Statistical Data Collection and Analysis of DataStage And Database Explorer

A Thesis submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

in

Artificial Intelligence

By

B. Bharath Kumar Reddy

Department of Computer and Information Sciences

University of Hyderabad

Hyderabad, India.

June - 2009

CERTIFICATE

This is to certify that the project report entitled "**Statistical Data Collection and Analysis of DataStage And Database Explorer**", submitted for partial fulfillment of the requirements for the award of Degree of Master of Technology in Artificial Intelligence (M.Tech (AI)) to the University of Hyderabad is a record of bonafide project work carried out by **Mr. B. Bharath Kumar Reddy (07MCMI15)** at India Software Labs, IBM India Pvt. Ltd., Hyderabad, for a period of one year during June 2008 to June 2009 under the guidance of Mr. Srinivas Kiran Mittapalli, Project Manager, IBM India Software Labs.

Dr. Vineet P Nair.,
Project Supervisor,
DCIS,
University of Hyderabad.

Mr. Srinivas Kiran Mittapalli,
Project Manager,
IBM India Software Labs,
Hyderabad.

Prof. Arun Agarwal
Head of the Department, DCIS,
University of Hyderabad,

Prof. T. Amaranath,
Dean (School of MCIS),
University of Hyderabad,

ACKOWLEDMENTS

I sincerely thank Prof**. Arun Agarwal**, Head, DCIS, University of Hyderabad for giving me an opportunity to work at IBM.

I thank **Dr. Rajeev Wankar**, Reader, DCIS, University of Hyderabad, due to which I am able to appear for this internship opportunity at IBM.

I thank **Dr. Vineet P Nair**, Lecturer, DCIS for his guidance. I wish to thank him for his support and guidance.

I thank all the faculty members of DCIS, UOH for their guidance. I wish to thank them all for being very patient, understanding and helpful.

I am very thankful to IBM India Pvt, Ltd., for offering me Internship at India Software Labs, Hyderabad.

I am very thankful to **Mr. Srinivas Kiran Mittapalli**, Project manager, IBM India Pvt Ltd., for offering me this project and for being so supportive, helpful and encouraging all the time. I wish to express my sincere gratitude to him.

I am very thankful to **Mrs. Malathi Sangineni** and **Mrs. Kiranmai Kota**. This project would not have been possible without their support, encouragement and continuous guidance. I wish to express my sincere gratitude to them.

I thank all the employees of IBM, India software Labs for being so friendly and helpful. They made me feel comfortable during my internship at IBM. I wish to thank them all for being very patient, understanding and helpful.

I thank my co-interns at IBM, for being helpful and encouraging throughout this internship period.

I thank **Dhyanamandali**, Yoga Institute, which helped me to grow spiritually and psychologically. I wish to express my gratitude to it.

I express my gratitude to my parents, sister and friends for their constant support and encouragement.

B. Bharath Kumar Reddy

To
God, My Parents
and
Dhyanamandali.

Abstract

This project report is divided into two parts. Part one discusses about the Statistical Data Collection and Analysis of the DataStage. Part two discusses about the Database Explorer and extending Teradata database to the Database Explorer.

Any operation that is performed in the DataStage consumes the resources such as Memory, CPU usage, IO usage and time. If the resources are not sufficient for the job which is running on DataStage then the job is aborted due to insufficient resources. One way to prevent the abortion of job due to lack of resources is by calculating the amount of the resources that the job consumes before the job initiates. To calculate the resources required, calculate the resources required for each Stage in DataStage. This is the Statistical Data Collection and Analysis of the DataStage. Part one of thesis describes about each Stage details and the amount of resources that are consumed by each stage of DataStage. Part two of thesis discusses about extending the Teradata Database to Database Explorer. Database Explorer is a tool which enables the users to connect to different types of databases from the single application.

Acronyms

CPU : Central Processing Unit

ETL : Extract Transform Load

SCD : Slowly Changing Dimension

FTP : File Transfer Protocol

API : Application Programming Interface

SQL : Structured Query Language

SMP : Symmetric Multi Processing

MPP : Massively Parallel

ODBC: Open Data Base Connectivity

Contents

List of Figures

PART -1

Statistical Data Collection and Analysis of DataStage

Chapter 1

Introduction to DataStage

Introduction

This chapter gives a brief description about, what is ETL, what is DataStage, features of DataStage, DataStage components, Stages in DataStage, Jobs in DataStage, source and target types supported by the DataStage and supported platforms.

1.1 ETL

ETL stands for Extract, Transform and Load. Extracting the data from the data sources, transform the data as required and load the output data to the Target. Reading data from source, cleaning it up and formatting it uniformly, and then writing it to the target repository to be exploited.

Figure 1.1: ETL flow

The input data used in ETL processes can be from any of these sources: a mainframe application, an ERP application, a CRM tool, a flat file, an Excel spreadsheet, or even a message queue. Extraction can be done via Java Database Connectivity, Open Database Connectivity technology or by creating flat files.

After extraction, the data is transformed, or modified, depending on the specific business logic involved so that it can be sent to the target repository. There are a variety of ways to perform the transformation, and the work involved varies according to requirement. The data may require reformatting only, but most ETL operations also involve cleansing the data to remove duplicates and enforce consistency. Part of what the software does is examine individual data fields and apply rules to consistently convert the contents to the form required by the target repository or application

For example, the category "male" might be represented in three different systems as M, male and 0/1. The ETL software would recognize that these entries mean the same thing and convert them to the target format.

After Transformation of the data to the required format, load the data to a target, so that it will be useful for future purpose. The target can be a Database table or a sequential file.

1.2Introduction to DataStage

DataStage is an Extraction, Transformation and Loading (ETL) tool works by extracting data from one or more operational databases. The data is transformed to eliminate inconsistencies, aggregated to summarize data, and loaded into the data warehouse. The end result is a stored database which contains stable, nonvolatile, integrated data. Therefore, we can see DataStage as an application which connects data sources to the targets and processes the data.

DataStage is a core component of the IBM WebSphere Data Integration Suite, which enables the user to tightly integrate enterprise information, despite having many sources or targets and short time frames. DataStage enables the user for building an enterprise data warehouse to support the information needs of the entire company, building a “real time” data warehouse, integrating dozens of source systems to support enterprise applications.

1.3 Features of DataStage

DataStage has the following features to aid the design and processing required for building a data warehouse:

1. Uses graphical design tools. With drag and drop techniques you can design a job.
2. The Source/Target file can be of any type and any number of data sources can be used. The type of the file can be Database table, Flat file or Dataset.
3. Access, transform and deliver information from multiple sources, in multiple formats, regardless of the volume of data or time frame required for processing.
4. You can define the Table definitions for files in a specified format, and you can view those table definitions at any point of the time in the jobs.

5. Transforms data. DataStage has a set of predefined transforms and functions used to convert source data.
6. The most comprehensive connectivity to source and target systems WebSphere DataStage supports a virtually unlimited number of heterogeneous data sources and targets in a single job.
7. DataStage can operate in real time, capturing messages or extracting data at a moment's notice on the same platform that also integrates bulk data.
8. DataStage includes Intelligent Assistants to make frequently used tasks easy to create.
9. The configuration file allows users to define the degree of parallelism without changes to application code.
10. DataStage director component enables the users viewing the job log. This is very useful to make changes to the job, when the job is not running properly.

1.4 DataStage Components

DataStage components can be categorized as two types. They are Client and Server components.

Client Components: DataStage client components are DataStage Administrator, DataStage Designer and DataStage Director.

DataStage Administrator is used to perform the administrative tasks, like adding or removing the users, projects or moving the projects, setting the environment variables at project level, setting the user defined variables at project level.

DataStage Designer is used to create the jobs. Each job consists of sources, targets and transformations required. These jobs are compiled to create the executables which run on the server.

DataStage Director is a user interface used to validate, schedule, run, and monitor DataStage server jobs and parallel jobs. These client tools connect to the DataStage server because all of the design information and metadata are stored on the server.

Administrator

Designer

Director

Figure 1.2: DataStage Client components

Server Components: Server components consist mainly of repository and DataStage server. Repository is a central store that contains all the information required to build a data mart or data warehouse. DataStage Server runs executable jobs that extract, transform, and load data into a data warehouse.

1.5 DataStage Stage

DataStage Stage is an operator which performs the specified work. The Stages in the DataStage are divided into six groups they are file stages, restructuring stages, processing stages, real time processing stages, development/debug and database stages.

In detail about the DataStage stages are discussed in the Chapter DataStage stages.

1.6 DataStage Jobs

DataStage jobs consist of individual stages. Each stage describes a particular processing this can be accessing a database or transforming data. For example, one stage may extract data from a data source, while another transforms it. Stages are added to a job and linked together using the Designer.

There are three kinds of jobs in DataStage, they are:

1. Server job
2. Parallel job
3. Mainframe job

Server jobs are compiled and run on the DataStage server. A server job will connect to databases on other machines as necessary, extract data, process it, and then write the data to the target data warehouse. These are for use on non-parallel systems and SMP systems.

Parallel jobs are developed using the DataStage Designer and compiled and run on the DataStage server. These types of jobs commonly connect to a data source, extract, and transform data and write it to a data warehouse. Parallel jobs must be compiled and run on a UNIX server, but you can develop them on a client attached to a Windows server. Once designed, parallel jobs can run on SMP, MPP, or cluster systems. Jobs are scaleable. The more processors you have, the faster the job will run.

Mainframe jobs are available only if you have Enterprise MVS Edition installed. A mainframe job is compiled and run on the mainframe. Data extracted by such jobs is then loaded into the data warehouse.

1.7 Supported Source and Target types

DataStage supports a virtually unlimited number of heterogeneous data sources and targets in a single job, including:

- Text files
- XML Files
- Complex data structures in XML
- Almost all types of databases, Informix, Sybase, Teradata, SQL Server, and the list goes on including access using ODBC
- Web services

1.8 Supported Platforms

IBM Information Server DataStage supports different types of platforms, they are:

- Windows NT, Windows 2000, Windows
- Server 2003
- IBM AIX
- HP Compaq Tru64
- HP HP-UX
- Red Hat Enterprise Linux AS
- Sun Solaris

Chapter 2

Tuning parameters of DataStage

Introduction

The Configuration file and the Partitioning techniques are used to run the DataStage job under different environmental conditions. This Chapter discusses about these topics.

2.1 The Key for Parallelism :: Configuration File

The Configuration file is the one which plays the key role in DataStage job development and on the processing of job. For running the DataStage job, DataStage must be aware of the resources that a job can use while processing. All these resources specifications are defined in the Configuration File. The configuration file gives the shape and size of the System to the DataStage. So that the DataStage organizes the resources required for a job according configuration file.

The configuration file describes available processing power in terms of processing nodes. These may not correspond to the actual number of processors in the system that DataStage is installed. The number of nodes you define in the configuration file determines how many instances of a process will be produced when you compile a parallel job. By using the Configuration file, it is possible to leave a couple of processors free to deal with other activities on your system.

In DataStage, there won't be necessarily one ideal configuration file for a given system because of the high variability between the way different jobs work. For this reason, multiple configuration files should be used to optimize overall throughput and match job characteristics to available hardware resources. At runtime, the configuration file is specified through the environment variable $APT_CONFIG_FILE. The Configuration file cab be changed by using the same environment variable. The number of nodes exist in default configuration file are half of the number physical processors on the system.

One of the great strengths of the DataStage Enterprise Edition is that, when designing jobs, you don't have to worry too much about the underlying structure of your system, beyond appreciating its parallel processing capabilities. If your system changes,

upgraded or improved, or if you develop a job on one platform and implement it on another, you don't necessarily have to change your job design, just change the configuration file.

The configuration file also gives you control over parallelization of your job during the development cycle. For example, by editing the configuration file, you can first run your job on a single processing node, then on two nodes, then four, then eight, and so on.

The default configuration file has the following characteristics:

- number of nodes = ½ number of physical CPUs
- disk and scratchdisk storage use subdirectories within the DataStage install file system

Syntactic characteristics of Configuration file:

- Braces { } begin and end the file.
- The word node begins every node definition.
- The word node is followed by the name of the node enclosed in quotation marks.
- Braces { } follow the node name. They enclose the information about the node (its options), including an enumeration of each disk and scratch disk resource. The legal options are: fastname, pools, and resource.
- Spaces separate items.
- Quotation (") marks surround the attributes you assign to options, that is, the names of nodes, disks, scratch disks, and pools.
- Comments are demarcated by /* . . . */, in the style of the C programming language. They are optional, but are recommended where indicated in the examples.

In the Configuration file design, variables are used to allocate the resources. The brief description of those variables is given below.

Node: The node's name is typically the network name of a processing node. Each node defined is followed by its name enclosed in quotation marks.

```
node "orch0"
```

Fastname: This option takes as its quoted attribute the name of the node as it is referred to on the fastest network in the system. The fastname is the physical node name that stages use to open connections for high volume data transfers.

`fastname "`*name*`"`

Pools: The pools option indicates the names of the pools to which this node is assigned. The option's attribute is the pool name or a space-separated list of names, each enclosed in quotation marks.

Syntax**:** `pools "`*node_pool_name0*`" "`*node_pool_name1*`" ...`

A node can be assigned to multiple pools as in the following example, where node is assigned to the default pool ("") as well as the pools node1, node1_css, and pool4.

Example: `pools "" "node1" "node1_css" "pool4"`

Resource: This indicates the resources that the nodes uses for the temporary storage of the data.

Syntax**:**

`resource` *resource_type* `"`*location*`"[{pools` *disk_pool_name*`"}]`

`|`

`resource` *resource_type* `"`*value*`"`

The resource type can be scratch disk or disk. For the resource type disk the syntax is:

Syntax**:** `resource disk "`*directory_path*`" [{pools "`*poolname*`"...}]`

Assign to this option the quoted absolute path name of a directory belonging to a file system connected to the node. The node reads persistent data from and writes persistent data to this directory. One node can have multiple disks.

For the resource type scratch disk it will be like this. The Syntax**:**

`resource scratchdisk "`*directory_path*`" [{pools` *poolname*`"...}]`

Assign to this option the quoted absolute path name of a directory on a file system where intermediate data will be temporarily stored.

The Sample configuration file is given below.

```
{
        node "node1"
        {
```

```
                fastname "HYDDEV121"
                pools ""
                resource disk "C:/IBM/InformationServer/Server/Datasets" {pools ""}
                resource  scratchdisk  "C:/IBM/InformationServer/Server/Scratch"  {pools
""}
        }
        node "node2"
        {
                fastname "HYDDEV121"
                pools ""
                resource disk "C:/IBM/InformationServer/Server/Datasets" {pools ""}
                resource  scratchdisk  "C:/IBM/InformationServer/Server/Scratch"  {pools
""}
        }
        node "node3"
        {
                fastname "HYDDEV121"
                pools ""
                resource disk "C:/IBM/InformationServer/Server/Datasets" {pools ""}
                resource  scratchdisk  "C:/IBM/InformationServer/Server/Scratch"  {pools
""}
        }
        node "node4"
        {
                fastname "HYDDEV121"
                pools ""
                resource disk "C:/IBM/InformationServer/Server/Datasets" {pools ""}
                resource  scratchdisk  "C:/IBM/InformationServer/Server/Scratch"  {pools
""}
        }
        }
```

2.2 Partitioning, Repartitioning, and Collecting Data

Partitioning: When you are using multiple nodes in Job running, the input data must be sent to all the nodes. This process of sending the input data to all the nodes in the system is called as Partitioning of data. The aim of most partitioning operations is to end up with a set of partitions that are as near equal size as possible, ensuring an even load across all processors.

Collecting: Collecting is the process of combining multiple partitions of a single data set to a single file.

Repartitioning: Repartitioning is partitioning already partitioned data. This type of repartitioning is required when, for example at the sort stage you have partitioned the data based on the first name column of data, after that at the aggregator stage you need to partition the output data from the sort stage based on the age column. This type of partitioning is called as repartitioning of the data.

There are different methods are available for partitioning and collecting of data, these are described below.

2.2.1 Partitioning

2.2.1.1 Round robin

In this partitioning mechanism, the first record goes to first processing node, second one goes to second partitioning node, and so on. When the last processing node is reached, it again starts from the first node, until all the records are completed from input data source. This method is useful for resizing partitions of an input data set that are not equal in size. The round robin method always creates approximately equal-sized partitions.

2.2.1.2 Random

In Random partitioning, the records are randomly distributed across all processing nodes. Like round robin, random partitioning can rebalance the partitions of an input data set to guarantee that each processing node receives an approximately equal sized partition.

2.2.1.3 Same

When the records need to be transferred from one stage to the next stage with out partitioning, then this partitioning mechanism is used. With this partitioning method, records stay on the same processing node; that is, they are not redistributed. Same is the fastest partitioning method. This is normally the method DataStage uses when passing data between stages in your job.

2.2.1.4 Entire

Each record of the input data is copied on to, every node. It is useful when you want the benefits of parallel execution, but every instance of the operator needs access to the entire input dataset.

2.2.1.5 Hash By Field

Partitioning is based on a function of one or more columns (the hash partitioning keys) in each record. The hash partitioner examines one or more fields of each input record (the hash key fields). Records with the same values for all hash key fields are assigned to the same processing node. This method is useful for ensuring that related records are in the same partition, which may be a prerequisite for a processing operation. Hash partitioning does not necessarily result in an even distribution of data between partitions.

