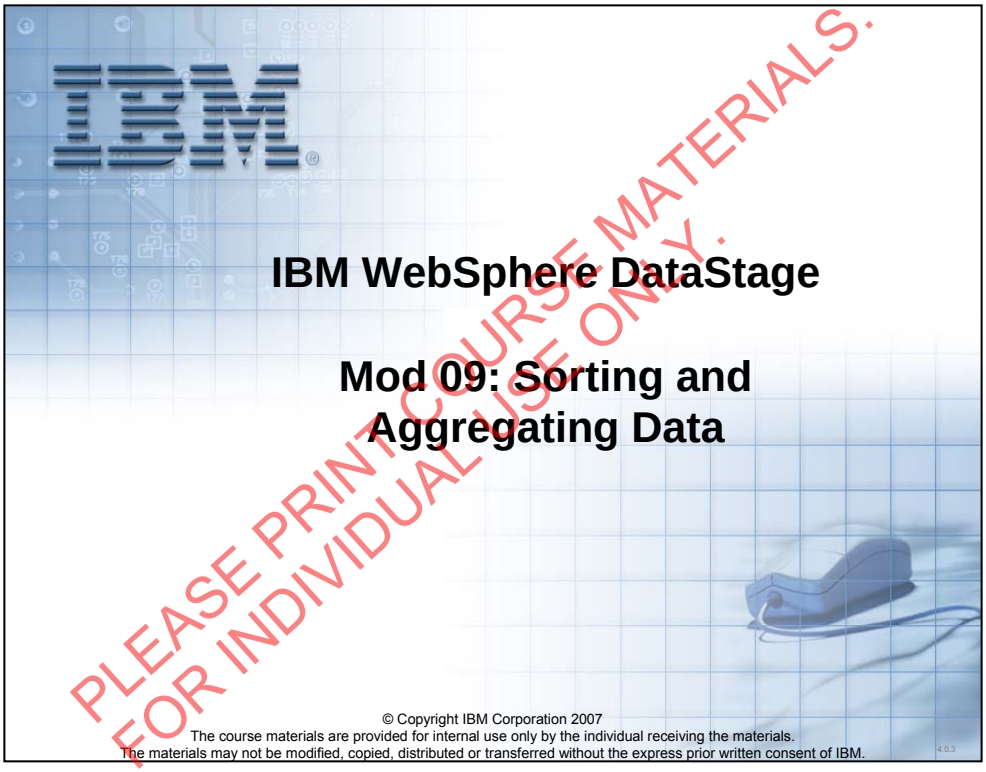

IBM
IBM WebSphere DataStage
Mod 09: Sorting and Aggregating Data
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:

- Sort data using in-stage sorts and Sort stage
- Combine data using Aggregator stage
- Combine data Remove Duplicates stage





2

**Notes:**

# Sort Stage





3

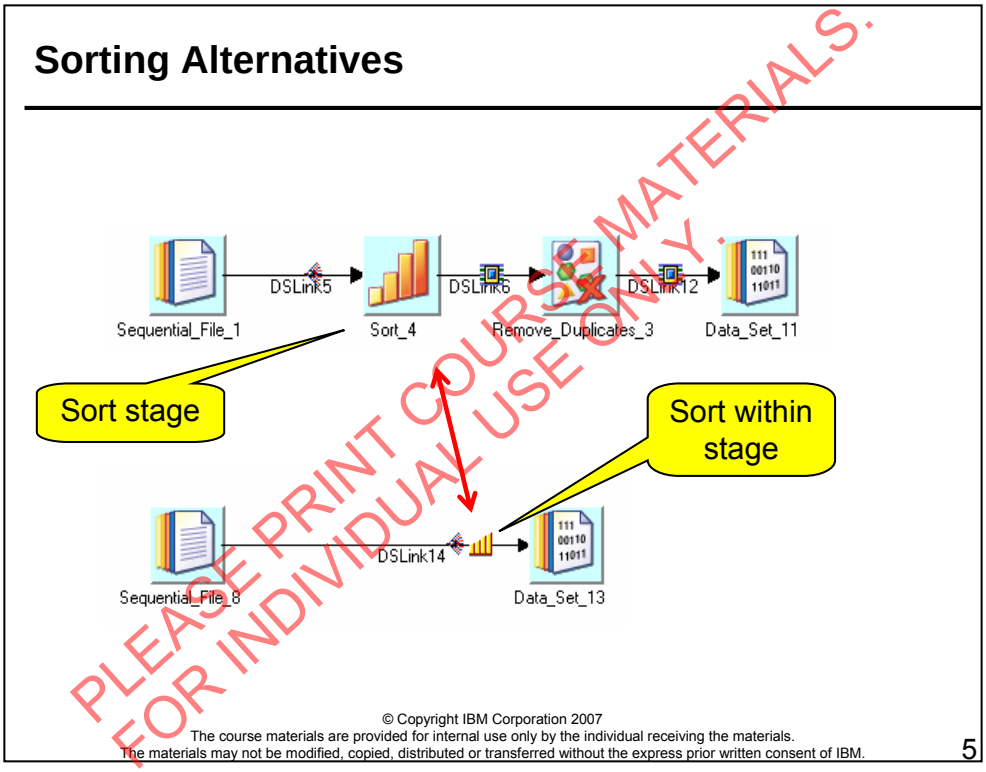
## Sorting Data

- Uses
  - Some stages require sorted input
    - Join, merge stages require sorted input
  - Some stages use less memory with sorted input
    - E.g., Aggregator
- Sorts can be done:
  - Within stages
    - On input link Partitioning tab, set partitioning to anything other than Auto
  - In a separate Sort stage
    - Makes sort more visible on diagram
    - Has more options

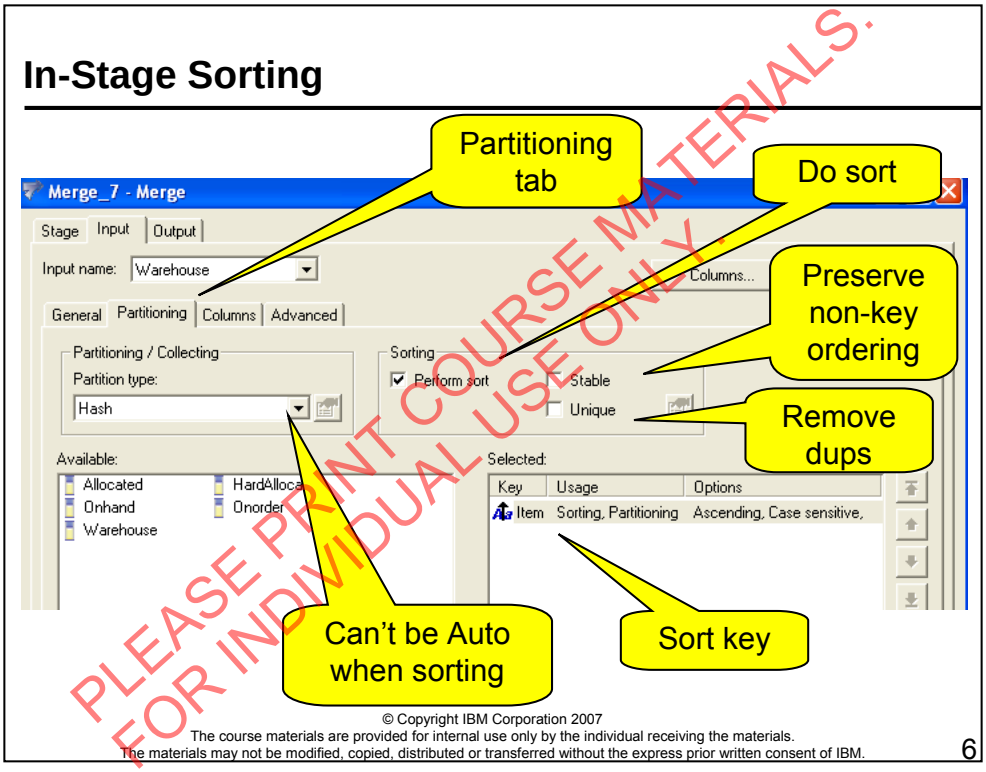




4

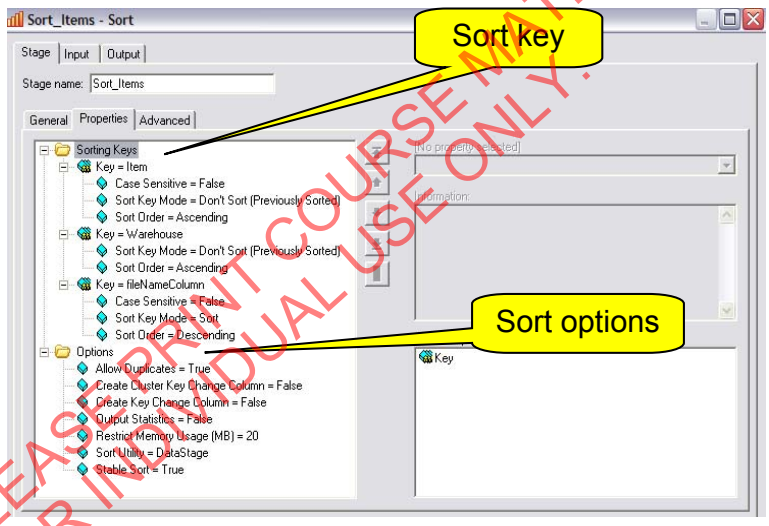



Which should you use? The first is more explicit and has more options. For example, the first has properties for specifying the amount of memory allocated.




Stable will preserve the original ordering of records within each key group. If set to false no prior ordering of records is guaranteed to be preserved by the sorting operation.

## Sort Stage








7

## Sort keys

- Add one or more keys
- Specify sort mode for each key
  - Sort: Sort by this key
  - Don't sort (previously sorted):
    - Assumes the data has already been sorted on this key
    - Continue sorting by any secondary keys
- Specify sort order: ascending / descending
- Specify case sensitive or not





8

## Sort Options

- Sort Utility
  - DataStage – the default
  - Unix: Don't use. Slower than DataStage sort utility
- Stable
- Allow duplicates
- Memory usage
  - Sorting takes advantage of the available memory for increased performance
    - Uses disk if necessary
  - Increasing amount of memory can improve performance
- Create key change column
  - Add a column with a value of 1 / 0
  - 1 indicates that the key value has changed
  - 0 means that the key value hasn't changed
  - Useful for processing groups of rows in a Transformer





9

## Partitioning V. Sorting Keys

- Partitioning keys are often different than Sorting keys
  - Keyed partitioning (e.g., Hash) is used to group related records into the same partition
  - Sort keys are used to establish order within each partition
- Example:
  - Partition on HouseHoldID, sort on HouseHoldID, EntryDate
    - Partitioning on HouseHoldID ensures that the same ID will not be spread across multiple partitions
    - Sorting orders the records with the same ID by entry date
      - Useful for deciding which of a group of duplicate records with the same ID should be retained





10

# Aggregator Stage





11

## Aggregator Stage

Purpose: Perform data aggregations
Specify:

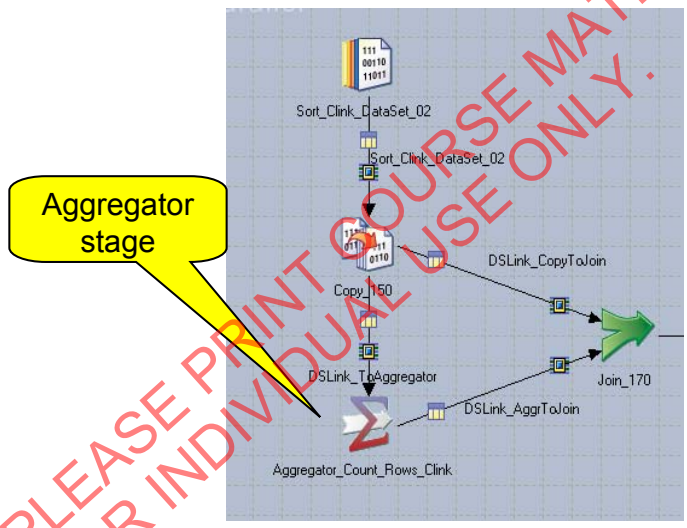
- One or more key columns that define the aggregation units (or groups)
- Columns to be aggregated
- Aggregation functions include, among many others:
  - count (nulls/non-nulls)
  - Sum
  - Max / Min / Range
- The grouping method (*hash table* or *pre-sort*) is a performance issue





12

## Job with Aggregator Stage








13

## Aggregation Types

- Count rows
  - Count rows in each group
  - Put result in a specified output column
- Calculation
  - Select column
  - Put result of calculation in a specified output column
  - Calculations include:
    - Sum
    - Count
    - Min, max
    - Mean
    - Missing value count
    - Non-missing value count
    - Percent coefficient of variation

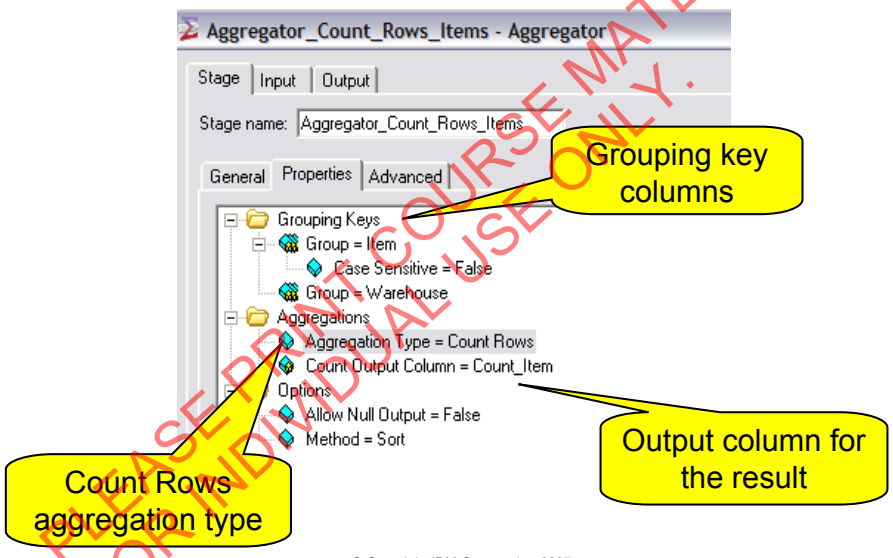




14

There are two basic aggregation types: Count rows, calculation. The former counts the number of rows in each group. With the latter type, you select an input column that you want to perform calculations on. Then you select the calculations to perform on that input column and the output columns to put the results in.

## Count Rows Aggregator Properties

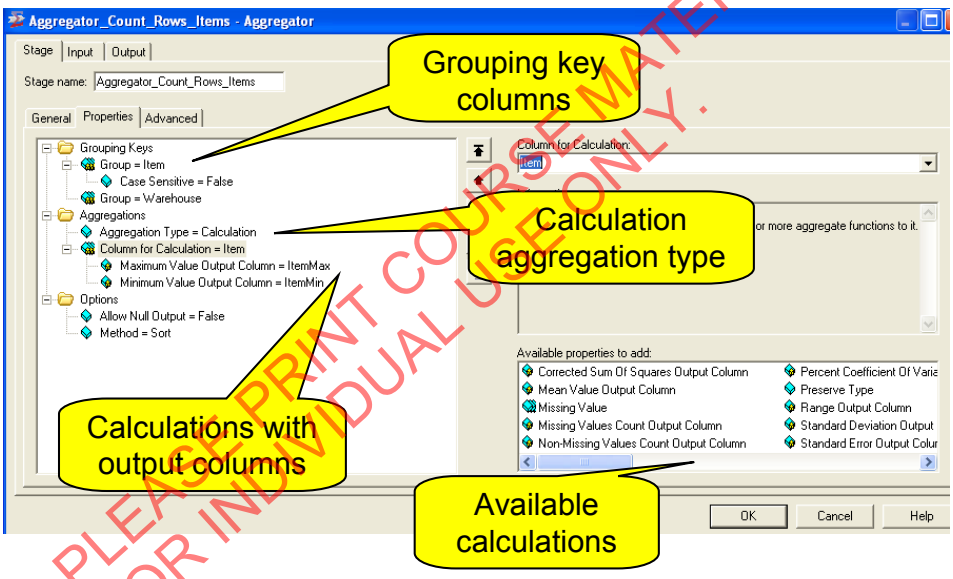







15

# Calculation Type Aggregator Properties








16

## Grouping Methods

- Hash (default)
  - Calculations are made for all groups and stored in memory
    - Hash table structure (hence the name)
  - Results are written out after all input has been processed
  - Input does not need to be sorted
  - Useful when the number of unique groups is small
    - Running tally for each group's aggregations needs to fit into memory
- Sort
  - Requires the input data to be sorted by grouping keys
    - Does not perform the sort! Expects the sort
  - Only a single aggregation group is kept in memory
    - When a new group is seen, the current group is written out
  - Can handle unlimited numbers of groups





17

## Remove Duplicates Stage





18

## Removing Duplicates

- Can be done by Sort stage
  - Use unique option
    - No choice on which duplicate to keep
    - Stable sort always retains the first row in the group
    - Non-stable sort is indeterminate

**OR**

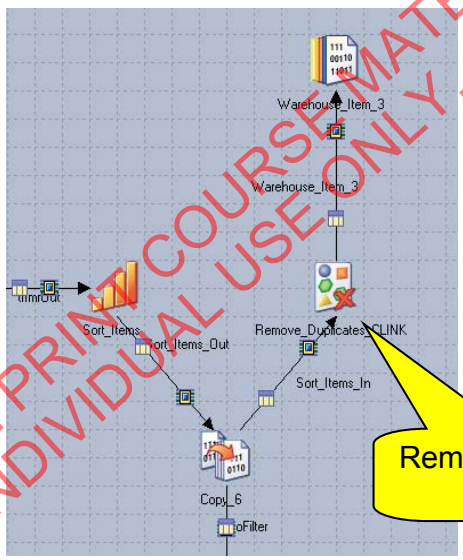
- Remove Duplicates stage
  - Has more sophisticated ways to remove duplicates
    - Can choose to retain first or last





19

## Remove Duplicates Stage Job

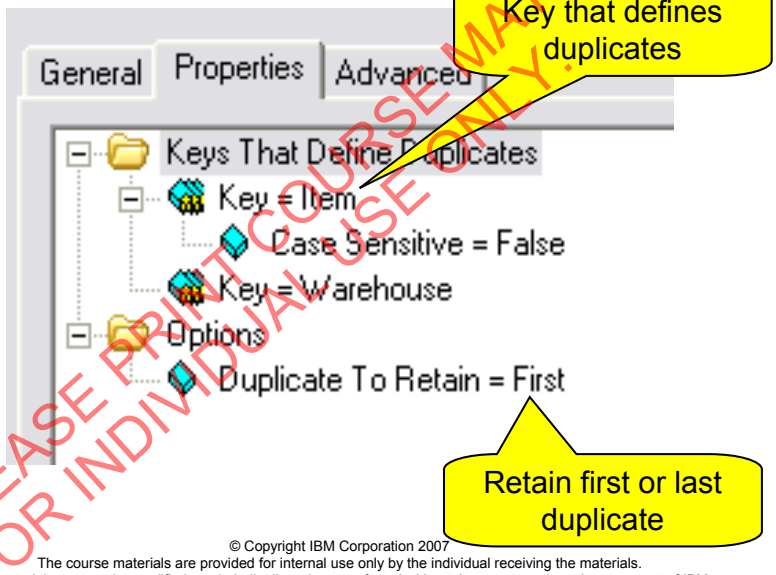







20

## Remove Duplicates Stage Properties








21

## Checkpoint

1. What stage is used to perform calculations of column values grouped in specified ways?
2. In what two ways can sorts be performed?
3. What is a stable sort?





22

**Notes:**

Write down your answers here:

1.

2.

