Using SAS/ACCESS® Software and Database Management Systems in a Distributed Environment

Jana D. Van Wyk, SAS Institute Inc., Cary, NC

Presented at SEUGI ’92 by Bengt Bengtsson, SAS Institute

ABSTRACT
The growth of distributed computing environments that use more than one database management system (DBMS) and more than one operating system has made interoperability among data, software and hardware resources increasingly important. Using features of the SAS® System as tools to access and analyze these data across the network can make this interoperability possible.

There are three types of software that can be used to build a distributed application. First, SAS/ACCESS® software allows SAS users to access data in numerous database management systems on a variety of hardware platforms. Second, many database management systems offer transparent distributed access support for remote databases. And, third, SAS/CONNECT™ software offers SAS users connectivity between SAS sessions and numerous operating systems and hardware configurations. This paper describes a distributed SAS application that utilizes these three types of software in a distributed environment.

AUDIENCE
This paper is intended for SAS users whose data needs involve one or more database management systems in a distributed environment and who will benefit from using SAS/ACCESS software, the remote processing features of their DBMS software and SAS/CONNECT software to develop applications that access distributed data.

INTRODUCTION
The computing needs of businesses, universities and governments are changing and expanding to include accessing and analyzing data stored in a variety of different database systems on vastly different hardware platforms that are connected by various network protocols. The software industry is challenged to respond to these needs. This paper explores the development of a fictitious company’s computing needs as its business grows and diversifies over a period of time. In particular, this paper shows how the SAS System is used with the DBMS software to provide the fictitious company with the data storage, data analysis, data access, and network connectivity it needs to facilitate a complex distributed business application.

There are many technical components of a distributed configuration. This paper focuses on describing three primary components of one typical application using both local and distributed features:

• the use of SAS/ACCESS software to access a local database
• the use of SAS/ACCESS software in a homogeneous distributed DBMS environment to access a database on a remote node
• the use of SAS/CONNECT and SAS/ACCESS software in a heterogeneous distributed DBMS environment to access disparate databases on more than one node.

The combined use of this software addresses the computing needs of a large, diverse business situation and shows an integrated environment for a distributed application.

Functions provided by SAS/ACCESS and SAS/CONNECT software are as follows:

• SAS/ACCESS software uses the DBMS software to communicate with a database on a local node.
• SAS/ACCESS software takes advantage of the DBMS software to access a database on a remote node.
• SAS/CONNECT software performs the network operations that allow connectivity to a remote node.
• SAS/CONNECT software provides remote submission and data transfer between the local and remote nodes.

By utilizing the remote submission and data transfer features of SAS/CONNECT software, together with the access to local and remote databases supported by SAS/ACCESS software, we have simple, yet flexible tools to conveniently access multivendor databases across a variety of hardware platforms.

Overview of the Evolving Application
To illustrate some of the features of using the SAS System with DBMS, SAS/ACCESS and SAS/CONNECT software, the evolution of the needs of a fictitious company and its distributed environment and applications are discussed. Our fictitious company, Bookworms Inc., began as a wholesale book distributor with all business operations at a warehouse site. As their business profits grew over the years, they took the opportunity to purchase several retail bookstores in nearby cities. This second phase of their business investment also prospered. After several years, the company changed again when it was purchased by a conglomerate whose corporate headquarters was in another state.

As this business enterprise evolved over the years, their computing needs also diversified and expanded, and with each change in the business new hardware and software was integrated into their computing environment to accommodate these needs.

The initial computing environment at the warehouse of Bookworms Inc. consisted of a VMS system with ORACLE RDBMS used to store all inventory control information and the SAS System to handle all their data analysis and report processing. The SAS/ACCESS Interface to ORACLE was used to extract and update the inventory information stored in a local ORACLE database on the VMS system.

During the second phase of the company’s expansion, the company expanded and bought a chain of retail stores that manages their bookstore inventory using UNIX systems linked by a wide area network (WAN). A SYBASE SQL Server at one of the bookstores is used to manage the inventory and sales data. Each bookstore is a node in the network and has a front-end point-of-sale system that updates the SYBASE database with current inventory information when a book is sold. By using the SAS/ACCESS interface to SYBASE running on the VMS system at the warehouse site, a SAS application running on the VMS system could utilize the underlying SYBASE client/server software to access data residing in the remote database stored in the SYBASE SQL Server on the UNIX node.

