155

DB2

update

September 2005

In this issue

A itc PUBLICATION

DB2 Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for $33.75 (£22.50) each including postage.

DB2 Update on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

DB2 LUW – how to estimate the size of your database

Have you ever been asked, “How big is our database and how quickly is it growing?”? A simple question, but very time consuming to calculate. Well, did you know about the GET_DBSIZE_INFO stored procedure? It is a useful stored procedure introduced in DB2 UDB V8.2 and it calculates a database’s size and maximum capacity (you can’t use it on DPF databases). It stores this information in a local table called SYSTOOLS.STMG_DBSIZE_INFO. This table has only three columns – a timestamp, the current database size, and the maximum capacity. The command is:

```
>db2 connect to sample

>db2 call GET_DBSIZE_INFO(?,?,?,-1)
```

If you try to issue this command within 30 minutes (default) of a previous invocation of the command, the calculations will not be made again; the value from the local table will be used. I can change this default time value via the fourth input parameter to the stored procedure. If I change it to 0, the calculations will be made each time the stored procedure is invoked. If I set it to 1440, the calculations will be made only every 24 hours.

You can select from the results easily enough as follows:

```
>db2 select snapshot_timestamp, db_size, db_capacity,(db_size/
db_capacity)*100 as percentage from systools.stmg_dbsize_info
```

Both of the *db_size* and *db_capacity* values are in bytes. I put in the percentage column at the end to give me a growth value. So what do these figures actually mean? What do they include? I used a Windows 2000 Professional system running DB2 V8 FP8. I can work out the space used by my table spaces by using the **>db2 list tablespaces show detail** command and adding up the used pages values. I did this for the SAMPLE database and came up with a figure of 18.9MB

compared with a value of 20.3MB from the stored procedure. I don't think the stored procedure takes into account log space or back-up space. If I run the command daily, then over time I will get a good indication of how quickly my database is growing.

This stored procedure saves you having to write and maintain your own code to do the calculations, which has got to be a good thing. Try it out and see if it will help you.

C Leonard
Freelance Consultant (UK)

Extracting data from an LDAP directory server to a DB2 table

INTRODUCTION

Many organizations store data in the LDAP directories. The applications on mainframes, specific to these organizations, need input from this LDAP directory. To verify or retrieve data from the server for each input record creates an overhead. Because DB2 on mainframes can store huge amounts of data, we will discuss how to extract data from the LDAP directory and load it into the mainframe's dataset/DB2 tables, from where the data can be validated and queries dealt with faster, thus enhancing the performance of the application. The extraction can be performed on a weekly or bi-monthly basis, depending on the needs of the organization.

This article does not endorse any product. It talks about a solution that is already implemented in a project.

WHAT IS LDAP?

LDAP (Lightweight Directory Access Protocol) is an Internet

protocol that e-mail programs use to look up contact information on a server. It is just like a database but it's not a relational database.

Examples of LDAP directories are:

- Sun's directory server
- Microsoft's Active Directory
- Open LDAP directory server.

BUSINESS PROBLEM: TYPICAL SCENARIO FOR THIS DISCUSSION

The employee information is present on a server. There is a need to validate the employee information for all existing and/or new IDs. Because it isn't feasible to validate the data from the server each time for each of the employees, we can extract all of the data in one go to DB2 tables using the approach discussed in this article. The data can be used further to generate different business reports.

As different companies will have different needs, the example given here for a particular project can be customized and used to meet the needs of different organizations.

THE TECHNICAL SOLUTION

Data can be extracted from the directory server by writing a Java program that uses JNDI (Java Naming and Directory Interface) to connect and retrieve data from the LDAP directory server. A search can be based on some search parameters, eg user ID or country, etc. In this discussion, let us assume that country is used as the search parameter. The search result obtained is written to a file and is loaded to a DB2 table.

Alternatively, REXX could be used, but it is not recommended for bulk data retrieval because a lot of CPU time will be consumed.

Prerequisites:

- The Java SDK needs to be available on the MVS node (eg SBRYMAS2).
- Java compilation and execution occurs within the Unix System Services (USS) environment on MVS.
- This environment can be accessed by adding an OMVS (Open MVS) RACF segment to the TSO userid. This can be done using the CLAS tool – option 2 then 9, and select Unix services from the menu.
- The system administrator may need to add a few datasets to the TSO log-on procedure used to log on to TSO so that the Unix system can be accessed.

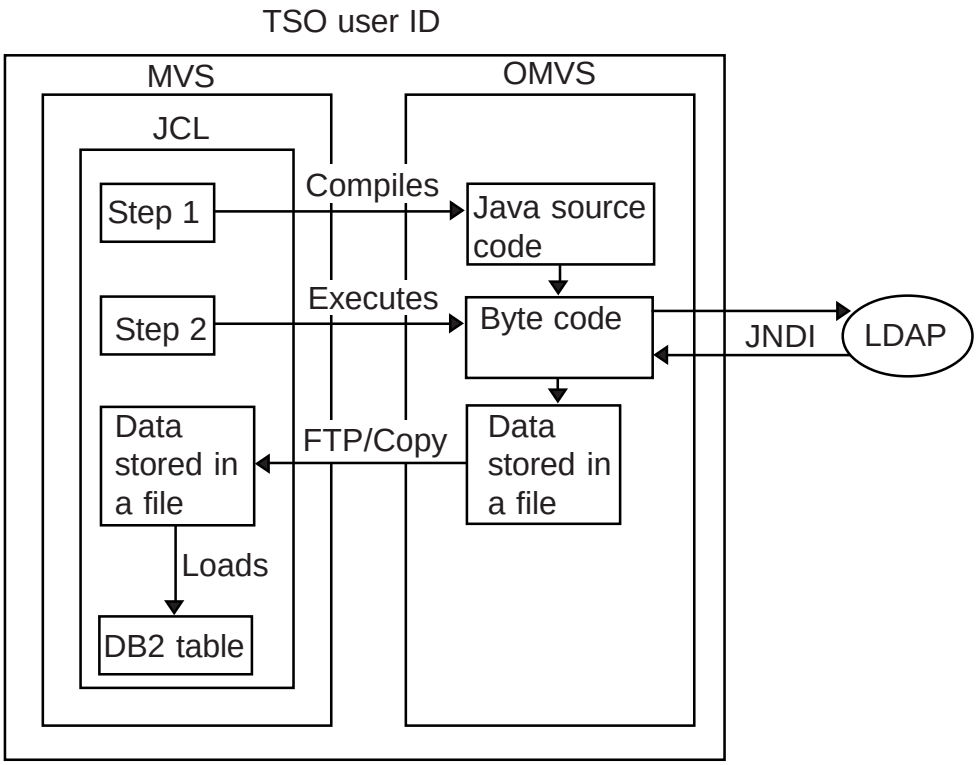

Figure 1: System flow diagram

HOW DOES IT WORK?

The process contains the following logic:

1 First, the JCL compiles the Java program on OMVS and produces the byte code. This happens only the first time. From the second run onwards this step can be skipped.

2 The second step executes the byte code generated in the first step.

3 This in turn reads the directory using JNDI for one country at a time.

4 Data retrieved in Step 2 is stored in the output file in OMVS.

5 Steps 2, 3, and 4 are performed in a loop for all of the countries, and data is appended to a single file.

6 Data is then copied (or FTPed) from OMVS to an MVS dataset.

7 This data is then loaded into a DB2 table.

8 The data can be queried from these tables as needed.

EXAMPLE OF THE TESTED CODE

Let us look at the code below, which was used in an IT project and can be customized to be used by any other organization.

Step A – Java programs (on OMVS)

The program EmpInfo.java reads the list of all the countries from the */u/abc/countrylist.txt* file (stored on the OMVS) and stores the list in an array.

From the array, it reads the countries one by one, and calls LDAPSearch.java for each country.

```
EmpInfo.java (Code present on OMVS)
public class EmpInfo{
  public static final String DIRHOME="/u/abc/";
```

```
 public static void main(String args[]){
 ArrayList arrlistCountries = null;
  String strCountryCode = null;
  FileOutputStream outputFile = null;
  OutputStreamWriter writer = null;
  BufferedWriter bufferedWriter = null;
  PrintWriter printWriter = null;
  PrintStream printStream = null;
  LDAPSearch lds = null;
  try{
                  arrlistCountries = getCountryList();
          outputFile = new FileOutputStream(DIRHOME + "LDAPExtract.dat");
                printStream = new PrintStream(outputFile);

                 if(arrlistCountries != null){

                          for(int c=0;c<arrlistCountries.size();c++){
                       strCountryCode = (String) arrlistCountries.get(c);

                           lds = new LDAPSearch();
                           lds.search(strCountryCode, printStream);

                           lds = null;
                           }
                     }
                 printStream.close();
    }catch(Exception e){
     lds = null;
     e.printStackTrace();
     System.exit(1000);
    }
 }// end of main method

 public static ArrayList getCountryList(){
  ArrayList arrlistCountries = null;
   try {
                           arrlistCountries = new ArrayList();
                           BufferedReader in = new BufferedReader(new
FileReader(DIRHOME+"countrylist.txt"));
                           String str;
                           while ((str = in.readLine()) != null) {
                            if(str != null){
                             str = str.trim();
                             arrlistCountries.add(str);
                            }
                           }
                           in.close();
    } catch (IOException e) {
     e.printStackTrace();
     System.exit(2000);
```

```
   } catch (Exception ex){
    ex.printStackTrace();
    System.exit(3000);
   }
 return arrlistCountries;
 } // end of getCountryList() method

}
```

Step B – LDAPSearch.java

1 The LDAPSearch.java program connects to the LDAP directory server using the appropriate host name.

2 The SEARCHBASE of the directory tree is constructed dynamically using the country code passed by EmpInfo.java. The SEARCHBASE identifies the starting point of the search in the directory server (eg SEARCHBASE “c=country, ou=organization unit, o=organization”, where country is passed by EmpInfo.java).

3 The search criterion (FILTER) is defined for all employees (primaryUserID=*).

4 A constraints object is created defining the type of search (SUBTREE_SCOPE). This indicates that the search will be made from the root node to the end.

5 The JNDI search is called using the searchbase, filter, and constraints, which return the directory server data for a given search base (given country).