## Checkpoint solutions

1. Aggregator stage
2. Using the Sort stage. In-stage sorts.
3. Stable sort preserves the order of non-key values.





23

## Unit summary

Having completed this unit, you should be able to:

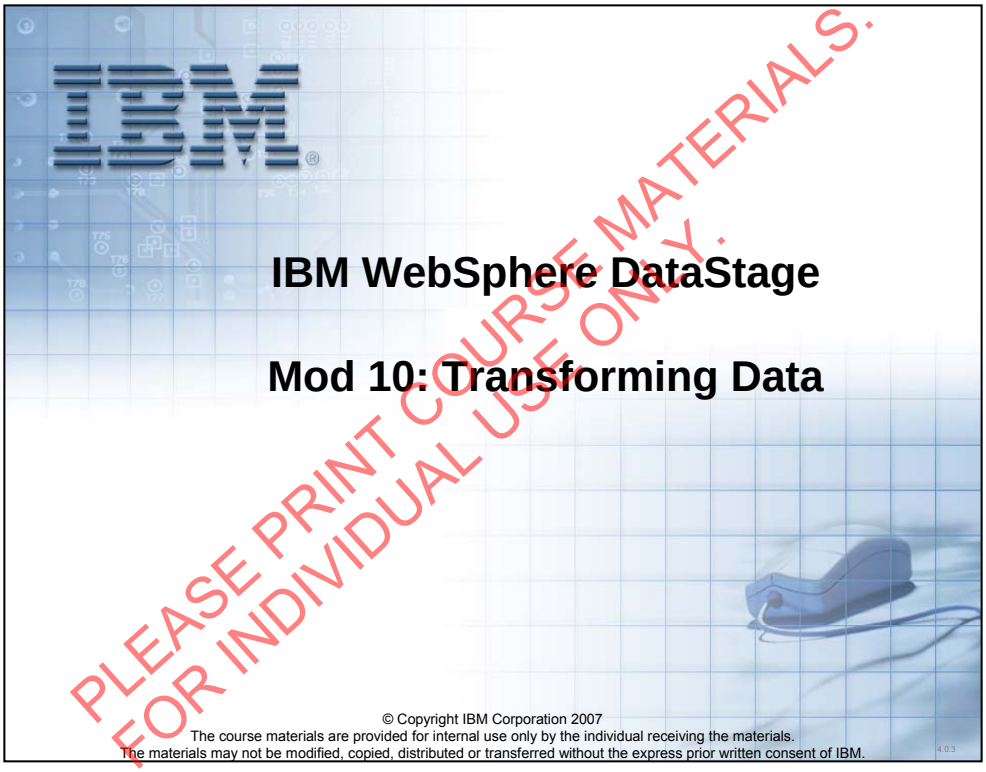
- Sort data using in-stage sorts and Sort stage
- Combine data using Aggregator stage
- Combine data Remove Duplicates stage





24

**Notes:**


IBM
IBM WebSphere DataStage
Mod 10: Transforming Data
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:

- Use the Transformer stage in parallel jobs
- Define constraints
- Define derivations
- Use stage variables
- Create a parameter set and use its parameters in constraints and derivations





26

**Notes:**

# Transformer Stage





27

## Transformer Stage

- Column mappings
- Derivations
  - Written in Basic
  - Final compiled code is C++ generated object code
- Constraints
  - Filter data
  - Direct data down different output links
    - For different processing or storage
- Expressions for constraints and derivations can reference
  - Input columns
  - Job parameters
  - Functions
  - System variables and constants
  - Stage variables
  - External routines

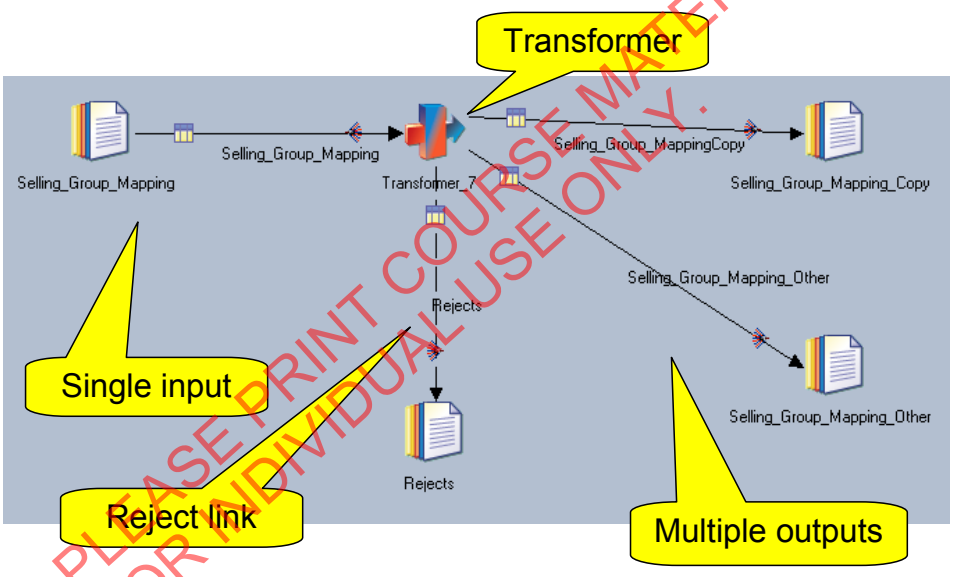




28

In this module, we'll discuss two stages that can be used to perform derivations to implement business logic. Our main focus in on the Transformer stage. We'll also briefly discuss the Modify stage.

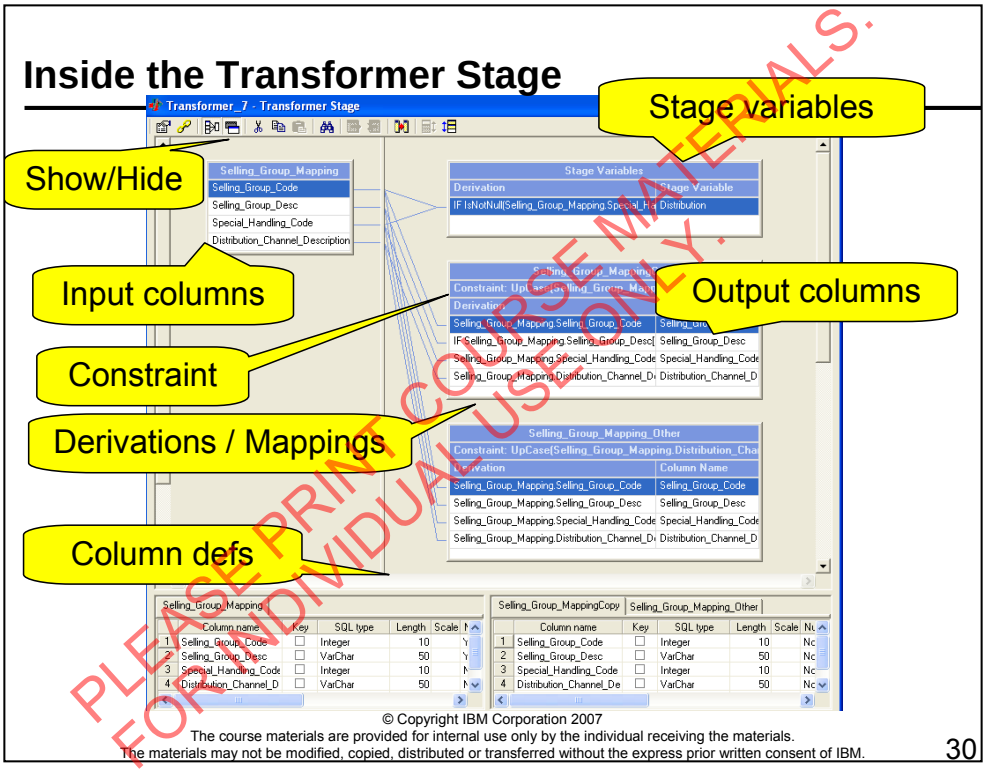
## Job With a Transformer Stage








29






Link naming conventions are important because they identify appropriate links in the stage properties screen shown above.

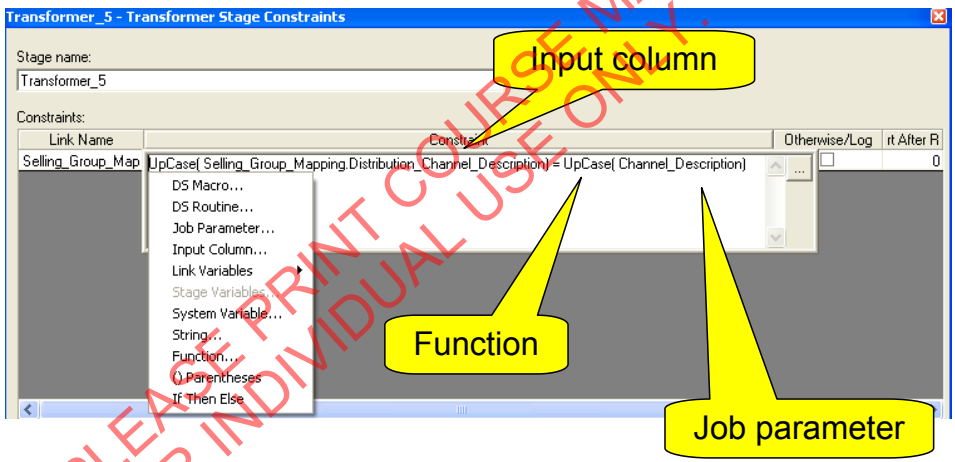
Four quadrants:

- Incoming data link (one only)
- Outgoing links (can have multiple)
- Meta data for all incoming links
- Meta data for all outgoing links – may have multiple tabs if there are multiple outgoing links

Note the constraints bar – if you double-click on any you will get screen for defining constraints for all outgoing links.

Double-click to the left of an output column or stage variable to define its derivation.

## Defining a Constraint



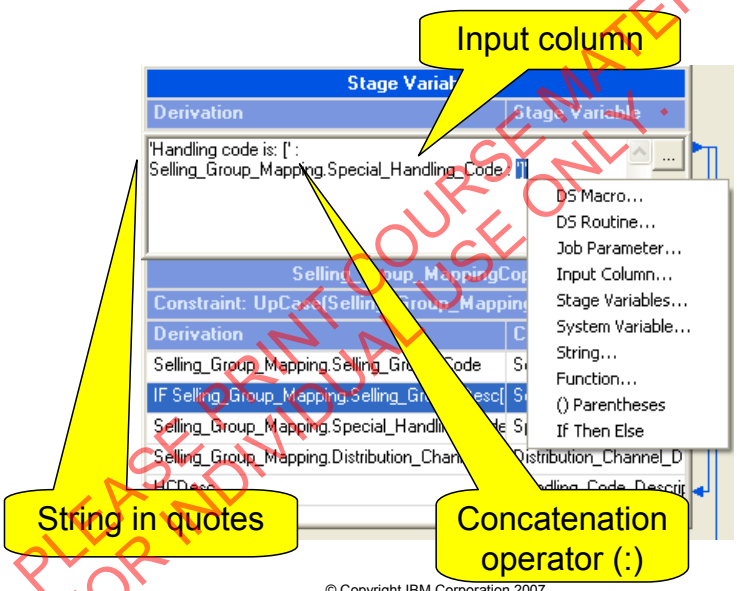





31

Click the Constraints icon at the top of the Transformer (yellow chain) to open the Constraints window. Select items from the menu or type in items to build the expression.

## Defining a Derivation








32

## IF THEN ELSE Derivation

- Use IF THEN ELSE to conditionally derive a value
- Format:
  - IF <condition> THEN <expression1> ELSE <expression1>
  - If the condition evaluates to true then the result of expression1 will be copied to the target column or stage variable
  - If the condition evaluates to false then the result of expression2 will be copied to the target column or stage variable
- Example:
  - Suppose the source column is named In.OrderID and the target column is named Out.OrderID
  - Replace In.OrderID values of 3000 by 4000
  - IF In.OrderID = 3000 THEN 4000 ELSE Out.OrderID





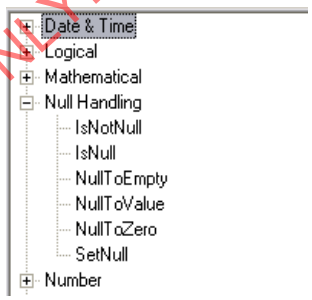
33

## String Functions and Operators

- Substring operator
  - Format: "String" [loc, length]
  - Example:
    - Suppose In.Description contains the string "Orange Juice"
    - InDescription[8,5] → "Juice"
- UpCase(<string>) / DownCase(<string>)
  - Example: UpCase(In.Description) → "ORANGE JUICE"
- Len(<string>)
  - Example: Len(In.Description) → 12

## Checking for NULLs

- Nulls can be introduced into the data flow from lookups
  - Mismatches (lookup failures) can produce nulls
- Can be handled in constraints, derivations, stage variables, or a combination of these
- NULL functions
  - Testing for NULL
    - IsNull(<column>)
    - IsNotNull(<column>)
  - Replace NULL with a value
    - NullToValue(<column>, <value>)
  - Set to NULL: SetNull()
    - Example: IF In.Col = 5 THEN SetNull() ELSE In.Col








35

If you perform a lookup from a lookup stage and choose the continue option for a failed lookup, you have the possibility of nulls entering your data flow.

Important: Be sure to use the NULL functions in stage variable derivations rather than output column derivations. Reference the stage variable in column derivations rather than performing the NULL function in the column derivation. The reason for this is that records containing NULLs are rejected BEFORE column derivations are performed (but not before stage variable derivations are performed).

## Transformer Functions

- Date & Time
- Logical
- Null Handling
- Number
- String
- Type Conversion





36

## Transformer Execution Order

- Derivations in stage variables
- Constraints are executed before derivations
- Column derivations in earlier links are executed before later links
- Derivations in higher columns are executed before lower columns



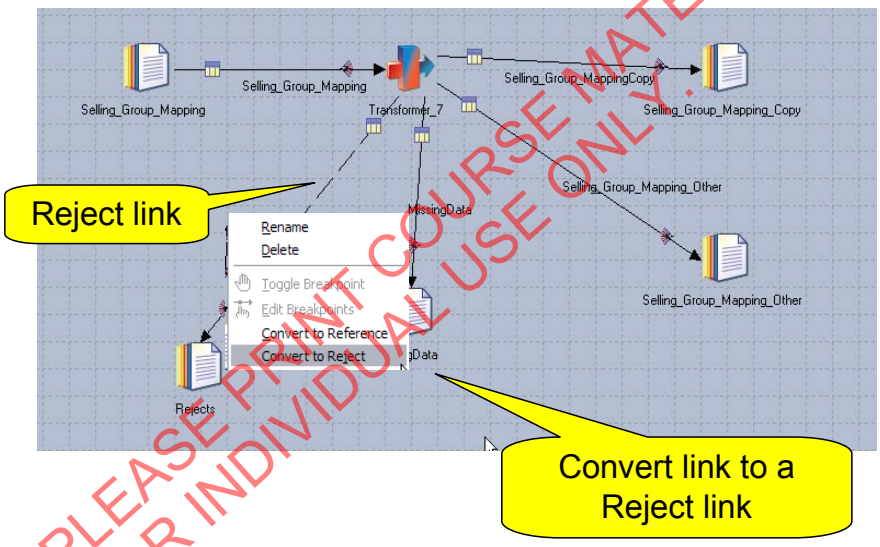


37

## Transformer Stage Variables

- Derivations execute in order from top to bottom
  - Later stage variables can reference earlier stage variables
  - Earlier stage variables can reference later stage variables
    - These variables will contain a value derived from the previous row that came into the Transformer
- Multi-purpose
  - Counters
  - Store values from previously read rows to make comparisons with the currently read row
  - Store derived values to be used in multiple target field derivations
  - Can be used to control execution of constraints

## Transformer Reject Links



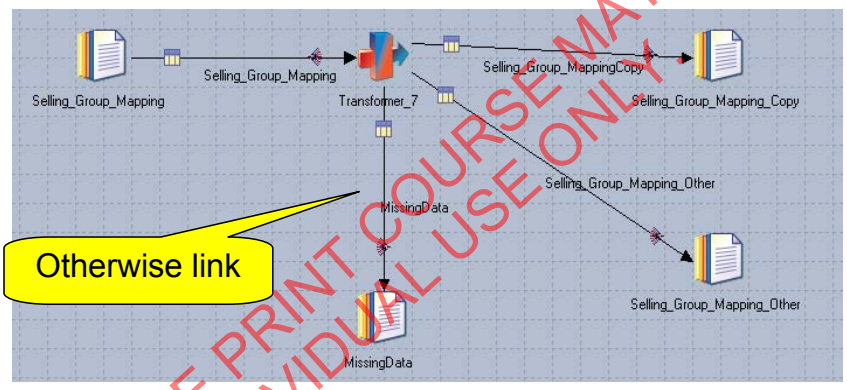





39

Rows are sent to the Reject link if a nullable column is used in a calculation or derivation and that column has a NULL value. It is important to test for and replace NULLs in any derivations that reference nullable input columns. Position these tests for and replaces of NULLs within stage variable derivations.

## Otherwise Link



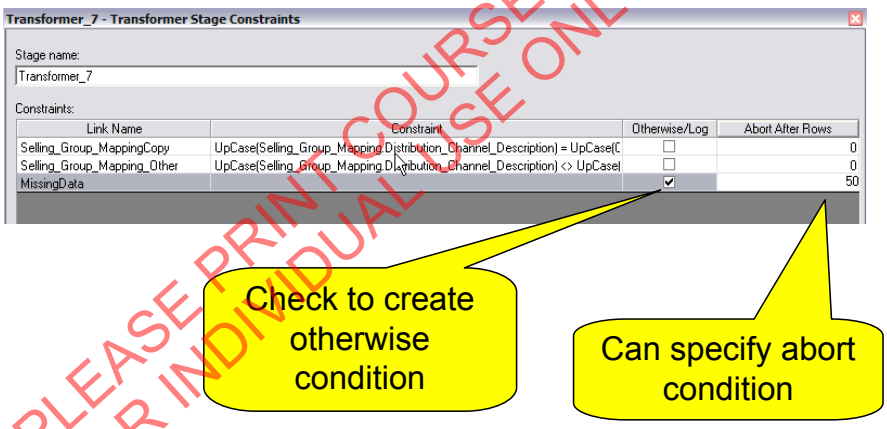





40

The otherwise link captures what doesn't go down any previous links. It needs to come after the other links in the link ordering. It's defined on the Constraints tab, as shown in the next slide.

## Defining an Otherwise Link







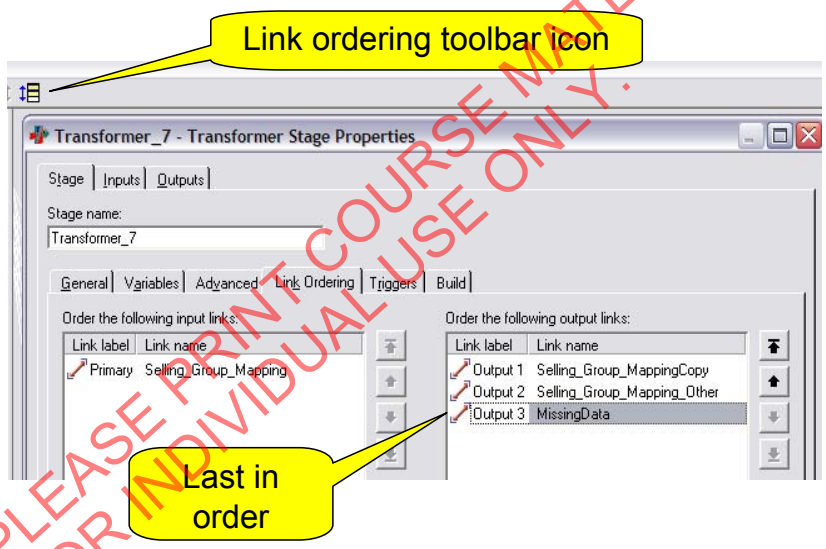

41

The Otherwise link must come after the other links carrying the main constraints. You can change the ordering of links, as shown on the next slide.

A warning is put in the log if any rows go down the otherwise link.

You can specify an abort condition for any output link. The abort occurs after the specified number of rows occurs in one of the partitions. It is not based on the total number of rows, but on the number of rows in a single partition.

## Specifying Link Ordering








42

The Link Ordering icon is at the last icon on the toolbar.

# Parameter Sets



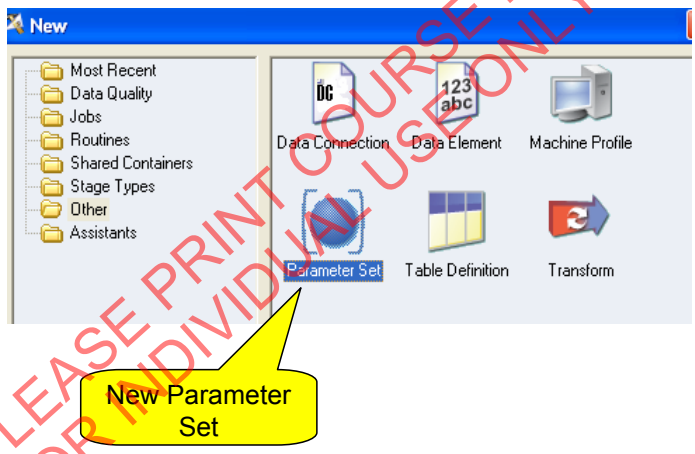


43

## Parameter Sets

- Store a collection of parameters in a named object
- One or more values files can be named and specified
  - A values file stores values for specified parameters
  - Values are picked up at runtime
- Parameter Sets can be added to the job parameters specified on the Parameters tab in the job properties

## Creating a New Parameter Set



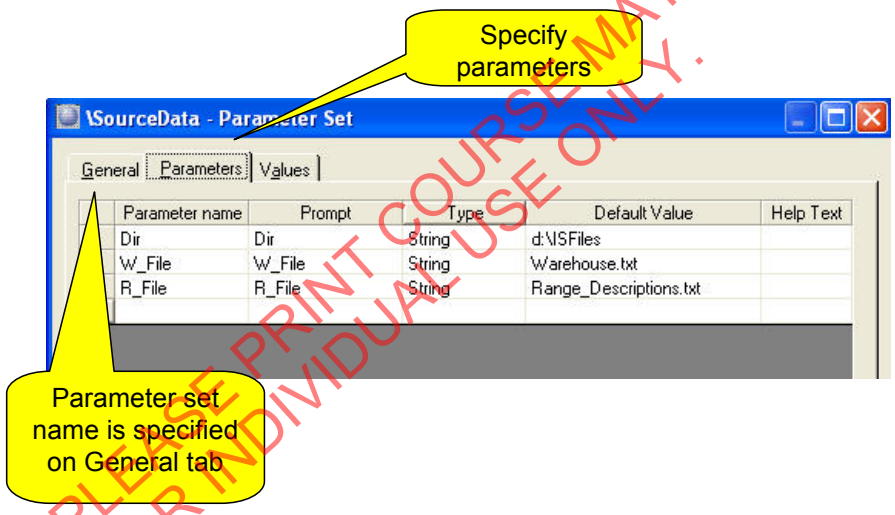





45

Click New and then select the Other folder.

