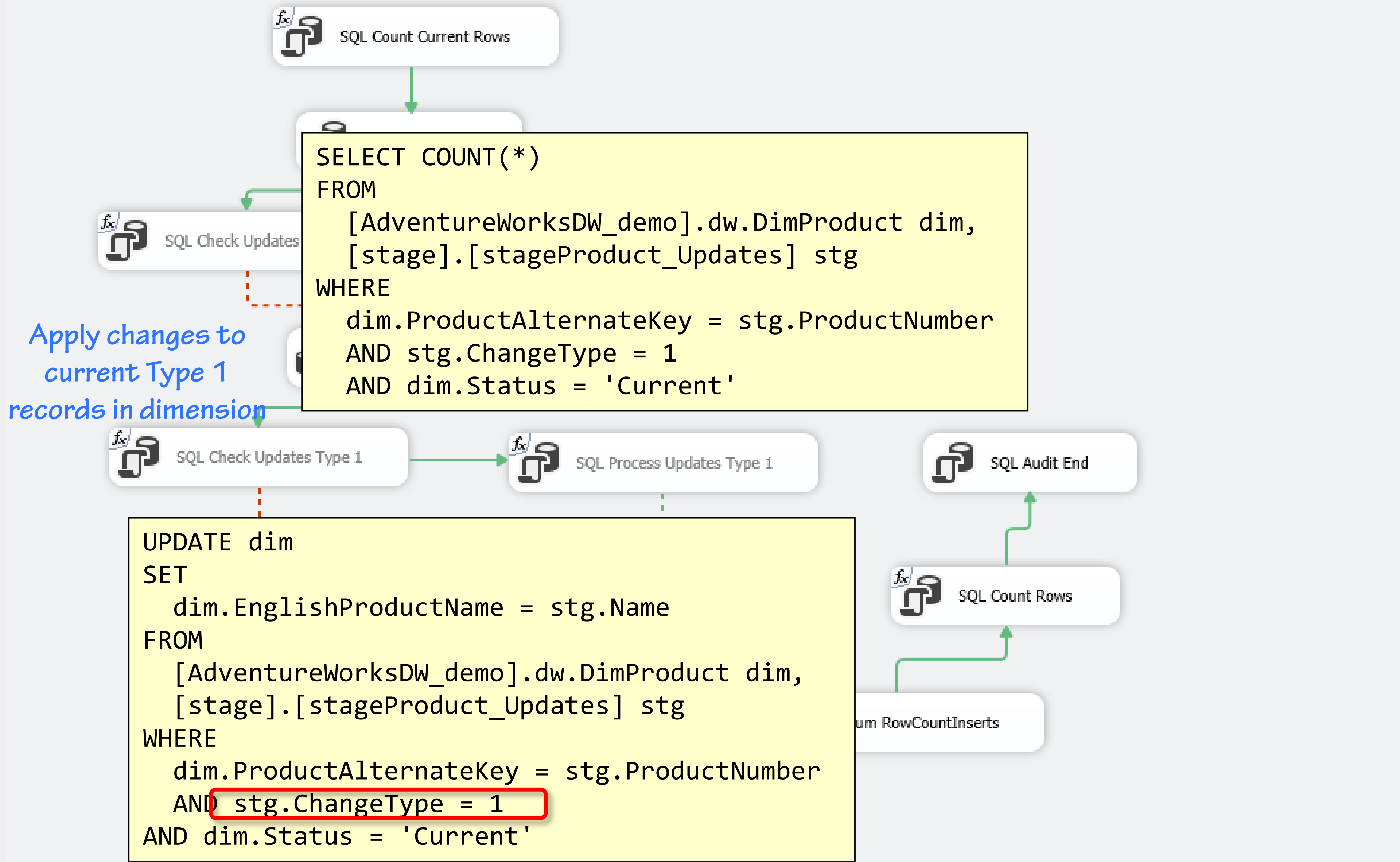
EXP Sum RowCountUpdates

EXP Sum RowCountInserts

Insert new records and Type 2 updates

CDC in Integration Services

Control Flow – Transform and Load



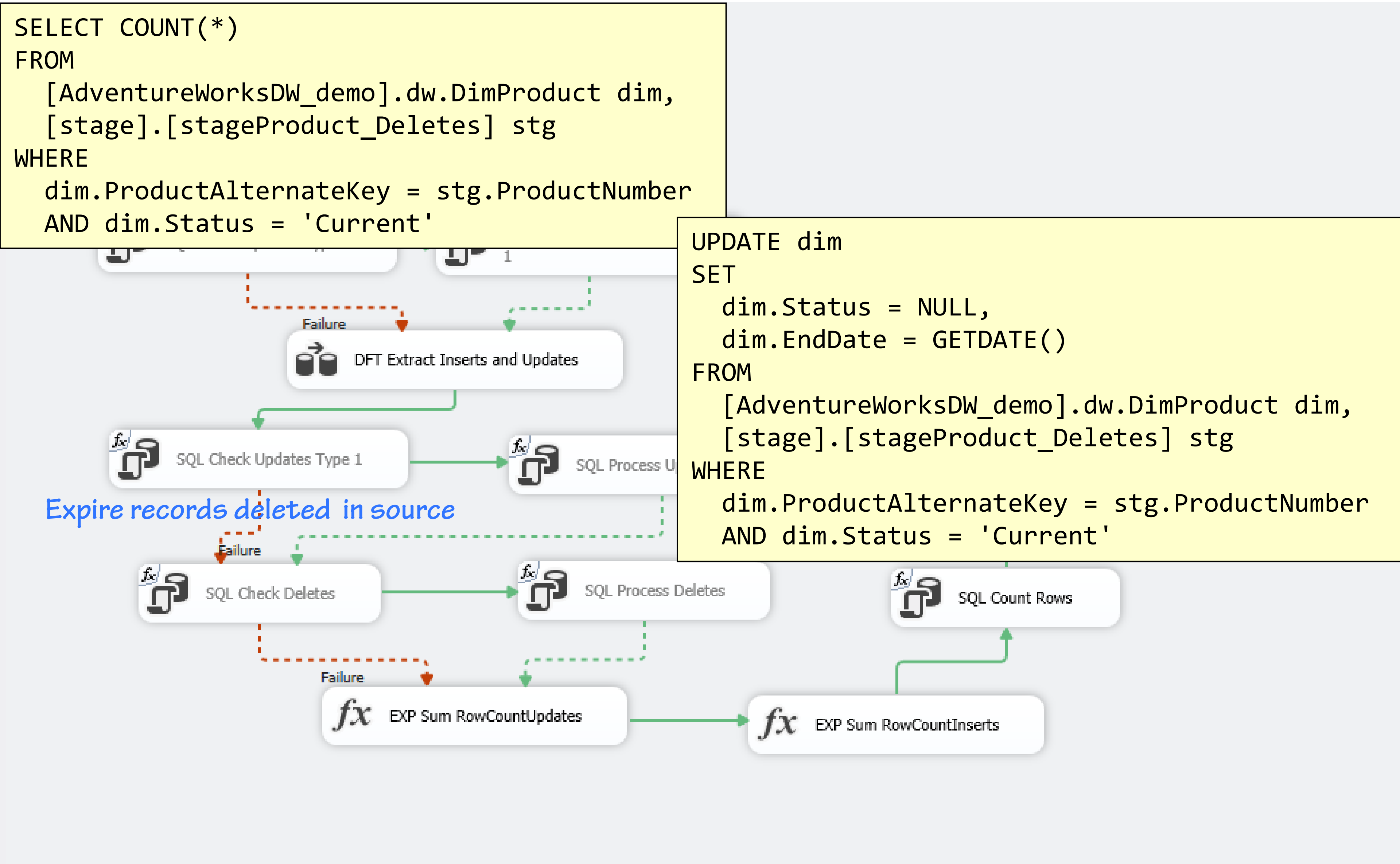
```
SELECT COUNT(*)
FROM
  [AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Updates] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND stg.ChangeType = 1
  AND dim.Status = 'Current'
```

Apply changes to current Type 1 records in dimension

```
UPDATE dim
SET
  dim.EnglishProductName = stg.Name
FROM
  [AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Updates] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND stg.ChangeType = 1
AND dim.Status = 'Current'
```

