**Initio**

**GD100**

**INTRODUCTION TO GRAPH DEVELOPMENT**

**Q. What is a graph?**

**Dataflow graphs: -**

With Ab Initio software, you create and execute data processing applications as dataflow graphs.

A dataflow graph – graph for short – is a picture in which data flows from one processing step to another. It’s the kind of picture you might draw when designing an application.

**Filter**

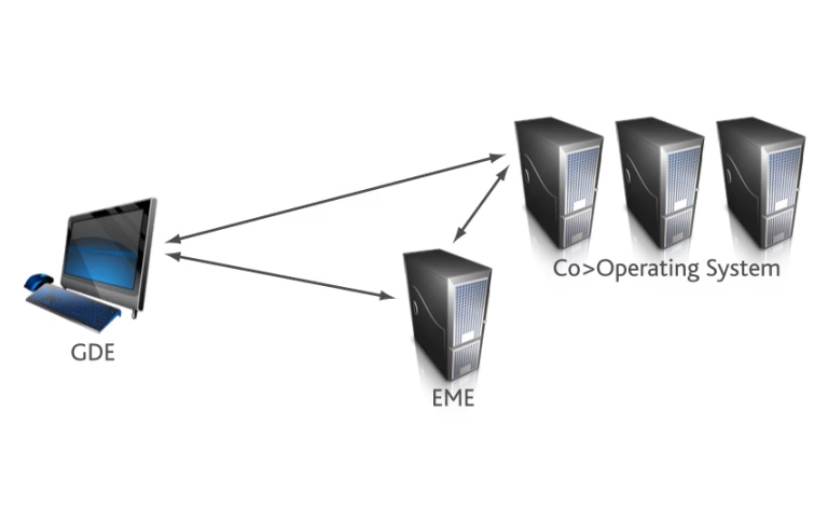
**Reformat**

**Sort**

Each processing step is performed by a component. Different kinds of components provide different kinds of functionality.

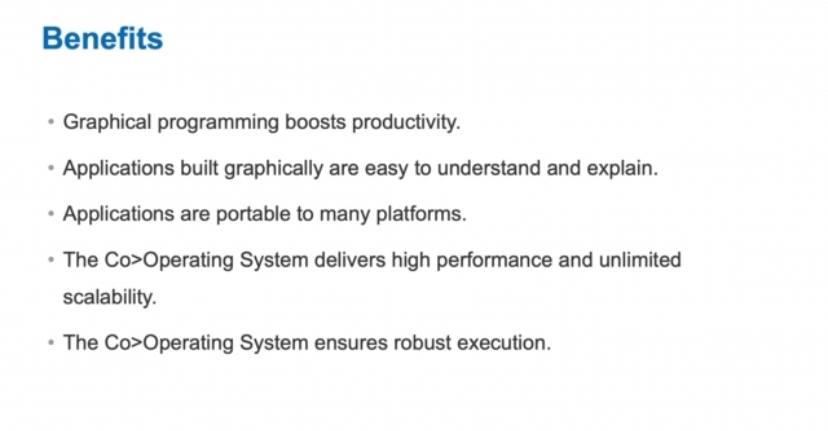
You build applications by choosing and configuring components and connecting them with flows that carry data from one component to another.

**Products: -**

****

* The Graphical Development Environment (or GDE) lets you create and edit graphs. It runs on your desktop computer.
* The Co>Operating System executes graphs on one or more server computers.
* The Enterprise Meta>Environment (or EME) provides source control for graphs and other metadata, including business metadata and operational metadata.

**Benefits: -**

****

- A big reason is that using the GDE boosts productivity. You can create applications much faster by graphically combining components than you can by coding in a conventional programming language.

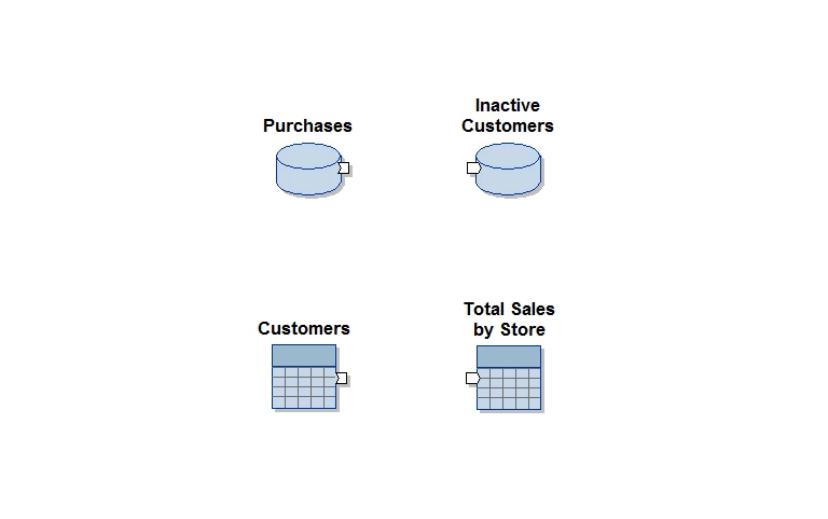
- Visual development brings clarity. It’s a lot easier to understand an application that’s drawn as a diagram than one that’s written as lines of code. When an application is built visually, more people – even non-technical people – can appreciate what it does and how it works.

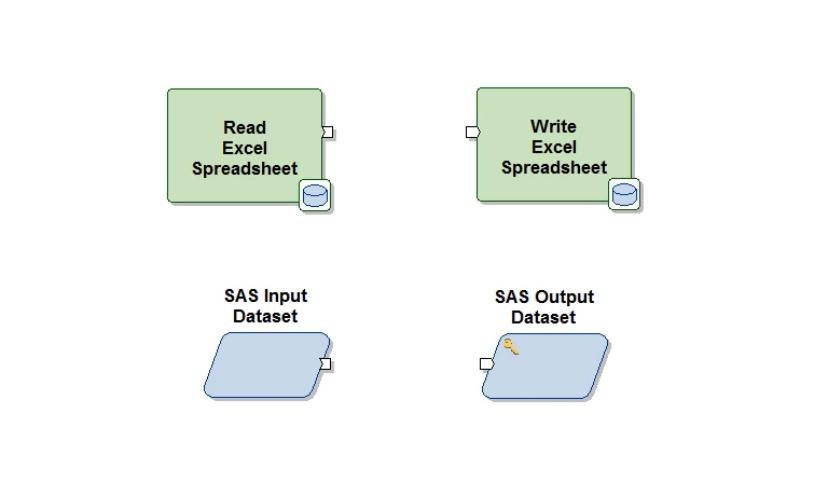
- Applications built with the GDE are highly portable and run identically on virtually any platform: Linux, Windows, AIX, mainframes running Z/OS, and more.

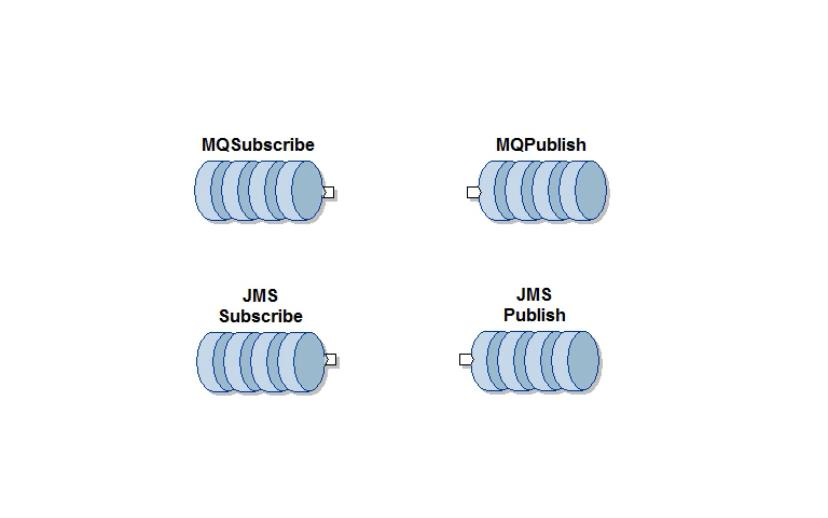
- The Ab Initio Co>Operating System delivers very high performance and unlimited scalability, while at the same time providing a robust execution environment that ensures reliable operation in even the most demanding situations.

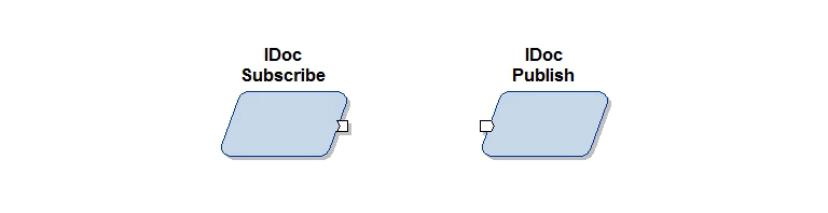
**Q. What can you do with a graph**

To start with, you can read data from and write data to practically anything: flat files, database tables… spreadsheets, SAS datasets… message queuing systems… and software systems like SAP and salesforce.com.

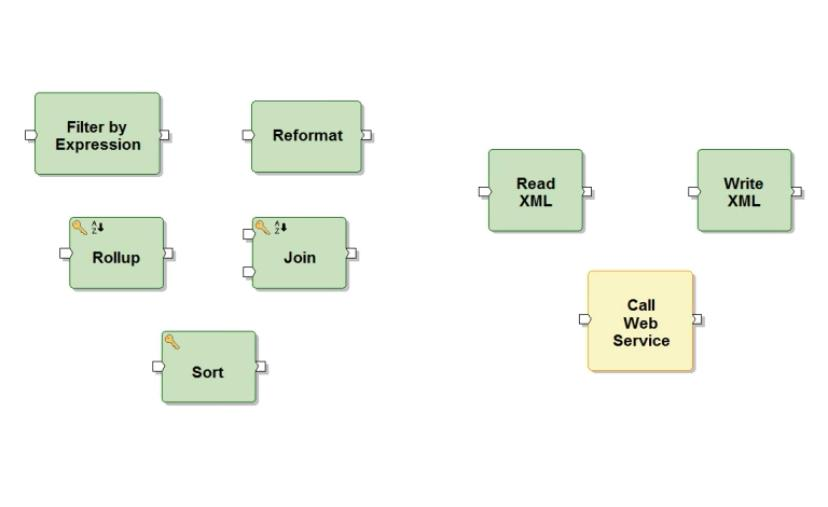






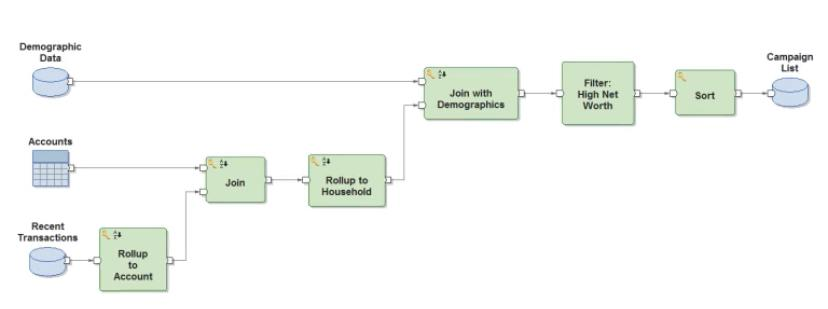


Between these sources and targets – using the processing components – you can transform and manipulate data in practically any way.



Let’s look at a few graphs.