6 All the required attributes (fields) are taken from the data retrieved and written to the output file.

Note: the code below can be customized to use multithreading as needed if the data is from many countries. This results in faster data retrieval.

```
import java.util.Hashtable;
import java.util.Enumeration;
import java.util.ArrayList;
import javax.naming.*;
import javax.naming.directory.*;
import java.io.*;
```

```
public class LDAPSearch{
 // JNDI Driver
 public String initctx = "com.sun.jndi.ldap.LdapCtxFactory";
 // Host Name or IP Address of the LDAP Directory Server :
 //                                    port number ( default is 389)
 public String host = "ldap://emprecords.mycompany.com:389";

 // dn of the tree from where one needs to start the search
 public String searchbase = "ou=emprecords,o=mycompany.com";

 // search criteria (* searches for all uids)
 public String filter = "(empUserID=*)";

 // attributes required in the search result
 public String attrs[] =
{"empUserId","firstName","sn","cn","empMailId","empPhoneNumber",
"facsimiletelephonenumber","employeeType","empLocation","co"
 };

 public Hashtable env = new Hashtable(2,1);
 public DirContext ctx = null;
 LDAPSearch(){
  try{
                env.put(Context.INITIAL_CONTEXT_FACTORY, INITCTX);
                env.put(Context.PROVIDER_URL, HOST);
                //Get reference to directory context
                ctx = new InitialDirContext(env);
  }catch(Exception e){
                e.printStackTrace();
                System.exit(6000);
  }
 }

 public void search(String strCountryCode, PrintStream printStream){
   NamingEnumeration results = null;
   SearchResult sr = null;
   String dn = null;
   Attributes ar = null;
   Attribute attr = null;
   Enumeration vals = null;
   String strSearchBase = null;
   String strSearchFilter = null;
   SearchControls constraints = null;
   StringBuffer sbAttr = new StringBuffer("");
   StringBuffer sbTempAttr = new StringBuffer("");

  try{
   constraints = new SearchControls();
    constraints.setSearchScope(SearchControls.SUBTREE_SCOPE);
   strSearchBase = "c="+ strCountryCode+","+SEARCHBASE;
```

```
 strSearchFilter = FILTER;
 // Perform search
 results = ctx.search(strSearchBase, strSearchFilter, constraints);

 while(results != null && results.hasMore()){
  if(sr != null){
   dn = sr.getName()+","+strSearchBase;
 }

  ar = ctx.getAttributes(dn,ATTRS);
  if(ar != null){
   for(int i=0; i<ATTRS.length;i++){
    attr = ar.get(ATTRS[i]);
    if(attr != null){
     for(vals = attr.getAll(); vals.hasMoreElements();){
      sbTempAttr.append(((String)vals.nextElement()).trim());
     }

              sbTempAttr = formatValue(ATTRS[i],sbTempAttr);
     sbAttr.append(sbTempAttr);
     sbTempAttr.delete(0,sbTempAttr.length());
    }
   }
  }
  printStream.println(sbAttr.toString());
  sbAttr.delete(0,sbAttr.length());
  }
}catch (Exception e){
 e.printStackTrace();
 System.exit(7000);
}
finally{
 env = null;
 try{
  ctx.close();
 }catch(Exception e){
  e.printStackTrace();
  System.exit(8000);
 }
 ctx = null;
 results = null;
 sr = null;
 dn = null;
 ar = null;
 attr = null;
 vals = null;
 strSearchBase = null;
 strSearchFilter = null;
 constraints = null;
 sbAttr = null;
```

```
    sbTempAttr = null;
   }
  }
 }
```

Step C – Countrylist.txt – input file (on OMVS)

The file */u/abc/countrylist.txt*, present on OMVS, contains the list of countries for which we require data:

```
***************************** Top of Data ******************************
AE
BH
CR
US
************************* Bottom of Data ******************************
```

Step D – JCL (on MVS)

Steps for the compilation and execution of Java code using a batch job:

1. The OMVS shell, BPXBATCH, is executed, and then the **javac** command. This will compile the Java source code and prepare the byte code on OMVS. The commands are passed as parameters to the BPXBATCH compiler.
2. The OMVS shell, BPXBATCH, is executed and then the Java command. This will execute the byte code.
3. The output is written to HFS datasets.
4. The output files on OMVS can be copied or FTPed to the datasets on MVS, from where it is loaded into the DB2 table

Note: the programs are compiled only the first time they are used. After that step 1 can be skipped unless any changes are made to the programs.

```
//*********************************************************************//
//TSTJOB1    JOB 'LDAP SEARCH',MSGCLASS=S,CLASS=S,NOTIFY=&SYSUID,
//            REGION=100M
//*********************************************************************//
//*# STEP NAME:   DELETE                                               //
//*# DESCRIPTION: DELETES THE OLD OUTPUT FILE ON MVS IF IT EXISTS AND//
```

```
//*#              DELETES THE OUTPUT AND ERROR FILES ON OMVS         //
//*#*****************************************************************//
//DELETE   EXEC PGM=IEFBR14
//SYSPRINT DD  SYSOUT=*
//DELMVSA  DD  DSN= ABC.PROD.OUTPUT.DATA,
//           DISP=(MOD,DELETE,DELETE)
//DELMVSB  DD  DSN= ABC.PROD.ERROR.DATA,
//           DISP=(MOD,DELETE,DELETE)
//STDOUTA  DD  PATH='/u/abc/java01.out',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//STDERRA  DD  PATH='/u/abc/java01.err',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//STDOUTB  DD  PATH='/u/abc/java02.out',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//STDERRB  DD  PATH='/u/abc/java02.err',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//STDOUTC  DD  PATH='/u/abc/java03.out',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//STDERRC  DD  PATH='/u/abc/java03.err',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=SIRWXU,
//             PATHDISP=(DELETE)
//*
//*********************************************************************
//*     Java Bytecode Compile for LDAPSearch
//*********************************************************************
//STEP1A   EXEC PGM=BPXBATCH,
//             PARM='sh javac LDAPSearch.java'
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDOUTA  DD  PATH='/u/abc/java01.out',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDERRA  DD  PATH='/u/abc/java01.err',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDENV   DD  DUMMY
//*
//*********************************************************************
//*     Java Bytecode Compile for EmpInfo
```

```
//*******************************************************************
//STEP1B   EXEC PGM=BPXBATCH,
//             PARM='sh javac EmpInfo.java'
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDOUTB  DD  PATH='/u/abc/java02.out',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDERRB  DD  PATH='/u/abc/java02.err',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDENV   DD  DUMMY
//*
//*******************************************************************
//*     Execute Java Bytecode
//*******************************************************************
//STEP2    EXEC PGM=BPXBATCH,REGION=0M,
//             PARM='sh java EmpInfo'
//STEPLIB  DD  DSN=SYS1.SCEERUN,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDOUTC  DD  PATH='/u/abc/java03.out',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRWXU,SIRWXG)
//STDERRC  DD  PATH='/u/abc/java03.err',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRWXU,SIRWXG)
//STDENV   DD  DUMMY
//STDIN  DUMMY
//*******************************************************************
//*     Copies the data from the Output file on OMVS to the Output file
//*     on MVS
//*******************************************************************
//COMPOUT   EXEC PGM=IKJEFT01,DYNAMNBR=300,COND=EVEN
//SYSTSPRT  DD SYSOUT=*
//HFSOUT    DD PATH='/u/abc/LDAPExtract.dat'
//HFSERR    DD PATH='/u/abc/java03.err'
//STDOUTL   DD DSN=ABC.PROD.OUTPUT.DATA,
                          DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(300,100)),
                              DCB=(RECFM=FB,LRECL=450,BLKSIZE=0)
//STDERRL   DD DSN=ABC.PROD.ERROR.DATA,
                          DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(300,100)),
                              DCB=(RECFM=FB,LRECL=450,BLKSIZE=0)
//SYSPRINT  DD SYSOUT=*
//SYSTSIN   DD DATA,DLM='/>'
 ocopy indd(HFSOUT) outdd(STDOUTL)
 ocopy indd(HFSERR) outdd(STDERRL)
/>
//*******************************************************************
//* STEP LOAD0: Loads table with the data in output dataset.
```

```
//* The Load contl card used in this JCL can be written separately as
//* per the Output file structure
//*********************************************************************
//LOADØ  EXEC DSNUPROC,SYSTEM=DSNB,
//             LIB='DBDCSUP.DB2.SE1DSNB.SDSNLOAD',
//             UID=''
//DSNUPROC.SYSRECØØ DD DSN= ABC.PROD.OUTPUT.DATA,
//             DISP=SHR
//DSNUPROC.SYSIN DD DSN=ABC.DSNB.CNTL.UNLØ,
//             DISP=SHR
//DSNUPROC.SYSUT1 DD DSN=ABC.TEST.SYSUT1.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SORTOUT DD DSN=ABC.TEST.SORTOU.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SORTWKØ1 DD DSN=ABC.TEST.SORTWKØ1.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SORTWKØ2 DD DSN=ABC.TEST.SORTWKØ2.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SORTWKØ3 DD DSN=ABC.TEST.SORTWKØ3.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SORTWKØ4 DD DSN=ABC.TEST.SORTWKØ4.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//DSNUPROC.SYSMAP DD DSN=ABC.TEST.SYSMAP.LSØ,
//             DISP=(MOD,DELETE,CATLG),
//             SPACE=(8192,(279489,139745),RLSE),
//             UNIT=SYSDA
//*********************************************************************
//*                      End of the JCL
//********************** Bottom of Data ******************************
```

Benefits of this solution are:

1 LDAP is not relational, so extracting data into DB2 has many advantages for the simple reason that the data can be made relational.

2 This is not difficult code and doesn't use any unusual

software. It requires a knowledge of common software languages like Java, JCL, and DB2.

3 The data extracted from LDAP, once in DB2, can be used further for many downstream applications.

4 To access data in LDAP you need a connection every time, resulting in heavy network traffic, which is costly to any company's business. With the data in DB2 tables, you make just one connection for multiple queries. As an example, the company in this discussion had to make 12 connections to access data in as many countries.

5 Having data in DB2 on a mainframe is of special importance because of the size and volume of data it can store.

6 This approach is used for bulk data retrieval with minimal network traffic.

Drawbacks:

1 It is not useful for daily extraction.

2 For weekly runs we have the old data available rather than current data.

CONCLUSION

Having data on an LDAP server is very common nowadays in many organizations. What we gain with this solution is the ability to extract all the data to a DB2 table in one go, manipulate the data for downstream applications, and query more efficiently. Other advantages are discussed earlier in this article. Readers can customize this code to suit their business needs.

Tapasya Puri (tapasyap@in.ibm.com)
Software Engineer
IBM Global Services (India)

DB2 Web services – part 2

This month we conclude the code for creating Web services.

```
          <TD height="25"><INPUT type="text" name="author" size="30"
                  maxlength="30" value="Nikola Lazovic"></TD>
        </TR>
        <TR>
          <TD height="35"><INPUT type="submit" name="Submit"
                 value="Execute"><INPUT type="reset" value="Reset"></TD>
        </TR>
      </TBODY>
     </TABLE>
</FORM>
<HR width="100%">
<br>Generate<a href="sample/List.dadx/WSDL"> WSDL</a>
<br>Generate<a href="sample/List.dadx/XSD"> XSD</a>
</body>
</html>
```

web.xml

```
<! ********************************************************** >
<! DB2 as a Web Service provider - servlet                    >
<!    Folder: C:\WORFSamp\WEB-INF                             >
<! ********************************************************** >
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>SampGroup</servlet-name>
    <servlet-class>com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker</
servlet-
class>
    <init-param >
       <param-name>faultListener</param-name>
       <param-value>org.apache.soap.server.DOMFaultListener</param-
value>
    </init-param>
    <load-on-startup>-1</load-on-startup>
  </servlet>
  <servlet-mapping>
     <servlet-name>SampGroup</servlet-name>
     <url-pattern>/sample/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
```

```
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

group.properties

```
# ************************************************************* >
# DB2 as a Web Service provider - /dadx SampGroup group properties >
#    Folder: C:\WORFSamp\WEB-INF\classes\groups\SampGroup         >
# ************************************************************* >
dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:sample
userID=db2admin
password=db2admin
autoReload=true
reloadIntervalSeconds=5
```

List.dadx

```
<! ******************************************************** >
<! DB2 as a Web Service provider - Web service definition   >
<!    Folder: C:\WORFSamp\WEB-INF\classes\groups\SampGroup  >
<! ******************************************************** >
<?xml version="1.0" encoding="US-ASCII"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<result_set_metadata name="ListAuthor" rowName="Author">
  <column name="JOURNAL" type="CHAR" nullable="false"/>
  <column name="JOURNAL_NO" type="SMALLINT" nullable="false"/>
  <column name="JOURNAL_DATE" type="DATE" nullable="false"/>
  <column name="TITLE" type="CHAR" nullable="false"/>
  <column name="ABSTRACT" type="VARCHAR" nullable="false"/>
</result_set_metadata>

<operation name="ListAuthor">
  <documentation> Lists the contributor's articles in the XEPHON table
of the
SAMPLE database </documentation>
  <call>
    <SQL_call>CALL DB2ADMIN.listauthor(:author)</SQL_call>
    <parameter name="author" type="xsd:string" kind="in"/>
    <result_set name="AuthorXephon" metadata="ListAuthor"/>
  </call>
</operation>

<operation name="listXephon">
  <documentation> Lists all the published articles in the XEPHON table
```

```
of the
SAMPLE database </documentation>
  <query>
    <SQL_query>SELECT * FROM DB2ADMIN.XEPHON</SQL_query>
  </query>
</operation>

<operation name="deleteXephon">
  <documentation> Deletes a published article from XEPHON table of the
SAMPLE
 database </documentation>
  <update>
    <SQL_update>delete from DB2ADMIN.XEPHON
                where  JOURNAL    = :journal and
                       JOURNAL_NO = :journalno and
                       AUTHOR     = :author
    </SQL_update>
    <parameter name="journal" type="xsd:string" kind="in"/>
    <parameter name="journalno" type="xsd:int" kind="in"/>
    <parameter name="author" type="xsd:string" kind="in"/>
  </update>
</operation>

<operation name="insertXephon">
  <documentation> Inserts a new published article into the XEPHON table
of the
SAMPLE database </documentation>
  <update>
    <SQL_update>insert into DB2ADMIN.XEPHON
        values(:journal,:journalno,:journaldate,:author,:title,:abstract)
    </SQL_update>
    <parameter name="journal" type="xsd:string" kind="in"/>
    <parameter name="journalno" type="xsd:int" kind="in"/>
    <parameter name="journaldate" type="xsd:date" kind="in"/>
    <parameter name="author" type="xsd:string" kind="in"/>
    <parameter name="title" type="xsd:string" kind="in"/>
    <parameter name="abstract" type="xsd:string" kind="in"/>
  </update>
</operation>

<operation name="updateXephon">
  <documentation> Updates a abstract in the XEPHON table of the SAMPLE
database
</documentation>
  <update>
    <SQL_update>update DB2ADMIN.XEPHON
                set    ABSTRACT   = :abstract
                where  JOURNAL    = :journal and
                       JOURNAL_NO = :journalno and
                       AUTHOR     = :author
```

```
    </SQL_update>
    <parameter name="abstract" type="xsd:string" kind="in"/>
    <parameter name="journal" type="xsd:string" kind="in"/>
    <parameter name="journalno" type="xsd:int" kind="in"/>
    <parameter name="author" type="xsd:string" kind="in"/>
  </update>
</operation>
</DADX>
```

Consumer.sql

This file can be invoked from the command window with the following commands:

- db2 connect to sample
- db2 -tdf Consumer.sql
- db2 connect reset.

```
DROP FUNCTION DB2ADMIN.ListContributor;
COMMIT;
CREATE FUNCTION DB2ADMIN.ListContributor (author VARCHAR(30))
RETURNS TABLE (Contributor VARCHAR(800))
LANGUAGE SQL CONTAINS SQL
EXTERNAL ACTION NOT DETERMINISTIC
RETURN
Select * from Table (
    db2xml.extractVarchars(
      db2xml.xmlclob(
       db2xml.soaphttpc('http://localhost:9080/SampleWorf/sample/
List.dadx/SOAP',
          'http://tempuri.org/sample/List.dadx',
          '<m:ListAuthor xmlns:m="http://tempuri.org/sample/List.dadx"
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/">
        <author xsi:type="xsd:string">' || author || '</author> </
m:ListAuthor>'
      ) ), '//Author/*')
) as X;

DROP FUNCTION DB2ADMIN.BabelFish;
COMMIT;
CREATE FUNCTION DB2ADMIN.BabelFish (translationmode VARCHAR(5),
                                    sourcedata VARCHAR(150))
  RETURNS VARCHAR(150) SPECIFIC BabelFish
  LANGUAGE SQL READS SQL DATA
  EXTERNAL ACTION NOT DETERMINISTIC
  RETURN
```

```
    db2xml.extractVarchar(
    db2xml.xmlclob(
     db2xml.soaphttpv ('http://services.xmethods.net:80/perl/
soaplite.cgi',
        'urn:xmethodsBabelFish#BabelFish',
        varchar('<tran:BabelFish xmlns:tran="urn:xmethodsBabelFish"
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/">
           <translationmode xsi:type="xsd:string">' || translationmode
||
          '</translationmode>
          <sourcedata xsi:type="xsd:string">' || sourcedata || '</
sourcedata>
        </tran:BabelFish>') ) ), '//return/');
consumer.html
<! ********************************************************** >
<! DB2 as a Web Service consumer - welcome page               >
<!    Folder: WebSphere Application Folder                    >
<!    Ver 6:  'C:\Program Files\IBM\WebSphere\AppServer\      >
<!             profiles\default\installedApps\                >
<!             localhostNode01Cell\SampleWorf_war.ear\        >
<!             SampleWorf.war'                                >
<! ********************************************************** >
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<META name="GENERATOR" content="IBM WebSphere Studio Homepage Builder
V6.0.0
Trial for Windows">
<META http-equiv="Content-Style-Type" content="text/css">
<title> DB2 Web Services </title>
</head>
<body>
<H1 align="center">DB2 as a Web service consumer</H1>
<HR width="100%">
<FORM name="myform" method="post" action="consumer.jsp">
<CENTER>
<TABLE>
  <TBODY>
    <TR>
      <TD align="center"><FONT color="#000099"><B>Call UDF
         &quot;ListContributor&quot;</B></FONT><BR>
      <BR>
          (1. enter a name of a contributor and
click the
      <B>Execute</B> button<BR>
                 2. on
      <B>consumer.jsp</B> select article to translate in chosen
language<BR>
```

```
               with
BabelFish Web
      service translator)<BR>
      <BR>
      </TD>
    </TR>
    <TR>
      <TD height="25" align="center"><INPUT type="text" name="author"
size="30"
maxlength="30" value="Nikola Lazovic"></TD>
    </TR>
    <TR>
      <TD height="35" align="center"><INPUT type="submit" name="Submit"
value="Execute">          &nbs
p;            
<INPUT type="reset" value="Reset"></TD>
    </TR>
  </TBODY>
</TABLE>
</CENTER>
</FORM>
<HR width="100%">
</body>
</html>
```

consumer.jsp

```
<! ********************************************************************* >
<! DB2 as a Web Service consumer - when provider is a DB2 Web service  >
<!    Folder: WebSphere Application Folder                             >
<! Ver 6:  'C:\Program Files\IBM\WebSphere\AppServer\profiles\default\ >
<!          installedApps\localhostNode01Cell\SampleWorf_war.ear\      >
<!          SampleWorf.war'                                            >
<! ********************************************************************* >
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ page errorPage="error.jsp"
         import="java.sql.Connection,
                 java.sql.DriverManager,
                 java.sql.ResultSet,
                 java.sql.ResultSetMetaData,
                 java.sql.Statement"
%>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>Published articles</TITLE>
<!--Styles-->
<STYLE TYPE="text/css">
<!--
H1 {
```

```
      text-align: center;
}
-->
</STYLE>
<SCRIPT language="JavaScript">
function showSite(Private, cnt) {
  var site = "babelfish.jsp";
  var selInd = Private.selectedIndex;
  var srct = document.forms[0].elements["txt" + cnt].value;
  if (selInd > 0)  {
    site = site + "?trmod=en_" + Private.options[selInd].value +
"&srct=" + srct
+ "&trglng=" + Private.options[selInd].text
    window.open(site, "_new");
  }
  return;
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<H1>Contributor's published Xephon articles</H1>
<BR><BR>
<TABLE border="1">
<TBODY>
      <H1>
      <TR>
            <TH>Journal</TH>
            <TH>Journal No</TH>
            <TH>Journal_date</TH>
            <TH>Title</TH>
            <TH>Abstract</TH>
             <TH>Language <FONT color="#000099">(BabelFish)</FONT></TH>
      </TR>
      </H1>
<%
      String auth;
      auth = request.getParameter("author");
      Connection cx;
Statement st = null;
ResultSet rs = null;
      int counter;
int cnt;
      try {
  Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
        try {
          cx = DriverManager.getConnection("jdbc:db2:SAMPLE", "db2admin",
"db2admin");
          try {
```

