An intro to
REXX for DB2/2

Jeffrey W. Fisher
IBM Toronto Development Lab
Workstation Database Product Planner

v 09/03/93

Disclaimer

The sample code provided in this presentation is intended solely to illustrate some of the interfaces provided for REXX and DB2/2. The code is delivered "as-is", without guarantee as to correctness or completeness. In order to make use of this code on your own system, you should first thoroughly test it in your own particular environment. You may also wish to add more complete editing and error checking, and any other code necessary in order to comply with your shop standards.

Introduction

Introducing.... REXX!

- **Procedures Language 2/REXX is:**
  - a 32-bit structured interpretive programming language
  - an integral part of OS/2
  - included in IBM's SAA strategy
- **Procedures Language 2/REXX consists of:**
  - 1. the REXX language
    - REstructured eXtended eXecutor
    - developed by Mike Cowlishaw of IBM Hursley
  - 2. environment-specific interfaces
  - 3. special purpose extensions

REXX Features

- **Ease of use**
  - English like SAY PULL IF..THEN..ELSE
- **Free format**
  - instructions can span multiple lines
  - multiple instructions per line
  - instructions can start in any column
  - mixed case support
- **Built-In Functions**
  - processing, searching, comparison operations

REXX Features

- **Typeless variables**
  - meaning actually depends on the usage
  - all data regarded as character strings
  - no need to pre-define variables
  - can perform math on any strings that contain valid numbers
  - variables stored internally in a variable pool table
- **Parsing capabilities**
  - extensive character string manipulation capabilities
- **Debugging tools**
  - trace for run-time debugging

What's it good for?

- **Command procedures to interface with OS/2**
  - much more flexible than C
- **Intelligent job control language**
  - automate tasks
  - setup files, download from host
  - handle exceptions, not just 'stop'

~~JCL~~

- **Database work**
  - prototyping
  - administration
  - testing and benchmark driver
- **Application Development**
  - less-complex application work
  - weak for intensive-numerical operations

Language Type Considerations

- ***COBOL or C***
  - must be compiled, linked, bound
  - static or dynamic SQL
  - more involved coding
  - API to access CMD.EXE
  - 1mil + skillbase
  - readable (¬C??)

- ***REXX***
  - interpretive: 'interprets at run-time if changed
  - dynamic sql only
  - quick creation / execution
  - built-in access to CMD
  - growing; VM/TSO/AS400
  - very readable

These languages compliment each other and the existing development environment.

REXX provides significant function and flexibility for DB2/2 programmers users, and administrators.

Considerations

- **Advantages**
  - runs interpretively
    - no SQLPREP, compile, link, SQLBIND
    - edit in one window, SAVE, run in another
  - supports full implementation of SQL and APIs
    - convenient and powerful means of writing database management routines
  - can (of course!) be used against DRDA hosts
    - need to bind REXX's bind files against the host

Considerations ...

- **Disadvantages**
  - runs interpretively
    - interprets 1st time after source change
    - source is "visible"
  - not a powerful quick report generator
    - more-involved coding for columns, etc.
  - all SQL is dynamic

Invoking REXX

- **From OS/2:**
  - at an OS/2 command prompt
  - in calls from CMD (batch) or another REXX program
    - makes the old batch language quickly obsolete!
  - from the Workplace Shell
- **From within an application:**
  - use the REXXSAA function
    - procedures run in the same session, process, and thread as the caller
  - see "OS/2 Technical Library" pubs for information
  - example C to REXX program in the Toolkit

SQL Interface Requirement

- **Must register the required interface calls**
  - in STARTUP.CMD (suggested, easiest)
  - in each program that use them
  - do not drop the interface calls - someone else may be using them!
- **STARTUP.CMD is most convenient**

```
RCY = RxFuncAdd('SQLEXEC','SQLAR','SQLEXEC')
RCY = RxFuncAdd('SQLDBS','SQLAR','SQLDBS')
```

All following examples assume the register functions were done at STARTUP or other time

DB2/2 Application Elements

- **REXX statements**
- **Predefined DB2/2 Variables**
- **DB2/2 Interfaces**

  SQLDBS API calls to DB2/2 routines

  SQLEXEC SQL statements
- **Other interfaces or commands**
  - OS/2 Command Processor
  - OS/2 Workplace Shell
  - Developer Connection
    - NetBIOS and APPC samples
  - write your own
  - vendor interfaces