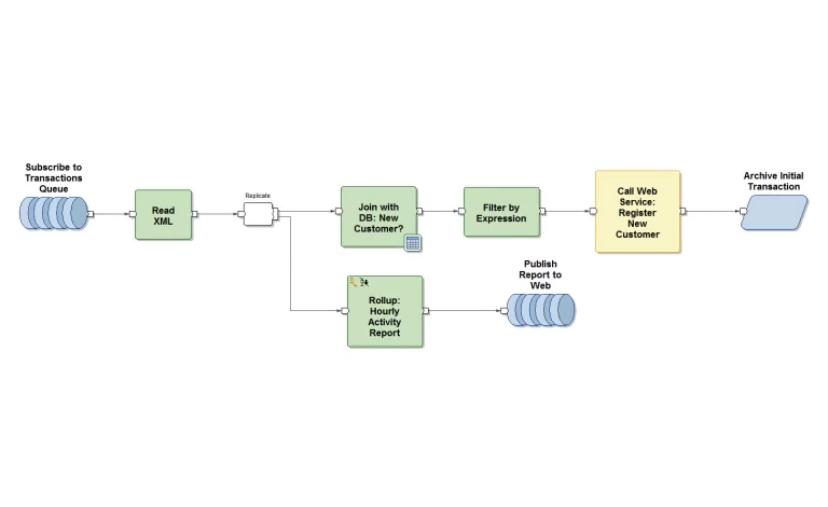
This graph reads transaction data from a file, aggregates it, joins it with account data from a database table, aggregates that, joins the result with a file of demographic information, and then filters, sorts, and writes the result to a file.



This graph reads XML messages from a queue, parses them, and does two things with the result.

First, it queries a database to determine if the transaction is associated with a new customer, and, if so, calls a web service to register the customer and archives the customer’s first transaction.

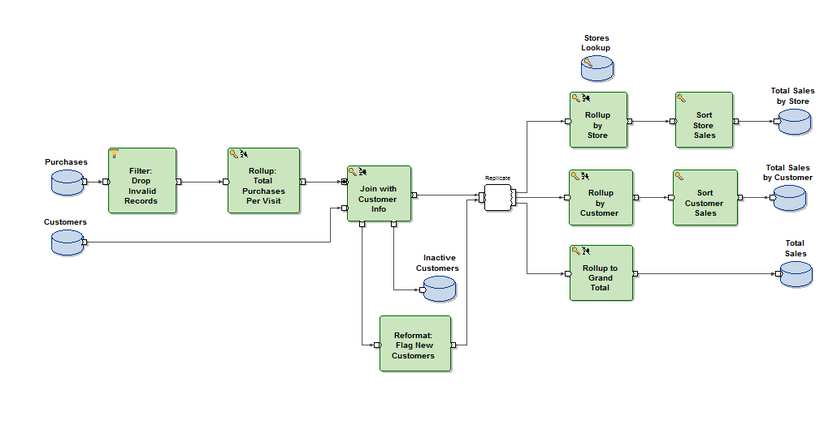
Separately, it produces an hourly report of activity which is published to a web server.

****

Simple building blocks; infinite combinations

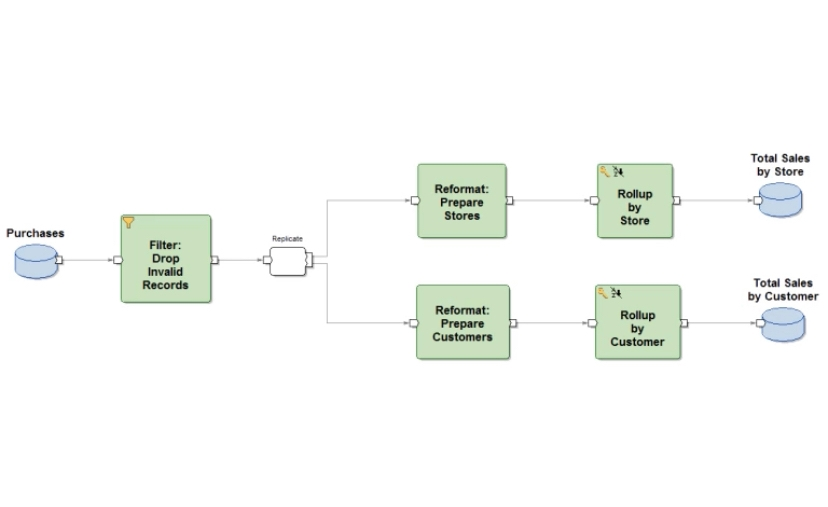
The Co>Operating System has many more components, but these are the most popular, and they account for the vast majority of components in graphs.

These simple building blocks can be combined in a truly infinite variety to perform virtually any data processing task.



**Q. Scalable performance**

* The Co>Operating System executes graphs in an inherently scalable fashion, delivering extremely high performance to applications that need it.
* Drawing an application as a dataflow graph immediately exposes two forms of parallelism.
* Component parallelism is the concurrent processing of data on independent branches of a graph.
* Pipeline parallelism is the concurrent processing of different subsets of data along a single branch, with a series of components that operate in a streaming fashion, each consuming input while producing output, a little at a time.
* While component parallelism and pipeline parallelism are limited by the size of the graph, a third form of parallelism, data parallelism, provides essentially unbounded concurrency.
* To take advantage of data parallelism, a graph simply partitions the data into subsets that can be processed independently. Components running in parallel process these subsets concurrently.



**Summary**

* Those are the basics of building graphs in the GDE: components, flows, and parameters.
* Parameters provide the business logic.
* The components and flows are simply the processing steps and the connections between them.
* A graph is a lot like the picture you might draw when designing an application.
* To go beyond the basics, start working your way through the lessons in this training.

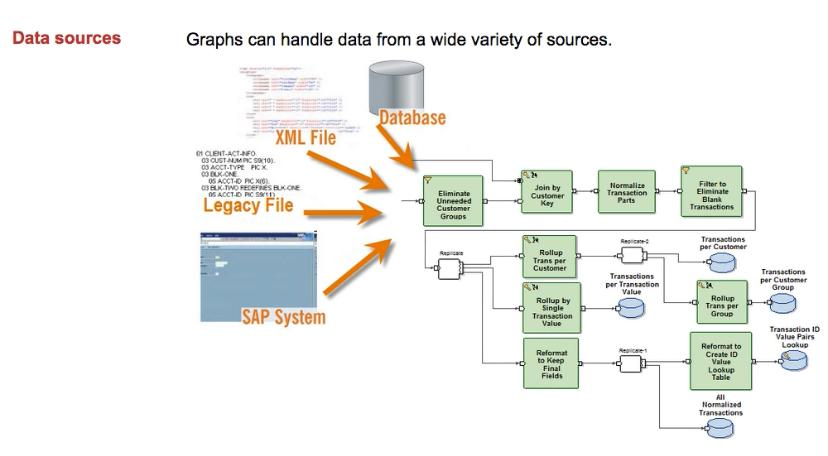
**GD101**

**BASIC GRAPH BUILDING**

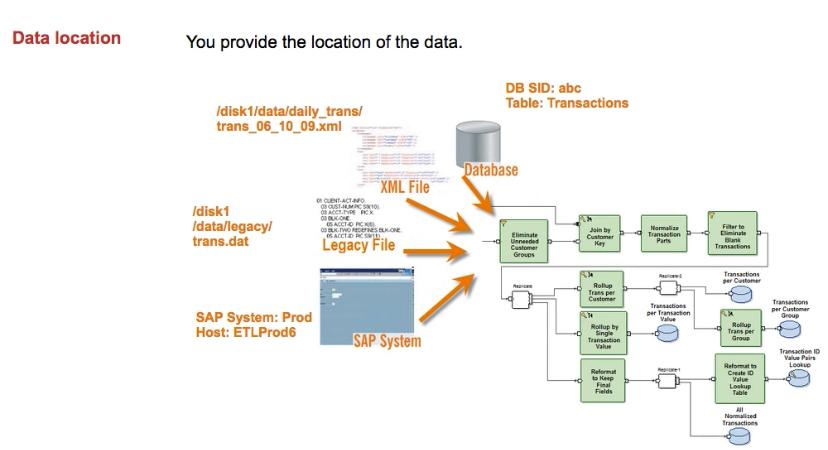
**Data source basics**

**Data source location and record format**

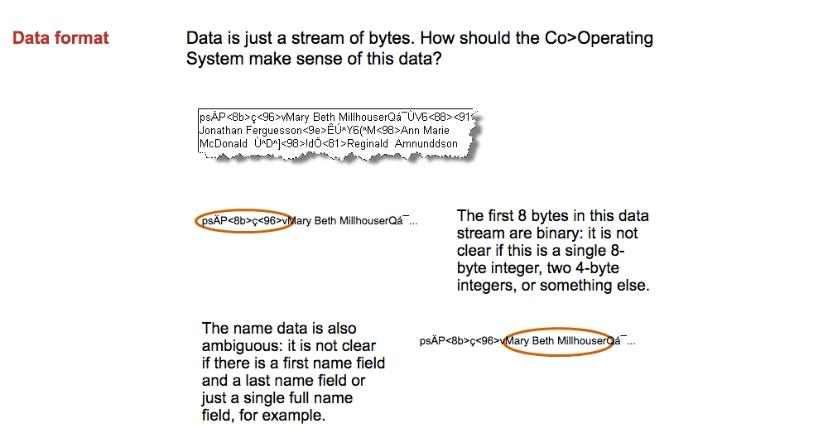
All graphs process data, so every graph includes one or more data sources. Graphs can handle data from a variety of sources. In this lesson, you'll learn how to access data from a flat file and from a table in a database.



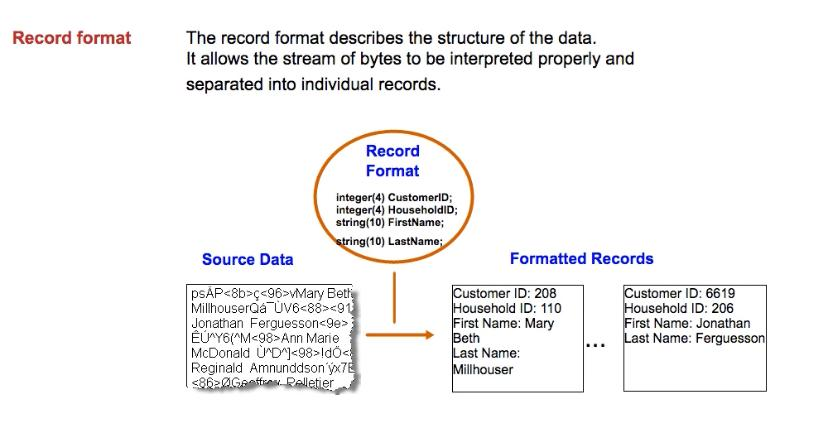
To access a data source in your graph, you provide two essential pieces of information. The first of these is the location. This can be the path to a flat file, the name of a database table, or the path to a legacy file or MVS data set on the mainframe. Whatever the data source, you must provide enough information so the Co>Operating System can locate it.



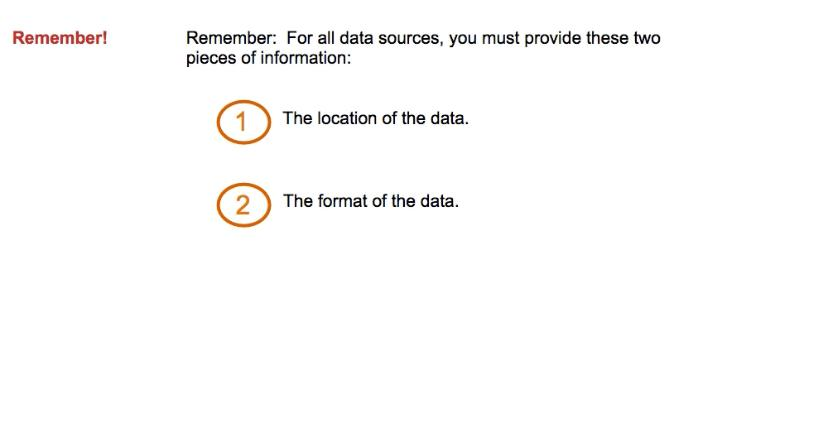
The second essential piece of information you provide is the data format. Without a format, the data is just a stream of bytes and the Co>Operating System can't make sense of it. In the first example, there are eight bytes of binary data. Those bytes could be a single 8-byte integer, two 4-byte integers or various other entities. The record format specifies which one it is.



For any data source, the record format describes the structure of the data including the names of the fields, their data types, lengths and other information. The record format is written using the Data Manipulation Language (DML).



