# DataStage Configuration Steps

Data Migration

From Netezza to Azure Synapse Analytics

Microsoft

## Credits

**Prepared by**
Data Migration Jumpstart Engineering Team
Advanced Solutions Delivery – Data & AI CTO

askdmjfordmtools@microsoft.com
dmjarchitects@microsoft.com

Revision 2
3/26/2020

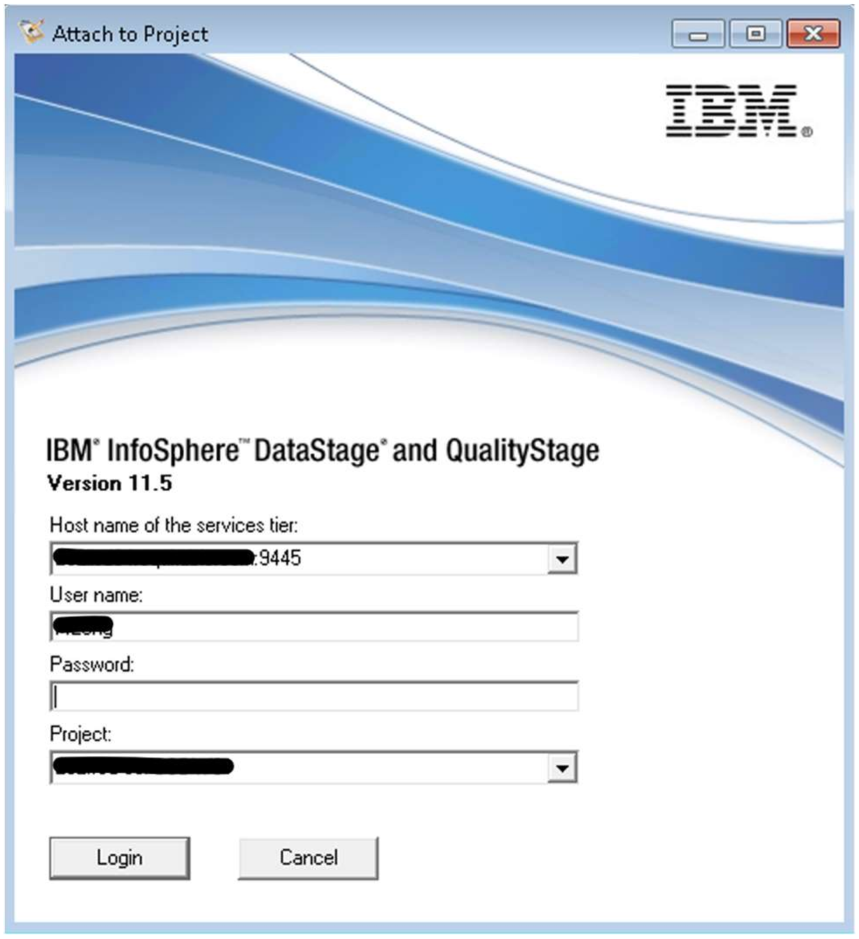
## Disclaimer

## Topics

## Utilizing DataStage

- A client DataStage application is installed on workstation/laptop (Designer)
- Clicking on the Designer icon opens the DataStage application which requires authentication to the server with name, password and project name
- Selection of a job is made from a tree structure on the left showing the jobs and sequences in hierarchical fashion

# IBM InfoSphere DataStage Logon Screen



Attach to Project

IBM® InfoSphere™ DataStage® and QualityStage

**Version 11.5**

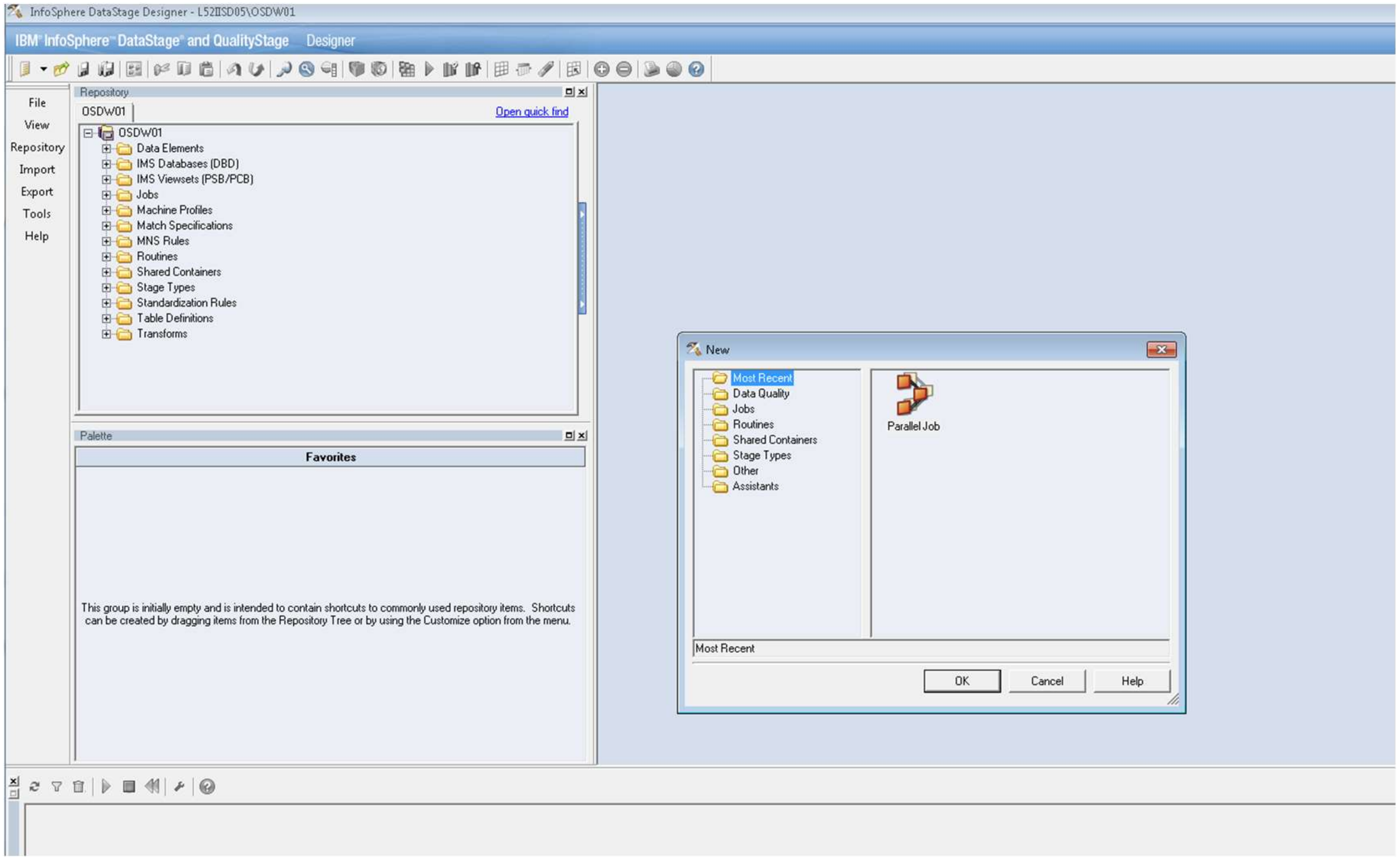
Host name of the services tier: [redacted]:9445

User name: [redacted]

Password:

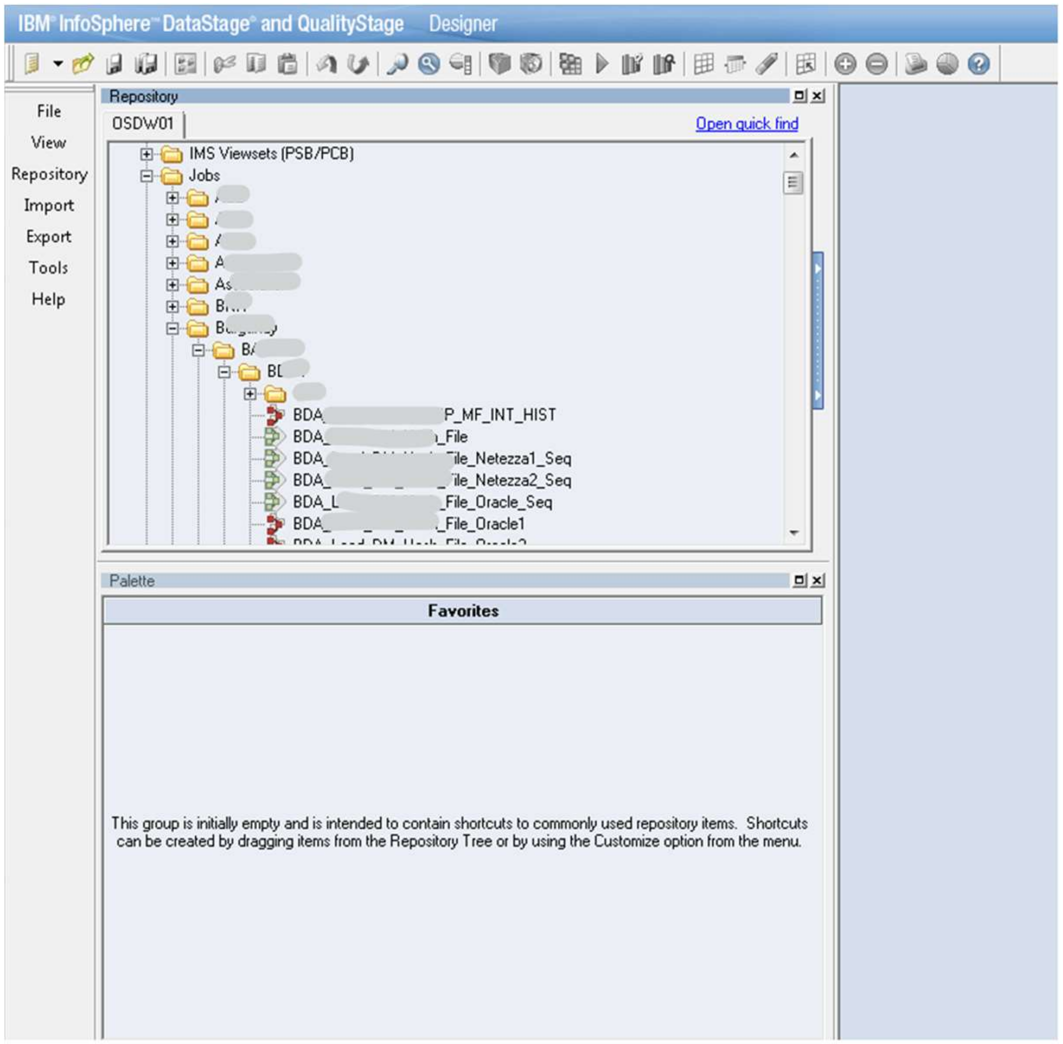
Project: [redacted]

# DataStage Initial Screen



This group is initially empty and is intended to contain shortcuts to commonly used repository items. Shortcuts can be created by dragging items from the Repository Tree or by using the Customize option from the menu.