Predefined DB2/2 Variables

- ## SQLCA variables

SQLCA.SQLCODE	SQL return code
SQLCA.SQLSTATE	SQLSTATE return code
SQLCA.SQLERRML	Length of error msg
SQLCA.SQLERRMC	Error msg text
SQLCA.SQLERRP	8-byte diagnostics (or CONNECT identity)
SQLCA.SQLERRD.1	diagnostics
SQLCA.SQLERRD.2	diagnostics
SQLCA.SQLERRD.3	diagnostics
SQLCA.SQLERRD.4	diagnostics
SQLCA.SQLERRD.5	diagnostics
SQLCA.SQLERRD.6	diagnostics
SQLCA.WARN.0	warning msg
SQLCA.WARN.1	warning msg
SQLCA.WARN.2	warning msg
SQLCA.WARN.3	warning msg
SQLCA.WARN.4	warning msg
SQLCA.WARN.5	warning msg
SQLCA.WARN.6	warning msg
SQLCA.WARN.7	warning msg

- ## Other DB2/2 predefined variables (partial list)

SQLMSG	Text msg for <0 SQLCODE
SQLISL	Isolation Level
SQLRODA	default output SQLDA structure
SQLRIDA	default input SQLDA structure
SQLRDAT	default SQLCHAR structure

Predefined DB2/2 Variables...

- **Use of the variables**

```
if SQLCA.SQLCODE \= 0 then
    do
        say 'Error  = ' SQLCA.SQLCODE
        say '  Msg  = ' SQLMSG
    end
```

SQLDBS

API calls to DB2/2 Routines

- **DB2/2 API Routines**
  - Syntax: Call SQLDBS 'string'

    call SQLDBS 'CREATE DATABASE' dbname 'ON D'

- **'Command String' is made up of 3 elements:**
  - API command and operands in a string
  - REXX variables
  - SQL Host Variables

English-like!
Readable!

SQLEXEC

SQL Statements

- **Embedded SQL Statements**
  - Syntax: Call SQLEXEC 'command string'

```
call SQLEXEC 'FETCH c1 INTO :id :name :salary :comm'
if SQLCA.SQLCODE \= 0 then signal ErrorSQL
```

- **'Command String' is made up of 4 elements:**
  - REXX variables
  - SQL keywords
  - SQL host variables
  - Pre-defined identifiers

SQL Keywords

- **Following SQL keywords are supported:**

PREPARE	DECLARE	COMMIT
DESCRIBE	OPEN	ROLLBACK
EXECUTE IMMEDIATE	FETCH	CONNECT
EXECUTE	CLOSE	

- **All other keywords must be PREPAREd**

Remember:
Dynamic SQL!

Host Variables

- **3 types of host variables in the SQLEXEC string:**
  - statement
  - data
  - SQLDA
- **Preceded in the command string by a colon (:)**

```
call SQLEXEC 'FETCH c1 INTO :name :id :salary'
```

Statement Host Variables

- **A string containing an SQL statement**
  - used in PREPARE, EXECUTE, EXECUTE IMMEDIATE

```
CREATTAB = 'CREATE TABLE DEPTORG  (',
        'DEPTNUM        SMALLINT NOT NULL   ,',
        'DEPTNAME       VARCHAR(25)         ,',
        'MANAGER        SMALLINT            ,',
        'DIVISION       VARCHAR(25)         ,',
        'LOCATION       VARCHAR(33)         )'
call SQLEXEC 'EXECUTE IMMEDIATE :creattab'
if SQLCA.SQLCODE \= 0 then signal ErrorSQL
```

Data Host Variables

- Datatypes that can be manipulated by REXX
  - used for transferring data to DB2/2 or receiving data from a FETCH statement for use by the program

Column Type	*NOT Nullable*	*Nullable*
small INTEGER	**500**	**501**
large INTEGER	**496**	**497**
FLOAT	**480**	**481**
DECIMAL	**484**	**485**
fixed-length CHAR	**452**	**453**
VARCHAR	**448**	**449**
LONG VARCHAR	**456**	**457**
Null-terminated CHAR	**460**	**461**
GRAPHIC DBCS	**468**	**469**
VARGRAPHIC DBCS	**464**	**465**
LONG VARGRAPHIC DBCS	**472**	**473**
DATE	**384**	**385**
TIME	**388**	**389**
TIMESTAMP	**392**	**393**