Remember! For all data sources, you must provide these two pieces of information: the location of the data and the format of the data.



**Input File and Input Table**

#### To read data from a database, you use the Input Table component. Both components can be found in the Datasets folder of the Organizer. To use a component, you drag it onto the GDE’s canvas.

#### Once you have created the component, the next step is to configure it. To start, right-click and select “Properties” or simply double-click. The properties dialog opens with several tabs.

#### Every component has a Description tab where you provide a label, or name, for the component and a comment about it. If you give the component a meaningful name, it will aid in understanding the graph. If you provide a comment, it is displayed when you hover over the component.

#### For the Input File component, you provide a path to the file you want to read. Later, you will learn how to use a parameter to hold the pathname, but for now, you will use a hard-coded value to point to the data file containing information about households. Notice the “file:” at the start of the location. This indicates that the file is a serial file. Ab Initio software also supports multifiles.

#### Now that you have indicated where the data is located, you must describe the format of the data. Remember, without a data format, the data is just a stream of bytes and the Co>Operating System can't make sense of it. You provide record formats on the Ports tab of the component's Properties. We will cover record formats in detail in another lesson.

#### You can choose to store the record format with the component or in an external file. Using an external file allows record formats to be shared. Note that the path to the external file storing this record format is referenced using a parameter. In this lesson, you will reference files containing record formats that have already been created. In later lessons, you will learn how to create record formats.

#### We have now seen that a parameter can be used to specify the path to a file – a file containing a record format or a file containing data, for example. Parameters are often used in graphs to specify paths and other types of values because they make graphs more portable and flexible. For example, pathnames can differ from computer to computer. On the computer where you develop a graph, the test data may live in a directory named "/disk1/testdata". But when you run the graph in production, it may need to process data in the directory "/disk5/proddata”. If you did not use a parameter, you would have to change the name in many places in the graph. With a parameter, you make the change in one place only.

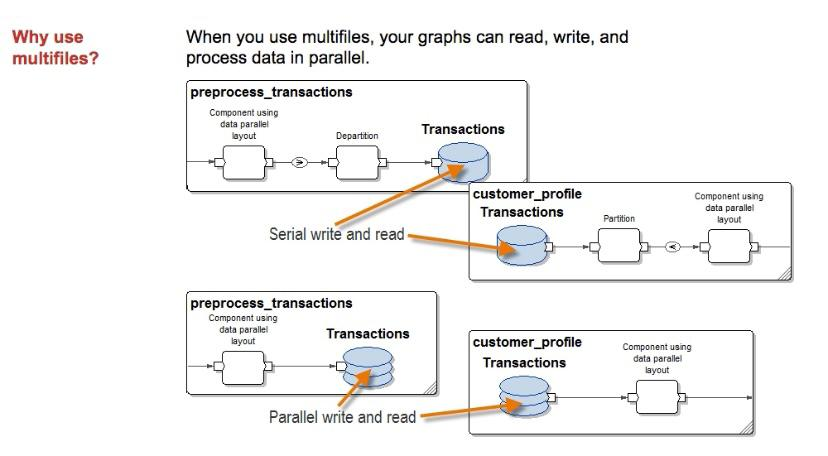
* Used Input File to read data from flat files.
* Used Input Table to read data from database tables.
* Used View Data to view the data accessed from files and tables.
* Used Output File to write data to a file.
* Used Output Table to load data into a database table.

**Using multifiles**

The partitions of multifiles can be read and written concurrently, in parallel. Records are written to or read from each partition at the same time.

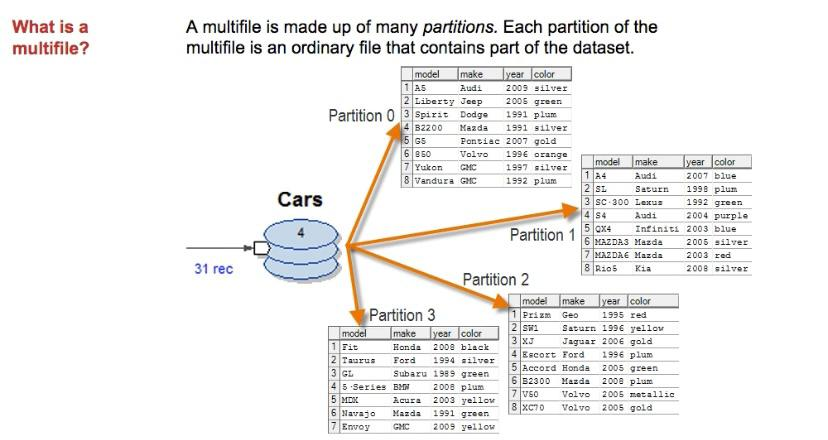
The first example shows two graphs: one writes the data to a serial target, and another uses that same data as a source. The components in the graph run with parallel layouts. Notice that the graphs require additional components -- Partition and Departition -- to read and write the serial files.

The second example shows two graphs that use multifiles. The components with parallel layouts can read and write the data directly from the multifiles. This is a better design: we use fewer components, and the graphs can read and write the data faster. When you’re using data parallelism in a graph, use multifiles as the fastest way to store data in the file system.

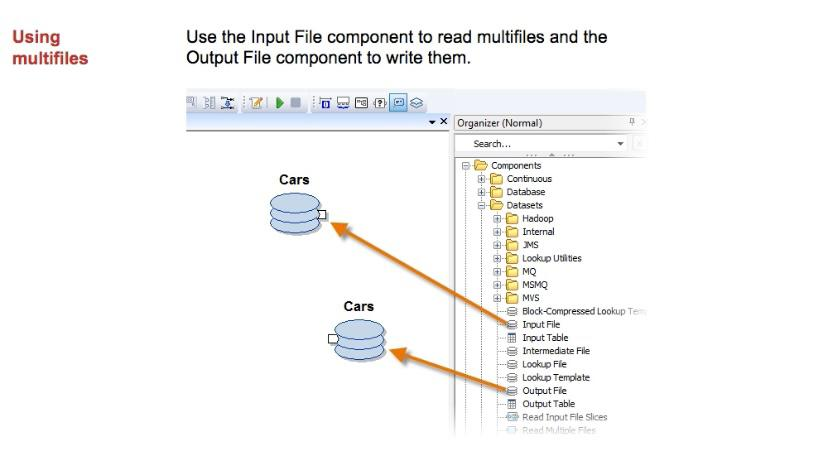


A multifile is made up of many pieces. Each piece, called a data partition, is an ordinary file. The data partitions can be located on different file systems, which can be distributed across different computers or associated with different physical devices.

Here we show how the Cars dataset might be stored in a multifile. The dataset has a total of 31 records. This multifile has 4 data partitions, so the 31 records are divided among the 4 partitions. Partitions are numbered starting at zero. The first partition, partition 0, contains 8 of the Cars records. The second, partition 1, also contains 8 records, and so on.

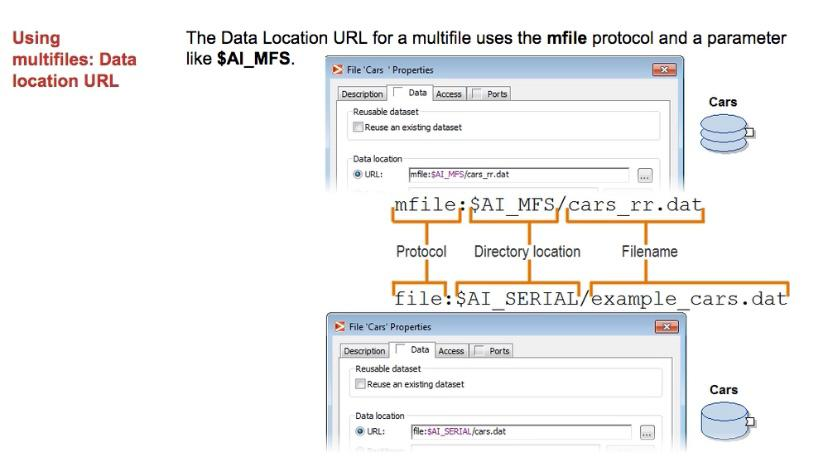


When you want to use a multifile as a data source, you use the Input File component to read the data. When you want to use a multifile as a data target, you use the Output File component to write the data. These are the same components you use for reading serial data sources and writing serial data targets.

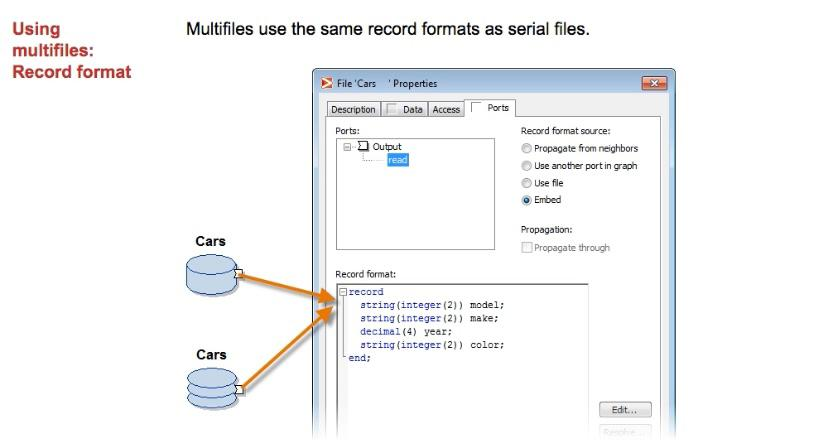


The data location URL for a multifile is different than the URL for a serial file. The first difference is the protocol. The protocol for a multifile is mfile. The protocol for a serial file is file. Depending on the protocol, the GDE changes the icon it displays for the File components. For a multifile, a File component icon has several platters; for a regular file, it has one platter.

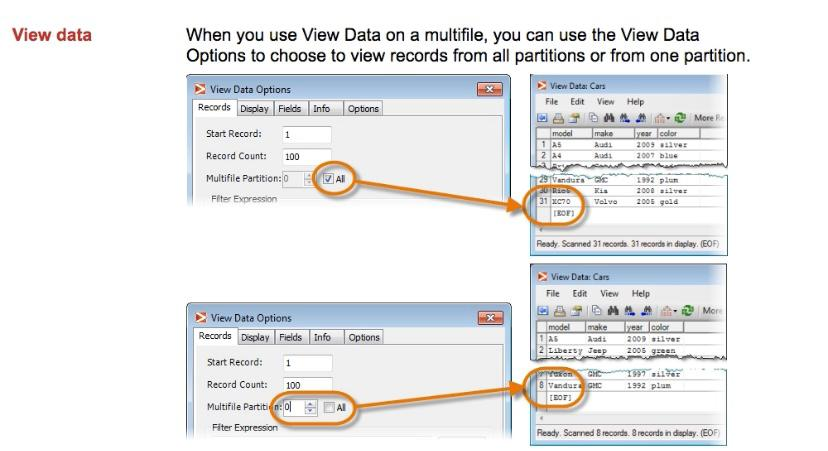
The second difference is the parameter that you use for specifying the directory path to the file. In the Ab Initio Environment, we use parameters like $AI\_SERIAL and $AI\_SERIAL\_REJECT to refer to serial directories. For multifile directories, we use parameters like $AI\_MFS and $AI\_MFS\_REJECT.



The record format for the Cars dataset is the same, whether the data is stored in a serial file or a multifile. Each partition in a multifile is simply an ordinary serial file. The file in each partition of the multifile has the same format as a serial file containing the same data.



When you use View Data with a multifile, you can choose to inspect a single partition or all partitions. In the first example, we see all 31 records from the Cars dataset. If you deselect the All option, you can choose which partition to view. In the second example, we've chosen to see only the records in partition 0 of the multifile.



**Simple components**

A graph processes data with components. Each component performs a separate processing step and different kinds of components process data in different ways:

**the Reformat component transforms data; the Sort component changes the order of records.**

The Organizer groups components by function, with each group of components in its own folder.