# Select Job from the Project

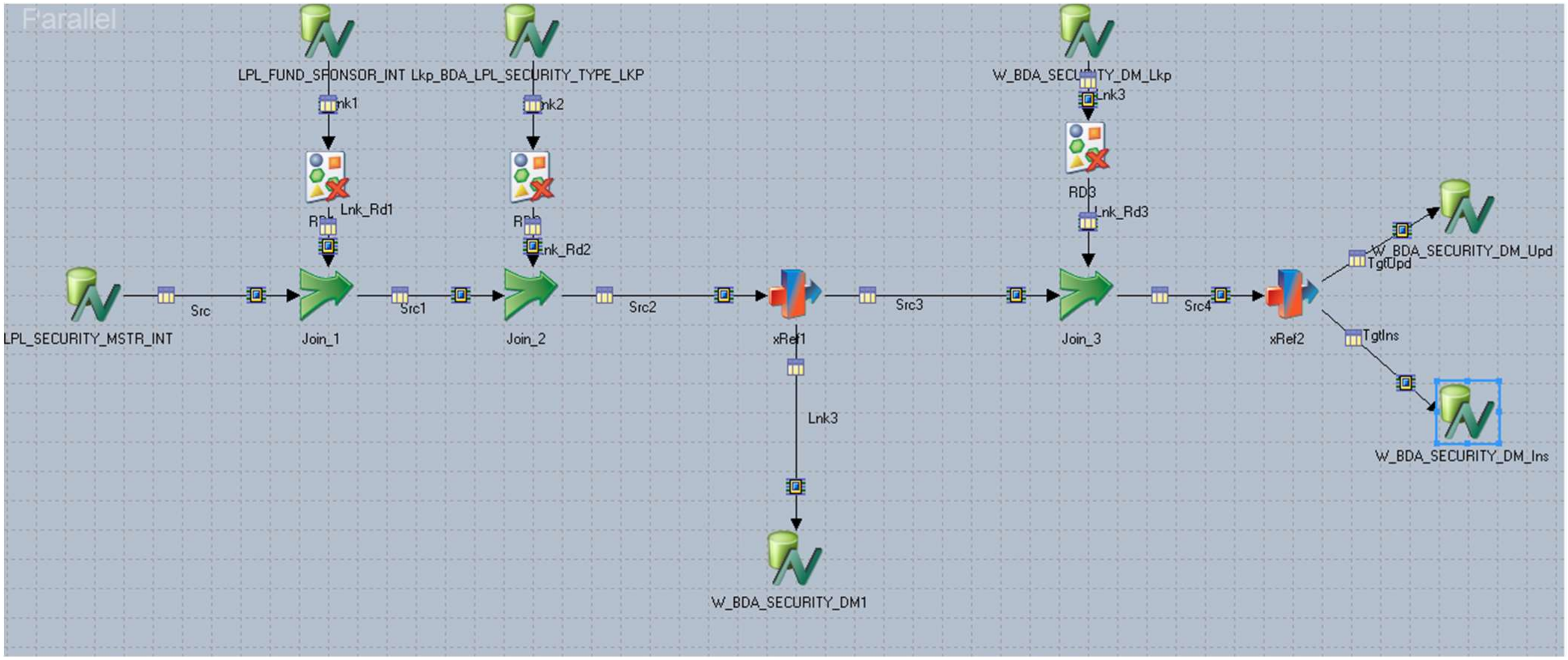


This group is initially empty and is intended to contain shortcuts to commonly used repository items. Shortcuts can be created by dragging items from the Repository Tree or by using the Customize option from the menu.

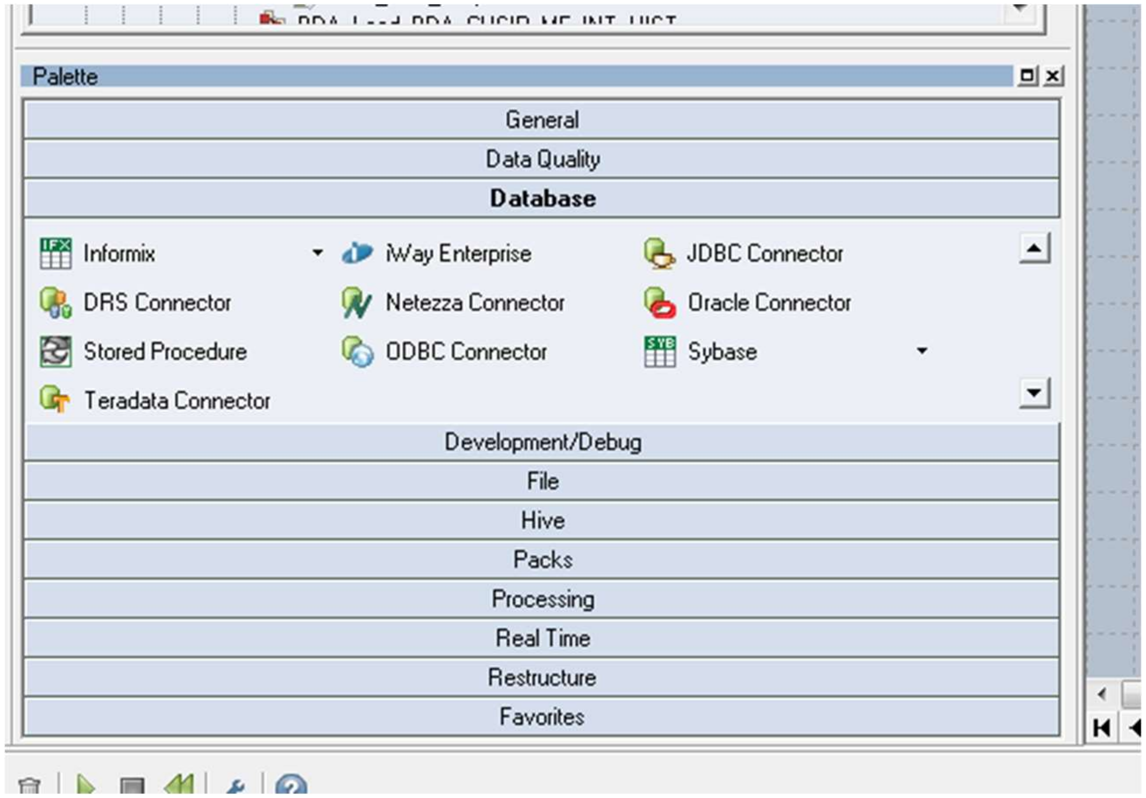
## Working with the DataStage palette

- On the left of the screen there are selections for the *palette*.
- The *Database* category allows the user to select the *connectivity* type for this job.
- The *Processing* category allows the user to select the type of processing to occur, i.e. *transformer*
- The *File* category allows the user to select a type of target or source file (as opposed to a database) and requires no connectivity information

# Sample Netezza Job – Update and Insert



## Selection for Connectivity of Available Sources/Targets



## Singleton and Bulk Load Inserts

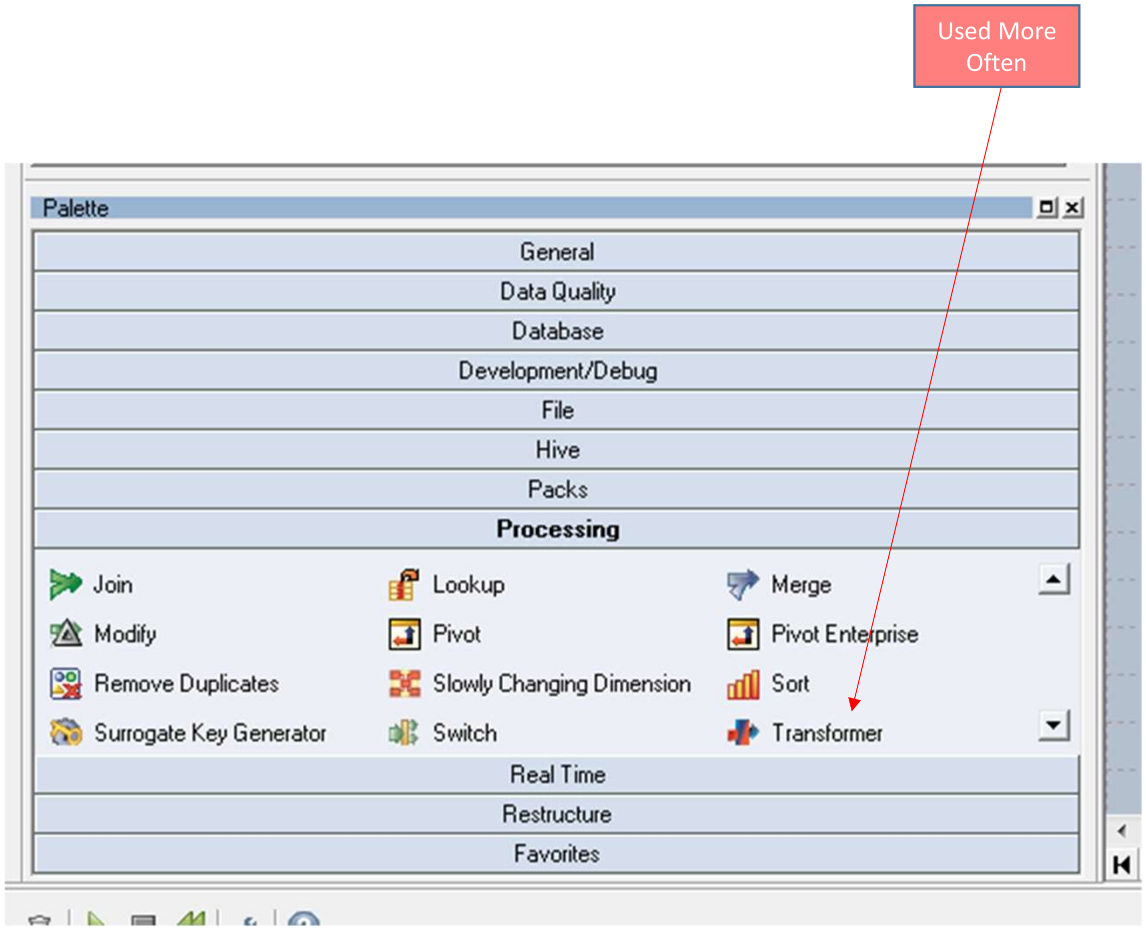
- Selection of **ODBC** Connector will only allow singleton transactions
- Selection of **DRS** Connector will allow for selection of Bulk Insert option
- All stored procedures run on Azure Synapse Analytics must be executed via the Stored Procedure Database option from this selection screen

## DRS Connection

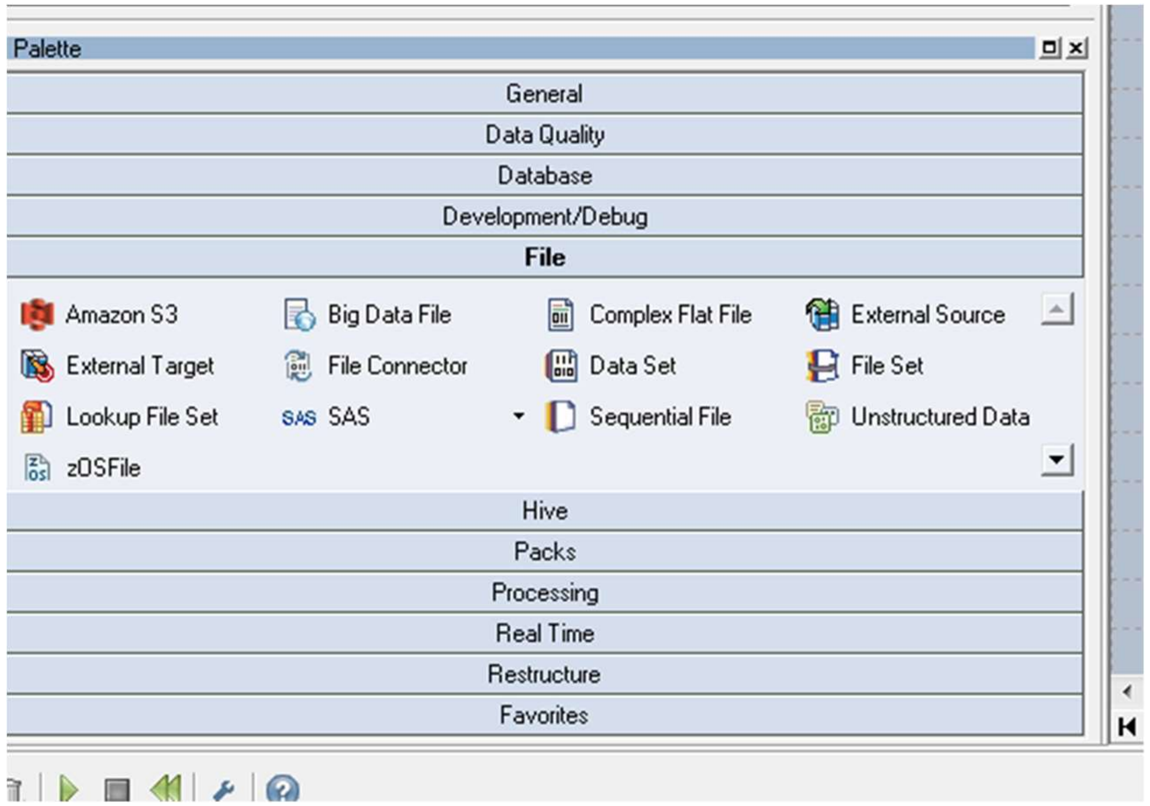
- Use the DRS Connector stage to access relational database management systems by using the native interfaces that are available for the corresponding databases.
- The choice of the database type is not determined when the stage is placed on the job canvas but is instead specified when the stage is configured.
- The DRS Connector stage supports IBM® DB2®, Oracle, and ODBC data sources. For other database types, you can configure the DRS Connector stage to use the ODBC database type and access the databases through the ODBC drivers that are included with InfoSphere Information Server.
- After the DRS Connector stage is placed on the job canvas, it needs to be configured to perform the operation intended by the job design.
- When a new DRS Connector stage is added to the server or parallel canvas, you must specify the connection information for the database that the stage connects to at run time.