Predefined Identifiers

- **2 types of identifiers are predefined**
  - Cursor names, range from C1 to C100
    - used in **DECLARE**, **OPEN**, **FETCH**, **CLOSE**
    - C1 thru C50: cursors declared without hold
    - C51 thru C100: curcors declared using WITH HOLD
  - Statement names, range from S1 to S100
    - used in **DECLARE**, **DESCRIBE**, **PREPARE**, **EXECUTE**
  - **Other names not allowed**

Use of CURSORs

- **Prepare the statement (dynamic SQL)**

  selstmt = 'select acct, id, job, name from mytab order by id'
  call SQLEXEC 'PREPARE s1 FROM :selstmt'

- **Declare the cursor**

  call SQLEXEC 'DECLARE c1 CURSOR FOR s1'

- **Open the cursor**

  call SQLEXEC 'OPEN c1'

- **Fetch rows until SQLCODE = +100**

  call SQLEXEC 'FETCH c1 into :acct, :id, :job, :name'.

- **Close the cursor**

  call SQLEXEC 'CLOSE c1'

Using NULL Indicators

- **Handled the same way as other languages**

```
call SQLEXEC 'FETCH c1 INTO :id,',
                  ':name :n_name,',
                  ':comm :n_comm'

if n_name < 0 then name = ' '
if n_comm < 0 then comm = ' '

say 'Id  : ' id
say 'Name: ' name
say 'Comm: ' comm
```

But, much simpler:
It's REXX!

Isolation Level

- **REXX plan bound to the database during CREATE or MIGRATE**
  - defaults to CURSOR STABILITY
- **3 bind files provided**
  - SQLARXCS.BND cursor stability
  - SQLARXRR.BND repeatable read
  - SQLARXUR.BND uncommitted read
- **Change isolation level thru:**
  - SQLDBS CHANGE ISOLATION LEVEL
  - manual rebind using the .BND file against the database

Advanced Topics

Connecting thru DDCS/2 v2

- **SQLJBIND: supplied with DDCS/2 v2**
- **Binds the following to the DRDA Host**
  - SQLPREP and SQLBIND
  - REXX plans
  - Command Line Processor
  - IMPORT and EXPORT
- **Must GRANT authorities to users or public**
  - packages created with COLLECTION ID of NULLID
- **Can also support backlevel EE and ES clients**
- **Full support of both interfaces**
  - supported routines
  - SAA SQL versus non-SAA SQL

Extended Screen Control

- **Use color to highlight**
  - menus
  - input edit errors
  - exceptional conditions
  - error messages
- **Coding Example:**

```
ansii.esc     = '1B'x
c.nor         = ansii.esc || '&lbr.0m'
c.red         = c.nor || ansii.esc || '&lbr.31m'
say c.red 'This line comes out red!'
```

Advanced Screen Control

- **New OS/2 2.x REXX functions**
  - can be used to control position of cursor on screen
    - menuing
    - input control

```
call SysCls
say c.itcyan '*******************************************'
say c.yellow ' Demo Menu                                  '
say c.yellow '                                            '
say c.yellow ' _ 0   Menu: Program Options & Defaults   '
say c.yellow '   1   Menu: Database Objects             '
say c.yellow '   2   Menu: Adjust Tuning Parms          '
say c.yellow '   x   Exit                               '
say c.itcyan '*******************************************'
row = 3
col = 2
pos = SysCurPos(row,col)
selection = SysGetKey('NOECHO')
call SysCls
SELECT
   WHEN selection = '0' then call ProgramOptions
   WHEN selection = '1' then call ObjectMenu
   WHEN selection = '2' then call ParmTuneMenu
   WHEN selection = 'x' then signal EndProg
   OTHERWISE          NOP
END
```

Use of Sound

- **Use sound to signal attention**
  - required input
  - abnormal circumstances
  - editting error

CALL BEEP(frequency,duration)

- **This is an OS/2-specific non-SAA function**

```
say c.red
call beep 220,1000
say '       >>>  An error has occured:'
say '            SQLCODE = ' SQLCA.SQLCODE
```

