Paper DM-10

To MDDB or not to MDDB - That is the question

Jeff Lessenberry, Jeff Lessenberry Consulting Group, Simpsonville, SC

ABSTRACT

The struggle to find the right presentation tool in today's global market may seem like a scene from a Shakespearian play. The MDDB has played a large role as a data discovery tool but with a changing work environment is it still the best tool for the job? The decision whether to use MDDBs has come down to some fundamental choices in how and where the client will be presenting the data. With the interface of choice becoming the internet browser, do MDDBs have a place in this new world environment or will they go the way of the horse drawn carriage? This paper will look at how MDDBs are used in a fat-client environment, their evolution to a thin-client environment, and the pros and cons of both.

INTRODUCTION

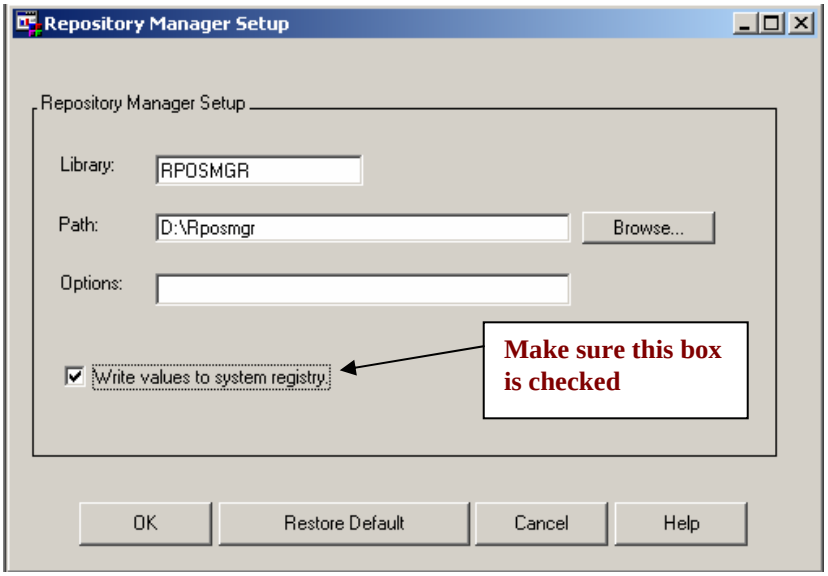
How many people have seen the exercise machine commercials on TV? In the commercial, you see these really athletically fit people using the machines. The commercial then says, "YOU can have a body like this too!" Well, how many people have bought these machines and they end up using them as a place to hang clothes? Many papers have been written for the SAS conferences about the great uses of the MDDB. I wrote a paper about changing MDDB hierarchies dynamically myself. Not unlike the unused exercise machines, MDDBs have been saddled with tasks that they were not originally designed for either. This paper will discuss the creation, the proper uses, and the pros and cons of both MDDBs and their little brother – the drillable JSP report.

CREATION OF A MDDB

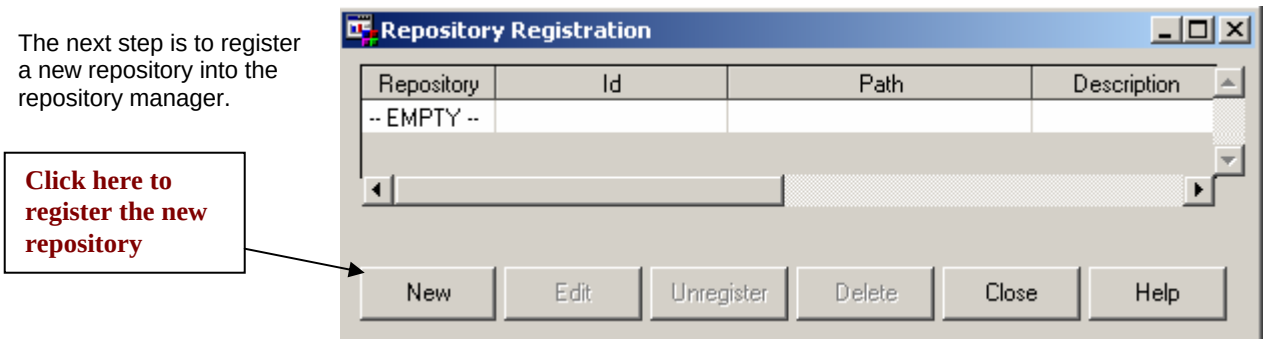
There are numerous steps to setting up a MDDB application. Some steps are common to both MDDB applications and drillable JSP applications like the setup of a web server, a tomcat server, or an IOM spawner. In this paper I will only discuss steps independently necessary for each type application.