2.2.1.6 Modulus

Partitioning is based on a key column modulo the number of partitions. This method is similar to hash by field, but involves simpler computation. The modulus partitioner assigns each record of an input data set to a partition of its output data set as determined by a specified key field in the input data set. This field can be the tag field.

The partition number of each record is calculated as follows:

partition_number = *fieldname* mod *number_of_partitions*

Where:

- Fieldname is a numeric field of the input data set.

- number_of_partitions is the number of processing nodes on which the partitioner executes. If a partitioner is executed on three processing nodes it has three partitions.

2.2.1.7 Range

Divides a data set into approximately equal-sized partitions, each of which contains records with key columns within a specified range. This method is also useful for ensuring that related records are in the same partition. A range partitioner divides a data set into approximately equal size partitions based on one or more partitioning keys.

2.2.1.8 DB2

Partitions an input data set in the same way that DB2 would partition it. For example, if you use this method to partition an input data set containing update information for an existing DB2 table, records are assigned to the processing node containing the corresponding DB2 record. Then, during the execution of the parallel operator, both the input record and the DB2 table record are local to the processing node. Any reads and writes of the DB2 table would entail no network activity.

2.2.1.9 Auto

The most common method you will see on the DataStage stages is Auto. This just means that you are leaving it to DataStage to determine the best partitioning method to use depending on the type of stage, and what the previous stage in the job has done. Typically DataStage would use round robin when initially partitioning data, and same for the intermediate stages of a job.

2.2.2 Collecting

2.2.2.1 Round robin

Reads record from the first partition, and then from the second partition, and so on. After reaching the last partition, starts over. If any partition becomes empty then it skips that partition and goes to the next partition.

2.2.2.2 Ordered

Reads all records from the first partition, then all records from the second partition, and so on. This collection method preserves the order of totally sorted input data sets. In a totally sorted data set, both the records in each partition and the partitions themselves are ordered. This may be useful as a preprocessing action before exporting a sorted data set to a single data file.

2.2.2.3 Sorted Merge

Read records in an order based on one or more columns of the record. The columns used to define record order are known as collecting keys. You use this sorted merge collector with a partition-sorted data set (as created by a sort stage). In this case, you specify as the collecting key fields those fields you specified as sorting key fields to the sort stage.

2.2.2.4 Auto

The most common method you will see on the DataStage stages is Auto. This normally means that DataStage will eagerly read any row from any input partition as it becomes available, but if it detects that, for example, the data needs sorting as it is collected, it will do that. This is the fastest collecting method.

Chapter 3

Performance Monitor

Introduction

Performance Monitor is also called as Perfmon tool. This tool is used in the Statistical Data Collection and Analysis of the DataStage to get the values of CPU usage, Memory usage, IO usage. This Chapter discusses about this tool.

3.1 Perfmon

Performance Monitor (perfmon) is a powerful tool available in all Microsoft operating systems from NT to Windows Server 2008. This tool is used for monitoring the performance of applications. Performance Monitor has approximately 350 different counters. The Performance Monitor displays graphical representation of system performance.

Enter *perfmon* in the command line editor to open the tool. The primary default view allows objects to be graphically displayed. This view enables you to view the monitored items over a short period of time (as short as every second) by choosing the options from with in the dialog boxes. This view is best for actively watching the performance of a system.

3.2 Adding the Counters

From the default view, you can add the counters by right clicking, and selecting the add counters button. First select the system, whether it is a local system or a remote system by selecting the radio buttons. Then select an item from the performance object. On selecting the performance object, the counters that correspond to the performance object are displayed in the counters list and instances list.

To get a more detailed explanation of any counter, right click anywhere in the perfmon graph and choose Add Counters. Select the counter and object that you are curious about, and click the Explain button.

3.3 Counters used in Analysis

Processor utilization: Processor\% Processor Time_Total, this gives the total processor utilization of the system.

Memory utilization: Process\Working Set_Total, this basically shows how much memory is in the *working set*, or currently allocated RAM. This gives the memory utilization of the system.

Disk Utilization: PhysicalDisk\Bytes/sec_Total, shows the number of bytes per second being written to or read from the disk. This gives the Disk utilization of the system.

3.3 Uses

From this single tool you can track a range of system processes and give real time graphical display of results. You can customize the data you want to collect in logs by defining thresholds for alerts and automatic actions, generate reports, and view past performance data. This tool can also be used to plan upgrades, send out alerts, tracking processes to be optimized, monitor results of tuning and configuration scenarios, and understand system workload and its effect on resource usage.

Chapter 4

DataStage Stages

Introduction

A Stage in the DataStage is an operator that performs the specified task. DataStage uses Stages in designing the job. This Chapter describes about these Stages used in DataStage

4.1 DataStage Stages

The DataStage stages are divided into six groups depending on the type of the work. The Stages can be divided as:

1. File Stages
2. Processing Stages
3. Restructure
4. Development/Debug
5. Database Stages
6. Real Time Processing stages

In brief these can be described as:

File stages: These stages are used as the source/target stages in the job design. These are the stages that read or write data contained in a file or set of files. Examples of file stages are the Sequential File and Data Set stages.

Processing Stages: These are stages that perform processing on the data. Examples of processing stages are the Aggregator and Transformer stages.

Restructuring Stages: These are stages that deal with and manipulate data containing columns of complex data type. These include Column Import, Column Export etc.,.

Development/Debug: These are stages that help you when you are developing and troubleshooting parallel jobs. Examples are the Peek and Row Generator stages.

Database: These are stages that read or write the data in a database. Examples of database stages are the Oracle Enterprise and DB2/UDB Enterprise stages.

Real Time Processing Stages: These are the stages that allow Parallel jobs to be made available as real time services. They comprise the source and target stages. These are part of the Web Services package. Examples are XML pack and Java pack.

4.1.1 File Stages

4.1.1.1 Sequential File Stage

Figure 4.1: Sequential File

The Sequential File stage is a file stage. This 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.

The stage executes in parallel mode if reading multiple files but executes sequentially if it is only reading one file. By default a complete file will be read by a single node (although each node might read more than one file). The stage executes in parallel if writing to multiple files, but executes sequentially if writing to a single file. Each node writes to a single file, but a node can write more than one file.

4.1.1.2 Dataset Stage

Figure 4.2: Dataset Stage

The Data Set stage is a file stage. It allows you to read data from or write data to a data set. The stage can have a single input link or a single output link. It can be configured to execute in parallel or sequential mode.

DataStage parallel extender jobs use data sets to manage data within a job. The Data Set stage allows you to store data being operated on in a persistent form, which can then be used by other DataStage jobs. Data sets are operating system files, each referred to by a control file, which by convention has the suffix .ds. Using data sets wisely can be key to good performance in a set of linked jobs.

4.1.1.3 Lookup File Set Stage

Figure 4.3: Lookup File Stage

The Lookup File Set stage is a file stage. It allows you to create a lookup file set or reference one for a lookup. The stage can have a single input link or a single output link. The output link must be a reference link. The stage can be configured to execute in parallel or sequential mode when used with an input link.

4.1.2 Processing Stages

4.1.2.1 Aggregator Stage

Figure 4.4: Aggregator Stage

The Aggregator stage is a processing stage. It classifies data rows from a single input link into groups and computes totals or other aggregate functions for each group. The summed totals for each group are stored in output stage.

The aggregator stage gives access to grouping and summary operations. One of the easiest ways to expose patterns in a collection of records is to group records with similar characteristics, then compute statistics on all records in the group. These statistics can be used to compare properties of the different groups.

4.1.2.2 Join Stage

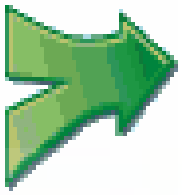


Figure 4.5: Join Stage

The Join stage is a processing stage. It performs join operations on two or more data sets input to the stage and then outputs the resulting data set. This stage can have any number of input links but only one output link.

In Join Stage the input datasets are identified as Left, Right and Intermediate data sets. The Join stage can perform four types of operations on the input links, they are Inner Join, Left Outer Join, Right Outer Join and Full Outer Join.

The Input datasets to the Join stage must be sorted and key partitioned, this ensures that the records with the same key column values will be transferred to the same partition and will process on one node only. For sorting of the records the Join stage uses the Scratch disk. Make sure the size of the scratch disk is sufficient to store the data.

4.1.2.3 Merge Stage

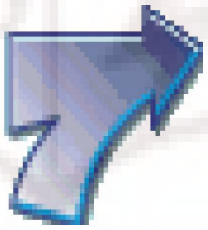


Figure 4.6: Merge Stage

The Merge stage is a processing stage. It can have any number of input links, the input links are called as Master link and Update links. It can have a single output link, and the same number of reject links as there is number of update links. The input datasets are called as Master Dataset and Update dataset.

The columns of the Master and Update datasets are combined, so as the output link have all the columns from Master and Update links. The records from Master and Update links are merged when both the records are having the same key columns. Same as Join stage, here the input records must be key partitioned and sorted, so that the input

records with the same column names will end up in same partition and be processed on the same node.

4.1.2.4 Lookup Stage

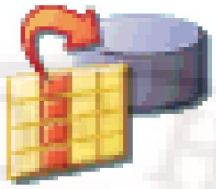


Figure 4.7: Lookup Stage

The Lookup stage is a processing stage. It is used to perform lookup operations on a data set read into memory from any other Parallel job stage that can output data.

The Lookup stage can have a reference link, a single input link, a single output link, and a single reject link. Depending upon the type and setting of the stage providing the look up information, it can have multiple reference links (where it is directly looking up a DB2 table or Oracle table, it can only have a single reference link). A lot of the setting up of a lookup operation takes place on the stage providing the lookup table.

For each record of the source data set from the primary link, the Lookup stage performs a table lookup on each of the lookup tables attached by reference links. In the stage editor a condition can be specified on each of the reference links, such that the stage will only perform a lookup on that reference link if the condition is satisfied.

The three stages (Join, Merge and Lookup) differ mainly in the memory they use, the treatment of rows with unmatched keys, and their requirements for data being input (for example, whether it is sorted).

4.1.2.5 Sort Stage

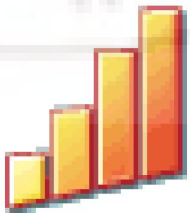


Figure 4.8: Sort Stage

The Sort stage is a processing stage. It can have one Input link and one Output link. In the stage editor page of sort stage sorting keys are specified. These keys are used to perform the sort on the specified column. There are two types of keys, they are Primary key and Secondary key. The column you specify first as a key to the stage is the primary key. After specifying the primary key you can specify different columns as keys. These are called as secondary keys. The data will be first sorted according to the primary key, and if multiple records matches with the same primary key values, then it is sorted according to the secondary key.

The output of the sort stage is highly dependent on the type of the job you are running (Sequential or Parallel) and on the partitioning mechanism you used. If you specify the job to be run sequentially then whole data will be sorted, but if you specify parallel then data will be sorted for each partition and will be passed through the output link.

The sort stage uses the temporary disk space for performing the sorting operation. For that case this uses the following locations in the following order, they are:

1. Scratch disks in the disk pool sort (you can create these pools in the configuration file).
2. Scratch disks in the default disk pool (scratch disks are included here by default).
3. The directory specified by the TMPDIR environment variable.
4. The directory: */tmp*.

4.1.2.6 Funnel Stage

Figure 4.9: Funnel Stage

The Funnel stage is a processing stage. It copies multiple input data sets to a single output data set. This operation is useful for combining separate data sets into a single large data set. The stage can have any number of input links and a single output link.

The Funnel stage operates in one of following three ways, for which all the input data sets must have the same metadata. They are:

Continuous Funnel: In this type there is no guaranteed order for combining the records. This takes one record from each input link. If data is not available on one input link, the stage skips to the next link.

Sort Funnel: This combines the input records in the order defined by the value(s) of one or more key columns and the order of the output records is determined by these sorting keys.

Sequence Funnel: This copies all records from the first input data set to the output data set, then all the records from the second input data set, and so on.

4.1.2.7 Copy Stage

Figure 4.10: Copy Stage

The Copy stage is a processing stage. It can have a single input link and any number of output links. The Copy stage copies the input data source to the any number of output targets. The input data set can be copied as it is, or you can make the changes to the dataset such as changing the order of the columns or dropping columns which are not required.

The only property that you must set in Copy stage when you are copying from a single source to single stage is that Force property to TRUE. This prevents DataStage from deciding that the Copy operation is superfluous and optimizing it out of the job. How ever for a single source to multiple destinations there is no need to set the Force property.

4.1.2.8 Filter Stage

Figure 4.11: Filter Stage

The Filter stage is a processing stage. It can have a single input link and any number of output links and, optionally it can have a single reject link.

The name itself states that Filter stage filters the input data records. The Filtering is based on the conditions specified, for each output link. You can specify different conditions for different output links. The records which satisfy the condition will output to the target stage. The records which do not satisfy the requirements can be output to the reject link specified. You can specify the condition to be on an input column also.

The operation of the filter stage is governed by the expressions you set in the Where property on the Properties Tab. The elements that can be used to specify the conditions are:

- Input columns.
- Requirements involving the contents of the input columns.
- Optional constants to be used in comparisons.
- The Boolean operators AND and OR to combine requirements.

4.1.2.9 Switch Stage

Figure 4.12 Switch Stage

The Switch stage is a processing stage. It can have one input link and output links up to 128 and a single reject link.

The Switch stage takes each input record and assigns it to an output data set based on the value of a selector field. The Switch stage performs same as the Switch statement

in C language, which causes the flow of control in a C program to branch to one of several cases based on the value of a selector variable. As same as default case, here the records which do not satisfy any of the conditions will be transferred to reject link.

In the Switch stage you specify on which column the Selector variable should apply by specifying a column name, and you can specify cases, for what condition the input record should be transferred to what output link.

4.1.2.10 Change Capture Stage

Figure 4.13: Change Capture Stage

The Change Capture stage is a processing stage. It can have two input links and one output link.

The two input links that are input to the Change Capture stage denoted as before and after datasets, and the output represents the changes made to the before dataset to obtain the after dataset. The stage produces a change data set, whose table definition is transferred from the after data set's table definition with the addition of one column: a change code with values encoding the four actions: insert, delete, copy, and edit.

The compare is based on a set a set of key columns, rows from the two data sets are assumed to be copies of one another if they have the same values in these key columns. If the two input records have the same key values then you can check whether one is the edited copy of the other.

The input data must be key partitioned and must be in ascending order. For sorting the input data you can use sort stage in between Change capture stage or you can inbuilt sorting function in inputs partitioning tab of input dataset.

4.1.2.11 Change apply stage

Figure 4.14: Change Apply Stage

The Change Apply stage is a processing stage. It can have two input links and one output link.

This takes the change data set, that contains the changes in the before and after data sets, from the Change Capture stage and applies the encoded change operations to a before data set to compute an after data set. The before input to Change Apply must have the same columns as the before input that was input to Change Capture, and an automatic conversion must exist between the types of corresponding columns. The change input to Change Apply must have been output from Change Capture without modification. The Change Apply stage reads a record from the change data set and from the before data set, compares their key column values, and acts accordingly.

4.1.2.12 Remove Duplicates Stage

Figure 4.15: Remove Duplicates Stage

The Remove Duplicates stage is a processing stage. It can have a single input link and a single output link.

The Remove Duplicates stage takes a single sorted data set as input, removes all duplicate rows, and writes the results to an output data set. Two rows are considered duplicates if they are adjacent in the input data set and have identical values for the key columns. Key columns are any columns you designate for determining two rows are identical. The data set input to the Remove Duplicates stage must be sorted so that all

records with identical key values are adjacent. This can be accomplished by inserting a sort stage between Remove duplicates stage and the input dataset or by using the sort facilities available on the Inputs page Partitioning tab.

4.1.2.13 Compress Stage

Figure 4.16: Compress Stage

The Compress stage is a processing stage. It can have a single input link and a single output link.

The Compress stage uses the UNIX compress or GZIP utility to compress a data set. It converts a data set from a sequence of records into a stream of raw binary data. A compressed data set is similar to an ordinary data set and can be stored in a persistent form by a Data Set stage. However, a compressed data set cannot be processed by many stages until it is expanded, that is, until its rows are returned to their normal format. Stages that do not perform column-based processing or reorder the rows can operate on compressed data sets. As compressing a data set removes its normal record boundaries, the compressed data set must not be repartitioned before it is expanded.

4.1.2.14 Expand Stage

Figure 4.17: Expand Stage

The Expand stage is a processing stage. It can have a single input link and a single output link.

The input to the Expand stage must be the data that is compressed previously by the Compress stage. The Expand stage uses the UNIX uncompress or GZIP utility to expand a data set. It converts a previously compressed data set back into a sequence of records from a stream of raw binary data.

4.1.2.15 Encode Stage

Figure 4.18: Encode Stage

The Encode stage is a processing stage. It can have a single input link and a single output link.

It encodes a data set using a UNIX encoding command, such as gzip, that you supply. The stage converts a data set from a sequence of records into a stream of raw binary data. As same that of Compress stage output, an encoded data set is similar to an ordinary one, and can be written to a data set stage. You cannot use an encoded data set as an input to stages that performs column-based processing or re-orders rows.

4.1.2.16 Decode Stage

Figure 4.19: Decode Stage

The Decode stage is a processing stage. It can have a single input link and a single output link.