Samples

sample: HELLO

```
/*
   Name: HELLO.cmd
*/
Start:
   say 'a conversation'
Begin:
   say 'Hello! What is your name?'
   pull who
   if who = ' ' then say 'Hello Stranger'
   else say 'Hello' who
   exit
```

sample: STARTUP.cmd

```
/*
   Name         : STARTUP.cmd
   Purpose      : Boot time startup functions
   Platform     : IBM DB2/2
   Dependency   : Place on root dir on c: drive;
*/

StartUp:
   address cmd '@ECHO OFF'

   say
   say 'STARTUP Processing'

   say 'Step 1: Register REXX SQL functions'
   rcy = RxFuncAdd('SQLDBS' , 'SQLAR' , 'SQLDBS')
   rcy = RxFuncAdd('SQLEXEC', 'SQLAR', 'SQLEXEC')

   say 'Step 2: Start DB2/2'
   call SQLDBS 'START DATABASE MANAGER'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL

   say 'Step 3: Logon'
   address cmd 'LOGON USERID /P=PASSWORD'
   if RC \= 0 then signal ErrorRC

   say
   say  '>>> STARTUP command complete <<<'
   say
```

sample: STARTUP.cmd

```
EndProg:
   address cmd 'EXIT'


 ErrorRC:
   call beep 220,1000
   say
   say '       >>>  OS/2 has returned a fatal condition code'
   say '            RC           = ' RC
   say
   pause
   signal EndProg


 ErrorSQL:
   call beep 220,1000
   say
   say '       >>>  SQL has returned a fatal condition code'
   say '            SQLCODE      = ' SQLCA.SQLCODE
   say '            MSG          = ' SQLMSG
   say
   pause
   signal EndProg
```

sample: Showauth Utility

```
/*
   Name        : SHOWAUTH.cmd
   Purpose      : Show authorizations of logged-on user
                 for each database on drive C
   Platform     : IBM DB2/2
*/
Status:
   address cmd '@echo off'
   call SQLDBS 'OPEN DATABASE DIRECTORY ON C USING :scanvar'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   scanid = scanvar.1
   say 'There are' scanvar.2 'databases on drive C:'
   do i=1 to scanvar.2
      call SQLDBS 'GET DATABASE DIRECTORY ENTRY :scanvar.1',
                  'USING :entry'
      if SQLCA.SQLCODE \= 0 then signal ErrorSQL
      call Start_Using entry.1
      call SQLDBS 'GET AUTHORIZATIONS :cvar'
      if SQLCA.SQLCODE \= 0 then signal ErrorSQL
      call Stop_Using
      if cvar.1 = '1' then cvar1 = 'yes'
        else cvar1 = 'no '
      if cvar.2 = '1' then cvar2 = 'yes'
        else cvar2 = 'no '
      if cvar.3 = '1' then cvar3 = 'yes'
        else cvar3 = 'no '
      if cvar.4 = '1' then cvar4 = 'yes'
        else cvar4 = 'no '
      if cvar.5 = '1' then cvar5 = 'yes'
        else cvar5 = 'no '
      if cvar.6 = '1' then cvar6 = 'yes'
        else cvar6 = 'no '
      if cvar.7 = '1' then cvar7 = 'yes'
        else cvar7 = 'no '
      if cvar.8 = '1' then cvar8 = 'yes'
        else cvar8 = 'no '
      if cvar.9 = '1' then cvar9 = 'yes'
        else cvar9 = 'no '
      if cvar.10 = '1' then cvar1
```

sample: Showauth Utility

```
      address cmd 'CLS'
      say
      say '  ----------------------------------------'
      say '  Your authorizations for Database:' entry.1
      say '             Direct      Indirect'
      say '    SYSADM     'CVAR1'        'CVAR6
      say '    DBADM      'CVAR2'        'CVAR7
      say '    CREATETAB  'CVAR3'        'CVAR8
      say '    BINDADD    'CVAR4'        'CVAR9
      say '    CONNECT    'CVAR5'        'CVAR10
      say '  ----------------------------------------'
      say
      pause

   end

   call SQLDBS 'CLOSE DATABASE DIRECTORY :scanid'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL

EndProg:
   exit

ErrorSQL:
   say '       >>>  SQL has returned a fatal condition code'
   say '            SQLCODE      = ' SQLCA.SQLCODE
   say '            MSG          = ' SQLMSG
   say
   call SQLDBS 'CLOSE DATABASE DIRECTORY :scanvar'
   pause
   signal EndProg
```

sample: Showauth Utility

```
Start_Using:
   arg DBNAME
   call SQLEXEC 'CONNECT TO' dbname 'IN SHARED MODE'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   return

Stop_Using:
   call SQLEXEC 'CONNECT RESET'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   return
```