REPOSITORY MANAGER

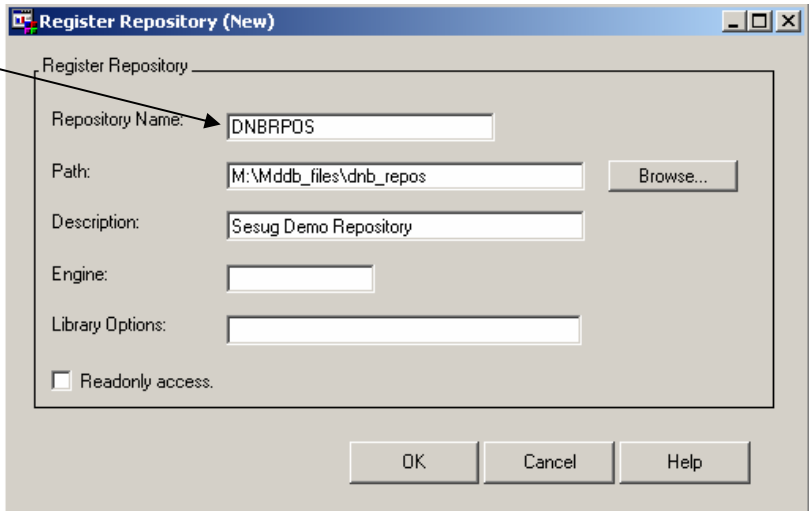
The first step is to define the location of the default repository manager in SAS (type repos). If you are using a UNIX based system as your application server, it must be defined on that system. To define the repository manager properly, you must have the ability to write the location to the system registry. Sometimes the root ID must be used to write to the system registry. It is important to write the location to the system registry because the location will then be stored as the default location for all sessions that are spawned from your application. After the repository manager has been defined you can then register new repositories.



The next step is to register a new repository into the repository manager.



Make sure the repository name is typed in ALL caps



Registering the repository is simply a matter of filling in the blanks on the form. Make sure to use all capital letters when entering the repository name because there have been synchronization issues with the repository name in the web MDDB applications if the name is not entered in the same case. The path should be where the repository will reside on the application server. Exit out of the repository manager before continuing on to the next step.

MDDB CREATION

The MDDB can be created by using PROC MDDB or by using the SAS/EIS MDDB creation screen. Since I like to be in control, I will use the procedure method. Here is the code used to create the demo MDDB:

```
proc mddb data=dnbdata.demo
          out=dnbcube.demo;

     class e_area ositesta fmsaname ositecit;

     var num_ultimates OSZSALE OBLINE emp_totaled tot_num_sites / n sum;

     hierarchy e_area ositesta fmsaname ositecit/
               name    = 'Geographic Info'
               display = YES;
run;
```