## Parameters Tab

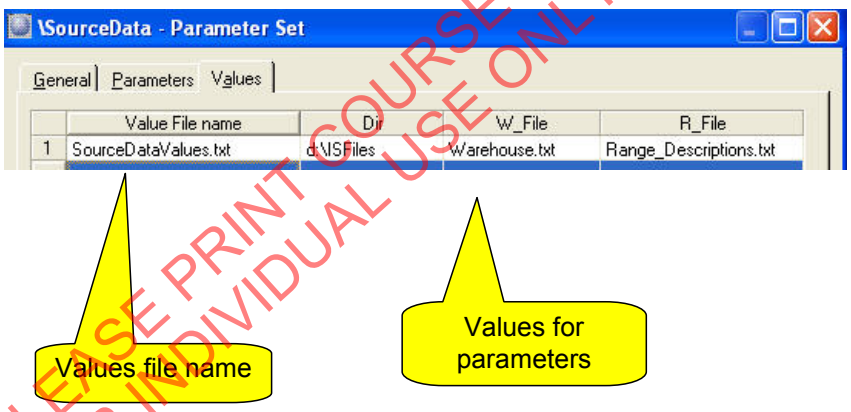







46

## Values Tab



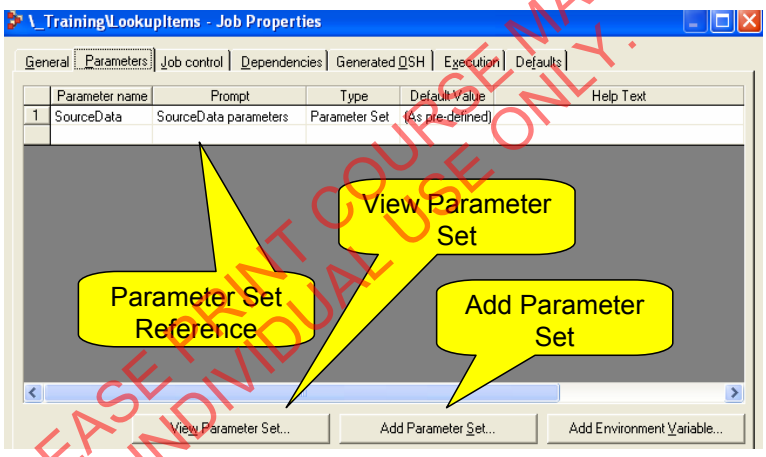



47

Parameter Set values files are located in the …\ParameterSets\Project\ directory of the Information Server install directory.

You can specify more than one value file with different sets of values.

## Adding a Parameter Set to Job Properties

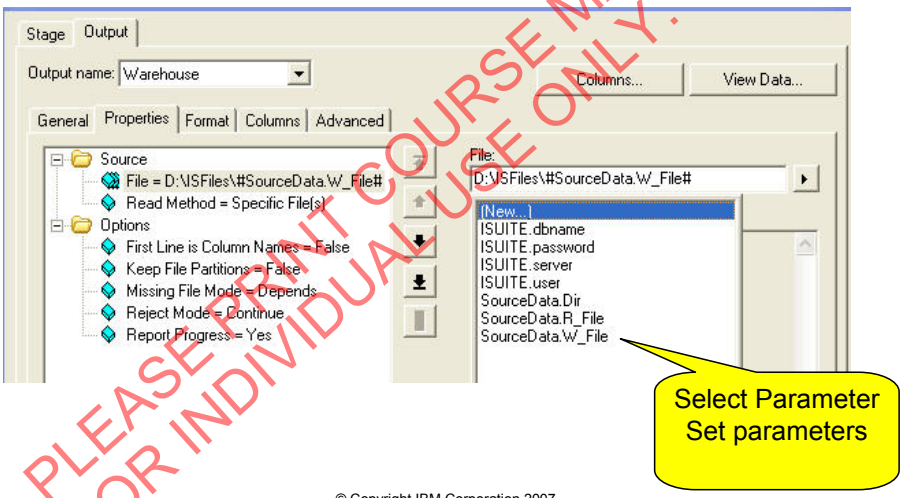







48

## Using Parameter Set Parameters








49

Notice that Parameter Set parameters are qualified by the name of the Parameter Set.

## Checkpoint

1. What occurs first? Derivations or constraints?
2. Can stage variables be referenced in constraints?
3. Where should you test for NULLS within a Transformer? Stage Variable derivations or output column derivations?





50

**Notes:**

Write down your answers here:

1.

2.

## Checkpoint solutions

1. Constraints
2. Yes
3. Stage variable derivations. Reference the stage variables in the output column derivations.





51

## Unit summary

Having completed this unit, you should be able to:

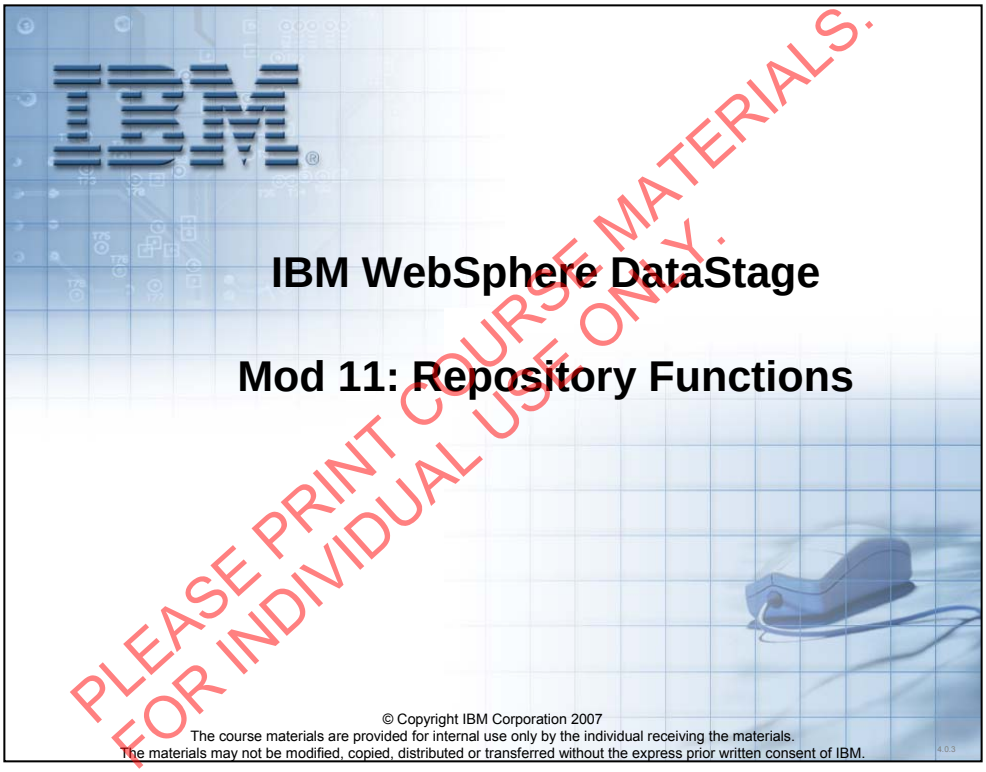
- Use the Transformer stage in parallel jobs
- Define constraints
- Define derivations
- Use stage variables
- Create a parameter set and use its parameters in constraints and derivations





52

**Notes:**


IBM
IBM WebSphere DataStage
Mod 11: Repository Functions
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:

- Perform a simple Find
- Perform an Advanced Find
- Perform an impact analysis
- Compare the differences between two Table Definitions
- Compare the differences between two jobs





54

**Notes:**

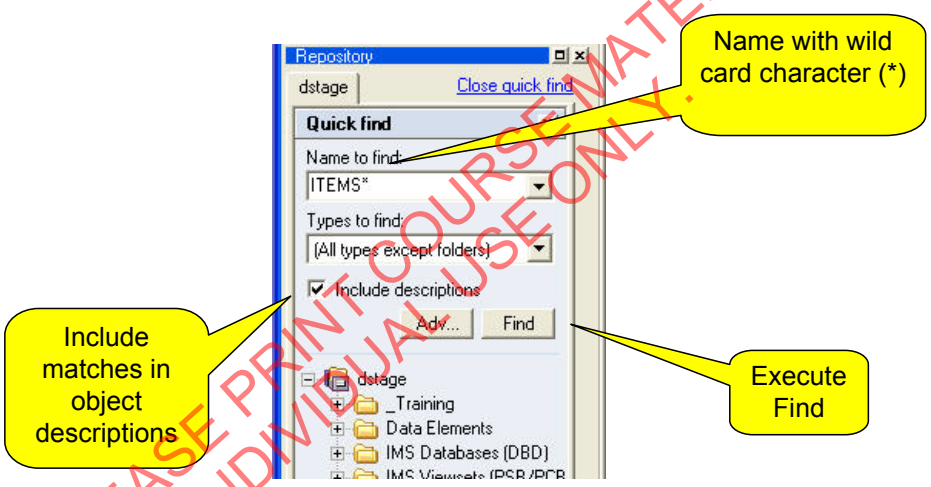
## Searching the Repository





55

## Quick Find



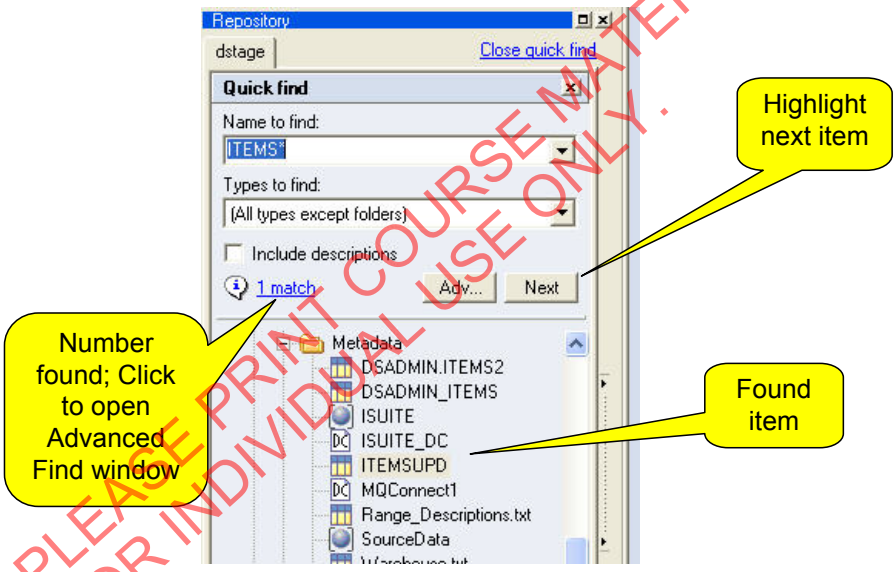





56

If the Include descriptions box is checked, the text in Short descriptions and Long descriptions will be searched.

## Found Results

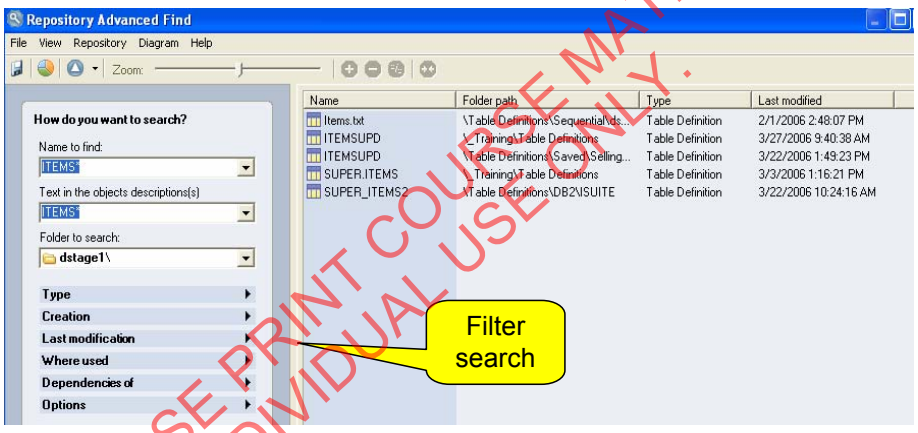







57

## Advanced Find Window








58

## Advanced Find Filtering Options

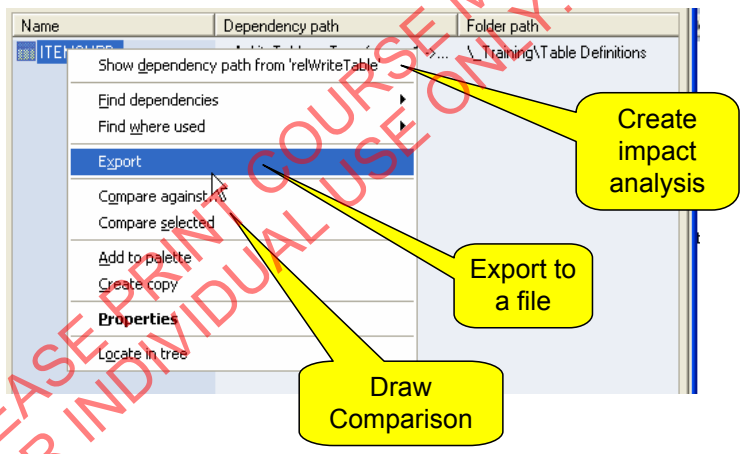
- Type: type of object
  - Job, Table Definition, etc.
- Creation: range of dates
  - E.g., Up to a week ago
- Last modification: range of dates
  - E.g., Up to a week ago
- Where used: objects that use specified objects
  - E.g., a job that uses a specified Table Definition
- Dependencies of: objects that are dependencies of objects
  - E.g., a Table Definition that is referenced in a specified job
- Options
  - Case sensitivity
  - Search within last result set





59

## Using the Found Results








60

# Impact Analysis





61

## Performing an Impact Analysis

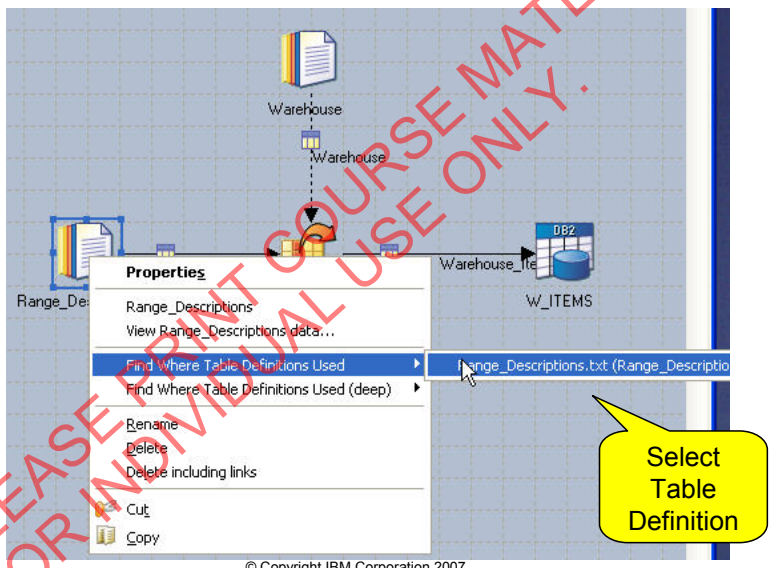
- Find where Table Definitions are used
  - Right-click over a stage or Table Definition
  - Select “Find where Table Definitions Used” or
  - Select “Find where Table Definitions Used (deep)”
    - Deep includes additional object types
  - Displays a list of the objects using the Table Definition
- Find object dependencies
  - Select “Find dependencies” or
  - Select “Find dependencies (deep)”
  - Displays list of objects dependent on the one selected
- Graphical functionality
  - Display the dependency path
  - Collapse selected objects
  - Move the graphical object
  - “Birds-eye” view





62

## Initiating an Impact Analysis from a Stage

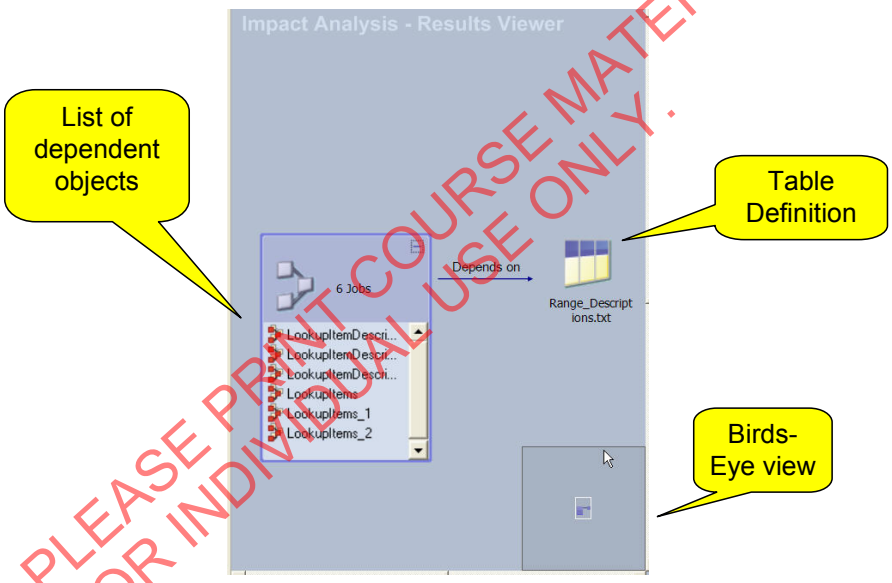







63

## Displaying the Dependencies Graphically

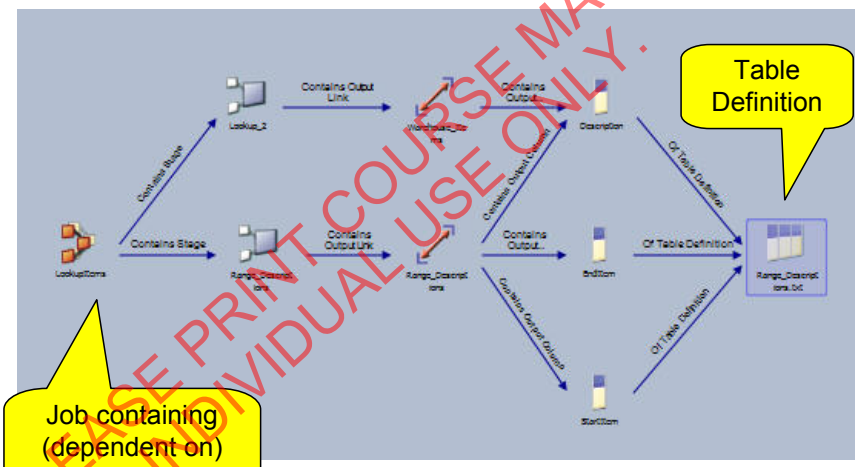







64

## Displaying the Dependency Path

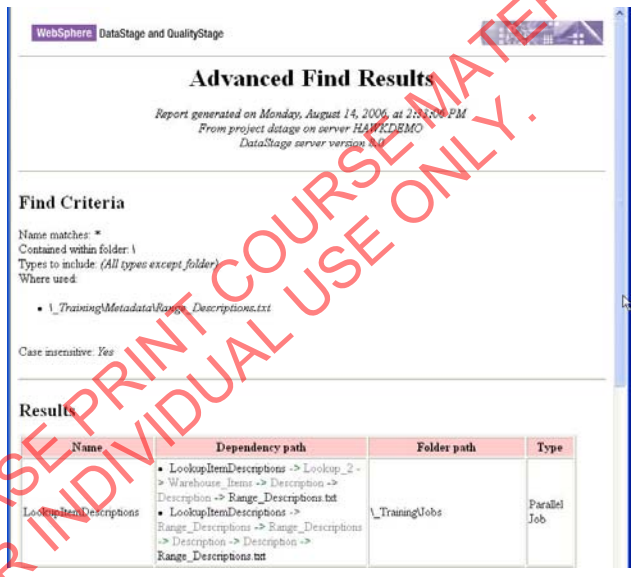







65

## Generating an HTML Report



WebSphere DataStage and QualityStage

**Advanced Find Results**

*Report generated on Monday, August 14, 2006, at 2:14:09 PM*
*From project dstage on server HAWKDEMO*
*DataStage server version 7.0*

