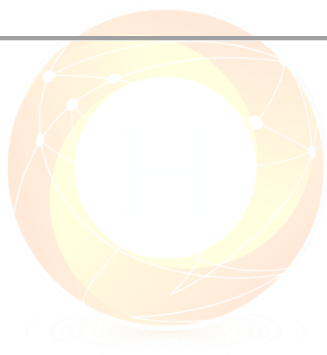


oHo Training

AB Initio Tutorial

Version 1.3



OHO TRAINING
training at your desk

Disclaimer: All data and information provided on this tutorial is for informational purposes only. Tutorials are written by our staffs and any similarities to the content or tutorials are strictly coincidental. The contents contained in this website are copyright protected. No person may duplicate, edit, or publish this content in whole or in part.

Table of contents

Preface	4
About AB Initio	4
Scalability	4
Development Time	5
Matadata Management	5
Integration	5
Application of Ab Initio Software	5
Ab Initio Software Architecture	6
Installation	7
A Graphical Method for Building Business Applications	8
Graph Programming	9
About This Tutorial	9
Audience	10
Tutorial Flow	10
Tutorial Layout	11
Tutorial Conventions	11
Addendum	11
Lesson 1 Setting Up	13
Installing the GDE	13
AB INITIO INTRODUCTORY SCREEN	14
Configuring the GDE	15
Building a Graph to Select Customers	16
Connecting Components with Flows	18
Removing Yellow Reminders	19
Running The Application	20
Testing the Results	21
Saving Graph	22

OHO TRAINING
training at your desk

Running the Tutorial Step Graph	22
Lesson 1 Exercise	23
Lesson 1 Summary	23
LESSON 2 Analyzing the Problem	25
The Problem	25
The Scenario	25
The Raw Data	25
Customer Records and Formats	26
Transaction Records and Formats	26
Your Mission	27
The Plan	27
Executing the Plan	27
Steps for Building and Running Graph	28
Lesson 3 Merging Records	30
Building a Graph to Comsolidate Data	30
Describing Data	31
Locating Input Data	31
Describing Input Data	33
Viewing Input Record Formats	34
Viewing Input Data	36
Describing Output Data	37
Assigning Parameter Values	38
Setting the MergeJoin Parameters	39
Transforming Data	39
Running and Testing Abbot's Data	41
LESSON 3 Exercises	43

OHO TRAINING
training at your desk

Preface

This tutorial teaches developers the basics of using Ab Initio software.

Ab Initio is a flexible and powerful suite of software at the heart of data processing systems of some of the largest companies in e-commerce, telecommunications, banking, and other industries. These companies use Ab Initio software for a variety of purposes including data warehousing, batch processing, click-stream analysis and application integration.

This tutorial focuses on the “core” of Ab Initio products, including the Co>Operating® System, core components (including Transformation and Sort packages), and the Graphical Development Environment. It mentions other Ab Initio products and roles that they can play in developing enterprise-wide data processing systems.

- By the time you finish this tutorial, you will know how to:
- Build applications that process data with Ab Initio components
- Describe different kinds of data with Ab Initio record formats
- Specify business rules with Ab Initio transform functions
- Execute Ab Initio applications on parallel and distributed systems

About AB Initio

OHO TRAINING
training at your desk

Ab Initio Software Corporation works From First Principles to deliver solutions that meet the requirements of enterprise level data processing systems. Some software providers deliver solutions that scale in one direction, but not another; inevitably, business requirements will eventually clash with such solutions, stressing systems in ways they were not designed to accommodate. Ab Initio has created solutions that scale with business requirements to operate in the most demanding environments.

Ab Initio software solves the toughest problems today facing developers and architects of enterprise-wide data processing systems: *scalability, development time, metadata management, and integration.*

Scalability

The Ab Initio Co>Operating System provides truly scalable parallel and distributed execution, allowing applications to tackle enormous—and rapidly growing—volumes of data. If you double the size of the data, it runs in the same amount of time; you just double the number of

processors. If you need it run in half the time, again, you just double the number of processors. Applications built with Ab Initio software regularly employ more than a hundred processors to make quick work of processing the world's largest datasets. With the Co>Operating System, virtually any amount of computational power can be brought to bear on a problem.

Development Time

The Ab Initio Graphical Development Environment (GDE) enables truly rapid application development and enhancement. Through the GDE, one may build and modify the most complex applications in a purely graphical way. This leads to very high productivity: in a handful of months, a small team can design, implement, and put into production a large data warehouse system using Ab Initio Software. Smaller systems can be put together in minutes.

Metadata Management

The Ab Initio Repository provides storage for all aspects of a data processing system, from design information to operations data. The Repository provides a rich store for the applications themselves, including data formats and business rules, enabling dependence analysis ("Where did this data come from?") and impact analysis ("If we made this change, what would be affected?").

Integration

Ab Initio software includes general capabilities for application integration. Using these capabilities, one can easily incorporate custom and third-party software packages into the Ab Initio component framework. There, they immediately benefit from the scalability provided by the Co>Operating System, the ease-of-use provided by the Graphical Development Environment, and the metadata management facilities of the Repository.

Application of Ab Initio Software

Ab Initio software can pull data from practically any kind of source (legacy systems, operational systems, database tables, flat files, SAS datasets, the Internet, and so on), apply arbitrarily complex transformations to it, and move that data to practically any kind of destination. This flexibility makes it easy to integrate existing programs written in virtually any language into complete, end-to-end solutions. As you will learn in this tutorial, all of this can be done without programming.

In a traditional data warehousing environment, Ab Initio has a place at both the “front end” of the warehouse, transforming and cleansing source data, and at the “back end” of the warehouse, extracting, transforming, and loading into other systems for analysis as shown in Figure 1:

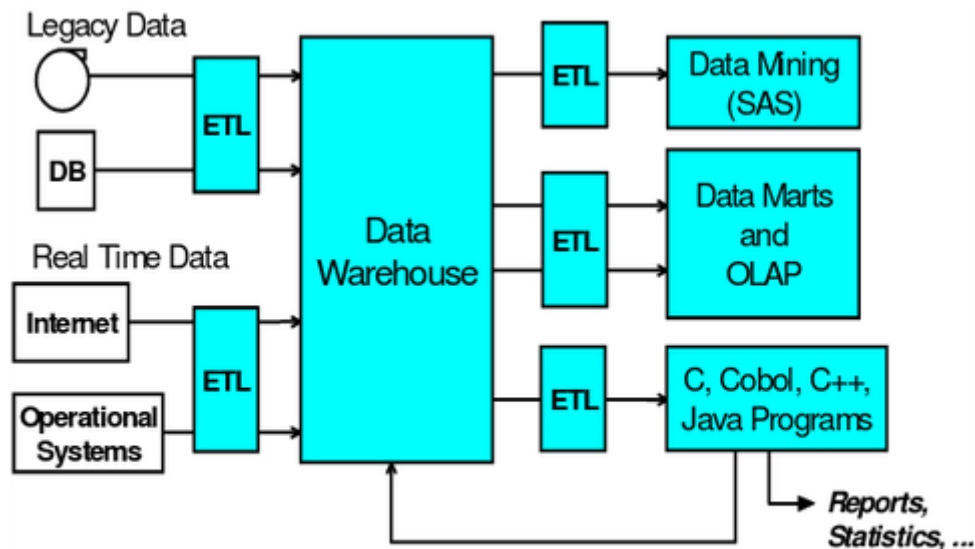


Figure 1 Ab Initio Solutions for the Data Warehouse

Ab Initio software performs many compute-intensive operations like multi-way joins and complex aggregations substantially faster than conventional software systems. Furthermore, Ab Initio's built-in scalability allows it to do all of these things in parallel across as many CPU's and servers as desired.

The flexibility and performance of Ab Initio software make it the ultimate in integration and scalability.

Ab Initio Software Architecture

Ab Initio software helps you build large-scale data processing applications and run them in parallel or distributed computing environments.

You build these applications using an application development environment to connect *components*. The Co>Operating System executes applications on Unix or NT servers by running the components in parallel.

Components are reusable software modules that are connected together to create applications. Components can come from Ab Initio's component library, or can be created from existing

custom programs, and 3rd party products. Existing programs written in any language can be made into components without any changes.

The high-level architecture of Ab Initio software is shown below:

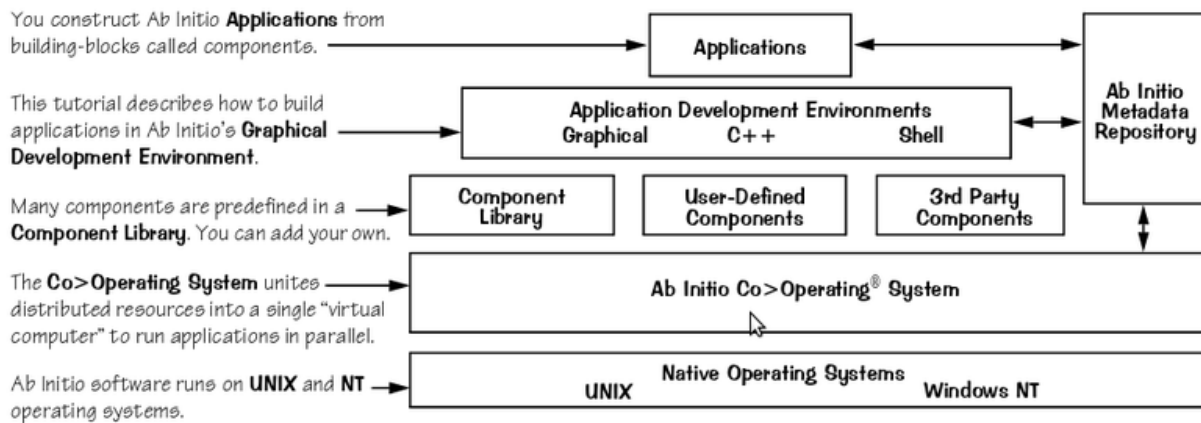


Figure 2 Ab Initio High Level Architecture

Installation

Ab Initio base software consists of two main pieces:

- Ab Initio Co>Operating System and core components
- Graphical Development Environment (GDE)

Your system administrator installs the Co>Operating System and core components on the server where your applications run. For parallel applications, your administrator installs the Co>Operating System on several servers; one server, the *control server*, is used as a central point of control. You install the Graphical Development Environment *client* on your desktop PC and configure it to communicate with the server. On Windows NT or Windows 2000, your desktop machine can be both server and client. Figure 3 shows three possible configurations.

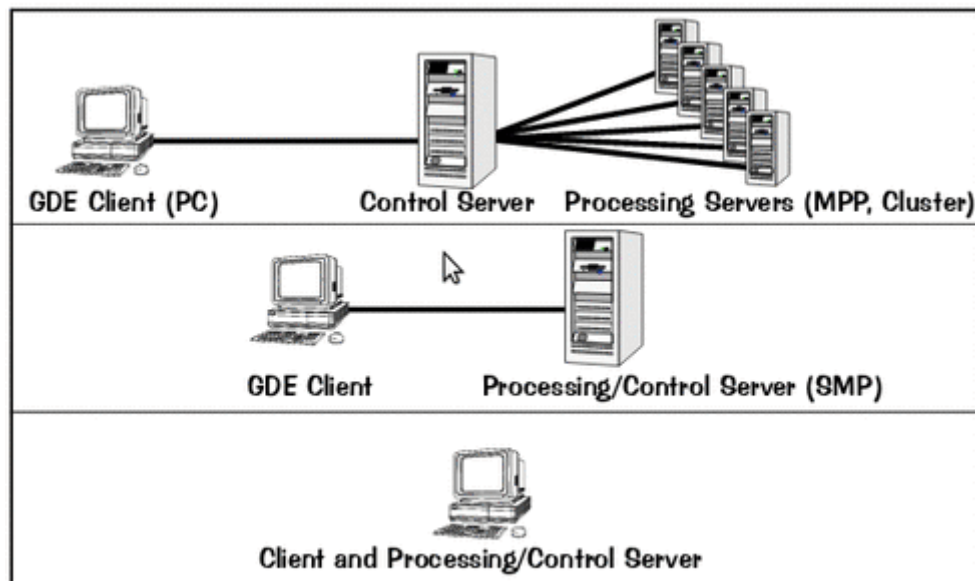


Figure 3 Three Possible System Configurations

A Graphical Method for Building Business Applications

A graph is a picture that represents the various processing stages of a task and the streams of data as they move from one stage to another.

If one picture is worth a thousand words, is one graph worth a thousand lines of code? Ab Initio application graphs often represent in a diagram or two what might have taken hundreds to thousands of lines of code. This can dramatically reduce the time it takes to develop, test, and maintain applications.

Working with the premise that application developers work visually,

Ab Initio based the Graphical Development Environment on the data flow model (see sidebar). This paradigm works because:

- Data flow diagrams allow you to think in terms of meaningful processing steps, not microscopic lines of code.
- Data flow diagrams capture the movement of information through the application—a flow that disappears from view when the application is turned into lines of computer code.