CDC in Integration Services

Control Flow – Transform and Load



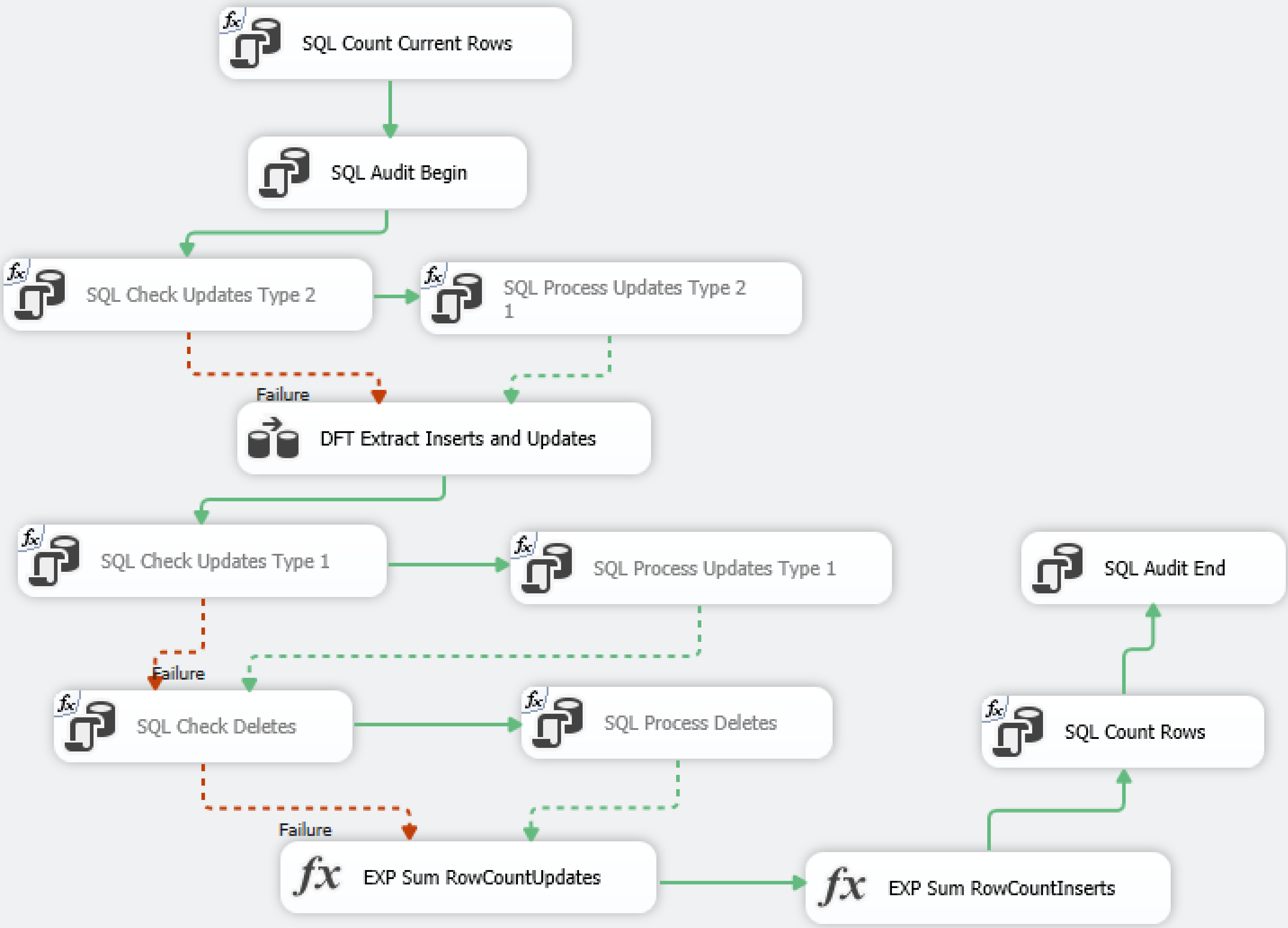
```
SELECT COUNT(*)
FROM
  [AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Deletes] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND dim.Status = 'Current'
```

```
UPDATE dim
SET
  dim.Status = NULL,
  dim.EndDate = GETDATE()
FROM
  [AdventureWorksDW_demo].dw.DimProduct dim,
  [stage].[stageProduct_Deletes] stg
WHERE
  dim.ProductAlternateKey = stg.ProductNumber
  AND dim.Status = 'Current'
```