**Find Criteria**

Name matches: *
Contained within folder: \
Types to include: *(All types except folder)*
Where used:

- *_Training\Metadata\Range_Descriptions.txt*

Case insensitive: *Yes*

**Results**

| Name | Dependency path | Folder path | Type |
|---|---|---|---|
| LookupItemDescriptions | • LookupItemDescriptions -> Lookup_2 -> Warehouse_Items -> Description -> Description -> Range_Descriptions.txt<br>• LookupItemDescriptions -> Range_Descriptions -> Range_Descriptions -> Description -> Description -> Range_Descriptions.txt | _Training\Jobs | Parallel Job |

# Job and Table Difference Reports

## Finding the Difference Between Two Jobs

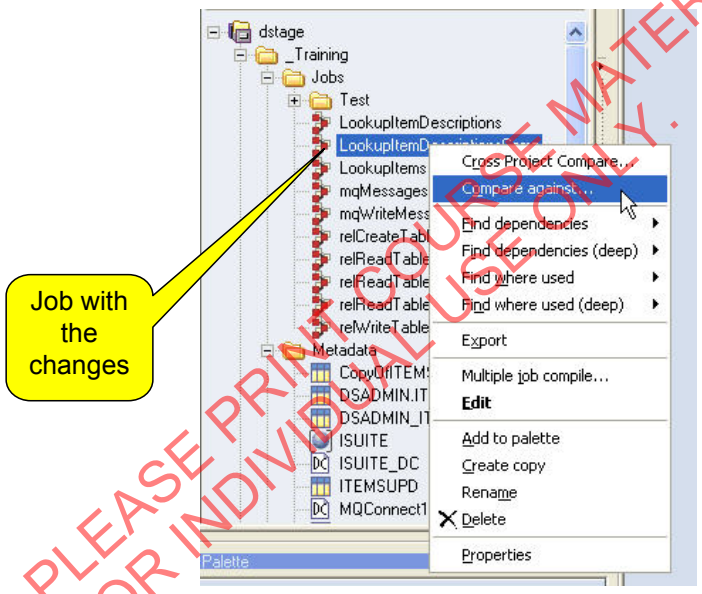
- Example: Job1 is saved as Job2. Changes are made to Job2. What changes have been made?
  - Here Job1 may be a production job. Job2 is a copy of the production job after enhancements or other changes have been made to it





68

## Initiating the Comparison








69

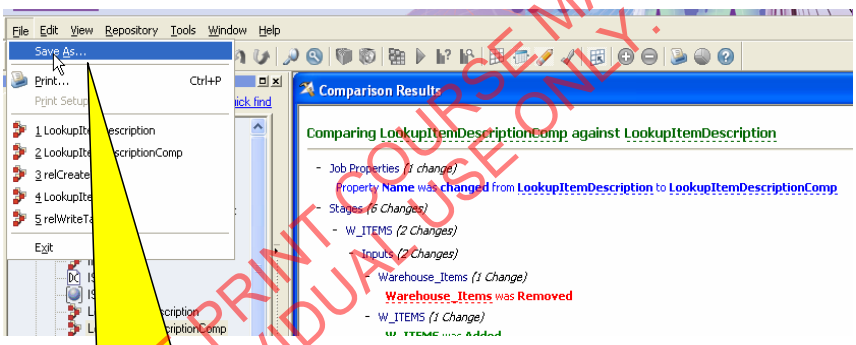
# Comparison Results






70

## Saving to an HTML File




Click when Comparison window is active

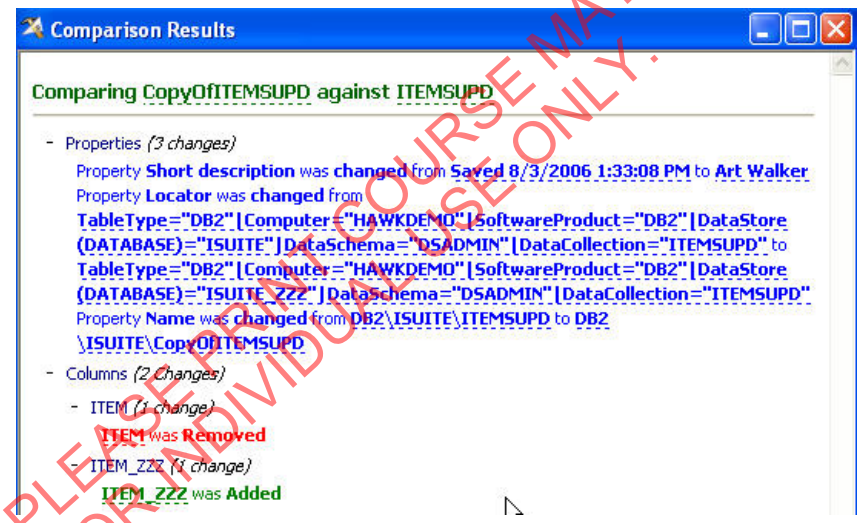




71

## Comparing Table Definitions

- Same procedure as when comparing jobs








72

## Checkpoint

1. You can compare the differences between what two kinds of objects?
2. What “wild card” characters can be used in a Find?
3. You have a job whose name begins with “abc”. You can’t remember the rest of the name or where the job is located. What would be the fastest way to export the job to a file?
4. Name three filters you can use in a Advanced Find?





73

**Notes:**

Write down your answers here:

1.

2.

## Checkpoint solutions

1. Jobs.  Table Definitions.
2. Asterisk (*). It stands for any zero or more characters.
3. Do a Find for objects matching “abc*”.  Filter by type job.  Locate the job in the result set, click the right mouse button over it, and then click Export.
4. Type of object, creation date range, last modified date range, where used, dependencies of, other options including case sensitivity and search within last result set.

## Unit summary

Having completed this unit, you should be able to:

- Perform a simple Find
- Perform an Advanced Find
- Perform an impact analysis
- Compare the differences between two Table Definitions
- Compare the differences between two jobs

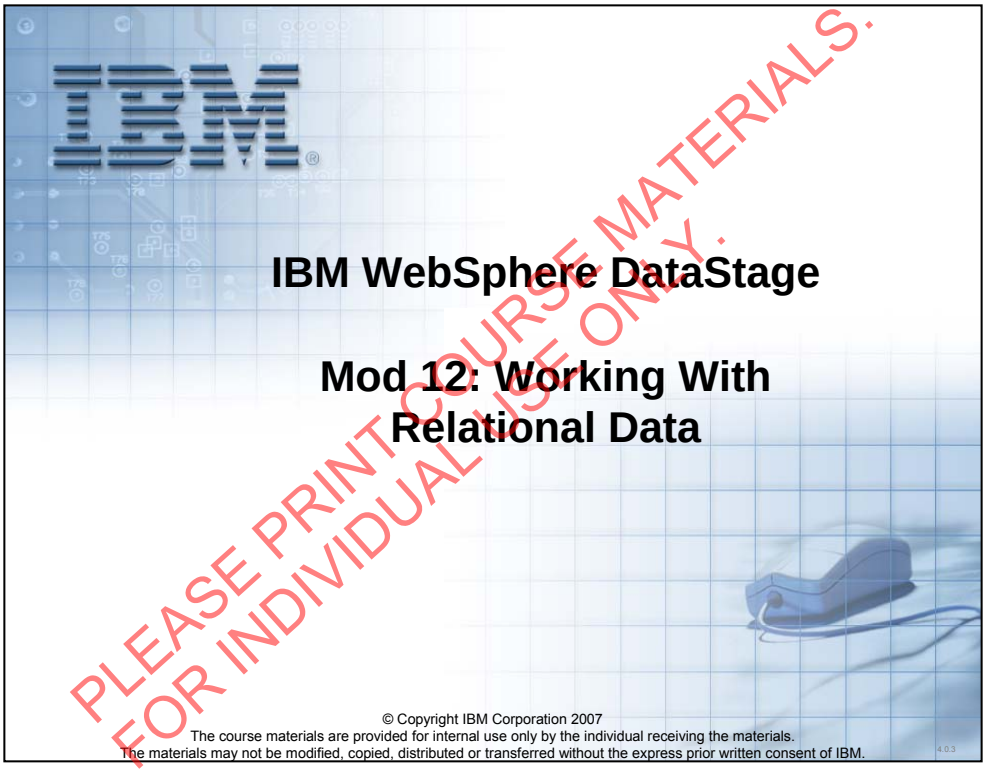




75

**Notes:**

76


IBM
IBM WebSphere DataStage
Mod 12: Working With Relational Data
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:
- Import Table Definitions for relational tables
- Create Data Connections
- Use Connector stages in a job
- Use SQL Builder to define SQL Select statements
- Use SQL Builder to define SQL Insert and Update statements
- Use the DB2 Enterprise stage





78

**Notes:**

## Working with Relational Data

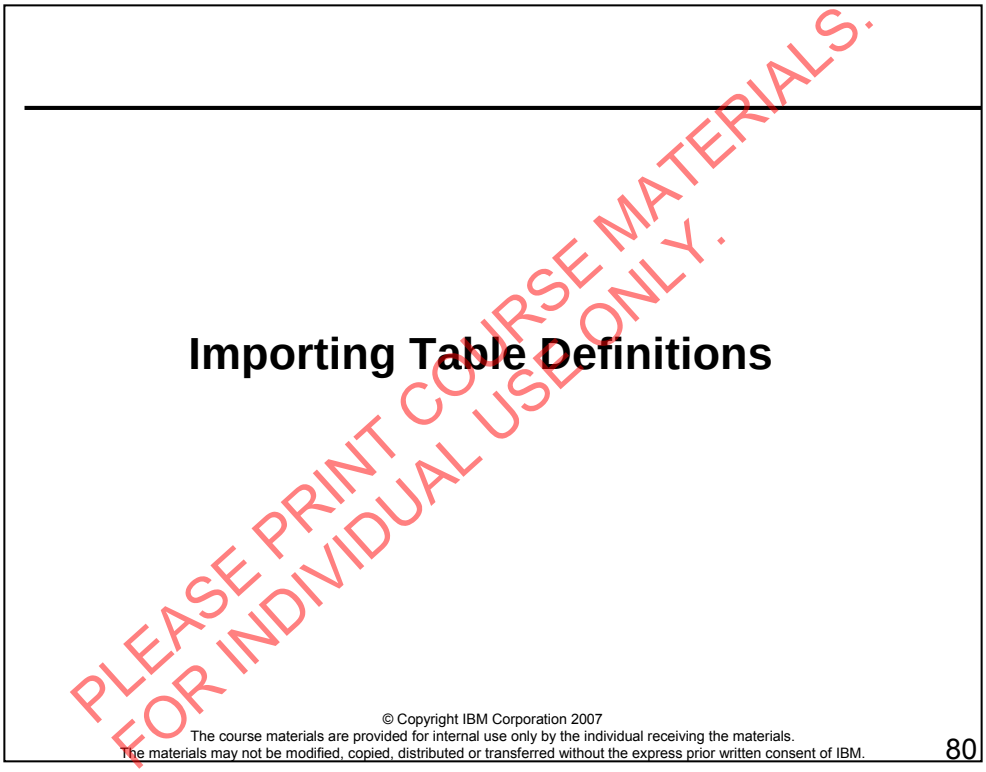
- Importing relational data
  - Import using ODBC or orchdbutil
    - orchdbutil is preferred, in order to get correct type conversions
- Data Connection objects
  - Store database connection information in a named object
- Stages available to access relational data
  - Connector stages
    - Parallel support
    - Most functionality
    - Consistent GUI and functionality across all relational types
  - Enterprise stages
    - Parallel support
  - Plug-in stages
    - Functionality ported from DataStage Server Jobs
  - Selecting data
  - Build SELECT statements using SQL Builder
- Writing data
  - Build INSERT, UPDATE, DELETE statements using SQL Builder





79

Our primary focus is on the Connector stages, but the DB2 Enterprise stage is also discussed.


Importing Table Definitions
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
80

## Importing Table Definitions

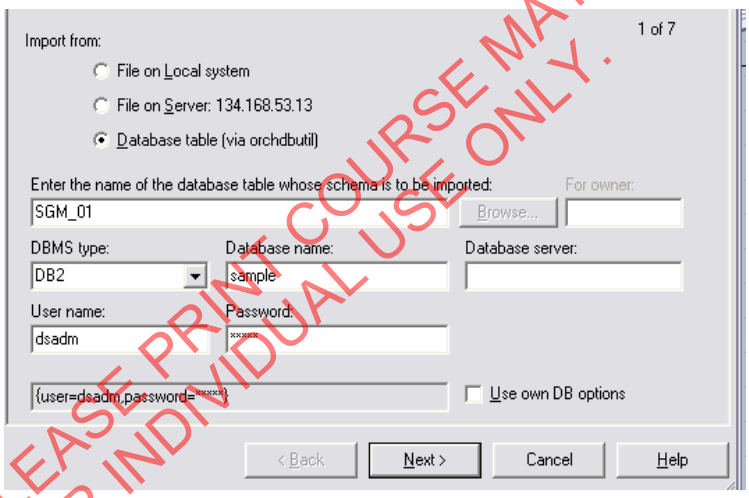
- Can import using ODBC or using Orchestrate schema definitions
  - Orchestrate schema imports are better because the data types are more accurate
- Import>Table Definitions>Orchestrate Schema Definitions
- Import>Table Definitions>ODBC Table Definitions



81

The orchdbutil GUI is limited to one table at a time. However, this utility is also available as a command-line utility that can be scripted to import a large number of Table Definitions.

# Orchestrate Schema Import

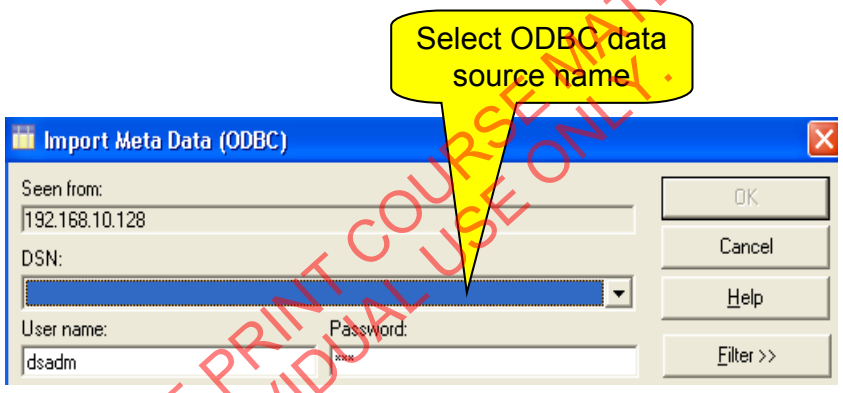







82

## ODBC Import








83

The ODBC data source that accesses the database containing the tables to be imported must have been previously defined.

## Connectors Stages





84

## Connector Stage Types

- ODBC
  - Conforms to ODBC 3.5 standard and is Level 3 compliant
  - Certified with Oracle, DB2 UDB, SQL Server, and others
  - DataDirect drivers
- DB2 UDB
  - 8.1 and 8.2
- WebSphere MQ
  - WSMQ 5.3 and 6.0 for Client / Sever
  - WSMB 5.0
- Teradata





85

## Connector Stage Features

- Common stage editor
- Convenient drop-down lists to choose properties
- Job parameters can be inserted into any property
- Required properties are identified with a visual indicator
- Warning indicator for properties requiring attention
- Metadata retrieval
- Integrated SQL Builder
- Parallel support
  - Read: parallel connections to the server and modified SQL queries for each connection
  - Write: parallel connections to the server
- Transaction isolation level support
  - Read Uncommitted
  - Read Committed
  - Repeatable Read
  - Serializable
- Before / After commands
  - Executed once before or after the job runs





86

## Metadata Retrieval

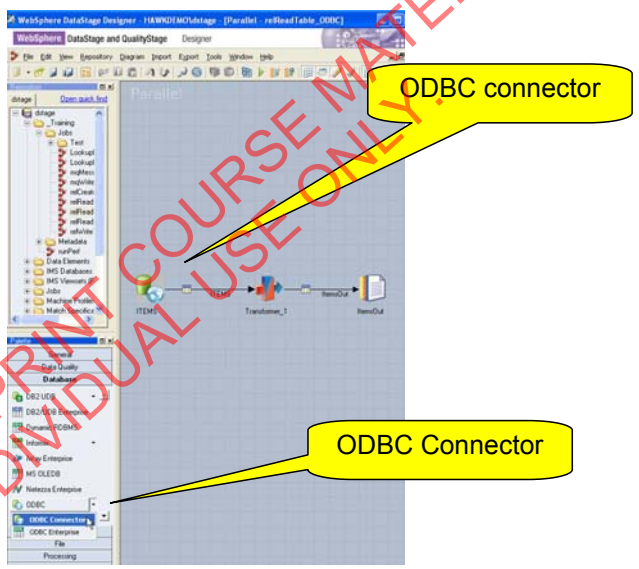
- List data sources
  - Can list both user and system DSNs
- List users defined for the given database
- Individual table or view metadata
  - Column data type, size, precision, etc.
- Database information
  - Name, vendor name, version number, etc.
- List of database schemas
- List tables that match the filter criteria
- List related tables
  - Those related by foreign key
  - Those whose foreign keys point to the primary key in the given table





87

## Connector Stages



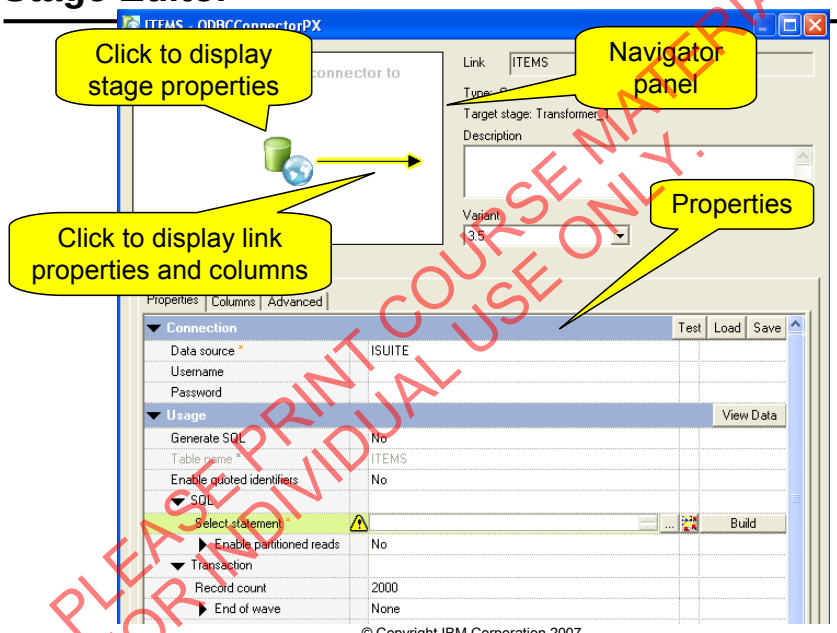





88

All the connector stages have a similar GUI and similar functionality. Here, the ODBC stage will be used for illustration.