The input to the decode stage must be the output of the Encode stage. It decodes a data set using a UNIX decoding command, such as gzip, that you supply. It converts a data stream of raw binary data into a data set.

4.1.2.17 Difference Stage

Figure 4.20: Difference stage

The Difference stage is a processing stage. It can have two input links and one input link.

The two input datasets are designated as before and after data sets. The Difference stage outputs a single data set whose records represent the difference between them. The comparison is performed based on a set of difference key columns. Two records are copies of one another if they have the same value for all difference keys. If two records have identical key columns, you can compare the value columns to see if one is an edited copy of the other.

The stage assumes that the input data sets have been key-partitioned and sorted in ascending order on the key columns you specify for the Difference stage comparison. You can achieve this by using the Sort stage or by using the built in sorting and partitioning abilities of the Difference stage.

4.1.2.18 Compare Stage

Figure 4.21: Compare Stage

The Compare stage is a processing stage. It can have two input links and a single output link.

The Compare stage performs a column-by-column comparison of records in two presorted input data sets. You can restrict the comparison to specified key columns. The Compare stage does not change the table definition, partitioning, or content of the records in either input data set. It transfers both data sets intact to a single output data set generated by the stage. The comparison results are also recorded in the output data set.

The stage outputs a data set with three columns:

- **Result:** Carries the code giving the result of the comparison.
- **First:** A sub record containing the columns of the first input link.
- **Second:** A sub record containing the columns of the second input link.

4.1.2.19 Transformer Stage

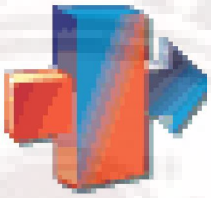


Figure 4.22: Transformer Stage

The Transformer stage is a processing stage. Transformer stages allow you to create transformations to apply to your data. These transformations can be simple or complex and can be applied to individual columns in your data. Transformations are specified using a set of functions.

The set of functions can be divided into groups based on the type of work they do. They are

- Date and Time Functions: These functions deal with the date and time details. For example Calculating the Days from the given date, calculating the duration from given time.
- Logical Functions: These are logical functions which will be applied on sets of data. These include AND, OR, NOT etc.,.
- Mathematical Functions: These functions deals with mathematical operations on data. Examples are COS, SIN, EXP, LOG etc.,.
- Null Handling Functions: These deals with null properties of data. Examples are ISNull, IsNotNull etc.,.
- Number Functions: These are applied on Numbers. Examples are Mantissa from Decimal, Mantissa from Float.
- String Functions: These deals with the String properties. Examples are len, compare etc.,.

- Type Conversion Functions: These are used to convert the format of the data. Examples are Decimal to float, String to Date etc.,.
- Utility Functions: Apply on the jobs information. Examples are Get Environment Variables.

4.1.2.20 FTP Stage

Figure 4.23: FTP Stage

The Ftp stage is a processing stage. It can have either an input link, or it can have an output link.

The FTP Enterprise stage transfers multiple files in parallel. These are sets of files that are transferred from one or more FTP servers into DataStage or from WebSphere DataStage to one or more FTP servers. The source or target for the file is identified by a URI (Universal Resource Identifier). The FTP Enterprise stage invokes an FTP client program and transfers files to or from a remote host using the FTP Protocol.

4.1.2.21 Pivot Stage

Figure 4.24: Pivot Stage

Pivot Enterprise stage is a processing stage. This stage pivots data horizontally. The pivot stage maps a set of columns in an input row to a single column in multiple output rows. This type of mapping operation is known as horizontal pivoting. The data output by the pivot stage usually has fewer columns, but more rows than the input data. You can generate a pivot index that will assign an index number to each row with a set of pivoted data.

4.1.2.22 Slowly Changing Dimension

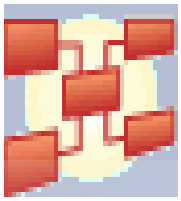


Figure 4.25: SCD Stage

The Slowly Changing Dimension (SCD) stage is a processing stage that works within the context of a star schema database. The SCD stage has a single input link, a single output link, a dimension reference link, and a dimension update link. The SCD stage reads source data on the input link, performs a dimension table lookup on the reference link, and writes data on the output link. The output link can pass data to another SCD stage, to a different type of processing stage, or to a fact table. The dimension update link is a separate output link that carries changes to the dimension.

4.1.3 Restructuring Stages

4.1.3.1 Column Import Stage

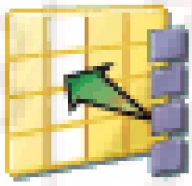


Figure 4.26: Column Import Stage

The Column Import stage is a restructure stage. It can have a single input link, a single output link and a single rejects link.

The Column Import stage imports data from a single column and outputs it to one or more columns. You would typically use it to divide data arriving in a single column into multiple columns. The data would be fixed-width or delimited in some way to tell the Column Import stage where to make the divisions. The input column must be a string or binary data, the output columns can be any data type. You can optionally save reject records, that is, record whose import was rejected, and write them to a rejects link. In addition to importing a column you can also pass other columns straight through the stage.

4.1.3.2 Column Export Stage

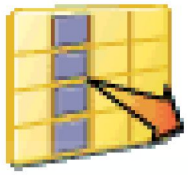


Figure 4.27: Column Export Stage

The Column Export stage is a restructure stage. It can have a single input link, a single output link and a single rejects link.

The Column Export stage exports data from a number of columns of different data types into a single column of data type string or binary. It is the complementary stage to Column Import. The input data column definitions determine the order in which the columns are exported to the single output column. Information about how the single column being exported is delimited is given in the Formats tab of the Inputs page. You can optionally save reject records, that is, records whose export was rejected. In addition to exporting a column you can also pass other columns straight through the stage.

4.1.4 Development/Debug Stages

4.1.4.1 Peek Stage

Figure 4.28: Peek Stage

The Peek stage is a Development/Debug stage. It can have a single input link and any number of output links.

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. The Peek stage can be helpful for monitoring the progress of your application or to diagnose a bug in your application.

4.1.4.2 Row Generator Stage

Figure 4.29: Row Generator Stage

The Row Generator stage is a Development/Debug stage. It has no input links, and a single output link.

The Row Generator stage produces a set of mock data fitting the specified meta data. This is useful where you want to test your job but have no real data available to process. The meta data you specify on the output link determines the columns you are generating.

4.1.5 Database Stages

4.1.5.1 DB2_UDB_Enterprise stage

Figure 4.30:DB2 stage

The DB2/UDB Enterprise stage is a database stage. It allows you to read data from and write data to a DB2 database. DB2 databases distribute data in multiple partitions. DataStage can match the partitioning as it reads or writes data from/to a DB2 database. The DB2/UDB Enterprise stage can have a single input link and a single output reject link, or a single output link or output reference link.

4.1.6 Real Time Processing Stages

4.1.6.1 Java Transformer Stage

Figure 4.31: Java Transformer Stage

The Java Transformer stage is an active stage. Use it to call a Java application that transforms data from an input link and writes the data to an output link in a job.

The Java Transformer Stage can have an input link and one output link, and can have a one optional reject link for writing the rejected rows and erroneous data. The class file specified at the Java transformer stage applies on the input data, and gives the output.

4.1.6.2 XML Transformer Stage

Figure 4.32: XML Transformer Stage

XML Transformer stage is a Real Time Processing stage. It can have one input link, optionally one output link. The XML Transformer is used to convert XML documents to another XML hierarchical format.

The major features of XML Transformer include:

- Generating multiple documents from the same input.
- XML schema validation.
- Robust error handling.
- Support for writing to disk or to an output link.
- Validation of XML input.
- Adherence to XSLT standard style sheets.

Chapter 5

DataStage Stage Jobs

Introduction

This Chapter discusses about the Software and Hardware configurations of the System that is used in the analysis of the DataStage and the Stage jobs.

5.1 Motivation

Consider the situation where a single user running a simple job on DataStage, the job executes successfully. But consider running job with more number of users, more number of jobs to be executed on DataStage at the same time, then there will be situations where the job may gets aborted because of lack of resources.

If you know the resources required by the jobs, before running the jobs, then there will be less number of situations where job gets aborted, rather than working blindly. This calculation of the required resources will be done in two steps. One way is that get the file which contains the required resources information. Install the file with the DataStage product installation. The second way is to develop a tool which takes the input from the user about the information of the job and then gives the required resources to run the job successfully.

5.2 Parameters used in Analysis

This project is about developing the file which contains the resources required for each stage. This is done with one stage, input and output stages. Calculate the resource usage of the job. The resources that calculated are memory usage, processor usage, disk usage, database usage of the job, time taken to run the job. Processor usage, memory usage, and disk usage of the job are calculated from Performance monitor. The time taken to run time can be further divided as production run time and start up time. Startup time is the time taken to start the actual execution of the job. Production run time is the time taken to execute the job. The Total time taken to run the job is sum of the production run

time, start up time and the overhead. The time values are calculated from the Run Director of the DataStage.

5.3 Configuration details

The Hardware and Software that is used in the analysis of the DataStage is given below.

5.3.1Hardware Information

The following lists the Hardware configuration that is used in the Analysis of the DataStage.

System Manufacturer : IBM
System Model : IBM x3850-[88632SA]
Processor : Intel® Xeon™ MP CPU 3.66GHz
Number of CPU's : 8
Memory : 4GB

5.3.2Software Information

Operating System : Microsoft Windows Server 2003 Enterprise Edition
DataStage : IBM Information Server DataStage version: 8.1

5.4 Jobs

5.4.1 Copy Stage Job

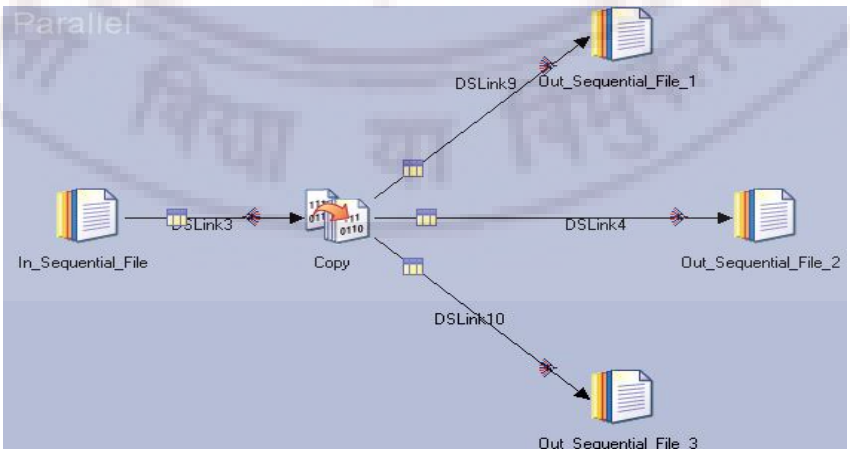


Figure 5.1: Copy stage job

This job copies the input data file to the three files. You can map the required columns to the output by using the mapping tab of output tab in Copy stage.

The results of the Copy stage run with 8 nodes and 1 node are collected.

5.4.2 Aggregator Stage Job

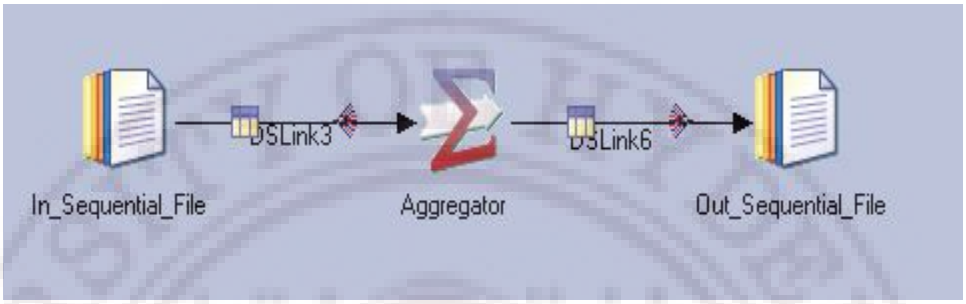


Figure 5.2: Aggregator stage job

The Aggregator job takes the input data file, and aggregates the data according to the Grouping keys, and computes the aggregate functions for each group. The data first will be grouped based on the first grouping key, and then groups it according to second grouping key, and does the grouping operations.

This job groups the input data according to two key columns, and computes the standard deviation of the third column.

The results of Aggregator job run with 8nodes and 1nodes with one and two grouping keys is collected.

5.4.3 Join Stage Job

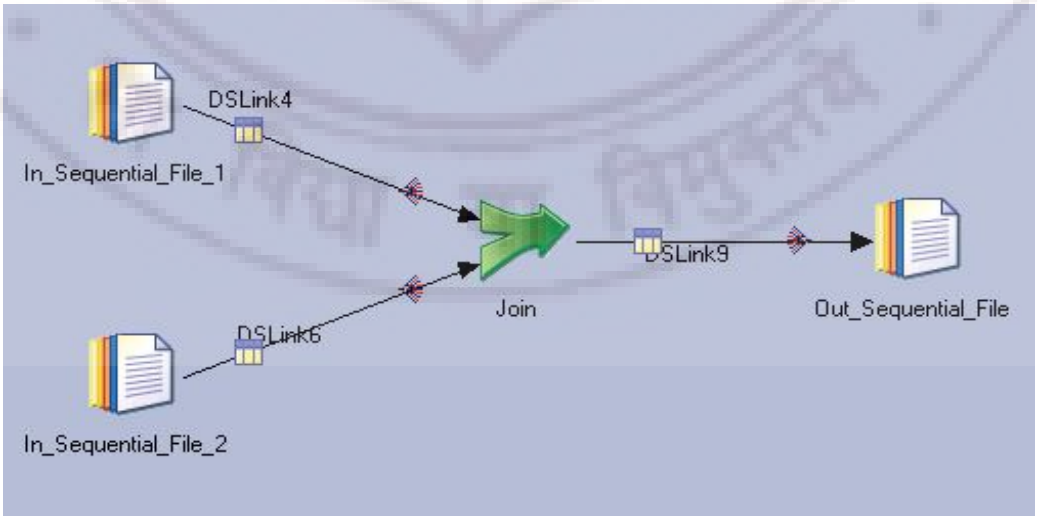


Figure 5.3: Join stage job

This Join job performs join operation on two input data sets that are input to the stage and then outputs the resulting data set.

The job takes the two input datasets as the left and right input datasets. This ordering of the input datasets can be changed by editing ordering at stage properties link ordering tab. This stage uses one key column to match the input records. The output column ordering can be changed at mapping tab of the output tab of stage properties tab.

This Join stage uses the Scratch disk for storing the input data before it starts sorting of the data. This stores the input data because it has to sort the input records before it starts joining. So for the files which are bigger in size the user must take care about the sufficient size of the disk space should be there. If there is no sufficient space on the disk then the job will be aborted.

The results of the Join stage run with 8nodes and 1 node, with Inner join and Left Outer join are collected.

5.4.4 Merge Stage Job

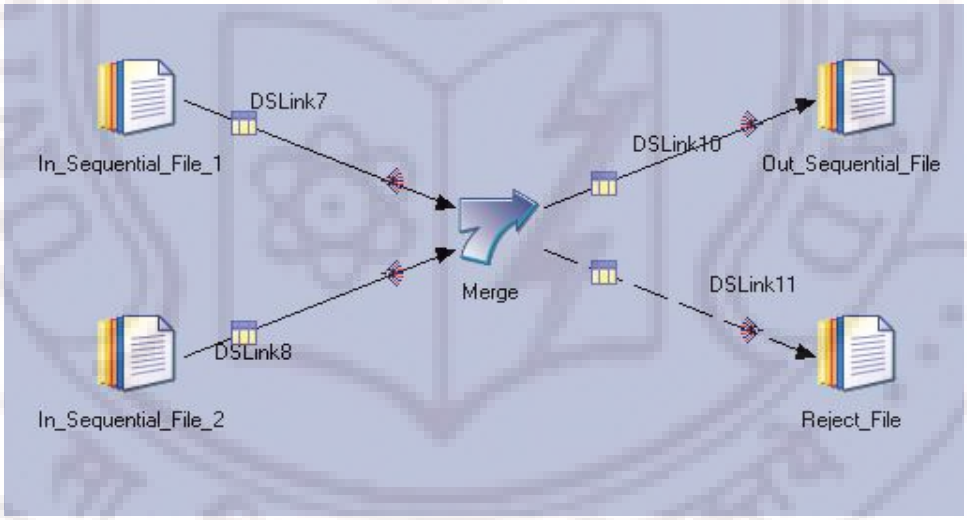


Figure 5.4: Merge stage job

This Merge stage takes the input data from two input data sets, merges based on the input column matching, the records which are not matched will be sent to the Reject link.

The two input datasets are called as Master input and update input dataset. The Master records are matched against the update file, and the records are matched based on the one key column. The records which are matched will send to the Master output link,

and the records from the update link which are not matched will be send to the update reject link. The order of the master, update, master output and update reject links cab be changed at the link ordering properties of stage properties of the stage.

The results of the Merge Stage run with 8 nodes and 1 node are collected.

