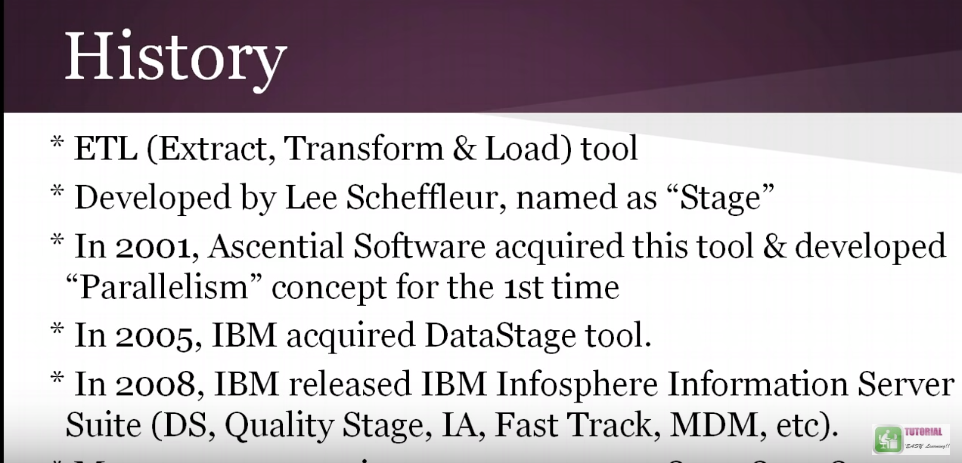
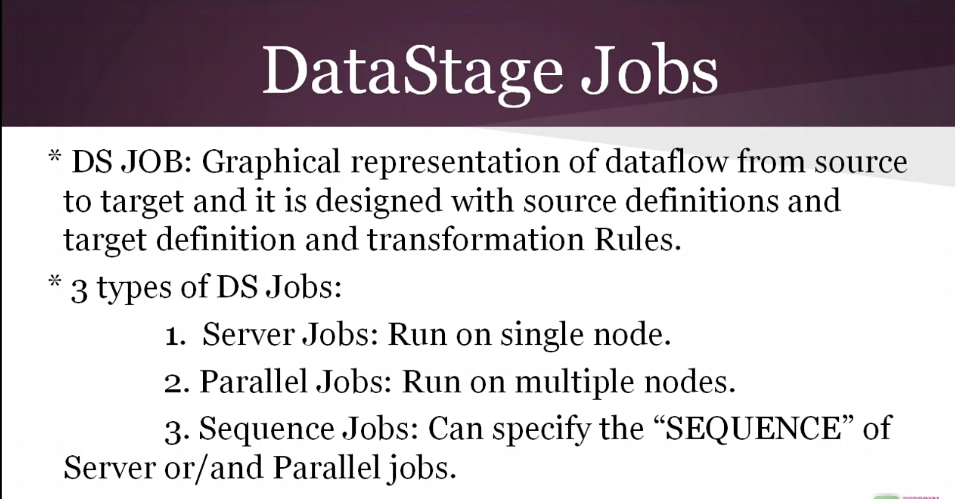
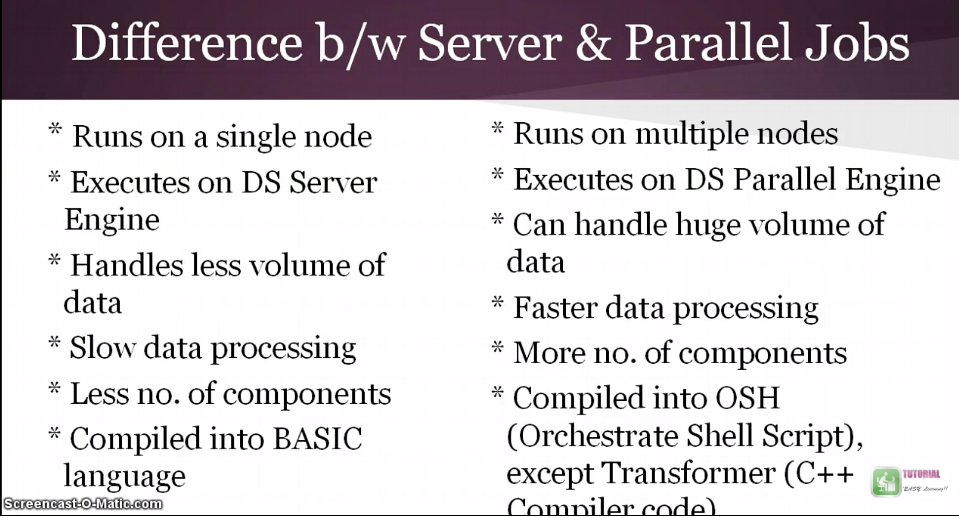
Data stage

Introduction:





Difference between Server and parallel Jobs.



The choice of server or parallel depends upon time to implement, functionality and cost.

When we have lots of functionality to implement for lower volume and hardware is less and ease of implementation we can go for Server jobs.

Parallel jobs are costly due to high scale of hardware, difficult to implement, extreme processing capabilities

for absurd volumes with vast array of operators for high-performance manipulation.

Parallel jobs take advantage of both pipeline parallelism and partitioning parallelism.

We can improve the performance of server job by enabling inter process row buffering. This helps stages

to xchange data as soon as it is available in the link. IPC stage also helps passive stage to read data from

another as soon as data is available. In other words, stages do not have to wait for the entire set of records

to be read first and then transferred to the next stage. Link partitioner and link collector stages can be used

to achieve a certain degree of partitioning parallelism.

Look up with sequential file is possible in parallel jobs and not possible in server jobs.

Datastage EE jobs are compiled into OSH (Orchestrate Shell script language).OSH executes operators –

Instances of executable C++ classes, pre-built components representing stages used in Data stage jobs.

Server Jobs are compiled into Basic which is an interpreted pseudo-code. This is why parallel jobs run faster,

Even if processed on one CPU.

<https://www.slideshare.net/shanker_uma/datastage-parallell-jobs-vs-datastage-server-jobs>

NODES:



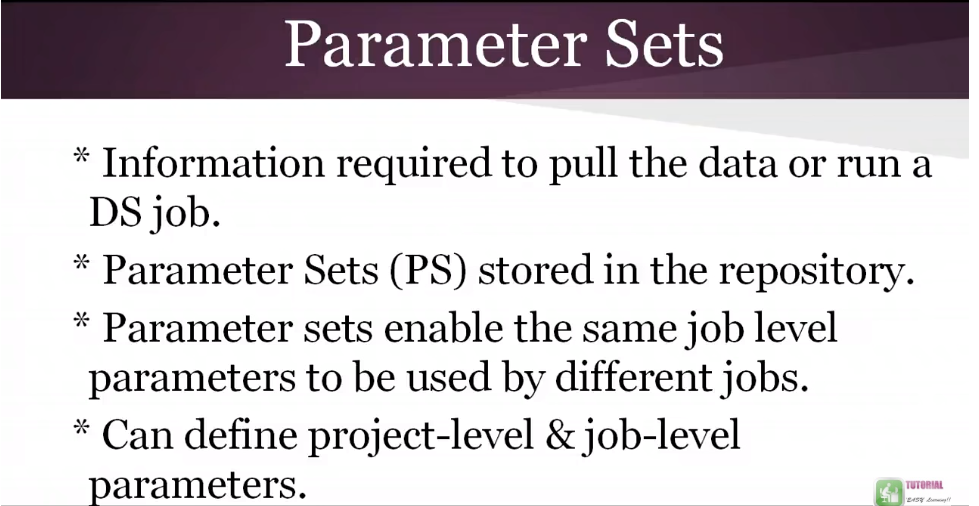
Nodal information is stored in Configuration file called Apt config file which represents the degree of parallelism.

How APT configuration looks like:



Before importing table definition of a database, a DB need to be configured by administrator.

Parameter and Parameter Sets Same like Informatica can be given for external parameter, source target connection and are likely to be used under different jobs.