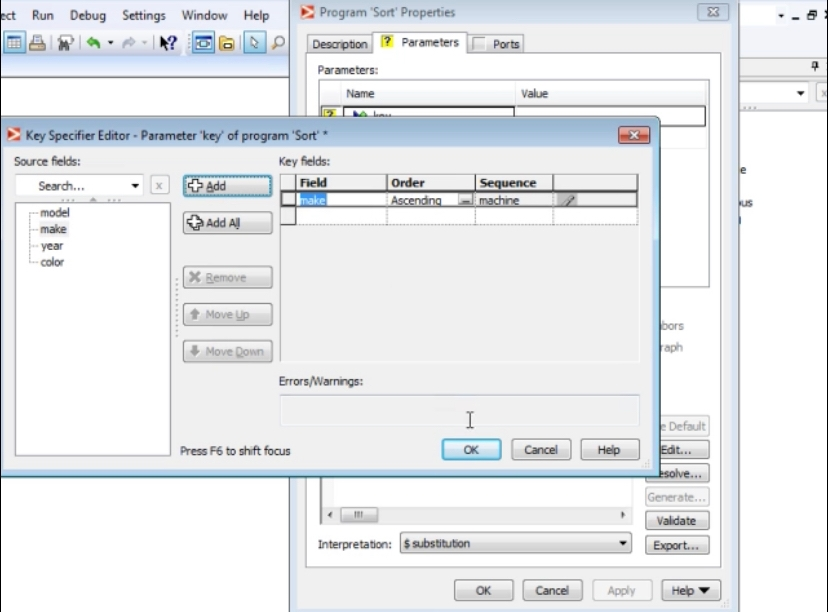
The Datasets folder, for example, contains components that you can use as data sources and targets.

You can find components that sort your data in the Sort folder and components to modify your data in the Transform folder.

**Sort component**

used the Sort component to change the order of records.

Sort components sort the data ascending or descending order.



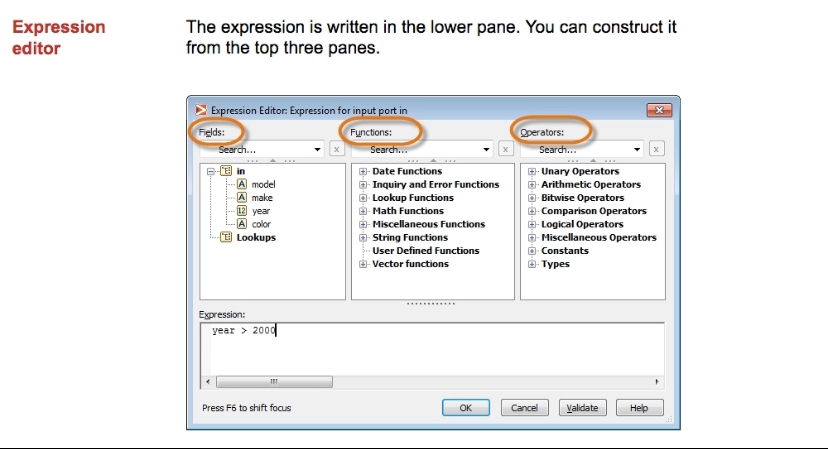
**Filter By Expression component**

**Chosen a subset of records with the Filter By Expression component.**

The selection expression used by the Filter By Expression component is written in DML, which stands for Data Manipulation Language.

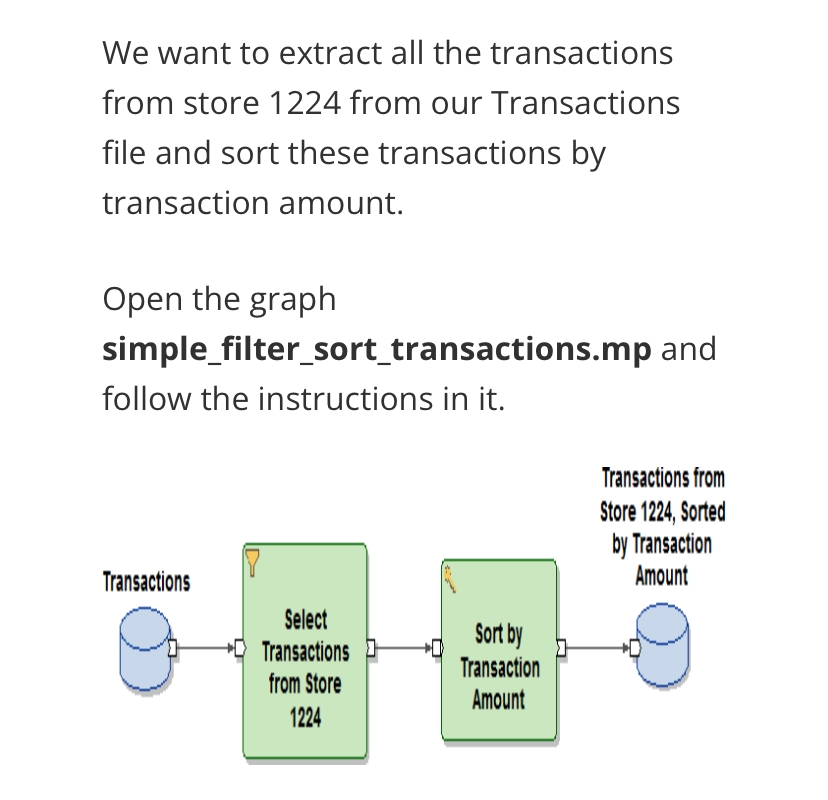
A DML expression is much like an expression in any other programming language -- it is simply a formula used to compute a value.

The Expression Editor shows a simple DML expression: "year > 2000".



The Filter by Expression component evaluates this expression for every record, returning either true or false. Records that result in true are sent to the component's output port; all others are discarded.

The expression is written in the lower pane. You can construct an expression using the fields, functions, and operators shown in the top three panes. The upper left pane shows the fields you can use in constructing an expression. In our example, the input record has four fields: model, make, year, and color. In another lesson, you'll learn about Lookups.



**Modifying data**

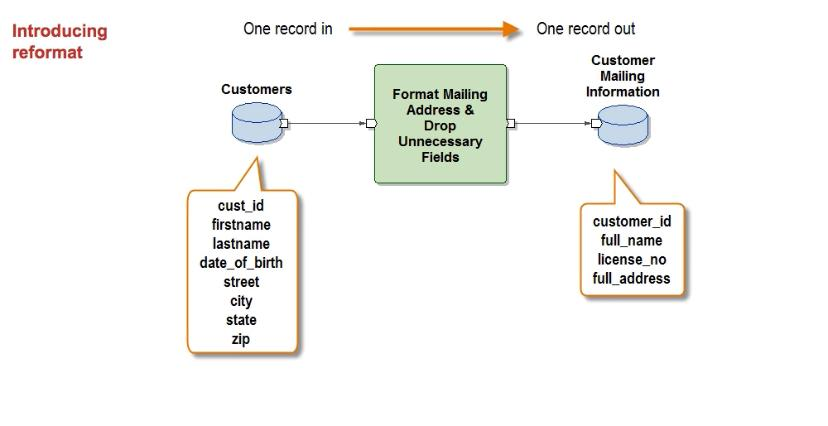
**The Reformat component**

Manipulate records at the field level.

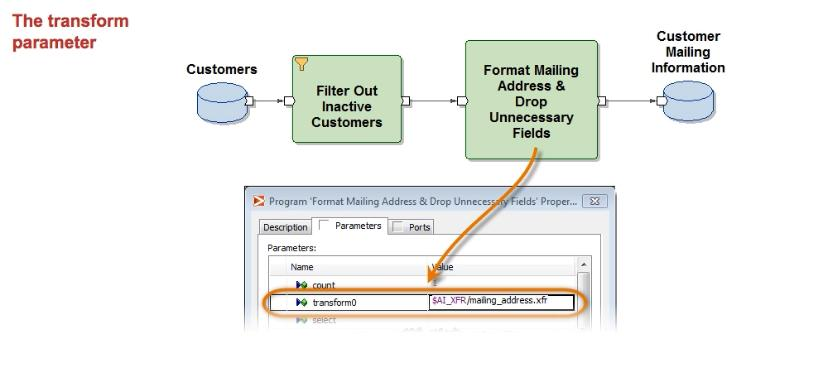
* What if we want to drop fields?
* What if we want to add fields?
* What if we want to modify existing fields?

We can do all these things with the Reformat component!

The Reformat component reads one record at a time and applies a transform function to it to create an output record in the target format.



The transform function provides rules for how to transform the input record into the output record. Most components in the Transform folder of the Organizer have one or more transform parameters, defined on the Parameters tab of the component.



To edit a transform function, we double-click on the component’s “transform” parameter.

This opens the transform editor, a graphical editor that helps create and edit transform functions.

The component’s input fields are displayed on the left, the outputs on the right.

In the middle are the rules that specify how to transform the inputs into the outputs.

There are several ways to create transform rules.

We can drag an input into a rule and then connect it to the appropriate output.

Or we can drag an input field and drop it directly on an output, or we can enter a rule manually.

Here we concatenate the first name, plus a space, plus the last name to create the full name field.

For more complex rules, we may want to use the expression editor.

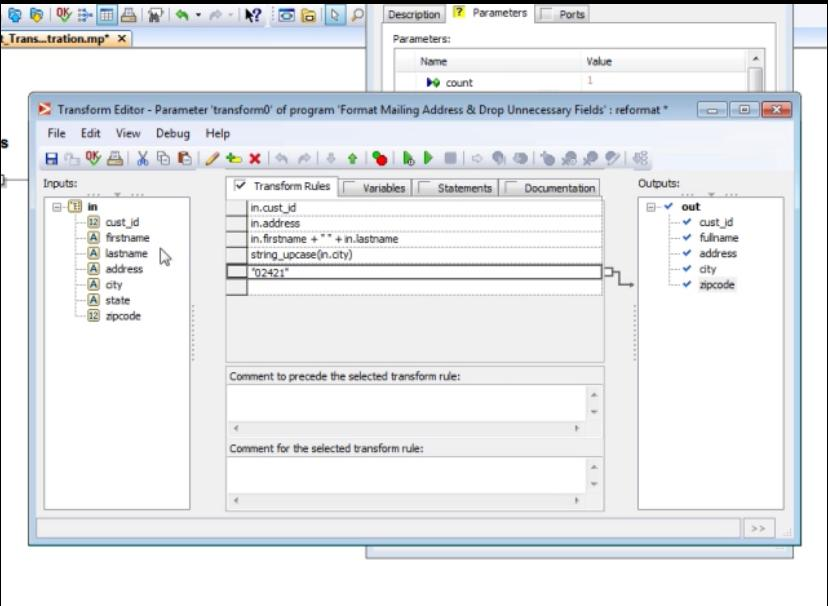
To open it, right-click on a rule, select “Edit Rule” and construct your expression.

When we’re finished, we can validate the expression and select “OK” to close the editor.

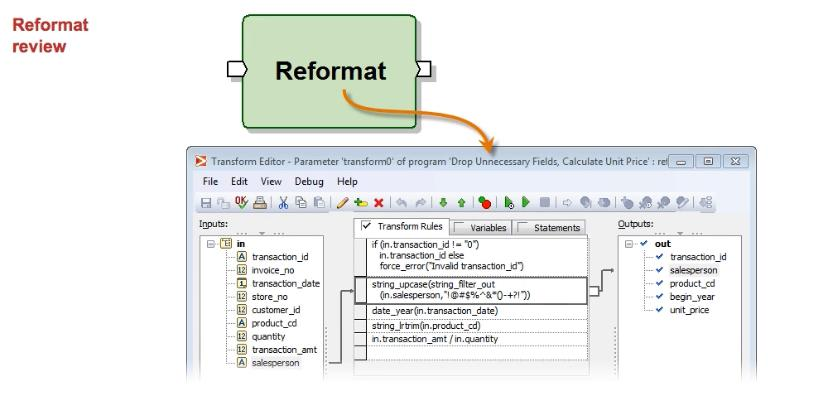
Of course, some rules are simply constants

Once we’ve defined our transform function, there should be a checkmark next to each output field.

