This presentation contains confidential information. If you agree to keep this information confidential, then you may proceed.

Copyright © MAX SOFTWARE LLC 1995-1999

MAX/REXX, MAX/DATA UTIL, MAX/PDF, MAX/2000, MAX RT-Compiler, MAX RUNTIME and MAX/BATCH are trademarks of MAX SOFTWARE LLC.

All rights reserved. No part of this presentation may be reproduced, stored in a retrieval system, or transmitted by any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, except as may be expressly permitted by the applicable copyright statues or in writing by the publisher.
Welcome to MAX/REXX

Currently available on OS/390 where a large percentage of the legacy business data resides.

MAX/REXX

Allows the REXX programming language to be used to solve business problems.

Provides the capabilities to process business data with the performance needed in a comprehensive language environment.
The competitiveness of an organization can be directly linked to its ability to respond quickly to changes and opportunities.

Responsiveness in an information driven business requires a standardized set of tools that can be used to rapidly develop or revise both applications and data.

REXX is the language that allows for quick response. However, classic REXX cannot manipulate all the data types and file formats that typify today's applications.
Provides the interfaces between REXX and VSAM, SAM, PDS and DB2 data.

Provides automatic access to any field in a SAM or VSAM file through the use of a COBOL or PL/1 record layout.

Can dramatically improve performance accessing data by selecting only a subset of the data to be processed, using the "WHERE" clause.
RX VSAM
access VSAM, SAM & PD S files forward, backward, update

RX SQL
full DB2 access with the efficiency of static SQL. Supports row/s at a time cursor based processing

J-Compiler
compile REXX into executable object modules allowing for security and change control. Execute directly from JCL or TSO

RXMVS
use the MVS functions to ENQUE, DEQUE, LINK, LOAD, ALLOCATE, SORT and do extensive date calculations

REXX

Slide 5 of 42
Rapid Development
Easier Application Maintenance
Faster Program Testing
Quick Problem Analysis
Rapid Development

REXX's interpretive nature and concise syntax allow fast prototyping and switching from coding to immediate execution.

MAX/REXX

Programmers using MAX/REXX can develop, test and implement programs in a fraction of the time needed for COBOL.
Easier Application Maintenance

MAX/REXX programs have a structured, top-down syntax, and programs are smaller and easier to read.

MAX/REXX uses an externally defined file layout. Changes to file format do not necessarily require modification or recompilation of the associated programs.
Original copybook shows the employee start date as an 8 position field containing the date in format YY/MM/DD.
In this example, the copybook is stored in an external file called "MXS.P390.COPYLIB (CBHDR2)"

```rexx
/* REXX */
  ...
  IF "RXVSAM" ("OPEN FILE(SYSUT1) COPYBOOK(MXS.P390.COPYLIB(CBHDRV2))" ;
    "SEQ")<>0 THEN DO
    SAY 'RC='VSAMCODE 'MSG='VSAMMSG
  . .
  END
  . .
  DO WHILE "RXVSAM" ("READNEXT FILE(SYSUT1) ")=0
    SAY 'EMPLOYEE: 'NAME_LAST', ' NAME_FIRST ',
        'STARTING DATE: 'START_DATE /* show start date */
  END
  . .
```

This instruction associates a copybook with a file at OPEN.

This instruction reads each record and creates variable names for all fields in each record.
**Notice the two digit year**

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Starting Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doe</td>
<td>John</td>
<td>02/15/94</td>
</tr>
<tr>
<td>Jones</td>
<td>Joanne</td>
<td>03/24/92</td>
</tr>
<tr>
<td>Jones</td>
<td>James</td>
<td>11/01/96</td>
</tr>
<tr>
<td>Smith</td>
<td>Matthew</td>
<td>07/21/98</td>
</tr>
<tr>
<td>Johnson</td>
<td>Sally</td>
<td>01/15/89</td>
</tr>
</tbody>
</table>
The copybook is changed to reflect the conversion of the file to contain the 4 digit year YYYYMMDD.

```
05 EMPLOYEE-AMOUNT PIC S9(9) COMP-3.
05 START-DATE   PIC X(10).
```

The program runs with no changes or recompilations. Changes in the source copybook are dynamically reflected in the output.