- Information Links
  - DRS Connection Overview
  - DRS Configuration
  - Settings for the DRS Connector stage
  - Defining a DRS Connector stage connection to the database

# Selection of Choices for Processing



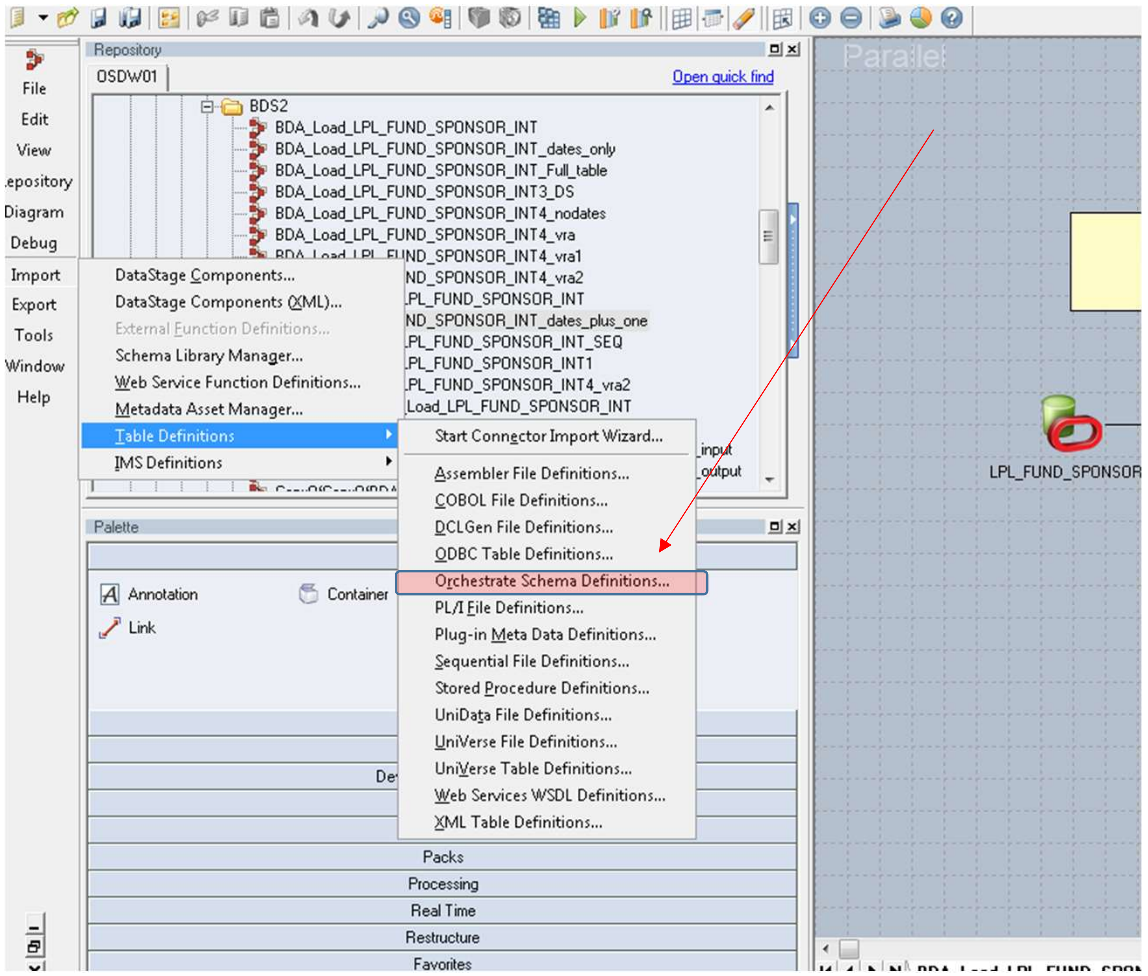
## Selection of File Options



## Setup Database Tables for use

- Any Database table(s) used in this job must be imported into the job
- Every table that is used in the job must be imported individually.
- The name used to set up the ODBC connectivity for the database is utilized in the "DSN Name" box
- Authorization credentials are specified in each individual import stage

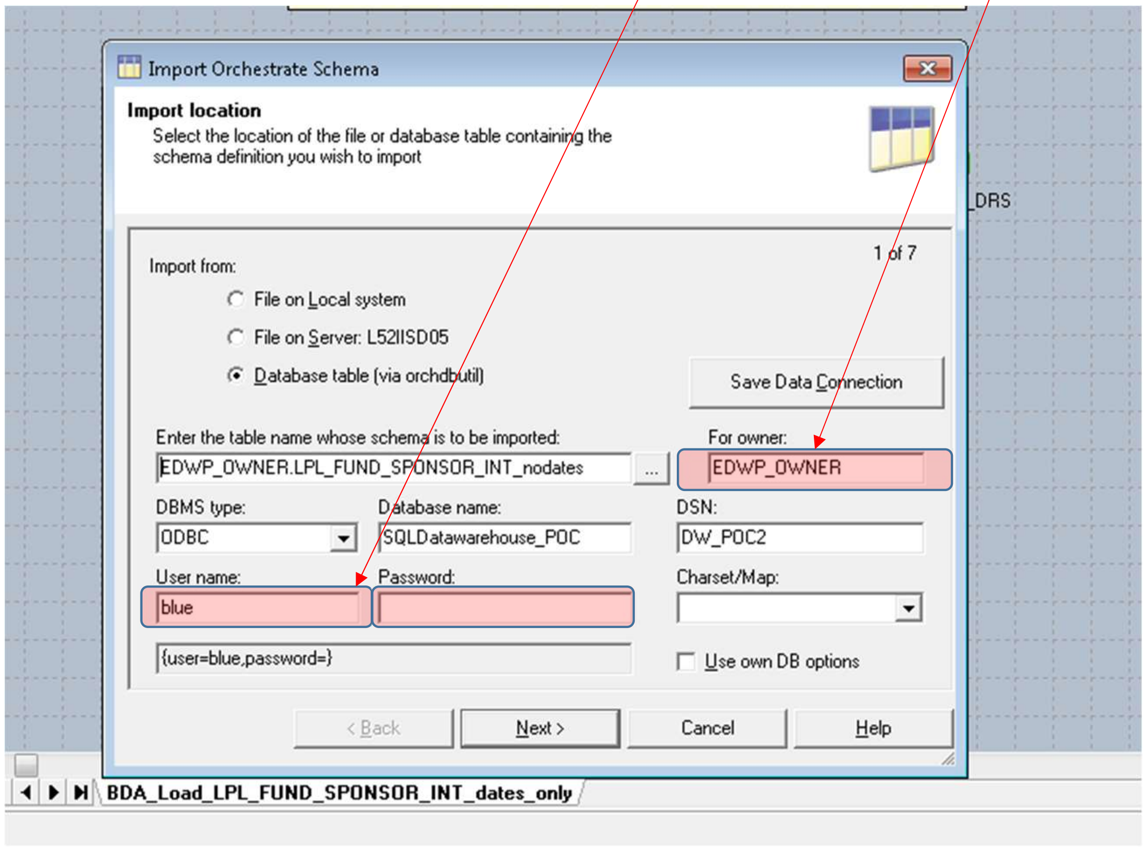
# Import Table Selections


File
Edit
View
Repository
Diagram
Debug
Import
Export
Tools
Window
Help
Repository
OSDW01
Open quick find
BDS2
BDA_Load_LPL_FUND_SPONSOR_INT
BDA_Load_LPL_FUND_SPONSOR_INT_dates_only
BDA_Load_LPL_FUND_SPONSOR_INT_Full_table
BDA_Load_LPL_FUND_SPONSOR_INT3_DS
BDA_Load_LPL_FUND_SPONSOR_INT4_nodates
BDA_Load_LPL_FUND_SPONSOR_INT4_vra
DataStage Components...
DataStage Components (XML)...
External Function Definitions...
Schema Library Manager...
Web Service Function Definitions...
Metadata Asset Manager...
Table Definitions
IMS Definitions
Start Connector Import Wizard...
Assembler File Definitions...
COBOL File Definitions...
DCLGen File Definitions...
ODBC Table Definitions...
Orchestrate Schema Definitions...
PL/I File Definitions...
Plug-in Meta Data Definitions...
Sequential File Definitions...
Stored Procedure Definitions...
UniData File Definitions...
UniVerse File Definitions...
UniVerse Table Definitions...
Web Services WSDL Definitions...
XML Table Definitions...
Palette
Annotation
Container
Link
Packs
Processing
Real Time
Restructure
Favorites
Parallel
LPL_FUND_SPONSOR

Data Migration Jumpstart Engineering

# Import each table used

The database schema must be included in the table import along with the username and password.


Import Orchestrate Schema

**Import location**

Select the location of the file or database table containing the schema definition you wish to import

1 of 7

Import from:

- ( ) File on Local system
- ( ) File on Server: L52IISD05
- [✓ Database table (via orchdbutil)]

Enter the table name whose schema is to be imported: EDWP_OWNER.LPL_FUND_SPONSOR_INT_nodates

For owner: EDWP_OWNER

DBMS type: ODBC

Database name: SQLDatawarehouse_POC

DSN: DW_POC2

User name: blue

Password:

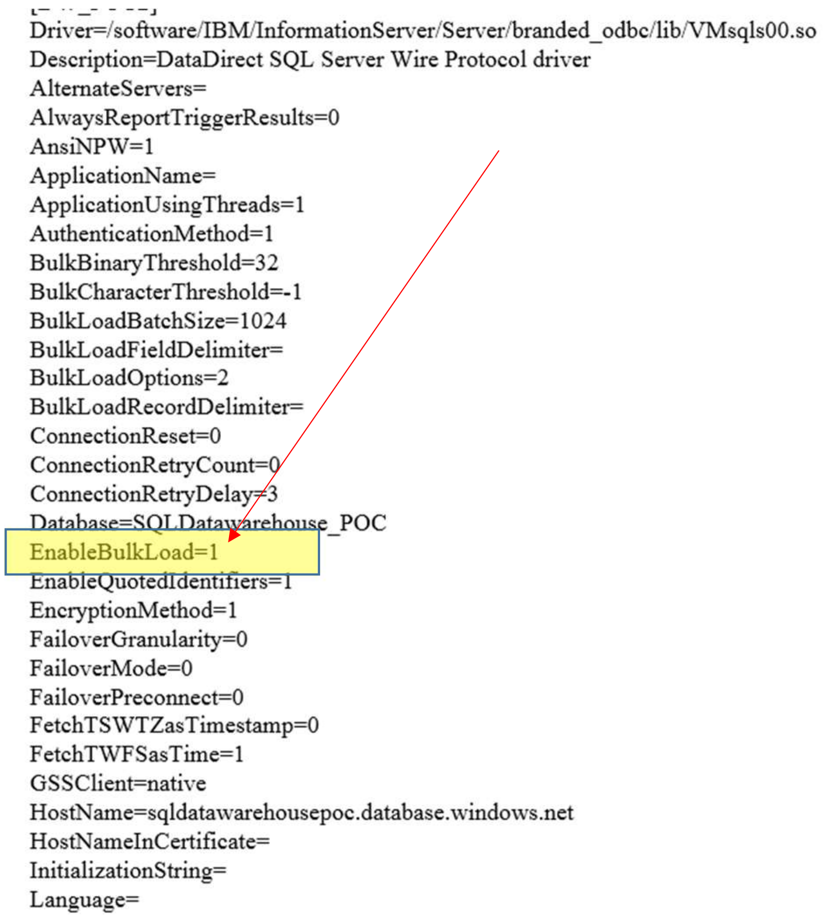
Charset/Map:

{user=blue,password=}

Use own DB options: [ ]

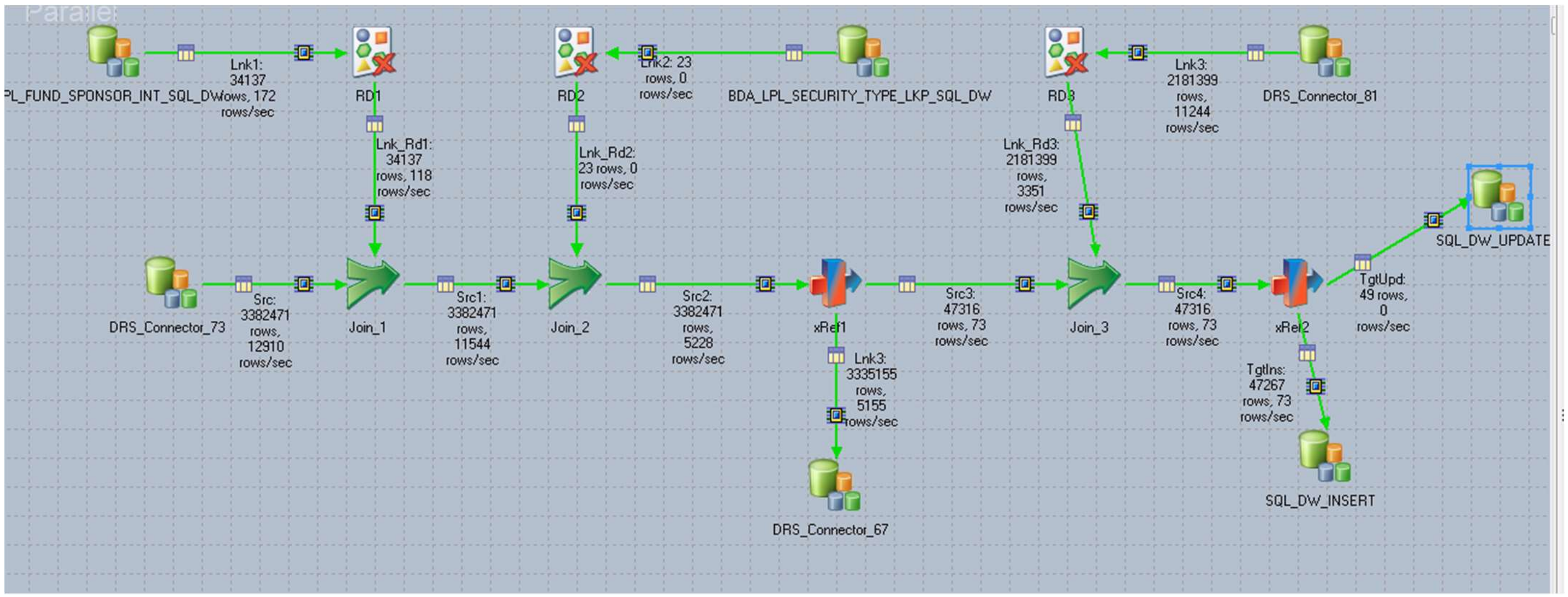
# DataStage ODBC.ini driver parameters

- In order to utilize the Bulk Load option, the *EnableBulkLoad* parameter must be set to 1.



```
Driver=/software/IBM/InformationServer/Server/branded_odbc/lib/VMsqls00.so
Description=DataDirect SQL Server Wire Protocol driver
AlternateServers=
AlwaysReportTriggerResults=0
AnsiNPW=1
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=1
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadFieldDelimiter=
BulkLoadOptions=2
BulkLoadRecordDelimiter=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=SQLDatawarehouse_POC
EnableBulkLoad=1
EnableQuotedIdentifiers=1
EncryptionMethod=1
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
FetchTWFSasTime=1
GSSClient=native
HostName=sqldatawarehousepoc.database.windows.net
HostNameInCertificate=
InitializationString=
Language=
```

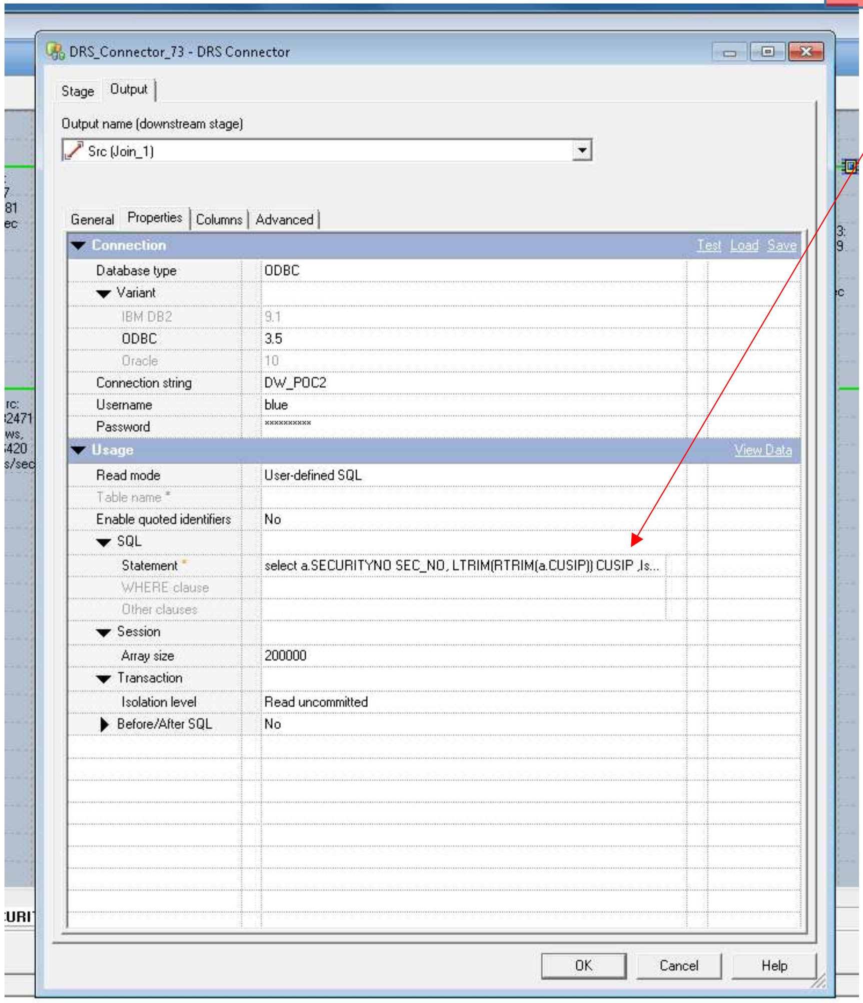
# Converted Sample Job – Update and Insert



Paralle
PL_FUND_SPONSOR_INT_SQL_DW
Lnk1: 34137 rows, 172 rows/sec
RD1
Lnk_Rd1: 34137 rows, 118 rows/sec
RD2
Lnk2: 23 rows, 0 rows/sec
BDA_LPL_SECURITY_TYPE_LKP_SQL_DW
Lnk_Rd2: 23 rows, 0 rows/sec
RD3
Lnk3: 2181399 rows, 11244 rows/sec
DRS_Connector_81
Lnk_Rd3: 2181399 rows, 3351 rows/sec
SQL_DW_UPDATE
DRS_Connector_73
Src: 3382471 rows, 12910 rows/sec
Join_1
Src1: 3382471 rows, 11544 rows/sec
Join_2
Src2: 3382471 rows, 5228 rows/sec
xRef1
Src3: 47316 rows, 73 rows/sec
Join_3
Src4: 47316 rows, 73 rows/sec
xRef2
TgtUpd: 49 rows, 0 rows/sec
Lnk3: 3335155 rows, 5155 rows/sec
TgtIns: 47267 rows, 73 rows/sec
DRS_Connector_67
SQL_DW_INSERT

# Utilize same method of pulling information

Data pulled by "user defined SQL" statement.


DRS_Connector_73 - DRS Connector

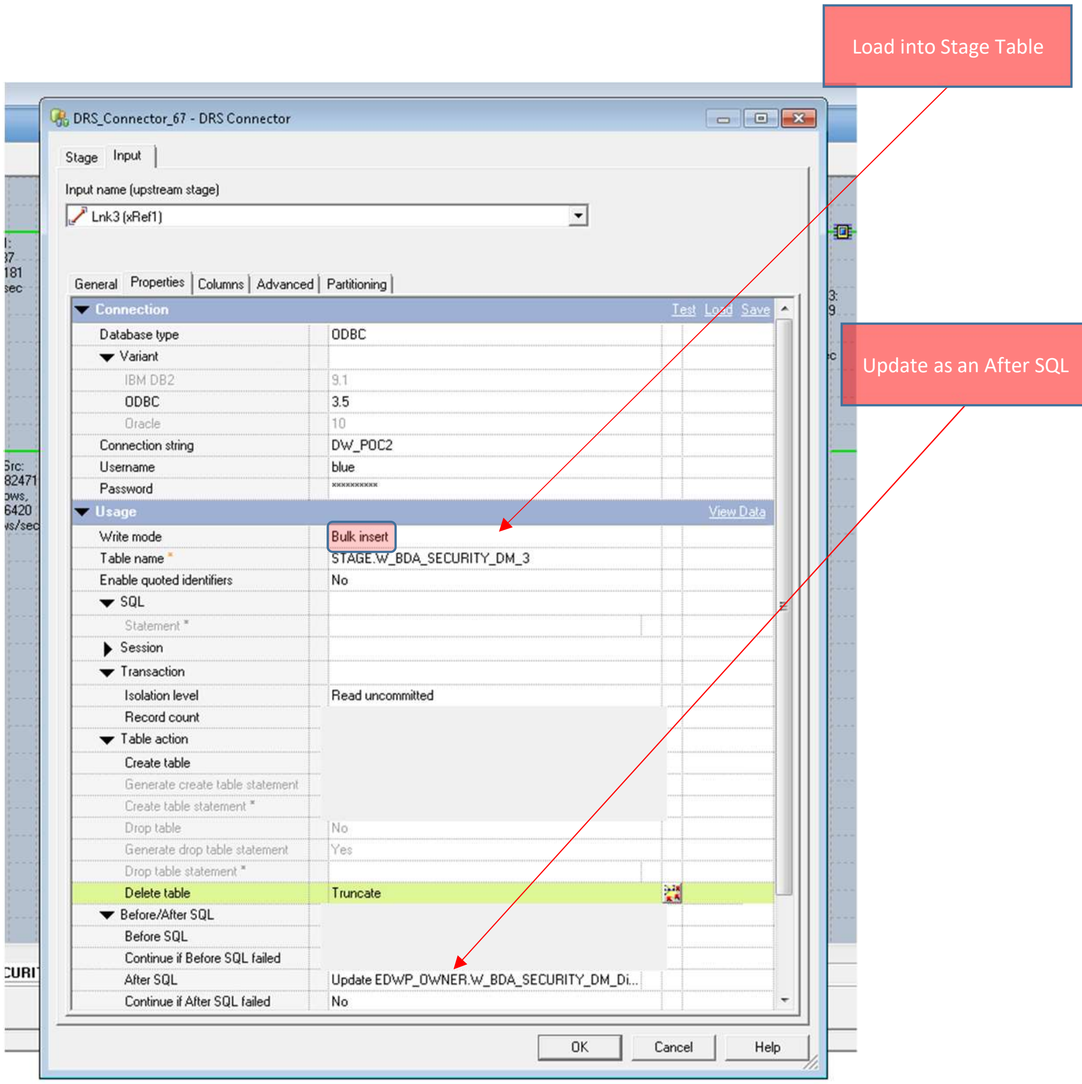
Stage | Output

Output name (downstream stage): Src (Join_1)

General | Properties | Columns | Advanced

| | |
|---|---|
| **Connection** | Test Load Save |
| Database type | ODBC |
| Variant | |
| IBM DB2 | 9.1 |
| ODBC | 3.5 |
| Oracle | 10 |
| Connection string | DW_POC2 |
| Username | blue |
| Password | ********** |
| **Usage** | View Data |
| Read mode | User-defined SQL |
| Table name * | |
| Enable quoted identifiers | No |
| SQL | |
| Statement * | select a.SECURITYNO SEC_NO, LTRIM(RTRIM(a.CUSIP)) CUSIP ,Is... |
| WHERE clause | |
| Other clauses | |
| Session | |
| Array size | 200000 |
| Transaction | |
| Isolation level | Read uncommitted |
| Before/After SQL | No |

For updating data, first, ingest into staging table using Bulk Insert.