In the third stage of the business expansion, our company of retail and wholesale book sales became very profitable and they are bought by a large conglomerate company. This conglomerate company maintains a DB2® database for monitoring financial and administrative information at the corporate headquarters. The DB2 database runs on the MVS operating system and now needs to be updated with the sales information that exists in both the ORACLE and SYBASE databases. By using SAS/CONNECT software to connect to the VMS site and transfer data across the network, together with the SAS/ACCESS Interface to DB2 running at the...
MVS site, an application on MVS can update data in the DB2 database with the ORACLE data on VMS or the SYBASE data on UNIX.

After the changes in their environment over several years of growth, the physical configuration of this company's network is illustrated by this figure.

This paper describes the technical details of how this physical configuration of hardware and software evolved and illustrates the software features that make the distributed data manipulation possible. Beginning with the initial local application at the warehouse, continuing with the addition of point-of-sale retail sites and ending with the addition of the corporate headquarters, each step in the evolution will be explained, and examples of the applications will be reviewed. An introduction to the SAS/ACCESS, SAS/CONNECT and distributed DBMS software terminology and features parallel to the three stages of business development are presented next.

Introduction to SAS/ACCESS Software
SAS/ACCESS software runs on a variety of hardware and provides an interface between the SAS System and several database management systems. Each SAS/ACCESS interface for relational database management systems includes the following three components:

1. the ACCESS procedure, which is used to create and edit SAS files called access and view descriptors that describe the data in the database
2. the database I/O engine, which is used by the DATA step and SAS procedures to read, add, delete, and update data based on the contents of view descriptors
3. the DBLOAD procedure, which creates and loads a database table with data from a SAS data set, a view descriptor of a DBMS table or a SAS SQL view.

By using these SAS/ACCESS components, you can perform the following tasks:

- Create SAS/ACCESS descriptor files using the ACCESS and DBLOAD procedures.
- Read data directly from the DBMS using the descriptors that were created.
- Create and load a table in the DBMS using the DBLOAD procedure.
- Delete, update or insert data in tables using the SQL procedure, SAS/FSP® software, SAS Screen Control Language, the DBLOAD procedure’s SQL statement, or the APPEND procedure.
- Create a SAS data set that contains DBMS data using the ACCESS procedure, the DATA step, the SQL procedure, or any procedures that can output a data file.

The power of the SAS/ACCESS interfaces comes primarily from accessing the DBMS data directly from SAS programs without making a copy of the data and storing it in a SAS data set. However, the database engine needs knowledge about which tables and columns are to be read from which database on which node in the network. The ACCESS procedure solves this problem by capturing the information needed to use a DBMS table and storing this information in access and view descriptors.

Access descriptors are created to describe the contents of a DBMS table or view including details such as database name, table name, column names, the corresponding SAS variable names, and the SAS formats and informats that will be used when data are accessed. View descriptors are created from access descriptors. Each view descriptor can define all or any subset of the columns and rows described in the access descriptor.

The DBLOAD procedure creates and loads a table in an existing database and provides a mechanism to allow ad hoc SQL statements, with the exception of SELECT statements, to be sent to the database. The DBLOAD procedure can load a table using data from a SAS data set, an SQL data view a SAS data step view, or data described by a view descriptor.

Transparent Distributed DBMS Access and SAS/ACCESS Software
Several SAS/ACCESS interfaces provide support for the transparent distributed access features of the underlying DBMS software. When using this feature of the DBMS software, users can access data, which resides in a database on a remote node, from their local node. The underlying software within the DBMS takes care of all network operations that support this distributed processing of data. The user of the DBMS software and the user of the SAS/ACCESS software simply use the correct syntax to indicate that the data resides in a database on a remote node.
Currently, these four SAS/ACCESS interfaces run under the VMS and/or UNIX operating systems and support transparent distributed access functionality:

- SAS/ACCESS Interface to ORACLE, Release 6.07
- SAS/ACCESS Interface to Rdb/VMS, Release 6.06 and 6.07
- SAS/ACCESS Interface to INGRES, Release 6.07
- SAS/ACCESS Interface to SYBASE, Release 6.07

The applications described in this paper contain examples using the distributed features of the SAS/ACCESS Interface to SYBASE. Appendix A of this paper will give more examples that show how to use the distributed features of SAS/ACCESS interfaces to ORACLE, Rdb/VMS, INGRES, and SYBASE.