<table>
<thead>
<tr>
<th>EMPLOYEE: DOE</th>
<th>JOHN</th>
<th>STARTING DATE: 1994/02/15</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE: JONES</td>
<td>JOANNE</td>
<td>STARTING DATE: 1992/03/24</td>
</tr>
<tr>
<td>EMPLOYEE: JONES</td>
<td>JAMES</td>
<td>STARTING DATE: 1996/11/01</td>
</tr>
<tr>
<td>EMPLOYEE: SMITH</td>
<td>MATTHEW</td>
<td>STARTING DATE: 1998/07/21</td>
</tr>
<tr>
<td>EMPLOYEE: JOHNSON</td>
<td>SALLY</td>
<td>STARTING DATE: 1989/01/15</td>
</tr>
</tbody>
</table>

Slide 13 of 42
### Comparison of reports before and after layout change

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>FIRST NAME</th>
<th>STARTING DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOE</td>
<td>JOHN</td>
<td>94/02/15</td>
</tr>
<tr>
<td>JONES</td>
<td>JOANNE</td>
<td>92/03/24</td>
</tr>
<tr>
<td>JONES</td>
<td>JAMES</td>
<td>96/11/01</td>
</tr>
<tr>
<td>SMITH</td>
<td>MATHEW</td>
<td>98/07/21</td>
</tr>
<tr>
<td>JOHNSON</td>
<td>SALLY</td>
<td>89/01/15</td>
</tr>
</tbody>
</table>

**------------------------------**

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>FIRST NAME</th>
<th>STARTING DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOE</td>
<td>JOHN</td>
<td>1994/02/15</td>
</tr>
<tr>
<td>JONES</td>
<td>JOANNE</td>
<td>1992/03/24</td>
</tr>
<tr>
<td>JONES</td>
<td>JAMES</td>
<td>1996/11/01</td>
</tr>
<tr>
<td>SMITH</td>
<td>MATHEW</td>
<td>1998/07/21</td>
</tr>
<tr>
<td>JOHNSON</td>
<td>SALLY</td>
<td>1989/01/15</td>
</tr>
</tbody>
</table>

The underlying program never changed, only the COBOL copybook.
MAX/REXX programs are much smaller than traditional programs, resulting in fewer opportunities for failure.

Built-in interactive trace capabilities facilitate faster testing and debugging.
Faster Program Testing

All MAX/REXX components optionally provide trace feedback.

```
OPEN FILE(SYSUT1) COPYBOOK(MXS.P390.COPYLIB(CBHDR))
SEQUID
RC=00000000 RXV00041 SUCCESSFUL COMPLETION
READNEXT FILE(SYSUT1) WHERE(10,EQ,'Y')
RC=00000000 RXV00041 SUCCESSFUL COMPLETION
REWRITE FILE(SYSUT1)
RC=00000000 RXV00041 SUCCESSFUL COMPLETION
READNEXT FILE(SYSUT1) WHERE(10,EQ,'Y')
RC=00000000 RXV00041 SUCCESSFUL COMPLETION
REWRITE FILE(SYSUT1)
RC=00000000 RXV00041 SUCCESSFUL COMPLETION

... ...

READNEXT FILE(SYSUT1) WHERE(10,EQ,'Y')
RC=00000000 RXV00046I VSAM OPERATION FAILED OPER=RKT RC=01F
CLOSE FILE(SYSUT1)
RC=00000000 RXV00041 SUCCESSFUL COMPLETION
... ...
```

TRACE provides both passed command and returned results
Quick Problem Analysis

Special variables for the return-code and a full text message follow every invocation. They are available regardless of how MAX/REXX was invoked. This facilitates the use of a common error routine to handle MAX/REXX errors.