DRS_Connector_67 - DRS Connector

Stage | Input

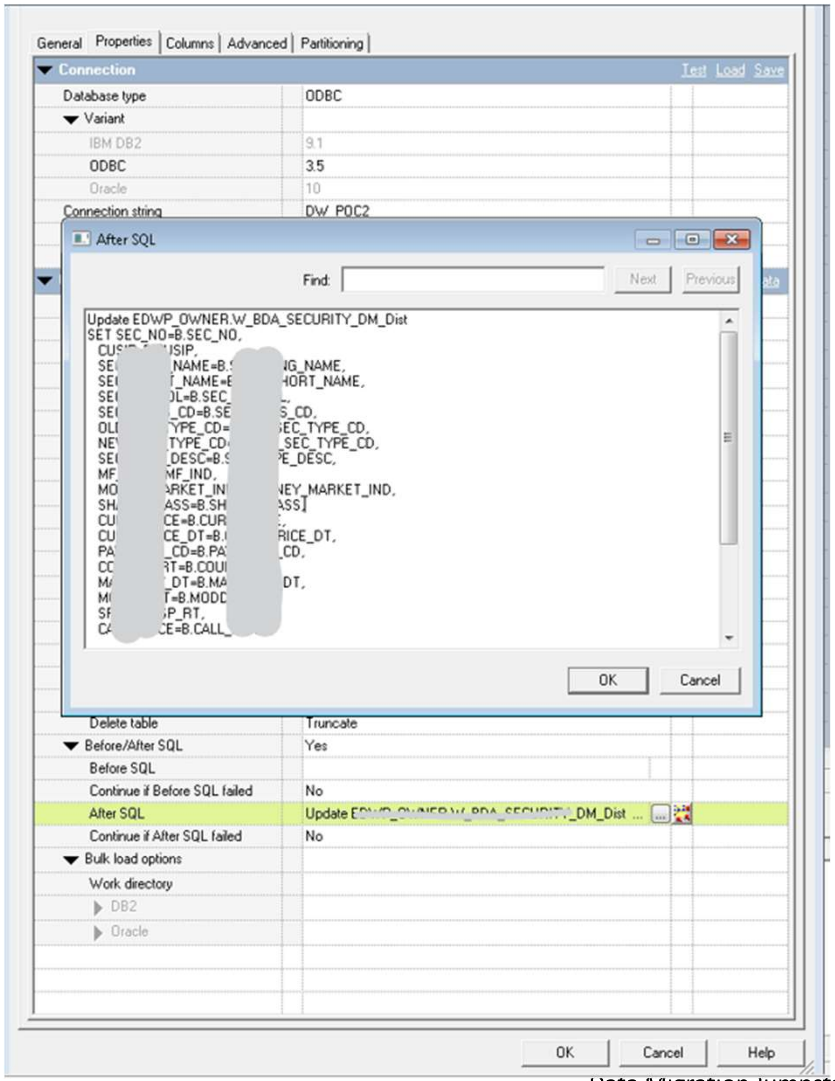
Input name (upstream stage)

Lnk3 (xRef1)

General | Properties | Columns | Advanced | Partitioning

| | |
|---|---|
| **Connection** | Test Load Save |
| Database type | ODBC |
| Variant | |
| IBM DB2 | 9.1 |
| ODBC | 3.5 |
| Oracle | 10 |
| Connection string | DW_POC2 |
| Username | blue |
| Password | ********** |
| **Usage** | View Data |
| Write mode | Bulk insert |
| Table name * | STAGE.W_BDA_SECURITY_DM_3 |
| Enable quoted identifiers | No |
| SQL | |
| Statement * | |
| Session | |
| Transaction | |
| Isolation level | Read uncommitted |
| Record count | |
| Table action | |
| Create table | |
| Generate create table statement | |
| Create table statement * | |
| Drop table | No |
| Generate drop table statement | Yes |
| Drop table statement * | |
| Delete table | Truncate |
| Before/After SQL | |
| Before SQL | |
| Continue if Before SQL failed | |
| After SQL | Update EDWP_OWNER.W_BDA_SECURITY_DM_Di... |
| Continue if After SQL failed | No |

# Detail example of the Update statement.

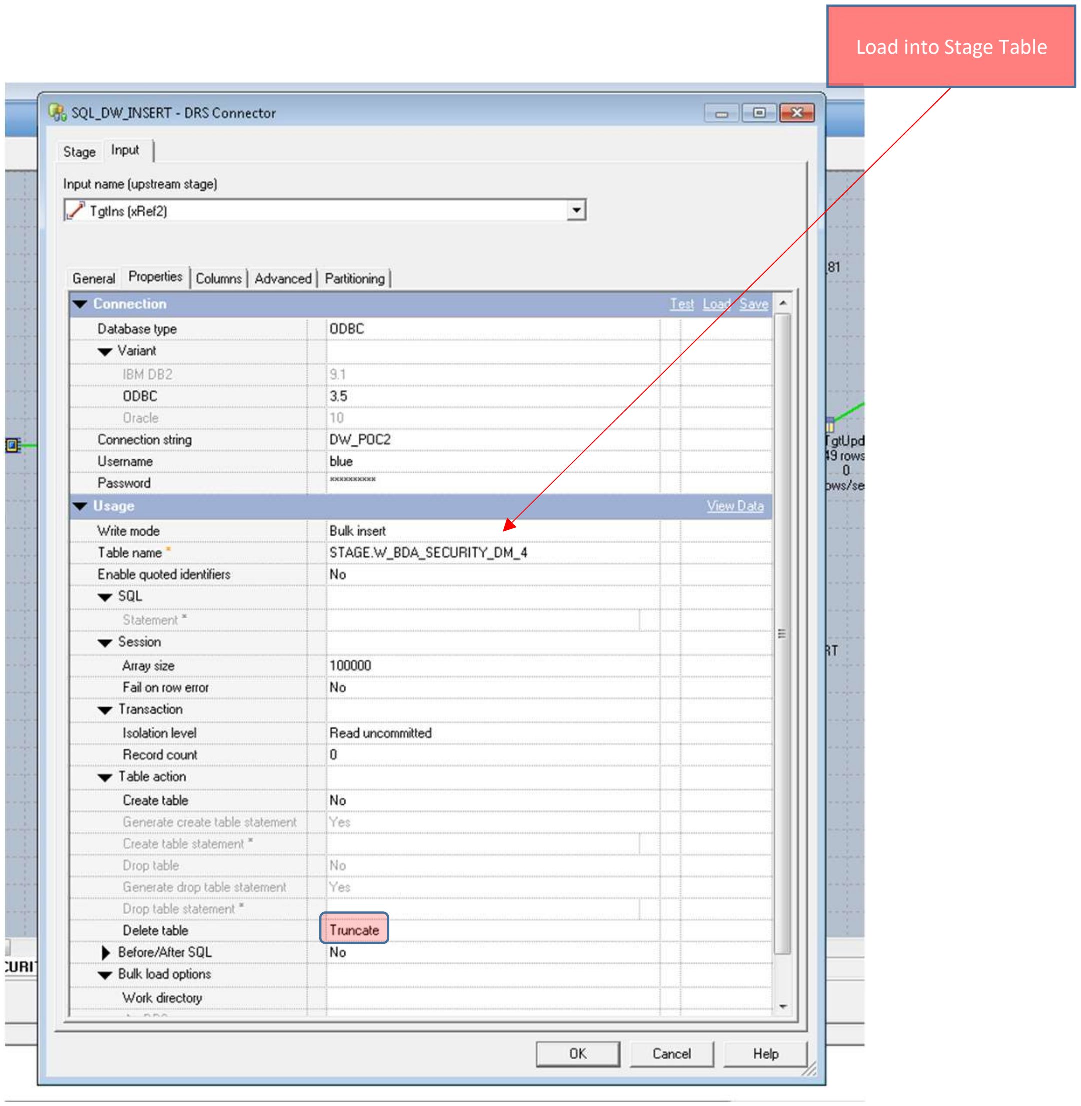


|  |  |
|---|---|
| Database type | ODBC |
| Variant |  |
| IBM DB2 | 9.1 |
| ODBC | 3.5 |
| Oracle | 10 |
| Connection string | DW_POC2 |
| Delete table | Truncate |
| Before/After SQL | Yes |
| Before SQL |  |
| Continue if Before SQL failed | No |
| After SQL | Update E[redacted]_DM_Dist ... |
| Continue if After SQL failed | No |
| Bulk load options |  |
| Work directory |  |
| DB2 |  |
| Oracle |  |

Find:

```
Update EDWP_OWNER.W_BDA_SECURITY_DM_Dist
SET SEC_NO=B.SEC_NO,
  CUS[redacted]USIP,
  SE[redacted]NAME=B.[redacted]NG_NAME,
  SE[redacted]T_NAME=[redacted]HORT_NAME,
  SE[redacted]OL=B.SEC_[redacted]L,
  SE[redacted]_CD=B.SE[redacted]S_CD,
  OL[redacted]TYPE_CD=[redacted]SEC_TYPE_CD,
  NE[redacted]TYPE_CD[redacted]_SEC_TYPE_CD,
  SE[redacted]_DESC=B.[redacted]PE_DESC,
  MF[redacted]MF_IND,
  MO[redacted]ARKET_IN[redacted]NEY_MARKET_IND,
  SH[redacted]ASS=B.SH[redacted]ASS
  CU[redacted]CE=B.CUR[redacted],
  CU[redacted]CE_DT=B.[redacted]RICE_DT,
  PA[redacted]_CD=B.PA[redacted]_CD,
  CC[redacted]RT=B.COU[redacted]
  MA[redacted]_DT=B.MA[redacted]DT,
  M[redacted]T=B.MODD[redacted]
  SF[redacted]P_RT,
  CA[redacted]CE=B.CALL_[redacted]
```

# Similar approach to Insert



Input name (upstream stage)

TgtIns (xRef2)

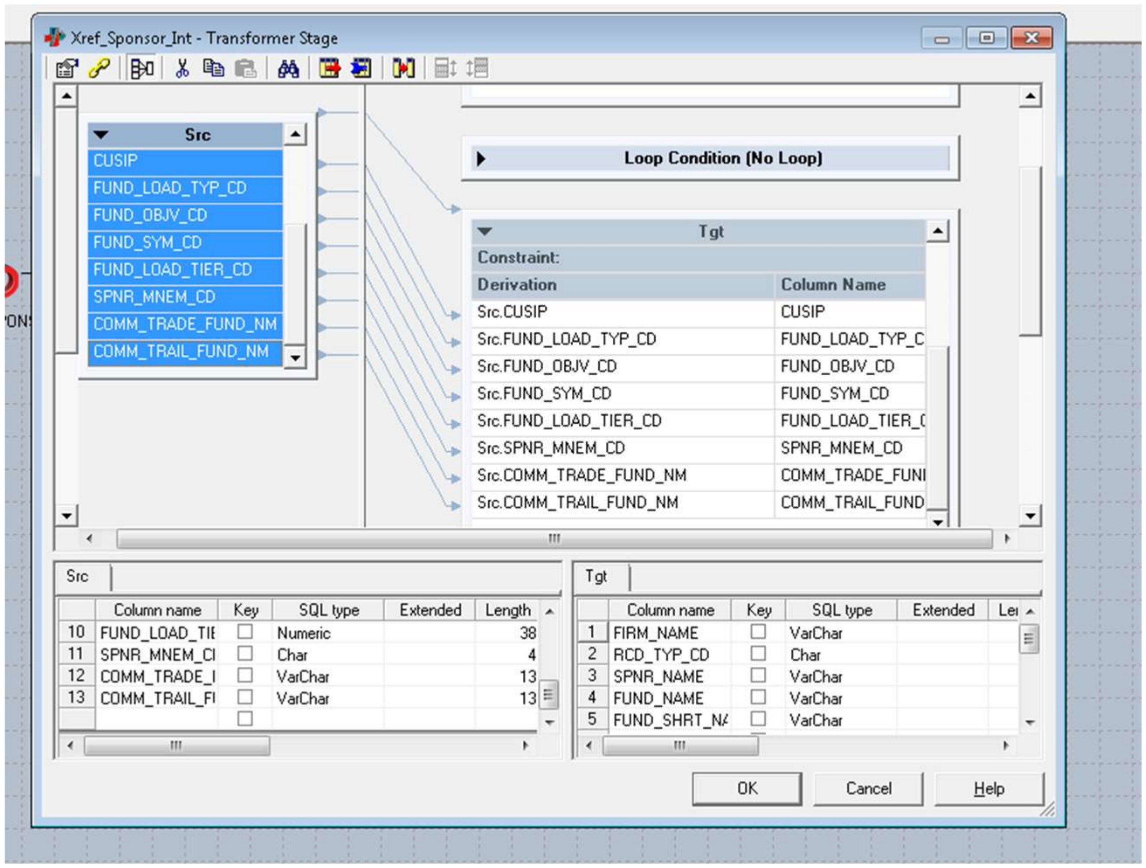
| | |
|---|---|
| **Connection** | |
| Database type | ODBC |
| Variant | |
| IBM DB2 | 9.1 |
| ODBC | 3.5 |
| Oracle | 10 |
| Connection string | DW_POC2 |
| Username | blue |
| Password | ********** |
| **Usage** | |
| Write mode | Bulk insert |
| Table name * | STAGE.W_BDA_SECURITY_DM_4 |
| Enable quoted identifiers | No |
| SQL | |
| Statement * | |
| Session | |
| Array size | 100000 |
| Fail on row error | No |
| Transaction | |
| Isolation level | Read uncommitted |
| Record count | 0 |
| Table action | |
| Create table | No |
| Generate create table statement | Yes |
| Create table statement * | |
| Drop table | No |
| Generate drop table statement | Yes |
| Drop table statement * | |
| Delete table | Truncate |
| Before/After SQL | No |
| Bulk load options | |
| Work directory | |