Expire records deleted in source

CDC in Integration Services

Control Flow – Transform and Load

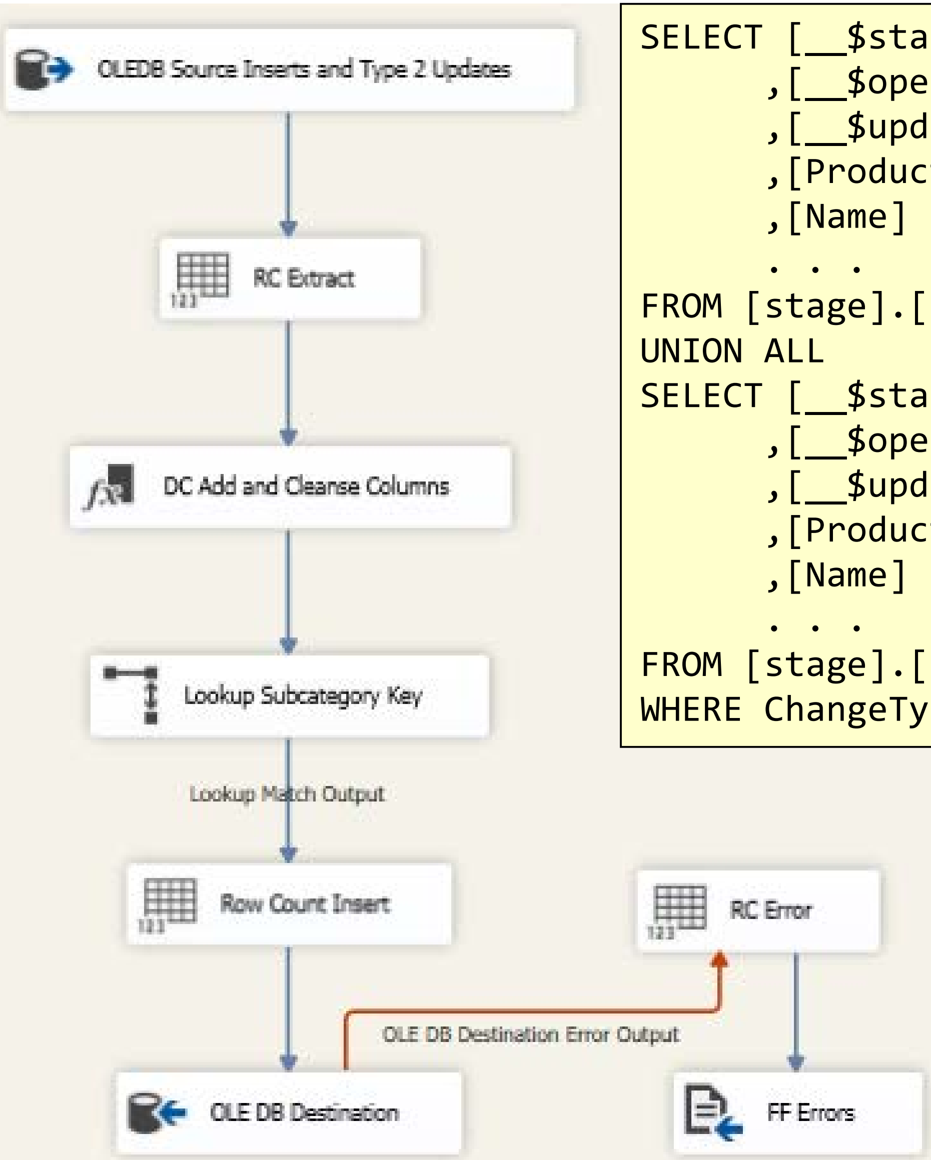


Count total updates and inserts and records in the dimension after all processing is complete

CDC in Integration Services

Data Flow – Transform and Load

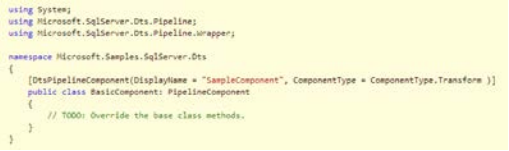
Union new records and Type 2 changes



```
SELECT [__$start_lsn]
      ,[__$operation]
      ,[__$update_mask]
      ,[ProductID]
      ,[Name]
      . . .
FROM [stage].[stageProduct_Inserts]
UNION ALL
SELECT [__$start_lsn]
      ,[__$operation]
      ,[__$update_mask]
      ,[ProductID]
      ,[Name]
      . . .
FROM [stage].[stageProduct_Updates]
WHERE ChangeType = 2
```

Custom Component

- **Custom development of transforms**
  - Use to satisfy specific business requirements
  - Refer to “Developing a Custom Data Flow Component” (http://tinyurl.com/kjz99w7)



```
using System;
using Microsoft.SqlServer.Dts.Pipeline;
using Microsoft.SqlServer.Dts.Pipeline.Wrapper;

namespace Microsoft.Samples.SqlServer.Dts
{
    [DtsPipelineComponent(DisplayName = "SampleComponent", ComponentType = ComponentType.Transform )]
    public class BasicComponent: PipelineComponent
    {
        // TODO: Override the base class methods.
    }
}
```

- **Third-party extension - Dimension Merge SCD Component**
  - Available for sale from Pragmatic Works
  - Product information and free trial download at http://tinyurl.com/kw5gujo

Summary

- **T-SQL Merge**
  - Bulk processing of inserts and updates (Type 1 & Type 2)
  - SQL Server 2008 or higher
- **Change Data Capture Components**
  - Minimal source impact; process inserts, updates (Type 1 & Type 2), deletes
  - SQL Server 2012 or higher
- **Custom Components**
  - Third-party management of Type 1 & Type 2 changes

Resources

- **Matt Masson: CDC in SSIS for SQL Server 2012**
  - http://tinyurl.com/m5gvtbg
- **SSIS Dimension Merge SCD Component**
  - http://tinyurl.com/ou5ne79
- **OUTPUT Clause for use with Merge**
  - http://tinyurl.com/ao3jma