Ab Initio calls this development method *graph programming*.

Graph Programming

Working with the GDE on your desktop is easier than drawing a data flow diagram on a white board. You simply drag and drop functional modules called *components* and link them with a swipe of the mouse. When it's time to run the application, the Ab Initio Co>Operating System turns the diagram into a collection of processes running on servers.

The Ab Initio term for a running data flow diagram is a *graph*. The inputs and outputs are *dataset components*; the processing steps are *program components*; and the data conduits are *flows*. (See sidebar.)

Figure 4 shows a sample Ab Initio application that reads customer records, generates new records with a different format, and selects those that qualify as “good” customers using a selection criterion.

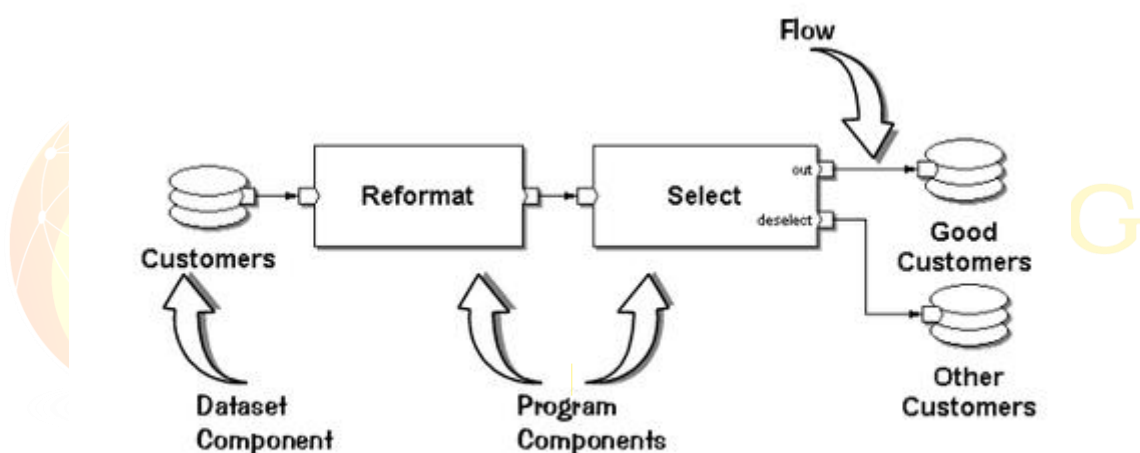


Figure 4 Sample Application

About This Tutorial

This tutorial poses a big data business problem, then helps you solve it step-by-step using Ab Initio's Graphical Development Environment (GDE). The lessons walk you through specific consecutive tasks for learning the GDE and solving the business problem. The exercises in Lessons 3, 4, and 5 not only reinforce the lesson, but provide portions of the solution.

Attention You *must* complete the exercises in Lessons 3, 4, and 5 to solve the problem. Answers are listed in Appendix A.

We recommend you read this tutorial at your computer, so you can drag and drop Ab Initio components, hook them together, and run programs. The tutorial data files, program files, and other materials are provided on-line. Lesson 1 tells you how to access those files.

For ease of use, the sample data used for these exercises is in ordinary serial files rather than parallel files or database tables. We'll discuss parallel applications in Lesson 7, and database tables in lesson 8.

Audience

Application developers and analysts who want to learn how to use Ab Initio software are the audience for this tutorial. We assume familiarity with Windows terms like double-click and pop-up menus. If you know a programming language, this tutorial should be a breeze.

Tutorial Flow

Table 1 describes the lessons in this tutorial. This tutorial also includes two appendices. Appendix A (page 141) supplies answers to exercises at the end of each lesson. Appendix B (page 147) summarizes Ab Initio's Data Manipulation Language.

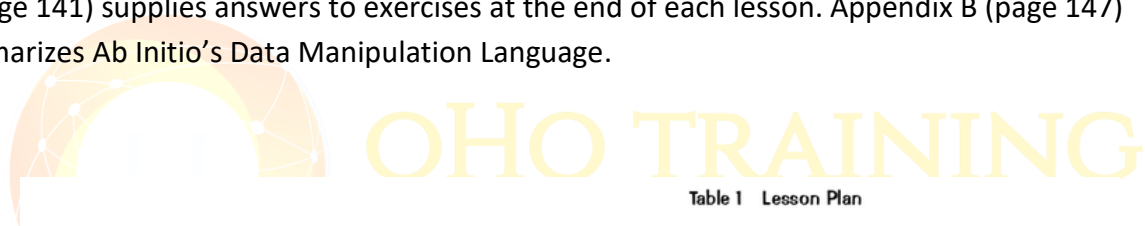


Table 1 Lesson Plan

Read lesson...	To learn...	So you can...
1 Setting Up	Graph-building basics	Build your first graph.
2 Analyzing the Problem	The task at hand, problem-solving ideas, and graph building instructions	Evaluate and subdivide the <i>big data</i> problem, then review graph building steps.
3 Merging Records	How to build a graph that joins data from two different sources	Piece together a small graph that sorts and merges data, and learn about record formats and transforms.
4 Aggregating Records	How to build a graph that sums records	Construct a graph that sorts and aggregates data.
5 Reformatting Data	How to transform and merge data from three different sources	Write record formats and transforms, then build a graph that transforms your data.
6 Putting It All Together	How to hook up the smaller graphs to form one large graph	Create a large graph to solve the big data problem.
7 Making It Parallel	How to make your application run faster through scalable parallel computing	Produce graphs that partition and repartition data, distribute files, and yield scalable performance.
8 Working With Databases	How to load and extract data from databases into and out of Ab Initio software	Use Ab Initio in concert with pre-existing databases to speed up your applications.
9 Improving Performance	Other ways (beyond parallelism) to improve graph performance	Add lookup tables, multistage transforms, and hash components to an application.

Tutorial Layout

This tutorial is in two-column format.

The main, right-hand column, concentrates on solving a specific problem step by step. The left-hand column is a sidebar with supplementary information, important tips, and helpful hints.

Tutorial Conventions

Table 2 Text Conventions

When you see...	This text convention denotes...
Screen Literal type	Programmer's code.
Boldface type	File names, commands, menu items, and user input.
<i>Italic</i> type	Titles of documents and emphasis for important terms in the main text.
<i>Sans Serif</i> type	Supplementary information in the sidebars, as distinguished from the primary information in the main text.
Sans Serif Bold type	Important terms in the sidebars.
<angle brackets>	Indicates substitution; replace the text inside the brackets.
Run >> Settings	Shorthand for "pull down the Run Menu and choose Settings."
Ctrl + Shift + F	Shorthand for "hold down the Ctrl and Shift keys while pressing F."

Addendum

This tutorial is written to accompany GDE version 1.5. Many component names have changed in GDE version 1.6; the GDE on-line help contains details about the changes.

To access the components mentioned in the tutorial:

1 Install the GDE and run it from the start menu. (See Lesson 1.)



2 Click to open the Component Organizer.

3 From the Organizer, choose **Organizer>>Add Top-level Folder**. A folder browser appears.

4 Browse to the install folder (usually **C:\Program Files\Ab Initio\Ab Initio GDE**) and then choose the **Tutorial Components** folder (Figure 5). Use this new folder for the rest of the tutorial.

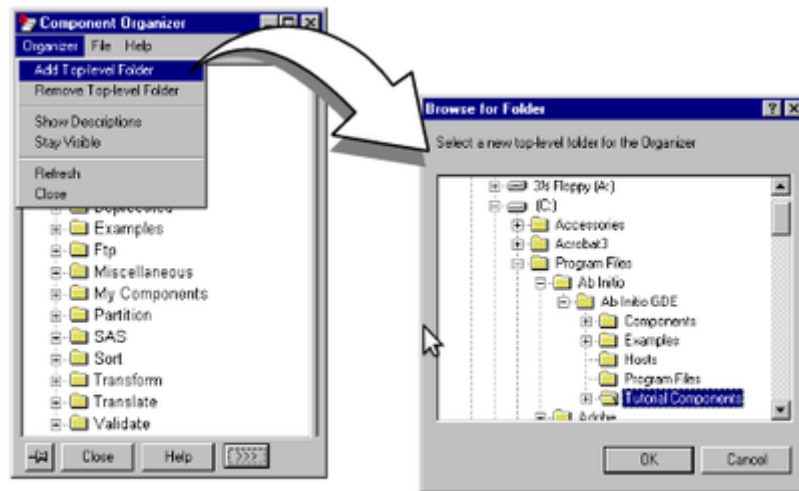
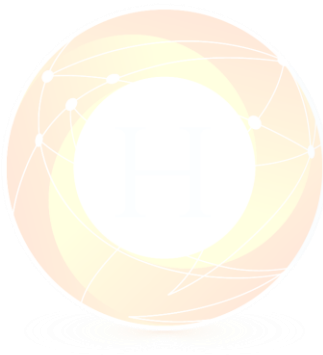


Figure 5 Finding Tutorial Components in GDE Version 1.6 or higher

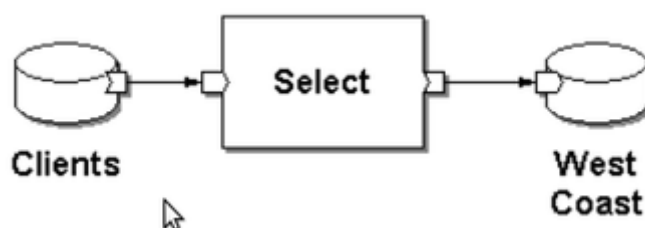


OHO TRAINING
training at your desk

Lesson 1 Setting Up

Lesson 1 explains how to set up Ab Initio's Graphical Development Environment, organize files, create a simple graphical application, and run that application. If you've completed the GDE on-line tutorial, skip to page 14, run the setup graph, then proceed to Lesson 2.

By the end of Lesson 1, you'll build and run a graph that selects customers based on their zip codes.



Installing the GDE

To install the GDE, follow these steps:

- Pick your installation media. Make sure you are using version 1.5 or greater of the GDE.

From diskette: Insert Ab Initio's Disk 1 in your computer disk drive.

•

Select **Run** from the start menu and type: [A:\setup](#)

From CD: Insert Ab Initio's CD in your computer CD drive.

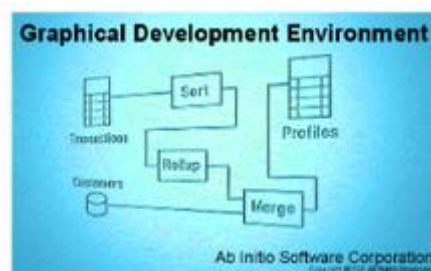
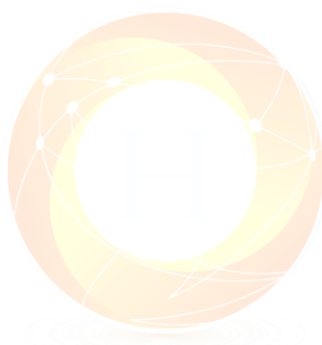
The setup program automatically starts. Follow the instructions from the installation wizard or run **setup.exe** from your CD.

- Click the **Start** menu, point to **Programs**, then select **Ab Initio**. See Figure 6.



Figure 6 Starting the Ab Initio Graphical Development Environment

AB INITIO INTRODUCTORY SCREEN



The introductory screen appears briefly (see sidebar) followed by the online tutorial. For now, dismiss the on-line tutorial. You can reach it later from the **Help** menu.

Like many Windows-based applications, the GDE starts with a new blank workspace (Figure 7). The workspace is named *Graph 1*. Right now, your workspace is gray, but you can change the *background color* (as we have) by pulling down **View >> Options** then clicking **Advanced**.

Underneath the graph name are pull-down menus and buttons. The buttons are grouped into toolbars. The Main, Tool, and Run toolbars are visible. To keep the first few lessons simple, we hide the Phase and Edit toolbars. We'll discuss these toolbars later.

Tip For a brief description of each button, click then a GDE button.

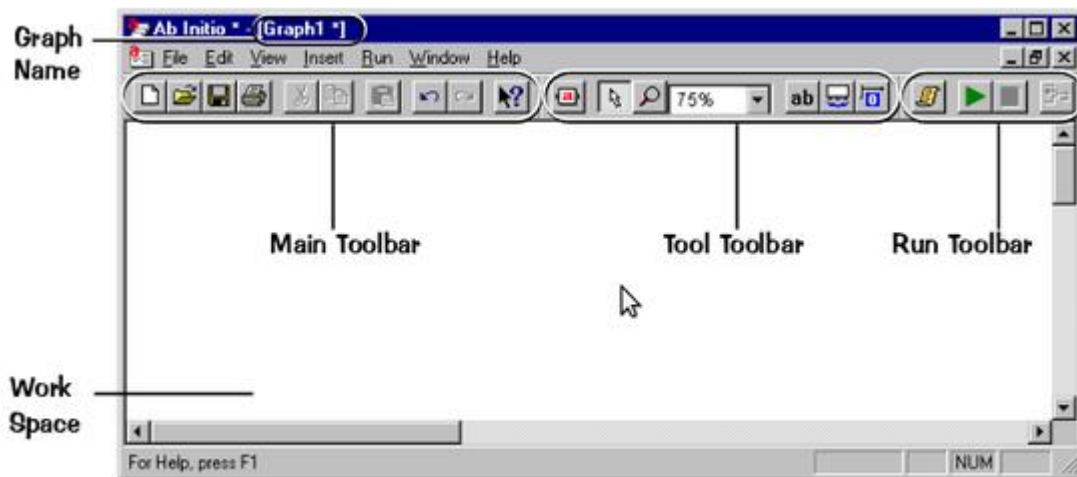


Figure 7 Ab Initio GDE Workspace

Configuring the GDE

Before using Ab Initio software, you must establish a connection between the GDE on your computer and the Ab Initio Co>Operating System on the control node. This connection permits file transfer via the file transfer protocol (FTP) and login access via either rexec or telnet. Follow these steps to configure the GDE:

- Pull down the **Run** menu and choose **Settings** from the main GDE window.

The Run Settings Dialog appears. (Figure 8)

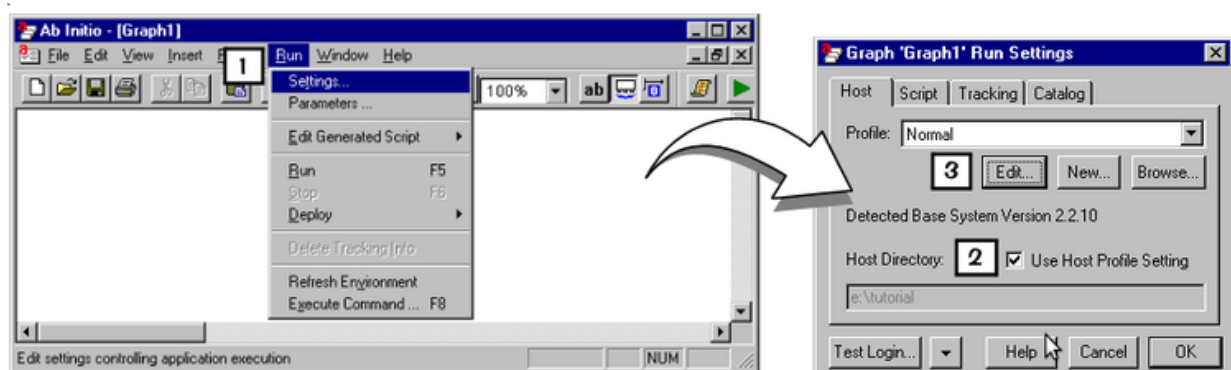


Figure 8 Configuring GDE Run Settings

- Check the **Use Host Profile Setting** checkbox if it is not already checked.
- Click the **Edit** button to open the Host Profile dialog.
- If you are running Ab Initio on your local NT system, check the **Local Execution (NT)** checkbox and go to step 6.

- If you are running Ab Initio on a remote UNIX system, fill in the yellow fields (path to the **Host** and host **Login**), then add a **Password**. If you don't add one now, you'll be prompted later.
- If you are running Ab Initio on a remote UNIX system, type **tutorial** as your **Host Directory**. If you are running Ab Initio on your local NT system, type the full path, for example: **C:\tutorial**
- Select a **Shell** type from the pull-down menu.
- Click **Test Login** to test the connection. You'll be prompted to create the directory. Click **Yes**. If the run settings are verified, click **OK** on each dialog to close it.

If Test Login fails, the error message should explain the problem. For example, if the Ab Initio Co>Operating System is installed in a non-standard location on the control node, enter the correct location.

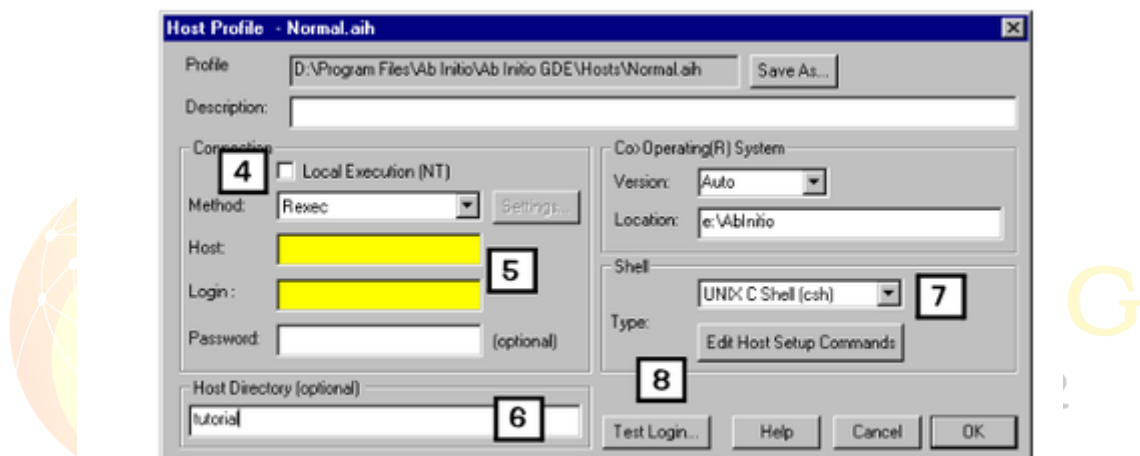


Figure 9 Host Profile Dialog

Building a Graph to Select Customers

Use the GDE to construct a simple application. This application selects clients by zip code (United States postal code). You'll take a list of client records and choose those who live on or near the west coast (zip code greater than or equal to 90000).



- Click to open the Component Organizer, shown in Figure 10.
- Double-click the **Examples** and **Transform** folders to display the contents.

Under the Examples folder, you'll find a **Tutorial** folder.

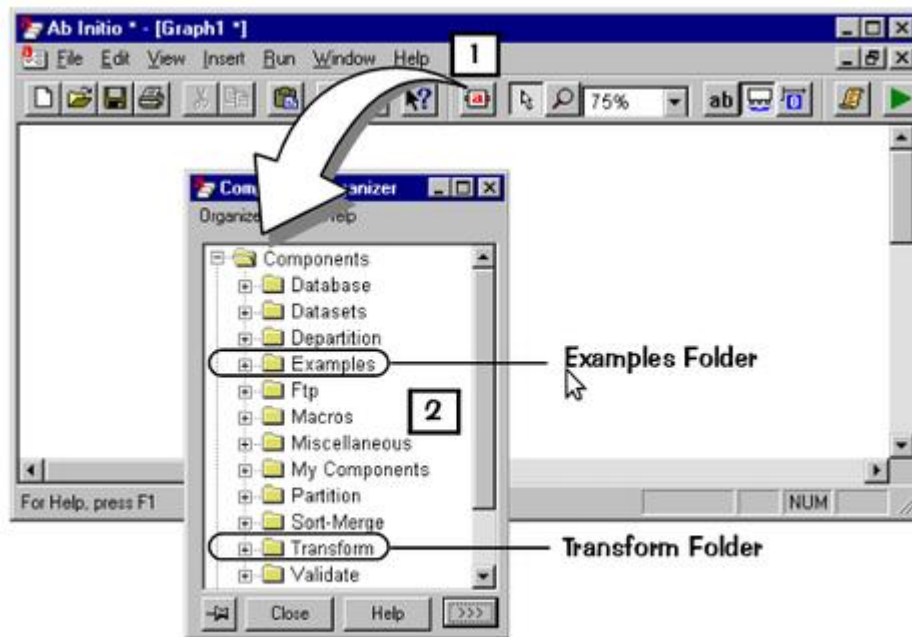


Figure 10 Component Organizer

- Expand the Tutorial folder, then drag and drop the **Clients** and **West Coast** dataset components onto your workspace.
- From the Transform folder, drag and drop the **Select** component onto your workspace. Right now, your **Select** component looks different from ours. For simplicity, pull down **View >> Options**, uncheck **Optional Ports**, then click **OK**. The components should now look alike.
- Click, drag, and arrange the three components as in Figure 11.

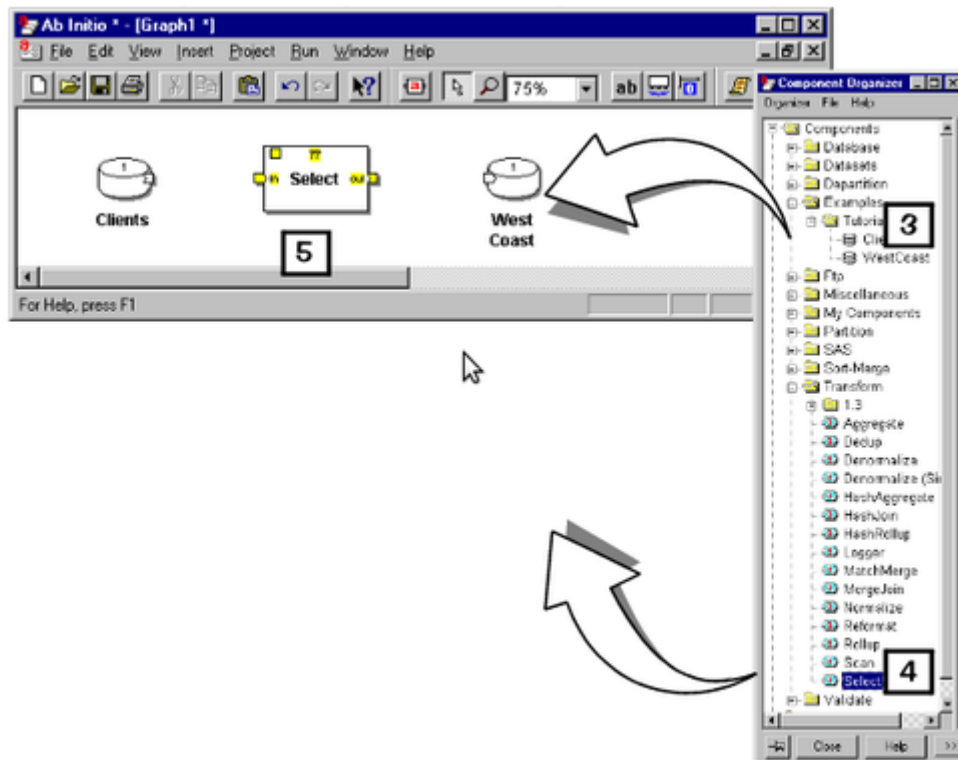


Figure 11 Selecting Components

Connecting Components with Flows

- Click the out port of the Clients dataset and drag the mouse until the cursor reaches the in port of the Select component, then release. You know when you are over a port if it turns blue. An arrow now appears between the two icons. This arrow, called a flow, indicates data streams out of the Clients component and into the Select component.
- Similarly, connect the out port of the Select component to the in port of the West Coast dataset. Your graph should look like Figure 12. (Don't worry if your flows are bent.)

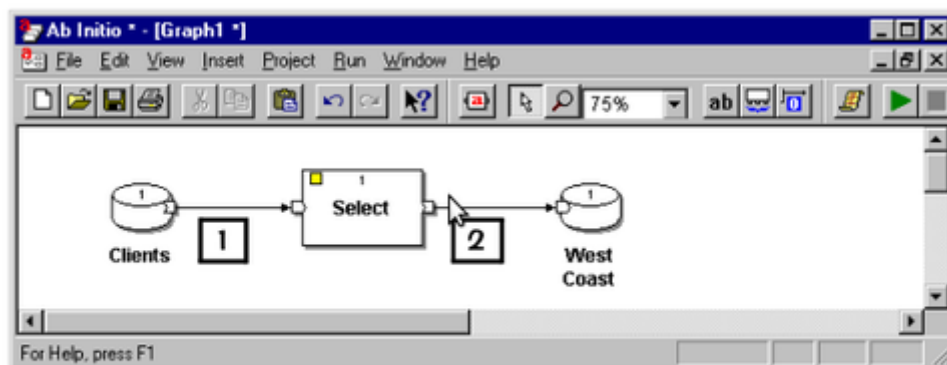


Figure 12 Connecting Components in the Select Application

Removing Yellow Reminders

The graph displays yellow highlighted reminders called *To Do Cues* when it needs more information. In this case, a yellow reminder appears on the **Select** component.

- Double-click the yellow box on the **Select** component to open the **Parameters** page of the Properties dialog (Figure 13). You're being prompted for an expression. Which records do you want to select?
- Note that **select_expr** is highlighted. Click **Edit...** to open the Expression Editor (Figure 14) for this expression.

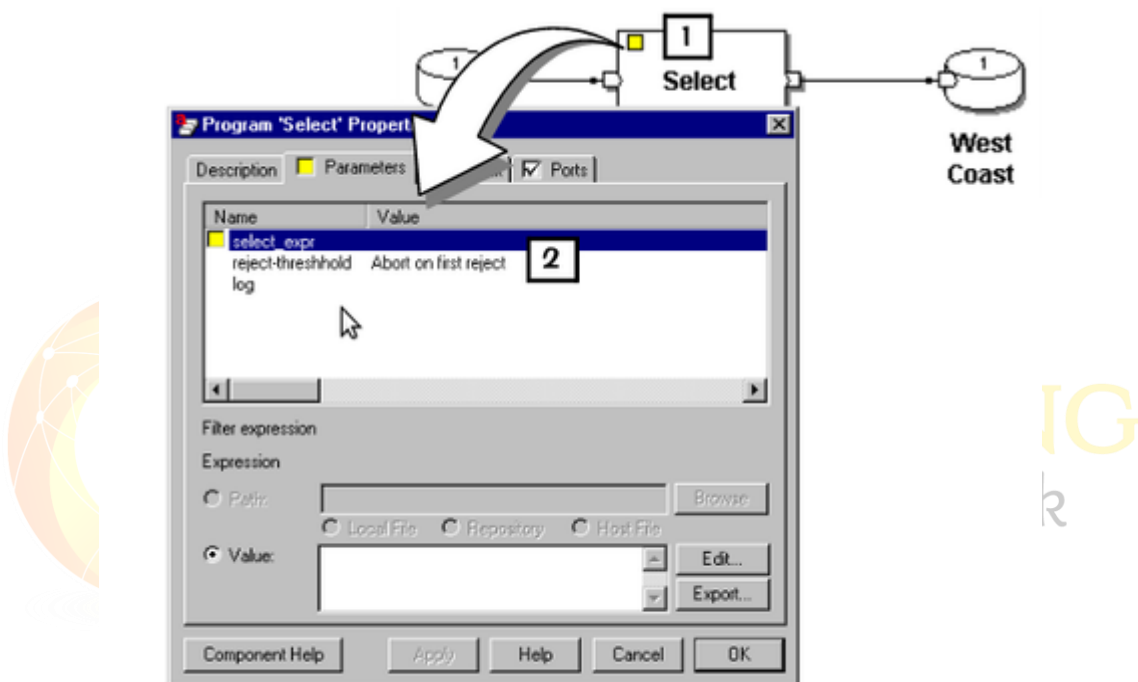


Figure 13 Parameters Page of the Properties Dialog

- click the **zip** field or drag it into the Expression Window. You'll see the word **zip** appear in the window (Figure 14).
- Click the plus sign beside **Comparison Operators** to expand the list, then double-click **x>=y**. You'll see the operator **>=** appear beside **zip**.
- Type **90000** after **>=** as shown in Figure 14.

- The Select component uses the expression **zip >= 90000** to choose all customer records with zip codes greater than or equal to 90000 (or those people who live on the west coast of the United States).
- Click **OK** to dismiss the Expression Editor. Your yellow reminder is gone. Click **OK** again to dismiss [the Properties](#) dialog. Your graph is ready to run.

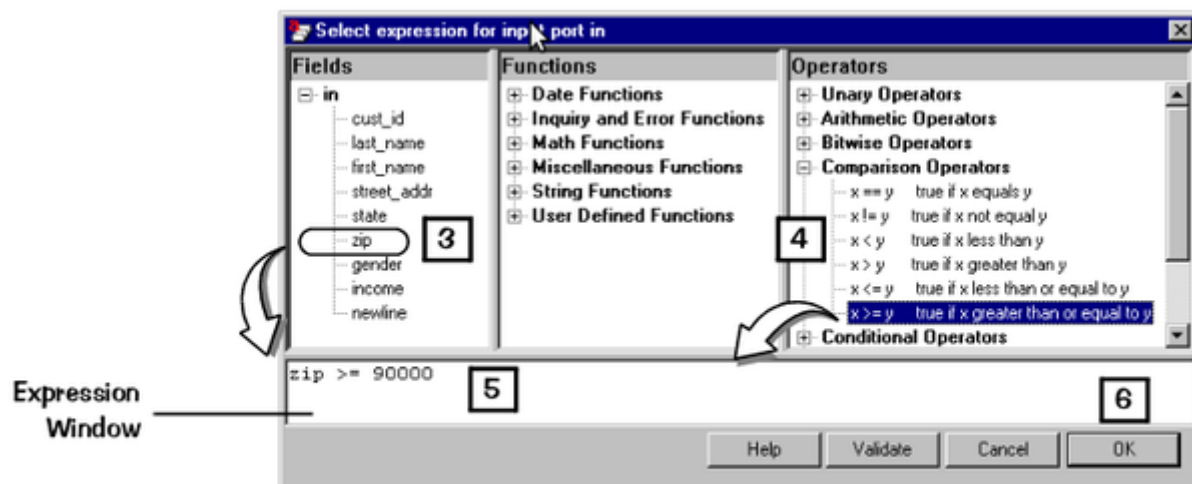


Figure 14 Expression Editor

Running The Application

To run your application, follow these steps:

1 Click the **RUN** Button

The GDE executes the application remotely on the host machine (control node) and displays an **Application Job Output** monitor on your machine (client node). When the application completes, the GDE displays the number of records that passed through each of the flows.

2 Watch the colour-coded status in the main window

The blue dot on the **Select** component indicates success. This application read 284 client records and selected 59 with zip codes greater than 90000, that is, west coast residents.

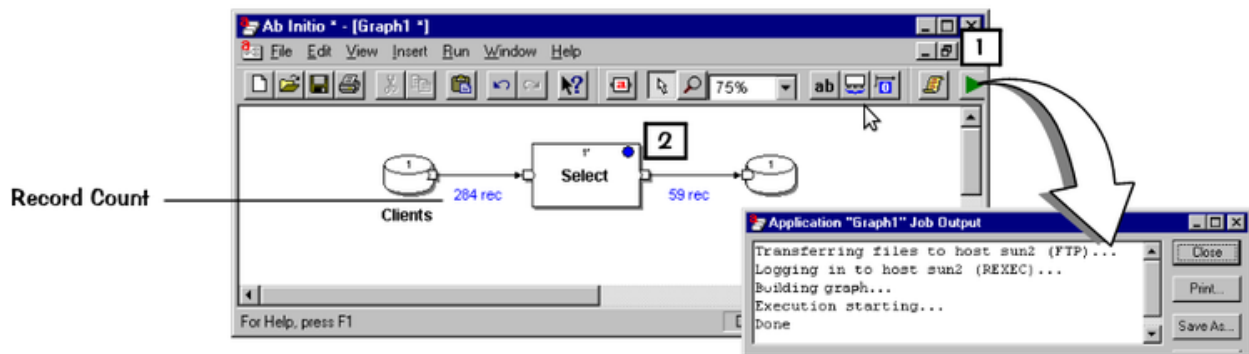


Figure 15 Runtime Version of the Select Application

Testing the Results

When the job completes, close the **Application Job Output** monitor window and look at a few records of the output file (Figure 16).

- Right-click the **West Coast** icon and select **View Data**. The View Data dialog appears.
- Use the pull-down menu, so **Records** are 1 through **End**. Also, make sure that **Show Results in Grid Mode** is checked.
- Click **OK** in the View Data dialog and check the customer records. The customer records should show that all customers in the output file have a zip code greater than or equal to 90000, west coast residents. Click **OK** when you're done, then close the Job Output Dialog.

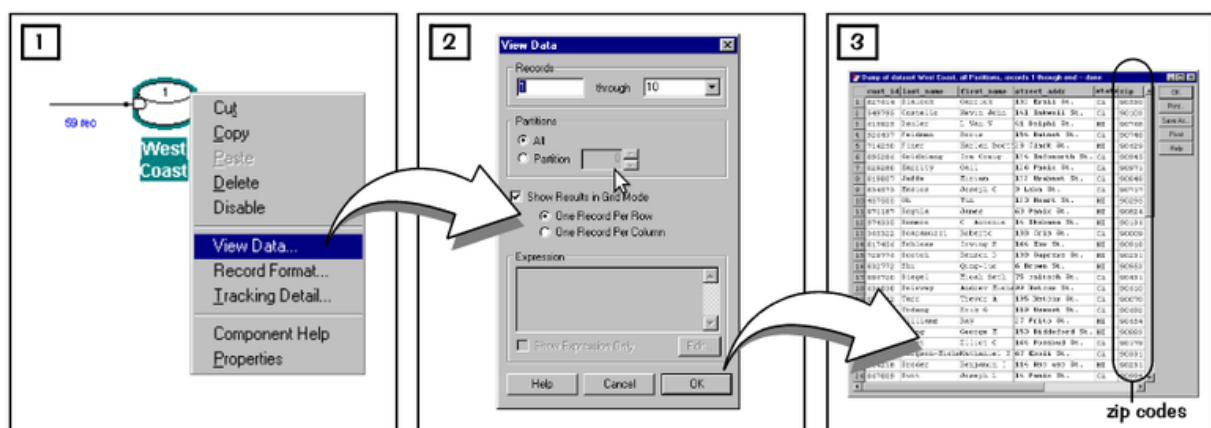


Figure 16 Viewing Output Data in the Select Application

Saving Graph

The default save location is the temporary directory (Figure 17) on your client node. For this tutorial, save your graphs there.

- Select **File >> Save As**. The Save As dialog appears.
- Type **select-customers.mp** in the **File name** text box to save your graph in a file named **select-customers.mp**.

Notice that all Ab Initio graphs use **mp** extensions. We'll discuss other Ab Initio file extensions later in this tutorial.

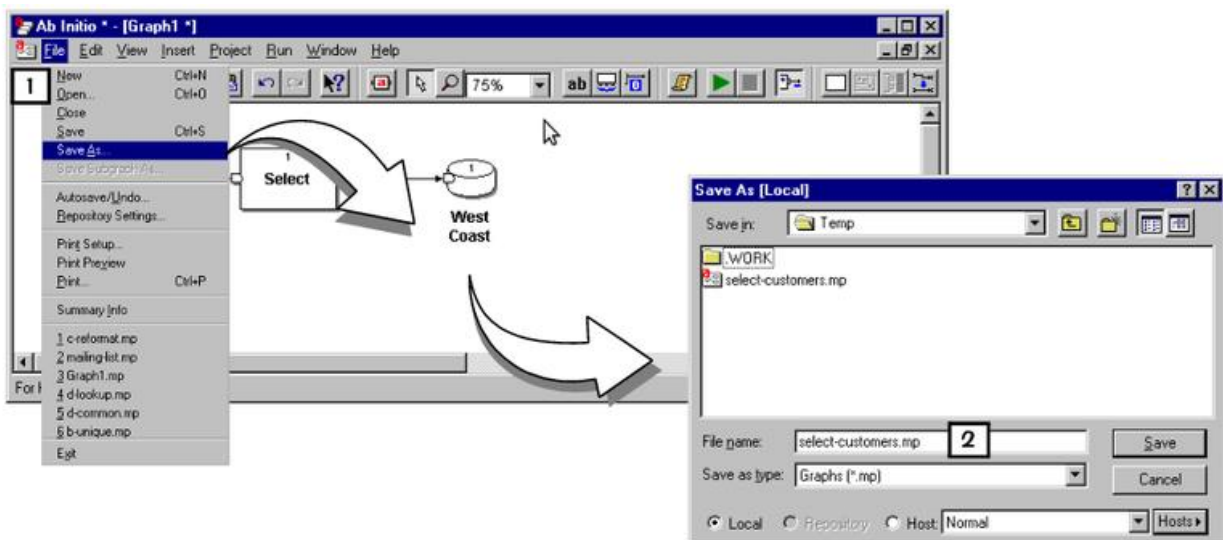


Figure 17 Saving the Select Application

Running the Tutorial Step Graph

The tutorial setup graph copies the files you need to complete this tutorial into your host directory (see sidebar). If you haven't configured the Run Settings (page 4), do so now. Then follow these steps:

- Pull down **Help >> Examples**.
- Double-click the **Advanced Tutorial** folder.
- Double-click **setup-tutorial.mp**.
- Click the **Run** button. You may get a warning message. Click **No** to ignore it. The graph should look like Figure 18.
- Select **File >> Close** then click **No**. You don't have to save it.

This graph copied 16 files in parallel to your host directory on the con-trol node. We'll talk more about parallelism in Lesson 7.

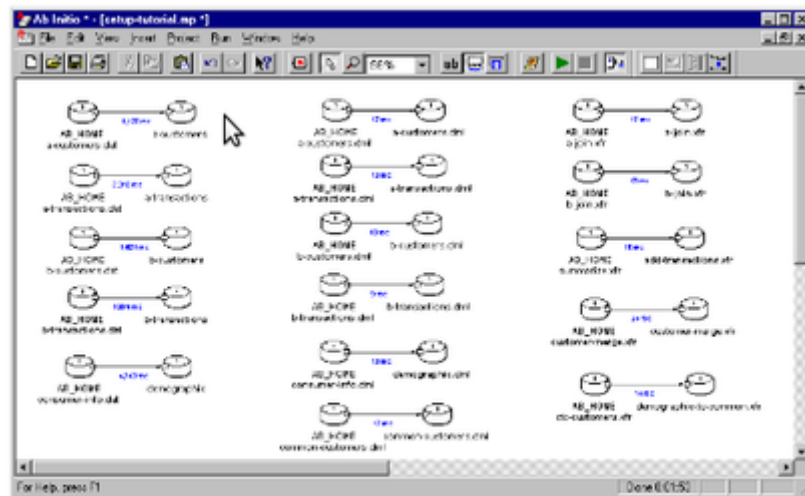


Figure 18 Setup Graph

Lesson 1 Exercise

Select customers by a different criterion.

- If necessary, open the **select-customers.mp** graph.
- Select customers by gender this time.
-

Hints: You need to write a Select expression that compares the gen-der field against “M” and “F”. DML conventions are listed in Appendix B.

Answer: See Appendix A

Lesson 1 Summary

Lesson 1 introduced the following ideas:

Component Organizer—storage place for components.

Components —the building blocks of Ab Initio applications. Dataset components are the inputs and outputs. Program components are the processing stages.

Deploy—method for copying a graph to the control node (sometimes called *host*) to run it without the GDE. The generated shell script and supporting files go to production.

DML—acronym for Ab Initio's Data Manipulation Language.

File Extensions—suffixes for your Ab Initio files. Use **dat** for data files, **dml** for record format files, and **xfr** for transform function files.

Flows—streams of data between components in your graph.

GDE Toolbars—menus and buttons at the top of the GDE workspace that let you manipulate the Graphical Development Environment including adding components and running applications. The toolbars are Main, Tool, Run, Phase, and Edit.

Graph—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 connected together embody your Ab Initio graph.

Ports—connection points between components and flows. Each component has at least one port (**in** or **out**). Some ports must be connected to flows; some ports may be connected to multiple flows. We'll talk about multiple flows later. Every port has a record format. Whenever possible, let record formats propagate.

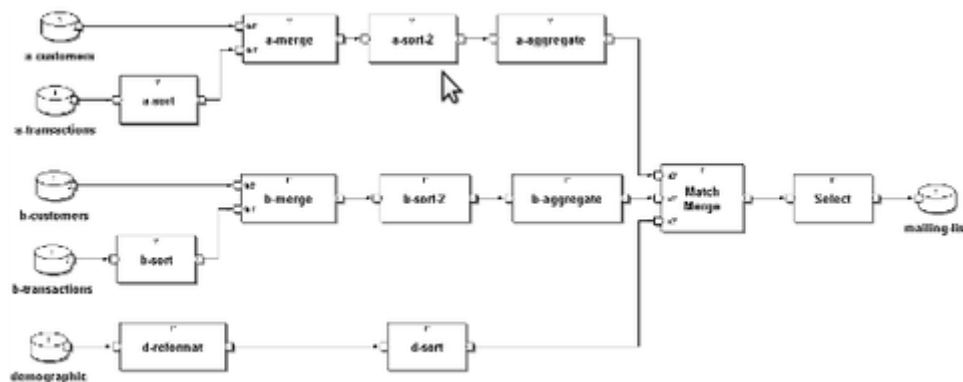
Run Settings—method of configuring the GDE to communicate with the control node. The Ab Initio Co>Operating System software resides on the control node.

Runtime Status—round colored indicators on your graph that give you direct feedback about the success or failure of your application. White is the starting position, green represents an in-progress state, blue stands for success, and red means error.

Yellow Reminders—highlighted areas in your graph that nudge you to supply missing information. These are also called "To Do" cues.

LESSON 2 Analyzing the Problem

- **Lesson 2 outlines the steps for investigating and resolving the big data problem you'll address in the rest of this tutorial.**
- By the end of Lesson 2, you'll understand the problem at hand, know how to divide it, learn which components to use first, and find graph building techniques. By the end of this book, you'll build a graph that looks like this



The Problem

In Lesson 1, you completed a simple example. Now you'll investigate and solve a *big data* business problem. The remainder of this tutorial builds the solution.

The Scenario

The Abbot and Becket retail stores just merged. You are chief of Information Services for the newly formed partnership.

As chief, you must combine the MIS operations of Abbot and Becket. The most time-critical operation is the direct mail campaign.

To accomplish this task, merge the customer and transaction records from the two chains, along with a consumer demographic list from a third party company to help identify new customers. Samples of customer records are shown in Table 3 on page 21. Samples of transaction records are shown in Table 4 on page 22.

The Raw Data

The MIS managers from Abbot and Becket apprised you of the state of their data. Both MIS departments report inconsistencies. The main problem is that customers may appear in records more than once, with different identifiers (IDs). Also, since both chains are national, some customers probably appear in both datasets with unrelated IDs. To answer business questions accurately, you must resolve these problems.

Customer Records and Formats

Table 3 shows an example of the format of Abbot's customer data, Becket's customer data, and the third-party demographic data. This formatted information is called a *record*.

Abbot's records have nine fields (or *columns* in relational database lingo).

Besides name and address, Abbot also flags *preferred* customers.

Becket's records have roughly the same data as Abbot's, though with different formats and *no preferred* flag. Both Abbot and Becket records are sorted by Customer ID (custid). This means the records are ordered from lowest to highest Customer ID number.

The third-party demographic records include names and addresses, plus some indicators of likely furniture buyers. Most of Abbot and Becket's customers—but not all—have entries in this file.

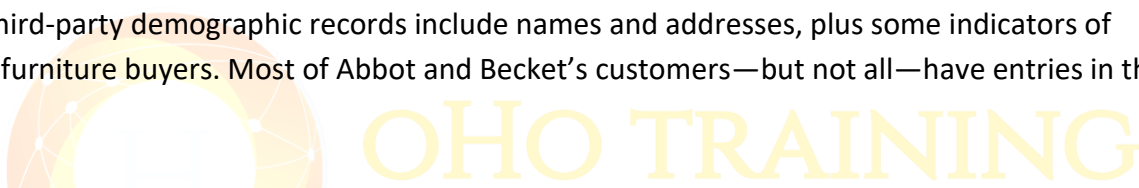


Table 3 Customer and Demographic Data

Abbot Customer Record		Becket Customer Record		Demographic Record	
Field	Explanation	Field	Explanation	Field	Explanation
custid	Customer ID	custid	Customer ID	id	Person ID
title	Mr., Ms., Mrs., or Miss	mr_or_ms	Mr. or Ms.	firstname	First name
fname	First name	firstname	First name	lastname	Last name
lname	Last name	lastname	Last name	street	Street address
street	Street address	street	Street address	city	City
city	City	zip	Zipcode	state	State
state	State	city	City	zip	Zipcode
zip	Zipcode	state	State	est_income	Estimated income
preferred	1 if preferred, 0 otherwise			own_or_rent	0 if owns home, R if rents
				since	Date moved to this address

Transaction Records and Formats

Both chains have kept complete records of purchases (Table 4) including a customer ID. The two chains carry much of the same merchandise, but their product IDs are completely different.

Abbot's transaction data is more detailed. Some transactions are merchandise returns, which encode the refunded amount in the price field. Becket's transaction data has few fields, with no information on discount pricing or product returns.

Table 4 Abbot and Becket Transactions

Abbot Transactions		Becket Transactions	
Field	Explanation	Field	Explanation
transid	Transaction ID	transid	Transaction ID
transtype	PUR for purchase, RET for return	prodid	Product ID
transdate	Date of transaction	price	Price
custid	Customer ID	quantity	Quantity
prodid	Product ID	transdate	Date of transaction
quantity	Quantity	custid	Customer ID
price	Price per unit		
discount	Percentage discount		
returnid	Matching purchase transid if return		
reason	Reason for return if return		

Your Mission

Under enormous time pressure, your job is to implement the direct mail campaign. You need to:

- Consolidate the customer and transaction records for the individual businesses.
- Sum the records from the individual businesses.
- Merge the records from the two businesses.
- Use these records, plus the demographic data in yet another format, to select the organization's *preferred customers*.

The Plan

After deliberation, you decide to:

- Pick criteria for *preferred customers*.
- Check the company records and demographic data (Table 3, page 21) to make sure you have enough information.
- Create a list of specific subtasks.
- Build graph segments that solve the subtasks, then link them.

Executing the Plan

The marketing department set the criteria for *preferred customers*. See the sidebar.

A glance at the customer records (Table 3, page 21) and the transaction records (Table 4, page 22) reveals that you have all the necessary input. You can glean the following information about individuals from the three sources in Table 3:

- Gender is specified in the **title** and **mr_or_ms** fields of Abbot's and Becket's customer records.
- Names [and](#) addresses are available in the two customer records files and the demographics file.
- Estimated income is available in the demographics file.
- Home ownership is recorded in the demographics file.
- Purchase amounts are available from the transaction files.

Your list of subtasks is:

- Extract all the relevant customer and transaction information and merge it in a single format (Lesson 3)
- Aggregate the records (Lesson 4)
- Reformat the demographic information (Lesson 5)
- Merge the customer, transaction, and demographic information, select the records for those who qualify as *preferred customers*, and sort the records by zip code for mailing (Lesson 6)

The next step is to start building graphs.

Steps for Building and Running Graph

The steps for building graphs are:

- Drag the dataset components (the inputs and outputs of your program) onto your workspace.
- Add the program components.
- Connect the flows to take advantage of *propagation*.

Propagation disseminates information automatically, fills in the blanks for you, and removes yellow reminders. Certain components, for example the Select component (in Lesson 1), propagate information automatically.

The sooner you connect flows, the sooner yellow reminders become useful.

- Remove all yellow reminders in your graph.

Yellow reminders tell you that information is missing. Double-click each yellow cue to expose its properties page

Again, start by supplying information to the dataset components, then proceed to the program components—working from the outside in.

As you proceed through this tutorial, you'll learn the different kinds of information Ab Initio requires.

- Run the graph.

Look at your graph and the job output window for status. The graph displays color-coded indication of success or failure.

Lesson 2 Summary

Lesson 2 discussed the following concepts:

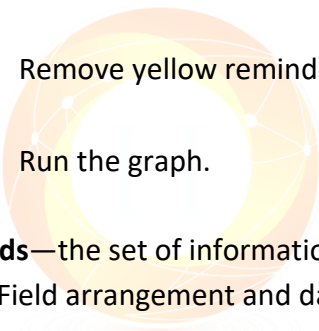
Big Data—large amounts of information, either in files, database tables, or SAS datasets. See sidebar.

Files—storage space for data. Exercises in this tutorial use data stored in files. In practice, Ab Initio recognizes files, database tables, and SAS datasets.

Graph Building Steps—the procedure for building graphs:

- Add dataset components.
- Add program components.
- Connect components with flows.
- Remove yellow reminders (To Do Cues).
- Run the graph.

Records—the set of information in data files. Records are composed of fields that describe the data. Field arrangement and data type comprise the format.



OHIO TRAINING
training at your desk

Lesson 3 Merging Records

You're in charge of consolidating records from three different sources and using it to make a mailing list of preferred customers. *Start by combining each chain's customer records with its transaction records.* To do so, group the data by a common field, then use a MergeJoin component to join the data.

By the end of Lesson 3, you'll build a graph that sorts data by a common key field (customer ID or **custid**), then merges customer and transaction records in one file.

Building a Graph to Comsolidate Data

- Just as you did in Lesson 1, create a graph. The steps are:
- Pull down File >> New in the Ab Initio GDE, then save as a-merge.mp. (Browse to your temporary directory, if necessary.)
- Drag five icons onto the graph from the Component Organizer.
- From the Datasets folder, choose two InputFile icons and one Output File icon.
- From the Sort-Merge folder, choose Sort.
- From the Transform folder, choose MergeJoin.
- Connect the icons with flows, as shown in Figure 19. Later, you'll let the yellow reminders guide you through completing the application

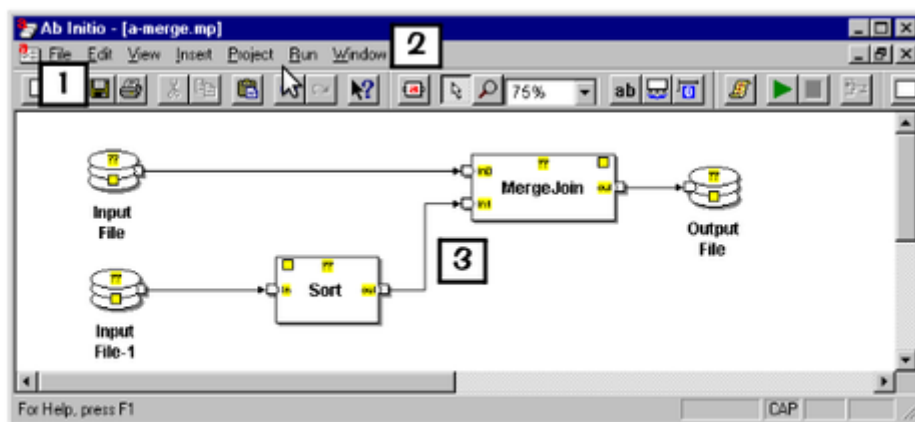


Figure 19 Components for Merging Abbot Customers and Transaction Data

Describing Data

Ab Initio's **InputFile** and **OutputFile** icons are component templates. In a previous lesson, we copied the data sources (Figure 18 on page 14) that made these templates useful. In practice, you need to:

- Tell Ab Initio the location of the input data.
- Supply a description of the data—either by filling in a grid in Ab Initio's Record Format Editor or by typing it using Ab Initio's Data Manipulation Language.

Locating Input Data

This tutorial provides Abbot's customer and transaction data. The customer data is in a file named **a-customers.dat**. It contains lines like:

00007602	Mr. Bill	Santiao	8392	Maple St.	Westfir	OR974920
00007603	Mrs. Susan	Sheldon	4324	Lee Highway	Holyoke	MA010410
00007604	Mr. Bill	Allen	11691	Mystic Road	Starbuck	AR993590

It's a fairly readable example of raw data, except for the extra digit after the zip code (the *preferred customer* flag). Often the data is unreadable EBCDIC, packed decimal, or binary.

The Abbot transaction data in **a-transactions.dat** has lines like:

```
087875123PUR09279600008372table5 00102455000
087875124PUR09199600008377dresser200104359505
087875125PUR09029600008225vase2 00100991510
```

Again, you'll use **View Data** to see this information more clearly.

To tell Ab Initio the location of the raw data and transaction files, associate them with the **InputFile** and **InputFile-1** icons, shown in Figure 20.

- Double-click the **InputFile** icon to display its properties.
- Label the file **a-customers** on the Description page.
-
- Type the location or path of the file using URL syntax, in this case: **file:dat/a-customers.dat**

This is the location of the data file of Abbot's customers. Another way to access this file is to **Browse**. Click the **Tutorial** folder, then the **dat** folder to find **a-customers.dat**. (Make sure your host profile is set according to Lesson 1.)

- Click **OK**.

The yellow reminder (Table 5) disappears from the top of the **a-customers** icon and from its Properties page; the double question marks (??) on the icon turn to 1; and the shape of the icon changes to a cylinder.

Table 5 Visual Cues on Your Graph

When you see	IT Means
Yellow reminders	Information is missing.
Double question marks	The missing information is a file location.
The number 1 on a dataset	Your file is serial, not parallel.

The number on the dataset refers to the level of parallelism, called *layout*. We'll talk about parallelism in Lesson 7.

The cylindrical dataset icon in the sidebar represents a serial file; the trisected icon represents a multifile. The default is a multifile icon.

To change a multifile icon to a file icon, you can edit the URL on the

Description page from **mfile:** to **file:**

- Double-click **InputFile-1**, label it **a-transactions**, then browse for or type the URL:
-

file:dat/a-transactions.dat

- Click **OK**.

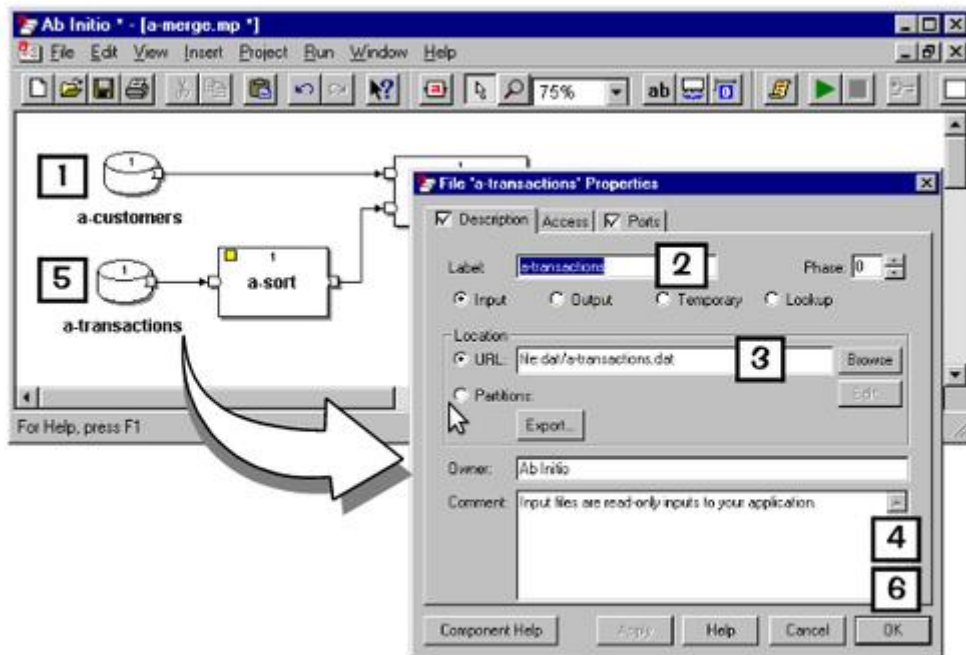


Figure 20 Abbot's Input Data

Describing Input Data

Now that you've told Ab Initio where your data is, you need to tell Ab Initio what the format of your data is. Ab Initio uses the DML language (see sidebar) to decipher your data. DML describes data formats, simple computations, business rules, and relationships between records.

In this case, we want to describe Abbot's record format.

- Double-click the yellow square in the middle of the **a-customers** icon to open the Properties dialog (Figure 21).
- Click the Ports tab.

From here, you can either invoke the Record Format Editor to create a new record format or specify a file that contains a record format. This tutorial supplies the file containing the record format.

- Select the **Path** radio button, then the **Host File** radio button.
- Browse or type the filename: **dml/a-customers.dml**
- Click **OK**.

The yellow square on the Read port of the Ports page and the middle of the InputFile icon disappear. This indicates that the system knows the format of the data that will flow through that port.

- Double-click **a-transactions**, click the Ports tab, click the Path radio button, then click the Host file radio button, and type or browse for: **dml/a-transactions.dml**
- Click **OK**. Your input files are now associated with DML files; the yellow reminders are gone.

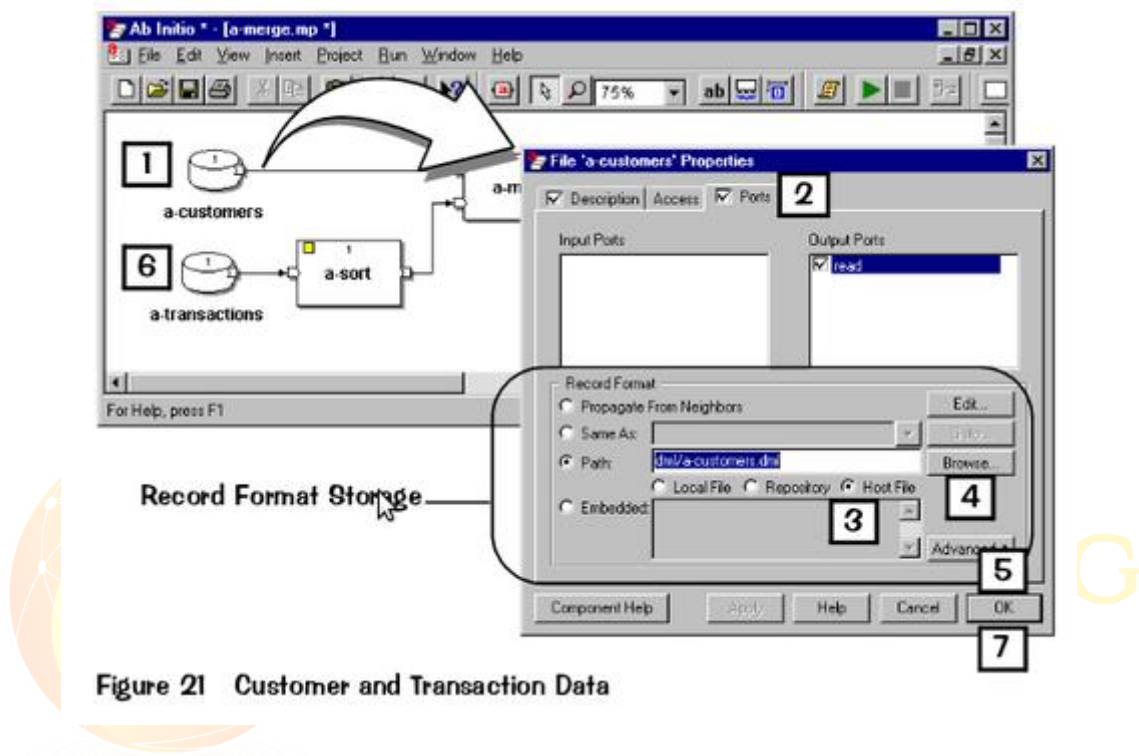


Figure 21 Customer and Transaction Data

Viewing Input Record Formats

DML can describe virtually any kind of data. Abbot's data uses only fixed-length ASCII strings and decimals, but DML can handle much more. (See sidebar page 35.) Look at the format of Abbot's customer records to see DML constructs that name the data fields and indicate their types.

- Right-click the **a-customers** port and choose Record Format to view the DML in the Record Format Editor (Figure 22).

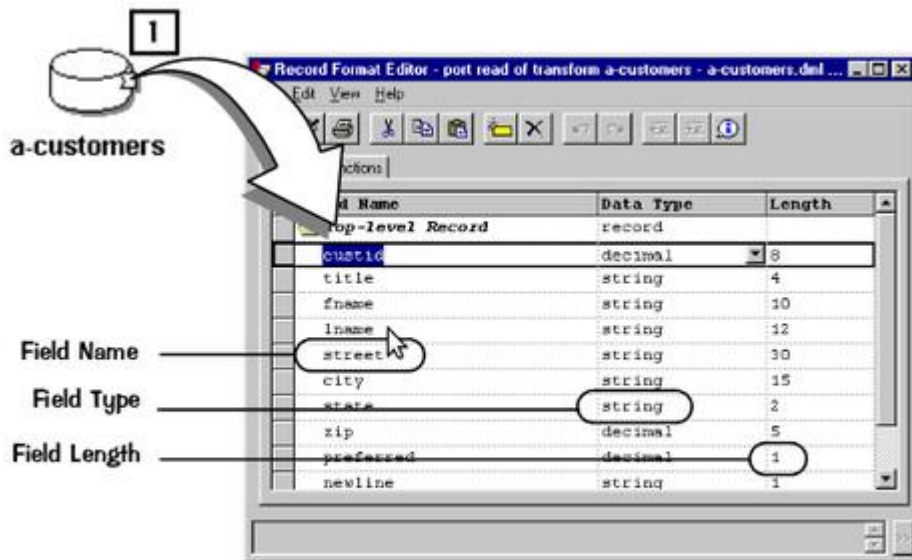


Figure 22 Abbot's Customer Records

- For a different view of the underlying DML, from the Record Format Editor, pull down View >> Text Mode (Figure 23).

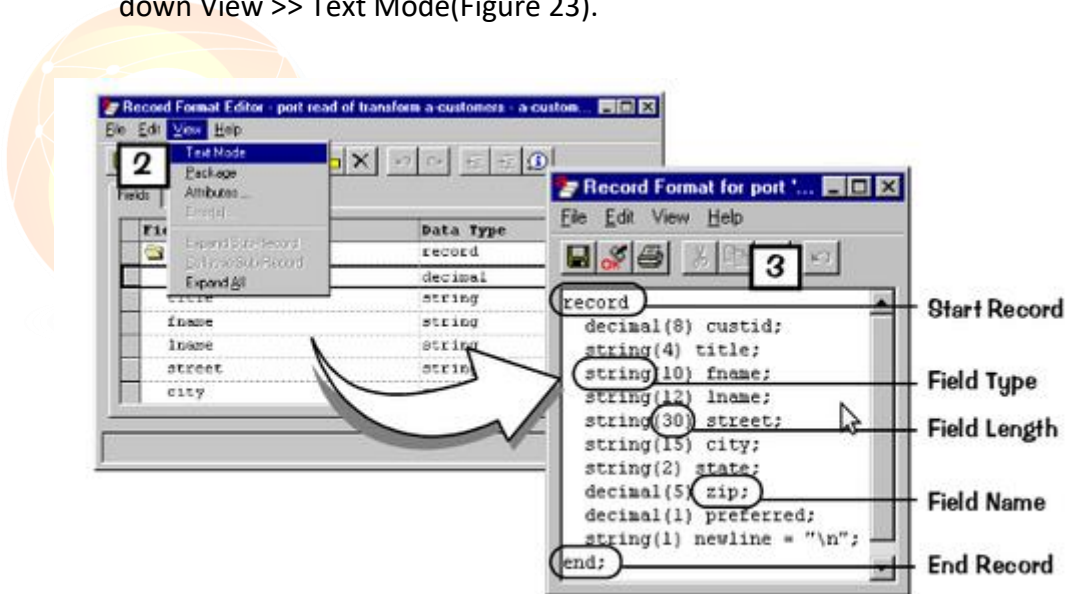


Figure 23 Viewing Abbot's Customer Records in Grid and Text Mode

- Look at the Record Format Editor in Text Mode. You can see that the graphical editor (Figure 22) handled the details of DML syntax.
- The **record... end** construct defines a record type.

- Between **record** and **end** are field declarations. Each consists of a type and a field name.
- In this example, fields have a fixed width in parentheses.
- Each field ends [with a](#) semicolon.
- The last field has an optional, but often-included default (="\\n"). Default field values simplify transform functions. We'll talk more about default fields in Lesson 5.

When you're done, pull down **View >> Grid Mode**. You'll use Grid Mode most of the time. To learn more about DML, see Appendix B.

Viewing Input Data

The GDE View Data feature displays data *as it is described by DML*. This view is more friendly than raw data because Ab Initio separates and labels the fields and translates as necessary so you can read them.

You used View Data in Lesson 1 to check the output. This time, check the input for **a-customers** and **a-transactions**. Let's recap the procedure:

- Right-click **a-customers** and select **View Data**.
- From the View Data dialog, check **Show Results in Grid Mode**.
- Press **OK** to display the first 10 records (Figure 24).

Hints:

To size a column, drag your cursor over the gray headers until you see a cross, then drag the column to expand or contract it.

To alphabetize a column, double click a header.

To view the list vertically (portrait) [rather than](#) horizontally (landscape), click the **Pivot** button.

	custid	title	fname	lname	street	city	state	zip	preferred	newline
1	00007602	Mr.	Bill	Santiago	8392 Maple St.	Westfir	OR	97492	0	\n
2	00007603	Mrs.	Susan	Sheldon	4324 Lee Highway	Holyoke	MA	01041	0	\n
3	00007604	Mr.	Bill	Allen	11691 Mystic Road	Starbuck	WA	99359	0	\n
4	00007605	Mr.	Dave	Robertson	985 Sugar Rd.	Platteville	CO	80651	0	\n
5	00007606	Mr.	John	Miller	6783 Lehigh Rd.	Greenville	SC	29613	0	\n
6	00007607	Ms.	Veronica	Jones	10625 Route 66	Charlottesville	VA	22901	0	\n
7	00007608	Mr.	Hector	Phillips	10191 Amber Way	Eagle Pass	TX	78853	1	\n
8	00007609	Mr.	Dave	Fuller	6410 Winter Drive	Escondido	CA	92025	0	\n
9	00007610	Ms.	Sarah	McLachlan	519 Main St.	Starbuck	WA	99359	0	\n
10	00007611	Ms.	Lena	Thomas	3145 Reece Ave.	Baldwin	IL	62217	0	\n

Figure 24 Viewing Abbot's Customer Data

Describing Output Data

Just like input datasets, you need to tell Ab Initio where to put the output data and describe its format. (See sidebar.)

- Double-click the **Output File** icon (Figure 25) to access its properties and label it **a-common**
- From the Description page, type the URL: **file:dat/a-common.dat** (Your output file will be stored in your working directory on the control node.)
- From the Ports page, select the Path **radio** button, then the Host File radio button, and type the location of the DML file: **dml/common-customers.dml**
- Press **OK** to accept your changes and close the dialog.

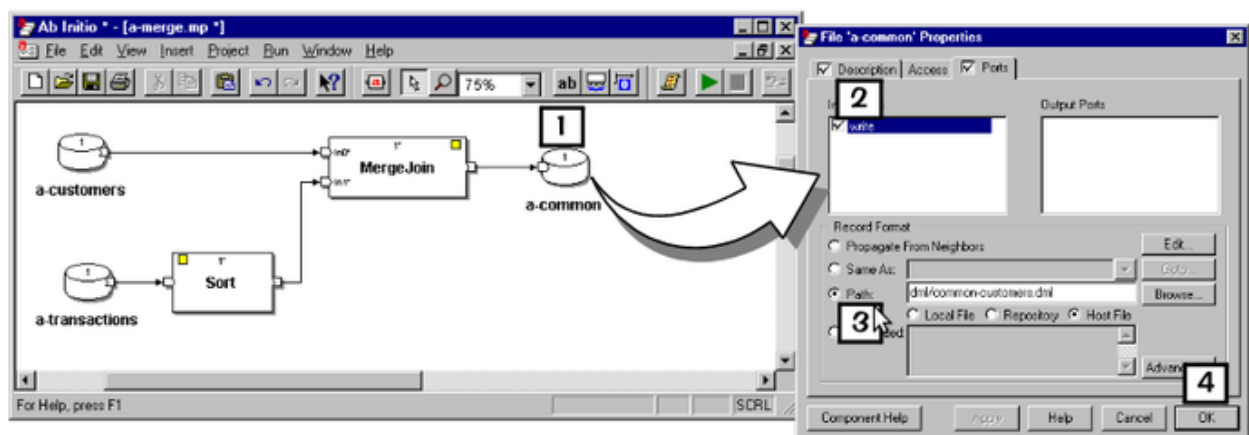


Figure 25 Output File for Abbot's Data

Assigning Parameter Values

Before you can merge the two input files, you need to sort both files on a common field, in this case **custid**. Since the customer file is already sorted by **custid**, you only need to sort the transaction file.

The transaction records flow from the **out** port of the dataset component to the **in** port of the Sort component. This component sorts the records and sends the re-ordered data to the flow connected to its out port.

- Double-click the **Sort** component to open its properties, change the label to **a-sort**, then click the Parameters tab. The yellow box tells you the component has a missing parameter. See Figure 26.
- Double-click the yellow box next to the **key** parameter to open the Key Editor.
- Select **custid** under Source Fields to sort by Customer ID.
- Click the **Add** button. You'll see **custid** under Key Fields. Click **OK** to close the Key Editor.
- Click **OK** again to close the Parameters page.

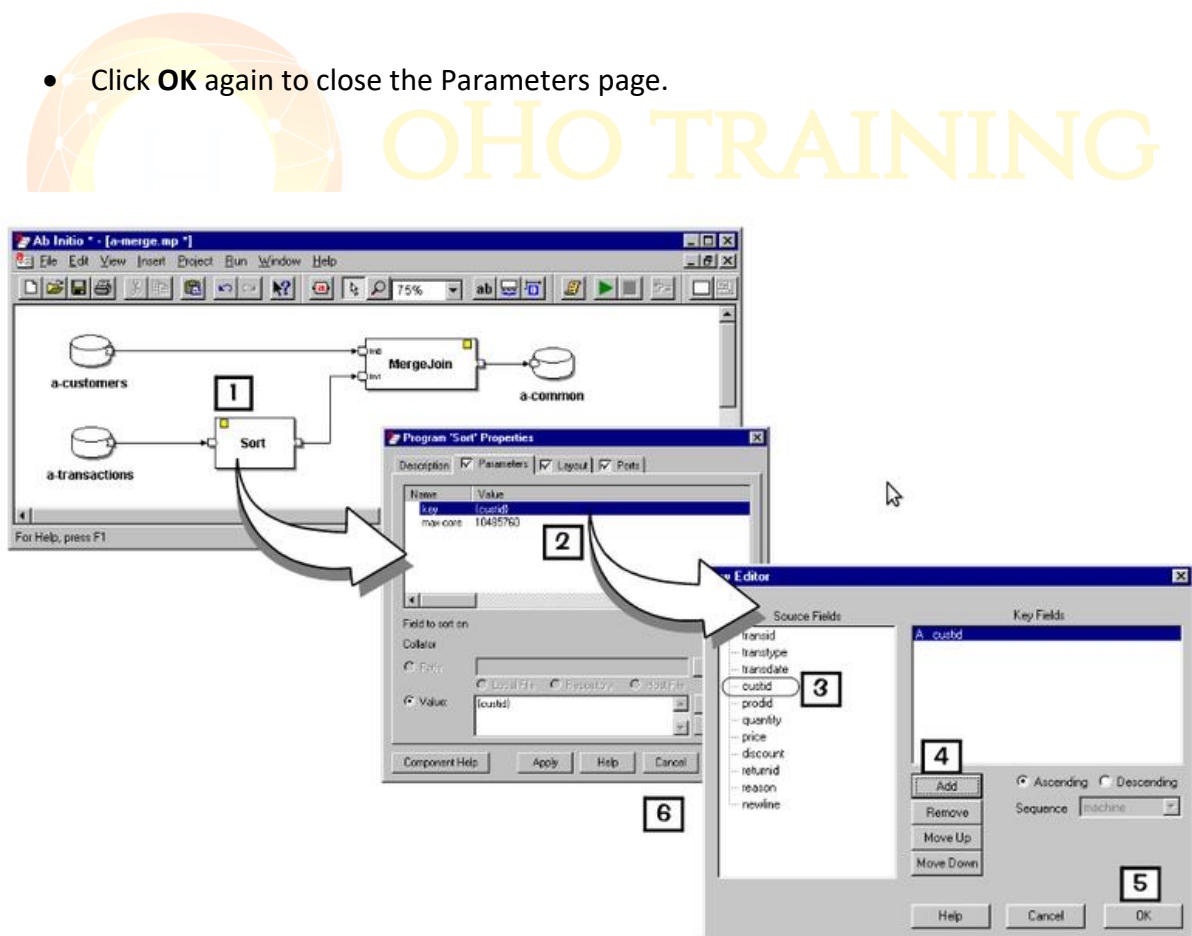


Figure 26 Parameter for Abbot's Sort Component

Setting the MergeJoin Parameters

The flow from the Sort component is one of two that feeds the MergeJoin component. Driven by a user-specified **key** and **transform function**, the component merges the stream of transaction records with the stream of customer records and sends the *joined* records to its **out** port.

- Double-click the MergeJoin component to open its properties, and label it **a-merge**. Click the Parameters tab. The yellow tells you the component has two required values, **key** and **transform**.
- Double-click **key** to open the Key Editor. Select **custid**, click the **Add** button, then click **OK** (Figure 27).
- Click **transform**, check the Host File radio button, then type the file name containing the transform function: **xfr/a-join.xfr**
- Set the **matchoptional0** and **matchoptional1** parameters to **True**. Don't worry about their meanings for now.
-
- Press **OK**, then **File >> Save** or click the **Save** button.

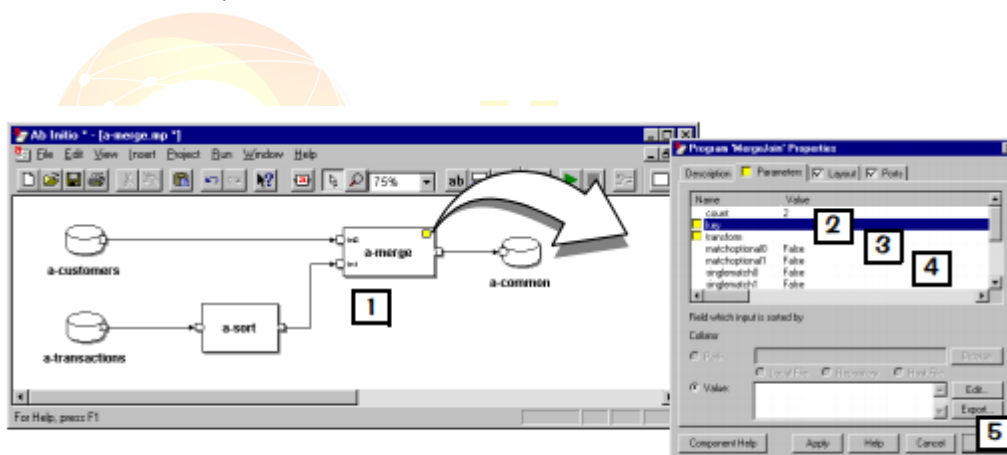


Figure 27 Parameter for Abbot's MergeJoin Component

Transforming Data

Although we're not giving you preformatted components, we're still providing transform function files. In the exercise at the end of this lesson, you'll write your own. For now, let's view the **a-merge** transform file:

- On the Parameters page of the **a-merge** component, double-click **xfr/a-join.xfr** to open the Function Editor (Figure 28). It displays three columns.

- The left column displays the *Input Tree*. It lists the input records and their fields. Arrows indicate the flow of data into the rules.
- The center column displays *Business Rules*, *Variables* and *Statements*. The *Business Rules* are assignment statements. *Variables* and *Statements* support complex business rules.
- *Variables* and *Statements* are beyond the scope of this tutorial. For more information, refer to the GDE on-line help.
- The right column displays the *Output Tree*. It lists the output records and their fields.
- Click any field in the Function Editor to select a cell. For example, Figure 28 shows what happens when you select the **custid** input field.