Introduction to SAS/CONNECT Software

SAS/CONNECT software is a cooperative processing product that allows a local SAS session to establish conversations with one or more remote SAS sessions. This gives the user access to data and computing resources on multiple hardware platforms, regardless of their physical location.

The following chart displays some of the communications protocols that are currently supported by access methods available with SAS/CONNECT software, Release 6.07.

<table>
<thead>
<tr>
<th>local session</th>
<th>remote session</th>
<th>MVS</th>
<th>VMS</th>
<th>UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MVS</td>
<td>TCP/IP</td>
<td>TCP/IP TELNET</td>
<td>TCP/IP TELNET</td>
</tr>
<tr>
<td>MVS</td>
<td>TCP/IP APPC</td>
<td>TCP/IP TELNET</td>
<td>TCP/IP TELNET</td>
<td></td>
</tr>
<tr>
<td>VMS</td>
<td>TCP/IP DECnet</td>
<td>TCP/IP TELNET</td>
<td>TCP/IP TELNET</td>
<td></td>
</tr>
<tr>
<td>UNIX</td>
<td>TCP/IP TELNET</td>
<td>TCP/IP DECnet</td>
<td>TCP/IP DECnet</td>
<td></td>
</tr>
</tbody>
</table>

SAS/CONNECT software offers two capabilities: distributed processing and data transfer. By providing remote submission capabilities, SAS/CONNECT software enables the user to execute SAS programs on remote systems with all output and messages displayed in the local session. SAS/CONNECT software also provides file transfer capabilities so that SAS data sets, SAS catalogs and external files can all be transferred across various hardware platforms, giving you the ability to combine data from seemingly incompatible systems. The ability to transfer files and distribute processing to the appropriate host maximizes the utilization of your computing resources.

SAS/CONNECT software includes these SAS procedures and commands:

- The SIGNON command or statement initiates the connection between a local SAS session and a remote SAS session.
- The RSUBMIT command or statement executes the locally entered SAS statements on the remote system, and all generated log and output messages are sent back to the local session for display.
- The UPLOAD procedure copies a file stored on the local system to the remote system.
- The DOWNLOAD procedure copies a file stored on the remote system to the local system.
- The SIGNOFF command or statement terminates the connection between a local SAS session and a remote SAS session.

THE EVOLVING APPLICATION

This paper examines how the Bookworms Inc. computing environment grew over a period of many years and looks at three distinct phases of their expansion. In the following sections of this paper the physical network configuration and the use of SAS software during these three phases are described:

1. using SAS/ACCESS software locally
2. using SAS/ACCESS software in a distributed homogeneous DBMS environment
3. using SAS/CONNECT and SAS/ACCESS software in a distributed heterogeneous DBMS environment.

During the first phase of developing the company's computing enterprise, their warehouse inventory control applications ran locally on a single VMS system. The goal of these applications was to use their VAX computer running the VMS operating system and link the data in their ORACLE database with the data analysis and presentation tools in the SAS System. The solution to this problem was to use the SAS/ACCESS Interface to ORACLE running under VMS to bridge the gap between the DBMS and the SAS System. All of the software ran locally on the single node VMS system.

The second phase of the computing environment was initiated in response to the company's entry into the retail bookstore business. Since the bookstores used a UNIX-based WAN with SYBASE SQL Server running on one node for the point-of-sale data acquisition, it was necessary to link the SYBASE database with the existing applications on the VMS system at the warehouse site. A DECnet network was installed to link the UNIX and VMS systems. An application was developed using SAS/ACCESS Interface to SYBASE running under VMS at the warehouse node. This SYBASE client application ran the SAS/ACCESS software as a frontend and accessed the SQL Server backend running on the UNIX system by using the client/server features of the SYBASE software and the DECnet network protocol. This part of the application illustrates using SAS/ACCESS software in a distributed homogeneous DBMS environment.

After Bookworms Inc. was purchased by a larger company with corporate headquarters that used DB2 under VMS to store their financial information, the third phase of the computing change was introduced. The problem of how to give the corporate headquarters access to the data in all three databases — ORACLE at the warehouse, SYBASE at the retail store and DB2 at the corporate site — was addressed during this third phase. The solution to this problem is to use SAS/CONNECT and SAS/ACCESS software as well as the distributed access features of the DBMS software to support data access and transfer between all three nodes. This distributed application shows how to use SAS/CONNECT and SAS/ACCESS software in a distributed heterogeneous DBMS environment.

This DBMS environment uses SAS software to allow a user at the local VMS site to read and update data in ORACLE on VMS, SYBASE on UNIX, and DB2 on VMS. SAS/CONNECT software's TCP/IP access method is used to connect from the VMS site to the VMS site and transfer data between these two nodes. From the local VMS site, SAS/CONNECT software remotely submits SAS code that invokes the ORACLE engine running on the VMS site to extract and update data in ORACLE. Similarly, from the local VMS site, it is possible to remotely submit a job to invoke the SYBASE engine on VMS working with the Open Client SYBASE software to extract or update data in the remote SYBASE database on the UNIX system.

Using SAS/ACCESS Software Locally

The initial computing environment for Bookworms Inc. served their needs as bookstore distributors. At the warehouse site they installed a VMS system with ORACLE as the database management system.
to store their inventory and sales information. There was no network needed and the software environment can be pictured as follows.

In the ORACLE database three tables were created. The TITLES table contained the columns BOOKID, TITLE, and AUTHOR. A few of the rows in TITLES are:

<table>
<thead>
<tr>
<th>BOOKID</th>
<th>TITLE</th>
<th>AUTHOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>12333</td>
<td>Sushi, Anyone?</td>
<td>Yokomoto, Akiko</td>
</tr>
<tr>
<td>92345</td>
<td>Onions, Leeks and Garlic</td>
<td>O'Leary, Michael</td>
</tr>
<tr>
<td>22312</td>
<td>The Gourmet Microwave</td>
<td>Ringer, Albert</td>
</tr>
<tr>
<td>76354</td>
<td>Gastronomic Treats</td>
<td>Dull, Anne</td>
</tr>
</tbody>
</table>

The SALES table contained columns for bookid, the number of books sold so far this year, and the year-to-date sales for this book. A few of the rows in SALES are:

<table>
<thead>
<tr>
<th>BOOKID</th>
<th>SOLD</th>
<th>YTDSALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>12333</td>
<td>1320</td>
<td>14454.00</td>
</tr>
<tr>
<td>92345</td>
<td>235</td>
<td>3757.65</td>
</tr>
<tr>
<td>22312</td>
<td>766</td>
<td>18767.00</td>
</tr>
<tr>
<td>76354</td>
<td>51</td>
<td>1119.45</td>
</tr>
</tbody>
</table>

Columns holding the bookid, the number of books available in the warehouse, the number of books desired on the shelf to meet ordering demand, and the number of books on order from the publisher are in the table STOCK. A few of the rows in STOCK are:

<table>
<thead>
<tr>
<th>BOOKID</th>
<th>ONSHELF</th>
<th>NEEDED</th>
<th>ONORDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>12333</td>
<td>55</td>
<td>160</td>
<td>105</td>
</tr>
<tr>
<td>92345</td>
<td>70</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>22312</td>
<td>304</td>
<td>32</td>
<td>321</td>
</tr>
<tr>
<td>76354</td>
<td>32</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>

The following application shows how someone at Bookworms Inc. using the SAS System under VMS can

- create an access and a view descriptor of an ORACLE table
- read data from an ORACLE table using the ORACLE engine
- update data in the ORACLE table using the SQL procedure and the ORACLE engine
- read data from an ORACLE table using the SQL procedure and the ORACLE engine.

One of the daily jobs that is run by the book purchaser lists the bookids of the books that need to be ordered from the publisher. The following code shows how the ACCESS procedure is used to create an access and view descriptor of the table named STOCK. In Release 6.06 the SAS/ACCESS interfaces allowed creation of descriptors through the full-screen interactive use of the ACCESS procedure only. In Release 6.07 of some SAS/ACCESS interfaces, there is an additional method of descriptor creation available: the ACCESS procedure supports line mode syntax, which can be used in interactive or batch mode. This is an example of line mode use of the ACCESS procedure.

```
proc access dbms=oracle;
  create oracc.stock.access;
  user=Orauser;
orapw=orapass;
table=stock;
  create oraview.stock.view;
  select all;
run;
```

Note that descriptors do not need to be created every time they are used. They can be created once, stored in a permanent SAS library and used repeatedly.

When the following PRINT procedure executes, the ORACLE engine uses the information in the descriptor to connect to the database and retrieve all columns in the view descriptor from the table. The WHERE statement used with the PRINT procedure is passed to the ORACLE engine, and only the rows that fulfill this subsetting restriction are retrieved from the database.

```
proc print data=oraview.stock;
  where onshelf < needed;
run;
```

After viewing the list of books that need to be ordered the purchaser may decide to order these books. When the order is placed, the table STOCK needs to be updated to reflect the fact that these
books were ordered. At this time the following code can be executed to update the table in the ORACLE database:

```
proc sql;
  update oraview.stock
    set onorder = needed - onshelf
    where onshelf < needed;
  select * from oraview.stock
    where onshelf < needed;
quit;
```

The UPDATE statement used in the SQL procedure uses the view descriptor of the ORACLE table and updates the ONORDER column of the table to show the number of books that were ordered.

In these examples, the SAS/ACCESS and the ORACLE RDBMS software are executing at the VMS site, since there is no need for any network operations in this application.

Using SAS/ACCESS Software in a Distributed Homogeneous DBMS Environment

As described in the introduction, after a period of time Bookworms Inc., which had been strictly a wholesale book distribution company, decided to expand into the retail book sales market. In the new computing environment they use UNIX systems for their point-of-sale applications to manage inventory, for example, to record the sale of each book. The SYBASE database management system provides the data storage of the inventory and sales data via a SYBASE SQL Server installed at one of the stores.

The physical configuration of these two nodes and their DBMS systems and installed software is illustrated by this figure:

The following SAS code will show how Bookworms Inc. using the SAS System under VMS can

- use the pass-through facility of the SQL procedure to read data from a table in the SYBASE database on the remote UNIX node
- update data in the SYBASE table using the pass-through facility of the SQL procedure.

When a book purchaser, who is running SAS at the warehouse, wants to read and update data stored in the STOCK table in the SYBASE database at the point-of-sale site, the following SQL procedure can be executed. This code is executed in the local SAS session on the VMS node, but SAS/ACCESS software uses the SYBASE distributed access features to transparently connect to the SYBASE server at the remote UNIX node and query the database named INVENTORY for information about the table STOCK.

This example shows the pass-through feature of the SQL procedure and reads data from the STOCK table in the SYBASE database named INVENTORY. The pass-through facility of the SQL procedure used together with the SYBASE engine allows you to send SYBASE Transact-SQL statements directly to the SYBASE DBMS from the SAS System. The pass-through facility allows you to connect to any SYBASE server, execute SQL statements and disconnect from the SYBASE server. There are four extensions to the SQL procedure syntax to support the pass-through facility:

1. connect to dbms( dbms-connect-options ) [as alias];
2. execute ( dbms-sql-stmt ) by dbms;
3. select ... from connection to dbms( dbms-select-statement );
4. disconnect from dbms;

The following code uses the pass-through facility to display data from the STOCK table in the user's SAS session.

```sql
proc sql;
  set onorder = needed - onshelf
  where onshelf < needed;
  select * from oraview.stock
  where onshelf < needed;
quit;
```
The CONNNECT statement is used to initiate a session with the SYBASE server on the remote UNIX node. The SELECT statement passes the statement between the parentheses to the SYBASE engine. SYBASE executes this SELECT statement within the context of the server on the remote node. The transparent distributed feature of the SYBASE client-server software allows the data to be extracted from the table on the remote node and accessed by the local SAS software. This SAS code is submitted in the local SAS session on the VMS node, but SAS/ACCESS software uses the SYBASE distributed access features to transparently extract data from the remote SYBASE database on the UNIX node.

After reviewing the output of the SELECT statement, the purchaser executing the query decides which books need to be ordered. When the order is placed, the table's columns are updated by executing the following EXECUTE statement in the current SQL procedure session:

```
execute
  (update stock
    set onorder = needed - onshelf
    where onshelf < needed)
  by sybase;
```

Again, this SAS code is submitted in the local SAS session on the VMS node, but SAS/ACCESS software uses the SYBASE distributed access features to transparently pass the update statement to the remote UNIX node and update the SYBASE database there. This statement updates the values in the ONORDER column to reflect that the books have been ordered.

After the order has been received at the bookstore, the following code can be executed in the SQL procedure session to update the ONSHELF and ONORDER columns and record that the order has been processed:

```
execute
  (update stock
    set onshelf = onshelf + onorder, onorder = 0
    where( onshelf < needed and onorder > 0 ))
  by sybase;
```

Using SAS/CONNECT and SAS/ACCESS Software in a Distributed Heterogeneous DBMS Environment

The preceding section described how the computing environment of Bookworms Inc. used the SAS/ACCESS Interface to SYBASE software in a distributed homogeneous environment. The company now enters the third phase of their growth when the wholesale distributorship and the retail bookstore chain of Bookworms Inc. is purchased by another company. In addition to the two applications already on the network — at the warehouse and at the bookstore — there is a third interface added: the corporate headquarters of the company that has acquired Bookworms Inc.

At the corporate headquarters, the company uses an MVS system running the DB2 DBMS to store their corporate data. It is now necessary to link together these three seemingly incompatible systems: the warehouse site running ORACLE under VMS, the bookstore node running SYBASE under UNIX, and the corporate site running DB2 under MVS.

The physical configuration of these three nodes and the software installed on each node is illustrated by this figure:

The solution to the problem of how to link together the three sources of enterprise data is to use SAS/ACCESS and SAS/CONNECT software. A description of this software environment shows the use of SAS/ACCESS and SAS/CONNECT software in a distributed heterogeneous DBMS environment. The application is distributed because databases exist on more than one node in a network, and the environment is heterogeneous since there are three distinct database management systems utilized in the applications: ORACLE, SYBASE and DB2.
This application requires the functionality of three types of software products:

- SAS/ACCESS interfaces are necessary to allow the SAS System to read and update data in each DBMS.
- SAS/CONNECT software links together the processing power of the SAS software running under the MVS and VMS nodes.
- Transparent distributed access of the SYBASE database, provided by the client/server SYBASE software and supported by SAS/ACCESS software, bridges the UNIX and VMS nodes and gives access to the SYBASE database.

Suppose a sales analyst at the corporate site, who is using the SAS System under MVS, wants to read the current sales figures for the bestselling books at both the warehouse and at the retail bookstores. The following steps of a distributed application show how the analyst completes this task. Each of these steps is described in detail.

- Use SAS/CONNECT software to connect from the local MVS session to the remote VMS system.
- Use SAS/CONNECT software to remotely submit code, from the MVS system to the VMS system, that uses the pass-through utility of the SQL procedure to create an SQL view of a table in the ORACLE database on the remote VMS node.
- Use SAS/CONNECT software to remotely submit code, from the MVS system to the VMS system, that uses the pass-through utility of the SQL procedure to create an SQL view of a table in the SYBASE database on the remote UNIX node.
- Use the SAS/ACCESS ORACLE engine to read data from the ORACLE table and the SAS/ACCESS SYBASE engine to read data from the SYBASE database; then use the DOWNLOAD procedure to copy data from the VMS system to the local MVS system.
- Use the DB2 engine and the APPEND procedure to add rows to a DB2 database.
- Use SAS/GRAFH® software and the DB2 engine to read data from the DB2 table and create a bar chart.

First, our application must gain access to the remote VMS system by using SAS/CONNECT software's ability to establish a remote connection. By submitting the following code that uses the SIGNON statement, the local MVS SAS session can establish a connection to the remote VMS session:

```
filename rlink 'vms.scr';
options comamid=tcp remote=vmsnode;
signon;
```

By defining the default fileref RLINK, the script file VMS.SCR is executed by the SIGNON statement. The OPTIONS statement specifies that the TCP/IP communications access method is used and that the remote session id with which a connection to the VMS system will be established is VMSNODE.

After the connection to VMS has been established by SAS/CONNECT software, the next step is to use the pass-through facility of the SQL procedure with the SAS/ACCESS ORACLE engine to create an SQL view of the table. The following SAS code is sent to the VMS host for execution:

```
rssubmit vmsnode;
proc sql;
connect to oracle
  ( user=orauser orapw=orapass );
create view orasql.wholev as
  select * from connection to oracle
  (select bookid, title, ytdsales
   from sales, titles
   where sales.bookid = titles.bookid and
   ytdsales > 5000);
disconnect from oracle;
quit;
endrssubmit;
```

These statements are shown bracketed by the SAS/CONNECT statements RSUBMIT and ENDRSUBMIT. When these statements are submitted on the local processor, SAS/CONNECT software submits the intermediate statements to the SAS session on the remote VMS system for execution. All output and messages are then sent back to the local MVS session.

The CONNECT statement in the SQL procedure code causes the SAS session to connect to the ORACLE database on the VMS system using the options specified within the parentheses. The CREATE statement creates an SQL view of the data described by the SELECT statement. This SELECT statement, which joins the relevant data between the tables SALES and TITLES, is stored in the SQL view and executed by the ORACLE engine when the view is used.

After the SQL view of the ORACLE data is created, the following code is submitted in the MVS session to create an SQL view of the bookstore sales data, which are stored in the SYBASE database:

```
rssubmit vmsnode;
proc sql;
connect to sybase
  (user=sybuser sybpw=sybpasseybaseinv);
create view sybsql.retailv as
  select * from connection to sybase
  (select bookid, title, ytdsales
   from sales, titles
   where sales.bookid = titles.bookid and
   ytdsales > 5000);
disconnect from sybase;
quit;
endrssubmit;
```

Similar to the pass-through example for ORACLE, the CONNECT statement in the SQL procedure causes the SAS session to connect to the SYBASE database on the UNIX system using the options specified within the parentheses. The CREATE statement creates an SQL view of the data described by the SELECT statement. This SELECT statement, which joins the relevant data between the tables SALES and TITLES, is stored in the SQL view and executed by the SYBASE engine when the view is used.

After the SQL views of the DBMS tables are created on the VMS session by remotely submitting the necessary SQL procedure code, SAS code containing the DOWNLOAD procedure can be remotely submitted to the VMS session. The following code, simple as it may look, is very powerful:
Data are then copied over the network from SYBASE and ORACLE to accomplish this, the following code is executed locally on the client software to connect to the database. The procedure requests the data in the procedure executes on the system and reads the information directly from the DB2 database using the SELECT statement that was specified when the view was created. These variables are then copied over the network to the client software for presentation. The information stored in the DB2 table and presented in this bar chart was made possible by the use of three SAS/ACCESS interfaces, which were used to read and update data in SYBASE, ORACLE and DB2. The information from the warehouse is in WORK.WHOLEV and the information from the retail store is in WORK.RETAILV. The DB2 database, a data set named AllDATA is created. This data set contains the contents of both the retail and wholesale sales information. A new variable named TYPE is created in this data set to indicate whether the information came from the retail or wholesale source. This data step is executed on the local MVS session and the new data set is created.

PROC ACCESS DBMS=DB2;
   CREATE DB2ACC.BESTSELL.ACCESS;
   TABLE=BOOKS.BESTSELL;
   CREATE DB2VIEW.BESTSELL.VIEW;
   SELECT ALL;
   RUN;
PROC APPEND BASE=DB2VIEW.BESTSELL
   DATA=ALLLDATA;
   RUN;

Now that the data from the ORACLE database on VMS and the SYBASE database on UNIX have been appended to the table in the DB2 database on MVS, any of the procedures in the SAS System can be used to analyze and present the information in the DB2 table. In the following code the GCHART procedure uses the DB2 view descriptor DB2VIEW.BESTSELL to read the data from the DB2 table and create a bar chart to represent the wholesale and sales summary information for Bookworms Inc.

The next step in our application creates an access and a view descriptor of a DB2 table and updates the DB2 table by appending the data in the data set named ALLDATA to the DB2 table. In order to accomplish this, the following code is executed locally on the MVS session:

Data ALLLDATA:
   SET RETAIL(IN=RETAIL) WHOLEV(IN=W0HOLE);
   IF WHOLE THEN TYPE="WHOLESALE";
   IF RETAIL THEN TYPE="RETAIL";
   RUN;

The information stored in the DB2 table and presented in this bar chart was made possible by the use of:

• three SAS/ACCESS interfaces, which were used to read and update data in SYBASE, ORACLE and DB2

• SAS/CONNECT software, which was used to remotely submit SAS code from the MVS system to the VMS system and transfer the data over the network by using the TCP/IP access method

• the transparent distributed access feature of the SYBASE client/server software, which was used to copy the data from the SYBASE server on the UNIX node to the MVS system.
SUMMARY

This paper has shown you how SAS/ACCESS software, SAS/CONNECT software and a DBMS's distributed access software can be used as tools to integrate your computing resources and data sources. When used with the capabilities of the base SAS system, these software products can be used to give your organization the ability to easily process your current data. Establishing communications between a local SAS session and a remote SAS session on a different platform, combined with the easy access of data in a variety of database systems, gives you convenient access to your data regardless of its physical location. Since there are a variety of DBMS interfaces supported by SAS/ACCESS software, plus numerous network connections provided by SAS/CONNECT software and a number of DBMS products that support transparent distributed access, there is tremendous potential for using them together to implement distributed applications.