5.4.5 Lookup Stage Job

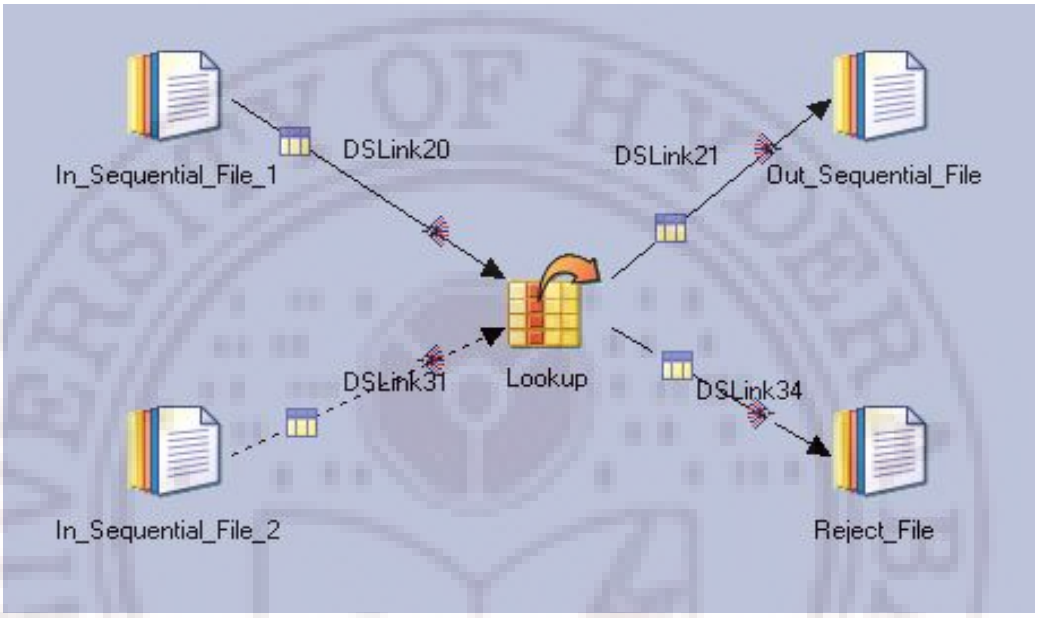


Figure 5.5: Lookup stage job

The Lookup Stage takes the two datasets as input, and lookup the input data set with the reference data set, stores the records with specified constraints in output file, and the records which does not match to the reject file.

This Lookup stage job has two input links which are referred to as primary link and lookup link. The ordering of the links cab changed at the link ordering tab of the stage properties tab. This stage matches the records of the primary file with the lookup file records. The records which are matched will send to the output put file, and the records which do not match will be send to the reject file. This stage must be run with smaller number records compare to that of the merge and join stages.

The results of Lookup stage run with 8 nodes and 1 node are collected.

5.4.6 Funnel Stage Job

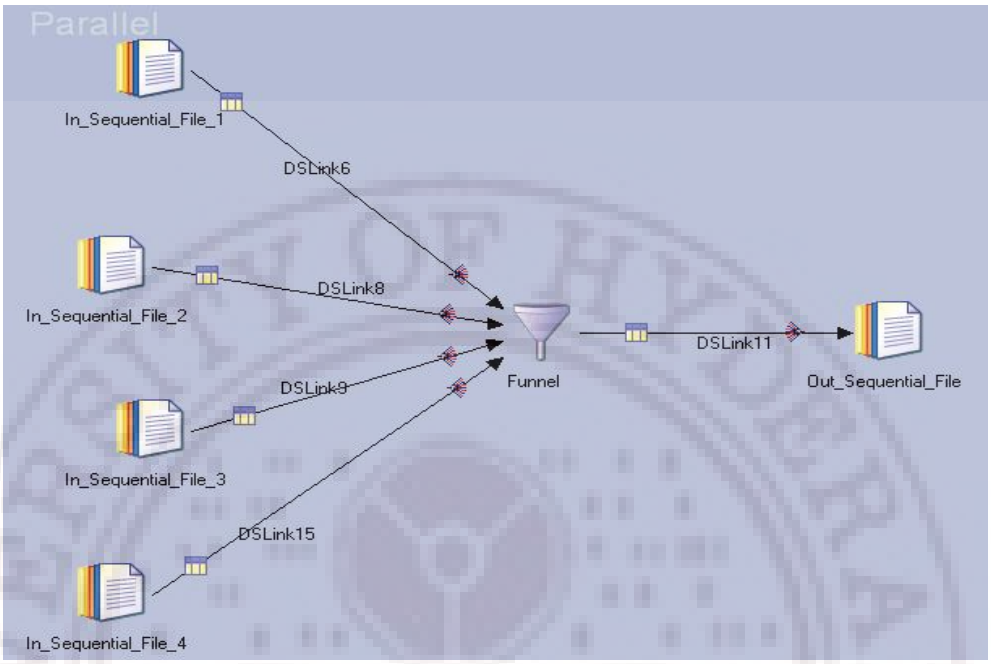


Figure 5.6: Funnel stage job

This job copies the four input data sets to the one output data set by matching the key columns.

This Funnel stage has four input links. In this all the data from the four input links will be stored in the output file. This Funnel stage is of three types, Continuous funnel in which one record from each file will be taken and send to the output link, if the records in one file complete then from next round it skips that file. In Sorted funnel the copy is based on the sorted keys defined, based on these keys the copying takes place. In Sequential funnel all the records from one file are copied first and then from the second and so on.

The results of the Funnel stage with the 8nodes and 1node with threes types of funnel stage are collected.

5.4.7 Sort Stage Job

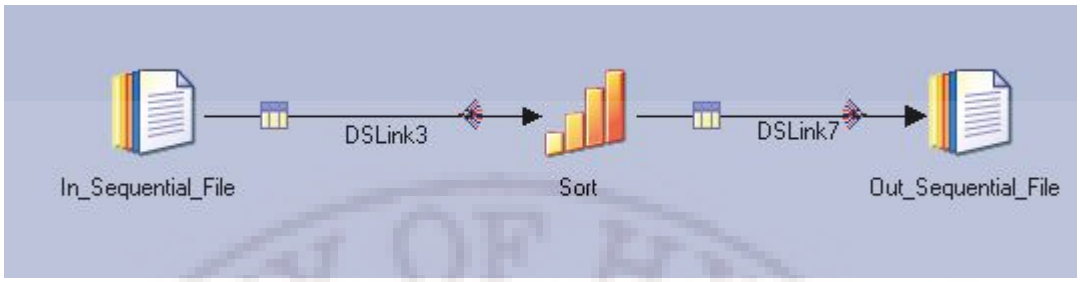


Figure 5.7: Sort stage job

This Sort stage sorts the input data set based on the primary key and secondary keys.

This Sort stage job has one input link having six columns. The stage sorts the data by using four key columns. The first column is the primary key column. If the two records matches on the primary key column, they will be sorted based on the secondary key column. If they match on the secondary column also they will be matched against the third column etc. In this you can also give ascending or descending order of sorting for each key column. You can also give the option on this stage as to allow duplicates or not. Partitioning and collecting of the records can be edited at the partitioning tab of the stage properties.

The results of the Sort stage run with 8 nodes and 1 node with 1key and 2 keys are collected.

5.4.8 Transformer Stage Job

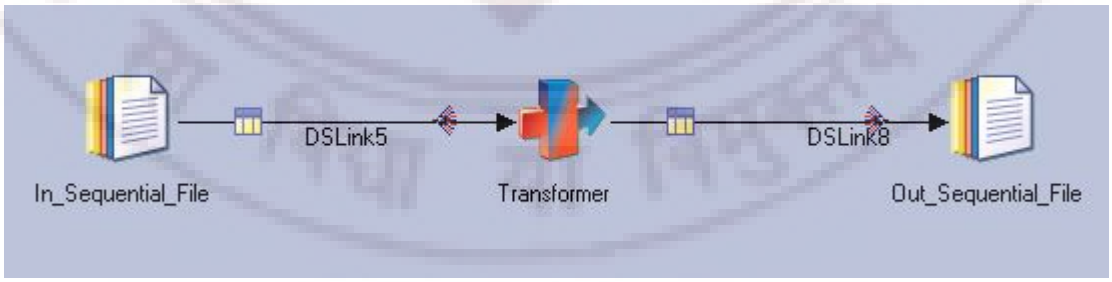


Figure 5.8: Transformer stage job

The Transformer Stage takes the input data and applies the transform functions on the data.

In the Transformer stage there are many functions which can be applied on the input records. These functions are grouped into 8 groups. The results of each group are displayed below.

There are 110 functions in transformer stage, which are grouped to 8 stages and the results for each group with 8nodes, 4nodes, 2nodes and 1node are collected.

5.4.9 Change Capture Stage Job

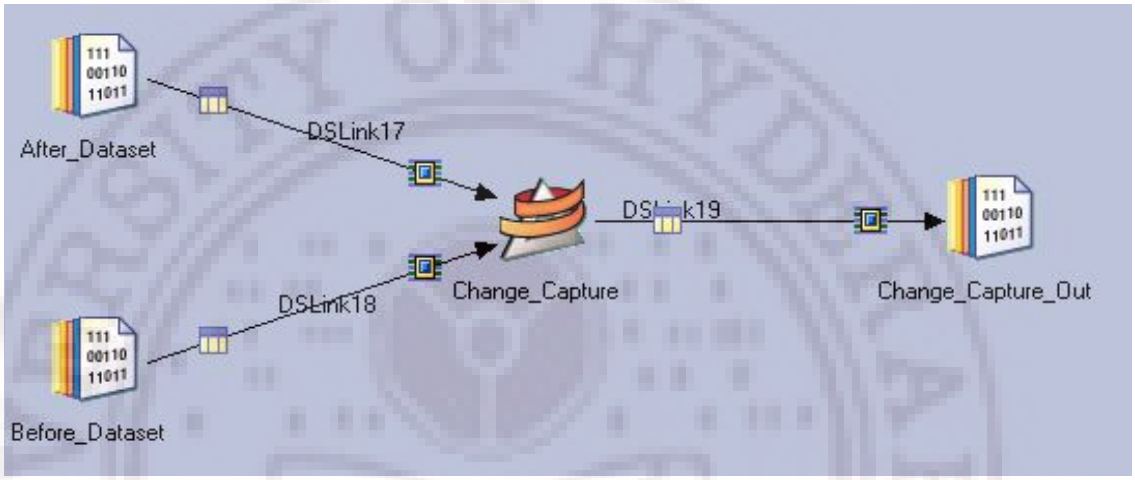


Figure 5.9: Change Capture stage job

This job gives the output, which contains the data that is used to obtain the input after data set from that of the before dataset.

The Change capture takes the two input datasets. These input datasets are called as before and after datasets. These are checked against the key column to get the file from which the after data set is obtained. In this stage for the key column it can have the sorting order also.

The results of Change capture stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.10 Change Apply Stage Job

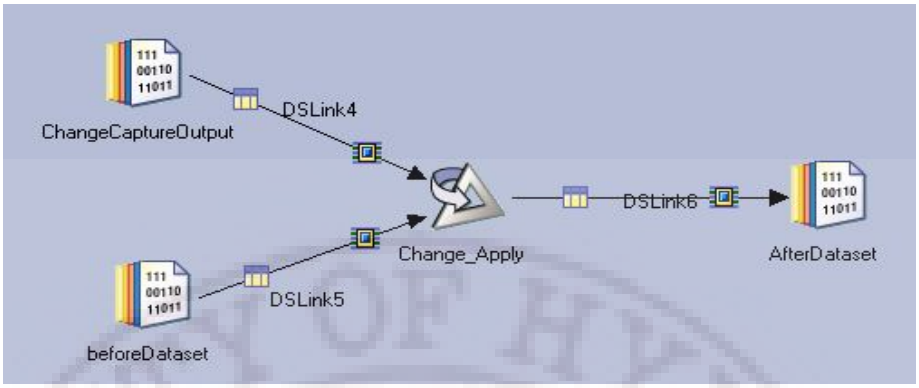


Figure 5.10: Change Apply stage job

Change apply stage takes the output of the Change capture stage and before data set as the input and gives the after dataset as the output.

If the before dataset, change capture stage input data set are those which are used in Change capture stage then you will get the output as the after stage in the change capture stage. The Change apply stage takes a key column and then matches the before dataset against the change capture output file to give the after dataset.

The results of Change Apply stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.11 Filter Stage Job

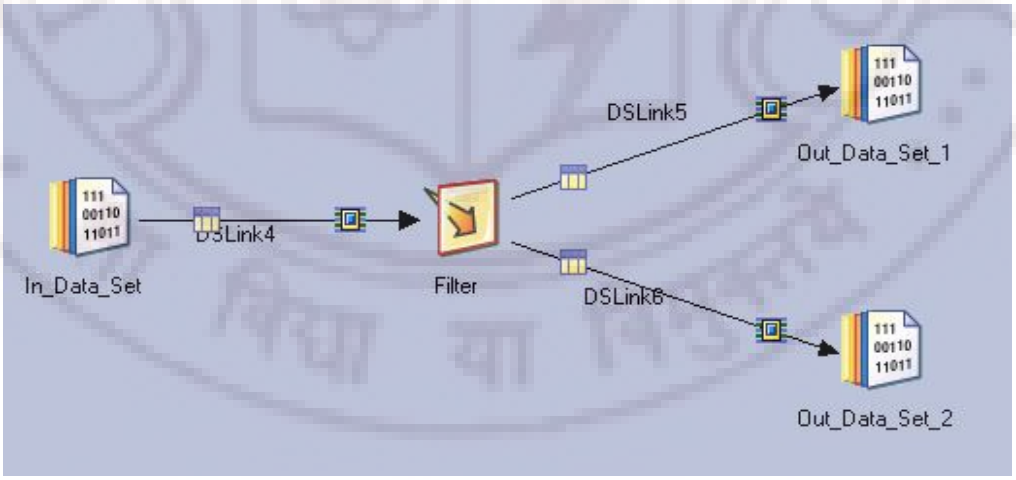


Figure 5.11: Filter stage job

This job filters the input columns according the constraints and outputs the data of each constraint to each link.

In this Filter stage, the output is based on the where clauses that are given in filter stage. The records which matches to the where clause will be sent to the particular output dataset. There are as many as output data files as the where clauses.

The results of filter stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.12 Remove Duplicates Stage Job

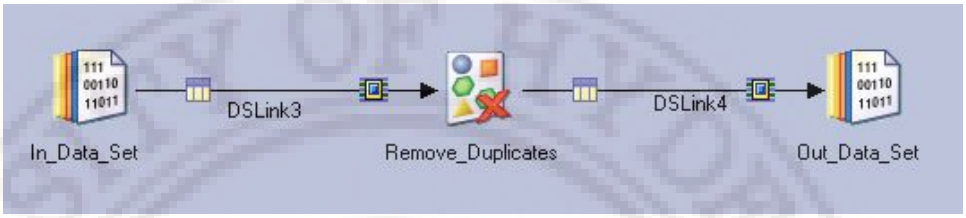


Figure 5.12: Remove Duplicates stage job

This job removes the duplicates from the input dataset based on the key columns. If the key column matches it considers the two records as the duplicates records and send only one record to the output stage.

The results of the Remove duplicate stage with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.13 Switch Stage Job

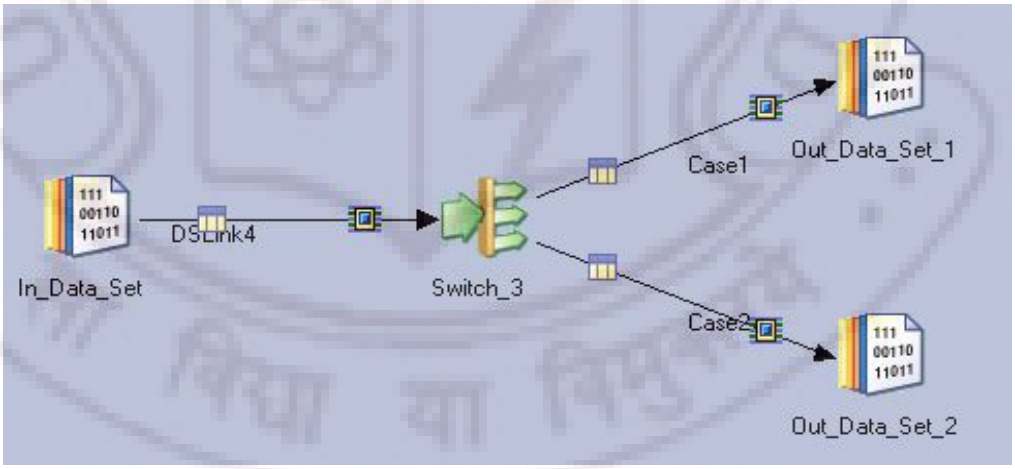


Figure 5.13: Switch stage job

This job takes the input record applies on the case statement. The records which match the case statement will send to the particular link associated.

In this Switch stage, for the given column name, if the records with the given condition matches they will send to the particular link. You can define the ordering of the

links in link ordering tab of stage properties page, and the partitioning and/or combining at the stage pages.