CALL 'RXVSAM' 'OPEN FILE(SYSUT1)' 
IF VSAMCODE<0 THEN CALL CLEANUP
  ... 
IF RXVSAM('FIND FILE(SYSUT1) MEMBER(RXPDS4) '< >0 THEN
  CALL CLEANUP
  ... 
CLEANUP:
  IF VSAMCODE>8 THEN DO
    SAY 'RXVSAM failed RC='VSAMCODE VSAMSG
  END
Rapid Development
   – able to switch from coding to execution without any intervening steps

Easier Application Maintenance
   – programs are smaller & easier to read

Faster Program Testing
   – built in interactive trace capabilities facilitate faster testing & debugging

Quick Problem Analysis
   – line number & full descriptive message provided

Standardization with REXX
   – MAX/REXX extends the use of REXX with SQL and Command Level Syntax that is already familiar to programmers.
MAX/REXX Features & Capabilities

🌟 Inserts, updates, or deletes records directly in SAM or VSAM files of any type, size or length.

🌟 Uses standard Command Level Syntax for accessing VSAM, SAM and PDS data files.

```
DO WHILE ...

   CALL "RXVSAM" "READ FILE(SYSUT1) INTO(RECORD) UPDATE"  
   IF VSARCODE<>0 THEN . . .
       . . .
   END

   CALL "RXVSAM" "REWRITE FILE(SYSUT1) FROM(RECORD)"
   IF VSARCODE<>0 THEN . . .
       . . .
   END
```

Simple commands to read and update a file
MAX/REXX Features & Capabilities

- Allows concurrent access to multiple VSAM, SAM or PDS files
- Processes SAM and VSAM files forward or backward

DO WHILE "RXVSAM"("READNEXT FILE(SYSUT1) INTO(RCD)")=0

One statement to read through an entire file
A special operand "WHERE" allows selected records to be returned to the REXX program from the READ.

```rexx
call "rxxsrh" "open file(file1) seq" /* open file */
/* only retrieve the record that are type 1 or type 2 */
/* record type is in position 10 */
do until usrcode<>0
   rc= rxxsrh ("readnext file(file1) into (rcd)", "where (10, eq, '1', or, 10, eq, '2')")
   if rc>0 then do
     ...
   end
```

This example would return records that contain '1' in position 10 or '2' in position 10.
MAX/REXX Features & Capabilities

- Processes PDS directory information
- Process PDS members

```
MEM_NAME='TESTPM1'
CALL 'RXVSAM' 'FIND FILE(SYSUT1) MEMBER('MEM_NAME')'
IF VSAMCODE<>0 THEN . . .
```

This statement will position to start of member
MAX/REXX Features & Capabilities

Uses COBOL or PL1 copybooks to automatically create REXX variables for each field in the record.

OPEN statement associates copybook with a file.

Record data is automatically accessed using field names as REXX variables.

```cobol
IF "RXVSAM" ("OPEN FILE (SYSUT1) COPYBOOK (RXS.P390. COPILIB (CBHDR)) ", "SEQUPD") <> 0 THEN DO
  ...
  DO WHILE "RXVSAM" ("READNEXT FILE (SYSUT1) ") = 0
    IF EMPLOYEE_STATE = 'CO' THEN AREA_CODE = '303'
    ...
    CALL "RXVSAM" "REWRITE FILE (SYSUT1)"
  ...
END
```
MAX/REXX Features & Capabilities

- Supports dynamic and static SQL statements for accessing DB2 databases
- Supports multiple concurrent SQL cursors for accessing DB2 databases
- Provides full data integrity with commit and rollback support for DB2 data

```
"RXSQL DECLARE C1 CURSOR FOR",
SELECT NAME, STGROUP FROM SYSTEM."DB"
"WHERE STGROUP = 'MAXG01'
...
"RXSQL DECLARE C20 CURSOR FOR",
"SELECT VTREE FROM SYSTBM.SYSVTREE"
```

Up to 100 cursor names supported per program
Provides feedback on all error conditions.

Includes numerous language extensions, such as Date calculations, SORT, ENQUE and CATALOG.

CALL 'RXMVS' 'DATE2JUL DATE('FINAL_DATE') INTO('JUL_FINAL')'

IF MVSCODE<>0 THEN . . .

This statement will convert a Gregorian formatted date to Julian formatted date.
A compiler feature provides the option to compile REXX source programs into executable object modules.

Supports standard security and authorization.