## Stage Editor

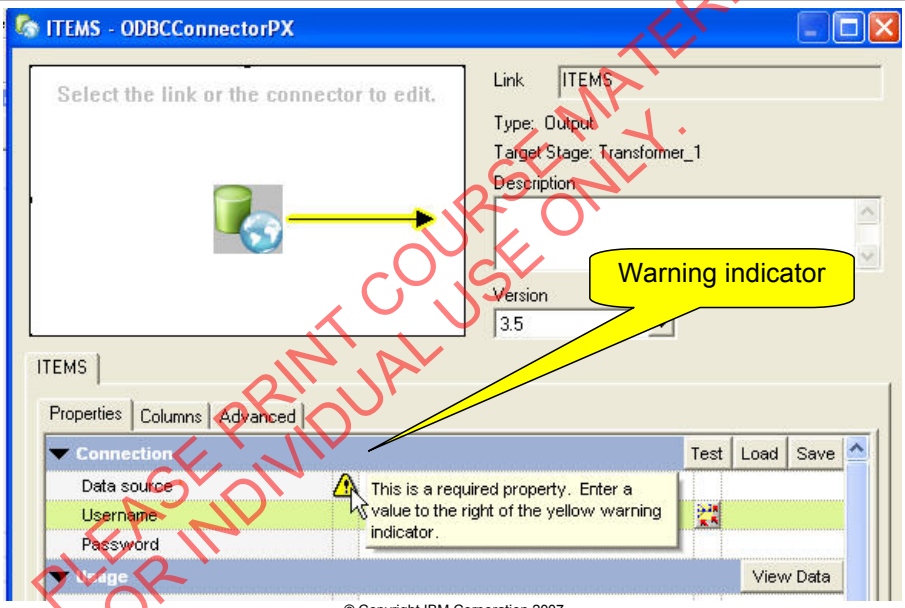







89

## Property Warning Indicators

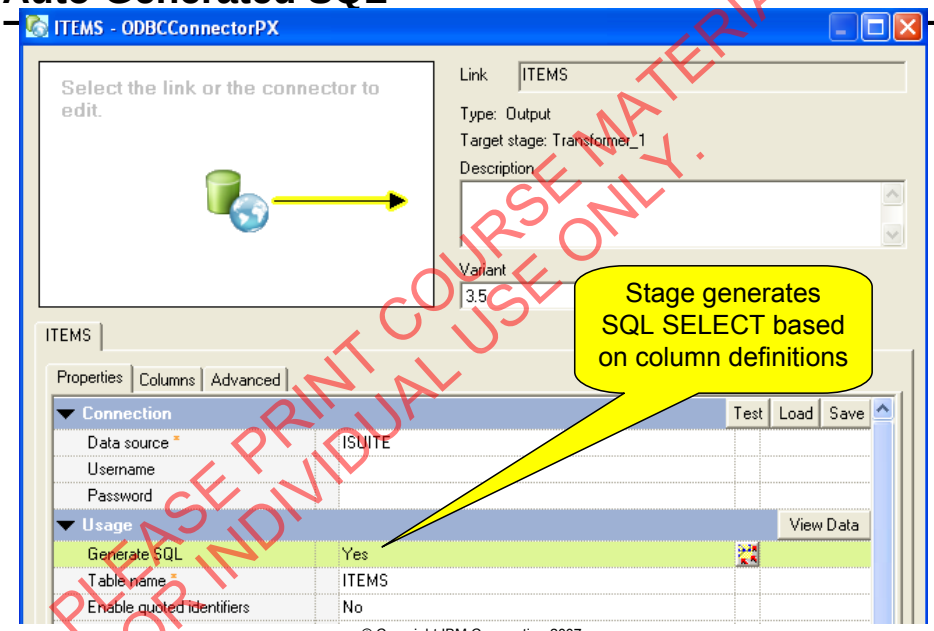







90

## Auto-Generated SQL

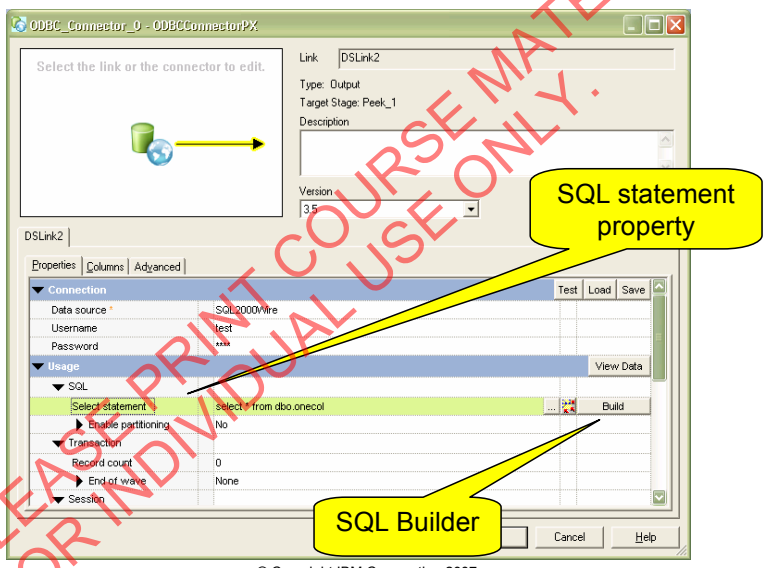







91

## Invoking SQL Builder

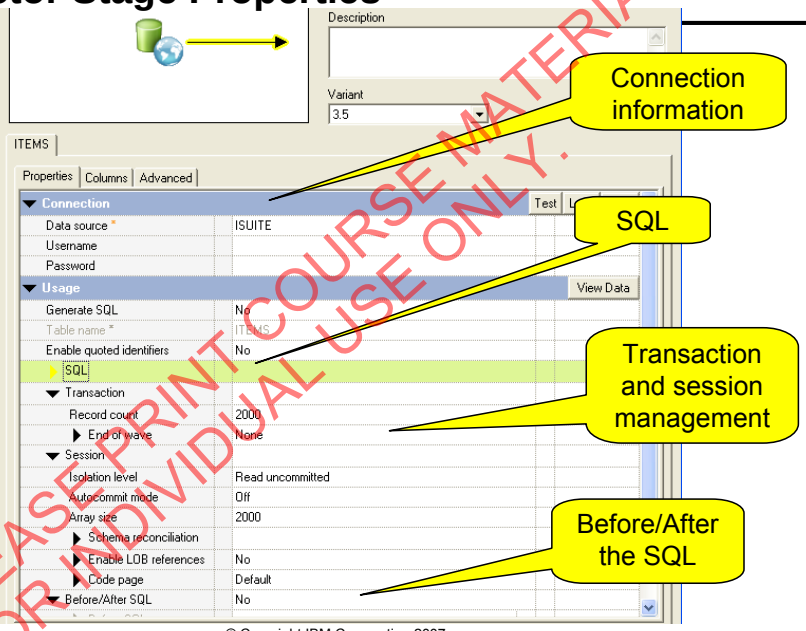







92

## Connector Stage Properties








93

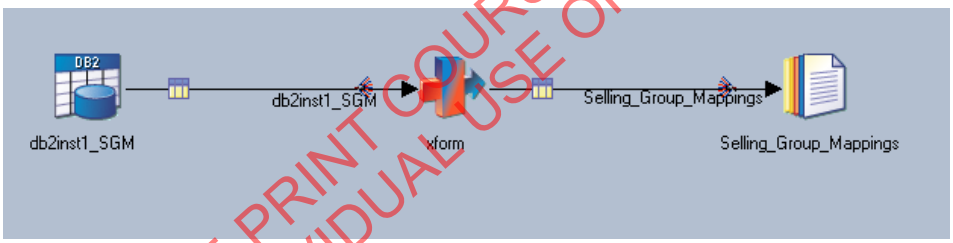
# DB2 Enterprise Stage





94

## DB2 Enterprise Source Stage

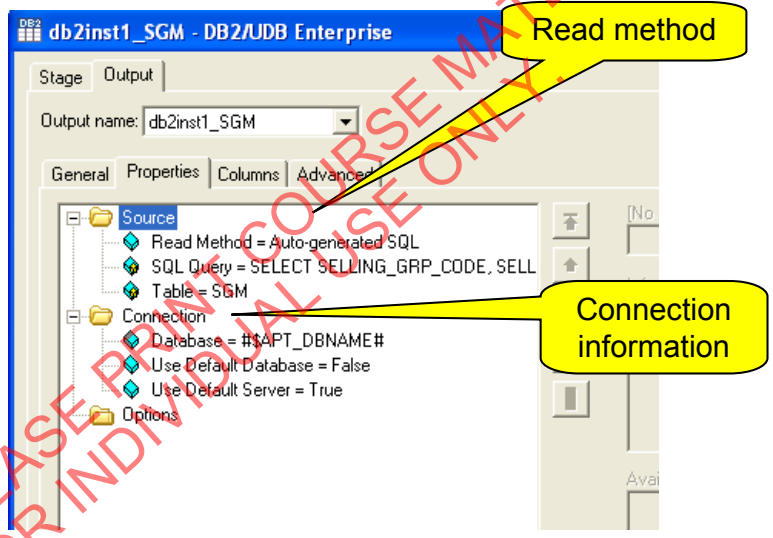







95

## DB2 Enterprise Source Stage Properties



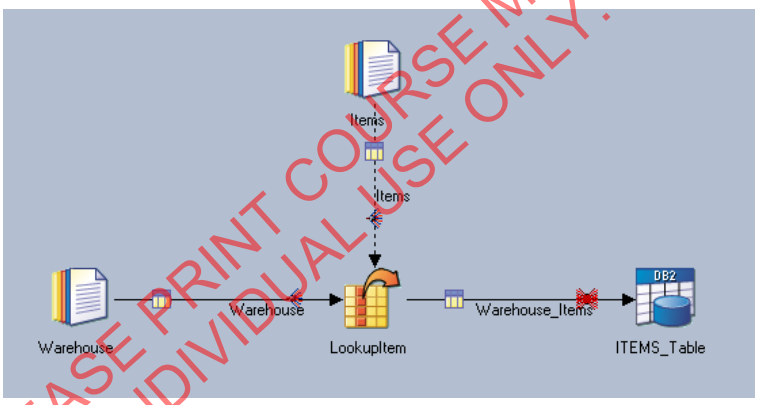





96

Read method includes: Auto-generated SQL, User-defined SQL, SQL Builder generated SQL, and Table. The Table method reads from the table without using an SQL query.

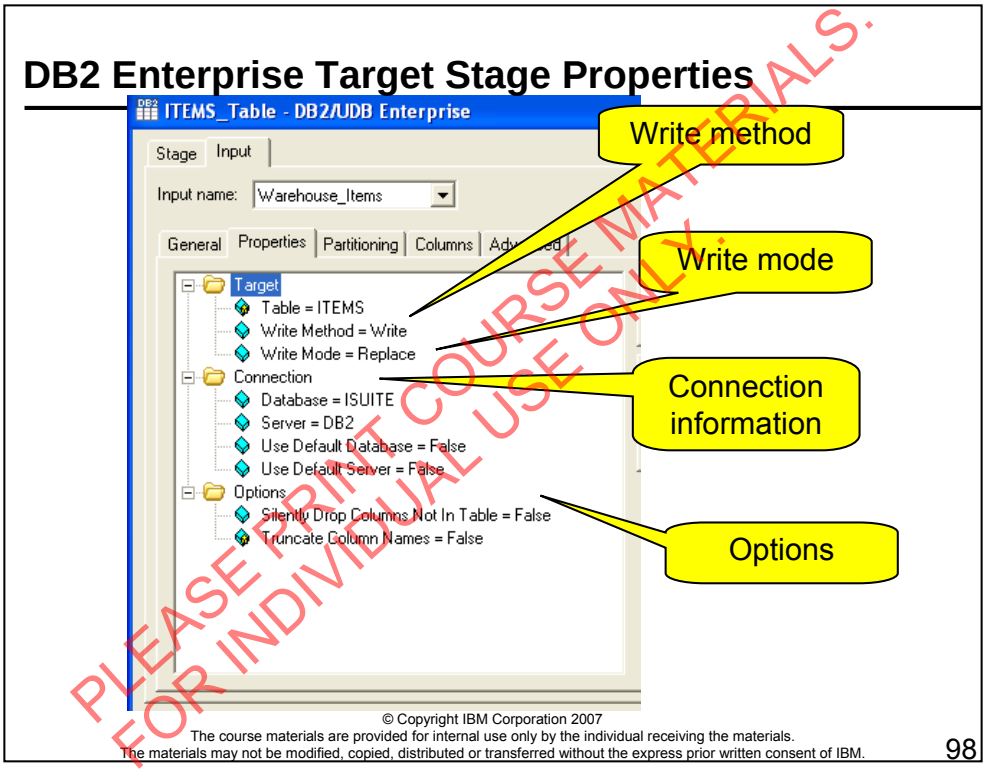
## DB2 Enterprise Target Stage








97




Write method includes: Write to INSERT; Load uses fast DB2 loader technology; Upsert uses an INSERT or an UPDATE SQL statement. Delete Rows uses a DELETE SQL statement.

Write mode includes: Append to existing table; Create new table; Replace by dropping table and creating new; Truncate table records only.

# Building a Query Using SQL Builder



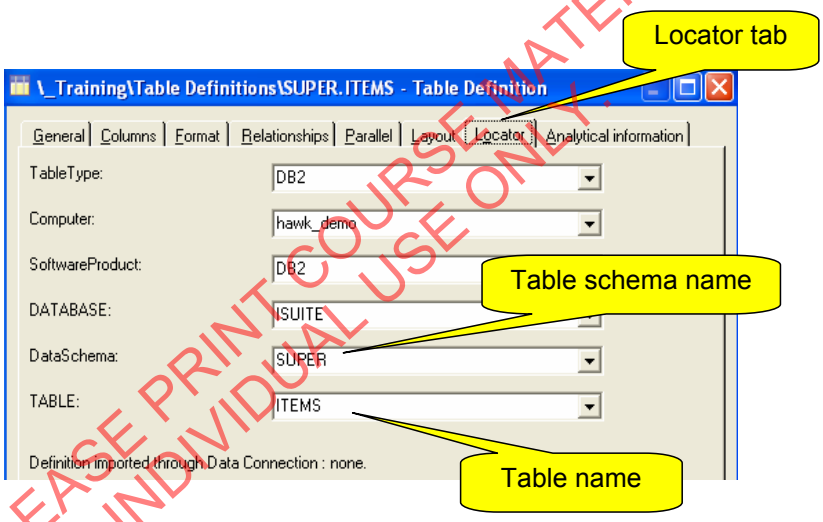


99

## SQL Builder

- Uses the Table Definition
  - Be sure the Locator tab information is specified fully and correctly
    - Schema and table names are based on Locator tab information
- Drag Table Definitions to SQL Builder canvas
- Drag columns from Table Definition to Select columns table

# Table Definition Locator Tab

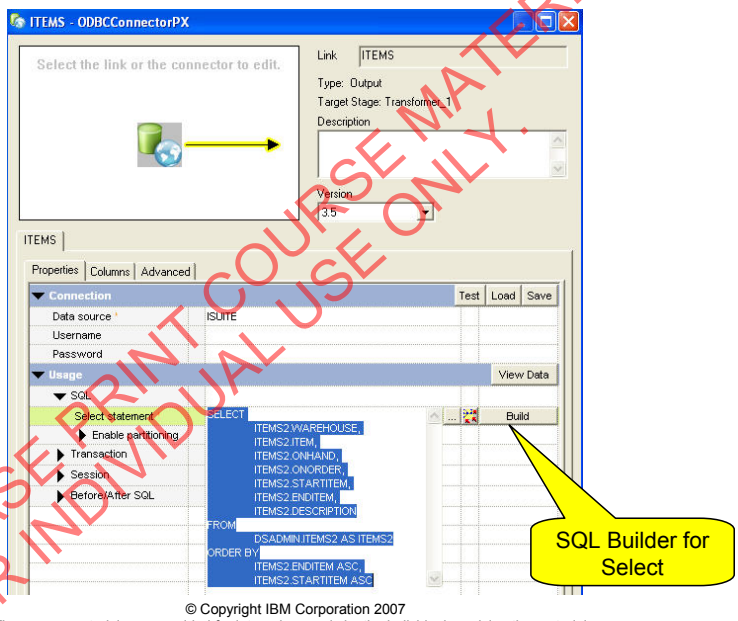




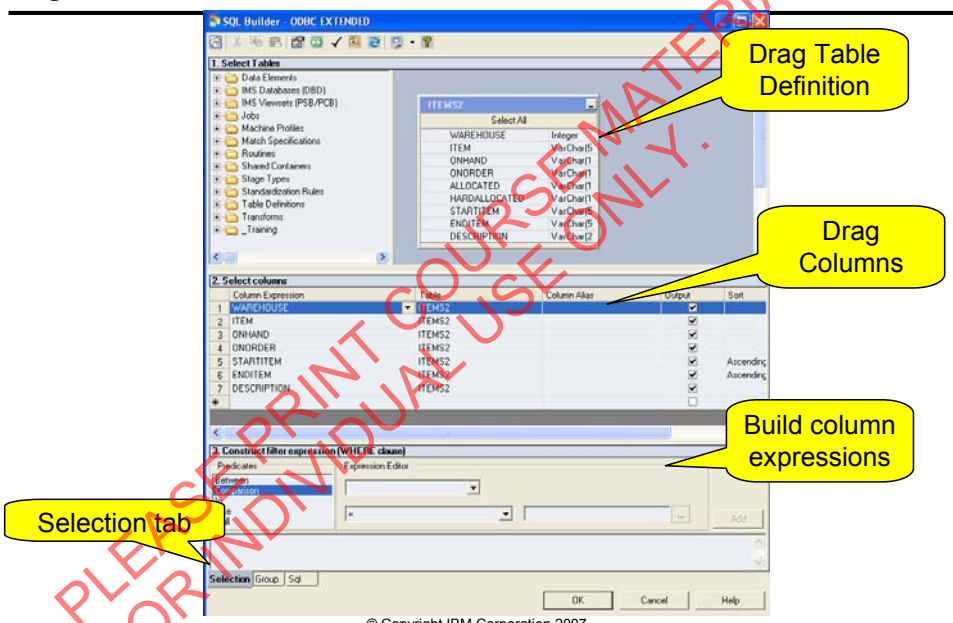



101

# SQL Builder Read Method



## SQL Builder Window



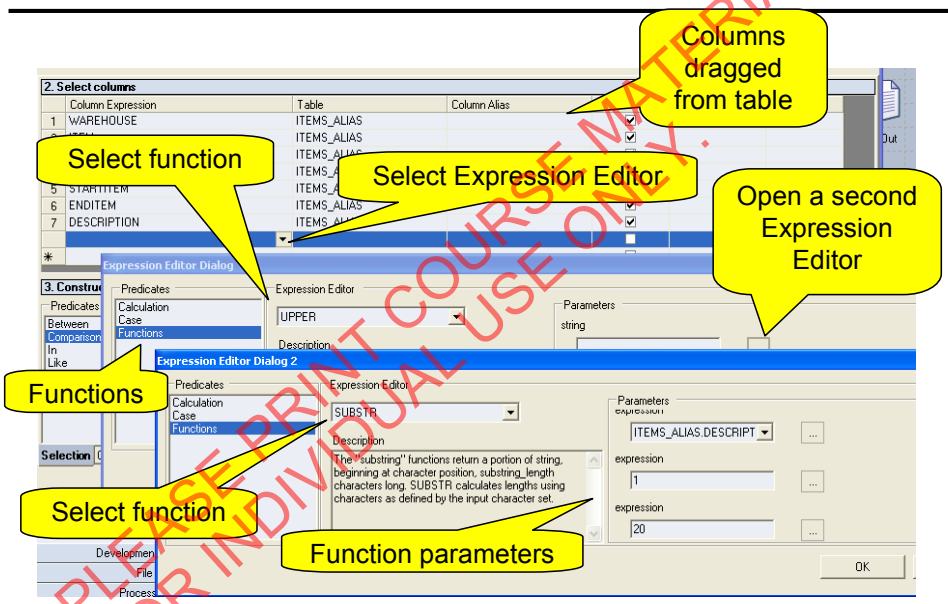





103

You build the query on the Selection tab, which is the first window you see when you open SQL Builder. Begin by dragging the Table Definitions for the tables to be read. Be sure the information on the Locator tab of the Table Definition is full and correct. In particular, be sure the table name and schema are specified. Otherwise the SQL statement that is corrected will not contain this information.

## Creating a Calculated Column



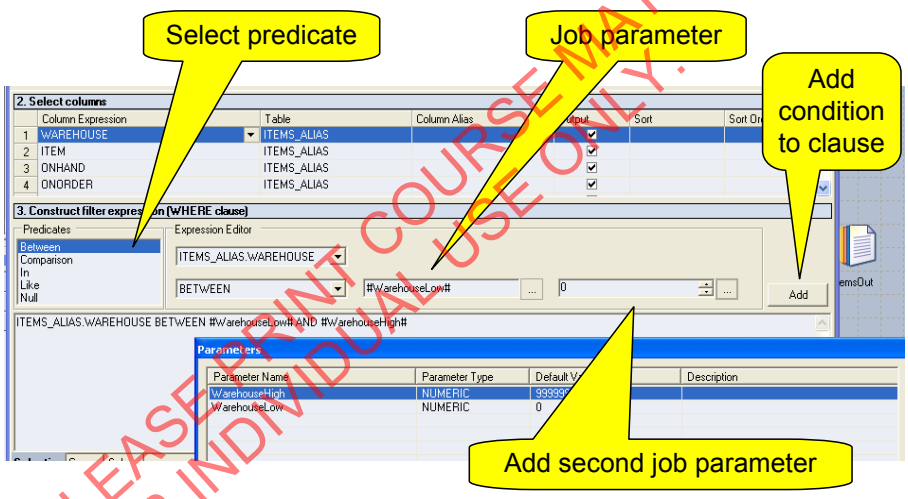



104



Here a new calculated column is being defined. The Expression Editor is selected from the column drop-down list. The UPPER function is selected. To define the parameter value the UPPER function is applied to, a second Expression Editor is opened. Here the SUBSTR function is selected to so select 20 characters from the beginning of the DESCRIPTION column.