sample: cursor fetches

```
/*
   Name         : DISPTB.cmd
   Purpose      : Display the "SAMPLE STAFF" table
   Platform     : IBM DB2/2
   Dependency   : Install the SAMPLE database
*/
Main:
   call Start_Using
   stmtbuf = 'select * from staff order by id'
   call SQLEXEC 'PREPARE s1 FROM :stmtbuf'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   call SQLEXEC 'DECLARE c1 CURSOR FOR s1'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   call SQLEXEC 'OPEN c1'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   do until SQLCA.SQLCODE \= 0
      call SQLEXEC 'FETCH c1 INTO :id,',
                   ':name    :n_name,',
                   ':dept    :n_dept,',
                   ':job     :n_job,',
                   ':years   :n_years,',
                   ':salary  :n_salary,',
                   ':comm    :n_comm'
      if SQLCA.SQLCODE = +100 then leave
      else if SQLCA.SQLCODE \= 0 then signal ErrorSQL
      if n_name   < 0 then name   = ' '
      if n_dept   < 0 then dept   = ' '
      if n_job    < 0 then job    = ' '
      if n_years  < 0 then years  = ' '
      if n_salary < 0 then salary = ' '
      if n_comm   < 0 then comm   = ' '
      say 'Identification: ' id
      say '          Name: ' name
      say '    Department: ' dept
      say '           Job  ' job
      say '         Years: ' years
      say '        Salary: ' salary
      say '    Commission: ' comm
      say
   end
```

sample: cursor fetches

```
EndProg:
   call SQLEXEC 'CLOSE c1'

   call Stop_Using

   exit


ErrorSQL:
   call beep 220,1000
   say
   say '       >>>  SQL has returned a fatal condition code'
   say '            SQLCODE      = ' SQLCA.SQLCODE
   say '            MSG          = ' SQLMSG
   say
   signal EndProg


Start_Using:
   call SQLEXEC 'CONNECT TO SAMPLE IN SHARED MODE'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   return


Stop_Using:
   call SQLEXEC 'CONNECT RESET'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL
   return
```

sample: Compile routine

```
/*
   Name        : COMPILE.cmd
   Purpose     : Performs the following functions:
              - SQLPREP
              - COBOL
              - LINK
              - SQLBIND
   Requirements : 1) MUST be run in directory in which source resides
              2) STARTDBM must already have been done
              3) Do not specify any extensions
              4) The beginning step requires a .SQB extension
              5) Single .BND files only!
*/
Questions:
   say 'Enter the name of your program, without the extension'
   pull progname
   say 'Enter any optional compiler directives'
   pull compdir
   say 'Enter the name of the database to be bound to'
   pull dbname
   say 'Enter the isolation level for plan' progname
   pull isolevel
Procedure:
   say 'Step 1: Precompile' progname
   step = 'precompile'
   address cmd 'SQLPREP',
            progname'.SQB',
            dbname,
            '/B',
            '/P='progname,
            '/M='progname'.PRE'
   if RC \= 0 then signal ErrorRC

   say 'Step 2: compile' progname
   step = 'compile'
   address cmd 'COBOL ',
            progname',',
            progname',',
            progname',',
            ',',
            '/CONFIRM 'compdir,
            '/MODEL"LARGE"'
   if RC \= 0 then signal ErrorRC
```

sample: Compile routine

```
    say 'Step 3: linkedit' progname
    step = 'linkedit'
    address cmd 'LINK ',
            progname'.OBJ,',
            progname'.EXE,',
            ',',
            'd:\COBOL\COBOL.LIB+',
            'c:\OS2\DOSCALLS.LIB+',
            'c:\SQLLIB\SQL_DYN+',
            'c:\SQLLIB\SQL_STAT+',
            'c:\SQLLIB\DSQCI',
            ',',
            ' /NOP'
    if RC \= 0 then signal ErrorRC

    say 'Step 4: BIND plan' progname
    step = 'bind'
    address cmd 'SQLBIND',
            progname'.BND',
            dbname,
            '/I='isolevel
    if RC \= 0 then signal ErrorRC

    say 'Step 5: Make the executable WINDOWABLE'
    step = 'winable'
    address cmd 'WINABLE',
            progname'.exe',
            '/c'
    if RC \= 0 then signal ErrorRC


EndProg:
    exit


ErrorRC:
    call beep 220,2500
    say
    say '       >>>  OS/2 has returned a fatal condition code'
    say '            abending step = ' step
    say '            RC            = ' RC
    say
    pause
    signal EndProg
```

sample: Directory Scan

```
/*
   Name        : DIRSCAN.cmd
   Purpose     : Database Directory Scan
   Platform    : IBM DB2/2
*/

address cmd '@echo off'
call SetColor
address cmd 'CLS'

Menu:
   say
   say c.gre '******************************'
   say c.yel '   Database Directory Scan'
   say c.gre '    Enter:'c.nor' Drive 'c.gre'to scan'
   say c.gre '       or:'c.nor' X 'c.gre'to EXIT'
   say c.gre '******************************'c.nor
   pull drive
   if drive = 'X'     then signal EndProg
   if drive = ' ' then
      do
         say c.red 'Invalid response: try again'
         signal menu
      end
   call DirScan

   signal menu
```

sample: Directory Scan

```
DirScan:
   step = 'open database directory'
   call SQLDBS 'OPEN DATABASE DIRECTORY',
            ' ON' drive,
            ' USING :scanvar'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL

   scan_flag = 'Y'
   address cmd 'CLS'
   say

   do i=1 to scanvar.2
      step = 'get database directory entry'
      call SQLDBS 'GET DATABASE DIRECTORY ENTRY :scanvar.1',
               ' USING :entry'
      if SQLCA.SQLCODE \= 0 then signal ErrorSQL
      say
      say c.itcya 'Database:' entry.1 'Alias:' entry.2
      say '  Internal name:' entry.4
      say '  Node name    :' entry.5
      say '  Drive        :' entry.3
      say '  Dbtype       :' entry.6
      say '  Comment      :' entry.7
      say '  Codepage     :' entry.8
      say '  Enttype      :' entry.9
      say c.nor
      pause
   end
   step = 'close database directory'
   call SQLDBS 'CLOSE DATABASE DIRECTORY :scanvar.1'
   if SQLCA.SQLCODE \= 0 then signal ErrorSQL

   address cmd 'CLS'
   return
```

sample: Database Monitor

```
/*
   Name        : mon_db.cmd
   Purpose     : Monitor log files for a selected database
   Platform    : DB2/2 and OS/2 2.1
   Author      : Jeff Fisher
                 Copyright IBM Corporation 1993
                 IBM Toronto Development Lab

    Written    : 08/31/91
*/

signal on syntax
signal on error

call BeginProg

MainControl:
    drivemap = SysDriveMap(C,LOCAL)
    occurance = 0
    drives = words(drivemap)

    do i = 1 to drives
      parse var drivemap drive.i drivemap
      db_drive.i = left(strip(drive.i), 1)
      call DirScan db_drive.i
    end
```

Putting it all together!

sample: Database Monitor

```
   if scan_flag = 'Y' then
      do
         step = 'close database directory'
         call SQLDBS 'CLOSE DATABASE DIRECTORY :scanvar.1'
      end

SelectMonitorDBMenu:
   call SysCls
   say
   say c.itcyan '*********************************************'
   say c.yellow ' Choose a Database to monitor:            '
   do z=1 to occurance
      say c.yellow '   'z '  ' dbname.z
   end
   say c.yellow ' Or, choose an option:                    '
   say c.yellow '   R    (refresh menu)                    '
   say c.yellow '   X    (to exit)                         '
   say c.itcyan '*********************************************'

   row = 3
   col = 2
   pos = SysCurPos(row,col)
   SelectDB = SysGetKey('NOECHO')

   if SelectDB = 'R' | SelectDB = 'r' then signal MainControl
   if SelectDB = 'X' | SelectDB = 'x' then signal EndProg

   say SelectDB
   call MonitorMain dbname.SelectDB
```

sample: Database Monitor

```
    signal MainControl


DirScan:
    arg scandrive
    step = 'open database directory'
    call SQLDBS 'OPEN DATABASE DIRECTORY ON' scandrive 'using :scanvar'
    if SQLCA.SQLCODE = 1057 then return
    if SQLCA.SQLCODE = -1031 then return
    if SQLCA.SQLCODE \= 0 then signal ErrorSQL

    scan_flag = 'Y'

    do loop1 = 1 to scanvar.2
      step = 'get database directory entry'
      call SQLDBS 'GET DATABASE DIRECTORY ENTRY :scanvar.1 USING :entry'
      if SQLCA.SQLCODE \= 0 then signal ErrorSQL
      occurance = occurance + 1
      dbname.occurance = entry.1
    end

    step = 'close database directory'
    call SQLDBS 'CLOSE DATABASE DIRECTORY :scanvar.1'
    if SQLCA.SQLCODE \= 0 then signal ErrorSQL

    return


MonitorMain:
```

sample: Database Monitor

```
/*

*/
    arg MonitorDB

    EndSession = 'N'

    call CollectStatus

    call FreeResources

    if EndSession = 'Y' then return
    else signal MonitorMain


CollectStatus:
/*

*/
    call SysCls

    step = 'Collect All Status'
    call SQLDBS 'COLLECT ALL STATUS for database' MonitorDB 'USING :SSTAT'

    call CheckSafeSQLcode
    if SafeSQLcode = 'Y' then signal CollectStatus
    else
        if SafeSQLcode = 'N' then signal ErrorSQL
```

sample: Database Monitor

```
if SSTAT.10 > 0 then call UserStat
else
   do
      say
      say c.yellow MonitorDB ': NO USERS AT THIS TIME '
      say c.grey
      say c.itblue '<enter> to continue; <X> to end'
      nullinput = SysGetKey('NOECHO')
      if nullinput = 'X' | nullinput = 'x' then EndSession = 'Y'
   end

return
```

sample: Database Monitor

```
UserStat:
/*

*/
  step = 'Get User Status loop'
  tod = time()
  CALL SQLDBS 'GET USER STATUS FOR DATABASE' MonitorDB 'USING :USTAT'
  if SQLCA.SQLCODE \= 0 then signal ErrorSQL

  do y=1 to USTAT.0
    row = 3
    col = 2
    pos = SysCurPos(row,col)

    if sstat.10 = 1 then
      say c.yellow MonitorDB 'has'c.itmagenta sstat.10 c.yellow'user at' tod
    else
      say c.yellow MonitorDB 'has'c.itmagenta sstat.10 c.yellow'users at' tod
    say

    if ustat.i.8 = '0' then ustat.y.8 = '00000'
    say c.itmagenta '#'y c.itcyan 'Authid:'c.itgreen USTAT.y.6 c.itcyan 'Auth lvl:'c.itgreen USTAT.y.8

    checkvar = VERIFY(USTAT.y.7,' ')
    if checkvar = 1 then USTAT.y.7 = '(local)'
    say c.itcyan '    Node  :'c.itgreen USTAT.y.7

    say
    say c.yellow   'Since CONNECT:'
    say c.itcyan   '  UOWs                :'c.itgreen USTAT.y.1
    say c.itcyan   '  SQL transactions    :'c.itgreen USTAT.y.2
    say c.itcyan   '  Elapsed time (secs) :'c.itgreen USTAT.y.4
    say
    say c.yellow   'Current UOW:'
    say c.itcyan   '  SQL transactions    :'c.itgreen USTAT.y.3
    say c.itcyan   '  Elapsed time (secs) :'c.itgreen USTAT.y.5
    call CalcTPS
    say c.itcyan   '  TPS rate            :'c.itgreen TPS
    if ustat.y.9 = 'S' then ustat.y.9 = 'Started'
    if ustat.y.9 = 'R' then ustat.y.9 = 'Open, no changes'
    if ustat.y.9 = 'C' then ustat.y.9 = 'Changed DB'
      say c.itcyan '  Status              :'c.itgreen USTAT.y.9
```

sample: Database Monitor

```
    if ustat.y.10 = 'N' then
      do
        ustat.y.10 = 'NO'
        say c.itcyan '  Waiting for LOCK?   :'c.itgreen USTAT.y.10
      end
    else
      do
        ustat.y.10 = 'YES'
        say c.itcyan '  Waiting for LOCK?   :'c.itred   USTAT.y.10 '<---'
      end

    say c.normal
    say c.itblue '<enter> to continue; <X> to end'
    nullinput = SysGetKey('NOECHO')
    if nullinput = 'X' | nullinput = 'x' then EndSession = 'Y'
  end
  return 0


CalcTPS:
/*

*/
  tps = 0

  SELECT
     WHEN ustat.y.3 = 0  then tps = 0
     WHEN ustat.y.5 = 0  then tps = 0
     OTHERWISE           tps = ustat.y.3 / ustat.y.5
  END

  return
```

sample: Database Monitor

```
CheckSafeSQLcode:
/*

*/
  SafeSQLcode = 'N'

  SELECT
    WHEN SQLCA.SQLCODE = 0 then
     do
      SafeSQLcode = '0'
     end
    WHEN SQLCA.SQLCODE = -1013 then
     do
      SafeSQLcode = 'Y'
      say
      say c.yellow '*** Database FRDEMO does not exist       ***'
      say c.yellow '*** create the database and try again    ***'
     end
    WHEN SQLCA.SQLCODE = -1033 then
     do
      SafeSQLcode = 'Y'
      say
      say c.yellow '*** Database cannot be opened for status ***'
      say c.yellow '*** Database directory being updated     ***'
     end
    WHEN SQLCA.SQLCODE = -1035 then
     do
      SafeSQLcode = 'Y'
      say
      say c.yellow '*** Database cannot be opened for status ***'
      say c.yellow '*** Another user has exclusive use of it ***'
     end
    WHEN SQLCA.SQLCODE = -1119 then
     do
      SafeSQLcode = 'Y'
      say
      say c.yellow '*** A Database Restore is in progress ***'
     end
```

sample: Database Monitor

```
    WHEN SQLCA.SQLCODE = -1117 then
      do
        SafeSQLcode = 'Y'
        say
        say c.yellow '*** A Roll Forward is pending ***'
      end
    OTHERWISE SafeSQLcode = 'N'
  END

  SELECT
    WHEN SafeSQLcode = 'Y' then
      do
        say c.yellow '    Press <enter> when        '
        say c.yellow '    ready in order to         '
        say c.yellow '    continue execution.....   '
        say c.grey
        pause
      end
    OTHERWISE NOP
  END

  return


FreeResources:
/*

*/
  step = 'Free Status Resources'
  call SQLDBS 'FREE STATUS RESOURCES'

  call CheckSafeSQLcode
  if SafeSQLcode = 'Y' then signal FreeResources
  else
    if SafeSQLcode = 'N' then signal ErrorSQL

  return
```

sample: Database Monitor

```
BeginProg:

   call RxFuncAdd 'SysLoadFuncs','RexxUtil','SysLoadFuncs'
   call RxFuncAdd 'SQLDBS','SQLAR','SQLDBS'
   call RxFuncAdd 'SQLEXEC','SQLAR','SQLEXEC'
   call SysLoadFuncs
   address cmd '@ECHO OFF'
   call SysCls
   call SetColor

   return


EndProg:

   say c.normal

   call SysCls

   exit


Error:

   call beep 220,1000
   say c.itred
   say '        >>>  Rexx has returned a signal on error'
   say '             abending step = ' step
   say '             source line   = ' sourceline(sigl)
   say
   pause
   signal EndProg
```

sample: Database Monitor

```
Syntax:
/*

*/
   call beep 220,1000
   say c.itred
   say '       >>>  Rexx has returned a signal on syntax'
   say '            abending step = ' step
   say '            source line   = ' sourceline(sigl)
   say
   pause
   signal EndProg


ErrorSQL:
/*

*/
   call beep 220,1000
   say c.itred
   say '       >>>  SQL has returned a fatal condition code'
   say '            abending step = ' step
   say '            SQLCODE       = ' SQLCA.SQLCODE
   say '            MSG           = ' SQLMSG
   say
   pause
   signal EndProg
```

sample: Database Monitor

```
SetColor:
/*

*/
   ansii.esc      = '1B'x
   c.normal       = ansii.esc || '[0m'
   c.highlite     = ansii.esc || '[1m'
   c.blackback    = ansii.esc || '[40m'
   c.grey         = c.normal || ansii.esc || '[37m'
   c.itred        = c.highlite || ansii.esc || '[31m'
   c.itnormal     = c.highlite || ansii.esc || '[32m'
   c.yellow       = c.highlite || ansii.esc || '[33m'
   c.itgreen      = c.highlite || ansii.esc || '[32m'
   c.itblue       = c.highlite || ansii.esc || '[34m'
   c.itmagenta    = c.highlite || ansii.esc || '[35m'
   c.itcyan       = c.highlite || ansii.esc || '[36m'
   c.white        = c.highlite || ansii.esc || '[37m'
   c.std          = c.normal || c.itcyan || c.blackback
   c.reset        = c.normal || c.grey || c.blackback
   return 0
```

Last Page

Fini!