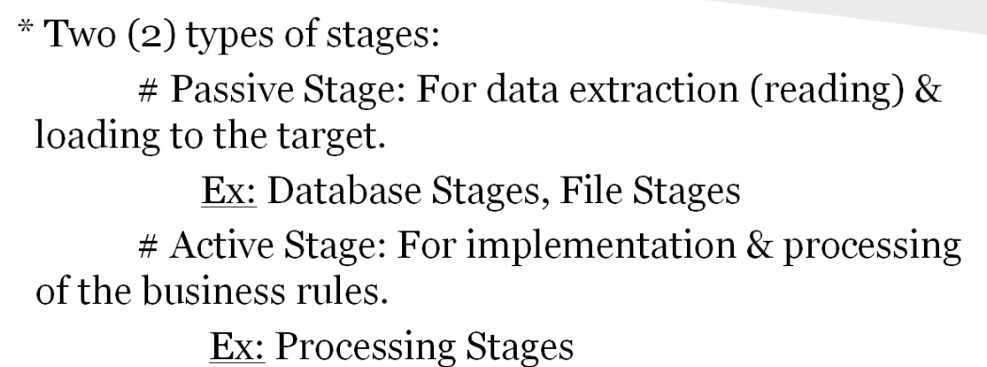


Create a parameter Set:

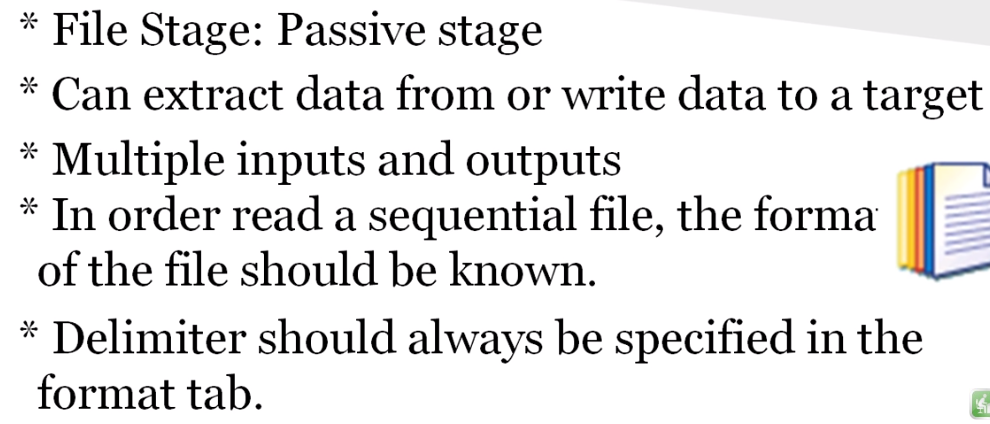
Designer --> Job properties --> Parameter --> Select the required parameter --> Create parameter set--> Name it—give value--> save.

Sequential File Stage in Data Stage Server Jobs

Types of stage:



Sequential File stage:



Difference between hash file and sequence file

Hash file stores the data based on hash algorithm and on a key value. A sequential file is just a file with no

key column. Hash file used as a reference for look up. Sequential file cannot be used for reference in lookup.

Sequential File:

* It is used to read/write a Flat File  Sequential File
* The Data will be loaded normally
* This is not an used for performance wise
* No limit for storage
* Single File only create for output
* Not query possible in Flat file or Sequential File
* It does not perform for remove duplicates
* It does not support for lookup operation

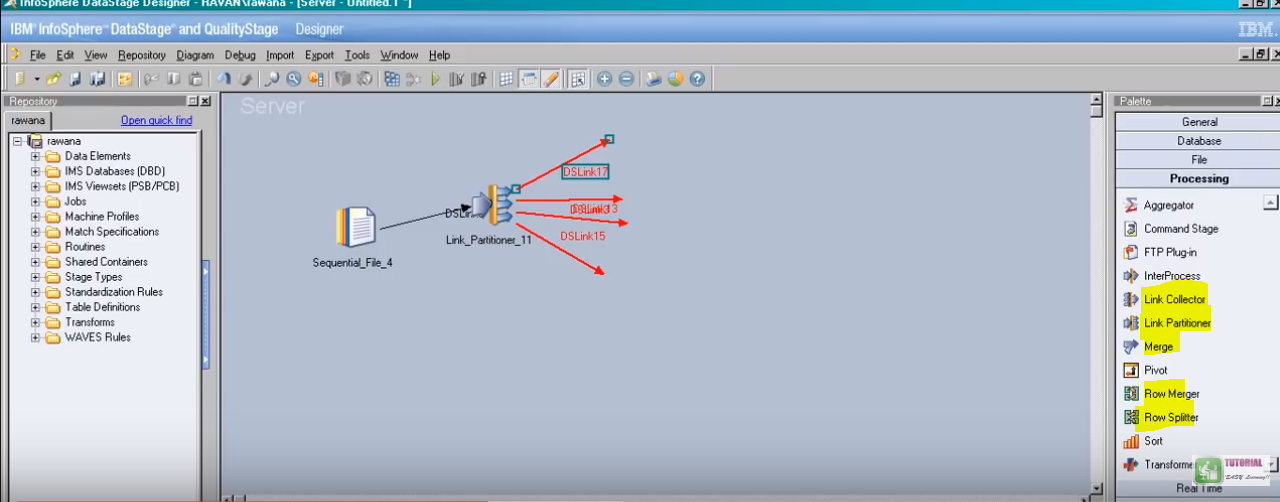
Hash File:

* It is used to read/write a  Hash File
* Data is loaded based on Hash Algorithm
* This is very effective to store the temporary Data
* it supports up to 2 GB then over-write
* Hash file Creates two files (Static file and Dynamic)
* It removes the Duplicate Records
* It supports for lookup operation

# Processing Stages in Server Jobs: Not required as of now, for further information go to link below.

<https://www.youtube.com/watch?v=TnqqB2Rf1v8&index=6&list=PLV8u8106ph7aaAc4Y59G3xvEs7oidXeyr>

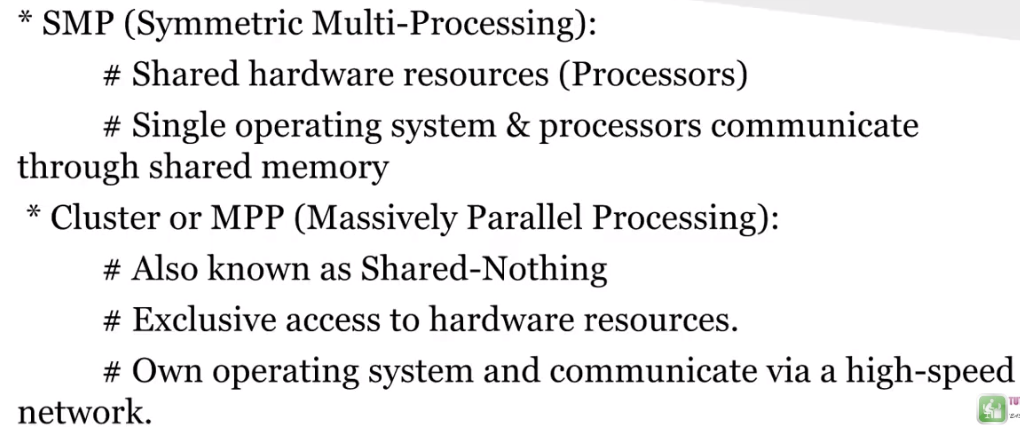
Need to focus on below highlighted terms.



Parallel Jobs: **Parallelism & Partitioning Techniques**

<https://www.ibm.com/support/knowledgecenter/zh/SSZJPZ_9.1.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/parallelprocessing.html>

Below comes in picture when interviewer asks you about tell me about your project architecture.