## Constructing a WHERE Clause

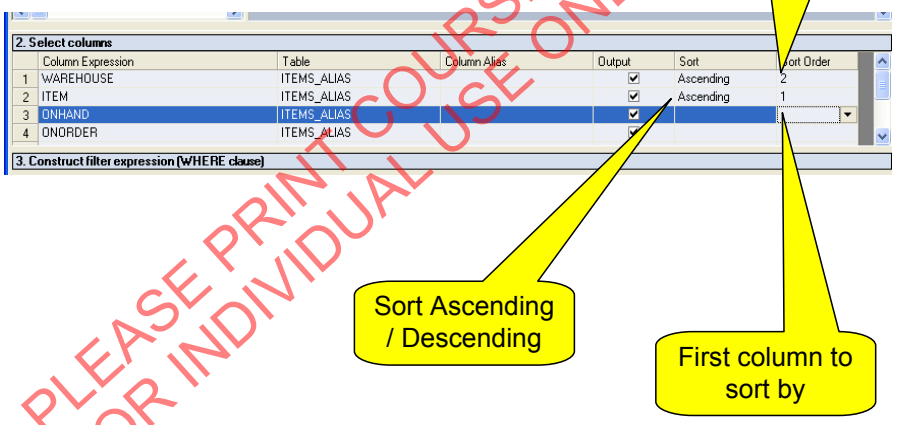







105

## Sorting the Data

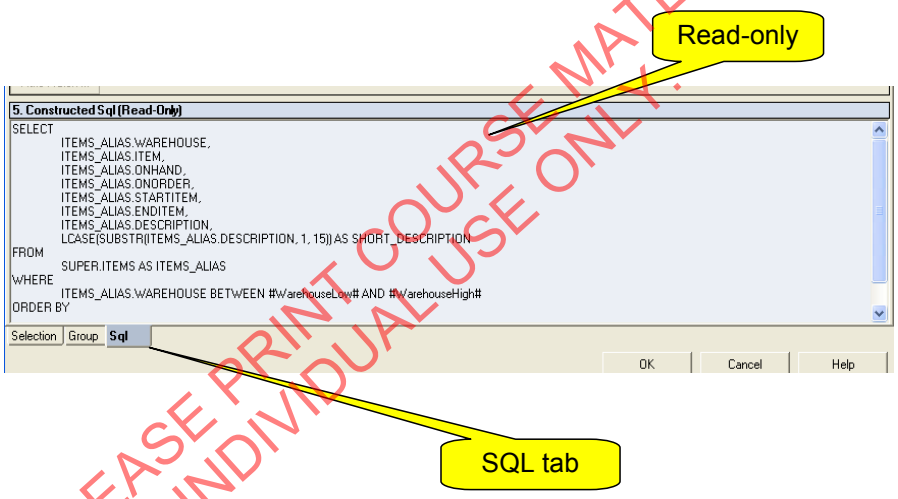







106

## Viewing the Generated SQL



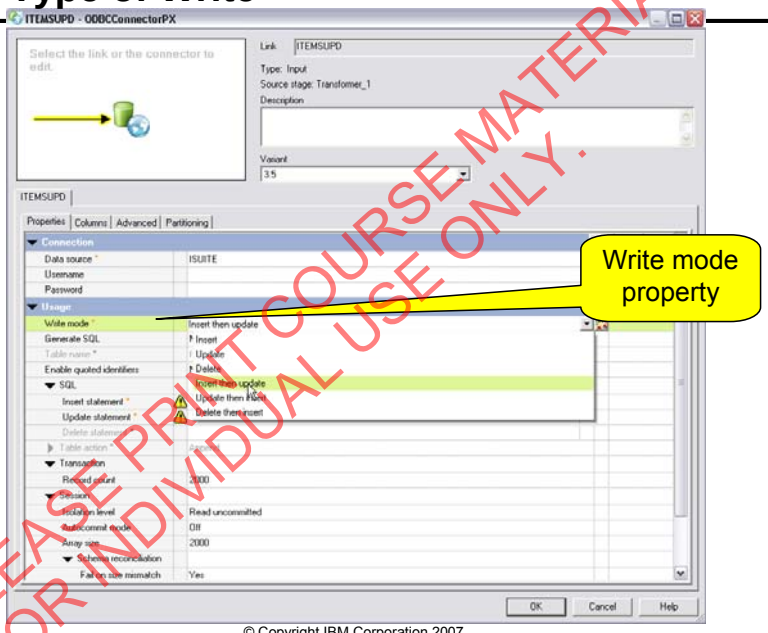




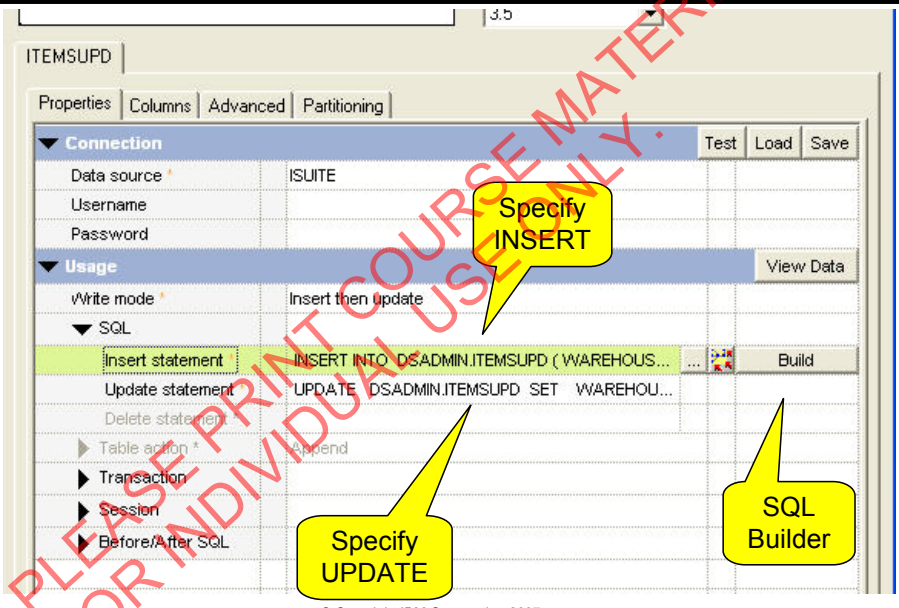

107

# Building SQL to Write to a Table

## Select Type of Write



## Invoking SQL Builder

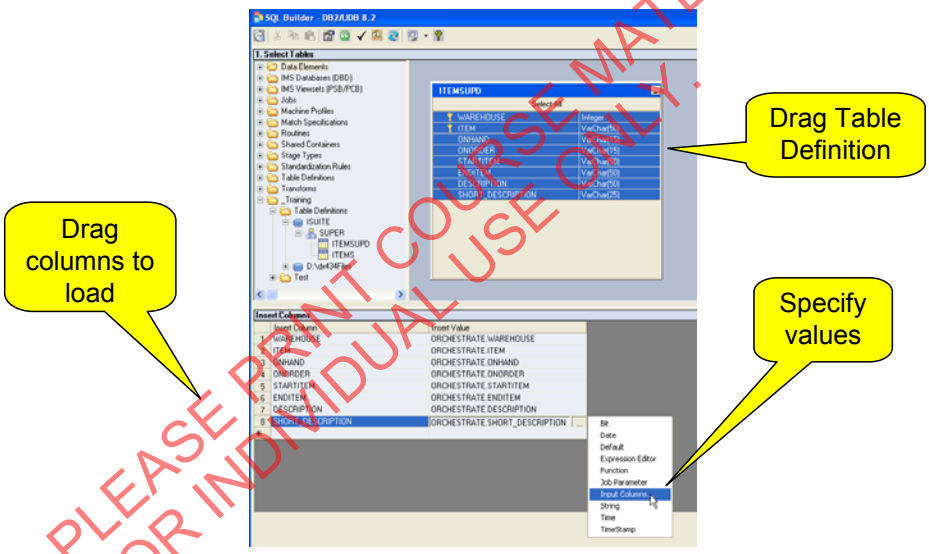







110

## SQL Builder INSERT

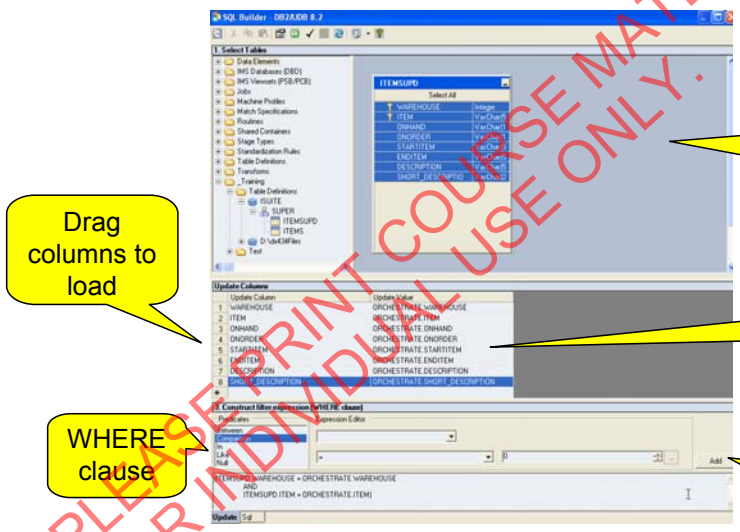







111

## SQL Builder Update

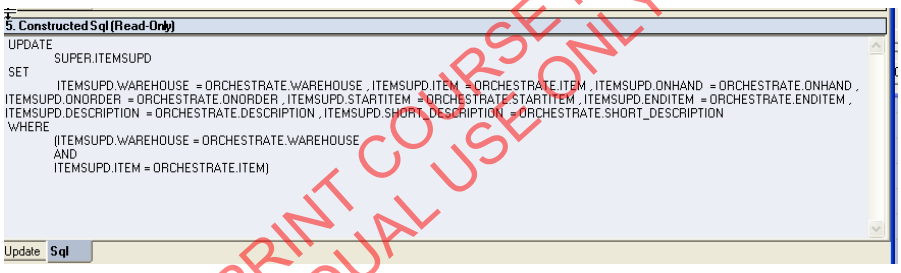







112

# Viewing the SQL

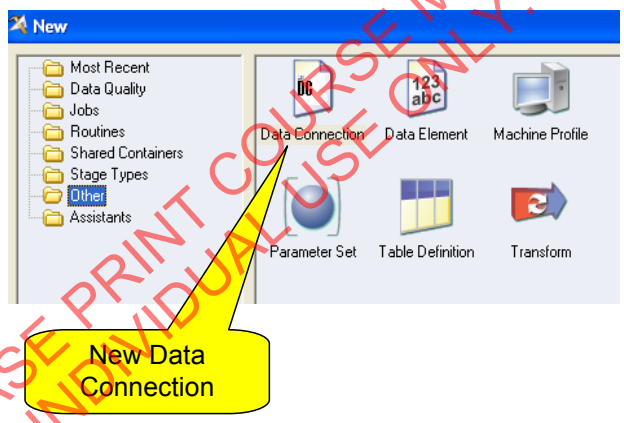


# Data Connections

## Data Connection

- Stores database parameters and values in a named object in the Repository
  - Optionally can create a parameter set with the parameters and values from with the Data Connection object
- Associated with a stage type
- Property values can be specified in a job stage of the given type by loading the Data Connection into the stage

## Creating a New Data Connection Object



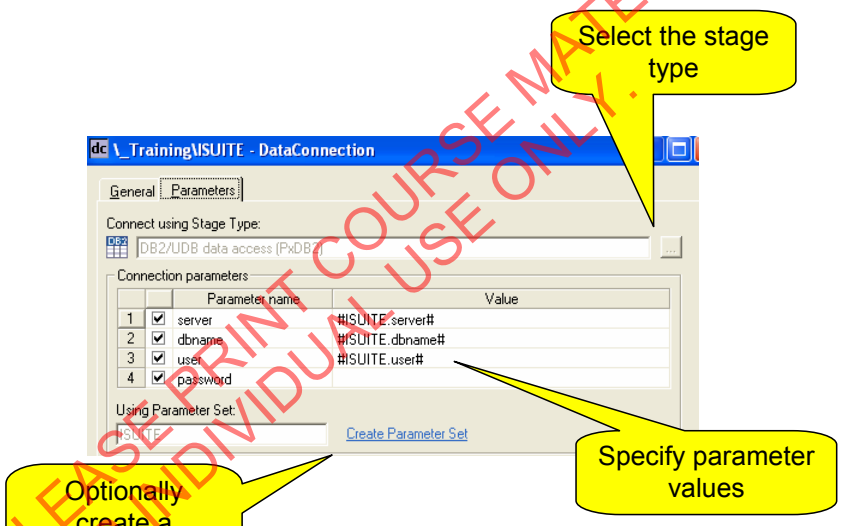





116

You can optionally save the parameters and values specified in a job stage into a new Data Connection. This provides another method for creating a Data Connection.

## Select the Stage Type

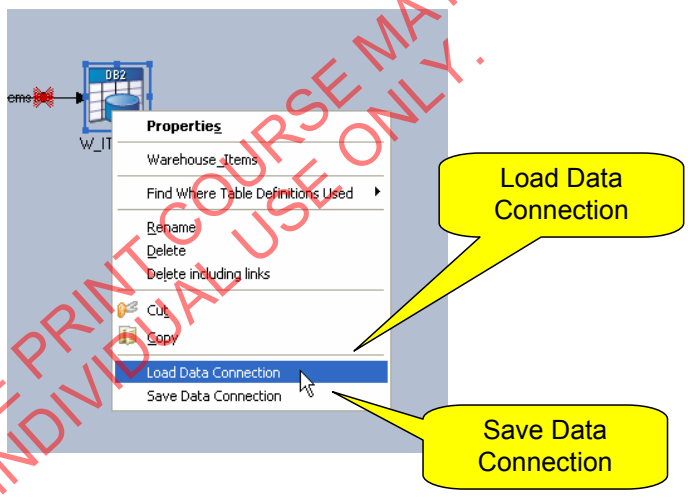







117

## Loading the Data Connection








118

Another way of loading the Data Connection is to drag and drop it onto the stage.

## Checkpoint

1. Which of the following types stages provide the most functionality in DataStage parallel jobs? ODBC Plug-in, ODBC Connector, ODBC Enterprise?
2. What are three ways of building SQL statements in Connector stages?
3. Which of the following statements can be specified in Connector stages? Select, Insert, Update, Upsert, Create Table.
4. What are two ways of loading Data Connection metadata into a database stage?





119

**Notes:**

Write down your answers here:

1.

2.

## Checkpoint solutions

1. Connector stages.
2. Manually. Using SQL Builder. Have the Connector stage generate the SQL.
3. All of them.
4. Click the right mouse button over the stage and click “Load Data Connection.” Drag the Data Connection from the Repository and drop it on the stage.

## Unit summary

Having completed this unit, you should be able to:
- Import Table Definitions for relational tables
- Create Data Connections
- Use Connector stages in a job
- Use SQL Builder to define SQL Select statements
- Use SQL Builder to define SQL Insert and Update statements
- Use the DB2 Enterprise stage

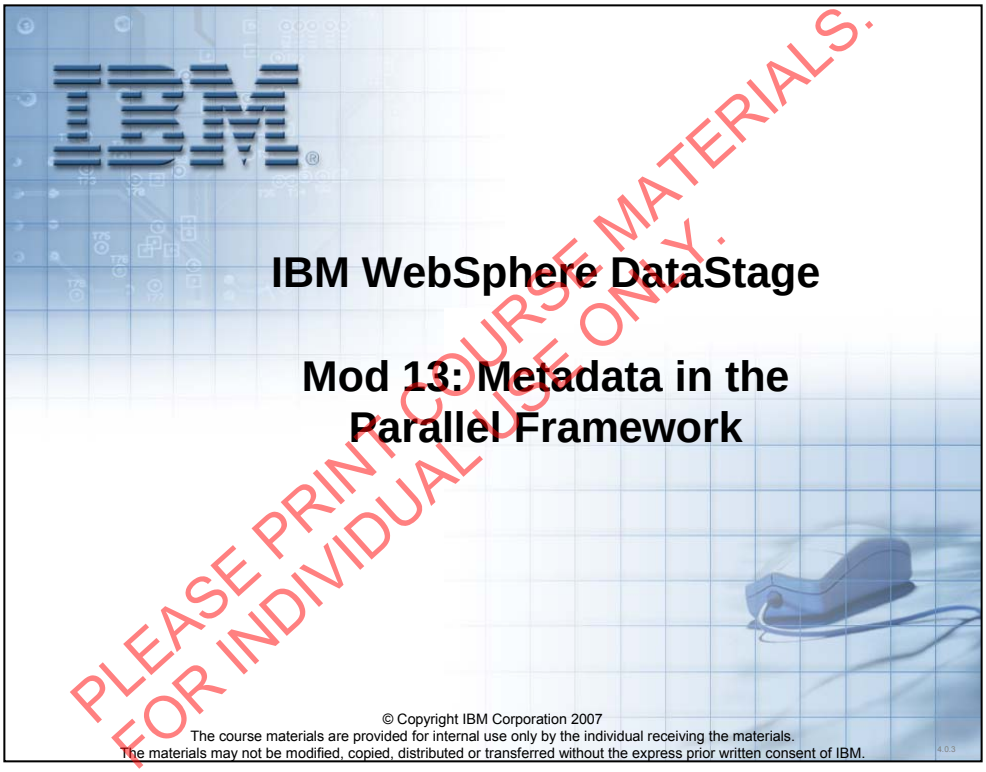




121

**Notes:**

122


IBM
IBM WebSphere DataStage
Mod 13: Metadata in the Parallel Framework
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:

- Explain schemas
- Create schemas
- Explain Runtime Column Propogation (RCP)
- Turn RCP on and off
- Build a job that reads data from a sequential file using a schema
- Build a shared container





124

**Notes:**

## Schema

- Alternative way to specifying column definitions and record formats
  - Similar to a Table Definition
- Written in a plain text file
- Can be imported as a Table Definition
- Can be created from a Table Definition
- Can be used in place of a Table Definition in a Sequential file stage
  - Requires RCP
  - Schema file path can be parameterized
    - Enables a single job to process files with different column definitions





125

The format of each line describing a column is:

column_name:[nullability]datatype;

column_name. This is the name that identifies the column. Names must start with a letter or an underscore (_), and can contain only alphanumeric or underscore characters. The name is not case sensitive. The name can be of any length.

nullability. You can optionally specify whether a column is allowed to contain a null value, or whether this would be viewed as invalid. If the column can be null, insert the word 'nullable'. By default columns are not nullable.

You can also include 'nullable' at record level to specify that all columns are nullable, then override the setting for individual columns by specifying 'not nullable'. For example:

```
record nullable (
        name:not nullable string[255];
        value1:int32;
        date:date)
```

datatype. This is the data type of the column.

## Creating a Schema

- Using a text editor
  - Follow correct syntax for definitions
  - Not recommended
- Import from an existing data set or file set
  - On DataStage Manager import > Table Definitions > Orchestrate Schema Definitions
  - Select checkbox for a file with .fs or .ds
- Import from a database table
- Create from a Table Definition
  - Click Parallel on Layout tab

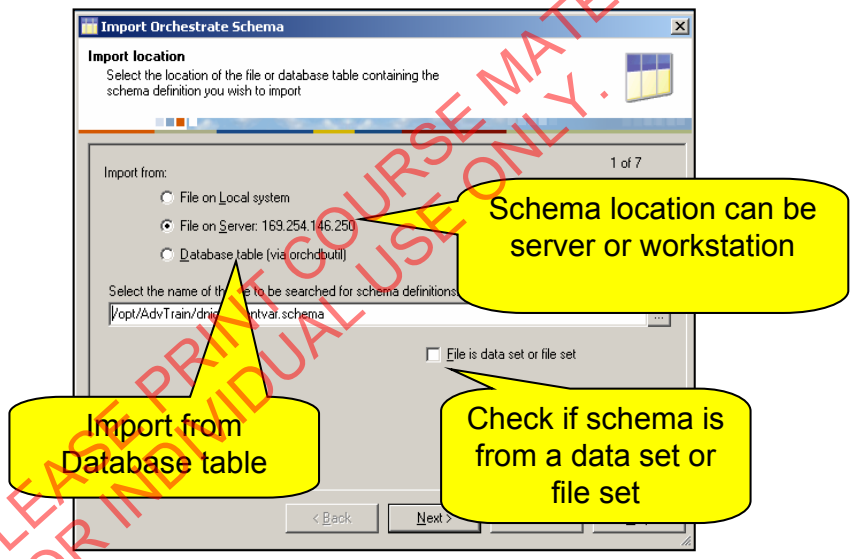




126

Another good way of capturing a schema is to set $OSH_PRINT_SCHEMAS and copy entries from the DataStage Director log.