APPENDIX A - SAS/ACCESS Interfaces with Support for Distributed Access

There are four SAS/ACCESS interfaces running on the VMS and/or UNIX operating systems that support syntax to allow use of the transparent distributed access feature of the DBMS. In each of these interfaces, syntax supporting distributed access to a DBMS can be indicated in the ACCESS procedure, the DBLOAD procedure, and the pass-through facility of the SQL procedure. This appendix gives one example of line mode syntax of the ACCESS procedure for the SAS/ACCESS interfaces to ORACLE, SYBASE and Rdb/VMS. A DBLOAD procedure example is given for the SAS/ACCESS Interface to INGRES.

SAS/ACCESS Interface to ORACLE, Release 6.07

The following code is an example of using the ACCESS procedure in the SAS/ACCESS Interface to ORACLE. The PATH=option specifies that the TCP/IP driver is to be used to access the remote node with the physical node name VMSNODE.

```plaintext
proc access dbms=oracle;
  create oraacc.titles.access;
  path="t:vmsnode";
  user=orauser; orapw=orapass;
  table=titles;
  create oraview.titles.view;
  select all;
  run;
  proc print data=oraview.titles;
  run;
```

In order for the ACCESS and PRINT procedures to execute successfully, the runtime version of Rdb/VMS and SAS/ACCESS Interface to Rdb/VMS must be installed on the local node and a runtime version of Rdb/VMS must be installed on the remote node.


SAS/ACCESS Interface to INGRES, Release 6.07

Before using SAS/ACCESS Interface to INGRES to access data in a remote INGRES database, you must create a virtual node name using the NETU utility provided by INGRES. After this has been done, specify the virtual node name and the database name in the DATABASE= option of the DBLOAD procedure to create a table in a remote database. The following code specifies that a remote database on the virtual node named VNODE is to be used when loading an INGRES table.

```plaintext
proc dbload dbms=ingres data=mydata;
  database="vnode::ingresdatabase";
  table=titles;
  load;
  run;
```

In order for the DBLOAD procedures to execute successfully, INGRES software, INGRES/NET software, and SAS/ACCESS Interface to INGRES must be installed on the local node and INGRES and INGRES/NET must be installed on the remote node.

For more information about the DATABASE= option see SAS/ACCESS® Interface to INGRES®: Usage and Reference.

SAS/ACCESS Interface to SYBASE, Release 6.07

The following code uses the DATABASE= and SERVER= option to specify that a database on a remote node should be used to create an access and a view descriptor of a table. The server REMSERVER must be specified in the SYBASE interfaces file that was created during the SYBASE Open Client installation. The interfaces file links the server name with the network protocol to be used and the physical network node name where the server resides.

```plaintext
proc access dbms=sybase;
  database="vnode::rdbordb";
  table=titles;
  select all;
  run;
```

For more information about the DATABASE= option and about the other ORACLE drivers that can be used to access data on a remote node see SAS® Technical Report P-221, SAS/ACCESS® Software: Changes and Enhancements, Release 6.07.

SAS/ACCESS Interface to Rdb/VMS, Release 6.06 and 6.07

To use the SAS/ACCESS Interface to Rdb/VMS in a distributed environment, the DATABASE= option can be used to indicate a remote node. The following code specifies that a database on the remote node named VMSNODE is to be used when creating the access and view descriptor.
When accessing a remote database, SYBASE Open Client and SAS/ACCESS Interface to SYBASE must be installed on the local node and SYBASE SQL Server must be installed on the remote node.

REFERENCES
For more information on the SAS/ACCESS Interface to DB2, SAS/ACCESS Interface to INGRES, SAS/ACCESS Interface to Rdb/VMS, SAS/ACCESS Interface to ORACLE, the SQL procedure or SAS/CONNECT software see the appropriate manual published by SAS Institute Inc.


For more information about how the pass-through facility of the SQL procedure is used with the SAS/ACCESS engines and about the line mode syntax of the ACCESS procedure see the following technical report.


For an example of another application that uses SAS/ACCESS software with SAS/CONNECT software see the following SUGI paper.


For more information about SYBASE client/server software see the following two publications by Sybase, Inc.


ACKNOWLEDGMENTS
I would like to thank Tony Fisher for his help in designing this application, and I would also like to thank Tony Fisher, Cheryl Garner, Maureen Hayes, Mark Moorman, Howard Plemmons, Christy Rahman, Mark Stranieri, and John West for their editorial comments; most of their suggestions were incorporated into the final draft.

SAS, SAS/ACCESS, SAS/CONNECT, SAS/GRAPH, and SAS/FSP are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. DB2 is a registered trademark or trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.