How a parallel job processes:

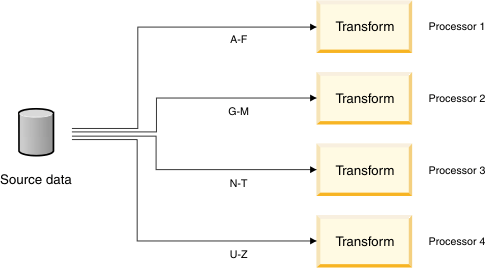
Pipeline Parallelism:

If you ran the example job on a system with at least three processors, the stage reading would start on one processor and start filling a pipeline with the data it had read. The transformer stage would start running on another processor as soon as there was data in the pipeline, process it and start filling another pipeline. The stage writing the transformed data to the target database would similarly start writing as soon as there was data available. Thus all three stages are operating simultaneously. If you were running sequentially, there would only be one instance of each stage. If you were running in parallel, there would be as many instances as you had partitions.

Partitioning Parallelism.

Data partitioning is an approach to parallelism that involves breaking the record set into partitions, or subsets of records. If no resource constraints or other data skew issues exist, data partitioning can provide linear increases in application performance. [Figure 2](https://www.ibm.com/support/knowledgecenter/SSZJPZ_9.1.0/com.ibm.swg.im.iis.productization.iisinfsv.overview.doc/topics/cisoarchparalbasics.html?view=kc#cisoarchparalbasics__arch_partit) shows data that is partitioned by customer surname before it flows into the Transformer stage.

*Figure 2. Data partitioning*



A scalable architecture should support many types of data partitioning, including the following types:

Hash key (data) values

Range

Round-robin

Random

Entire

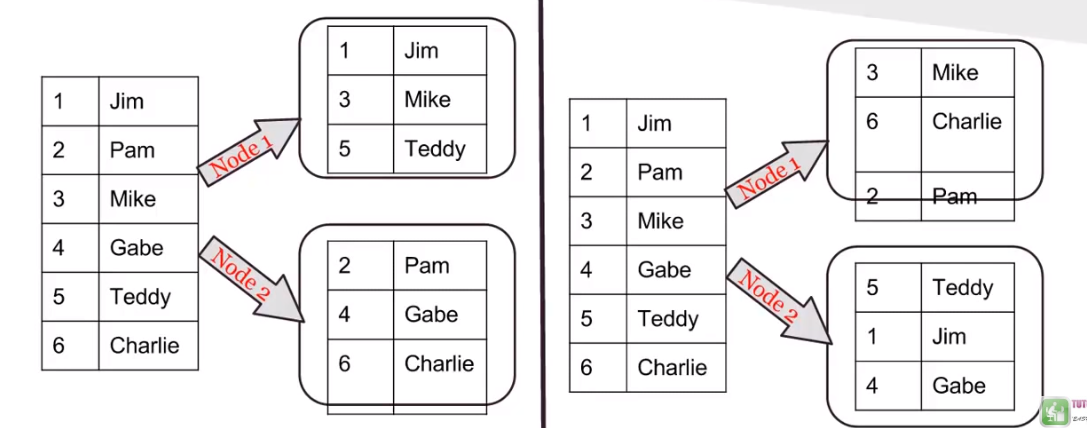
Modulus

Database partitioning

Round Robin: Suppose we have two nodes and total 6 records, then 1st will be written in node 1 and 2nd in node to and so on.

Random: Goes to random Nodes.

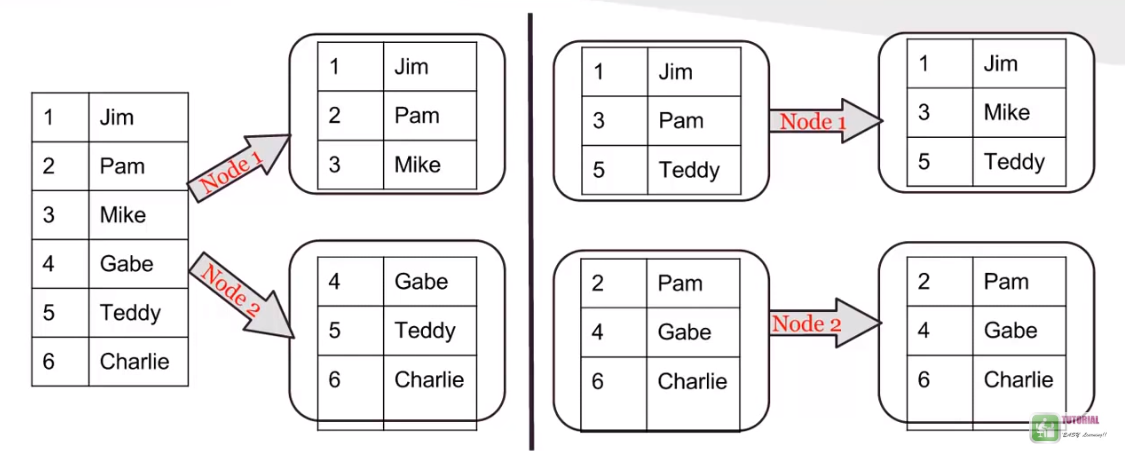
Round Robin Random



Range: Specify a range like 1 to 60 marks go in this node and 61 to 100 go in another.

Same: Definite records suggested by Name.