The input database name & the output MDDB name

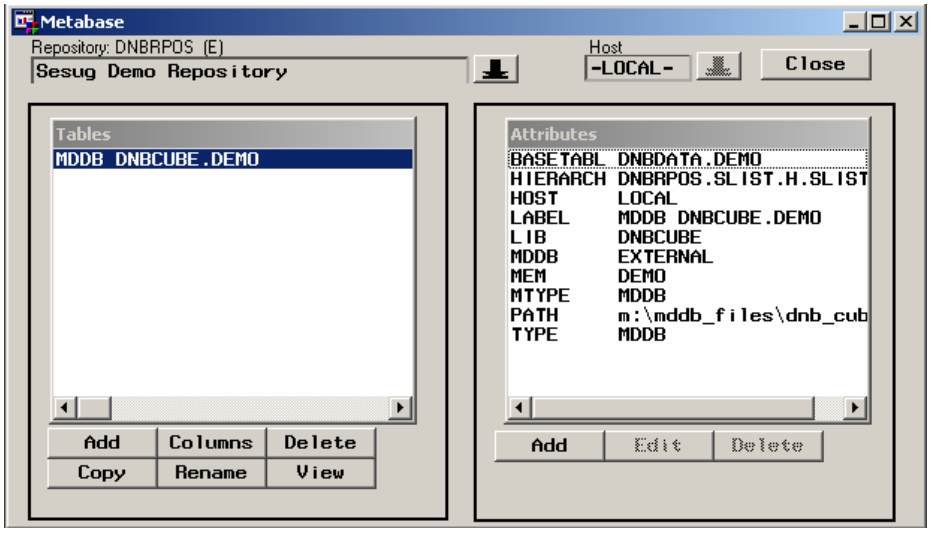
Class (drillable) variables

Analysis variables and the statistics to calculate

Drilldown hierarchy

MDDB METABASE REGISTRATION

The final step for defining the MDDB is to setup the metabase registry (type metabase.) The metabase registry is used to define the variables that are available in the MDDB for use in your application. It also defines any labels or formats that might be seen in the reports. Sometimes the labels and formats get out of sync. This can be corrected manually by updating the label and format variables in the column dataset that resides in the repository library. Here it would be: DNBRPOS.COLUMN



MDDB JSP APPLICATION

The last step is to write the JSP application that will present the MDDB to the web browser. Here is the JSP code from the demo:

```
<%-- Copyright (c) 2001 by JLCG --%>
<%@taglib uri="http://www.sas.com/taglib/sasads" prefix="sasads"%>
<html><head>
<title>Sesug Demo MDDB</title>
<link rel="stylesheet" type="text/css" href="./includes/winstyles.css">
</head>
<h3>Sesug Demo MDDB</h3>

<sasads:MDCommandProcessor id="demo_MDCP" scope="session" />

<sasads:Connection id="connection" scope="session" persistedName="iomtest"
 serverArchitecture="IOM"  initialStatement="%include 'c:\\sas8\\autoexec.sas';run;"
 accessMethod="IOM"/>

<sasads:MDModel id="demo_MDM" database="DNBCUBE.DEMO"
      connection="connection" metabase="DNBRPOS" scope="session"
      statisticLabelHidden="True">

   <sasads:MDRowAxis>
Geographic Info
   </sasads:MDRowAxis>

      <sasads:MDMeasure measure="num_ultimates" selectedStatistics="SUM" />
      <sasads:MDMeasure measure="tot_num_sites" selectedStatistics="SUM" />
      <sasads:MDMeasure measure="emp_totaled"   selectedStatistics="SUM" />
      <sasads:MDMeasure measure="OBLINE"        selectedStatistics="SUM" />
      <sasads:MDMeasure measure="OSZSALE"       selectedStatistics="SUM" />

      <sasads:MDTotal      level="e_area"       state="true"/>

</sasads:MDModel>

<sasads:MDExportToExcel id="demo_MDE" formName="export7cForm"
   formAction="sesug_demo2.jsp" render="false"/>

  <!--Create and write out the MenuBar -->
<sasads:MenuBar separator="|" menuType="SELECTOR_EXPAND" >
    <sasads:Menu label="Query" image="assets/Query.gif" alternateText="Query Selector" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="QUERY_SELECTOR" />
    </sasads:Menu>
    <sasads:Menu label="Sort" image="assets/sort.gif" alternateText="Sort Selector" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="SORT_SELECTOR" />
    </sasads:Menu>
    <sasads:Menu label="Total" image="assets/total.gif" alternateText="Totals Selector" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="TOTALS_SELECTOR" />
    </sasads:Menu>
    <sasads:Menu label="Find" image="assets/Rows.gif" alternateText="Find" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="FINDER" />
    </sasads:Menu>
    <sasads:Menu label="TopN" image="assets/TopTen.gif" alternateText="TopBottom Selector" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="TOP_BOTTOM_SELECTOR" />
    </sasads:Menu>
    <sasads:Menu label="Subset" image="assets/subset.gif" alternateText="Subset Selector" >
           <sasads:MDSelectorMenuItem model="demo_MDM" selectorType="SUBSET_SELECTOR" />
    </sasads:Menu>
    <sasads:Menu label="Export To Excel" image="assets/export.gif"
     alternateText="Export To Excel"
     customAction="javaScript: document.export7cForm.submit()" >
           <sasads:MDSelectorMenuItem selector="demo_MDE" />
    </sasads:Menu>
```

Define Tag lib

Define Command Processor for the

Start connect session with the IOM

Define the MDDB to be used and its repository

Set default hierarchy

Define analysis columns and turn totals on

Define overhead Menubar buttons

```
</sasads:MenuBar>
```

```
<!--Create and write out the MDTable -->
<sasads:MDTable id="demo_MDM2" borderWidth="1" cellSpacing="0"
   maxColumns="20" maxRows="100" cellPadding="1"
   commandProcessor="demo_MDCP" scope="session" model="demo_MDM"
   detailDataStyleSheet="sasads.css" upArrowImage="assets/up_03b.gif"
   leftArrowImage="assets/left_03b.gif"
   rightArrowImage="assets/right_03b.gif"  >

   <sasads:MDNavigationBar
   leftArrowImage="assets/left_03b.gif"
      rightArrowImage="assets/right_03b.gif"
      disabledLeftArrowImage="assets/left_03g.gif"
      disabledRightArrowImage="assets/right_03g.gif"
      doubleLeftArrowImage="assets/double_left_03b.gif"
      doubleRightArrowImage="assets/double_right_03b.gif"
      disabledDoubleLeftArrowImage="assets/double_left_03g.gif"
      disabledDoubleRightArrowImage="assets/double_right_03g.gif" />
</sasads:MDTable>

</BODY>
</HTML>
```

Define table layout parameters

Define navigation button images

That completes the task list for the web MDDB application. Here is a screenshot of the finished project:

Sesug Demo MDDB

Query | Sort | Total | Find | TopN | Subset | Export To Excel

Ultimate Area: NORTHEAST

State Code	# Ultimates	Total Number of Sites	# Employees	# Lines	Sales Volume
MA	219	28,561	1517588	37,820	$225,539,878,121
NJ	253	38,209	2658249	32,835	$537,977,684,803
NH	27	2,655	193690	1,102	$16,027,981,667
ME	10	978	51063	375	$5,409,755,448
CT	121	17,485	1286284	9,209	$333,172,759,866
RI	27	6,330	348533	2,060	$46,733,773,726
NY	554	88,416	5466978	90,090	$1,274,146,860,546
DE	36	5,036	384322	8,717	$68,848,253,431
MD	124	14,002	1123316	22,523	$97,479,206,854
PA	303	48,515	2217289	55,632	$387,066,002,495
VA	194	27,673	1394319	30,012	$221,760,858,508
VT	10	1,165	48497	1,247	$3,420,691,848
DC	48	34,171	3208618	13,599	$94,199,673,990

The state code is drillable or expandable by clicking the blue arrow

Each column's analysis value is drillable to the detail data that makes up that number

Notice the Menubar. It allows the user to redefine the display

CREATION OF A DRILLABLE JSP REPORT

The creation of a drillable JSP report is somewhat easier. Again, I will skip some steps common to both processes. Drillable JSP reports have two major steps for setup: the JSP application (which will display the report to the screen) and the SAS code (which will control the JSP drilling process).

THE DRILLABLE JSP REPORT CODE

The JSP code used to create the demo application is very simple. Here is the demo code:

```
<%-- Copyright (c) 2001 by JLCG --%>
<%@taglib uri="http://www.sas.com/taglib/sasads" prefix="sasads"%>
<html>
<head>
<sasads:Connection id="con" scope="session" persistedName="iomtest"
 serverArchitecture="IOM"  initialStatement="%include 'c:\\sas8\\autoexec.sas';run;"
 accessMethod="IOM" />

<%
    // Create a new factory
    com.sas.rmi.Rocf rocf = new com.sas.rmi.Rocf();
                session.setAttribute("rocf", rocf);

    String demo_level = request.getParameter("demo_level");
          if (demo_level == null)
           demo_level = "A";

    String demo_clickval = request.getParameter("clickval");
          if (demo_clickval == null)
           demo_clickval = "null";
%>
<title>Sesug Demo Drillable JSP</title>
<link rel="stylesheet" type="text/css" href="./includes/winstyles.css">
</head>
        <h3>Sesug Demo Drillable JSP</h3>
</head>

<body bgcolor="#ffffff" text="#000000">

<table width="100%" cellspacing="0" cellpadding="0" border="0">
  <tr> <td valign="top">
<%
    session.setAttribute("LocalHost",
                    java.net.InetAddress.getLocalHost().getHostAddress());

    test.testSocketListener socket = new test.testSocketListener();
    int port = com.sprint.gmg.common.OpenPorts.getInstance().getNextPort();
    socket.setPort( port );
    socket.setup();
    socket.start();
%>
    <%@ include file="includes/sesug_demo2a.txt"%>
<%
  socket.write(out);
  socket.close();
%>
    </td> </tr>
</table><BR><BR>
</body>
</html>
```

Define Tag lib

Start connect session with the IOM

This session variable defines the drill level

This session variable defines the drill level subset variable

This section defines and opens the socket to write the output back to the screen

Include the SAS code that creates the report

Write and close the socket

THE SAS CODE

There is a multilayered SAS macro code that defines the drilling functions. Here is the code for the demo:

```
<sasads:Submit id="submit2" connection="con" display="none">

    %let demo_level = %str(<%=demo_level%>);
    %let demo_clickval = %str(<%=demo_clickval%>);

%macro demo_levelA();
    proc summary data=dnbdata.demo nway ;
         class e_area ositesta fmsaname ositecit;
         var num_ultimates tot_num_sites emp_totaled OBLINE OSZSALE;
         output out=demo_cut1(drop=_type_ _freq_);
    run;quit;

    data demo_cut1;
           set dnbdata.demo;
           e_area_desc2 = "<A
HREF='./sesug_demo2a.jsp?demo_level=B&clickval="!!trim(left(e_area))!!"'>"!!e_area!!
"</A>";

    run;

    ods listing close;

    ODS PATH work.templat(update) sasuser.templat(read)
              sashelp.tmplmst(read);
    proc template;
         define style styles.test;
         parent=styles.default;
         style table from table/
               rules=all;
         end;
    run;

    Filename sock SOCKET '<%=session.getAttribute("LocalHost")%>:<%=port%>';

    ods html body=sock
             style=styles.test

stylesheet=(url="http://localhost:8082/gold/CLIMB/includes/sasweb.css");

    title1;title2;title3;

    proc tabulate data=demo_cut1
                  out=work.demoReport(drop=_type_ _page_ _table_) ;
         class e_area_desc2;
         var num_ultimates tot_num_sites emp_totaled OBLINE OSZSALE;
         table e_area_desc2='' ,
               num_ultimates='Number Ultimates'*sum=''*F=comma9.0
               tot_num_sites*sum=''*F=comma8.0
               emp_totaled='Number Employee'*sum=''*F=comma10.0
               OBLINE*sum=''*F=comma8.0
               OSZSALE*sum=''*F=dollar18.0
               / BOX="Summary of Areas"  misstext=' ' rts=15;
    run;

    ods html close;
%mend;
```

Pull the session variable values

Presummarize data

Create new drillable variable

Create temporary template in work directory and set rules=all for the table

Create socket filename reference

Direct output to ods html

Create output table using proc tabulate

********* I am not showing macros for all levels **********

```
%macro demo_doit();
       %if &demo_level = A %then %do;
           %demo_levelA
       %end;
       %if &demo_level = B %then %do;
           %demo_levelB
       %end;
       %if &demo_level = C %then %do;
           %demo_levelC
       %end;
       %if &demo_level = D %then %do;
           %demo_levelD
       %end;
%mend;

%demo_doit
```

Determine drilldown level and run appropriate macro

That completes the task list for the drillable JSP report application. Here is a screenshot of the finished project:

Sesug Demo Drillable JSP

Area: NORTHEAST

Summary of States	Number Ultimates	Total Number of Sites	Number Employee	# Lines	Sales Volume
CT	121	17,485	1,286,284	9,209	$333,172,759,866
DC	48	34,171	3,208,618	13,599	$94,199,673,990
DE	36	5,036	384,322	8,717	$68,848,253,431
MA	219	28,561	1,517,588	37,820	$225,539,878,121
MD	124	14,002	1,123,316	22,523	$97,479,206,854
ME	10	978	51,063	375	$5,409,755,448
NH	27	2,655	193,690	1,102	$16,027,981,667
NJ	253	38,209	2,658,249	32,835	$537,977,684,803
NY	554	88,416	5,466,978	90,090	$1,274,146,860,546
PA	303	48,515	2,217,289	55,632	$387,066,002,495
RI	27	6,330	348,533	2,060	$46,733,773,726
VA	194	27,673	1,394,319	30,012	$221,760,858,508
VT	10	1,165	48,497	1,247	$3,420,691,848

METHOD COMPARISON

MDDB setup steps	Drillable JSP Report
1. Define Repository Manager default location	1.Setup the JSP application
2. Assign any new repositories	2. Write SAS code for drilldowns
3. Create the MDDB	
4. Define the MDDB metabase	
5. Setup the MDDB JSP application	

If we were basing our decision solely on the ease of setup, the drillable JSP report would be the hands down winner. Our comparison cannot stop there; we must look at the functionality of both products. Here is a chart comparing some functionality of the current demo:

	MDDB	Drillable JSP Report
Row Drillable	**X**	**X**
Column Totals	**X**	**X**
Row Expandable	**X**	
Pull Detail Records	**X**	
Change Drill Sequence	**X**	
Sort Columns	**X**	
Top N Values	**X**	
Subset Report	**X**	
Export to Excel	**X**	

After looking at the functionality table, why would we ever decide to use anything other than MDDBs? Well, there are some things to consider.

1. Cost - To create a MDDB or use HOLAP, you must have a SAS/MDDB server license. This license is not a cheap add-on. If you do not plan to use HOLAP then you only need the license where you are creating your MDDBs.
2. Platform - As a fat-client application in EIS, MDDBs work great with large data sources, but as a thin-client application MDDBs work best with well planned summarized data sources.
3. Need – As a data discovery tool, MDDBs work great, but as a mere reporting tool it is like using a sledgehammer to hammer a nail. MDDBs are sometimes referred to as ‘cubes’ because they allow the data to be examined from multiple angles.
4. Control – When using the MDDB application, the layout and form is a little constricted, but with the drillable JSP you can use any of the SAS output tools to produce your report. You can even put the report into other formats like Microsoft Excel or PDF.

After looking at the pros and cons of both methods, I think there is a place for both. If I need to do some simple reporting then I would use the drillable JSP report. If I need to do data discovery rather than just simple reporting, I would use the MDDB method. Of course, your budget may be the last deciding factor.

CONCLUSION

The decision of which reporting method to use is not always an easy one to make. The decision should be based on the needs of the end users. Do the end users want an electronic form of a paper report or do they want to dive head first into their data with MDDBs? The decision should be made based on the user requirements and not by what looks cool to the developers.

ACKNOWLEDGMENTS

CONTACT INFORMATION

Jeff Lessenberry
Jeff Lessenberry Consulting Group
1906 Jonesville Road
Simpsonville, SC 29681-4231
864-967-4433
jlcg@charter.net