## Importing a Schema

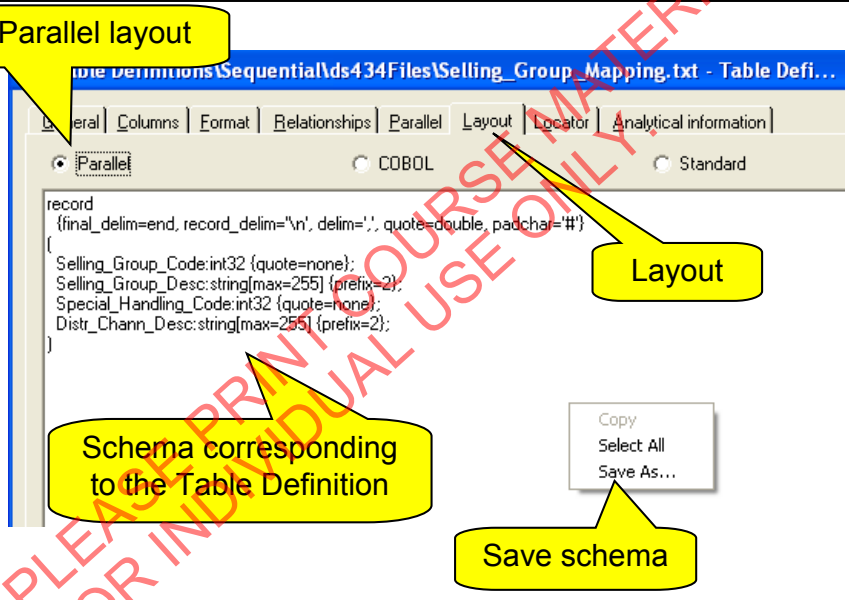







127

## Creating a Schema From a Table Definition

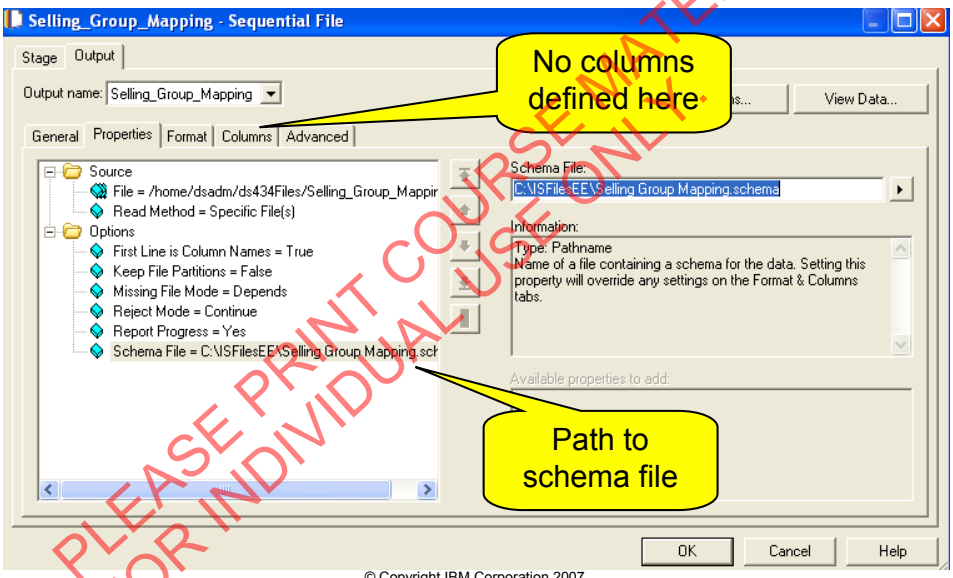







128

## Reading a Sequential File Using a Schema






129

Schemas can only be used when Runtime Column Propogation is turned on in the stage. This is discussed later in this module.

## Runtime Column Propagation (RCP)

- When RCP is turned on:
  - Columns of data can flow through a stage without being explicitly defined in the stage
  - Target columns in a stage need not have any columns explicitly mapped to them
    - No column mapping enforcement at design time
  - Input columns are mapped to unmapped columns by name
- How implicit columns get into a job
  - Read a file using a schema in a Sequential File stage
  - Read a database table using “Select *”
  - Explicitly define as an output column in a stage earlier in the flow
- Benefits of RCP
  - Job flexibility
    - Job can process input with different layouts
  - Ability to create reusable components in shared containers
    - Component logic an apply to a single named column
    - All other columns flow through untouched





130

## Enabling Runtime Column Propagation (RCP)

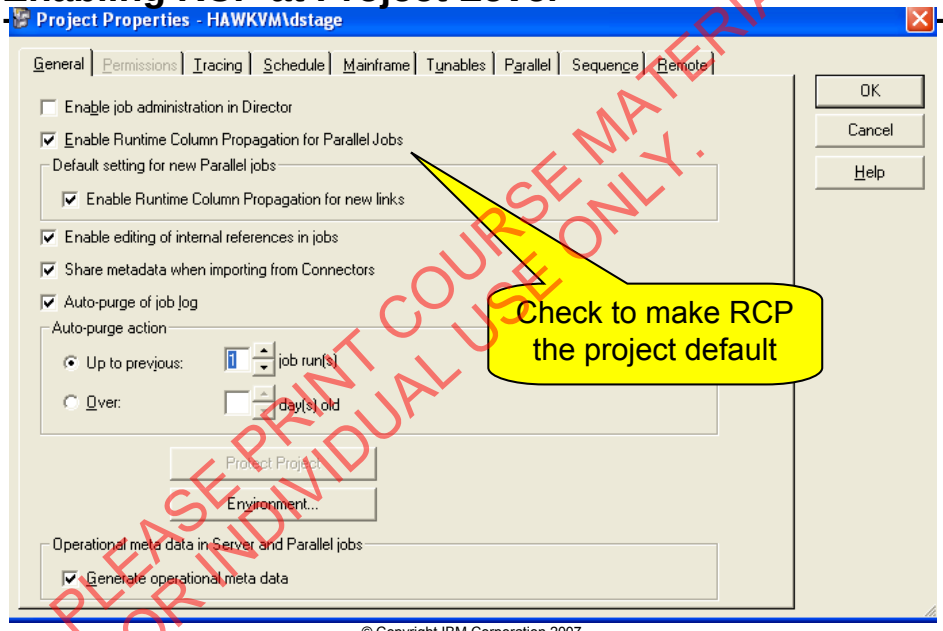
- Project level
  - DataStage Administrator Parallel tab
- Job level
  - Job properties General tab
- Stage level
  - Link Output Column tab
- Settings at a lower level override settings at a higher level
  - E.g., disable at the project level, but enable for a given job
  - E.g., enable at the job level, but disable a given stage





131

# Enabling RCP at Project Level

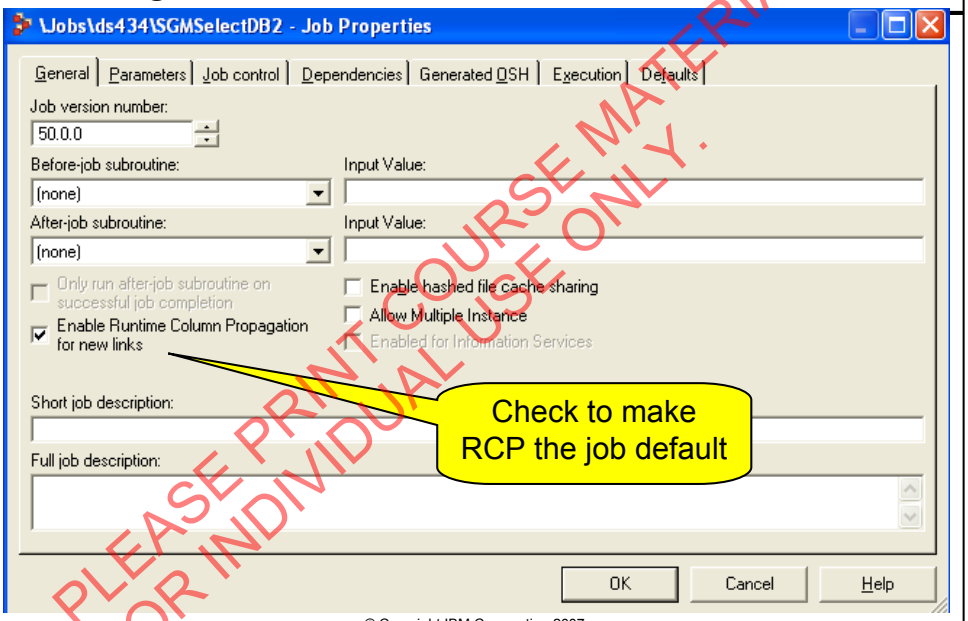







132

## Enabling RCP at Job Level



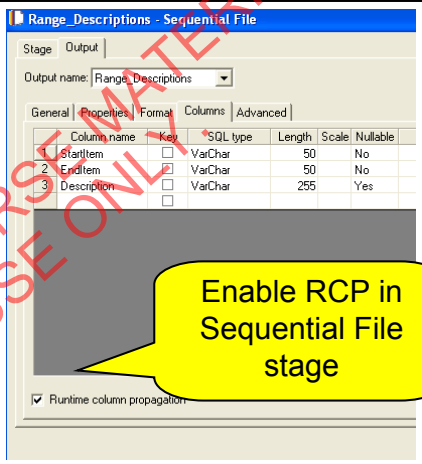


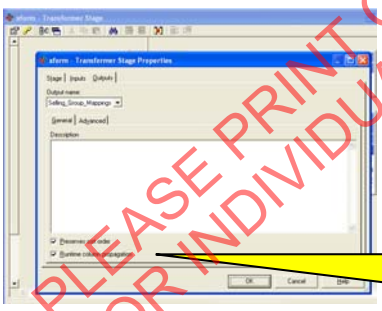



133

# Enabling RCP at Stage Level

- Sequential File stage
  - Output Columns tab
- Transformer
  - Open Stage Properties
  - Stage Properties Output tab



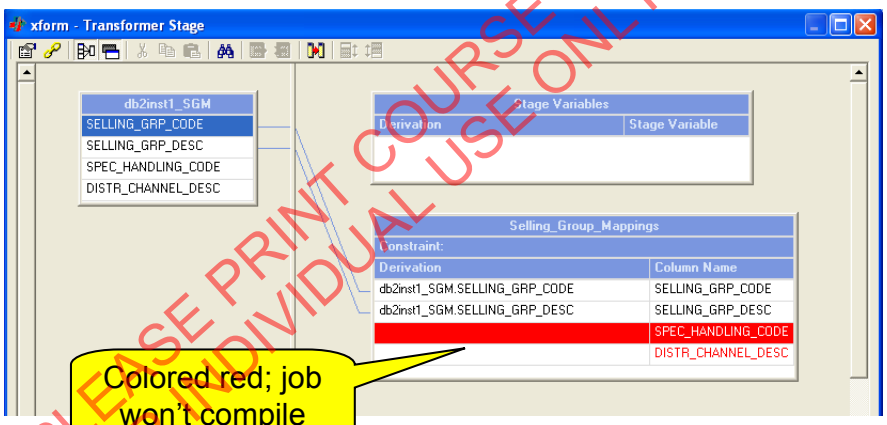





134

## When RCP is Disabled

- DataStage Designer enforces Stage Input to Output column mappings.



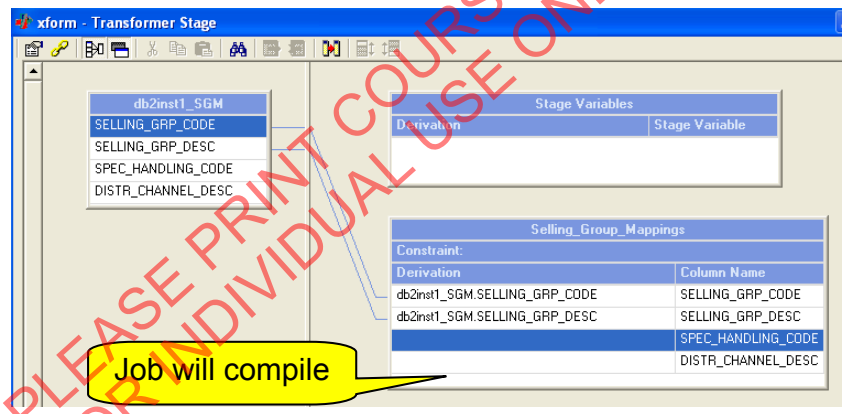



135

Modify operators can add or change columns in a data flow.

## When RCP is Enabled

- DataStage does not enforce mapping rules
- Runtime error if no incoming columns match unmapped target column names








136

# Shared Containers

## Shared Containers

- Encapsulate job design components into a stored container
- Provide reusable job design components
- Example
  - Apply stored Transformer business logic

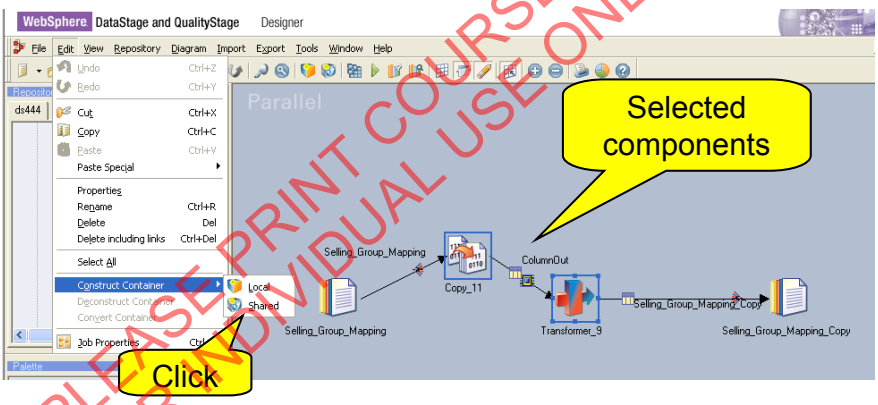




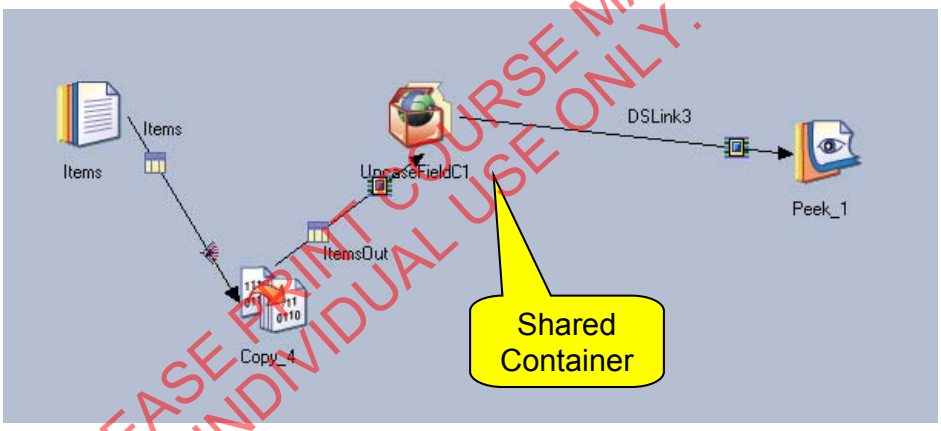
138

# Creating a Shared Container

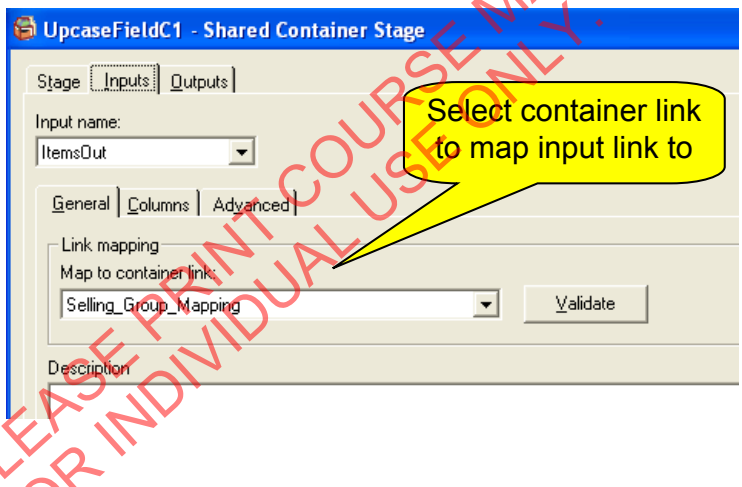
- Select stages from an existing job
- Click Edit>Construct Container>Shared



## Using a Shared Container in a Job



## Mapping Input / Output Links to the Container








141

## Checkpoint

1. What are two benefits of RCP?
2. What can you use to encapsulate stages and links in a job to make them reusable?





142

**Notes:**

Write down your answers here:

1.

2.

## Checkpoint solutions

1. Job flexibility. Ability to create reusable components.
2. Shared containers

## Unit summary

Having completed this unit, you should be able to:

- Explain schemas
- Create schemas
- Explain Runtime Column Propogation (RCP)
- Turn RCP on and off
- Build a job that reads data from a sequential file using a schema
- Build a shared container





144

**Notes:**


IBM
IBM WebSphere DataStage
Special Topic 5: Job Control
PLEASE PRINT COURSE MATERIALS.
FOR INDIVIDUAL USE ONLY.
© Copyright IBM Corporation 2007
The course materials are provided for internal use only by the individual receiving the materials.
The materials may not be modified, copied, distributed or transferred without the express prior written consent of IBM.
4.0.3

## Unit objectives

After completing this unit, you should be able to:

- Use the DataStage Job Sequencer to build a job that controls a sequence of jobs
- Use Sequencer links and stages to control the sequence a set of jobs run in
- Use Sequencer triggers and stages to control the conditions under which jobs run
- Pass information in job parameters from the master controlling job to the controlled jobs
- Define user variables
- Enable restart
- Handle errors and exceptions





146

**Notes:**

## What is a Job Sequence?

- A master controlling job that controls the execution of a set of subordinate jobs
- Passes values to the subordinate job parameters
- Controls the order of execution (links)
- Specifies conditions under which the subordinate jobs get executed (triggers)
- Specifies complex flow of control
  - Loops
  - All / Some
  - Wait for file
- Perform system activities
  - Email
  - Execute system commands and executables
- Can include Restart checkpoints





147

## Basics for Creating a New Job Sequence

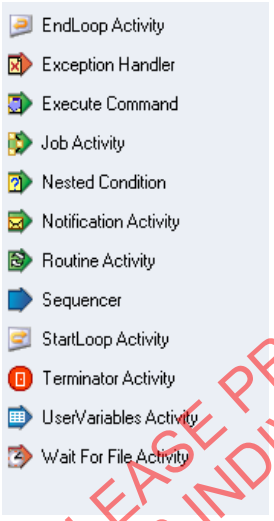
- Open a new job sequence
  - Specify whether its restartable
- Add stages
  - Stages to execute jobs
  - Stages to execute system commands and executables
  - Special purpose stages
- Add links
  - Specify the order in which jobs are to be executed
- Specify triggers
  - Triggers specify the condition under which control passes across a link
- Specify error handling
- Enable / Disable restart checkpoints





148

## Job Sequencer Stages



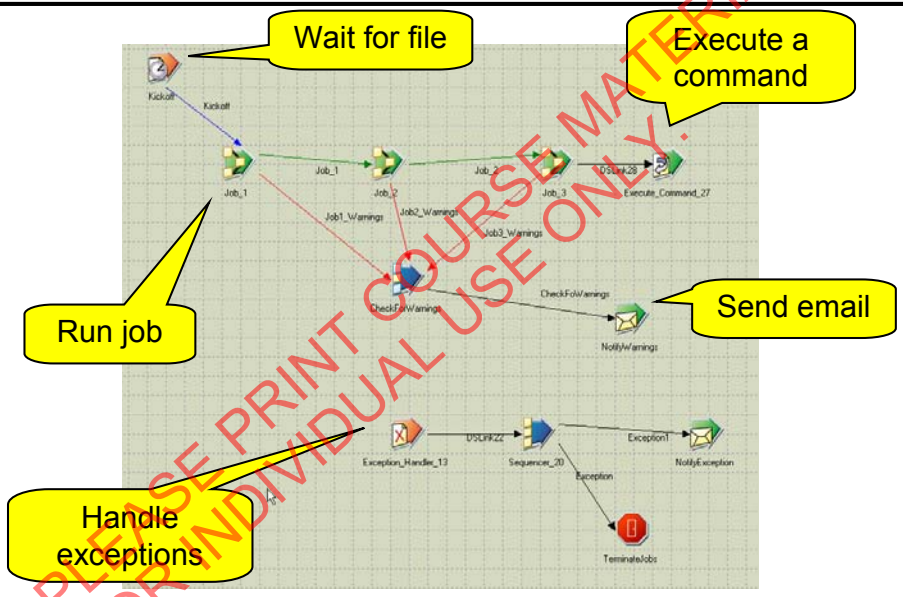

- Run stages
  - Job Activity: Run a job
  - Execute Command: Run a system command
  - Notification Activity: Send an email
- Flow control stages
  - Sequencer: Go if All / Some
  - Wait for File: Go when file exists / doesn't exist
  - StartLoop / EndLoop
  - Nested Condition: Go if condition satisfied
- Error handling
  - Exception Handler
  - Terminator
- Variables
  - User Variables





149

## Example



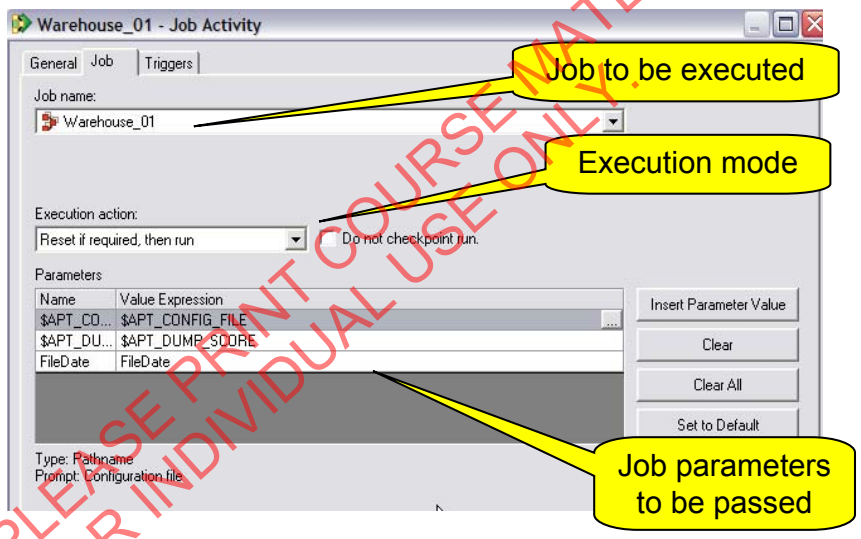
## Sequence Properties






151

## Job Activity Stage Properties

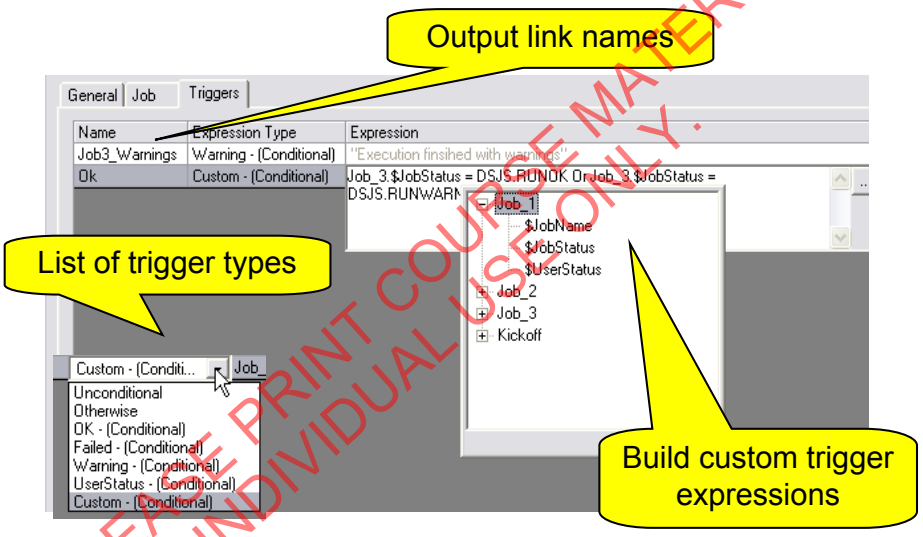







152

## Job Activity Trigger

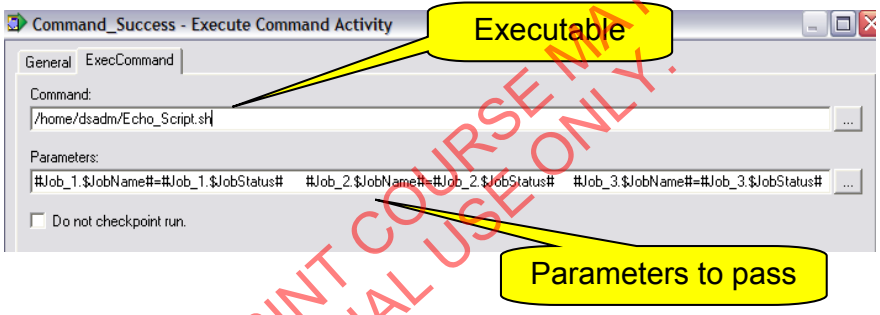






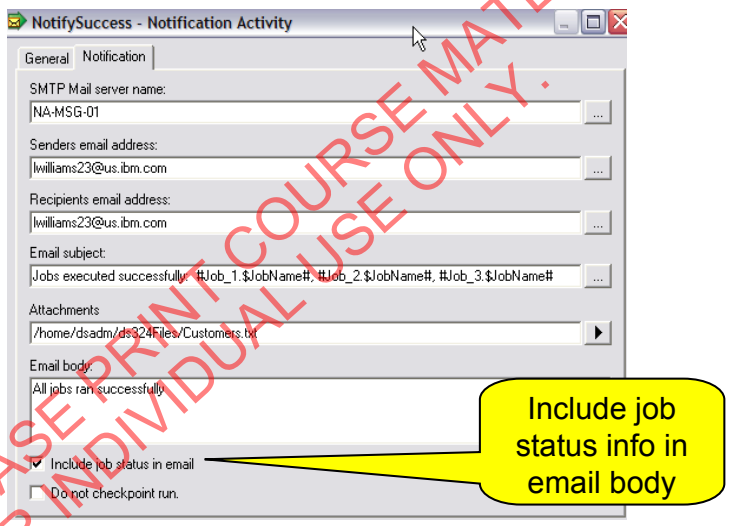

153

## Execute Command Stage

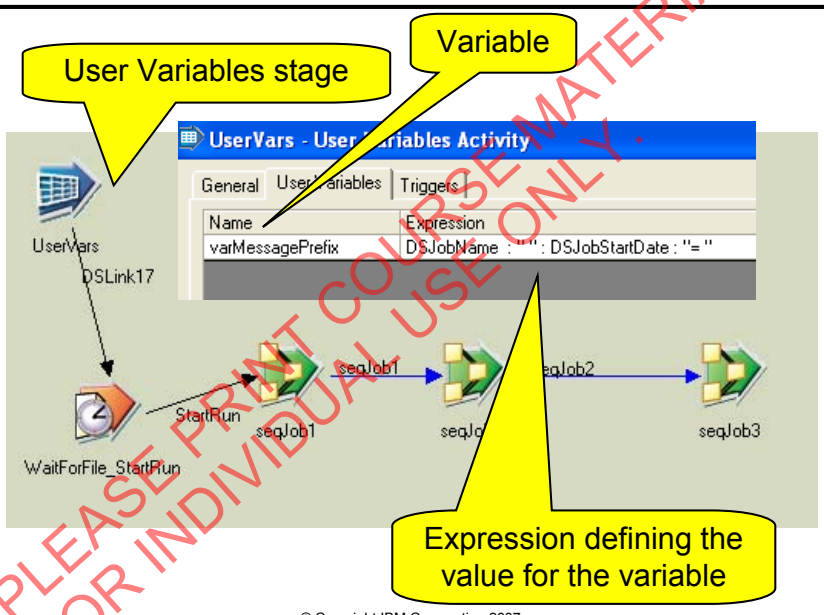



- Execute system commands, shell scripts, and other executables
- Use e.g. to drop or rename database tables

## Notification Activity Stage



## User Variables Stage

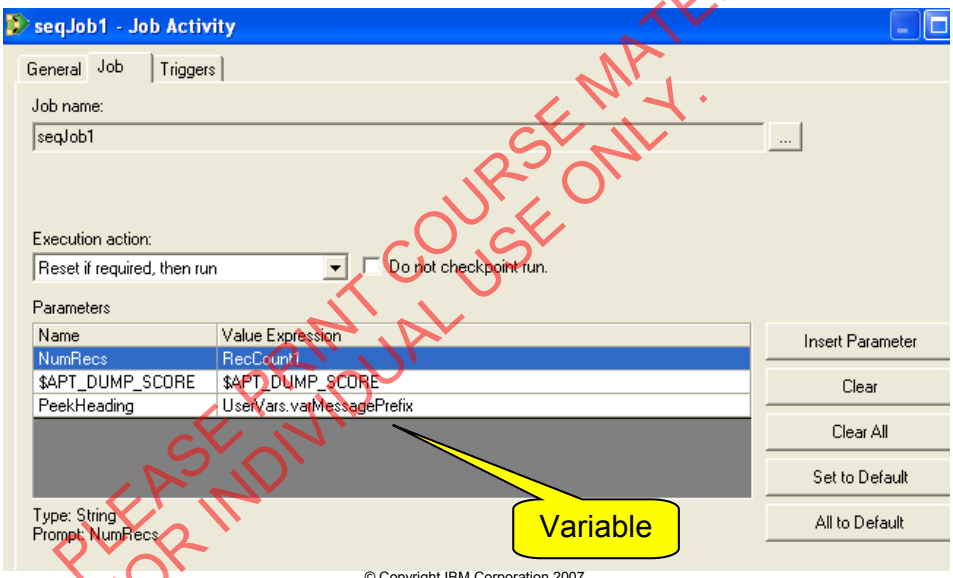







156

## Referencing the User Variable








157

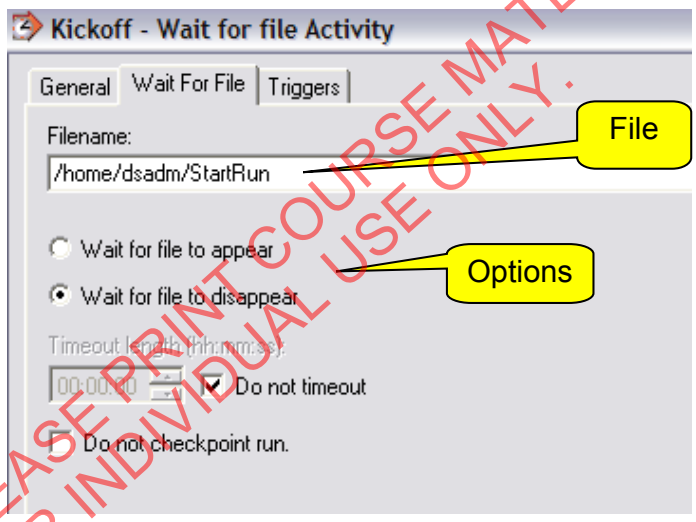
# Flow of Control Stages





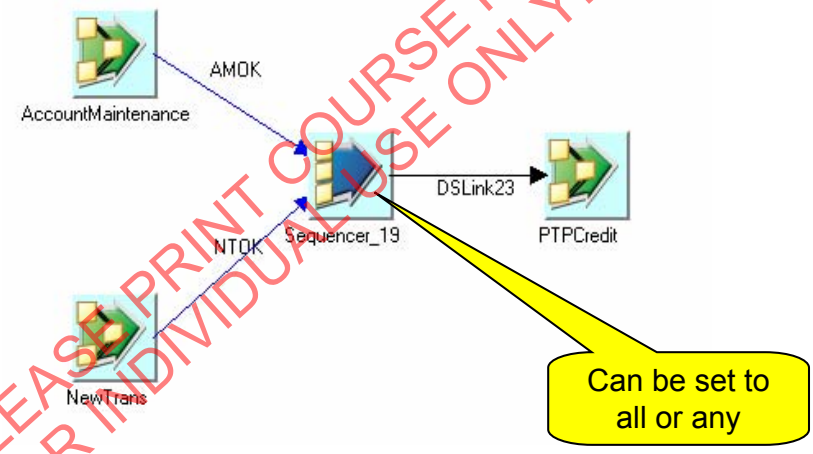
158

## Wait for File Stage



## Sequencer Stage

- Sequence multiple jobs using the Sequence stage

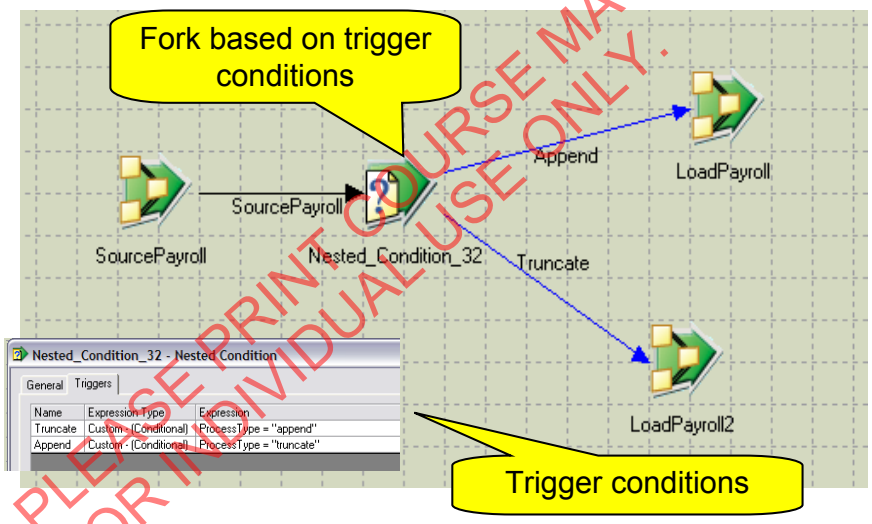




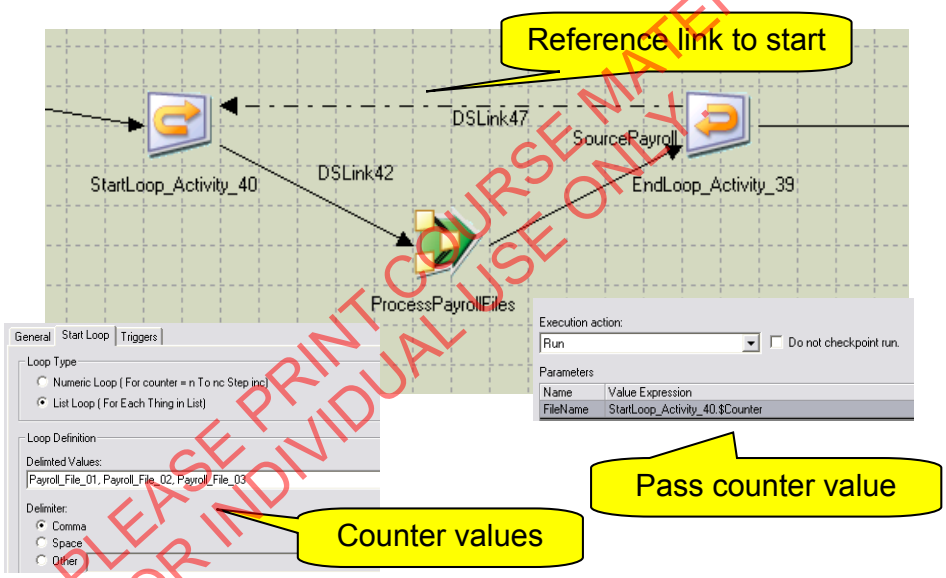



160

## Nested Condition Stage

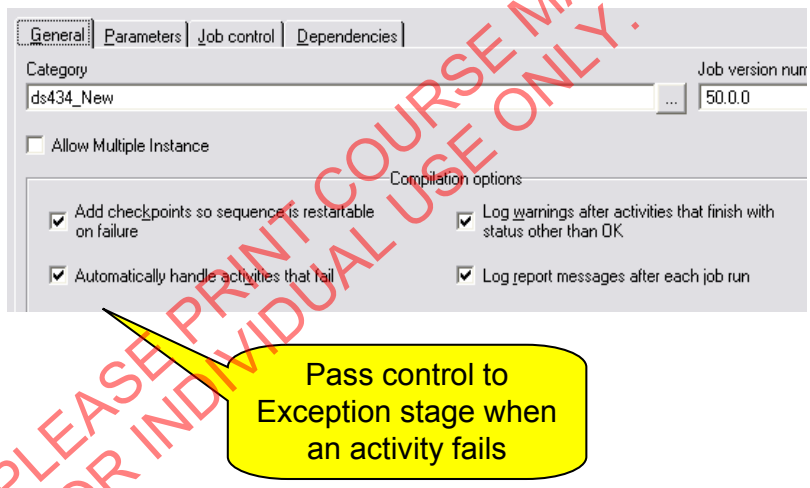


## Loop Stages

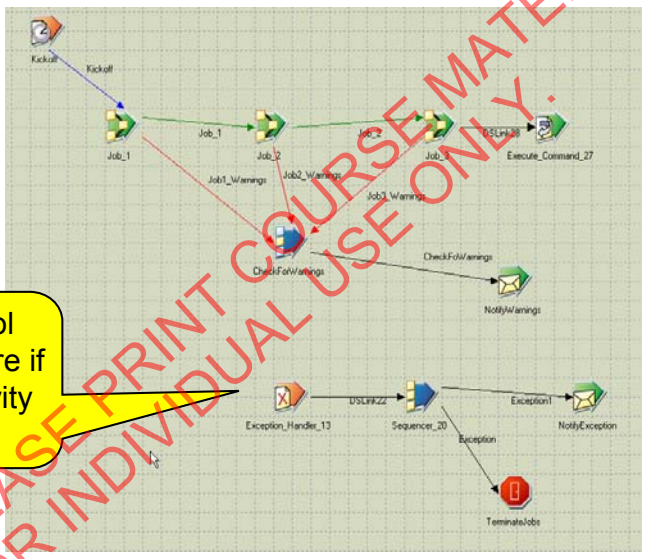


# Error Handling

## Handling Activities that Fail



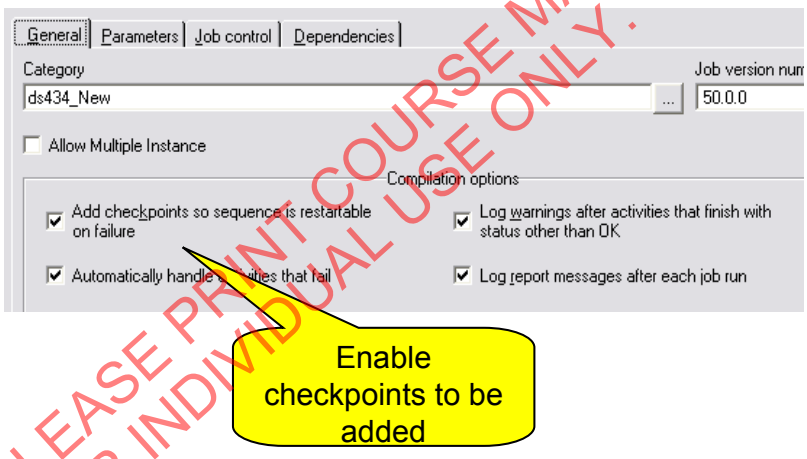
## Exception Handler Stage




Control goes here if an activity fails

# Restart

## Enable Restart



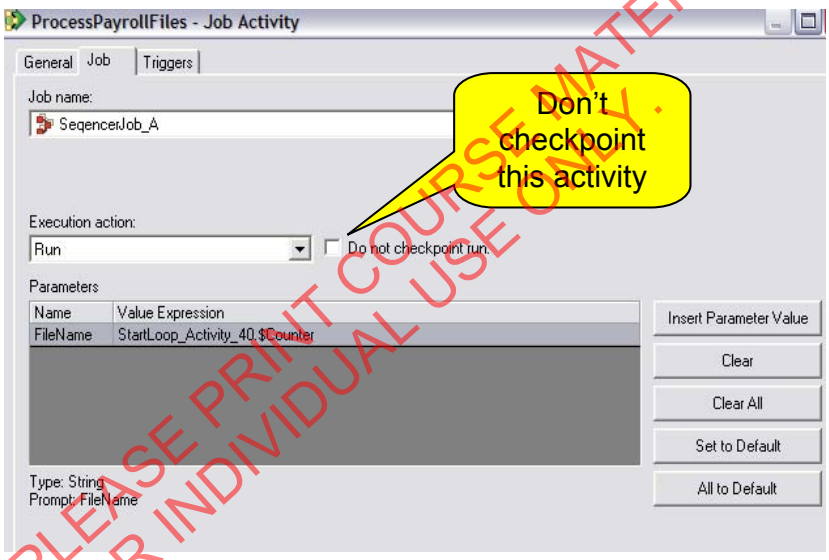





167

If a Sequence fails, and it is set to "Add check points so sequence is restartable on failure", then when the Sequence is re-run, activities that completed successfully in the prior run are skipped over (unless the "Do not checkpoint run" option was set for an activity).

## Disable Checkpoint at a Stage






168

## Checkpoint

1. Which stage is used to run jobs in a job sequence?
2. Does the Exception Handler stage support an input link?





169

## Checkpoint solutions

1. Job Activity stage
2. No, control is automatically passed to the stage when an exception occurs.

## Unit summary

Having completed this unit, you should be able to:

- Use the DataStage Job Sequencer to build a job that controls a sequence of jobs
- Use Sequencer links and stages to control the sequence a set of jobs run in
- Use Sequencer triggers and stages to control the conditions under which jobs run
- Pass information in job parameters from the master controlling job to the controlled jobs
- Define user variables
- Enable restart
- Handle errors and exceptions





171

**Notes:**