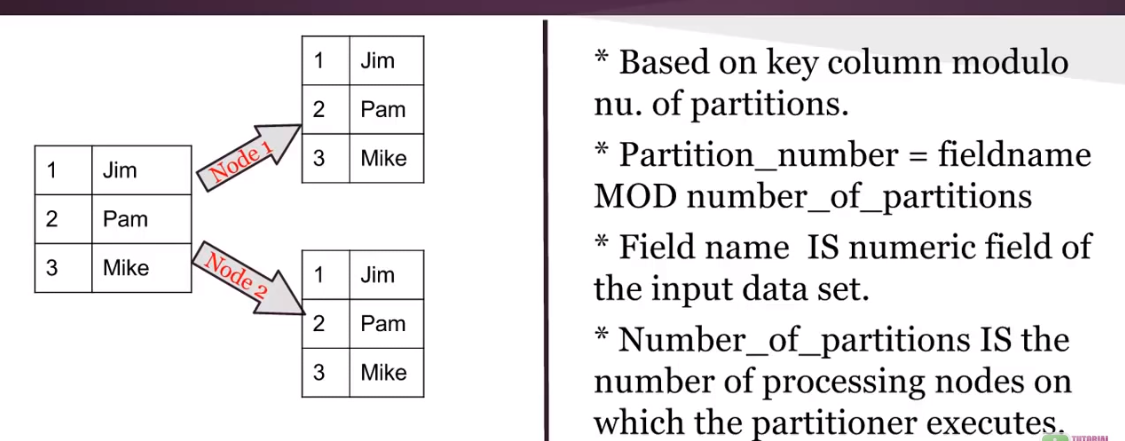
Range Same

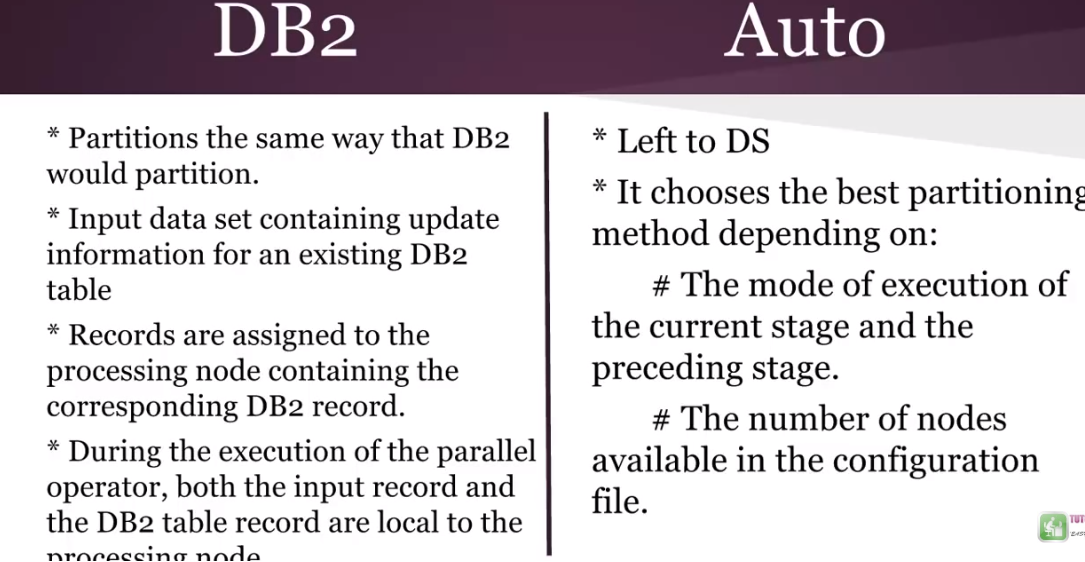


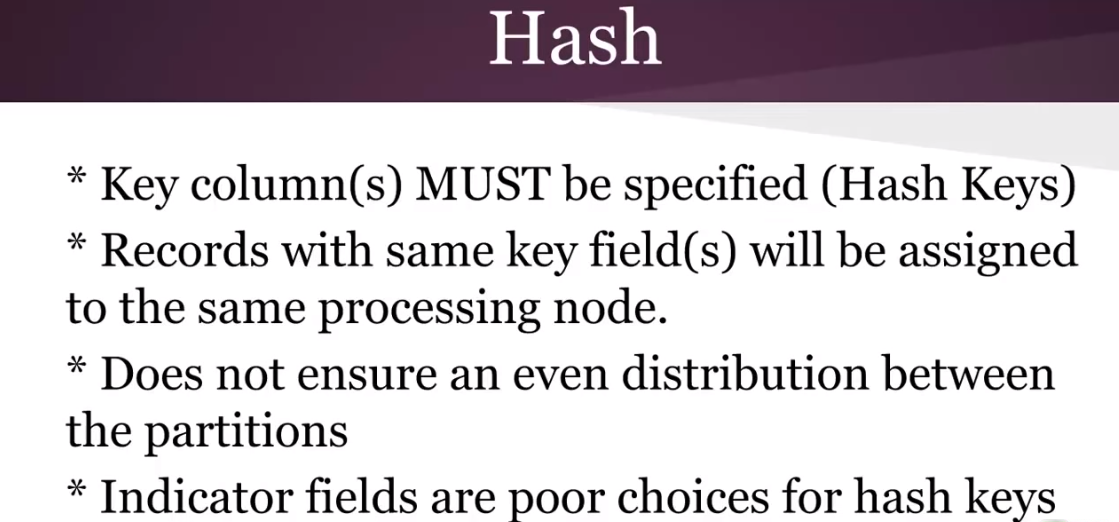
Entire: Used whenever lookup is used so data goes in reference as well.

Modulus: Not used generally.

Entire Modulus

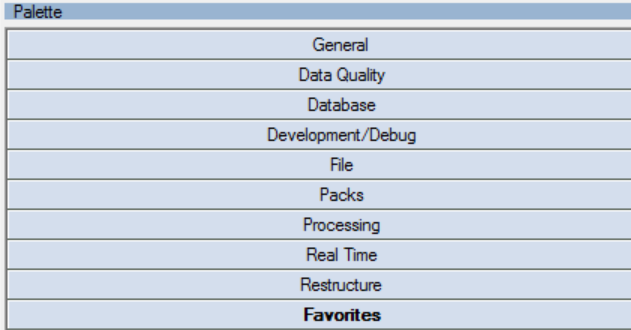






# Datastage parallel stages groups

* General objects
* Data Quality Stages
* Database connectors
* Development and Debug stages
* File stages
* Processing stages
* Real Time stages
* Restructure Stages



### General elements:

**Link:** It indicates a flow of the data. There are three main types of links in Datastage: stream, reference and lookup.

A link is a representation of a data flow that joins the stages in a job. A link connects data sources to processing stages, connects processing stages to each other, and also connects those processing stages to target systems. Links are like pipes through which the data flows from one stage to the next. Links are used to specify how the data flows from one stage to another.

Input link

An input link is displayed as a solid line, which indicates the primary flow of data. An input link is used to connect a data source to a stage so that data can be processed.

Output link

An output link is also displayed as a solid line, again indicating the primary flow of data. An output link is connected to a stage and generally moves processed data from the stage.

Reference link

A reference link is a specific type of input link, and is displayed as a dotted line, which indicates that table lookups are being performed. It is an input link on a Transformer or Lookup stage that defines where the lookup tables exist.

Reject link

A reject link is displayed as a dashed line, which indicates that records were rejected. Records are rejected when they do not meet the business logic of the job. A reject link is an output link that identifies errors when the stage is processing records and then routes rejected records to a target stage.

**Container**: Containers are reusable objects that hold user-defined groupings of stages and links. Containers create a level of reuse that allows you to use the same set of logic several times while reducing the maintenance. Containers make it easy to share a workflow, because you can simplify and modularize your job designs by replacing complex areas of the diagram with a single container.

Local container

A local container simplifies your job design. A local container can be used in only one job. However, you can have one or more local containers within a job.

Shared container

A shared container facilitate reuse. They can be used in many jobs. As with local containers, you can have one or more shared containers within a job.

**Annotation** is used for adding floating data stage job notes and descriptions on a job canvas.

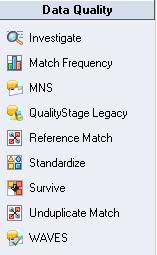
Annotation

You enter this text yourself and you can add as many of this type of annotation as required. Use it to annotate stages and links in your job design. These annotations can be copied and pasted into other jobs.

Description Annotation

You can add only one of these types of annotations for each job design. When you create a description annotation, you can choose whether the Description Annotation displays the full description or the short description from the job properties. Description Annotations cannot be copied and pasted into other jobs.

### Data Quality Stage:

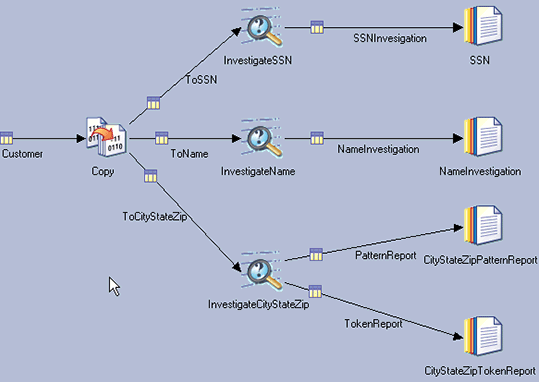


Investigate Stage:

The Investigate stage shows the actual condition of data in legacy sources and identifies and corrects data problems before they corrupt new systems. Investigation parses and analyzes free-form fields, counts unique values, and classifies or assigns a business meaning to each occurrence of a value within a field.

Investigation achieves these goals:

* Uncovers trends, potential anomalies, metadata discrepancies, and undocumented business practices.
* Identifies invalid or default values.
* Reveals common terminology.
* Verifies the reliability of fields that are proposed as matching criteria.
* The stage provides pre-built rule sets for investigating patterns on names and postal addresses for a number of different countries

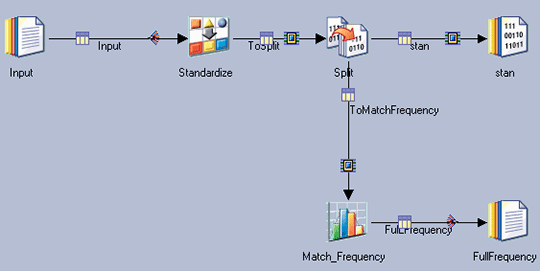


The Investigate stage takes a single input, which can be a link from any database connector, from a flat file or data set, or from any processing stage. Inputs to the Investigate stage can be fixed length or variable.

Match Frequency:

The Match Frequency stage gives you direct control over the disposition of generated frequency data. This stage provides results that can be used by the Match Designer and match stages, but enables you to generate the frequency data independent of running the matches.