We validate the transform by clicking on the “OK” button, save it, and close the editor.

****

transforming data with the Reformat component. Reformat uses the Transform Editor and record format propagation to perform a number of tasks, including adding, dropping, and modifying fields within a record.



Reformat is a powerful component that can perform many tasks. However, there are some things Reformat cannot do, such as bringing in another data source, grouping records by a key field, keeping track of records within a key group, or splitting a single record into multiple output records.

****

**The Join component**

**Join types and unused port**

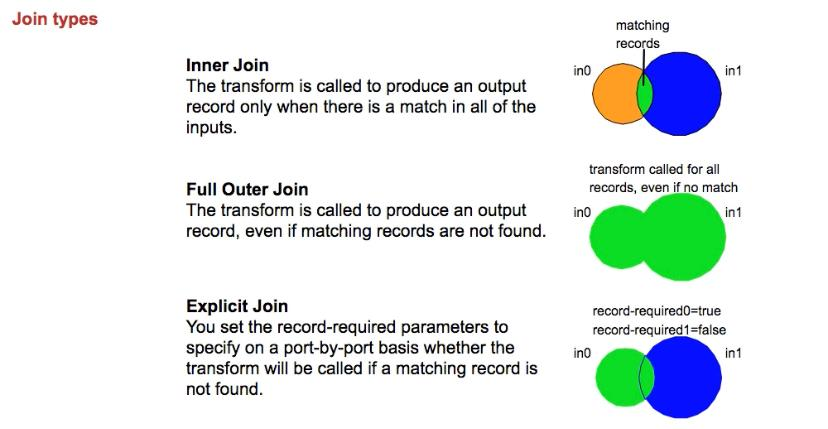
The second factor that influences the number of records produced by a join is the join type. The three possible join types are inner join, outer join and explicit join. In an inner join, the transform is called to produce an output record only when there is a match in all of the inputs.

**In an outer join,** the transform is called to produce an output record, even if matching records are not found. Therefore, the transform must account for the situation where a record does not have a match in another data set. You'll hear more about that soon.

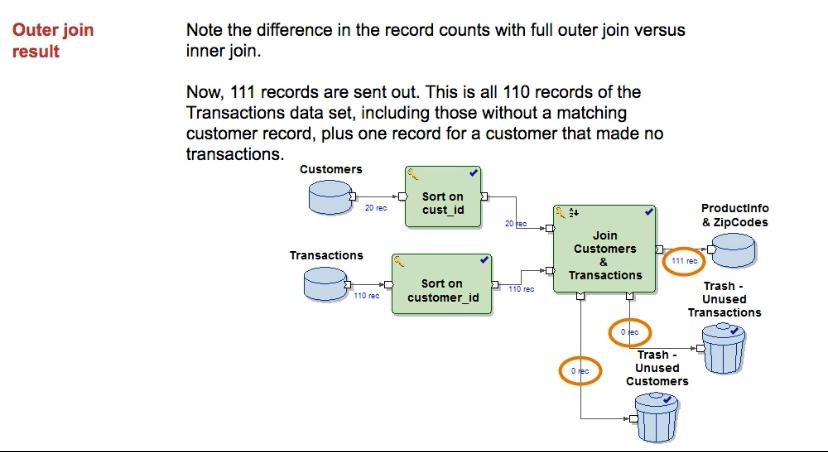
**In an explicit join,** you can set the record-required parameters to specify for each port whether the transform is called when a matching record is not found. For example, you might have two inputs where record-required0 is true and record-required1 is false. The transform is called for every in0 record, regardless of whether there is a matching record in in1. Again, you must account for records without matches in your transform.

**With an inner join,** some input records may not have a match. Those records are sent to the corresponding unused port. So, records from in0 that don't have a match are sent to unused0, and records from in1 that don't have a match are sent to unused1. If we show these optional ports and connect components to them, we can see how many records from each input are unused.

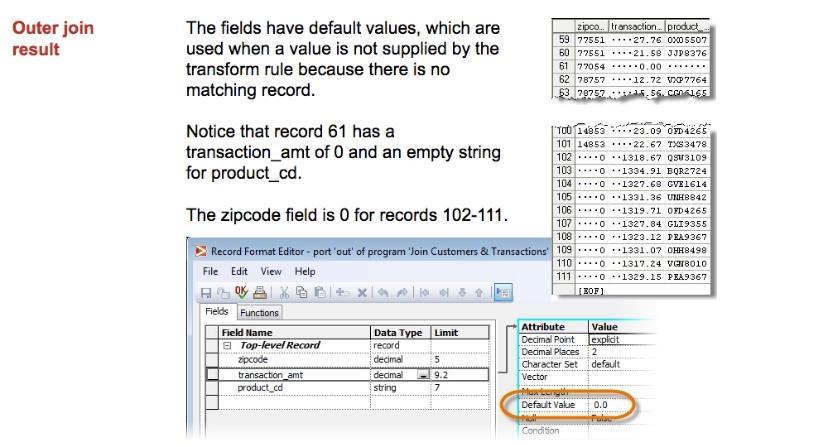
For each join type, records that don’t have a match are sent to the corresponding unused port. Matching records are sent to the component’s transform.

****

The results from running the full outer join graph should look like this. As you see, 111 records are produced by the Join component. The output includes all 110 records from the Transactions data set, including records without a matching customer record. The last record is for a customer who made no transactions.

****

Recall that our transform has rules that use values from both input data sets. With a full outer join, it's possible that one of the inputs will not be present. So, your transform must account for this possibility. You can do that in a number of ways. You can give the output field a default value or allow it to be NULL. You’ll learn more about these in an upcoming lesson. In another lesson, you’ll learn another technique, prioritized rules, that allows you to provide several rules for a given output. For now, we are using a record format where the fields have default values.

****

**GD103**

**Transforms in detail**

**Wildcard rules**

In this transform, each of the input fields is mapped to an output field of the same name. With large numbers of fields, this approach has obvious drawbacks. A better, more flexible approach is to use a wildcard rule.

A wildcard rule is a single rule with a value like "in.\*”. It assigns input fields to identically-named output fields. The wildcard rule is used if no other rule assigns the output field.

In this example, the wildcard rule applies to every field. There’s no need to connect the rule to any outputs; it will apply wherever it is needed.

**Q. What is Abinitio?**

“Abinitio” is a latin word meaning “from the beginning.”

Abinitio is a tool used to extract, transform, and load data.

It is also used for data analysis, data manipulation, batch processing, and graphical user interface based parallel processing.

Ab Initio is popular software that offers data processing applications and enterprise application integration. It provides a single platform for data analysis, complex events, batch processing, data manipulation, quantitative, and qualitative data processing.

In this post, our experts have put together the top 30 Ab Initio interview questions that help you through your interview process. Go through the below Ab Initio interview questions and answers to enhance your skill.

Ab Initio software helps you build large-scale data processing applications and run them

in parallel environments.

Ab Initio software consists of two main programs:

**-Co>Operating System**, which your system administrator installs on a host UNIX

or Windows NT Server, as well as on processing nodes. (The host is also referred to as

the control node.)

-**Graphical Development Environment (GDE),** which you install on your PC

(Client node) and configure to communicate with the host (control node).

**Applications of Ab initio**

* It is used for application development and execution environments.
* It is a tool used in most of the big industries like insurance, banking, logistics, stock market, retail, finance, telecommunication to process complex and enormous volumes of data which gives stability and reliability.
* Ab initio solves the most challenging data processing issues for many of the leading organizations in many fields like finance and insurance.
* ETL for data warehouses, data marts and some operational data sources.
* It is also used for the parallel data cleansing and also for the validation.
* It can be seen in the parallel data transformation and also in the filtering.
* Ab initio ETL tool can also be seen in the high-performance analytics.
* It can be seen in real time and also in the parallel data capture.

**BASIC TERMS of In Ab Initio**

**Q. What is Dataset ?**

In Simple terms dataset is a file. A file can be main frame file or any fixed or delimited

files.

There are various **types of datasets**

FIDXED EBCDIC

FIXED ASCII

DELIMITED ASCII

SAS dataset etc...

You can also think of dataset as a table in database world.

**Q. What is Component?**

Component is Ab Initio Program

There are various components like SELECT, FILTER, SORT, JOIN, MERGE, DEDUP,

ROLLUP, SCAN, USER DEFINED etc.…

**Q. What is Port?**

A port is a connection point for the input or output to a component.

**Q. What is Flow?**

A flow carries a stream of data between components in a graph. Flows connect

components via ports.

Ab Initio supplies four kinds of flows with different patterns:

straight, fan-in, fan-out, and all-to-all. We will discuss various kinds of flows as we go

through this training.

**Q. What is Graph?**

A graph is a diagram that defines the various processing stages of a task and the streams

of data as they move from one stage to another. Visually, stages are represented by

components and streams are represented by flows. The collection of components and

flows comprise an Ab Initio graph.

**Q. What is Field?**

A filed is Equivalent to column of a table in Database world. Filed is also called variable

**Q. What is Key?**

Key is used many places in Ab Initio Development, We use key to sort the data, Join

two files, Rollup data etc...

**Important Components -**

* Reformat
* Rollup
* Normalize
* Scan
* Input file
* output file
* Sort

#### **Q. What are the components or functions available in ab initio?**

The main components in ab initio are here below,

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| Dedup | To remove duplicates |
| Join | To join multiple input dataset based on a common key value. |
| Sort | This component reorders the data. It takes the collation order and dumps data to memory. |
| Filter | Any conditional related removal of data. |
| Replicate | This is component is mainly for parallelism as an additional copy of data is useful while any other nodes go unavailable. |
| merge | This component is to combine multiple input data. |

**Q. Basic Components**

An example graphs

Let’s look at a slightly more complex version of the last application.

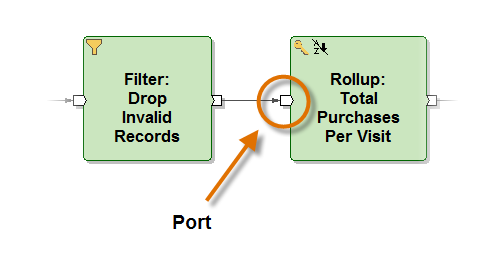
* This graph reads data from three files: one containing information about items purchased, another containing information about customers, and a third with information about stores.
* It produces three sales reports and a file of inactive customers.
* It uses several different kinds of processing components: Filter by Expression, Rollup, Join, Reformat, Replicate, and Sort.

**Diagram

Description automatically generated**

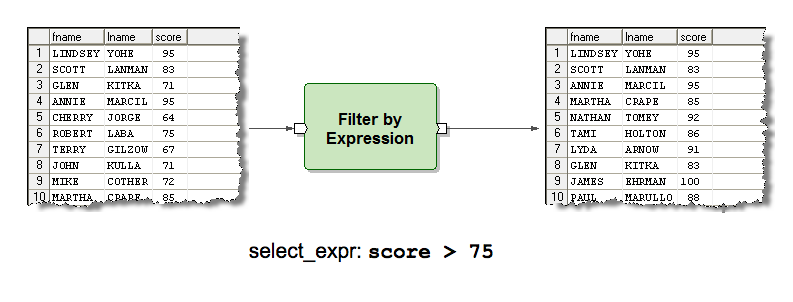
**port**

The place where a flow connects to a component is called a port**.**



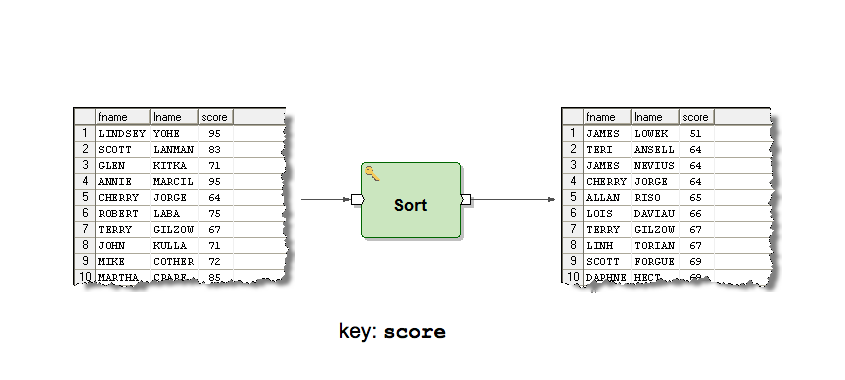
**Filter by Expression**

The Filter by Expression component reads records from its input port and evaluates a selection expression for each one. If the result is true, it writes that record to its output port.