The results of the Switch stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.14. Column Import Stage Job

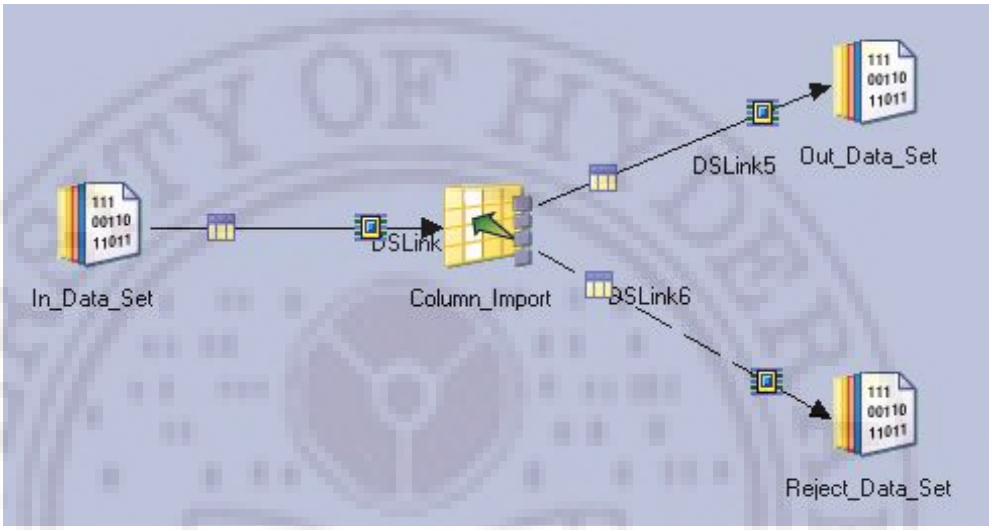


Figure 5.14: Column Import stage job

This job imports one column from the input dataset and outputs it to the output dataset.

In the column import stage properties tab you can give which column to import. The options are want to keep the imported column in the output dataset or to drop it, and the reject mode of the data.

The results of the Column Import stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.15 Slowly Changing Dimension Stage Job

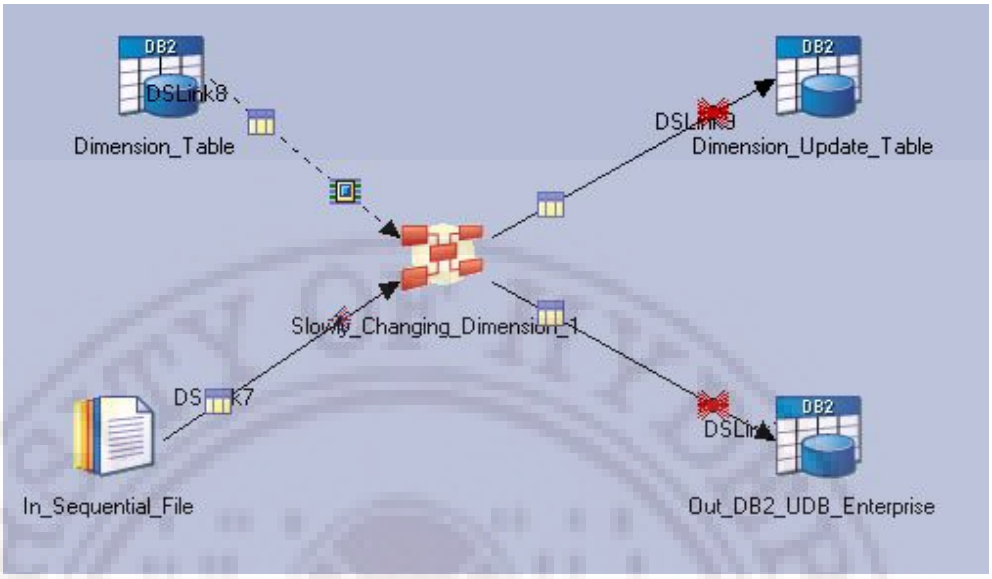


Figure 5.15: Slowly Changing Dimension stage job

This job takes the data from the input file and dimension reference table, and writes the data to the output link, and the updates to the reference table are written to the dimension update table.

The results of slowly changing dimension stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.16 Pivot Stage Job

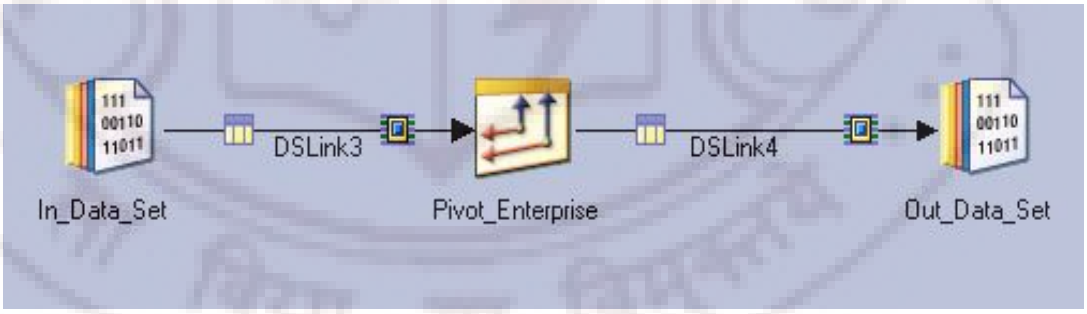


Figure 5.16: Pivot stage job

This job maps three columns in an input row to a single column in multiple output rows.

This job takes a six column input data set, and combines the three columns to on column using the pivot stage with a delimiter between the columns. This pivoting will be performed in pivot properties tab of pivot stage properties, by specifying the derivation.

The results of the pivot stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.17 FTP Stage Job

Figure 5.17: FTP stage job

This FTP stage job has an input link, keeps the data from the input data set on the destination machine.

In the FTP stage properties you have to give the URI, which is the destination file path. The path must be correct and valid, otherwise the job gets aborted. The options in this stage are that you can select ftp or secure ftp transfer. The username and password must be given.

The results of FTP stage with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.18 Row Generator Stage Job

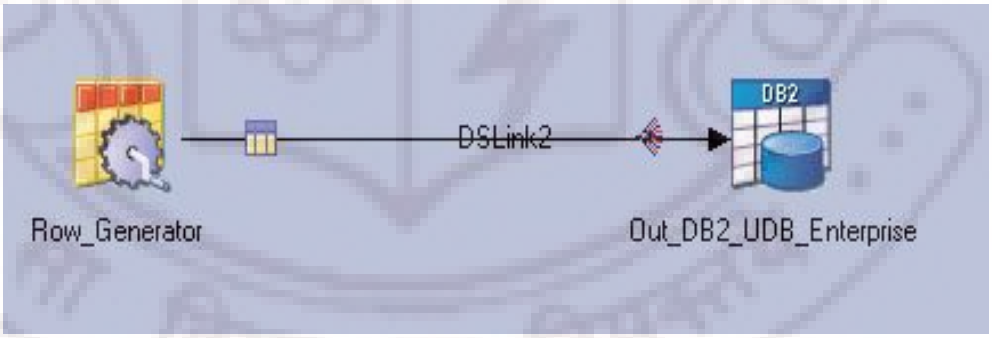


Figure 5.18: Row Generator stage job

This job generates one column of mock data according to the file definition given. In the stage properties you can edit the number of records to be generated. In the row properties you can edit how the rows should be generated cyclic or randomly. And the seed value, increment value, limit etc. The constraints can be number of records, number of columns, starting value etc.

The results of Row generator stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.19 Peek Stage Job

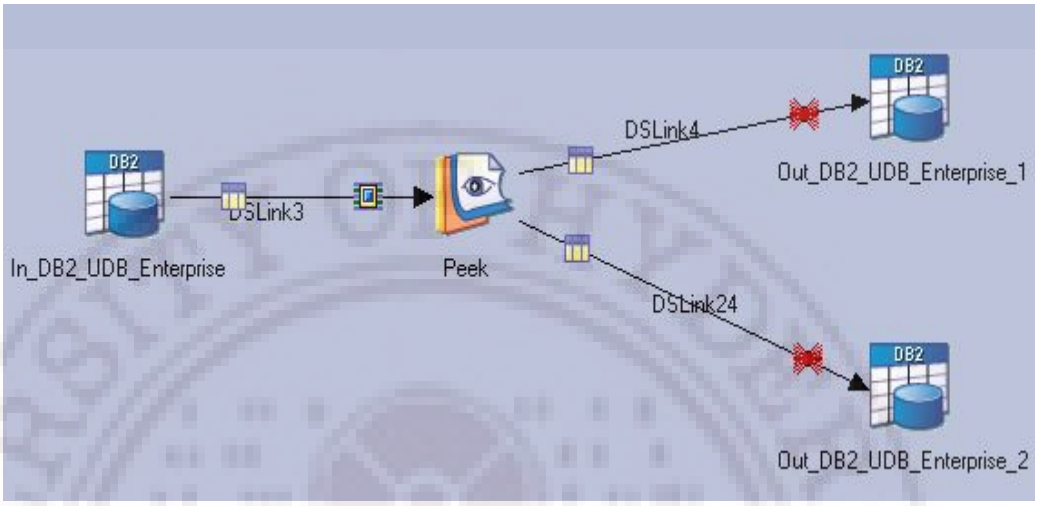


Figure 5.19: Peek stage job

This job copies the input dataset to the output data set, and record column values to the other dataset.

In the job running process, if you want to know the processing of the records you can keep this stage in between the stages. So that user can view the processing details. The job log can be sent to the director or to a stage to store the details. This job is used in monitoring the job.

The results of Peek stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.20 Difference Stage Job

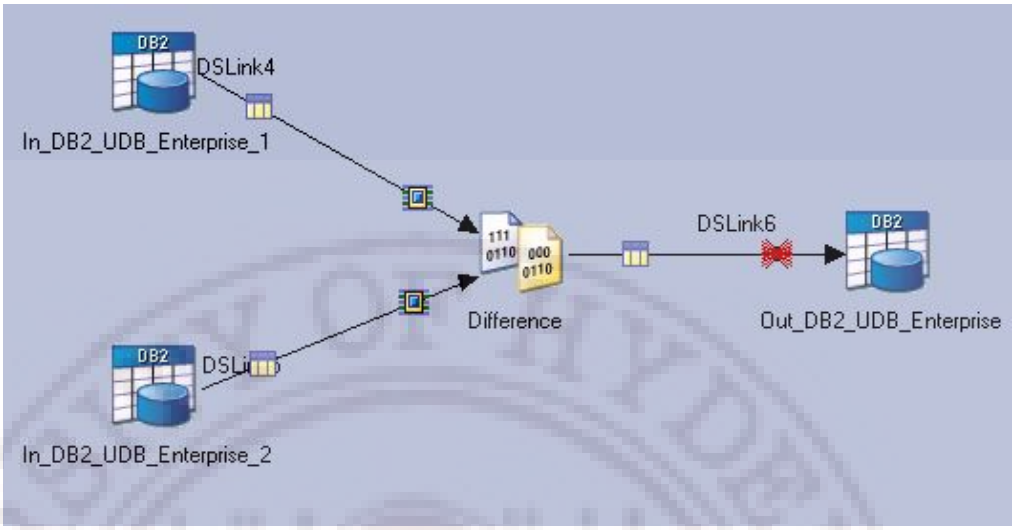


Figure 5.20: Difference stage job

This job takes the two input datasets, compares the two datasets with their key column values, and outputs the data having the two dataset records and the difference between the key columns.

The results of the Difference stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.21 Compress Stage Job

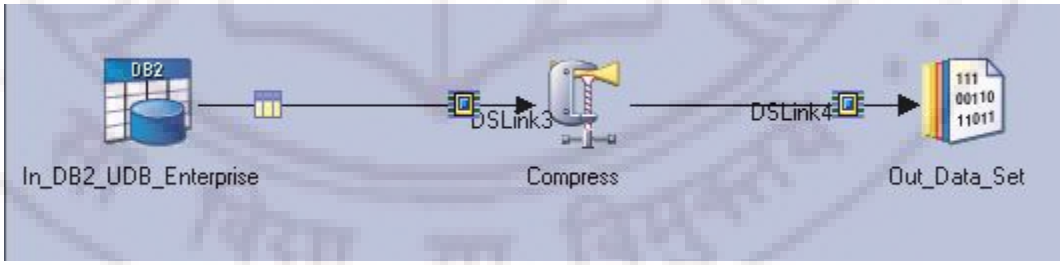


Figure 5.21: Compress stage job

This job takes the input data from the database stage, and compresses it using the default command of DataStage which is compress, and the data is stored in a Dataset. This job uses the Runtime column propagation option, so that no need to give the table definitions, DataStage at run time generates the table definitions. As the target stage is the Dataset stage, you are not able to find the size of the output data, number of output

records, and output number of columns. The results of the Compress stage run with the 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.22 Expand Stage Job

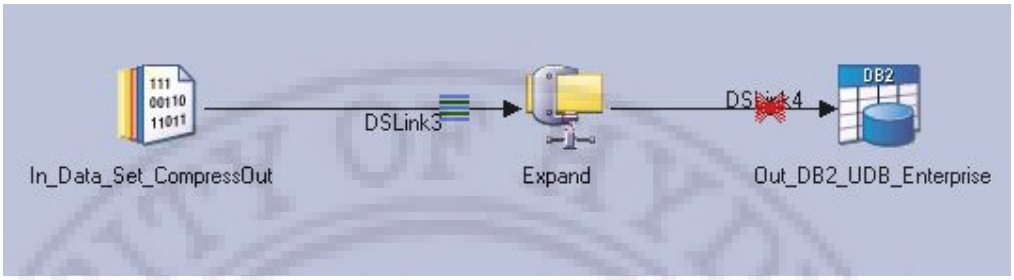


Figure 5.22: Expand stage job

This job takes compressed data, which is the output of the compressed stage as the input and applies the DataStage's default command decompress to expand the compressed data. This job uses the Runtime column propagation for defining the metadata of the table. As Dataset stage is used as the source dataset, you will not get the number of input records, number of input columns etc. The results of the Expand stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.23 Compare Stage Job

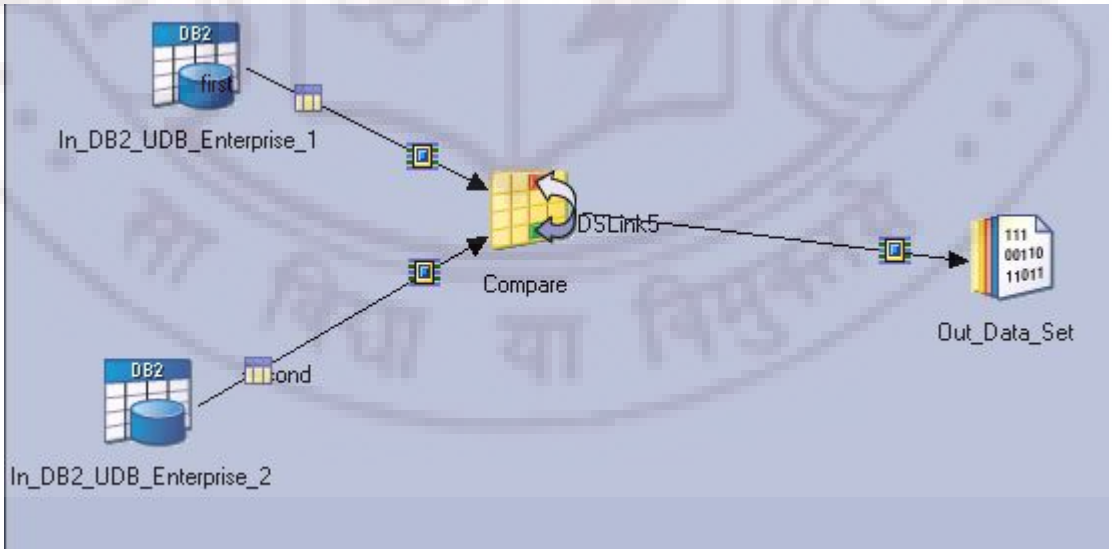


Figure 5.23: Compare stage job

This job takes the two input datasets from the DB2 database, compares the two input datasets by using key columns. The output contains three columns, one is the result column which gives the comparison result of two datasets that the two datasets notified with first and second, and the remaining two are the two input datasets. The stage uses the Runtime propagation for defining the table definitions. As the target stage is the Dataset stage, we are not able to calculate number of output records, number of output columns etc.

The result of the Compare stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.24 Encode Stage Job

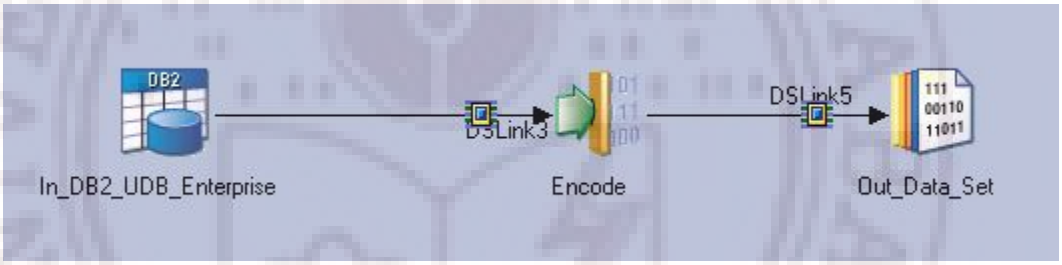


Figure 5.24: Encode stage job

This stage takes the input dataset using the DB2 database stage, and encodes it. The output data is stored in a persistent data set. This stage uses the Runtime column propagation option for defining the table definitions. As the target stage is dataset stage you are not able to get the number of output records, number of output columns and the output file size.