Data Migration Jumpstart Engineering

## Changes to Transformers

- When changing the sources and targets, assure that you are replacing the existing stages by deleting them and then pasting the new ones in.
- Do not delete the connection arrows.
- Open each Transformer processing module and change necessary sources and/or targets information

# Change Transformer Processor



Src

| Src |
|---|
| CUSIP |
| FUND_LOAD_TYP_CD |
| FUND_OBJV_CD |
| FUND_SYM_CD |
| FUND_LOAD_TIER_CD |
| SPNR_MNEM_CD |
| COMM_TRADE_FUND_NM |
| COMM_TRAIL_FUND_NM |

Tgt

Constraint:

| Derivation | Column Name |
|---|---|
| Src.CUSIP | CUSIP |
| Src.FUND_LOAD_TYP_CD | FUND_LOAD_TYP_C |
| Src.FUND_OBJV_CD | FUND_OBJV_CD |
| Src.FUND_SYM_CD | FUND_SYM_CD |
| Src.FUND_LOAD_TIER_CD | FUND_LOAD_TIER_( |
| Src.SPNR_MNEM_CD | SPNR_MNEM_CD |
| Src.COMM_TRADE_FUND_NM | COMM_TRADE_FUNI |
| Src.COMM_TRAIL_FUND_NM | COMM_TRAIL_FUND |

Src

| | Column name | Key | SQL type | Extended | Length |
|---|---|---|---|---|---|
| 10 | FUND_LOAD_TIE | ☐ | Numeric | | 38 |
| 11 | SPNR_MNEM_CI | ☐ | Char | | 4 |
| 12 | COMM_TRADE_I | ☐ | VarChar | | 13 |
| 13 | COMM_TRAIL_FI | ☐ | VarChar | | 13 |
| | | ☐ | | | |

Tgt

| | Column name | Key | SQL type | Extended | Le |
|---|---|---|---|---|---|
| 1 | FIRM_NAME | ☐ | VarChar | | |
| 2 | RCD_TYP_CD | ☐ | Char | | |
| 3 | SPNR_NAME | ☐ | VarChar | | |
| 4 | FUND_NAME | ☐ | VarChar | | |
| 5 | FUND_SHRT_N/ | ☐ | VarChar | | |

## Executing Stored Proc on Azure Synapse Analytics

- If an additional step is required to creating a sequence
- Create the stored procedure on Azure Synapse Analytics SQL Pool
- Create a job to execute the stored procedure
- Set up a "Sequence" job to execute the job that inserts and updates and then executes the stored proc

# Stored Procedure to Build Key and Insert

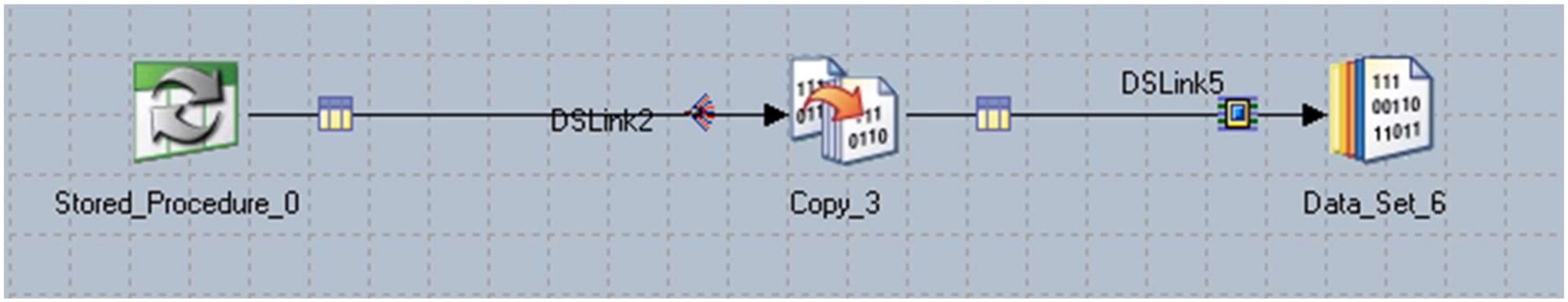
```
CREATE PROCEDURE STAGE_INSERT AS
BEGIN
DECLARE @security_id bigint
Select @security_id = MAX(ROW_ID) FROM SAMPLE_SCHEMA.W_BDA_SECURITY_DM_Dist
--
if Exists(select 1 from sys.tables where name = 'W_BDA_SECURITY_DM_4_ROWNUM') Drop Table STAGE.W_BDA_SECURITY_DM_4_ROWNUM
create table STAGE.W_BDA_SECURITY_DM_4_ROWNUM
with (distribution = replicate, heap) as
select (@security_id +row_number() over(order by sec_no, cusip asc)) as NEW_ROW_NUM,
SEC_NO, CUSIP,
SEC_LONG_NAME,
SEC_SHORT_NAME,
SEC_SYMBOL,
SEC_CLASS_CD,
OLD_SEC_TYPE_CD,
NEW_SEC_TYPE_CD,
SEC_TYPE_DESC,
MF_IND,MONEY_MARKET_IND,SHARE_CLASS,CURR_PRICE,CURR_PRICE_DT,PAY_FREQ_CD,COUPON_RT,MATURITY_DT
,MODDY_RT,SP_RT,CALL_PRICE,DIVIDEND,FACTOR_DT,FACTOR,NOTE,INTEGRATION_TSTP
FROM STAGE.W_BDA_SECURITY_DM_4

INSERT INTO SAMPLE_SCHEMA.W_BDA_SECURITY_DM_Dist
(ROW_ID,
SEC_NO,
CUSIP,
SEC_LONG_NAME,
SEC_SHORT_NAME,
SEC_SYMBOL,
SEC_CLASS_CD,
OLD_SEC_TYPE_CD,
NEW_SEC_TYPE_CD,
SEC_TYPE_DESC,
MF_IND,MONEY_MARKET_IND,SHARE_CLASS,CURR_PRICE,CURR_PRICE_DT,PAY_FREQ_CD,COUPON_RT,MATURITY_DT,
MODDY_RT,SP_RT,CALL_PRICE,DIVIDEND,FACTOR_DT,FACTOR,NOTE,INTEGRATION_TSTP
)
SELECT
B.NEW_ROW_NUM, B.SEC_NO, B.CUSIP, B.SEC_LONG_NAME,
  B.SEC_SHORT_NAME, B.SEC_SYMBOL,
 B.SEC_CLASS_CD, B.OLD_SEC_TYPE_CD, B.NEW_SEC_TYPE_CD, B.SEC_TYPE_DESC, B.MF_IND, B.MONEY_MARKET_IND,
 B.SHARE_CLASS, B.CURR_PRICE, B.CURR_PRICE_DT, B.PAY_FREQ_CD, B.COUPON_RT, B.MATURITY_DT,
 B.MODDY_RT, B.SP_RT, B.CALL_PRICE, B.DIVIDEND, B.FACTOR_DT, B.FACTOR, B.NOTE, B.INTEGRATION_TSTP
 FROM STAGE.W_BDA_SECURITY_DM_4_ROWNUM B
 END
```

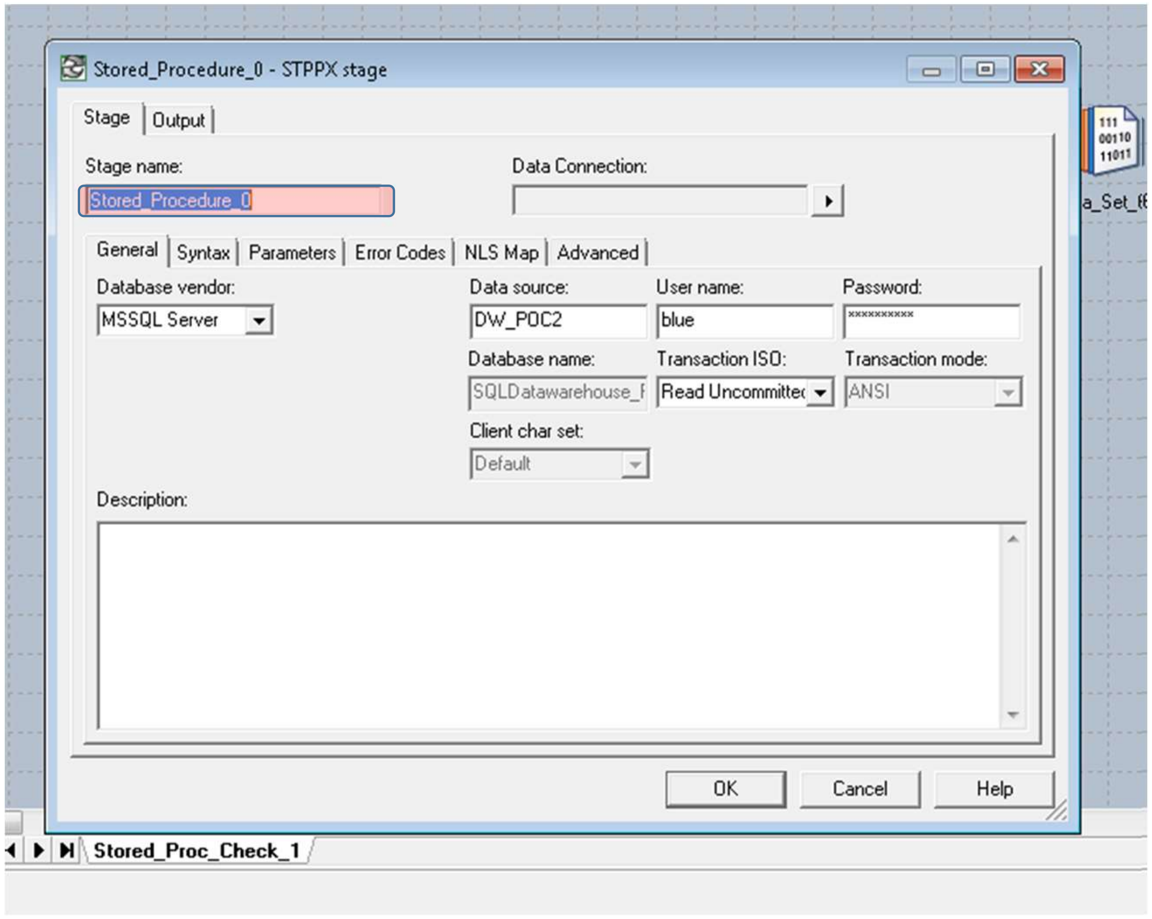
Stage Table with sequencer

Target Table

## DataStage – Job to Execute Stored Procedure



# Stored Proc General Setup



Stored_Procedure_0 - STPPX stage

Stage | Output

Stage name: Stored_Procedure_0

Data Connection:

General | Syntax | Parameters | Error Codes | NLS Map | Advanced

Database vendor: MSSQL Server

Data source: DW_POC2

User name: blue

Password: **********

Database name: SQLDatawarehouse_I

Transaction ISO: Read Uncommittec

Transaction mode: ANSI

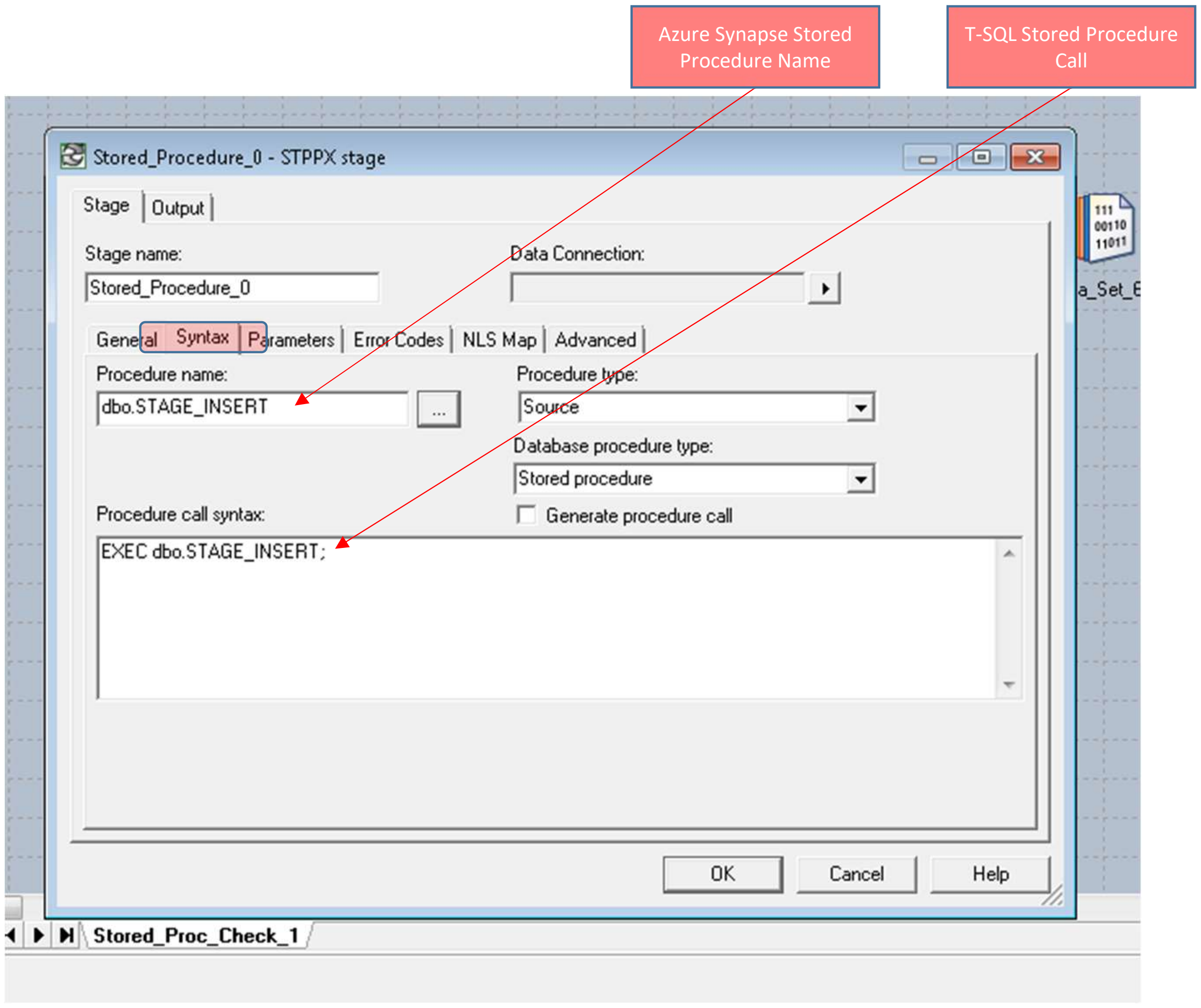
Client char set: Default

Description:

OK | Cancel | Help

Stored_Proc_Check_1

# Set Up Syntax for Stored Proc Exec



Azure Synapse Stored Procedure Name
T-SQL Stored Procedure Call
Stored_Procedure_0 - STPPX stage
Stage
Output
General
Syntax
Parameters
Error Codes
NLS Map
Advanced
OK
Cancel
Help
Stored_Proc_Check_1

Stage name: Stored_Procedure_0

Data Connection:

Procedure name: dbo.STAGE_INSERT

Procedure type: Source

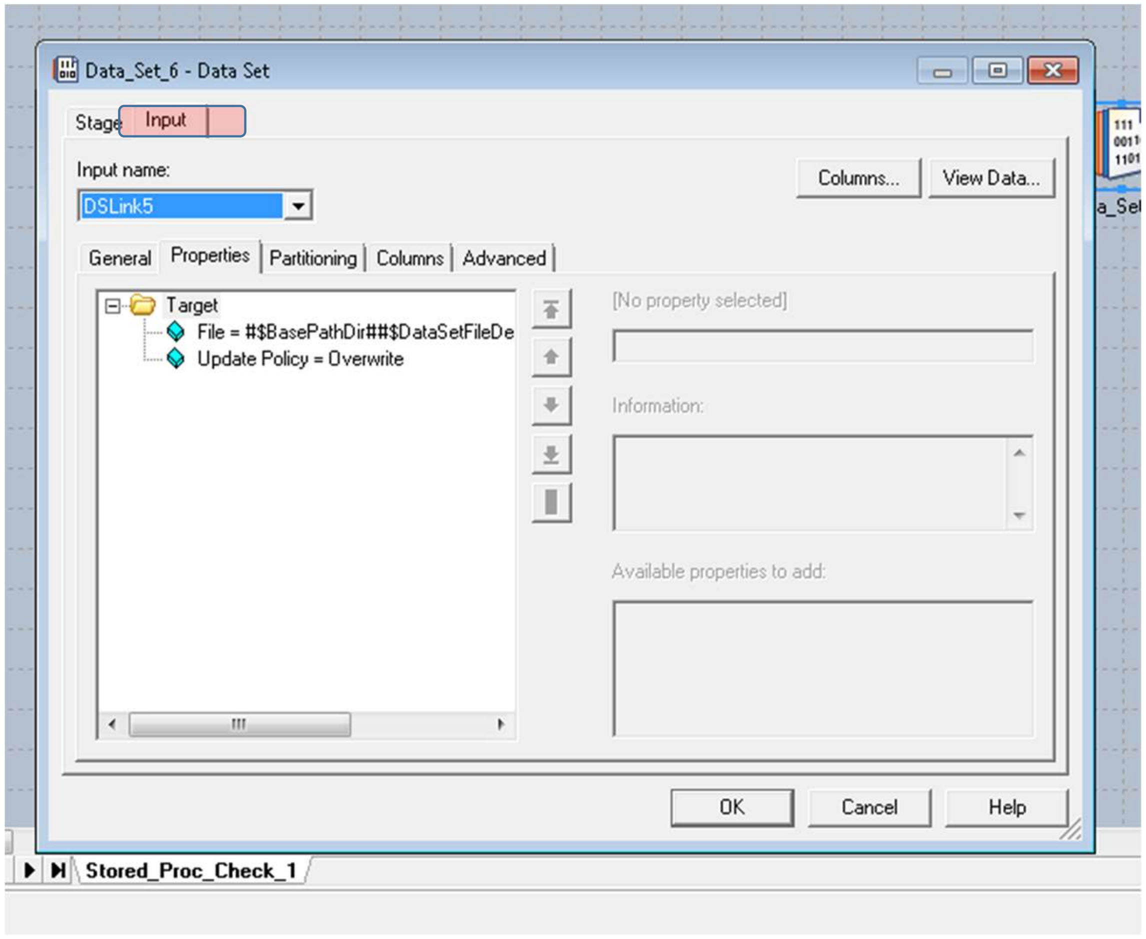
Database procedure type: Stored procedure

☐ Generate procedure call

Procedure call syntax:

```
EXEC dbo.STAGE_INSERT;
```

## Set up "Copy" process Input Properties



Data_Set_6 - Data Set

Stage | Input

Input name: DSLink5

Columns... | View Data...

General | Properties | Partitioning | Columns | Advanced

Target
- File = #$BasePathDir##$DataSetFileDe
- Update Policy = Overwrite

[No property selected]

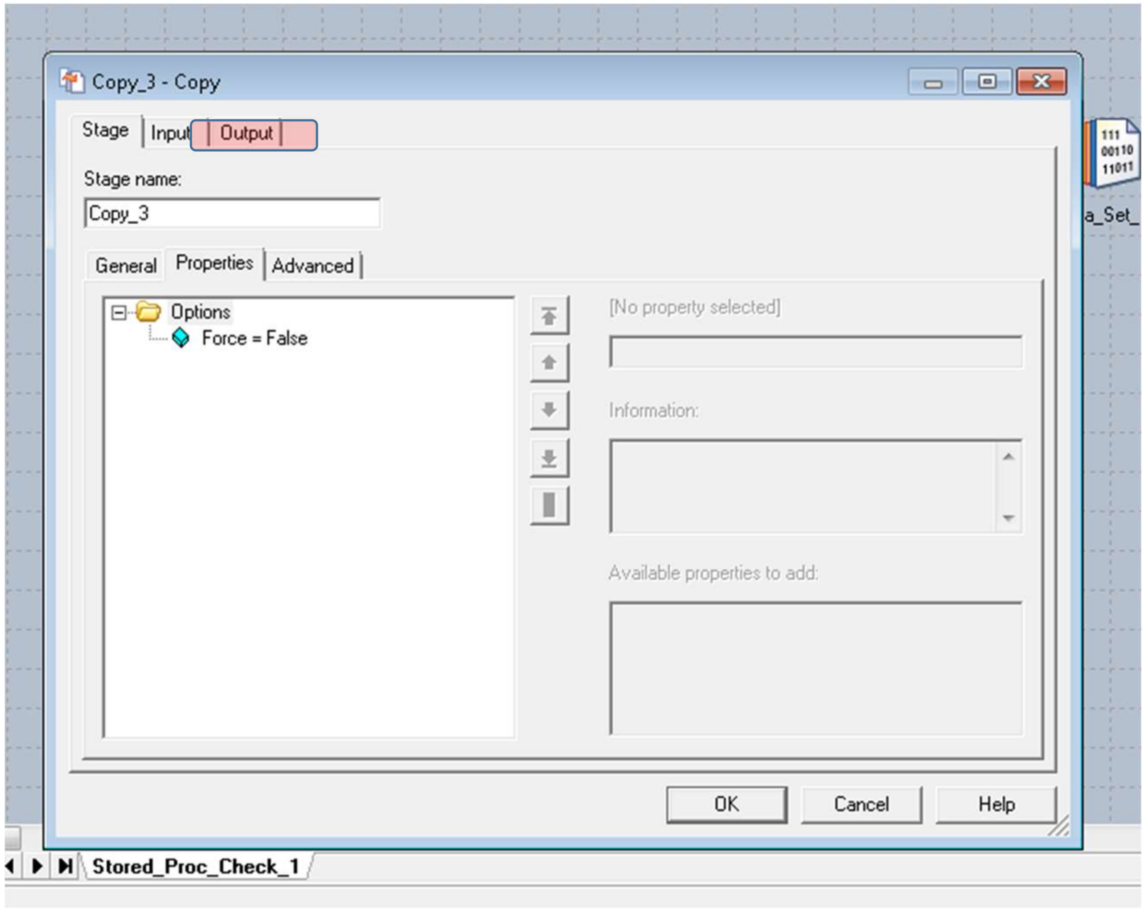
Information:

Available properties to add:

OK | Cancel | Help

Stored_Proc_Check_1

## Set up "Copy" Process Output Properties for Stored Proc


Copy_3 - Copy

Stage | Input | Output

Stage name:

Copy_3

General | Properties | Advanced

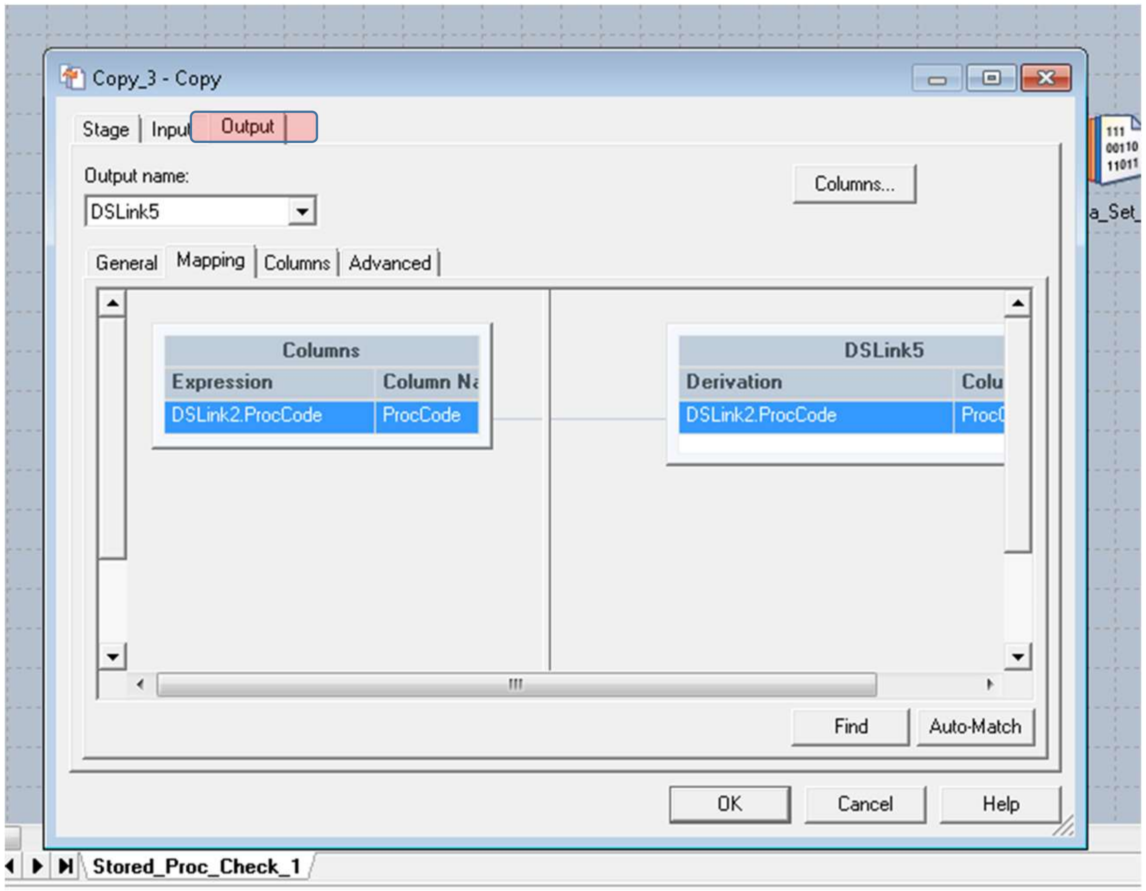
Options
Force = False

[No property selected]

Information:

Available properties to add:

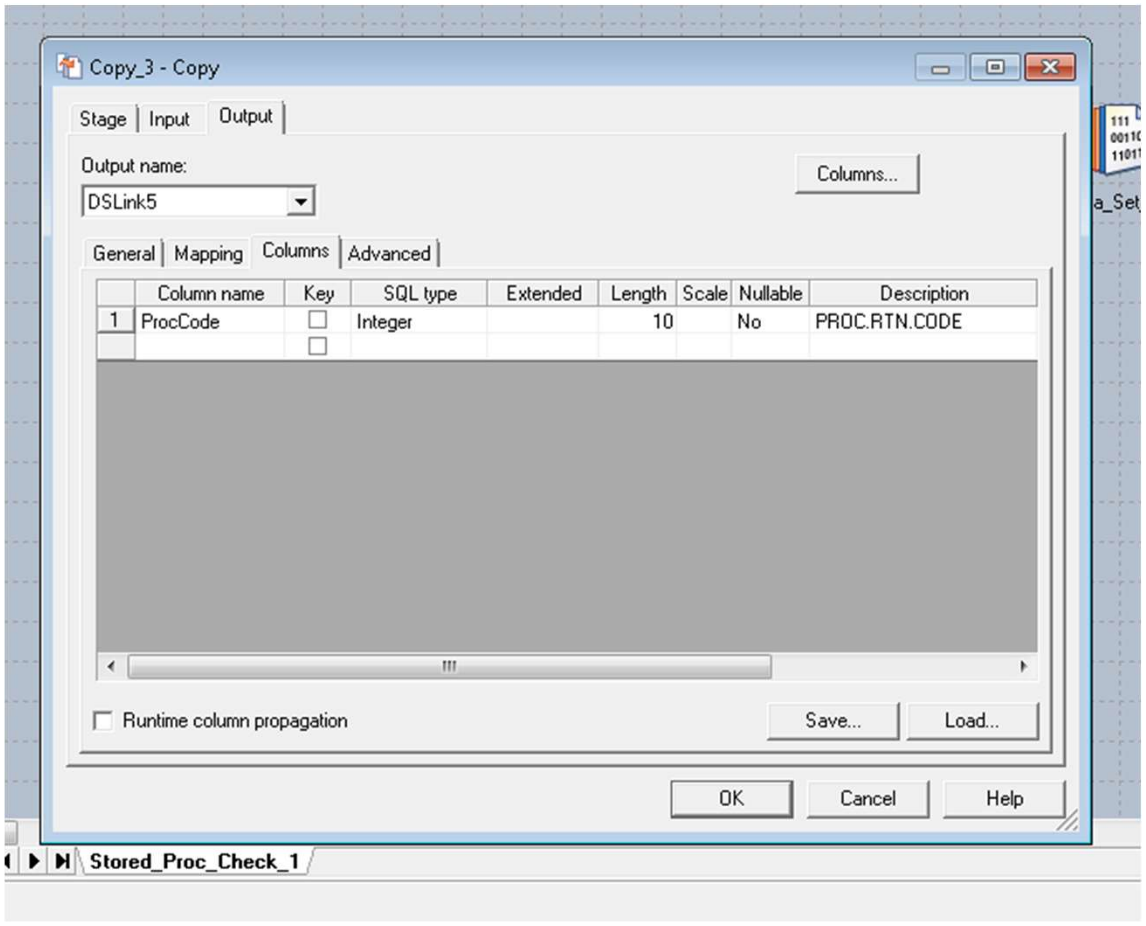
# Set up "Copy" process for Output Mapping



| Columns | |
|---|---|
| Expression | Column Na |
| DSLink2.ProcCode | ProcCode |

| DSLink5 | |
|---|---|
| Derivation | Colu |
| DSLink2.ProcCode | Proc |

## Set up "Copy" process Output Columns



Copy_3 - Copy

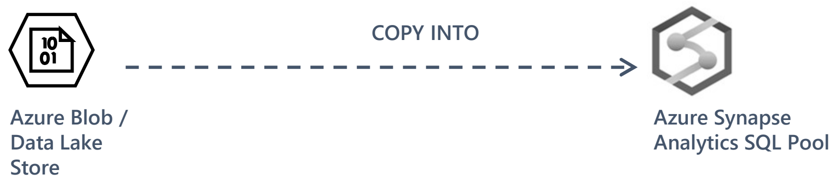
Output name: DSLink5

| | Column name | Key | SQL type | Extended | Length | Scale | Nullable | Description |
|---|---|---|---|---|---|---|---|---|
| 1 | ProcCode | ☐ | Integer | | 10 | | No | PROC.RTN.CODE |
| | | ☐ | | | | | | |

☐ Runtime column propagation

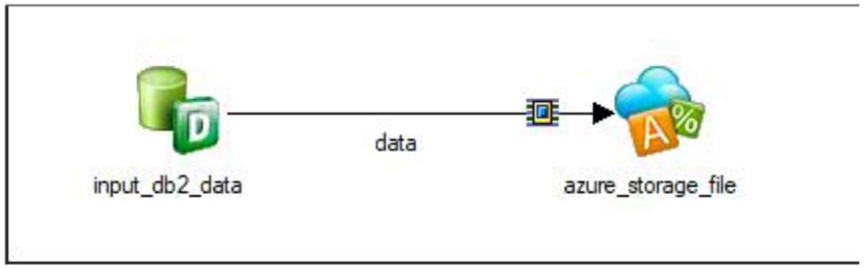
## Large Data Ingestion in Azure Synapse Strategy

- Large Data Set ingestion should follow a pattern
  - Land data intro Azure Blob Container or Data Lake Store
  - Ingest into a staging table in Azure Synapse Analytics using COPY INTO
  - CTAS or Insert into Final Table
  - Refresh statistics as needed
- DataStage can orchestrate the reading from source (e.g. flat files) and ingesting into Blob Storage as a first step


COPY INTO
Azure Blob / Data Lake Store
Azure Synapse Analytics SQL Pool

## Writing Data Into Azure Storage with DataStage

- Large Data Set ingestion can take advantage of COPY INTO process from Azure Synapse Analytics from data landed in Blob Storage
- Requirements: IBM Infosphere Information Server Datastage 11.7fp1 and above
- High Level Steps:
  1. Configure Azure Storage Connector Connection Properties
  2. Configure Azure Storage Connector to write to Azure Blob Storage
  3. Additional Configuration for Parallel Write
- Documentation Link



## Bulk Loading into Azure Synapse with DataStage

- COPY INTO statement can be executed:
  - Within a Stored Procedure
  - Or as an ad-hoc T-SQL
- COPY INTO will take the data from Blob Storage/ADLS and ingest into Azure Synapse Analytics in parallel
- Data should be landed on an uncompressed rowstore table for performance reasons
- T-SQL should execute a last step of ingesting or CTAS into a final table.
- Recompute of statistics is always recommended when data has changed significantly

# Bulk Loading with COPY INTO statement

```
CREATE SCHEMA stage;
-- Create the Staging Table
-- Drop Table [stage].[SalesData]
Create table [stage].[SalesData](
                    Region varchar(50)
                    ,Product varchar(50)
                    ,SaleDate date
                    ,Amount DECIMAL (20,10)
                    )
with (distribution = round_robin, HEAP);


/* Documentation
https://docs.microsoft.com/en-us/sql/t-sql/statements/copy-into-transact-sql?view=azure-sqldw-latest
*/

COPY INTO [stage].[SalesData] (Region 1, Product 2, SaleDate 3, Amount 4)
FROM 'https://acccount.blob.core.windows.net/yourcontainer'
WITH (
   FILE_TYPE = 'CSV',
   CREDENTIAL=(IDENTITY= 'Shared Access Signature', SECRET='Your secret here'),
                    FIELDQUOTE = '"',
                    FIELDTERMINATOR=',',
                    ROWTERMINATOR = '\n',
                    Identity_INSERT = 'OFF',
                    ENCODING = 'UTF8',
                    DATEFORMAT = 'ymd',
                    FIRSTROW = 2
);

CREATE SCHEMA Sales;
-- CTAS with HASH distribution on Date and Columnar format
-- DROP TABLE Sales.Transactions
CREATE TABLE Sales.Transactions
WITH
(
 DISTRIBUTION = HASH(SaleDate)
 ,CLUSTERED COLUMNSTORE INDEX
)  AS SELECT  * FROM [stage].[SalesData];

Create Statistics stat_stage_salesdata_Region on Sales.Transactions (Region) with fullscan;
Create Statistics stat_stage_salesdata_Product on Sales.Transactions (Product) with fullscan;
Create Statistics stat_stage_salesdata_SaleDate on Sales.Transactions (SaleDate) with fullscan;
Create Statistics stat_stage_salesdata_Amount on Sales.Transactions (Amount) with fullscan;
```

No compression

Not necessary if using AAD / Role based access control

CTAS into final table, with compression and distribution

Statistics are created