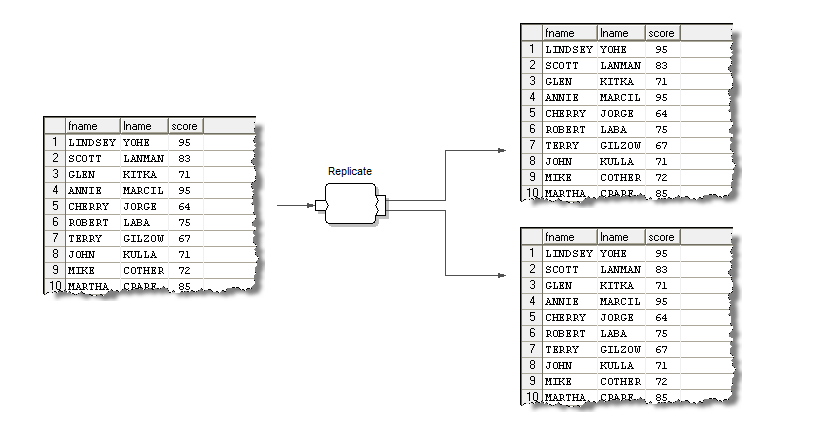
**Sort**

The Sort component reads records from its input port, sorts them by one or more key fields, then writes the records in sorted order to its output port.



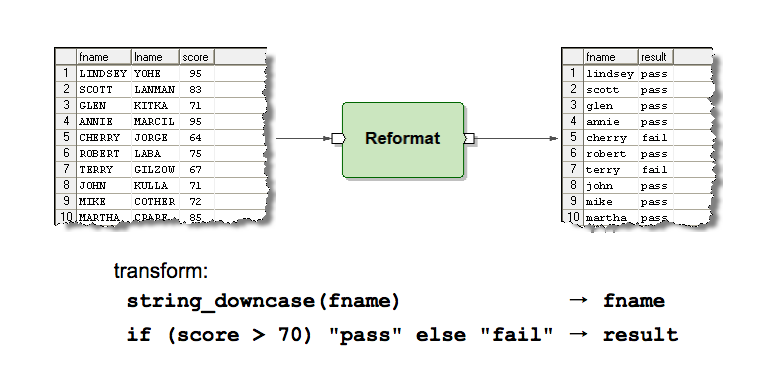
**Replicate**

The Replicate component simply copies every record arriving on its input port to every flow connected to its output port.



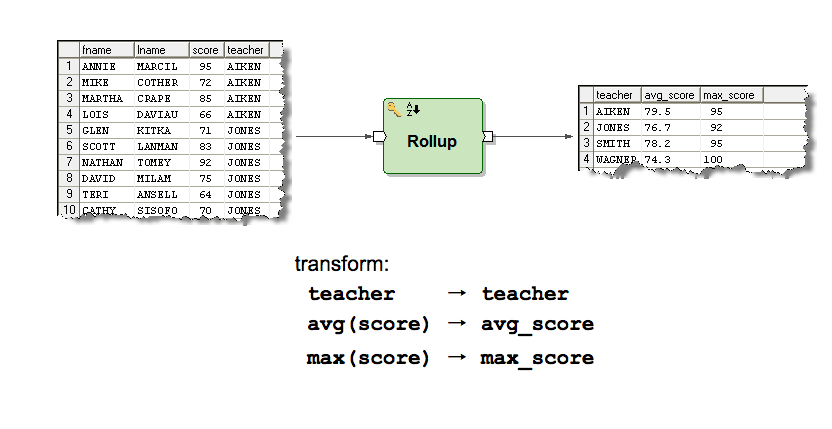
**Reformat**

The Reformat component reads records from its input port and, for each, applies a set of rules to create a new record, which it writes to its output port.

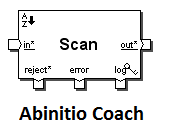


**Rollup**

The Rollup component aggregates collections of records grouped by a key. It applies a set of rules to produce an output record for each group.

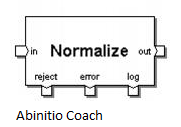


**Scan Component**



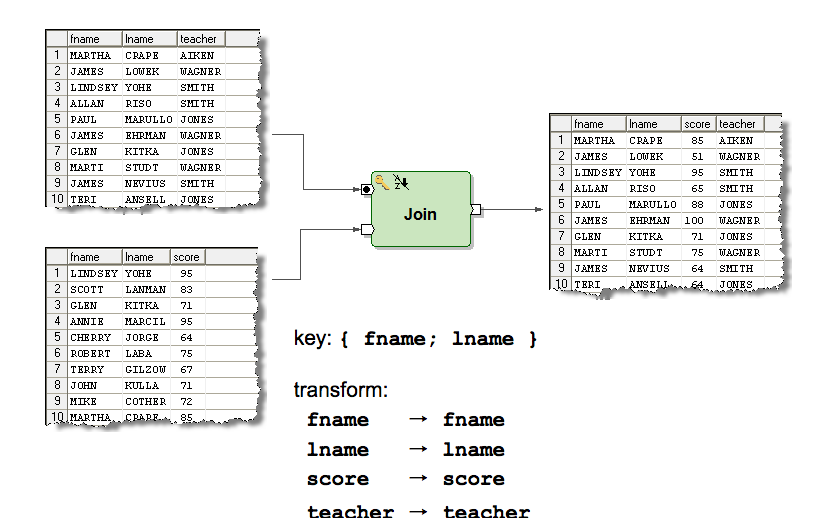
For every input record, Scan generates an output record that consists of a running cumulative summary for the group to which the input record belongs, up to and including the current record. For example, the output records might include successive year-to-date totals for groups of records.

**Normalize Component**

Normalize component is also one of the multistage components in ab initio.  
  
**Purpose**  
Normalize generates multiple output records from each of its input records. You can directly specify the number of output records for each input record, or you can make the number of output records dependent on a calculation.  
In contrast, to consolidate groups of related records into a single record with a vector field for each group — the inverse of NORMALIZE — you would use the accumulation function of the ROLLUP component.

**Join**

The Join component combines the data arriving on its input ports. Each pair of input records with matching keys is combined using rules to produce an output record.



#### **Q. What are the components or functions available in ab initio?**

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| Dedup | To remove duplicates |
| Join | To join multiple input dataset based on a common key value. |
| Sort | This component reorders the data. It takes the collation order and dumps data to memory. |
| Filter | Any conditional related removal of data. |
| Replicate | This is component is mainly for parallelism as an additional copy of data is useful while any other nodes go unavailable. |
| merge | This component is to combine multiple input data. |

#### **Q. List the commonly used components in an Ab Initio graph?**

The commonly used components in an Ab Initio graph are,

* Input file/output file
* Lookup file
* Input table/output table
* Join
* Sort
* Partition
* Partition by key
* Gather
* Reformat
* Concatenate

**Q. What are the types of parallel processing?**

types of parallel processing are:

**Component parallelism:** An application that has multiple components running on the system simultaneously. But the data are separate. This is achieved through component level parallel processing.

**Data parallelism:** Data is split into segments and runs the operations simultaneously. This kind of process is achieved using data parallelism.

**Pipeline parallelism:** An application with multiple components but running on the same dataset. This uses pipeline parallelism.

**Q. Explain the architecture of Abinitio?**

* Co>Operating system (Co>Op)
* Component Library
* Graphical Development Environment (GDE)
* Enterprise Meta>Environment (EME)
* Data Profiler
* Conduct>IT

**Q. Explain the relation between eme, gde, and co-operating system.**

**Co>Operating System** is installed on the operating system.

CoOperating System It operates on top of the operating system, and the ab initio provides this and it the base for all Ab Initio processes.

Air commands are one of the features that can be installed on different operating systems like UNIX, Linux, IBM etc

This co-op is installed on particular o.s platform that is called native o.s .coming to the eme, it’s just as a repository in Informatica, its hold the metadata, transformations, dbconfig files source and targets information.

**These are the following features that it provides,**

– Manages and runs Ab Initio graphs and control the ETL processes

– Providing the extensions

– ETL processes monitoring and debugging

– Metadata management and interaction with the EME

**Enterprise Meta>Environment (EME)**

Is nothing but a repository for storing and managing metadata.

Enterprise Meta-Environment (EME): It’s an environment for storage and also metadata management (Both business and technical metadata).

The metadata is accessed from the graphical development environment and also the web browser or the cooperating command line. It is an ab initio repository for any placeholders.

Eme is said as enterprise metadataenv, gde as graphical development env and the co-operating system can be said as abinitio server relation b/w this co-op, eme and gde are as follows operating system is the abinitio server.

**Graphical Development Environment (GDE)**

Is the graphical application for designing and running Ab Initio graphs.

Users can access the EME metadata through the GDE web browser or Co>Operating system command line.

Coming to gde its is an end-user environment where we can develop the graphs (mapping just like in Informatica) designer uses the gde and designs the graphs and save to the eme or sandbox it is at the user side. Where eme is at the server-side.

**GDE** It’s a designing component and used to run the ab initio graphs.

Graphs are formed by the components (predefined or user-defined) and flows and the parameters. It provides the ETL process in Ab Initio that is represented by graphs.

Ability to run, debug the process logs jobs and trace execution logs

**Q. Mention what is the role of Co-operating system in Abinitio?**

* Manage and run Abinitio graph and control the ETL processes
* Provide Ab initio extensions to the operating system
* ETL processes monitoring and debugging
* Meta-data management and interaction with the EME

#### Co>Operating System operates on the top of the operating system and works as a base for all Ab Initio processes.

#### It can run on operating systems like Windows, Linux, Solaris, AIX, HP-UX, and z/OS.

#### It provides Ab Initio extensions through which ETL processes can be controlled. It manages metadata by interacting with EME, manages, and runs Ab Initio graphs.

#### **Q. What is the different way to achieve the partitions?** There are multiple ways to do the partitions.

|  |  |
| --- | --- |
| **Partitions** | **Description** |
| **Expression** | Data split according to the data manipulation language. |
| **Key** | Grouping the data by specific keys |
| **Load balance** | Dynamic load balancing |
| **Percentage** | Segregate the data where the output size is on the fractions of 100 |
| **Range** | Split the data evenly based on a key and a range among the nodes |
| **Round robin** | Distributing the data evenly in blocksize across the output partitions. |

**Q.** **Explain what does dependency analysis mean in Abinitio?**

In Ab initio, dependency analysis is a process through which the EME examines a project entirely and traces how data is transferred and transformed- from component-to-component, field-by-field, within and between graphs.

The dependency analysis is a process through which the EME analyzes the project for the dependencies within and between the graphs. It examines the entire project and tracks how data is being transformed and transferred from component to component and field by field. The steps involved in dependency analysis are translation and analysis.

Dependency analysis is a procedure through which the EME examines a project completely and traces how data is transferred and distorted from field-by-field, component-to-component, within and among graphs.

Dependency analysis will answer the questions regarding data lineage. That is where does the data come from, what applications produce and depend on this data etc.  
We can retrieve the maximum (surrogate key) from the existing data, the by using the scan or next\_in\_sequence/reformat we can generate a further sequence for new records. ([E Learning pottal](https://svrtechnologies.com/)).

**Q. Explain how Abinitio EME is segregated?**

* Data Integration Portion
* User Interface ( Access to the meta-data information)

The Ab Initio EME is logically segregated into data integration portion and user interface to access metadata information

**Q. Mention how can you connect EME to Abinitio Server?**

To connect with Ab initio Server, there are several ways like

* Set AB\_AIR\_ROOT
* Login to EME web interface- http://serverhost:[serverport]/abinitio
* Through GDE, you can connect to EME data-store
* Through air-command

**Q. List out the file extensions used in Abinitio?**

The file extensions used in Abinitio are

* .mp: It stores Ab initio graph or graph component
* .mpc: Custom component or program
* .mdc: Dataset or custom data-set component

dataset template files or custom dataset components

* .dml: Data manipulation language file or record type definition

record format files

* .xfr: Transform function file
* .dat: Data file (multifile or serial file)
* .dbc - database table files
* .ksh - shell scripting file

**Q. Mention what information does a .dbc file extension provides to connect to the database?**

* The .dbc extension provides the GDE with the information to connect with the database are
* Name and version number of the data-base to which you want to connect
* Name of the computer on which the data-base instance or server to which you want to connect runs, or on which the database remote access software is installed
* Name of the server, database instance or provider to which you want to link

**Q. What information does a .dbc file extension provide to connect to the database?**

* The .dbc file provides the below information to connect to a database.
* Name and version number of the database
* Name of the system on which the database server or instance is running.
* Name of the server or database instance

**Q. Explain how you can run a graph infinitely in Ab initio?**

To execute graph infinitely, the graph end script should call the .ksh file of the graph. Therefore, if the graph name is abc.mp then in the end script of the graph it should call to abc.ksh. This will run the graph for infinitely.

Calling the .ksh file of the graph at the end script runs the graph infinitely. If the graph name is xyz.mp, then the end script of the graph should make a call to xyz.ksh.

**Q. Mention what the difference between “Look-up” file and “Look is up” in Abinitio?**

**Lookup file**

Lookup file defines one or more serial file (Flat Files); it is a physical file where the data for the Look-up is stored.

A lookup file represents one or more serial files, also known as Flat files. It is a physical file where lookup data is stored, which is small enough to be held in memory.

**Look-up**

While Look-up is the component of abinitio graph, where we can save data and retrieve it by using a key parameter.

Lookup is a component of the Ab Initio graph where data resides along with a key parameter. The data can be retrieved using this key parameter.

**Q.** **What is a local lookup?**

The local lookup function will be used before the lookup function call when the lookup file is a multifile and partitioned/sorted on a particular key.

It will be local to a partition, depending on the key. The data records in the lookup file can be loaded into memory. This way, the transform function retrieves records faster than retrieving from disk.

**Q. Mention what are the different types of parallelism used in Abinitio?**

Different types of parallelism used in Abinitio includes

* **Component parallelism:**

A graph with multiple processes executing simultaneously on separate data uses parallelism

It is used by the graph that has multiple processes executing simultaneously on separate data.

* **Data parallelism:**

A graph that works with data divided into segments and operates on each segments respectively, uses data parallelism.

It is used by the graph that works with data divided into segments, which operates on each segment, respectively.

* **Pipeline parallelism:**

A graph that deals with multiple components executing simultaneously on the same data uses pipeline parallelism. Each component in the pipeline read continuously from the upstream components, processes data and writes to downstream components. Both components can operate in parallel.

It is used by the graph that deals with multiple components executing simultaneously on the same data.

**Q. Explain what is Sort Component in Abinitio?**

The Sort Component in Abinitio re-orders the data. It comprises of two parameters “Key” and “Max-core”.

**Key:**

* It is one of the parameters for sort component which determines the collation order
* It represents the collation order.

**Max-core:**

* This parameter controls how often the sort component dumps data from memory to disk.
* It defines how often the sort component should dump data from memory to disk.

**Q. Mention what dedup-component and replicate component does?**

**Dedup component:**

* It is used to remove duplicate records.
* It is used to remove duplicate records from the flow based on a specified key.

**Replicate component:**

* It combines the data records from the inputs into one flow and writes a copy of that flow to each of its output ports.
* It is used to combine input records from multiple sources into one flow and write a copy of that flow to output ports.

#### **Q. What are the types of partition components in Ab Initio?**

**Q. Mention what is a partition and what are the different types of partition components in Abinitio?**

In Abinitio, partition is the process of dividing data sets into multiple sets for further processing. Different types of partition component includes

1. **Partition by Round-Robin:**

* Distributing data evenly, in block size chunks, across the output partitions.
* It distributes the data evenly in block size chunks.

1. **Partition by Range:**

* You can divide data evenly among nodes, based on a set of partitioning ranges and key.
* Based on a set of partitioning ranges and key, it divides data evenly among nodes.

1. **Partition by Percentage:**

* Distribution data, so the output is proportional to fractions of 100.
* It distributes data in such a way that the output is proportional to fractions of 100.

1. **Partition by Load balance:**

* Dynamic load balancing.
* It distributes data based on dynamic load balancing.

1. **Partition by Expression:**

* Data dividing according to a DML expression.
* It divides data based on a DML expression.

1. **Partition by Key:**

* Data grouping by a key.
* It partitions data based on a key.

**Q. Explain what is SANDBOX?**

A SANDBOX is referred for the collection of graphs and related files that are saved in a single directory tree and behaves as a group for the purposes of navigation, version control, and migration.

A sandbox is like a directory that contains a collection of Ab Initio graphs and related files. It is local to any user and is a replica of the Project in the EME. It will be helpful for version control, migration, and navigation.

A SANDBOX is referred for the compilation of graphs and connected files that are saved in a particular directory tree and behaves as a set for version control, navigation, and relocation.

**Q. Explain what is de-partition in Abinitio?**

De-partition is done to read data from multiple flow or operations and is used to re-join data records from different flows. There are several de-partition components available which include Gather, Merge, Interleave, and Concatenation.

**Q. List out some of the air commands used in Abintio?**

Air command used in Abinitio includes

**air object Is<EME path for the object-/Projects/edf/..> :** It is used to see the listings of objects in a directory inside the project

**air object rm<EME path for the object-/Projects/edf/..> :** It is used to remove an object from the repository

**air object versions-verbose<EME path for the object-/Projects/edf/..> :** It gives the version history of the object.

Other air command for Abinitio include air object cat, air object modify, air lock show user, etc.

**Q. What is Rollup Component?**

Roll-up component enables the users to group the records on certain field values. It is a multiple stage function and consists initialize 2 and Rollup 3.

A rollup component is used to group the records based on certain field values. It is a multi-stage transform function which contains functions like initialize, rollup and finalize.

**Q. Mention what is the syntax for m\_dump in Abinitio?**

The syntax for m\_dump in Abinitio is used to view the data in multifile from unix prompt. The command for m\_dump includes

**m\_dump a.dml a.dat:** This command will print the data as it manifested from GDE when we view data in formatted text

**m\_dump a.dml a.dat>b.dat:** The output is re-directed in b.dat and will act as a serial file.b.dat that can be referred when it is required.

**Q. What do you mean by the overflow errors?**

Overflow errors are the errors raised when dealing with processing bulky sets of data. While processing data, bulky calculations might not fit the memory allocated for them. And when a character of more than 8-bits is stored, an overflow error is raised.

While processing data, bulky calculations are often there and it is not always necessary that they fit the memory allocated for them. In case a character of more than 8-bits is stored there, these errors result simply.

#### **Q. How to connect EME to Ab Initio Server?**

EME can be connected to the Ab Initio server in several ways. The following are the ways to connect to EME.

* Set AB\_AIR\_ROOT
* Air commands
* EME web interface at [http://serverhost:[serverport]/abinitio](http://serverhost[serverport]/)
* Connect to EME data store through GDE

**Q. What are the kinds of layouts that Ab Initio supports?**

A layout defines which component should run where. Ab Initio has two kinds of layouts.

**Serial layout -** the level of parallelism is 1.

**Parallel layout -** the level of parallelism depends on the data partition.

**Q. Mention the kind of layouts does Abinitio supports?**

There are successive and parallel layouts supported by Abinitio. A graph can have both at the same time. The parallel one depends on the amount of data parallelism. If the multi-file system is 4-way parallel, then a part in a graph can run four-way parallel if the layout is distinct such as it’s same as the degree of parallelism.

**Q. How to add default rules in the transformer?**

Go to component properties, navigate to the parameter tab page, and double click on the transform parameter.

The transform editor page will open. Click on the edit menu and select the 'Add Default Rules' option from the drop-down. You can choose from Match names and Wildcard options.

Double click on the change parameter in the parameter tab page in constituent properties  
Click on Edit menu in change editor.  
Select Add Default Rules from the decline list box.  
It shows Match Names and Wildcard options. Select either of them.

**Q. What is a surrogate key?**

The system generated unique sequential number is called a surrogate key. It acts as a primary key.

**Q. What is the relation between limit and ramp?**

Limit and ramp are used to set the reject tolerance of a graph. Limit is the number of rejects and ramp is rate of rejection.

The formula for rejects tolerance is, **limit + (ramp\*no\_of\_records\_processed)**

**Q. What is a ramp limit?**

* A limit is an integer parameter which represents several reject events
* Ramp parameter contains a real number representing a rate of reject events of certain processed records.
* The formula is – No. of bad records allowed = limit + no. of records x ramp.
* A ramp is a percentage value from 0 to 1.
* These two provides the threshold value of bad records.

#### **Q. How will you handle it if DML is changing dynamically?**

We have a lot of ways through which we can handle dynamically changing DML. Some of the methods are,

* Use a conditional DM
* Call the vector functionality while calling the DMLs
* Use the MULTI REFORMAT component

**Q. What are the air commands used in Ab Initio?**

* Air object ls
* Air object rm
* Air project modify
* Air object versions -verbose
* Air lock show -user
* Air sandbox status

#### **Q. What are the types of output formats that we can get after processing data?**

* Charts
* Tables
* Vectors
* Plain Text files
* Maps
* Raw files
* Image files

**Q. How data is processed and what are the fundamentals of this approach?**

There are certain activities which require the collection of the data and the best thing is processing largely depends on the same in many cases. The fact is data needs to be stored and analyzed before it is processed. This task depends on some major factors are they are

1. Collection of Data

2. Presentation

3. Final Outcomes

4. Analysis

5. Sorting

These are also regarded as the fundamentals that can be trusted to keep up the pace in this matter.

**Q. What are the factors on which storage of data depends?**

It depends on the sorting and filtering. In addition to this, it largely depends on the software one uses.

**Q. What do you mean by data sorting?**

It is not always necessary that data remains in a well-defined sequence. It is always a random collection of objects. Sorting is nothing but arranging the data items in desired sets or sequence.

**Q. When running a stored procedure definition script how would you guarantee the definition could be rolled back in the event of problems?**

There are quite a few factors that determine the approach such as what type of version control is used, what is the size of the change, what is the impact of the change, is it a new procedure or replacing an existing and so on.

If it is new, then just drop the wrong one.

if it is a replacement then how big is the change and what will be the possible impact, depending upon you can have the entire database backed up or just create a script for your original procedure before messing it up or you just do ed and change the file back to original and reapply. you may rename the old procedure as old and then work on new and so on.

few issues to keep in mind are synonyms, dependencies, grants, any job calling the procedure at the time of change and so on. In nutshell, the scenario can be varied and the solution also can be varied.

**Q. Have you used the rollup component? Describe how?**

If the user wants to group, the records on particular field values then rollup is the best way to do that. Rollup is a multi-stage transform function and it contains the following mandatory functions.

1. initialize

2. rollup

3. finalize

Also, need to declare one temporary variable if you want to get counts of a particular group.

For each of the group, first, it does call the initialize function once, followed by rollup function calls for each of the records in the group and finally calls the finalize function once at the end of last rollup call.

**Q. What is AB\_LOCAL expression where do you use it in ab-initio?**

ablocal\_expr is a parameter of the table component of Ab Initio.LOCAL() is replaced by the contents of ablocal\_expr. Which we can make use in parallel unloads. There are two forms of AB\_LOCAL() construct, one with no arguments and one with a single argument as a table name(driving table).

The use of AB\_LOCAL() construct is in Some complex SQL statements contain grammar that is not recognized by the Ab Initio parser when unloading in parallel. You can use the LOCAL() construct in this case to prevent the Input Table component from parsing the SQL (it will get passed through to the database). It also specifies which table to use for the parallel clause.