You can generate frequency information by using any data that provides the fields that are needed by a match. Then you can let the generated frequency data flow into a match stage, store it for later use, or both.



In this example, input data is being processed in the Standardize stage with a rule set that creates consistent formats. The data is then split into two data streams. One stream passes data to a standard output and the other passes data to the Match Frequency stage.

Legacy Stage:

The Legacy stage converts earlier Quality Stage jobs to a compatible Parallel job's executable format.

Quality Stage Project. Displays the project name.

Quality Stage Job. Displays the job name.

Quality Stage grid. This grid displays the following information that can change depending on where in the job workflow the stage appears.

QS Stage: The name of the stage.

QS Stage Type: The type of stage in the former job.

QS File Flow: The input and output links.

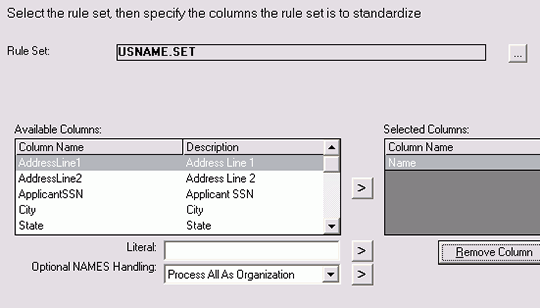
QS File Name: The names of the input and output links.

QS File Type: The type of files, such as data and result.

Standardize Stage:

You apply out-of-the-box rules with the Standardize stage to reformat data from multiple systems. This stage facilitates effective matching and output formatting.

The Standardize stage applies consistent representations, corrects misspellings, and incorporates business or industry standards. It formats data, places each value into a single domain field, and transforms data into a standard format.



Survive Stage:

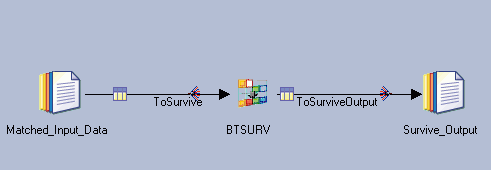
The Survive stage consolidates duplicate records, which creates a best-of-breed representation of the matched data.

Survivorship consolidates duplicate records, creating the best representation of the match data so companies can use it to load a master data record, cross-populate all data sources.

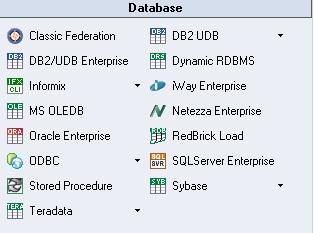
Supplies missing values in one record with values from other records on the same entity.

Populates missing values in one record with values from corresponding records which have been identified as a group in the matching stage.

Enriches existing data with external data.



### Database Connectors:



* **Oracle Enterprise**:  It allows you to read data from and write data to an Oracle database. It can also be used in conjunction with a Lookup stage to access a lookup table hosted by an Oracle database. The stage editor has up to three pages, depending on whether you are reading or writing a database:

Stage Page. This is always present and is used to specify general information about the stage.

Inputs Page. This is present when you are writing to a Oracle database. This is where you specify details about the data being written.

Outputs Page. This is present when you are reading from a Oracle database, or performing a lookup on an Oracle database. This is where you specify details about the data being read.

* **ODBC Enterprise** permits reading data from and writing data to a database defined as an ODBC source. In most cases it is used for processing data from or to Microsoft Access databases and Microsoft Excel spreadsheets.
* **DB2/UDB Enterprise** permits reading data from and writing data to a DB2 database.
* **Teradata permits** reading data from and writing data to a Teradata data warehouse. Three Teradata stages are available: Teradata connector, Teradata Enterprise and Teradata Multiload

The Stored Procedure stage supports calling Teradata macros, stored procedures (internal and external), scalar functions, and table functions. Aggregate functions are better suited to be invoked from the Teradata API stage, since the generated SELECT statement must query a table.

Although the Teradata API stage was not designed to invoke Teradata macros, stored procedures, and user-defined functions, it is possible to invoke such objects with the "User-defined SQL" option. However, there are several restrictions when invoking these objects from the Teradata API stage.

The Teradata API stage can only import a macro's input parameters into the Columns tab.

The Teradata API stage can return only one row from a macro invoked on the stage's reference link.

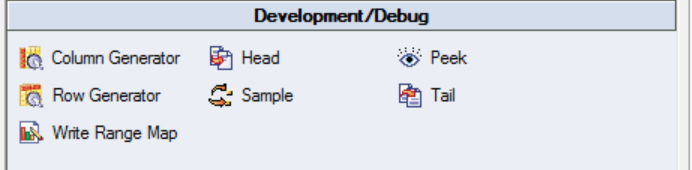
The Teradata API stage can return only one result set from a macro invoked on the stage's output link.

If a macro is invoked from the Teradata API stage's reference or output link, the first statement in the macro must be a SELECT statement.

The Teradata API stage only connects in ANSI transaction mode, so it can only call procedures that were compiled in ANSI transaction mode. Attempting to invoke a stored procedure that was compiled in Teradata transaction mode will return an error that says, "Invalid session mode for procedure execution".

* **SQL Server Enterprise** permits reading data from and writing data to Microsoft SQL Server database.
* **Sybase** permits reading data from and writing data to Sybase databases.
* Dynamic Relational Stage (Dynamic DBMS, DRS stage) is used for reading from or writing to a number of different supported relational DB engines using native interfaces, such as Oracle, Microsoft SQL Server, DB2, Informix and Sybase.

Development/Debug Stage:

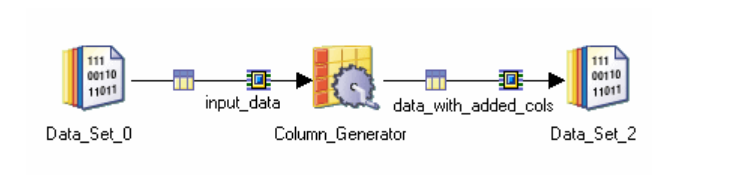


Column Generator:

The Column Generator stage is a Development/Debug stage. This stage adds columns to incoming data and generates mock data for these columns for each data row processed.

The Column Generator stage is a Development/Debug stage. It can have a single input link and a single output link.

The Column Generator stage adds columns to incoming data and generates mock data for these columns for each data row processed. The new data set is then output.

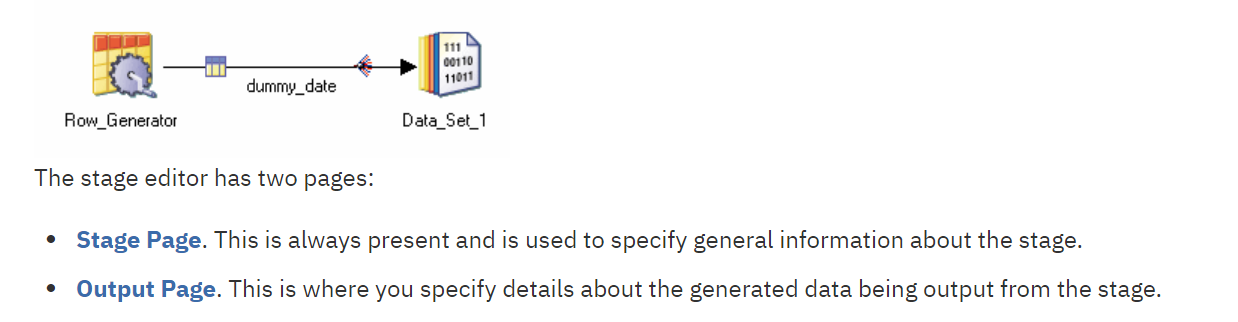


* [**Stage Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Stage_Page_column_generator_stage.html?view=kc). This is always present and is used to specify general information about the stage.
* [**Input Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Input_Page_column_generator_stage.html?view=kc). This is where you specify details about the input link.
* [**Output Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Outputs_Page_column_generator_stage.html?view=kc). This is where you specify details about the generated data being output from the stage

Row Generator:

The Row Generator Stage has no input links, and a single output link.