```
      st = cx.createStatement();
            rs = st.executeQuery("SELECT * " +
          "FROM table (DB2ADMIN.ListContributor('" + auth + "')) as X");
            counter = 0;
      cnt = 0;
            while (rs.next()) {
              counter++;
              if (counter == 1) out.println("<TR align='left'>");
              if (counter == 5) {
                out.println("<TH><TEXTAREA rows='5' cols='50'
name='txt"+cnt+"'>"+rs.getString(1)+"</TEXTAREA></TH>");
                out.println("<TH><select name='lng"+cnt+"'
onchange='showSite(lng"+cnt+", "+cnt+")' tabindex='1' class='button'>");
                out.println("<option value=' '> </option>");
                out.println("<option value='fr'>French</option>");
                out.println("<option value='de'>German</option>");
                out.println("<option value='it'>Italian</option>");
                out.println("<option value='es'>Spanish</option>");
                out.println("</select></TH>");
                out.println("</TR>");
                cnt++;
                counter = 0;
          } else out.println("<TH>"+rs.getString(1)+"</TH>");
            }
          } catch (Exception ex) {
   System.err.println("Error in executeQuery statement : '" + ex + "'");
    }
          cx.close();
  } catch (Exception connectionException) {
          System.out.println ("Error in conecting: '" +
connectionException + "'");
        }
      } catch (Exception forNameException) {
  System.out.println("Error in forName: '" + forNameException + "'");
      }
%>
</TBODY>
</TABLE>
</FORM>
</BODY>
</HTML>
```

babelfish.jsp

```
<! ********************************************************************* >
<!DB2 as Web Service consumer - when provider is not a DB2 Web service >
<!    Folder: WebSphere Application Folder                              >
<! Ver 6:  'C:\Program Files\IBM\WebSphere\AppServer\profiles\default\ >
<!          installedApps\localhostNode01Cell\SampleWorf_war.ear\      >
```

```
<!          SampleWorf.war'                                                    >
<! ********************************************************************** >
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.Ø1 Transitional//EN">
<HTML>
<HEAD>
<%@ page errorPage="error.jsp"
         import="java.sql.Connection,
                 java.sql.DriverManager,
                 java.sql.ResultSet,
                 java.sql.ResultSetMetaData,
                 java.sql.Statement"
%>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>BabelFish Translator</TITLE>
<!--Styles-->
<STYLE TYPE="text/css">
<!--
H1 {
       text-align: center;
}
-->
</STYLE>
</HEAD>
<BODY>
<FORM>
<H1>BabelFish Translator</H1>
<BR><BR>
<TABLE border="1">
<TBODY>
       <H1>
       <TR>
  <TH width="5Ø%">English</TH>
         <TH width="5Ø%" style='color:blue'><%=
request.getParameter("trglng") %></TH>
       </TR>
       </H1>
       <TR>
<%
       String trm;
       String srctxt;
       trm = request.getParameter("trmod");
srctxt = request.getParameter("srct");
       out.println("<TH width='5Ø%' align='left'>"+srctxt+"</TH>");
       Connection cx;
Statement st = null;
ResultSet rs = null;
       try {
          Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
          try {
           cx = DriverManager.getConnection("jdbc:db2:SAMPLE", "db2admin",
```

```
"db2admin");
          try {
          st = cx.createStatement();
      rs = st.executeQuery("VALUES DB2ADMIN.BabelFish('" + trm + "', '"
+ srctxt
+ "')");
            while (rs.next()) {
         out.println("<TH width='50%' align='left'
style='background:#CCFFFF'>"+rs.getString(1)+"</TH>");
            }
          } catch (Exception ex) {
   System.err.println("Error in executeQuery statement : '" + ex + "'");
    }
          cx.close();
        } catch (Exception connectionException) {
        System.out.println ("Error in conecting: '" +
connectionException + "'");
        }
      } catch (Exception forNameException) {
     System.out.println("Error in forName: '" + forNameException + "'");
      }
%>
</TR>
</TBODY>
</TABLE>
</FORM>
</BODY>
</HTML>
```

error.jsp

```
<! ********************************************************************** >
<! DB2 as a Web Service consumer - error page for jsp                     >
<!    Folder: WebSphere Application Folder                               >
<! Ver 6:  'C:\Program Files\IBM\WebSphere\AppServer\profiles\default\ >
<!          installedApps\localhostNode01Cell\SampleWorf_war.ear\      >
<!          SampleWorf.war'                                            >
<! ********************************************************************** >
<%@page isErrorPage="true" contentType="text/html"
%>
<html>
<head><title>Error</title></head>
<body bgcolor="RED">
<h1> Unfortunately, we are unable to fulfill your request.</h1>
<hr/>
<h2> A more detailed error message has been recorded to
assist in uncovering the problem condition. Please try back later.
</h2>
<%
application.log(exception.getMessage()) ;
```

```
%>
<hr/>
</body>
</html>
```

BabelFishService.wsdl

```
<! ********************************************************************* >
<! Optional: only for documentation purpose (for building DB2 UDF)     >
<! ------------------------------------------------------------------- >
<! AltaVista's BabelFish web service is currently disabled, therefore  >
<! all applications dependent of it are rendered useless at the moment.>
<! (only temporarily or permanent we will se)                          >
<! ------------------------------------------------------------------- >
<! To see BabelFish in action:                                         >
<!   1. change text in some 'abstract' field in consumer.jsp form, with>
<!      for example word Good and                                      >
<!   2. choose appropriate language from combo box                     >
<! *********************************************************************>
<?xml version="1.0"?>
<definitions name="BabelFishService"
  xmlns:tns="http://www.xmethods.net/sd/BabelFishService.wsdl"
targetNamespace="http://www.xmethods.net/sd/BabelFishService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
      <message name="BabelFishRequest">
            <part name="translationmode" type="xsd:string"/>
            <part name="sourcedata" type="xsd:string"/>
      </message>
      <message name="BabelFishResponse">
            <part name="return" type="xsd:string"/>
      </message>
      <portType name="BabelFishPortType">
            <operation name="BabelFish">
                  <input message="tns:BabelFishRequest" />
                  <output message="tns:BabelFishResponse" />
            </operation>
      </portType>
      <binding name="BabelFishBinding" type="tns:BabelFishPortType">
            <soap:binding style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="BabelFish">
                  <soap:operation
soapAction="urn:xmethodsBabelFish#BabelFish"/>
                  <input>
                        <soap:body use="encoded"
namespace="urn:xmethodsBabelFish"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
                  </input>
                  <output>
                        <soap:body use="encoded"
namespace="urn:xmethodsBabelFish"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                  </output>
            </operation>
      </binding>
      <service name="BabelFishService">
 <documentation>Translates text of up to 5k in length, between a variety
of languages.</documentation>
            <port name="BabelFishPort" binding="tns:BabelFishBinding">
                  <soap:address
  location="http://services.xmethods.net:80/perl/soaplite.cgi"/>
            </port>
      </service>
</definitions>
```

Nikola Lazovic
DB2 System Administrator
Postal Savings Bank (Serbia and Montenegro)

DB2 LUW – how to copy a database

This article looks at what the **db2look** command gives me and whether I need anything else (like **db2cfexp**) in order to get a total picture of my set-up. There is a very good help section (>**db2look -help**), which describes all the options. One of the main uses of **db2look** is to copy a database from one system to another (along with **db2move**). Let's look at some examples. The commands executed below were carried out on a Windows 2000 Professional system running DB2 UDB V8.1 FP8 using the SAMPLE database and the system administrator userid (db2admin).

To get the DDL for a database I would use:

```
>db2look -d sample -e -o hm01.txt
```

The **-e** option extracts the DDL statements for all tables in the

database (you can use the **-z** option to restrict the output for a particular schema name) to an output file specified by the **-o** parameter.

The above gives you just the DDL and no dbm or db configuration settings.

To get just the db and dbm configuration settings use the **fd** flag:

```
>db2look -d sample -fd -o hm02.txt
```

(If you use the **-f** option instead of the **-fd** option, you don't get the db2fopt lines – as shown below:

```
!db2fopt SAMPLE update opt_buffpage 250;
!db2fopt SAMPLE update opt_sortheap 256;
```

These allow you to mimic a production system on a test box where you might not have the memory capacity of the production box).

The help screen says that the **-f** option should also give you a list of the registry values that have been set. When I issued the command

```
>db2look -d sample -f -o hm04.txt
```

all I got was the db cfg and dbm cfg settings, not the registry settings. I could always get the registry settings by using the **db2set** command.

So what does the **db2cfexp** command give me over **db2look**? Well, this command exports configuration information that is not available using **db2look**. I can use this to take the configuration settings from one machine to another (at this other machine I would use **db2cfimp** to import the configuration file generated by **db2cfexp**). Both the **db2cfexp** and **db2cfimp** commands have good help pages.

One thing that I found **db2cfexp** gave me was the registry settings, which are in the second output block titled [REGISTRY_LOCAL]. The command to use is:

```
>db2cfexp hm06.txt
```

The **-l** option gives me the table space, buffer pool, and partition group information, so, if I wanted a total snapshot of my system, I would issue the following commands:

```
>db2look -d sample -fd -e -l -o hm05.txt
```

and:

```
>db2cfexp hm06.txt
```

If I wanted to move the data I would use the **db2move** command after the **db2look** command.

If I were rolling out database connection information then I would use just **db2cfexp** and **db2cfimp**.

To get just the DDL for a single table (say the employee table in the sample database) I would use:

```
>db2look -d sample -e -tw employee -o ddl_for_employee.txt
```

One point to note is that if I combine the flags, the output is generated in LaTeX format:

```
>db2look -d sample -fdle -o hm05.txt
— USER is:
— Use LaTeX format
— Output is sent to file: hm05.txt
```

Use the following commands to format the files:

```
latex hm05.txt
dvips hm05.dvi > hm05.ps
```

Where do I get the **LaTeX** and **dvips** commands from? Check out the Web (Google is a good place to start) – you can either buy products or some are available as shareware. I haven't really used them, because all I want to do is get the DDL from one system and run it on another.

There are a lot of options for the **db2look** command and there are flags for most things that you want to do, so, before writing your own SQL or scripts, check out the help pages – they may save you a lot of work!

C Leonard
Freelance Consultant (UK)

SQL analyser utility

DB2 Update, issue 151, May 2005 published an article entitled ‘Guidelines for coding efficient SQL’. Following on from that article, I have written a REXX program that helps you analyse embedded SQL in multiple programs in a single parse and gives guidelines to improve them. This could be used for white box testing, as a first step to find out which SQL to tune in your application programs. This tool could be very useful during migration scenarios (it was used successfully during migration from IMS to DB2). Migration tools don’t produce optimized SQL. Doing an Explain on every SQL will be very time consuming in large applications where there could be thousands of embedded SQL. So you could run this program and pick SQL from the output of the program, which will be a smaller subset. The program currently handles only SELECT statements but can be further modified easily to include other DML statements and check all the guidelines mentioned in ‘Guidelines for coding efficient SQL’.

The biggest factor in the performance of a business application is the speed of the SQL. The SQL having the greatest impact on the load on a system and the productivity of its end users can usually be improved to a large degree. The most common solution to an SQL tuning problem is a prescription for some combination of changes to the database and, more often, changes to the SQL itself. These changes allow the formerly slow statement to run faster, with no changes in functionality and no change in the calling application, except in the SQL itself. This common solution is especially attractive because it is usually simple and it rarely has unintended side effects. The SQL analyser utility helps you analyse hundreds of embedded SQL in multiple programs in a single parse and gives guidelines to improve them.

```
/****************************** REXX **********************************/
/*                          SQL ANALYSER UTILITY                      */
/* This Utility scans COBOL-DB2-CICS programs and analyses embedded   */
```

```
/* SQL. It collects various statistics and then prescribes changes to */
/* the SQL based on guidelines. This is useful to improve performance,*/
/* correctness, readability, & maintainability of the SQL.            */
/*                                                                    */
/*      INPUT: DATASET   ( DSORG : PS , RECFM : FB ).                 */
/*    Give input DATASET which contains COBOL programs with embedded  */
/*             SQLs.                                                  */
/*      OUTPUT: DATASET   ( DSORG : PS , RECFM : FB )                 */
/*    In the output dataset a member is created for each program with */
/*     details.                                                       */
/**********************************************************************/
CALL PDS_PARSER
EXIT
/**********************************************************************/
/* This function picks each member of the input PDS and passes it to  */
/* the SQL_REPORT_GENERATOR function to extract embedded SQL.         */
/**********************************************************************/
 PDS_PARSER:

   /* GIVE SOURCE PDS NAME IN INDSNAME                          /
      GIVE WHERE OUTPUT NEEDS TO BE PLACED IN OUTDSNAME      */
   INDSNAME ='SSGLAX.REXX.SQL.SELECT.INPUT'
   OUTDSNAME='SSGLAX.REXX.SQL.SELECT.OUTPUT'
   ADDRESS ISPEXEC
   'LMINIT DATAID(INPDSN) DATASET('"'"INDSNAME"'"') ENQ(SHR) ORG(DSO)'
   IF RC = 8 THEN
   DO
      SAY 'INVALID DATASET NAME ' || INDSNAME
      EXIT
   END
   ELSE IF RC <> Ø THEN
   DO                              /* RETURN CODES                  */
      SAY 'INCORRECT LMINIT -FAILED WITH RC ' || RC
      EXIT
   END
   IF DSO <> 'PO' THEN
   DO
      SAY ' DATASET ' || INDSNAME || ' NOT A PDS '
      EXIT
   END
   ADDRESS ISPEXEC
   'LMOPEN DATAID(&INPDSN) OPTION(INPUT)'
   IF RC <> Ø THEN
   DO                              /* RETURN CODES                  */
      SAY 'INCORRECT LMOPEN -FAILED WITH RC ' || RC
      EXIT
   END
   DO UNTIL LMRC <> Ø
     ADDRESS ISPEXEC
        'LMMLIST DATAID(&INPDSN) OPTION(LIST) MEMBER(MEMVAR) STATS(YES)'
```

```
        IF RC > 8 THEN
        DO                              /* RETURN CODES             */
          SAY 'INCORRECT LMMLIST - FAILED WITH RC ' || RC
          EXIT
        END
      LMRC=RC
      IF RC=Ø THEN
      DO
       PROGRAM = STRIP(MEMVAR,'B')
       INPDS = INDSNAME||'('||PROGRAM||')' /* STRIP TO REMOVE */
                                            /* TRAILING BLANKS */
       OUTPDS = OUTDSNAME||'('||PROGRAM||')' /* STRIP TO REMOVE */
       CALL SQL_REPORT_GENERATOR
      END
   END /* DO UNTIL */
 RETURN
/*********************************************************************/
/* This function scans each member (program) and extracts the SQL    */
/* from it. It stores the SQL in STRING_SQL array. If it finds a     */
/* SELECT query then it calls FUNC_SELECT to handle it. This program */
/* handles only the SELECT case. It can be extended to handle other  */
/* DML statements.                                                   */
/*********************************************************************/
SQL_REPORT_GENERATOR:
   ADDRESS TSO
   "PROFILE NOPREFIX"
   "ALLOC DA("INPDS") F(INFILE) SHR REUSE"
   "EXECIO * DISKR INFILE (FINIS STEM IN."
   WRITE_FLAG=Ø
   COUNT = Ø
   CALL INITIALIZE
   NOL_PROG=IN.Ø /* NOL = NO OF LINES */
     DO I=1 TO NOL_PROG
      X = LEFT(IN.I,72)
      PARSE VALUE X WITH COB_NUM  X
      /* THIS REMOVES THE COBOL NUMBERING */
      FIRST_WORD = WORD(X,1);SECOND_WORD = WORD(X,2)
      IF FIRST_WORD='EXEC' & SECOND_WORD='SQL' THEN
      DO
        CALL EXTRACT_SQL
      END
      ELSE ITERATE
      DO J=1 TO COUNT_E
        NOW_STRING_SQL=WORDS(STRING_SQL.J) /* NOW = NUM OF WORDS */
        DO K=1 TO NOW_STRING_SQL
          SQL_COMMAND = WORD(STRING_SQL.J,K)
          SELECT
            WHEN SQL_COMMAND='SELECT' THEN
            DO
               WRITE_FLAG=1
```

```
               CALL FUNC_SELECT
               CALL WRITE2STRING
             END
             /*
             WHEN SQL_COMMAND='UPDATE' THEN CALL FUNC_UPDATE
             WHEN SQL_COMMAND='INSERT' THEN CALL FUNC_INSERT
             WHEN SQL_COMMAND='DELETE' THEN CALL FUNC_DELETE
             */
             OTHERWISE  /* DO NOTHIN */
           END /* SELECT */
         END /* K */
       END /* J */
       CALL INITIALIZE
       END/* I */
     CALL WRITE_TO_FILE
     "FREE DD (INFILE)"
RETURN /* END OF SQL_REPORT_GENERATOR */
/*********************************************************************/
/* This function extracts all the details of the SELECT SQL .        */
/*********************************************************************/
 FUNC_SELECT:
     J=1 ;
     SQL_TYPE = 'SELECT'
     FLAG_INTO=WORD_FOUND('INTO')
     FLAG_WHERE=WORD_FOUND('WHERE')
     FLAG_GROUP_BY=WORD_FOUND('GROUP BY')
     FLAG_HAVING = WORD_FOUND('HAVING')
     FLAG_ORDER_BY=WORD_FOUND('ORDER BY')
     FLAG_SELECT_ALL=WORD_FOUND('SELECT *')
     FLAG_DISTINCT=WORD_FOUND('DISTINCT')
     FLAG_LIKE=WORD_FOUND('LIKE')
     FLAG_NOT_IN=WORD_FOUND('NOT IN')
     FLAG_SUBSTR=WORD_FOUND('SUBSTR')
     FLAG_CURSOR=WORD_FOUND('CURSOR')
     IF FLAG_CURSOR=1 THEN
     DO
       /* EXTRACT(FROM_STRING,TO_STRING,I,NO OF LINES) */
       EXTRACT_STRING =  EXTRACT('DECLARE','FOR',J,COUNT_E)
     END
     IF FLAG_INTO=Ø THEN
     DO
       EXTRACT_STRING =  EXTRACT('SELECT','FROM',J,COUNT_E)
       CALL STRIP_COLUMNS EXTRACT_STRING,'SELECT'
     END
     ELSE
     DO
       EXTRACT_STRING =  EXTRACT('SELECT','INTO',J,COUNT_E)
       CALL STRIP_COLUMNS EXTRACT_STRING,'SELECT'
       EXTRACT_STRING =  EXTRACT('INTO','FROM',J,COUNT_E)
     END
```

```
      EXTRACT_STRING = EXTRACT('FROM','WHERE',J,COUNT_E)
      CALL HANDLE_CASE EXTRACT_STRING,'TABLE_S'
      IF FLAG_GROUP_BY=1 THEN
      DO
        EXTRACT_STRING = EXTRACT('WHERE','GROUP BY',J,COUNT_E)
        CALL HANDLE_CASE EXTRACT_STRING,'NUMBER'
        IF FLAG_HAVING=1 THEN
         EXTRACT_STRING = EXTRACT('GROUP BY','HAVING',J,COUNT_E)
        ELSE
        DO
          IF FLAG_ORDER_BY=1 THEN
             EXTRACT_STRING = EXTRACT('GROUP BY','ORDER BY',J,COUNT_E)
          ELSE
             EXTRACT_STRING = EXTRACT('GROUP BY','NULL',J,COUNT_E)
        END
        CALL STRIP_COLUMNS EXTRACT_STRING,'GROUP BY'
      END
      IF FLAG_HAVING=1 THEN
      DO
        IF FLAG_ORDER_BY=1 THEN
           EXTRACT_STRING = EXTRACT('HAVING','ORDER BY',J,COUNT_E)
        ELSE
           EXTRACT_STRING = EXTRACT('HAVING','NULL',J,COUNT_E)
        CALL STRIP_COLUMNS EXTRACT_STRING,'HAVING'
        CALL HANDLE_CASE EXTRACT_STRING,'NUMBER'
      END
      IF FLAG_ORDER_BY=1 THEN
      DO
        IF FLAG_GROUP_BY=Ø THEN
        DO
           EXTRACT_STRING = EXTRACT('WHERE','ORDER BY',J,COUNT_E)
           CALL HANDLE_CASE EXTRACT_STRING,'NUMBER'
        END
        EXTRACT_STRING = EXTRACT('ORDER BY','NULL',J,COUNT_E)
        CALL STRIP_COLUMNS EXTRACT_STRING,'ORDER BY'
      END
      IF FLAG_GROUP_BY=Ø & FLAG_ORDER_BY=Ø THEN
      DO
        EXTRACT_STRING = EXTRACT('WHERE','NULL',J,COUNT_E)
        CALL HANDLE_CASE EXTRACT_STRING,'NUMBER'
      END
   RETURN
/*********************************************************************/
/* This function searches for a string in the extracted SQL and      */
/* returns a 1 when the word is found and Ø otherwise                */
/*********************************************************************/
   WORD_FOUND:
   ARG SEARCH_STRING
         DO H=1 TO COUNT_E
           NOOFWORDS1 = WORDS(STRING_SQL.H)
```

```
            PARSE VALUE STRING_SQL.H WITH TEMP1 (SEARCH_STRING) TEMP2
            NOOFWORDS2 = WORDS(TEMP1) + WORDS(TEMP2)
            IF NOOFWORDS1<>NOOFWORDS2 THEN RETURN 1
          END
    RETURN Ø
/*********************************************************************/
/* This function extracts a string between FROM_STRING and TO_STRING. */
/* Result are stored in EXTRACT_STRING.                               */
/*********************************************************************/
  EXTRACT:
  ARG FROM_STRING,TO_STRING,J,COUNT_E
    /* WE ARE PARSIN STRING_SQL.<COUNT> WHERE <COUNT>=J. */
    /* 1<= <COUNT> <= COUNT_E */
    PARSE VALUE STRING_SQL.J WITH (FROM_STRING) S1
    EXTRACT_STRING=''
    FLAG_END=Ø;FLAG1=Ø
    /* NULL IS USED FOR THE LAST TERM LIKE IN ORDER-BY TO NULL */
    IF TO_STRING='NULL' THEN FLAG_END=1
    DO A = J TO COUNT_E UNTIL FLAG1=1
      X = LEFT(STRING_SQL.A,72)
      IF A = J THEN X = S1 /*FOR THE FIRST LINE.OVERWRITES PREVIOUS */
      NOOFWORDS1 = WORDS(X) /* VALUE OF X */
      TEMP3=''
      PARSE VALUE X WITH TEMP4 (TO_STRING) TEMP5
      NOOFWORDS2 = WORDS(TEMP4) + WORDS(TEMP5)
         /* IF TO_WORD FOUND */
         IF NOOFWORDS1<>NOOFWORDS2 & FLAG_END=Ø THEN /*FLAG_END  */
         DO                              /*EXTRACT TILL END OR NOT*/
            FLAG1=1
            IF TEMP4='' THEN LEAVE /* IF THE TO_STRING APPEARS */
            /* AS THE FIRST WORD OF THE SENTENCE THEN JUST LEAVE */
             /* IF BOTH "FROM" & "TO" LIES IN THE SAME SENTENCE */
            IF A=J THEN PARSE VALUE X WITH TEMP3 (TO_STRING) TEMP5
            ELSE PARSE VALUE X WITH TEMP3 (TO_STRING) TEMP5
            EXTRACT_STRING=EXTRACT_STRING||TEMP3
            LEAVE
         END /*IF NOOFWORDS<> .. */
      IF FLAG1=Ø THEN EXTRACT_STRING=EXTRACT_STRING||X
    END /*DO UNTIL*/
    J = A
  RETURN EXTRACT_STRING /*EXTRACT PROGRAM END*/
/*********************************************************************/
/* This function does the following:                                 */
/* Finds number of comparison and logical operators used in the query */
/* using function STRIP_STR.                                         */
/* Finds Tables accessed.                                            */
/*********************************************************************/
  HANDLE_CASE:
  ARG EXTRACT_STRING,CASE
    IF CASE='TABLE_S' THEN /* EXTRACT THE TABLES ACCESSED */
```

```
DO
  FLAG_HANDLE=Ø;STRING_HANDLE='';
  DO UNTIL FLAG_HANDLE=1
    PARSE VALUE EXTRACT_STRING WITH S1 ',' TEMP_STRING
    /*CASE: TABLE1 T1, TABLE2 T2 ... */
    PARSE VALUE S1 WITH S12 S11
    COUNT_TABLE = COUNT_TABLE + 1
    TABLE.COUNT_TABLE = S12
    IF TEMP_STRING='' THEN FLAG_HANDLE=1
    EXTRACT_STRING = TEMP_STRING
  END
END /*CASE TABLE */
IF CASE='NUMBER' THEN
DO
   HANDLE_STRING = EXTRACT_STRING
   FLAG_OR=Ø;
   COUNT_OPER.6=Ø;COUNT_OPER.7=Ø;
   VISIT_1_U=Ø
   DO H = 1 TO 7
    COUNT_OPER.H=Ø
   END
   DO UNTIL FLAG_OR=1
     PARSE VALUE HANDLE_STRING WITH TEMP1 'OR' TEMP2
     IF TEMP2<>'' THEN COUNT_OPER.6=COUNT_OPER.6+1
     ELSE FLAG_OR=1
     HANDLE_STRING = TEMP2
     FLAG_AND=Ø
     IF TEMP1<>'' THEN
     DO
       DO UNTIL FLAG_AND=1
         PARSE VALUE TEMP1 WITH TEMP3 'AND' TEMP4
          IF TEMP4<>'' THEN
          DO
           COUNT_OPER.7=COUNT_OPER.7+1
           CALL STRIP_STR TEMP3
          END /* TEMP4 */
          ELSE
          DO
           FLAG_AND=1;
           CALL STRIP_STR TEMP1
          END
          TEMP1 = TEMP4
       END /*UNTIL FLAG_AND */
     END /*TEMP1*/
   END /* UNTIL FLAG_OR */
   /* COUNT_OPER.I WILL HAVE COUNT OF NUM OF COMPARATORS */
   /* 1:'<=' 2:'>=' 3:'=' 4:'<' 5:'>' 6:'OR' 7:'AND'     */
   /* COUNT_U GIVES NUMBER OF UNIQUE WHERE CLAUSE        */
   /* STRING_U.I GIVES UNIUE COLUMNS NAMES               */
NO_OF_LE  = COUNT_OPER.1
```

```
    NO_OF_GE  = COUNT_OPER.2
    NO_OF_E   = COUNT_OPER.3
    NO_OF_L   = COUNT_OPER.4
    NO_OF_G   = COUNT_OPER.5
    NO_OF_OR  = COUNT_OPER.6
    NO_OF_AND = COUNT_OPER.7
    NO_OF_DISTINCT = COUNT_U
    NO_OF_OP = NO_OF_AND + NO_OF_OR + 1
    END /*CASE NUMBER */
  RETURN /*HANDLE CASE FUNC*/
/*********************************************************************/
/* This function counts comparison operators used in the query.      */
/*********************************************************************/
STRIP_STR:
ARG TEMP_SS
  PARSE VALUE TEMP_SS WITH A '<=' B
  IF B<>'' THEN
    DO
    CALL UNIQUE '<=',A
    COUNT_OPER.1 = COUNT_OPER.1 + 1
    RETURN
    END
  PARSE VALUE TEMP_SS WITH A '>=' B
  IF B<>'' THEN
    DO
    CALL UNIQUE '>=',A
    COUNT_OPER.2 = COUNT_OPER.2 + 1
    RETURN
    END
  PARSE VALUE TEMP_SS WITH A '=' B
  IF B<>'' THEN
    DO
    CALL UNIQUE '=',A
    COUNT_OPER.3 = COUNT_OPER.3 + 1
    RETURN
    END
  PARSE VALUE TEMP_SS WITH A '<' B
  IF B<>'' THEN
    DO
    CALL UNIQUE '<',A
    COUNT_OPER.4 = COUNT_OPER.4 + 1
    RETURN
    END
  PARSE VALUE TEMP_SS WITH A '>' B
  IF B<>'' THEN
    DO
    CALL UNIQUE '>',A
    COUNT_OPER.5 = COUNT_OPER.5 + 1
    RETURN
    END
```

```
 RETURN
/*************************************************************************/
/* This function finds unique columns used in the where clause.          */
/*************************************************************************/
  UNIQUE:
  ARG OPER_U,STR_U
    IF VISIT_1_U = Ø THEN /* FIRST VISIT */
    DO
     VISIT_1_U=1
     COUNT_U = 1
     STRING_U.COUNT_U = STRIP(STR_U,'B')
    END
    ELSE
    DO
     FLAG_NEW=Ø
     DO H = 1 TO COUNT_U
      IF STRING_U.H = STR_U THEN
      DO
       FLAG_NEW=1
       LEAVE
      END
     END
     IF FLAG_NEW = Ø THEN
     DO
       COUNT_U = COUNT_U + 1
       STRING_U.COUNT_U = STRIP(STR_U,'B')
     END
    END
  RETURN
/*************************************************************************/
/* This function finds the following:                                    */
/* Columns accessed in the query.                                        */
/* Columns in Group By.                                                  */
/* Columns in Order By.                                                  */
/*************************************************************************/
  STRIP_COLUMNS:
  ARG EXTRACT_STRING,CASE
        IF EXTRACT_STRING = '*' THEN NO_OF_COLUMNS='ALL'
        ELSE
        DO
            FLAG_HANDLE=Ø;COUNT_HANDLE=Ø;TEMP_COLUMN_FUNC=Ø
            TEMP_SCALAR_FUNC=Ø
            IF FLAG_DISTINCT=1 & CASE = 'SELECT' THEN ,
            PARSE VALUE EXTRACT_STRING WITH,
            'DISTINCT' EXTRACT_STRING
            DO UNTIL FLAG_HANDLE=1
             PARSE VALUE EXTRACT_STRING WITH TEMP_COLUMN ',' TEMP_STRING
              IF CASE = 'SELECT' THEN
              DO
                   IF WORDS(TEMP_COLUMN) > 1 THEN
```

```
         PARSE VALUE TEMP_COLUMN WITH TEMP_COLUMN 'AS' REST
       /* COLUMN_FUNC FUNCTIONS */
       PARSE VALUE TEMP_COLUMN WITH TEMP1 '(' REST
       IF REST<>'' THEN
       DO
        IF CHECK_FUNCTION_TYPE(TEMP_COLUMN) = 'C' THEN
        DO
          TEMP_COLUMN_FUNC = TEMP_COLUMN_FUNC + 1
          COLUMN_FUNCTION.TEMP_COLUMN_FUNC = ,
                             STRIP(TEMP_COLUMN,'B')
          FLAG_COLUMN_FUNC=1
        END
        ELSE
        DO
          TEMP_SCALAR_FUNC = TEMP_SCALAR_FUNC + 1
          SCALAR_FUNCTION.TEMP_SCALAR_FUNC = ,
                             STRIP(TEMP_COLUMN,'B')
          FLAG_SCALAR_FUNC=1
        END
        IF TEMP_STRING='' THEN FLAG_HANDLE=1
       END
       ELSE
       DO
          /* TABLES */
          IF TEMP_STRING<>'' THEN
          DO
            COUNT_HANDLE=COUNT_HANDLE+1
            COLUMN.COUNT_HANDLE = TEMP_COLUMN
          END
          ELSE
          DO
            COUNT_HANDLE=COUNT_HANDLE+1
            COLUMN.COUNT_HANDLE = TEMP_COLUMN
            FLAG_HANDLE=1
          END
       END
    END /* IF CASE = 'SELECT'  */
    ELSE
    DO
       IF TEMP_STRING<>'' THEN
       DO
         COUNT_HANDLE=COUNT_HANDLE+1
         COLUMN.COUNT_HANDLE = TEMP_COLUMN
       END
       ELSE
       DO
         COUNT_HANDLE=COUNT_HANDLE+1
         COLUMN.COUNT_HANDLE = TEMP_COLUMN
         FLAG_HANDLE=1
       END
```

```
            END
            EXTRACT_STRING = TEMP_STRING
          END
        END /* EXTRACT_STRING = *  */
        IF CASE='SELECT' THEN
        DO
          COUNT_COLUMN_FUNC = TEMP_COLUMN_FUNC
          COUNT_SCALAR_FUNC = TEMP_SCALAR_FUNC
          NO_OF_COLUMNS = COUNT_HANDLE
          DO H = 1 TO NO_OF_COLUMNS
             COLUMN_OF_TABLES.H = STRIP(COLUMN.H,'B')
             COLUMN_FUNCTION.H = STRIP(COLUMN_FUNCTION.H,'B')
          END
        END
        IF CASE= 'ORDER BY' THEN
        DO
          NO_OF_ORDER_BY = COUNT_HANDLE
          DO H = 1 TO NO_OF_ORDER_BY
             COLS_N_ORDER_BY.H = STRIP(COLUMN.H,'B')
          END
        END
        IF CASE= 'GROUP BY' THEN
        DO
          NO_OF_GROUP_BY = COUNT_HANDLE
          DO H = 1 TO NO_OF_GROUP_BY
             COLS_N_GROUP_BY.H = STRIP(COLUMN.H,'B')
          END
        END
    RETURN
/*********************************************************************/
/* This function accepts a function name as the input argument and   */
/* returns 'C' if the function is a Columns function and 'S' if it   */
/* is a Scalar function.                                             */
/*********************************************************************/
   CHECK_FUNCTION_TYPE:
   ARG TEMP_FUNC_CFT
    PARSE VALUE TEMP_FUNC_CFT WITH FUNC_CFT '(' REST
    SELECT
      WHEN FUNC_CFT ='AVG'   THEN  RETURN 'C'
      WHEN FUNC_CFT ='COUNT' THEN  RETURN 'C'
      WHEN FUNC_CFT ='MAX'   THEN  RETURN 'C'
      WHEN FUNC_CFT ='MIN'   THEN  RETURN 'C'
      WHEN FUNC_CFT ='SUM'   THEN  RETURN 'C'
      OTHERWISE  RETURN 'S'
    END /* SELECT */
/*********************************************************************/
/* This function initializes variables and flags.                    */
/*********************************************************************/
   INITIALIZE:
      TABLE.=' '
```

```
      COLUMN_FUNCTION. =' '
      SCALAR_FUNCTION. =' '
      SQL_TYPE=' '
      NO_OF_COLUMNS=' '
      NO_OF_DISTINCT=' '
      NO_OF_AND=Ø
      NO_OF_OR=Ø
      NO_OF_G=Ø
      NO_OF_GE=Ø
      NO_OF_E=Ø
      NO_OF_LE=Ø
      NO_OF_L=Ø
      NO_OF_OP=Ø
      FLAG_INTO=Ø
      FLAG_WHERE=Ø
      FLAG_DISTINCT=Ø
      FLAG_LIKE=Ø
      FLAG_HAVING=Ø
      FLAG_LIKE=Ø
      FLAG_SELECT_ALL=Ø
      FLAG_NOT_IN = Ø
      FLAG_GROUP_BY=Ø
      FLAG_ORDER_BY=Ø
      FLAG_COLUMN_FUNC=Ø
      FLAG_SCALAR_FUNC=Ø
      NO_OF_ORDER_BY = Ø
      NO_OF_GROUP_BY = Ø
      COUNT_TABLE = Ø
      COUNT_U = Ø
   RETURN  /* INITALIZE */
/*********************************************************************/
/* This function extracts an SQL string between EXEC SQL and END EXEC */
/* and stores it in STRING_SQL array.                                  */
/*********************************************************************/
  EXTRACT_SQL:
    FLAG_E=Ø;COUNT_E=Ø;SAME_LINE=Ø
    PARSE VALUE X WITH EXEC SQL REST /* TO HANDLE CASE */
    IF REST='' THEN                  /*EXEC SQL <DDL COMMAND> .... */
     M=I+1
    ELSE
     M=I;
    DO UNTIL FLAG_E=1
     Y=LEFT(IN.M,72)
     PARSE VALUE Y WITH COB_NUM Y /*REMOVES THE COBOL NUMBERING */
     FIRST_WORD = WORD(Y,1)
     IF FIRST_WORD <> 'END-EXEC' & FIRST_WORD <> 'END-EXEC.' THEN
     DO
      COUNT_E=COUNT_E+1;
      STRING_SQL.COUNT_E=Y
      M=M+1;
```

```
     END
     ELSE FLAG_E=1
    END /*UNTIL */
   RETURN
/*********************************************************************/
/* This function writes the SQL statistics and guidelines into the   */
/* STRING array                                                      */
/*********************************************************************/
   WRITE2STRING:
      COUNT=COUNT+1;STRING.COUNT= '  '
      COUNT=COUNT+1;STRING.COUNT= '********************************',
       '********************************'
      COUNT=COUNT+1;STRING.COUNT= '  '
      COUNT=COUNT+1;STRING.COUNT= 'PROGRAM NAME   : 'PROGRAM
      COUNT=COUNT+1;STRING.COUNT='    '
      COUNT=COUNT+1;STRING.COUNT= 'SQL'
      COUNT=COUNT+1;STRING.COUNT= '--'
      COUNT=COUNT+1;STRING.COUNT='  '
      DO H = 1 TO COUNT_E
       COUNT=COUNT+1;STRING.COUNT= STRING_SQL.H
      END
      COUNT=COUNT+1;STRING.COUNT='  '
      COUNT=COUNT+1;STRING.COUNT='  '
      IF FLAG_ORDER_BY=1 THEN
      DO
         COUNT=COUNT+1;STRING.COUNT= 'COLUMNS IN ORDER BY'
         COUNT=COUNT+1;STRING.COUNT= '--------------------'
         DO H = 1 TO NO_OF_ORDER_BY
            COUNT=COUNT+1;STRING.COUNT= COLS_N_ORDER_BY.H
         END
         COUNT=COUNT+1;STRING.COUNT='  '
      END
      IF FLAG_SCALAR_FUNC=1 THEN
      DO
         COUNT=COUNT+1;STRING.COUNT= '  SCALAR FUNCTION   '
         COUNT=COUNT+1;STRING.COUNT= '--------------------'
         DO H = 1 TO COUNT_SCALAR_FUNC
            COUNT=COUNT+1;STRING.COUNT= SCALAR_FUNCTION.H
         END
         COUNT=COUNT+1;STRING.COUNT='  '
      END
      IF FLAG_COLUMN_FUNC=1 THEN
      DO
         COUNT=COUNT+1;STRING.COUNT= '  COLUMN FUNCTION   '
         COUNT=COUNT+1;STRING.COUNT= '--------------------'
         DO H = 1 TO COUNT_COLUMN_FUNC
            COUNT=COUNT+1;STRING.COUNT= COLUMN_FUNCTION.H
         END
         COUNT=COUNT+1;STRING.COUNT='  '
      END
```

```
IF FLAG_GROUP_BY=1 THEN
DO
   COUNT=COUNT+1;STRING.COUNT= 'COLUMNS IN GROUP BY'
   COUNT=COUNT+1;STRING.COUNT= '--------------------'
   DO H = 1 TO NO_OF_GROUP_BY
      COUNT=COUNT+1;STRING.COUNT= COLS_N_GROUP_BY.H
   END
   COUNT=COUNT+1;STRING.COUNT='  '
END
COUNT=COUNT+1;STRING.COUNT= 'COLUMNS IN UNIQUE WHERE'
COUNT=COUNT+1;STRING.COUNT= '----------------------'
DO H = 1 TO COUNT_U
   COUNT=COUNT+1;STRING.COUNT= STRING_U.H
END
COUNT=COUNT+1;STRING.COUNT='  '
IF FLAG_SELECT_ALL=Ø THEN
DO
   COUNT=COUNT+1;STRING.COUNT= 'TABLES ACCESSED'
   COUNT=COUNT+1;STRING.COUNT= '--------------'
   DO H = 1 TO COUNT_TABLE
      COUNT=COUNT+1;STRING.COUNT= TABLE.H
   END
   COUNT=COUNT+1;STRING.COUNT='  '
END
COUNT=COUNT+1;STRING.COUNT= 'SQL_TYPE       : 'SQL_TYPE
COUNT=COUNT+1;STRING.COUNT= 'NO OF COLUMNS  : 'NO_OF_COLUMNS
COUNT=COUNT+1;STRING.COUNT= 'NO OF OPERATOR : 'NO_OF_OP
COUNT=COUNT+1;STRING.COUNT= 'NO OF DISTINCT : 'NO_OF_DISTINCT
COUNT=COUNT+1;STRING.COUNT= 'NO OF AND      : 'NO_OF_AND
COUNT=COUNT+1;STRING.COUNT= 'NO OF OR       : 'NO_OF_OR
COUNT=COUNT+1;STRING.COUNT= 'NO OF >=       : 'NO_OF_GE
COUNT=COUNT+1;STRING.COUNT= 'NO OF <=       : 'NO_OF_LE
COUNT=COUNT+1;STRING.COUNT= 'NO OF =        : 'NO_OF_E
COUNT=COUNT+1;STRING.COUNT= 'NO OF >        : 'NO_OF_G
COUNT=COUNT+1;STRING.COUNT= 'NO OF <        : 'NO_OF_L
COUNT=COUNT+1;
IF FLAG_DISTINCT=1 THEN
   STRING.COUNT= 'DISTINCT       : YES'
ELSE
   STRING.COUNT= 'DISTINCT       : NO'
COUNT=COUNT+1;
IF FLAG_LIKE=1 THEN
   STRING.COUNT= 'LIKE           : YES'
ELSE
   STRING.COUNT= 'LIKE           : NO'
COUNT=COUNT+1;
IF FLAG_SELECT_ALL=1 THEN
   STRING.COUNT= 'SELECT *       : YES'
ELSE
   STRING.COUNT= 'SELECT *       : NO'
```

```
COUNT=COUNT+1;
IF FLAG_NOT_IN=1 THEN
   STRING.COUNT= 'NOT IN         : YES'
ELSE
   STRING.COUNT= 'NOT IN         : NO'
COUNT=COUNT+1;STRING.COUNT='    '
COUNT=COUNT+1;STRING.COUNT='GUIDELINES'
COUNT=COUNT+1;STRING.COUNT='----------'
IF FLAG_SELECT_ALL=1 THEN
DO
 COUNT=COUNT+1;STRING.COUNT=     ,
'1) SELECT * IS USED IN THE SQL. SQL SHOULD ALWAYS LIST THE NAMED'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'COLUMNS TO BE RETURNED TO THE PROGRAM. SELECT * SHOULD NEVER'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'BE USED.'
END
IF FLAG_WHERE=Ø THEN
DO
 COUNT=COUNT+1;STRING.COUNT=     ,
 '2) LIMIT THE DATA SELECTED IN SQL. RETURN THE MINIMUM NO OF',
 'COLUMNS'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'AND ROWS NEEDED BY YOUR APPLICATION PROGRAM BY MAKING EFFICIENT'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'USE OF THE WHERE.'
END
IF FLAG_DISTINCT=1 THEN
DO
 COUNT=COUNT+1;STRING.COUNT=     ,
 '7) AVOID USING DISTINCT.IF DUPLICATES WILL NOT CAUSE A',
 'PROBLEM,DO'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'NOT CODE DISTINCT.'
END
IF FLAG_DISTINCT=1 ³ FLAG_GROUP_BY=1 ³ FLAG_ORDER_BY=1 THEN
DO
 COUNT=COUNT+1;STRING.COUNT=     ,
 '1Ø) TRY TO SORT ONLY ON INDEXED COLUMNS. WHEN USING ORDER',
 'BY GROUP'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'BY, DISTINCT. IT IS BEST TO USE ONLY USE ONLY INDEXED COLUMNS.'
END
IF FLAG_SCALAR_FUNC=1 THEN
DO
 COUNT=COUNT+1;STRING.COUNT=     ,
 '13) AVOID SQL SCALAR FUNCTIONS IF YOU HAVE ALTERNATIVE METHODS.'
 COUNT=COUNT+1;STRING.COUNT=     ,
 'AVOID THE USE OF THE SQL SCALAR FUNCTIONS USED FOR DATA TYPE '
 COUNT=COUNT+1;STRING.COUNT=     ,
```

```
        'CONVERSIONS CHARACTER STRING MANIPULATION, AND DATE/TIME'
        COUNT=COUNT+1;STRING.COUNT=     ,
        'CONVERSIONS'
       END
       IF FLAG_LIKE=1 THEN
       DO
        COUNT=COUNT+1;STRING.COUNT=     ,
        '22) USE IN INSTEAD OF LIKE. IF YOU KNOW THAT ONLY A CERTAIN OF'
        COUNT=COUNT+1;STRING.COUNT=     ,
        'OCCURRENCES EXIST,USING IN WITH THE SPECIFIC LIST IS MORE'
        COUNT=COUNT+1;STRING.COUNT=     ,
        'EFFICIENT THAN USING LIKE.'
       END
       IF FLAG_NOT_IN=1 THEN
       DO
        COUNT=COUNT+1;STRING.COUNT=     ,
        '23) AVOID USING NOT (EXCEPT WITH EXISTS). NOT SHOULD ONLY BE',
        'USED AS'
        COUNT=COUNT+1;STRING.COUNT=     ,
        'AN ALTERNATIVE TO VERY COMPLEX PREDICATES.'
       END
       IF FLAG_COLUMN_FUNC=1 THEN
       DO
        IF CHECK_COLUMN_IN_GROUP_BY() = 1 THEN
        DO
         COUNT=COUNT+1;STRING.COUNT=     ,
         '45) ALL COLUMNS NOT IN COLUMN FUNCTIONS MUST BE INCLUDED IN',
         'THE'
         COUNT=COUNT+1;STRING.COUNT=     ,
         'GROUP BY CLAUSE.'
        END
       END
       COUNT=COUNT+1;STRING.COUNT='    '
       COUNT=COUNT+1;STRING.COUNT='    '
   RETURN /* WRITE2STRING */
/*********************************************************************/
/* This function returns Ø if all columns not in column function are  */
/* included in GROUP BY else it returns Ø                             */
/*********************************************************************/
  CHECK_COLUMN_IN_GROUP_BY:
    IF NO_OF_GROUP_BY <> NO_OF_COLUMNS THEN RETURN 1
    DO H = 1 TO NO_OF_GROUP_BY
     TEMP_FLAG = Ø
     DO K = 1 TO NO_OF_COLUMNS
       IF COLS_N_GROUP_BY.H = COLUMN_OF_TABLES.K THEN
       DO
        TEMP_FLAG=1
        LEAVE
       END
     END /* K */
```

```
        IF TEMP_FLAG = Ø THEN RETURN 1
     END /* H */
  RETURN Ø
/*********************************************************************/
/* This function writes the STRING array into the output file.       */
/*********************************************************************/
  WRITE_TO_FILE:
     IF WRITE_FLAG=1 THEN
     DO
      /* WRITE INTO FILE */
      "ALLOC DA("OUTPDS") F(OUTFILE) SHR REUSE"
      "EXECIO * DISKW OUTFILE (FINIS STEM STRING."
      "FREE DD (OUTFILE)"
     END
     CALL INITIALIZE_WRITE
   RETURN
/*********************************************************************/
/* This function initializes the STRING array.                       */
/*********************************************************************/
   INITIALIZE_WRITE:
      STRING.  = ''
   RETURN
```

Editor's note: we will publish a sample input program and sample output from the analyser program in the next issue.

T S Laxminarayan (ts_laxminarayan@yahoo.com)
Systems Programmer
Tata Consultancy Services Ltd (India)

DB2 LUW – the DB2 ping command

Did you know that DB2 has its own **ping** command? You can use it to test the response time from a server to a client. One of the features of this command on V8 is that you can specify the size of any outbound or return packets, or you can just take a default value of 10 bytes. I tested the following from a Windows 2000 Professional client running DB2 UDB V8.2 connecting to DB2 8.1 on AIX. The command in its simplest form is:

```
>db2 connect to <client>

>db2 ping <server>
```

My windows client has the SAMPLE database on it and my Unix server has a database called HMDB on it. To test the connection between the two I connected to the Windows client and issued the command below (remember I have to be connected to a database to issue the **ping** command).

```
>db2 connect to sample
>db2 ping hmdb 5 times

Elapsed time: 14229 microseconds
Elapsed time: 34 microseconds
Elapsed time: 13 microseconds
Elapsed time: 13 microseconds
Elapsed time: 14 microseconds

>db2 ping hmdb 5 times

Elapsed time: 212 microseconds
Elapsed time: 24 microseconds
Elapsed time: 13 microseconds
Elapsed time: 15 microseconds
Elapsed time: 14 microseconds
```

You can see that the first elapsed time value is a lot higher than subsequent values – therefore I would always use the **times** option. For the subsequent values, I could check whether there was a great variation in the connection times, which might point to network issues (I might have to increase the number of repetitions from 5 to 20).

I could also then try with a sending packet size of 1000 bytes and a response packet size of 10,000 bytes. The command would look like this:

```
>db2 ping hmdb 5 times request 1000 response 10000

Elapsed time: 10782 microseconds
Elapsed time: 10276 microseconds
Elapsed time: 10359 microseconds
Elapsed time: 9939 microseconds
Elapsed time: 10417 microseconds
```

As you can see, the response times are longer than when we

didn’t specify a packet size, and the individual times are all close together – indicating that I don’t have any network problems.

Suppose I had a z/OS system as my server? That’s not an issue – as long as the database is catalogued I can **ping** it.

This is another command to put into your armoury for when people complain about slow response times and network problems.

C Leonard
Freelance Consultant (UK)

PolarLake has announced the availability of free Proof-Of-Concept (POC) programs for IBM customers wishing to utilize PolarLake's Enterprise Service Bus product alongside their existing IBM technology investment.

The POC program includes access to all relevant PolarLake products – including the Integration Suite – and five days of on-site professional services. During this time PolarLake can design, build, and create real applications, solving integration challenges and managing the orchestration and mediation of software services – without code. Because PolarLake works seamlessly alongside the existing IBM product stack – including DB2 UDB, WebSphere MQ, and WebSphere Application Server – IBM customers are able to deploy these applications within their own environments in order to confirm the suitability of the ESB approach to their own specific requirements.

For further information contact:
URL: www.polarlake.com/en/html/news.

* * *

Site running Adabas and DB2 will be interested that Software AG has announced Event Replicator for Adabas, which allows companies to proactively and in real-time 'push' data changes in Adabas out to systems such as Adabas, DB2, Oracle, SQL Server, and Sybase, as well as XML-based devices and applications. This results in better system resource usage, cost avoidance, improved risk management, enhanced customer service, and greater confidence in business decisions based on continually-refreshed information.

Event Replicator automatically replicates any updates, additions, or deletions to specified Adabas datasets, then sends the changed data fields to the targeted systems or devices. The data is transformed to the format for each subscribed system so no changes need to be made to the application's infrastructure.

For further information contact:
URL: www1.softwareag.com/Corporate/News/latestnews/20050720_EventReplication_page.asp.

* * *

NEON Enterprise Software has announced Bind ImpactExpert, which is designed to help manage the large number of optimizer changes required when migrating to DB2 Version 8.

The product prechecks the plans stored in DB2 to determine those that will cause application performance problems after rebinding. It then manages the rebind process, ensuring consistent or improved performance during the migration.

Bind ImpactExpert identifies rebinds that are unnecessary and those that will improve application performance, and suppresses those that will degrade application performance. It can also automatically pinpoint poorly-performing SQL statements for further analysis.

For further information contact:
URL: www.neonesoft.com/product_bind.html.

* * *

DB2 users hearing that Oracle will support advanced XML features in its 10g database, including XQuery for accessing unstructured XML data, needn't be too alarmed that Oracle's features are better than DB2's. IBM has announced that its 'Viper' update to DB2, which will be available some time next year, will have hybrid support for both traditional structured data and native support for XML.

For further information contact your local IBM representative.