**Q. How to get DML using Utilities in UNIX?**

If your source is a COBOL copybook, then we have a command in Unix which generates the required in Ab Initio. here it is,

**Q. Define ramp limit in AB Initio?**

Generally, the ramp is referred to the percentage value ranging from 0 to 1. To represent the number of reject events, the limit parameter possesses an integer called a ramp limit. The reject events can be calculated by the following formula,

#### **Q. What are is\_valid and is\_define used for?**

* **is\_valid -** It is used to test if a value is valid or not. If the expression is a valid data item, the value will be 1. If the expression is not a valid data item, the value will be 0.
* **is\_define -** It is used to test if an expression is not NULL. If the expression is non NULL, the value will be 1. The value will be 0 otherwise.

**Q. State the first\_defined function with an example?**  
This function is similar to the function NVL() in Oracle database  
It performs the first values which are not null among other values available in the function and assigns to the variable  
**Example:** A set of variables, say v1,v2,v3,v4,v5,v6 are assigned with NULL.  
Another variable num is assigned with value 340 (num=340)  
num = first\_defined(NULL, v1,v2,v3,v4,v5,v6,NUM)  
The result of num is 340.

#### **Q. How does the force\_error function work?**

If any mentioned conditions are not met, the force\_error it forces an error. It will be useful when you want to stop the execution of a graph if it doesn't meet the set condition.

It will send the records to the reject port and error message to the error port.

#### **Q. How to improve the performance of a graph?**

We can improve the performance of a graph through the following methods.

* Use lookup instead of join and merge components.
* When we have to join two files and don't want duplicates, use a union function instead of a duplicate remover.
* Minimize the use of sort components.
* Reduce the use of regular expression functions in the transfer functions.
* Don’t use broadcast as partitioner for large datasets.
* Use only the fields that are required in sort, reformat, and join components.

#### **Q. What is a multifile system?** Multifile is a set of directories on different nodes in a cluster. They possess an identical directory structure. The multifile system leads to better performance as it is parallel processing where the data resides on multiple disks.

It is created with the control partition on one node and data partitions on the other nodes to distribute the processing in order to improve the performance.

#### **Q. What kind of layouts does Ab Initio support?**

* Supports serial and parallel layouts.
* A graph layout supports both serial and parallel layouts at a time.
* A multi-file system is a 4-way parallel system.
* A component in a graph system can run 4-way parallel system.

#### **Q. How data is processed, and what are the fundamentals of this approach?** Certain activities require the collection of the data, and the best thing is processing largely depends on the same in many cases. Before processing the data, it has to reside on some placeholder like well-defined storage. This task depends on some major factors are they are

* Collection of Data
* Presentation
* Final Outcomes
* Analysis
* Sorting

**Q. What is the difference between partitioning with key and round-robin?**

**Partition by key:**

In this, we have to specify the key based on which the partition will occur. It results in well-balanced data due to the key-based partitions. It is useful for key- dependent, parallelism.

**Partition by round-robin:**

In this, distributing data evenly in block size chunks, the records are partitioned in a sequential way across the output partition. It is not key-based, and results are well-balanced data, especially with a block size of 1. It is useful for record independent parallelism.

**Q. How do you improve the performance of a graph?**

* Reduce the usage of multiple components on certain phases.
* Use a refined and well-defined value of max core values for sort and join components.
* Minimize the use of regular expression functions like re\_index in the transfer functions
* Minimize sorted join component and, if possible, replace them with in-memory join/hash join.
* Use only required fields in the sort, reformat, join components.
* Using Phase or the flow buffering during the cases of merge or sorted joins.
* Use hash join if the two sets of input are small; else better to choose the sorted join for the huge input size.
* For large dataset better not use broadcast as partitioned
* Reduce the number of sort components while processing.
* Avoid repartitioning of data unnecessarily.

**Q. What is data encoding?**

Data needs to be kept classified in many cases, and it can be complete through this approach. It ensures information scraps in a form which no one else than the sender and the receiver can appreciate.  
Data needs to be kept confidential in many cases and it can be done through this approach. It simply makes sure of information remains in a form which no one else than the sender and the receiver can understand.

**Q. Explain data sorting?**

It is not always essential that data remains in a well-defined series. It is always a chance compilation of objects. Sorting is nil but arranging the data items in preferred sets or progression.

**Q. Define the evaluation of parameters order?**

* The host setup script will be implemented first.
* All ordinary parameters, that is, built-in, are evaluated.
* All Sandbox parameters are implemented.
* The plan script project-start.ksh is executed.
* All figure parameters are evaluated.
* Graph parameters are evaluated
* The create Script of the graph is executed

**Q. Define a local lookup?**

Local lookup file has documentations which can be located in major memory, and they are also used to change function for retrieving records much earlier than retrieving from the disk.

**Q. What do you mean by primary keys and foreign keys?**

In RDBMS, the association between the two tables is represented as Primary key and foreign key relationship. Whereas the primary key table is the close relative table and the foreign key table is the infant table.

**Q. Explain an outer join?**

An outer join is utilized when one wants to choose all the records from a port whether it has fulfilled the join criteria or not.

**Q. Define local and formal parameter?**

Both are graph level parameters, but in local you need to initialize the worth at the time of announcement whereas globe no need to initialize the data it will produce at the time of operation the graph for that parameter.

**Q. How Abinitio EME is segregated?**

It is logically divided into two segments:

* User Interface
* Data Integration Portion

**Q. State the difference between check-point and phase?**

The difference between them is:

**CHECKPOINT:**

- The break of the procedure will be sustained following the checkpoint

- Data from the checkpoint is gathered and carry on executing after correction.

- When a graph fails in the centre of the procedure, a recovery point is shaped known as a Checkpoint

**PHASE:**

- All the phases will sprint one by one

- The middle file will be deleted

- If a graph is shaped with phases, each phase is assigned to some division of memory one after another.

**Q. Name the operations that support avoiding duplicate record?**

* Performing aggregation
* Utilizing the Rollup part
* Using Dedup class

**Q. What do you mean by ramp limit?**

An edge is an integer parameter which represents a numeral of reject events. Ramp parameter encloses a real number on behalf of a rate of decline events of certain processed records; it has a percentage value from 0 to 1.

**Q. What is the use of aggregation when we have rollup as we know rollup component in abinitio is used to summarize a group of data record? Then where we will use aggregation?**  
Aggregation and Rollup both can summarize the data but rollup is much more convenient to use. To understand how a particular summarization being rollup is much more explanatory compared to aggregate. Rollup can do some other functionality like input and output filtering of records. Aggregate and rollup perform the same action, rollup display intermediate result in main memory, Aggregate does not support intermediate result.

**Q. Describe the elements you would review to ensure multiple scheduled batch jobs do not collide with each other?**  
Because every job depends upon another job for example if your first job result is successfully then another job **will execute otherwise your job doesn’t work.**

**Q. How to create a repository in abinitio for stand-alone system(LOCAL NT)?**

If you are trying to install the Ab -Initio on a stand-alone machine, then it is not necessary to create the repository, While installing It creates automatically for you under abinitio folder ( where you installing the Ab-Initio) If you are still not clear please ask your Question on the same portal.

**Q. Describe the process steps you would perform when defragmenting a data table. Does this table contain mission-critical data?**

**1)** We can move the table in the same or other tablespace and rebuild all the indexes on the table. alter table move this activity reclaims the defragmented space in the table analyze table table\_name compute statistics to capture the updated statistics.

**2)**Reorg could be done by taking a dump of the table, truncate the table and import the dump back into the table.

**Q. Why might you create a stored procedure with the with recompile option?**

Recompile is useful when the tables referenced by the stored proc undergoes a lot of modification/deletion/addition of data. Due to the heavy modification activity, the execute plan becomes outdated and hence the stored proc performance goes down. If we create the stored proc with recompile option, the sql server won’t cache a plan for this stored proc and it will be recompiled every time it is run.

**Q. Explain PDL with an example?**

To make a graph behave dynamically, PDL is used

Suppose there is a need to have a dynamic field that is to be added to a predefined DML while executing the graph

Then a graph level parameter can be defined

Utilize this parameter while embedding the DML in the output port.

**For Example:** define a parameter named field with a value “string(“ | “) name;”

Use ${mystring} at the time of embedding the DML in out port.

Use $substitution as an interpretation option.

**Q. Describe the Evaluation of Parameters order?**

Following is the order of evaluation:

* The host setup script will be executed first
* All Common parameters, that is, included, are evaluated
* All Sandbox parameters are evaluated
* The project script – project-start.ksh is executed
* All form parameters are evaluated
* Graph parameters are evaluated
* The Start Script of the graph is executed

**Q. Explain the difference between the truncate and delete commands?**  
  
**Truncate:**  
It is a DDL command, used to delete tables or clusters. Since it is a DDL command hence it is auto commit and Rollback can’t be performed. It is faster than delete.

**Delete:**  
It is DML command, generally used to delete a record, clusters or tables. Rollback command can be performed, to retrieve the earlier deleted things. To make deleted things permanently, “commit” command should be used.

**Q. What is BROADCASTING and REPLICATE?**

**Broadcast –** Takes data from multiple inputs, combines it and sends it to all the output ports.  
Eg – You have 2 incoming flows (This can be data parallelism or component parallelism) on Broadcast component, one with 10 records & other with 20 records. Then on all the outgoing flows (it can be any number of flows) will have 10 + 20 = 30 records.

**Replicate –** It replicates the data for a particular partition and send it out to multiple outports of the component, but maintains the partition integrity.

**Eg –** Your incoming flow to replicate has a data parallelism level of 2. with one partition having 10 recs & other one having 20 recs. Now suppose you have 3 output flows from replicate. Then each flow will have 2 data partitions with 10 & 20 records respectively.

**Q. What are the benefits of data processing according to you?**

Well, processing of data derives a very large number of benefits. Users can put separate many factors that matter to them. In addition to this, with the help of this approach, one can easily keep up the pace simply by deriving data into different structures from an unstructured format.

In addition to this, the processing is useful in eliminating various bugs that are often associated with the data and cause problems in a later section. It is because of no other reason than this, data processing has a wide application in several tasks.

**Q. What exactly do you understand with the term data processing and businesses can trust this approach?**

Processing is a procedure that simply covert the data from a useless form into a useful one without making a lot of efforts. However, the same may vary depending on factors such as the size of data and its format.

A sequence of operations is generally carried out to perform this task and depending on the type of data, this sequence could be automatic or manual. Because in the present scenario, most of the devices that perform this task are PC’s automatic approach is more popular than ever before. Users are free to obtain data in forms such as a table, vectors, images, graphs, charts and so on. This is the best things that business owners can simply enjoy.

**Q. What are the primary keys and foreign keys?**

In RDBMS the relationship between the two tables is represented as Primary key and foreign key relationship. Whereas the primary key table is the parent table and the foreign key table is the child table. The criteria for both the tables are there should be a matching column. company

**Q. When using multiple DML statements to perform a single unit of work, is it preferable to use implicit or explicit transactions, and why?**

Because implicit is using for internal processing and explicit is using for using open data required.

**Q. Describe how you would ensure that database object definition (Tables, Indices, Constraints, Triggers, Users, Logins, Connection Options, and Server Options, etc) are consistent and repeatable between multiple database instances (i.e.: a test and production copy of a database)?**

Take an entire database backup and restore it in a different instance.

Take statistics of all valid and invalid objects and match.

Periodically refresh.

**Q. What is an outer join?**

An outer join is used when one wants to select all the records from a port – whether it has satisfied the join criteria or not.

**Q. What is local and formal parameter?**

Two are graph level parameters but in local you need to initialize the value at the time of declaration whereas globe no need to initialize the data it will print at the time of running the graph for that parameter.

**Q. What is the MAX CORE of a component?**  
MAX CORE is the space consumed by a component that is used for calculations  
Each component has different MAX COREs  
Component performances will be influenced by the MAX CORE’s contribution  
The process may slow down / fasten if a wrong MAX CORE is set.