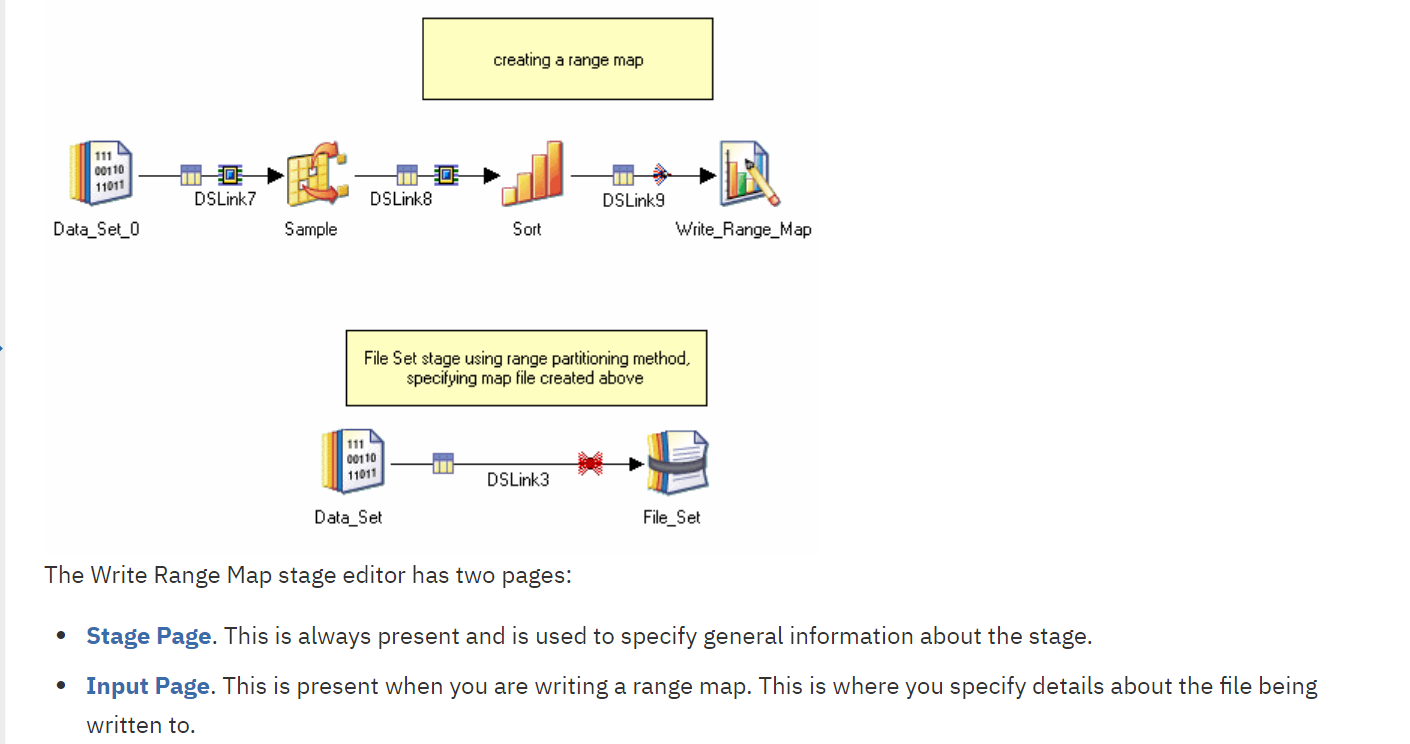
The Row Generator stage produces a set of mock data fitting the specified metadata. This is useful where you want to test your job but have no real data available to process.



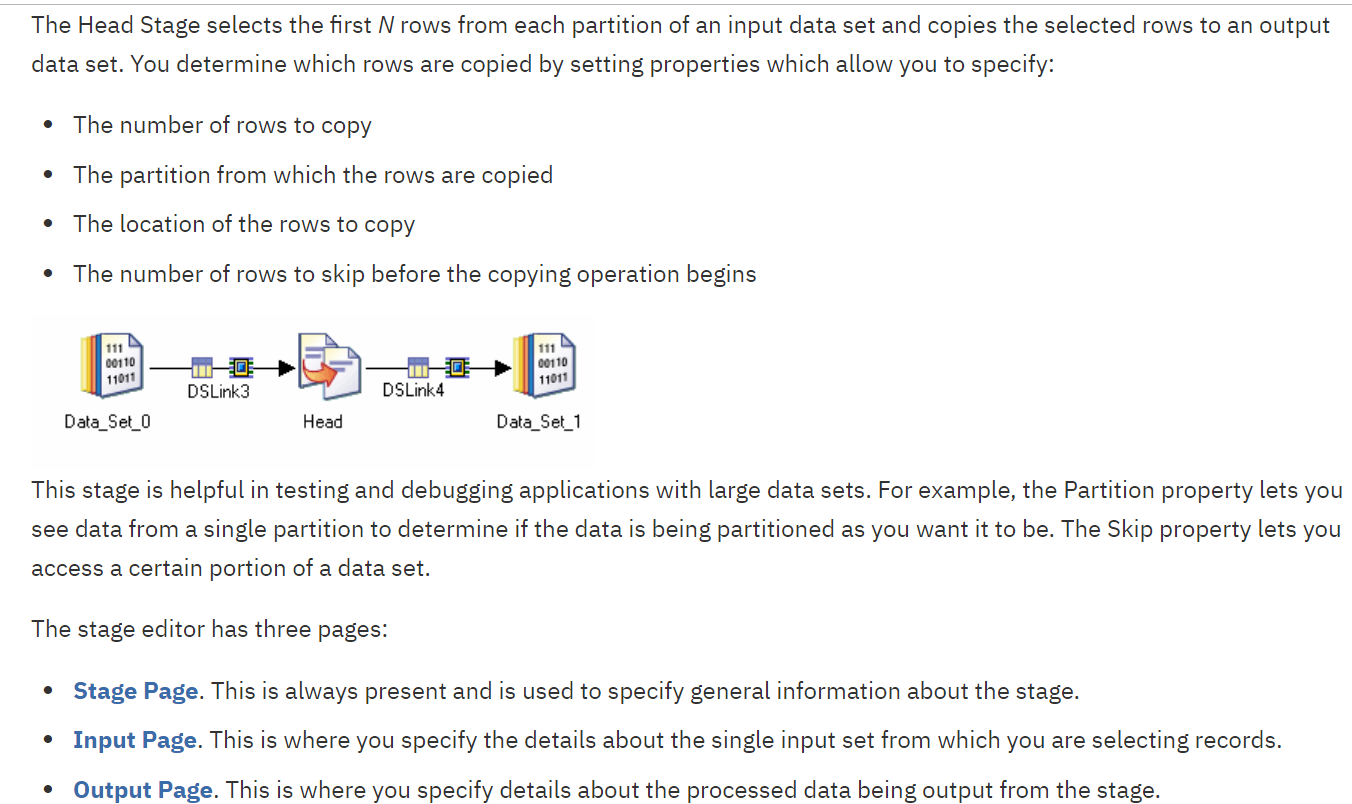
Write Range Map:

It allows you to write data to a range map. The stage can have a single input link. It can only run in sequential mode. The Write Range Map stage takes an input data set produced by sampling and sorting a data set and writes it to a file in a form usable by the range partitioning method. The range partitioning method uses the sampled and sorted data set to determine partition boundaries. .