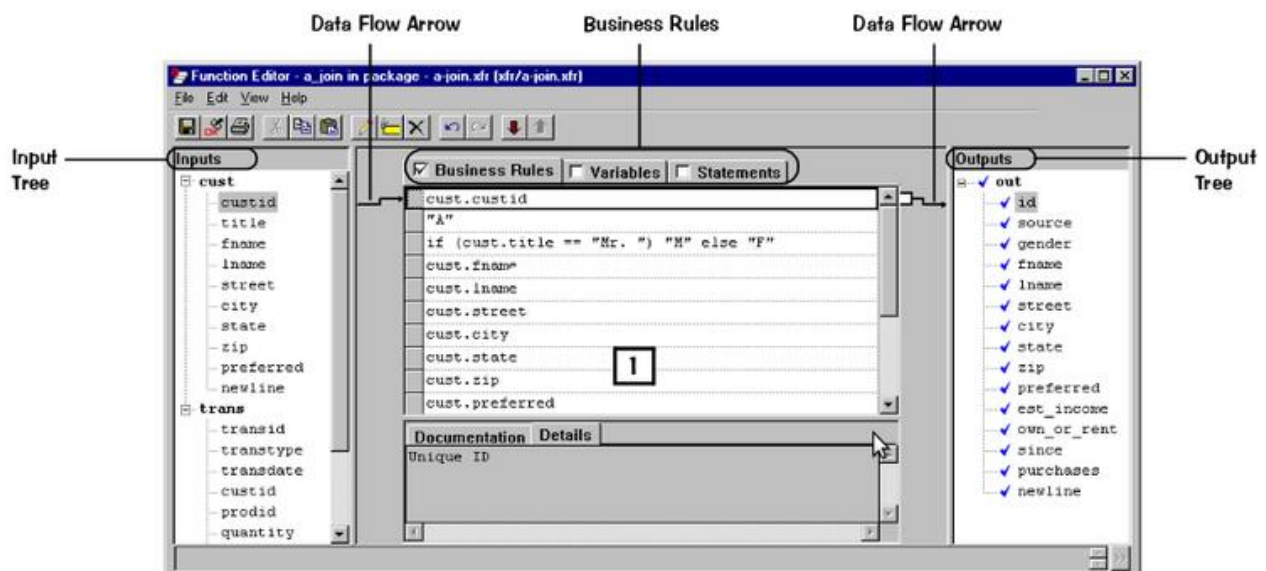


Figure 28 Abbot's MergeJoin Transform File

When you select a cell, the editor displays a box around the business rule and draws arrows to depict the data flow among the three columns, revealing which inputs and output the rule uses. The first rule in the center column tells Ab Initio to copy the data from the **custid** field in the **Inputs** to the **id** field in the **Outputs**.

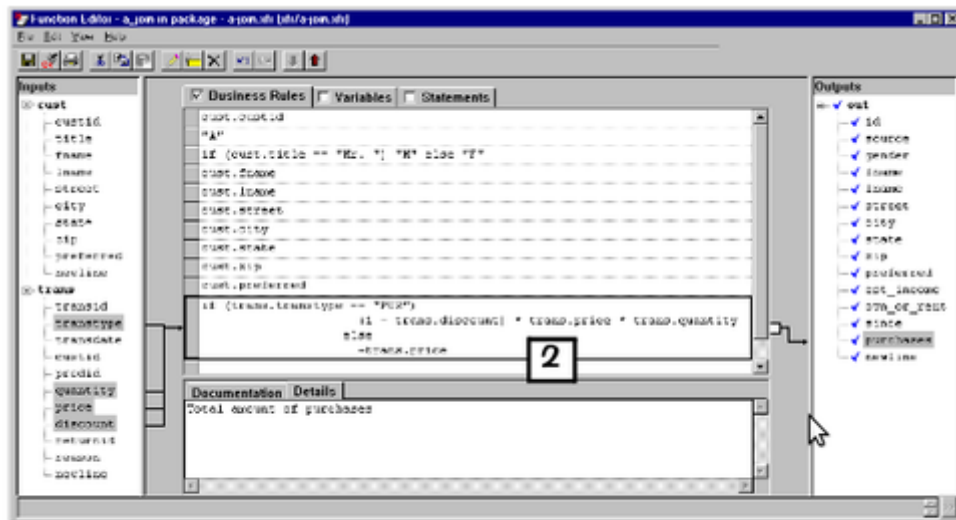


Figure 29 Data Flow in the Function Editor

- Click another cell to view other Data Flow Arrows. For example, Figure 29 shows what happens when you select the **purchases** output field. The editor displays a box around the selected business rule and draws arrows from the **Inputs** toward the **Outputs**.

Running and Testing Abbot's Data

- Click the **Run** button.

As the job executes, the GDE displays status information in the Application Job Output monitor and provides indicators on the graph with the number of records processed (Figure 30).

Make sure you get the correct number of records out (2469). If you didn't, make sure you correctly set the **matchoptional0** and **matchoptional1** parameters of **a-merge** to **True**.

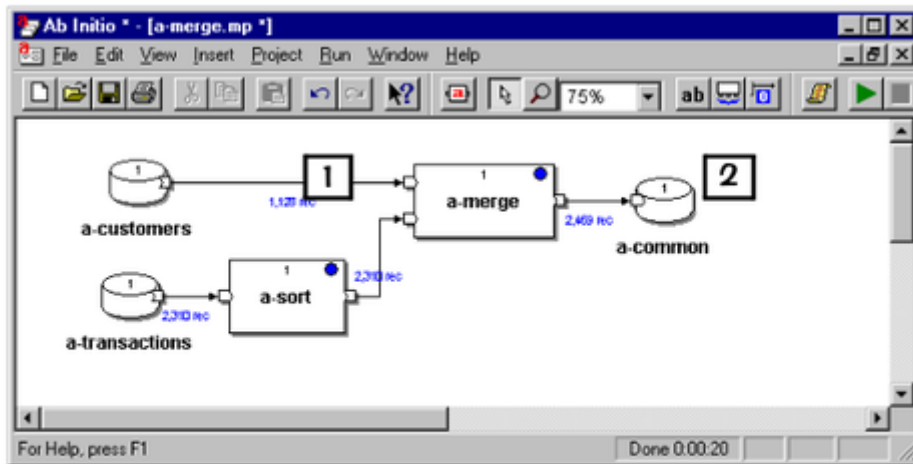


Figure 30 Runtime Version of Abbot's Merge Graph

2. Test the results by viewing the data on your output dataset. (Right-click **a-common** and select **View Data** in grid mode.)

Your output data should include both customer and transaction data. Compare the output dataset to your input datasets.

Note that the output has one record for each transaction, and that customer information may be repeated. This behavior is similar to a data-base join operation. See Figure 31. In the next lesson, we'll consolidate the information to yield one record per customer.

Customer Information										Transaction Data				
id	source	gender	fname	lname	street	city	state	zip	preferences	own	since	purchases	new	
1	00007602	A	M	Bill	Santiago	6392 Maple St.	Westfir	OR	97492	0	U		105500	\n
2	00007602	A	M	Bill	Santiago	6392 Maple St.	Westfir	OR	97492	0	U		13775	\n
3	00007603	A	F	Susan	Sheldon	4324 Lee Highway	Holyoke	MA	01041	0	U		42735	\n
4	00007603	A	F	Susan	Sheldon	4324 Lee Highway	Holyoke	MA	01041	0	U		56611	\n
5	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		316500	\n
6	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		22485	\n
7	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		65646	\n
8	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		76131	\n
9	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		73056	\n
10	00007604	A	M	Bill	Allen	11691 Mystic Road	Starbuck	VA	99359	0	U		92769	\n

Figure 31 Viewing Abbot's Merged Data


3. Save and close the graph: **a merge-mp**

LESSON 3 Exercises

The Abbot and Becket customer information (on page 21) and the transaction data (on page 22) are roughly the same, but the data and format differ. Your job is to create the **b-customers** dataset. We've given you **b-customers.dat**, but want you to create **b-customers.dml**.

Your starting material includes the Becket fields with an explanation. (See the sidebar for details.)

FIELD	EXPLANATION
custid	Customer ID
mr_or_ms	Title
firstname	First name
lastname	Last name
street	Street address
zip	Zipcode
city	City
state	State

- 
- Start a new graph; save it as **b-merge.mp**. In it, create an input dataset and name it **b-customers**. Set the URL to: **file:dat/b-customers.dat** and click OK.
 - Double-click **b-customers** to open the **Properties** dialog.
 - Click the Ports tab.
 - Select the Path radio button, then the Host File radio button.
 - Press the **New** button to open the Record Format Editor (Figure 32).
 - Fill in the fields using Becket's customer information (page 46). For each field, type a name in the **Field Name** column, a Data Type in the **Data Type** column, and a terminating character in the **Length** column. The first field is shown in Figure 32. Fill in the remaining fields. The correct answer is on the next page in Figure 33.

(For the Expression Editor, right-click the grid and choose Edit Rule.).

To get....

In The Length Column type...

comma terminator
newline terminator

","
"\n"

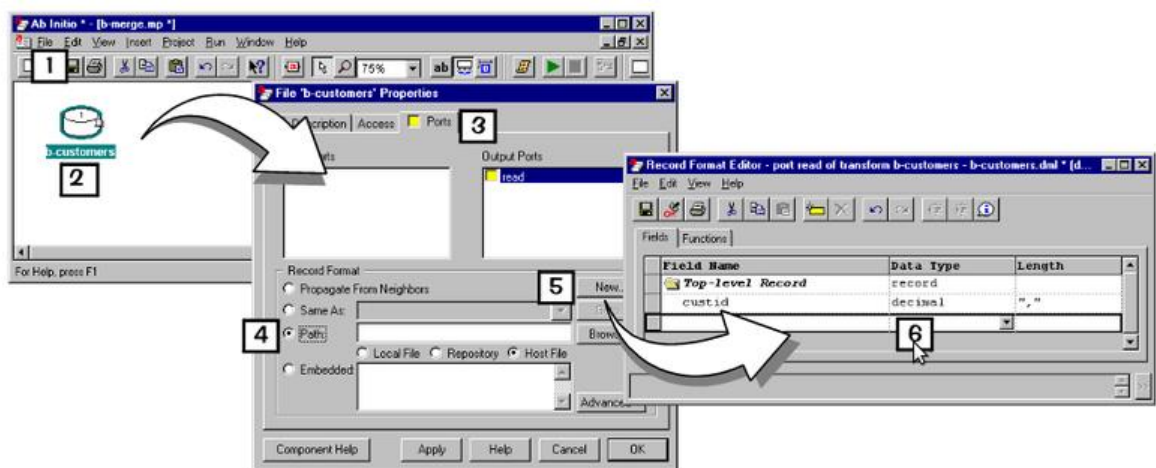


Figure 32 Record Format Editor

Your Record Format Editor should look like Figure 33.

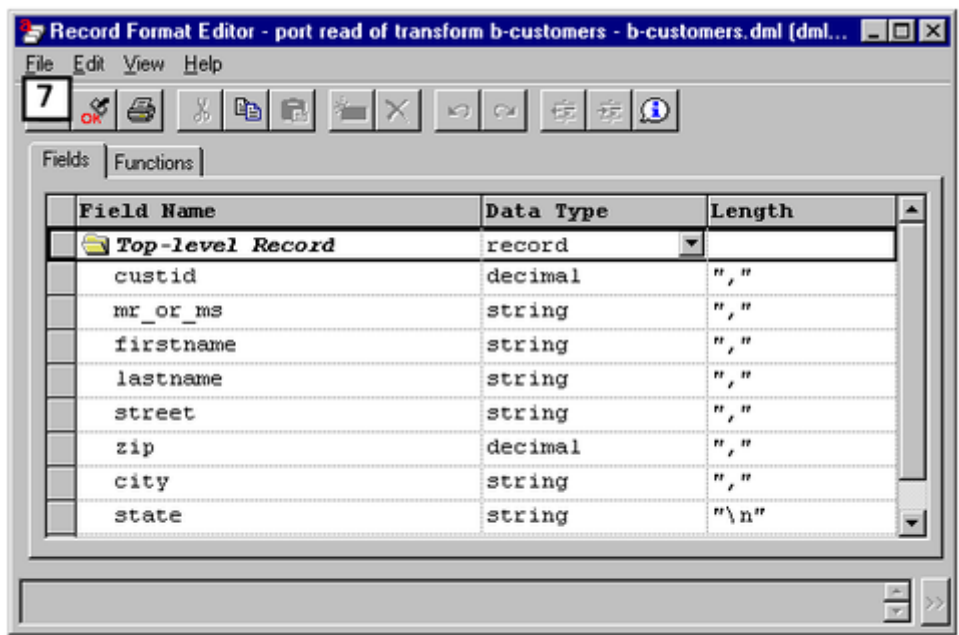


Figure 33 Becket's Record Format

- To finish, save the file as: **dml/b-customers.dml**

Close the Record Format Editor, and press **OK** to dismiss the Properties dialog. Test your results with **View Data**.

Note: If you're having trouble, refer to the file: **answers/b-customers.dml**

To get full tutorial, login with www.ohotraining.com



OHO TRAINING
training at your desk



Matt Demon, from Harvard University

E-mail: info@ohottraining.com

Website: <https://www.ohottraining.com>

training at your desk