The results of the Encode stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.25. Column Export Stage Job

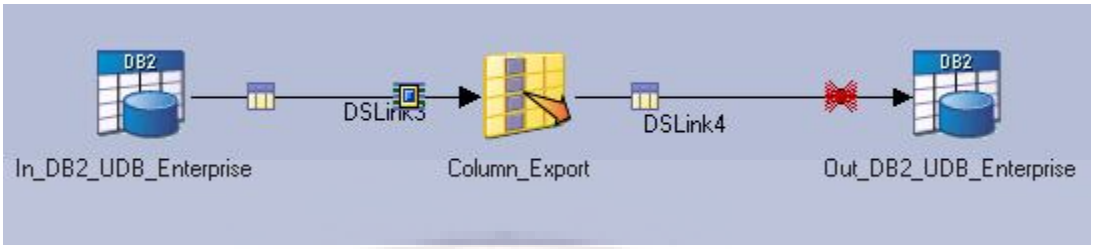


Figure 5.25: Column Export stage job

This job takes the input data from the DB2 stage, combines three input columns into one output column. The output data is stored in database. The exported column is of type String. The results of the Column export stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

5.4.26 Java Transformer Stage Job

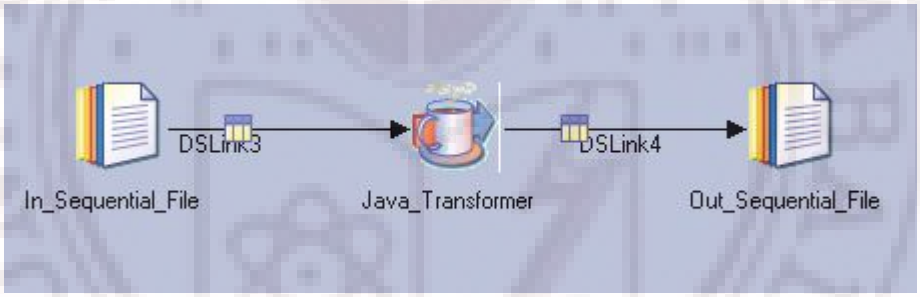


Figure 5.26: Java Transformer stage job

This job applies the java file in Java transformer stage on the input data file, and store the result in the output file.

In this Java transformer stage you should give the class file name in general tab of the stage properties. The values of variables should be given in properties tab of the stage properties.

For running this stage, it requires JVM. For setting up the JVM environment define DATASTAGE_JRE, DATASTAGE_JVM, DSHOME environment variables in administrator.

The results of Java Transformer stage run with 8 nodes, 4 nodes, 2 nodes and 1 node are collected.

With this, Part-1, Statistical Data Collection and Analysis of the DataStage is completed. Part-2 discusses about Database Explorer and, Extending Teradata database to the Database Explorer.

PART-2

Database Explorer

Chapter 6

Introduction to Database Explorer

Introduction

This Chapter discusses about the Database Explorer, operations performed on the Database Explorer.

6.1 Database Explorer

Database Explorer is tool which enables the users to connect to multiple databases from the single application. This tool is developed by IBM Corporation. By using Database Explorer we can connect to different databases they are Oracle, Sybase, SQLServer, DB2, Informix etc.

In general for connecting to different database you need to open each window to connect to it. With out having each window opened for each database, by using the Database Explorer to all the databases from the single window. This Database Explorer has Graphical user interface. So for the end user also it is easy to use the Database Explorer.

The Database explorer is developed using the Java language. Swings and the JDBC connections are used to develop this Database Explorer. Swings API is used to develop the front end, and JDBC API is used to connect to the databases and do SQL operations on the databases. These SQL operations are performed on the metadata of the databases rather than the tables of the database directly. This metadata of the database is called as the data dictionary.

The Database Explorer retrieves the data from the database and displays it in a window. This data includes tables in database, indexes, triggers, users, and views etc. All this data that has been retrieved will be displayed in a tree format.

6.2 Operations performed on Database Explorer

The typical operations that are performed on a normal database can be performed on Database Explorer. The operations performed are on tables, users, indexes, triggers and stored procedures.

The operations performed on table create table, drop table, update table, view table metadata, copy table metadata. Creating table using the metadata of another table. This can be performed by copying one table metadata and use the copied information to create another table.

The functions performed on the users are view metadata of the user, which gives the user id, user privileges, and roles of the user.

The functions on the indexes like create index, view metadata of the index.

The operations that are performed on the triggers are creating triggers, view metadata of the trigger, drop the trigger.

The functions that are performed on the stored procedures are create stored procedures are create stored procedure, view stored procedure.

Chapter 7

Extending Teradata database to Database Explorer

Introduction

The Database Explorer supports many databases, but it does not suppors the Teradata Database. The Teradata database is extended to the Database Explorer using the database Data Dictionary of the Teradata Database.

This chapter describes about what is Teradata Database, What is Data Dictionary, Extending the Teradata Database to the Database Explorer.

7.1 Teradata Database

The Teradata Database is a complete relational database management system developed by NCR Teradata Corporation. With the Teradata RDBMS, you can access, store, and operate on data using Teradata Structured Query Language (Teradata SQL).

The Teradata RDBMS was developed to allow users to view and manage large amounts of data as a collection of related tables. The Teradata Database's capacity includes Terabytes of detailed data stored in billions of rows. The parallel processing power of this database makes it faster in executing the queries.

7.2 Data Dictionary

The Data Dictionary (DD) is the system catalog for a database. It contains metadata: table, view and index definitions, parameters and attributes of macros and stored procedures, resource usage statistics, and much more. The DD is a system database repository containing data about user databases and properties of those databases. When a data definition statement is processed, the system tables are updated automatically.

The Teradata Database Data Dictionary is composed of tables and views that reside in the system database called DBC. These tables and views are reserved for use by the system and contain information about the system's associated data. Data Dictionary system tables include current definitions, control information, and general information about the following: Databases, Roles, Profiles, Accounts, Tables, Views, Columns,

Indexes, Constraints, Triggers, Access Rights, Disk space, Events, Resource Usage, Stored Procedures.

When a table is created, the complete data definition is stored in the Data Dictionary along with the following details:

- Table location, identification, database name, table name, creator name, version, database name, and user names of all owners in the hierarchy.
- Each column in the table, including column name, data type, length, and phrases.
- User/creator access privileges on the table.
- Indexes defined for the table.
- Constraints defined for the table.
- Date and time the object was created.

When a view or macro is created, the definition of the object is stored in the Data Dictionary, along with the following details:

- The text of the view or macro.
- Creation time attributes.
- User and creator access privileges on the view or macro.

When a user creates a stored procedure, the following details are automatically entered in the Data Dictionary:

- Creation time attributes of the stored procedure.
- Parameters of the stored procedure, including parameter name, parameter type, data type, and default format.
- User and creator access privileges on the stored procedure.

7.3 Querying the Teradata Data Dictionary

By applying the queries on the Database Data dictionary, the user can retrieve the data from the database. The example of this query is given below.

```
      SELECT DatabaseName, CreatorName, OwnerName, PermSpace ROM
DBC.Databases ;
```

The result is all the databases in the Teradata Database would be listed, as shown below:

DatabaseName	CreatorName	OwnerName	PermSpace
pers	SYSADMIN	SYSADMIN	500,000
Accounting2	Jacobs	Vettes	250,000
SQLDBA	DBC	SYSADMIN	150,000
abc123	SYSADMIN	SYSADMIN	1,460,000
PERSONNEL	SYSADMIN	SYSADMIN	1,500,000
Accounting1	Hillstein	Vettes	500,000
Test1	SYSADMIN	Test1	1,000,000
Jane	Jane	Test1	500,000
.	.	.	.
.	.	.	.
.	.	.	.

7.4 Extending Teradata database to Database Explorer

By using Teradata database Data Dictionary that is metadata of the Teradata database, the Database Explorer can be extended. The Database Explorer retrieves the data from the remote or local database and it displays in the window.

So in extending the Teradata database support to the Database Explorer, by using the metadata of the database the data which is required will be retrieved.

7.5 Screen shots: Teradata extension to Database Explorer

This is the Home page of the Database Explorer. This contains the Menu bar containing the File, Edit, View, Data Source, Command, Window and Help options.

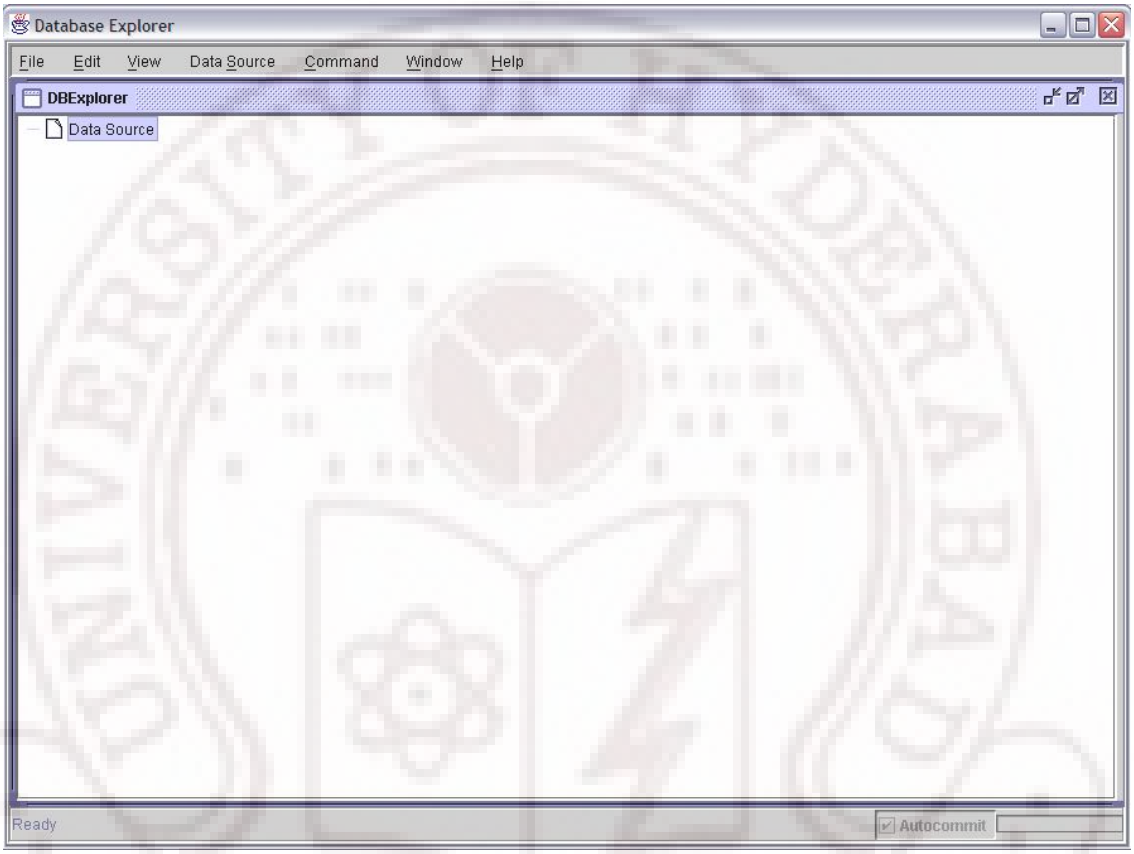


Figure 7.1: Home page of Database Explorer

To add a Data Source to the Database Explorer right click on the Data Source item in the window. It gives the button *Add Data Source*. Click on the *Add Data Source* button to add a data source to the Database Explorer.

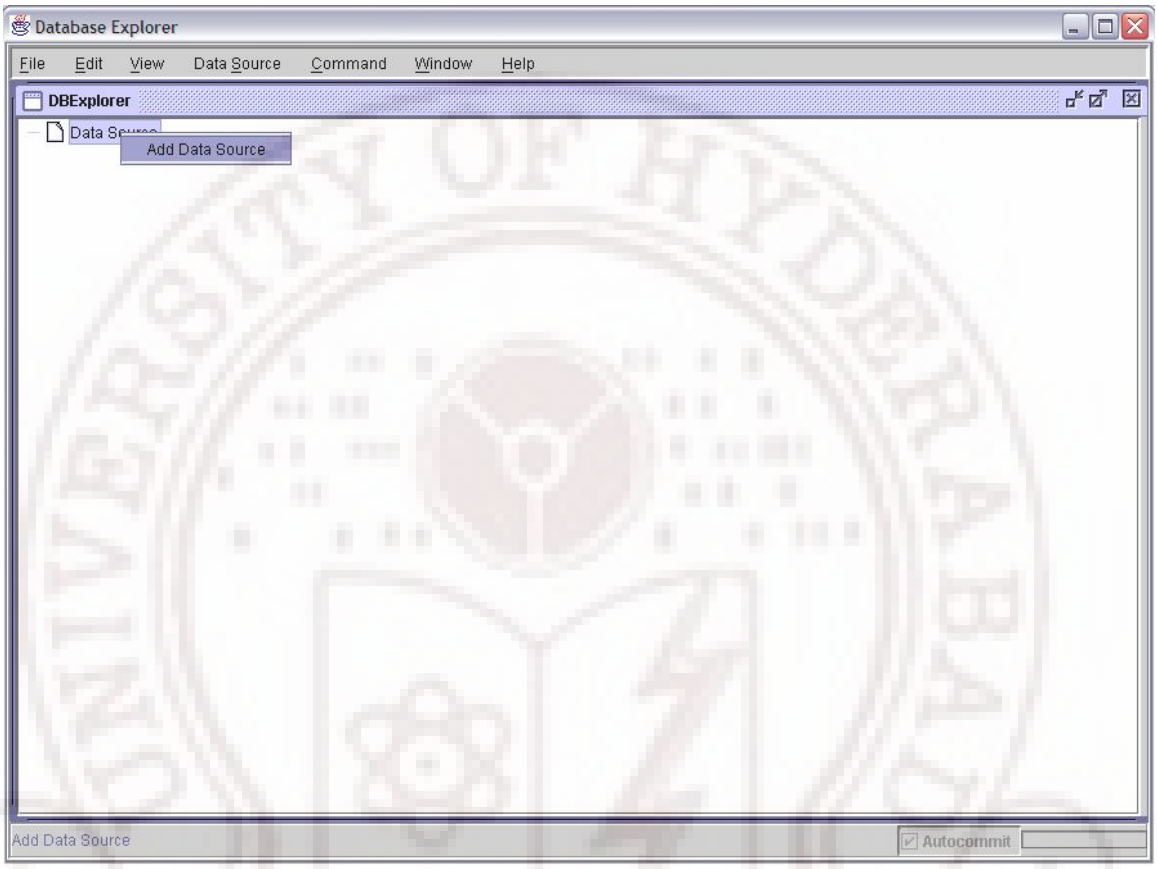


Figure 7.2: Add Data Source

On clicking the button *Add Data Source*, an Add Database dialogue box appears in the Main window. Enter the Details of Data Source that is to be added to the Database Explorer. The details includes Database name, Server Type and Database URL. Then click on the add button to add the Data source to the Database Explorer.

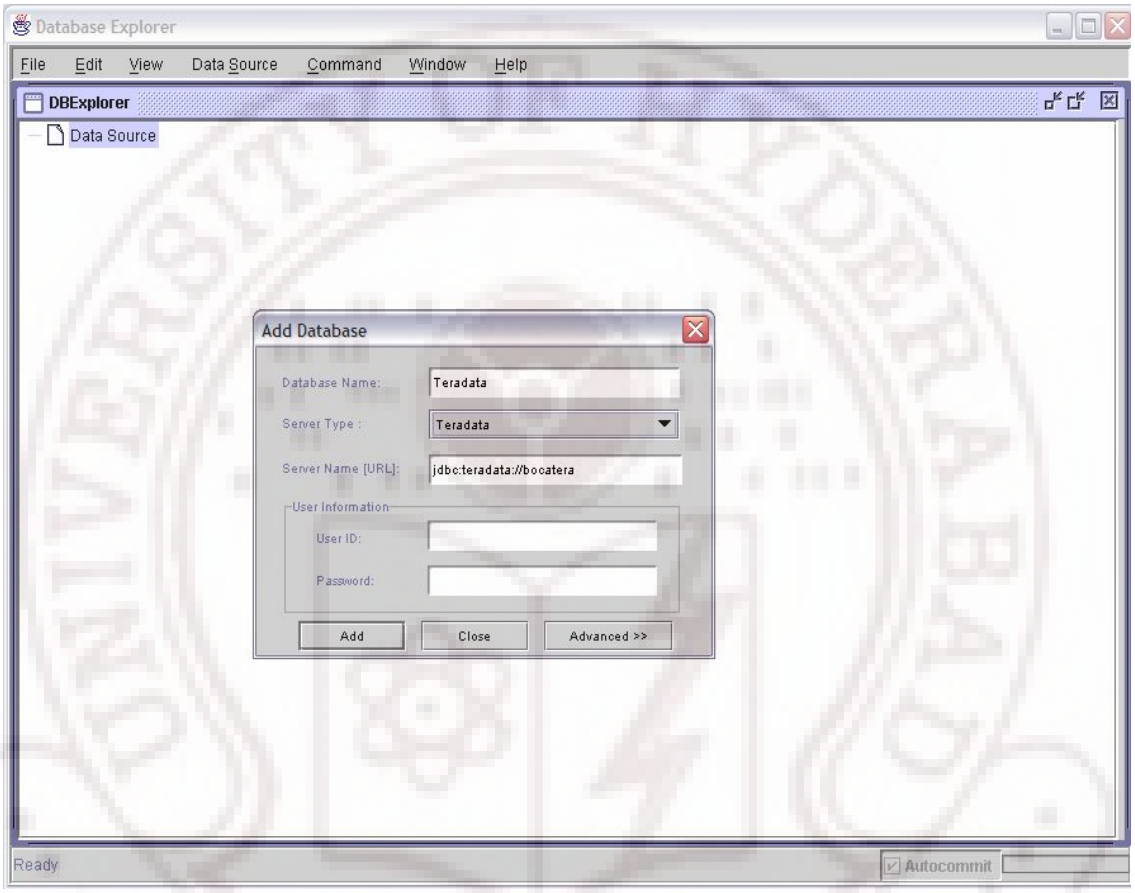


Figure 7.3: Add Database

If adding of the Data Source is successful, then the database name entered in the Add Database dialogue box appears as a child of the Data Source in the Database Explorer window. But still now the connection is not established with the server. To connect to server double click on the Database name you entered or right click on it and select connect option in right selection menu. A new dialogue box appears with name Connect Database Information – Database Name you entered. Enter the User ID and Password of the database server, and then click on the connect button of the dialogue box. After that the tree structure with tables, indexes, users, views etc. will be displayed at the node of the database.

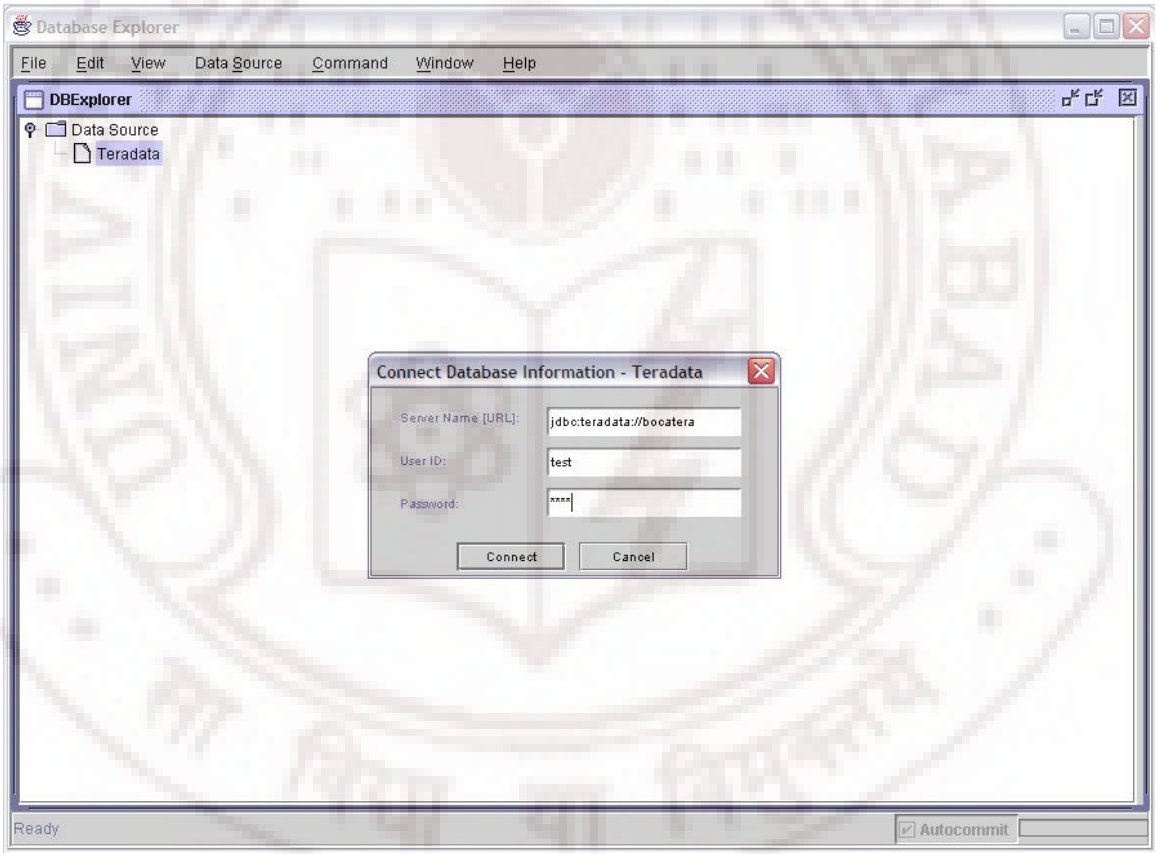


Figure 7.4: User Authentication

The below figure shows the tree, in which Teradata database shows the tables, indexes, views, users, and Triggers of the Teradata database.

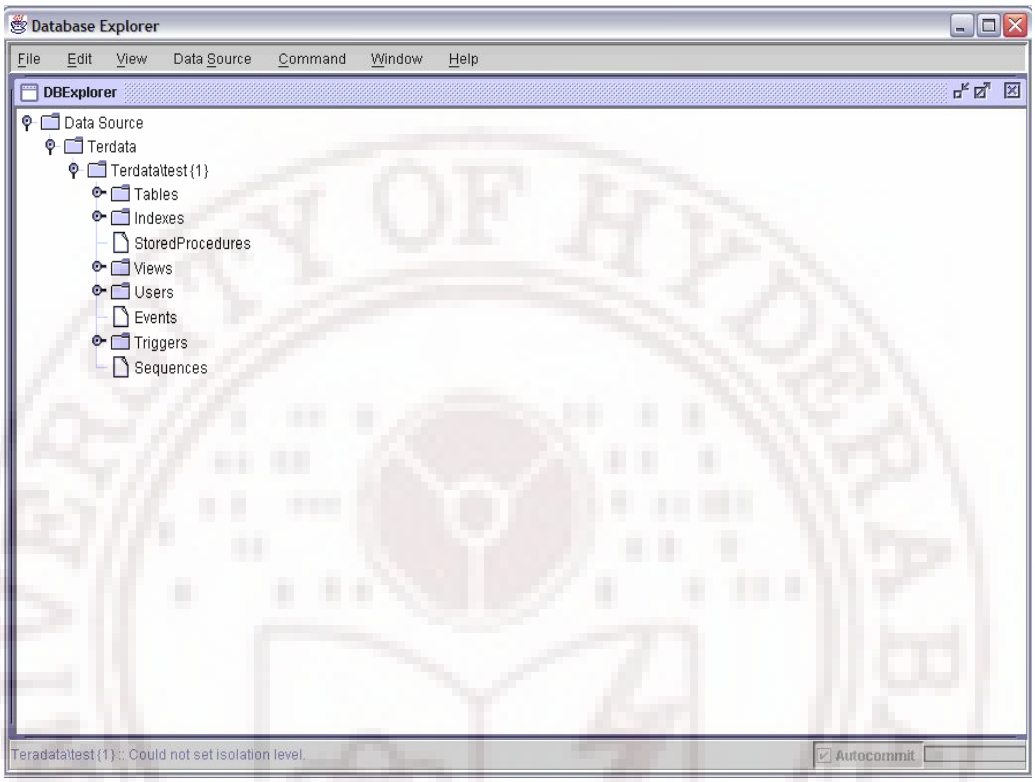


Figure 7.5: Teradata database details

Like the above procedure you can add any database that you want to connect to. The below figure shows, four number of different databases that are connected to the Database Explorer.

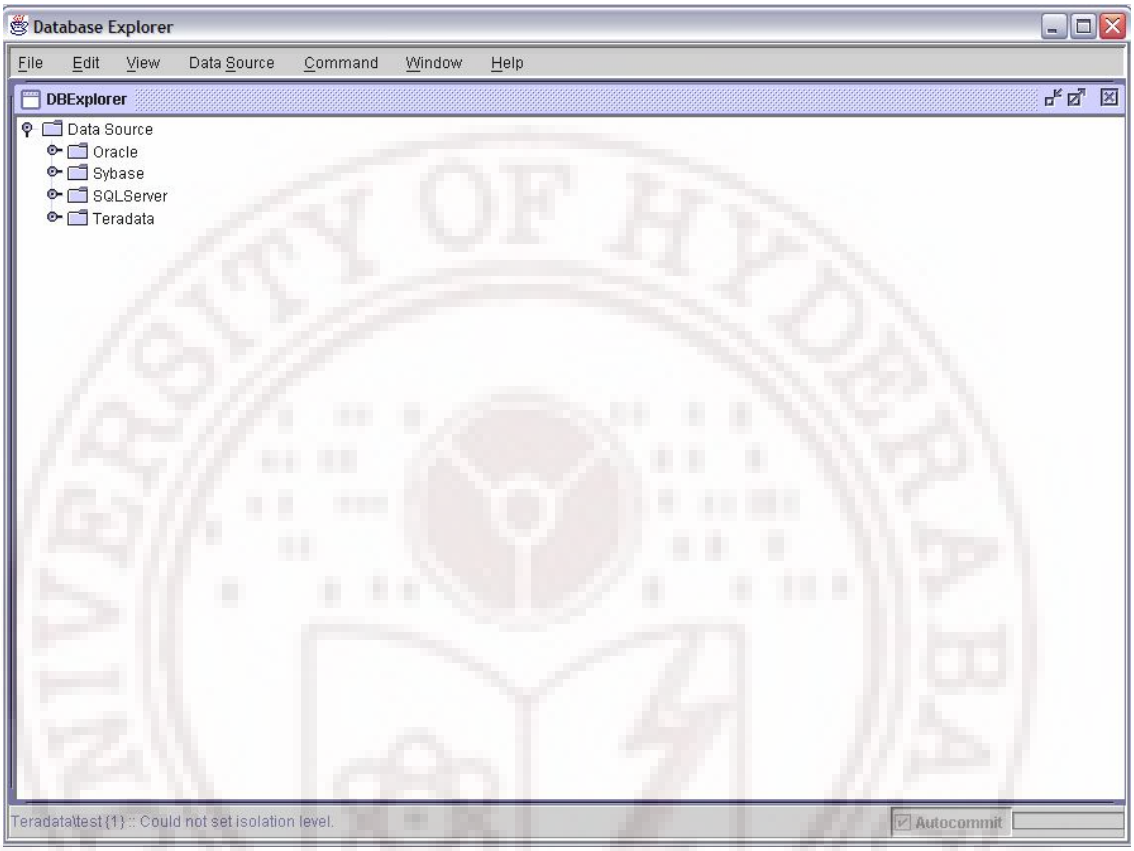


Figure 7.6: Adding Databases

The below figure shows the tree, in which Teradata database shows the tables, indexes, views, users, and Triggers of the Teradata database. In the figure other databases are also connected to the Database Explorer.

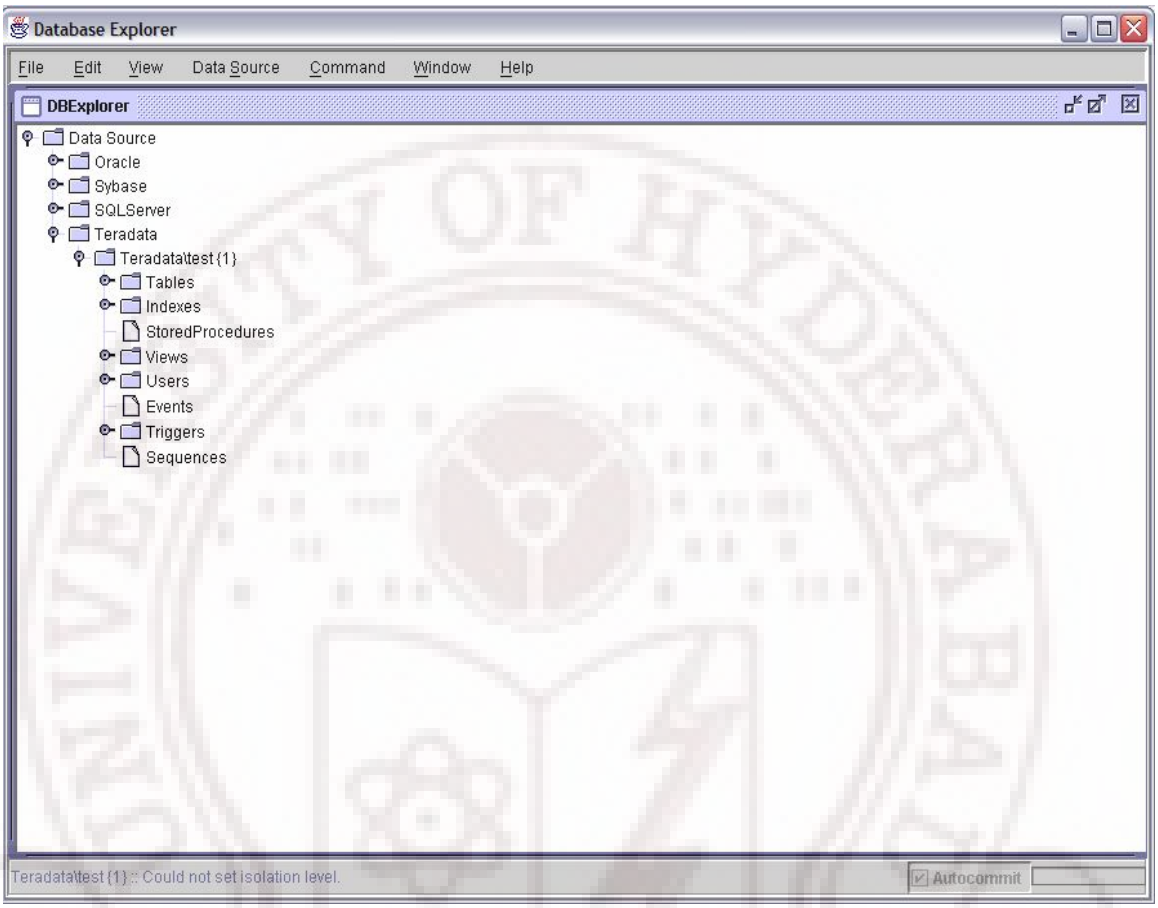


Figure 7.7: Teradata database extension

To view the data in the table of the Teradata database, expand the node Teradata tables, id displays all the list tables in a tree format. Select the table and right click on the table name it displays the right selection menu of a table. Click on the View data option.

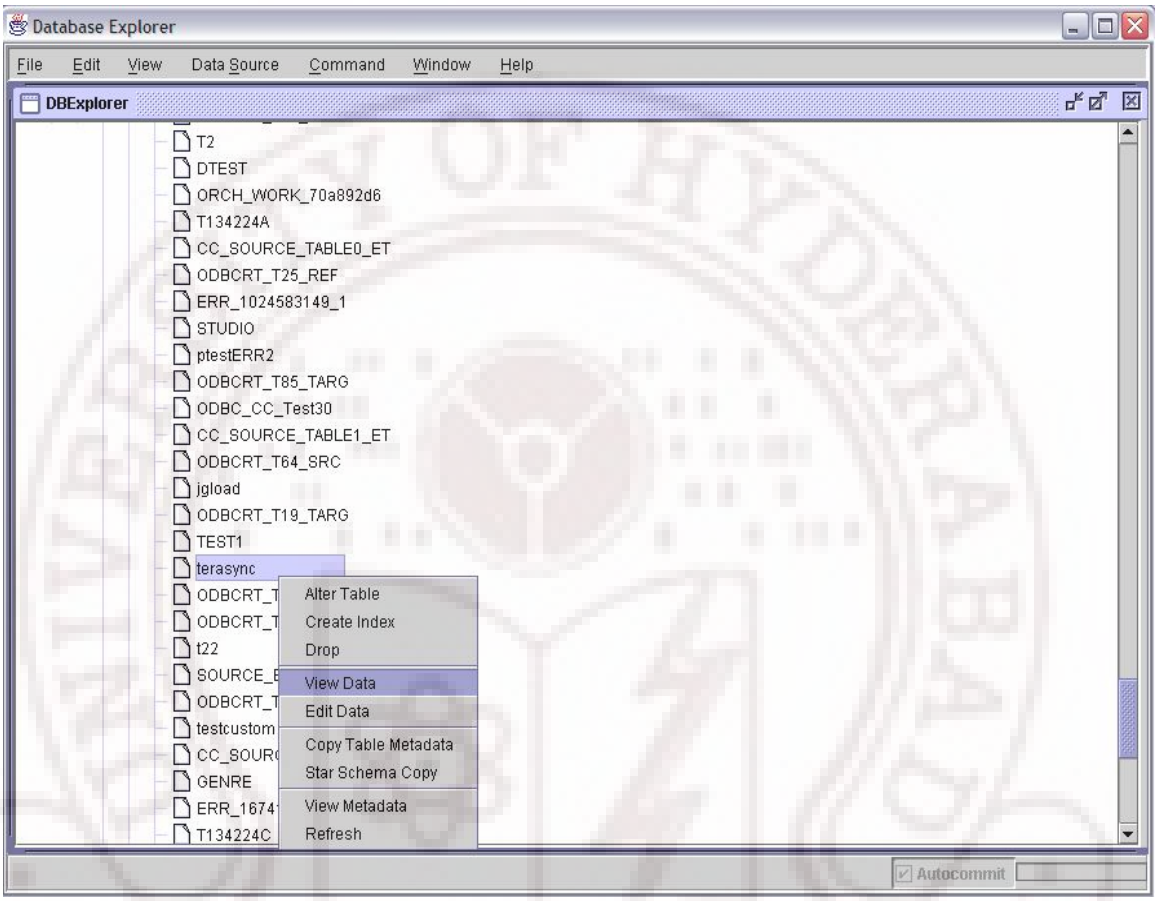


Figure 7.8: View data in a table of Teradata database

The below figure shows the data in the table *terasync*. The *terasync* table has 14 columns. The data of the table is displayed in a new window.

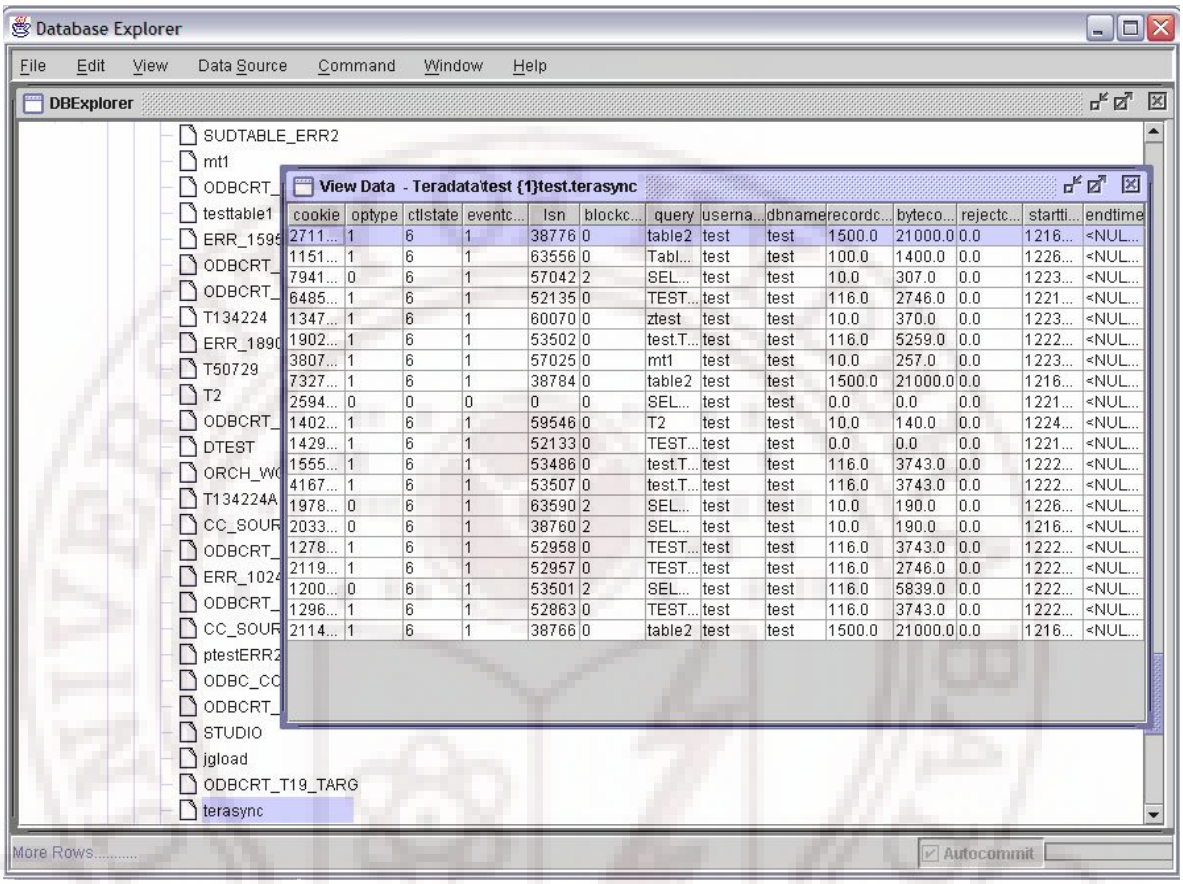


Figure 7.9: Data in Teradata database table

To view the Metadata of the table in the database, right click on the selected tables, select *View Metadata* option from the right selection menu list.

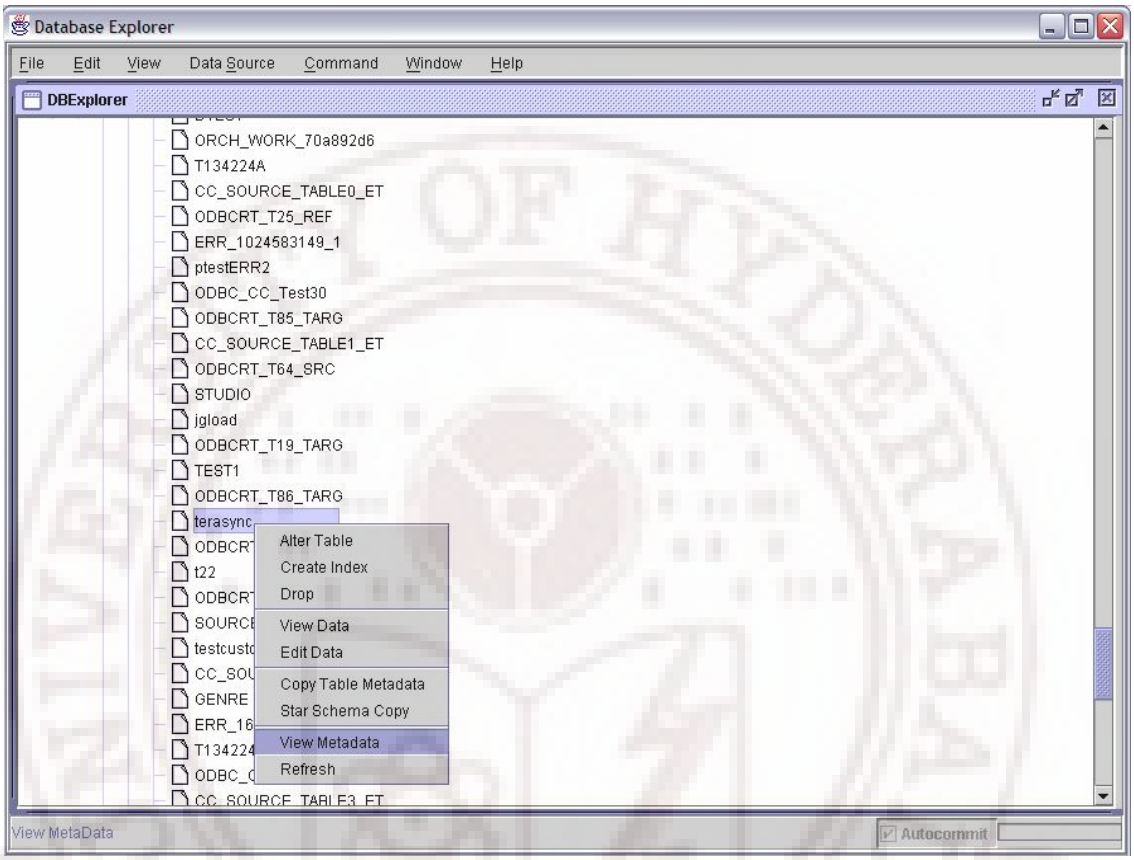
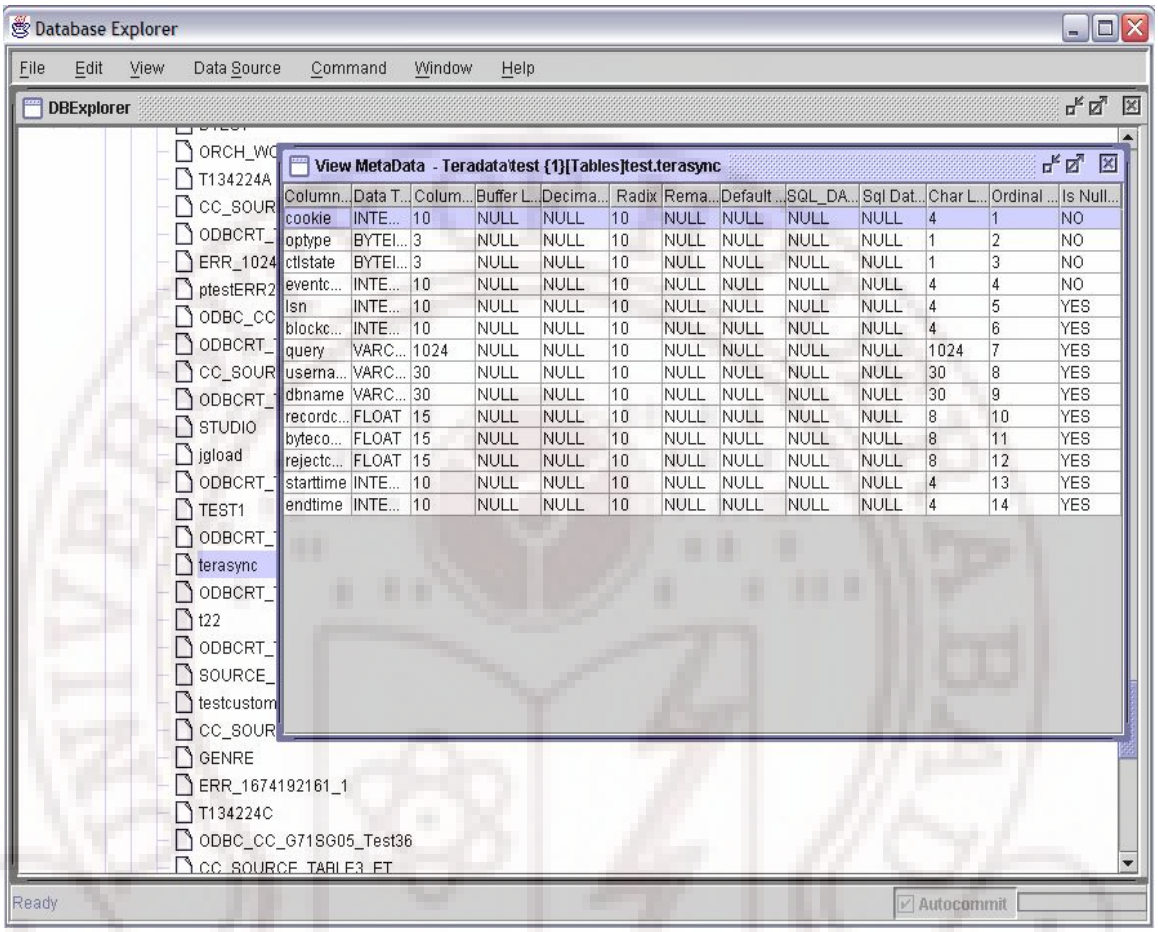


Figure 7.10: View Metadata of a table in Teradata database

The below figure displays the Metadata of the table *terasync.* The fourteen columns data is displayed in a new window.



Column...	Data T...	Colum...	Buffer L...	Decima...	Radix	Rema...	Default ...	SQL_DA...	Sql Dat...	Char L...	Ordinal ...	Is Null...
cookie	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	1	NO
optype	BYTEI...	3	NULL	NULL	10	NULL	NULL	NULL	NULL	1	2	NO
ctlstate	BYTEI...	3	NULL	NULL	10	NULL	NULL	NULL	NULL	1	3	NO
eventc...	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	4	NO
lsn	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	5	YES
blockc...	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	6	YES
query	VARC...	1024	NULL	NULL	10	NULL	NULL	NULL	NULL	1024	7	YES
userna...	VARC...	30	NULL	NULL	10	NULL	NULL	NULL	NULL	30	8	YES
dbname	VARC...	30	NULL	NULL	10	NULL	NULL	NULL	NULL	30	9	YES
recordc...	FLOAT	15	NULL	NULL	10	NULL	NULL	NULL	NULL	8	10	YES
byteco...	FLOAT	15	NULL	NULL	10	NULL	NULL	NULL	NULL	8	11	YES
rejectc...	FLOAT	15	NULL	NULL	10	NULL	NULL	NULL	NULL	8	12	YES
starttime	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	13	YES
endtime	INTE...	10	NULL	NULL	10	NULL	NULL	NULL	NULL	4	14	YES

Figure 7.11: Metadata of a table in Teradata database

To view the users of the database expand the *Users* node under the database. To view the details of the user right click on the user, select *view user details*.

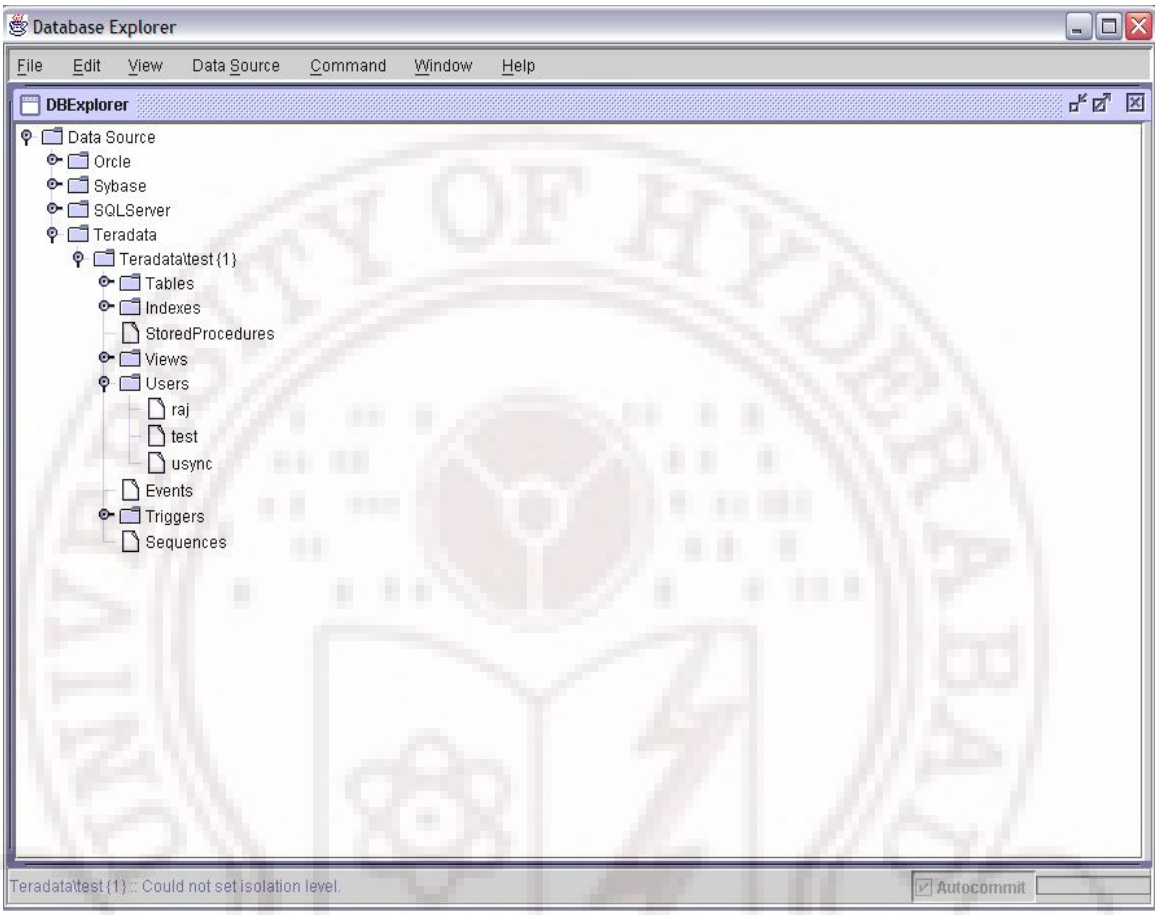


Figure 7.12: Users of Teradata database

Chapter 8

Conclusions and Future work

8.1 Conclusions

In Part one, Statistical Data Collection and Analysis of each Stage of DataStage is performed with the Stage and input and output Stages. The results are stored in table. Which in turn are used to develop a tool for the estimation of required resources that job consumes.

In the Part two, Database Explorer has been extended with the Teradata database. This extension is performed using the metadata of the Teradata database. Now the Database Explorer supports the Teradata Database also.

8.2 Future work

Develop a tool by using the Analysis results of the DataStage. These results are converted to an xml document, and install with the installation process of the IBM Information Server DataStage in future versions.

The databases that are not extended to the Database Explorer cab be extended to it. The Database Explorer tool can be made as a plug-in to Eclipse software.

References

1. IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide, Version 8 Release 1
2. IBM WebSphere DataStage Java Pack guide, Version 8
3. Ascential Software XML pack designer guide, Version 2.2.0, October 2003
4. Quux, “Windows Perfmon”, 04/04/2007, http://adminfoo.net/2007/04/windows-perfmon-top-ten-counters.html
5. Basu.Ds, “FTP-Stage – Invalid URI”, 06/02/2007, found at www.dsxchange.com
6. Tom J Bob, 27/09/2007, IBM internal Video files on DataStage
7. Teradata Database Data Dictionary, Version 2 Release 5.1, April 2004
8. “The Swing Tutorial”, http://java.sun.com/docs/books/tutorial/uiswing/
9. “JDBC Tutorial”, http://java.sun.com/docs/books/tutorial/jdbc/index.html