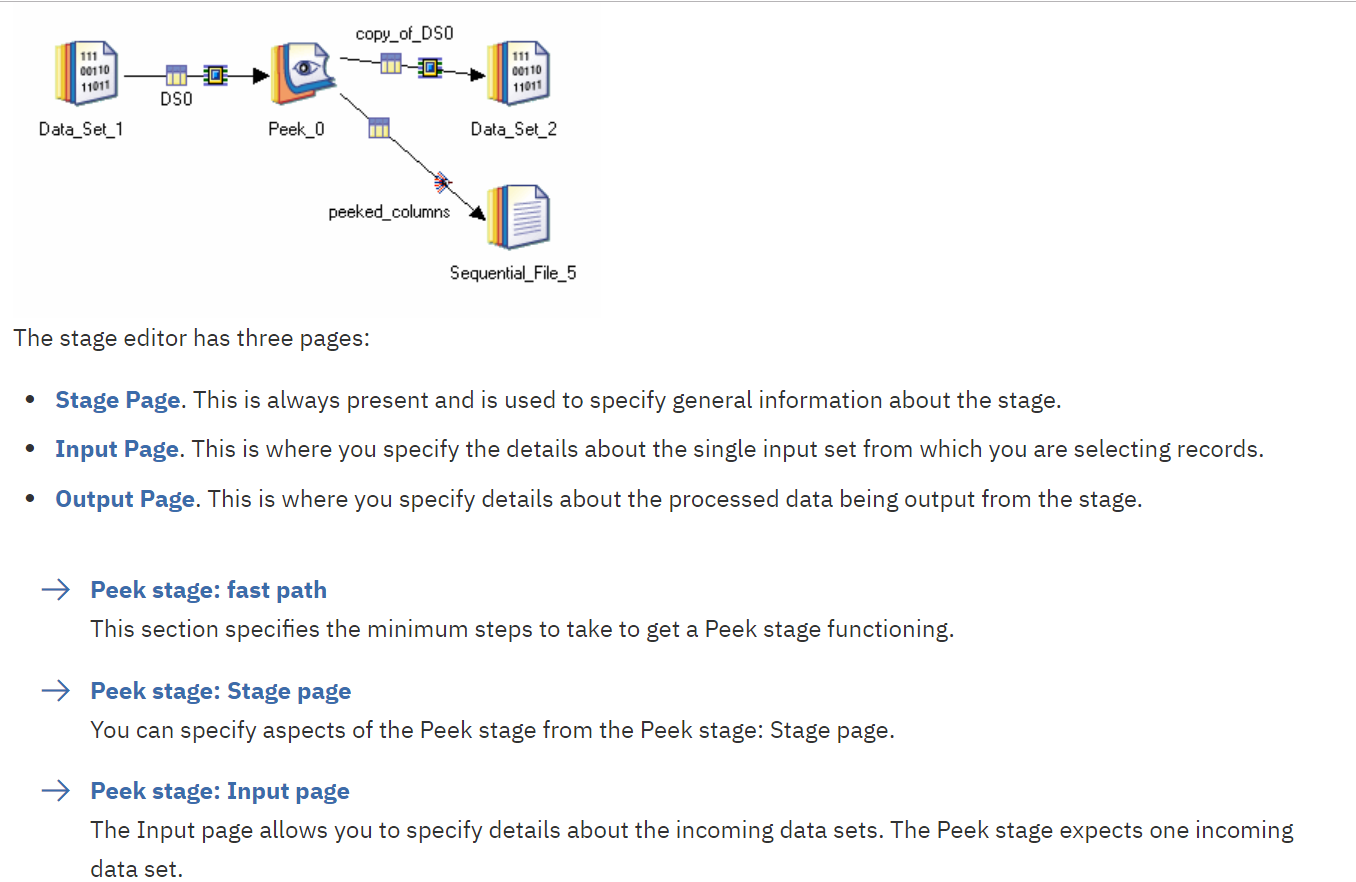
A typical use for the Write Range Map stage would be in a job which used the Sample stage to sample a data set, the Sort stage to sort it and the Write Range Map stage to write the range map which can then be used with the range partitioning method to write the original data set to a file set



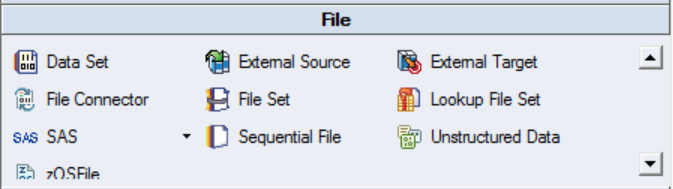
Head Stage:



Peek Stage: The Peek stage lets you print record column values either to the job log or to a separate output link as the stage copies records from its input data set to one or more output data sets.



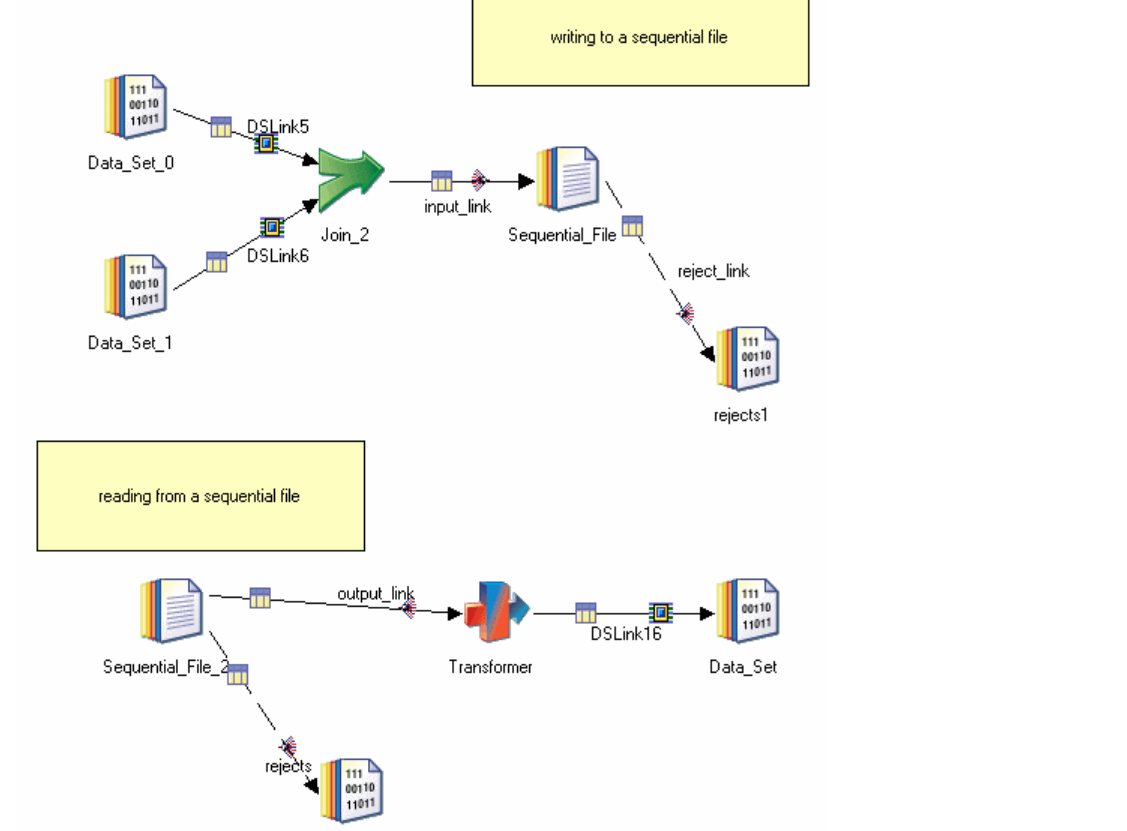
File Stage:



Sequential File Stage:

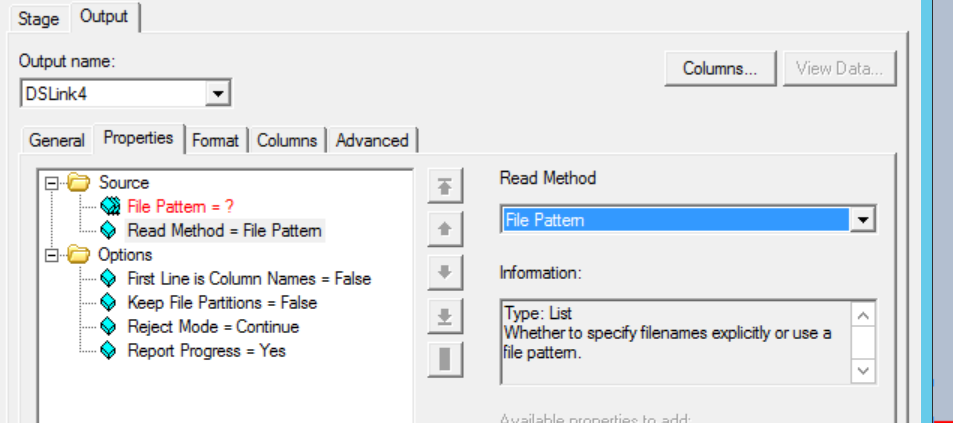
The Sequential File stage is a file stage that allows you to read data from or write data one or more flat files.