Supports both interpretive and compiled programs, in any combination.
MAX/REXX Uses

- Create low cost internal MIS applications
- Generate test files
- Prototype online and batch systems
- Solve ad hoc or ongoing production problems
- Develop robust applications
- Resolve DB2 and data administration problems
Max/REXXX Uses

Copy/convert data between DB2, VSAM and SAM

The following sample will:

- Extract data from DB2 table
- Write to an output file using field mapping
- Print output file using field mapping.

```plaintext
/* BEGIN *******************************************/
NRO_ENTER
/* GET ENABED FIELDS */
/
/* ****************************************** */
/
/* Allocate a temp file - if this is running under REXX */
/
/* ****************************************** */
/
IF ADFERSE('REX') THEN DO
/* running under rexx?? */
   temporary = temp name
   tempname = 'TEMP'(REXFNAME,1)
   * temp dname
   * free if not freed
   "ADDT( "(REXFNAME)" ) "NEW INTEGER EXES(REX)"
   "RECH(2,1) EXCHTES".
   * allocate a temp file
   "EXCH(1)"
   * for the output
   "EXTCH(E) EXCH(1) EXCH(1)"
/
/
IF RC=0 THEN DO
   /* ALLOCATE FIELD */
   NEXT 8
/
   END
   /* endif rc=0 */
   /* endif address=sp */
   "NAME FOR BACH
   /* */
```
MAX/REXX Uses

Prepare to Process the Data.

/* Open the output sequential file */
/* put your copybook here */
CENSRT='HXS.P390.EXEC(S(TEMP CB)'
IF EXVSAM("OPEN FILE("TEMPDD") COPYBK("CENSRT") LOAD")<>0 THEN DO
  SAY 'RC='VSAMCODE 'HSG='VSAMMSG
  CALL CLEANUP
/* cleanup and return */
/* endif open<>0 */

01 TEMP-COPYBOOK.
  05 CB-DATABASE-NAME PIC X(12).
  05 CB-DSN-NAME PIC X(12).
  05 CB-NAME-NAME PIC X(20).

/* Connect to a DB2 subsystem */
/* */
CALL "EXSQL" "CONNECT" SUBSYS
IF SQLCODE<>0 THEN DO
  SAY 'EXSQL CONNECT Failed' SUBSYS
  EXIT 8
END
/* */
/* */
MAX/REXX Uses

Main Program Loop required to transfer the data from DB2 to SAM file.

```c
/* Extract name, dname, tname, from system sytables */
/* Set copybook names and write to sequential output file */

CALL "RXXPL" "CURSOR CL CURSOR FOR", /* Declare CURSOR */
    "SELECT NAME, TENANT, TENNAME",
    "FROM SYSTEM TENANT ORDER BY TENANT, TENNAME, NAME"
EXIT CO->0 THEN CALL CURSOR

CALL "RXXPL" "OPEN CL" /* Open CURSOR */
EXIT CO->0 THEN CALL CURSOR

DO WHILE CURSOR("VALUES CL") <>
    /* Loop until problem/NOX */
    CE_TOMBAR_NAME=NAME (TENBAR 1, 12, ' ')
    /* Build fields for */
    CE_TEN_NAME=NAME (TENBAR 1, 12, ' ')
    /* output record */
    CE_NAME_NAME=NAME (TENBAR 1, 20, ' ')

    EXIT "VALUES CL" ("CLOSE") DO /*
        EXIT "VALUES CL" ("CLOSE") /* report the error */
        CALL "RXXPL" "CLOSE CL" /* close the file */
        CALL CURSOR /* cleanup and return */
    END
END
EXIT CO->100 THEN CALL CURSOR /* end if write <>0 */

CALL "RXXPL" "CLOSE CL" /* close the file */
CALL "RXXPL" "CLOSE CL" /* close CURSOR */
```
Display Output
Results

```max
/* Reopen the sequential file and read data back in */
/* */
/* */
IF FOPEN(TEMPD,'O',REXX,NOCTE)#0 THEN DO
  SAY 'PC=VSENDSEG, MSG=VSENDSEG ' report the error
  CALL CLEAR
  cleanup and return
  endif
END

DO WHILE EXIST('TEMPD')=0 /* read the file */
  SAY 'DATABASE=DE DATABASE LNAME ' 
  'NAME=' TEMPD 
END

CLOSE('TEMPD') /* close the file */
end do while

CALL "REXX disconnect"
/* Disconnect from EB2 */
EXIT 0
/* */
RETURN
```

Slide 31 of 42
Use for System Reporting
MAX/REXX can be used to quickly provide reports on system information such as RACF data.

```plaintext
* RACF RECORD TYPE 100
  01 RACF-0100.
  05 RECORD-TYPE-0100 PTC X(4).
  05 FILLER PTC X(4).
  05 GPBD-NAME PTC X(8).
  05 FILLER PTC X(4).
  05 GPBD-SUPGRP-ID PTC X(8).
  05 FILLER PTC X(4).
  05 GPBD-CREATE-DATE PTC X(10).
  05 FILLER PTC X(4).
  05 GPBD-OWNER-ID PTC X(8).
  05 FILLER PTC X(4).
  05 GPBD-UGACC PTC X(8).
  05 FILLER PTC X(4).
  05 GPBD-WTFRMR-CC PTC X(4).
  05 FILLER PTC X(4).
  05 GPBD-USTALL-DATA PTC X(254).
  05 FILLER PTC X(4).
  05 GPBD-CODEL PTC X(4).
```
MAX/REXX Uses

Sample MAX/REXX Program

/* CODE */
/* DO: calculate new base amount for all COLORADO employees */

IF ADDRESS()="T00" THEN "ALLOC F(MySQL) DA/Max_rexx.Begin()"$=

/* OPMX the file using a copybook and the field names can be used to access the data in each field of the record */

IF "COPYBM"("OPEN F(MySQL) COBOL COPYBOOK MAX_REXX_Begin())","SEQ")<>0 THEN DO
   SAY 'BC="WELCOME 'LOC="VAGNAG
   CALL CRENVM
   HRTD
   /* */
   /* use MAX/REXX to list all group information on the MAX database */
   /* */
   DO WHILE "COPYBM"("OPEN F(MySQL) COBOL COPYBOOK MAX_REXX_Begin())","GROUP"
      DATA "GROUP_NAME="GROUP_NAME
      /" group name)
      SAY GROUP_NAME
      DATA GROUP_NAME="GROUP_NAME" /" member
      SAY MEMBER
      DATA MEMBER="MEMBER_NAME" /" member
      SAY MEMBER
      DATA "GROUP_NAME="GROUP_NAME" /" group name)
      SAY GROUP_NAME
      DATA GROUP_NAME="GROUP_NAME" /" member
      SAY MEMBER
      DATA MEMBER="MEMBER_NAME" /" member
      SAY MEMBER
   HRTD
   CALL CRENVM
   HRTD
   CRNVM:
   CALL "REXX" CRNVM
   IF ADDRESS()="T00" THEN EXIT
   ELSE CALL CRENVM
   EXIT 0

33 of 42

Slide 33 of 42
MAX/REXX makes it easier to work with packed numeric data which is difficult to handle in classic REXX.

At OPEN time, data format is associated to a Copybook.

/* REXX */
/* DOC: calculate new base amount for all COLORADO employees */

IF ADDRESS()="TSO" THEN "ALLOC FI(SYSUT1) DA('MXS.TEST.KSDS') SHR" /* */
/* */
/* OPEN the file using a copybook and the field names can be used */
/* to access the data in each field of the record */
/* */
/* */
IF "RXVSAM" ("OPEN FILE(SYSUT1) COPYBOOK(MXS.P390.COPYLIB(CBHDR))", 
" SEQUPD")<>0 THEN DO
  SAY 'RC='VSAMCODE 'MSG='VSAMMSG
END
CALL CLEANUP

This is a sample of RXVSAM which shows opening a file using a COPYBOOK. This allows for access of the data by using the copybook field names.
MAX/REXX Data Conversions

MAX/REXX will convert the packed data from the record so that it can be easily manipulated by simply using the COBOL field names as a REXX variable. Data will be converted to packed format at the time of the rewrite.

```rexx
/* use HRX/REXX to calculate a new base salary amount for all employees in COLORADO. use the WHERE clause for it's high */
/* speed search capability in finding these records. */
/* note that the data in the record is in packed format but */
/* by using the copybook feature of HRX/REXX this calculation can be done by the REXX program */

DO WHILE "EXVSAM"("REDSNXT FILE(SYSUT1) WHERE (63,EQ,'CO')")=0
  EMPLOYEE_AMOUNT=EMPLOYEE_AMOUNT+EMPLOYEE_AMOUNT*.12
  EMPLOYEE_AMOUNT=EMPLOYEE_AMOUNT+.0050 /* round the value */
  EMPLOYEE_AMOUNT=TRUNC(EMPLOYEE_AMOUNT,2) /* truncate to 2 dec */
  CALL "EXVSAM" "REWRT FILE(SYSUT1)" /* rewrite the record */
  IF VSAMCODE<>0 THEN DO
    SAY 'RC='VSAMCODE 'MSG='VSAMMSG
    CALL CLEARNUP
  END
END
END
```

This program does a calculation on packed data and then updates the record.
This is a sample record before & after the EMPLOYEE-AMOUNT has been recalculate.

### RXVSAM OUTPUT
Print of record prior to calculation:

<table>
<thead>
<tr>
<th>EMPLOYEE-RECORD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME-FIRST</td>
<td>A</td>
</tr>
<tr>
<td>NAME-LAST</td>
<td>A</td>
</tr>
<tr>
<td>EMPLOYEE-ADDRESS</td>
<td></td>
</tr>
<tr>
<td>STREET-ADDR</td>
<td>C</td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
</tr>
<tr>
<td>EMPLOYEE-AMOUNT</td>
<td>P</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>JOH</td>
</tr>
<tr>
<td></td>
<td>DOE</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CO</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
</tbody>
</table>

### RXVSAM OUTPUT
Print of record showing the newly calculated value:

<table>
<thead>
<tr>
<th>EMPLOYEE-RECORD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME-FIRST</td>
<td>A</td>
</tr>
<tr>
<td>NAME-LAST</td>
<td>A</td>
</tr>
<tr>
<td>EMPLOYEE-ADDRESS</td>
<td></td>
</tr>
<tr>
<td>STREET-ADDR</td>
<td>C</td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
</tr>
<tr>
<td>EMPLOYEE-AMOUNT</td>
<td>P</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>JOH</td>
</tr>
<tr>
<td></td>
<td>DOE</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CO</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>78</td>
</tr>
</tbody>
</table>
MAX/REXX programming is straightforward and concise. Data conversions to/from DB2 columns and REXX variables are handled automatically.

```
/* REXX */
/* DOG: SQLSRCH: extract data from DB2 table */
/* */
/* Input SQL statement: */
/* "SELECT NAME, DBNAME, TNIGHT" */
/* "FROM SYSTEM.SYSTABLES ORDER BY DBNAME, TNIGHT, NAME" */
/* */
/* */
/* APPL SUBSYS /* GET PASSED PARMS */
/* */
IF SUBSYS="" THEN DO /* Subsys specified? */
   SAY 'DB2 Subsystem not specified - defaulting to DSN'
   SUBSYS='DSN'
END /* */
/* */
/* Connect to a DB2 subsystem */
/* */
CALL "EXSQL" "CONNECT" SUBSYS IF SQLCODE<>0 THEN DO /* DB2 Subsystem not specified - defaulting to DSN */
   SAY 'EXSQL CONNECT Failed' SUBSYS EXIT 8 /* */
END /* */
/* */
/* */
This is a sample of RXSQL that extracts data from a DB2 table & displays it. */
```
A simple loop with one RX/SQL statement can be used to display all the table information.

```sql
/* Extract name, disease, tenure, from system.sysTables */
/* Set variable names and display each retrieved set of variables */
/* */
CALL "REXX" "DECLARE CL CURSOR FOR", /* Declare CURSOR */
"SELECT NAME, DISEASE, TENURE"
"FROM SYSTEM.SYS TABLES ORDER BY TENURE, TENURE, NAME"
/* EXEC SQL CALL CURSOR */
/* /* */
CALL "REXX" "OPEN CL" /* Open CURSOR */
/* EXEC SQL CALL CURSOR */
/* /* */
DO WHILE REXX("FETCH CL")=0 /* Loop until problem/NOX */
DISPLAY REXX("GET NAME") /* Format the data for */
DISPLAY REXX("GET TENURE") /* display */
DISPLAY REXX("GET TENURE") /* */
/* /* */
/* MAX 'Data retrieved = 'TRIM */
/* */
CALL "REXX" "CLOSE CL"
```

Data retrieved = DSMDE06 SYSPKAGE SYSPACKDEP
Data retrieved = DSMDE06 SYSPKAGE SYSPACKLIST
Data retrieved = DSMDE06 SYSPKAGE SYSPACKSTMT
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
Data retrieved = DSMDE06 SYSPKAGE SYSPKSYSTEM
MAX/REXX 1-Compiler

- MAX/REXX compiles REXX source programs into executable object modules.
- The compiled programs may be executed directly from JCL, called from a program, or invoked as a TSO command procedure.
- The compiler provides the same security and change control as other languages such as COBOL or PL1.
- An optional, optimizing compiler is available for even greater performance.
### RSYSAM Open and Close Statements
- **OPEN**: Open a file
- **CLOSE**: Close a file

### RSYSAM PDS Specific Access & Processing Statements
- **DIR**: Retrieve the directory information
- **FIND**: Position to a member within the PDS
- **ADDMEM**: Add a new member to a PDS
- **REPMEM**: Replace a member of a PDS
- **DELMEM**: Delete a member of a PDS
- **RENAME**: Rename a member of a PDS

### RSYSAM Record Access and Positioning Statements
- **DELETE**: Delete a record
- **READ**: Read a record direct mode
- **READNEXT**: Read next record
- **READPREV**: Read previous record
- **REWRITE**: Update a record
- **STARTER**: Start sequential processing
- **ENDBR**: End sequential processing
- **WRITE**: Add a record

### RSYSAM PDS Processing Statements
- **DIR**: Loads PDS Dir entries into variable array
- **ADDMEM**: Add a new member directory entry
- **REPMEM**: Replace a member directory entry
- **DELMEM**: Delete a member directory entry
- **RENAME**: Rename a member
- **FIND**: Position to begin of a member
- **READNEXT**: Read forward to next record
- **REWRITE**: Update record
- **WRITE**: Write a new record
DB2 Connection Statements
CONNECT
DISCONNECT

SQL Object Manipulation Statements
ALTER INDEX
ALTER TABLESPACE
ALTER TABLE
CREATE INDEX
CREATE TABLESPACE
CREATE TABLE
CREATE VIEW
DROP

SQL Administration Statements
GRANT
{PLAN PRIVILEGES}
{SYSTEM PRIVILEGES}
{TABLE OR VIEW PRIVILEGES}
{USER PRIVILEGES}

REVOKE
{DATABASE PRIVILEGES}
{PLAN PRIVILEGES}
{SYSTEM PRIVILEGES}
{TABLE OR VIEW PRIVILEGES}
{USER PRIVILEGES}

SET CURRENT SQLID

SQL Query Statements
DECLARE CURSOR
OPEN CURSOR
FETCH CURSOR
CLOSE CURSOR
SELECT INTO HOST VARIABLE
SELECT INTO TABLE
SELECT INTO TABLE VARIABLE ARRAY

SQL Data Update Statements
DELETE
{Positioned DELETE} WHERE CURRENT OF
INSERT
UPDATE
{Positioned UPDATE} WHERE CURRENT OF
The following is a sample list of companies that are users of this technology.

America West Airlines  National Cancer Institute
Anderson Consulting   National Institute of Health
BP Oil Company       Perot Systems Corporation
Centers for Disease Control   Raytheon E-Systems, INC
Deutsche Bank       REEBOK
Defense Mega Centers  Schweizerische Mobiliar
DOW Chemical             State of Minnesota
Ernst & Young LLP    Texas Farm Bureau
Fleet Mortgage          TRW/BDM-Honeywell
General Foods/Kraft    U.S. Army Aviation & Missile Command
MITSUBISHI Australia  West Deutsche Landesbank/West LB