The stage can have a single input link or a single output link, and a single rejects link.



* [Stage Page](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Stage_Page_sequential_file_stage.html?view=kc). This is always present and is used to specify general information about the stage.
* [Input Page](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Inputs_Page_sequential_file_stage.html?view=kc). This is present when you are writing to a flat file. This is where you specify details about the file or files being written to.
* [Output Page](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_outputs_page.html?view=kc). This is present when you are reading from a flat file or have a reject link. This is where you specify details about the file or files being read from.

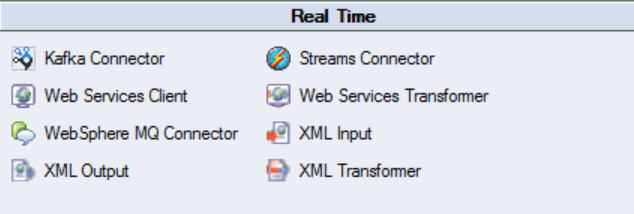
In case you have one file give specific file and name of the file but in case you have multiple files of same structure give file pattern and pattern name like Geo\_\*.txt



Lookup file set Stage:

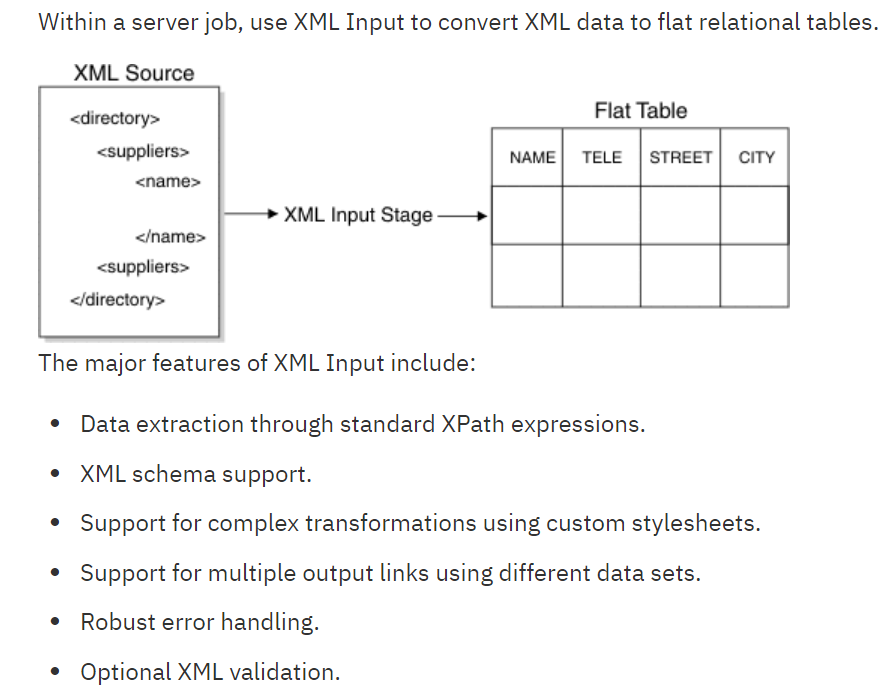
<https://www.ibm.com/support/knowledgecenter/en/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Lookup_File_Set_Stage.html>

Real Time Stage:

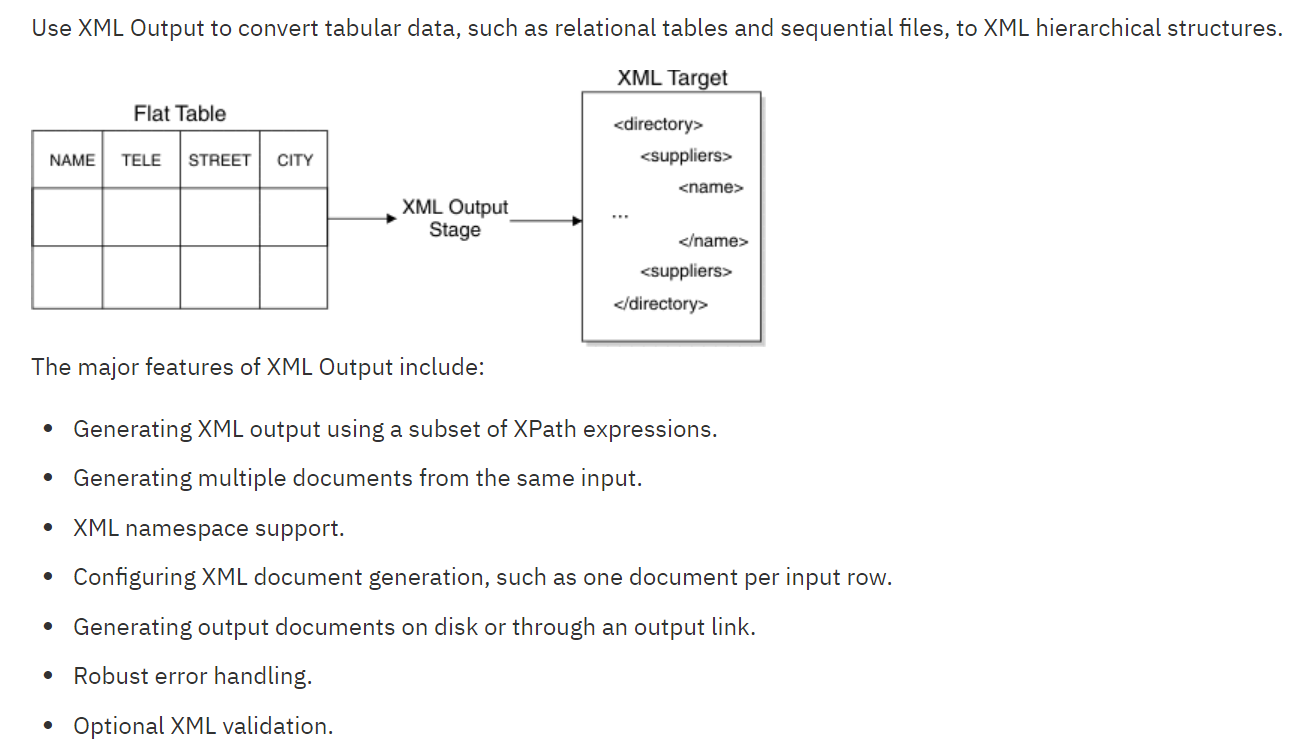


XML Input:

<https://www.ibm.com/support/knowledgecenter/en/SSZJPZ_11.5.0/com.ibm.swg.im.iis.ds.pack.xml.doc/topics/g_pack_xmldesigner_Using_XML_Input.html>



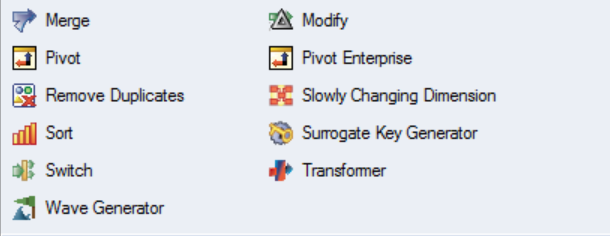
XML Output:



<https://datastage4u.wordpress.com/category/datastage-stages/>

Processing Stage:





Aggregator:

