BCI433 NOTES

Lab1

Fill in the blanks or select True or False in the space provided:

1. IBM AS/400 was designed to support for large or huge business. True / False false

2. To sign on to power systems, you must have a valid \_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_. password and userid

3. The CL command to go to the menu PROGRAM: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. go program

4. The CL command to run program INTROLAB: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. call introlab

5. The function key to cancel a command or an operation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.f12

6. Everything on power systems that has a name, takes up space and is not of an \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.object

7. Source physical file objects contain source code as \_\_\_\_\_\_\_\_\_\_\_\_\_. program

8. Formatted output is stored in an \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ waiting for you to decide to print it or not. Spooled files

Question 2:

Briefly answer the following questions:

1. What is the field exit key for entering a negative number?

2. What does a source physical file is used for? 2- Source physical file – object that stores program

source code

3. An object in IBM i has an object type of \*PGM. What does it means to the object? 3- program

4. What are the steps to create a CL program in IBM i ACS? 4- WRKMBRPDM QCLLESRC

5. Is a spooled file an object? Why? 5- yes because take place in acs

6. Where is the default for your system libraries (of the library list) stored? 6- in the current library

q1- where does the ibmi store objects if you do do not specify ?

\* curlibl

q2 editcode does not change the actual pf field?

fals

Q3- write the cl command to list all objects in alibrary dc433?

WORKOBJPDM DC433

Q4- write cl command to enter data in My file file.pf.data ?

update myfile

Q5- change your current library to SENECA moniter library message not found ?

CHGCURLIB SENECA

MONMSG (LIBRFARY NOT FOUND)

Q6-name 2 ways to add seneca library list on RDI?

1-right click on objects properties initial library then add the library name and the position

2- creat clle program then add the command (addlible seneca )as apart of the program code . b 12:45 PM 2021-06-20

Q6- how and when IBMI builds the library list ?

# Week 2 Quick Check Answers

## Fill in the blanks and True/False

1. In RDi or Eclipse, a perspective has (composed of): **views**
2. It's important that you protect your RDi workspace. So you keep it on your USB drive. This way makes you have a backup of your source code. **True**
3. The CL command WRKOBJ STUDENTS will show a list of STUDENTS objects that you have **authority** to use.
4. The CL command to show your library list **DSPLIBL**
5. The CL command to add BCI433LIB library to your library list **ADDLIBL**
6. The CL command to list all objects which have the same name as your user ID **WRKOBJ user\_name**
7. What are the object types you see after you run the CL command in last question?   
   **OUTQ, MSGQ, USRPRF, LIB**

## Short Answer Questions

Step 1. EDITCODES is a program in Library BCI433LIB. When running the program (CALL EDITCODES), we got the message: “Program EDITCODES in Library \*LIBL not found”. What does it mean about BCI433LIB?

**BCI433LIB was not our current library**

Step 2. For the situation in last question, give three different solutions to run the program.

* Change the current library to BCI433LIB and then CALL EDITCODES
* CHGCURLIB BCI433LIB
* CALL EDITCODES
* Call the program with a qualified name
* CALL BCI433LIB/EDITCODES

Step 3. Why you need to call the program STRJOB in ACS or RDi when doing your labs?

This program will display your name along with your user ID on all spooled file pages to ensure that you are indeed the one who created the spooled the spooled file. Thus, if it's a compilation output, we can guareentee that you wrote the code

Step 4. How and when library list created?

It's created when you sign on and it's deleted when you signoff. The system administator decides what libraries are going to be in the list upon signingin by changing the value of QSYSLIBL(contains initial system libraries) and QUSRLIBL(contains initial user libraries) system values.

Step 5. In RDi, when you compile a program but the RDi has no response. What has happened? Or what you should do to fix the problem?

A previous (compiled) version of the object is in use (that is, a program on ACS is running, locking the program object file itself and any other files it's using). To fix the problem, end the program using the objects you plan on replacing.

Step 6. What are the three different ways to show your spooled files, e.g. a compiler listing?

* WRKSPLF then find the required spooled file
* DSPSPLF FILE(spooled file name here) SPLNBR(\*LAST)
* On ACS, under the tool bar (File, Edit, View ...), there is a list of icons. Select the 8th icon from the left. This will show a new window with all your spooled files. Click on the one you want. **This is the preferred way to show a spooled file to a professor**

Step 7. What are the three different ways to show or query data in a physical file?

* RUNQRY \*N file\_name
* Right click on a file with a \*file.pf-dta extension in the Remote Systems window and select Show in Table and select Data.

Lab3

# Week 3 Quick Check Questions

## Fill in the blank and true/false

1. DDS (Data Description Specifications) is a traditional mean to describe **data attributes** on IBM i.
2. In QDDSSRC.\*file.pf-src, you create, code and save a member name STUDDSPF with the DSPF type. That means you've created a display file – STUDDSPF. **True**
3. In DDS, we use EDTCODE to edit **numeric fields** fields in display or printer files.
4. In RPGLE, \*INLR means **Last Record Indicator**
5. In the DDS code of Lab 3, we use the keyword (or function) **Position Cursor (DSPATR(PC))** to set up data validation to the field MARRIED ('Y' and 'N').
6. In the source code (LPEX) editor, free-format RPGLE code can be anywhere after column **7**
7. A Display file statement in an RPGLE program requires a **DCL-F (Declare File)** statement.

## Short answer questions

Step 1. What are the three types of a (named) field?

* Named
* Reference

Step 2. Name 4 files that we can create with DDS code?

* Printer files
* Display files
* Physical files
* Logical files

Step 3. In DDS, an indicator, e.g. 90, can apply to a named field or to the field's attributes (e.g. 'Protected'). What are the different results?

* If applied to a named field, the field will display when the indicator is on
* If applied to an attribute, the field will have that attribute when the indicator is on.

Step 4. In RDi Screen Designer, a display file's field shows like 666,666,666. What kinds of properties or keywords are on the field?

* 9 for length
* Y - numeric
* Edit code is 1 (EDTCDE(1))

Step 5. What are the difference(s) between EXFMT and WRITE in RPGLE syntax?

* Write -> Displays screen record
* EXFMT -> Displays screen record and prompts for user input

Step 6. How to declare a externally-described display file STUDMARKS.\*file.dspf in RPGLE program?

DCL-F STUDMARKS WORKSTN;

Lab4

# Week 4 Quick Check Questions

## Fill in the Blank and True/False

1. RPGLE built-in function names start with: **'%'eof**
2. The CL command to set a program to DEBUG mode so you can set breakpoints is: **STRDBG**
3. We use the CL command MonMsg to check for your chat with other users. **FALSE**
4. CHGUSRPRF is the CL command to allow you to setup initial program for your ACS. **FALSE**, We use CHGPRF to change our initial program, CHGUSRPRF is for administrators

## Short Answer Questions

Step 1. How to get today's Date in RPGLE programs?

%DATE()

Step 2. Write RPGLE statement to calculate the days left to the end of the summer term(Aug. 14).

%DATE(D'2020-08-14' : %DATE() : \*DAYS)

Step 3. Write 3 items of user's info that's stored in a user's profile.

* Initial Program
* Initial Menu
* Password

Step 4. If a user cannot login due to the wrong initial program of the user profile. How the user should fix it?

On the sign on screen, they can enter \*NONE for the initial program

# Week 5 Quick Check Questions

## Fill in the blanks

1. The first record format in a printer file DDS code is an **Absolute** record.
2. A printer file is an object whose object type is **\*file**
3. The constant in Palette that is solely available in Report Designer for Printer file (not for screen files) is: **Page Number**

## Short Answer Questions

Step 1. In RDi Report Designer, what is the easiest way to create a data field in a record format from a db table's column?

Naviagate to the table in the remote systems view, right-click, show in table -> fields. Then drag a field from the table into the display file to create a data field (or reference field).

Step 2. What is the purpose to create the design report in RDi Report Designer?

Displays all our created record formats all at once, useful for ensuring alignment between fields part of different record formats.

Step 3. The two physical files SHIFTWEEK and SHIFTRATES are in library SENECAPAY. What you should do before compiling or running the RPGLE program?

Ensure that SENECAPAY is in library list

Step 4. Write a CL command to override a database file and give an explanation.

OVRDBF \_\_ORIG-DB-FILE\_\_ \_\_OVERRIDE-DB-FILE\_\_ When this statement is ran, ORIG-DB-FILE will replace OVERRIDE-DB-FILE instead

Lab6

# Week 6 Notes

There 2 kinds of fields in record formats: §

Named fields – similar to the ones in display/screen files §

(data) fields – can be created by dragging from database table's columns, i.e. physical file's fields in RDi

Lab7

Dynamic Program Calls: ► a program calls another program: § An OPM program can call another OPM program or an ILE program § An ILE program can call an OPM program or another ILE program § Static Procedure Calls: ►used to call: § a procedure within the same module § a procedu

Lab8

# Lab Question Week 8

1. When you display the players to the user and the user goes back to the main  
   menu and decides to display the players again, you need to remove the data  
   in the subfile. How did you do this done?  
   **A. Turn on the SFLCLR attribute and WRITE to the control record format to effectively clear the data**

Lab9

# Week 9 general notes

## Embedded SQL

We use an SQL precompiler directive to inform the compiler that what's comming up next is SQL.

Example:

EXEC SQL

SQLSTATEMENT;

**NOTE:** Make sure the file/member that contains your source code is of type: SQLRPGLE. If the file type is RPGLE, the SQL precompiler directive will show up as an error.

### More information

At compile time, the SQL precompiler that will look through all the code and every time it encounters a directive it knows there is a SQL statement. The SQL precompiler will translate that SQL statement into RPGLE code so that the RPGLE compiler is able to parse the code.

## Host variables

When we refer to an RPGLE variable inside of an SQL statement we refer to it as a host variable. Host variables have a colon(':') prepended to their name when they are in use in an SQL statement.

### Externalally described data structure

To bring in all the host variables(RPGLE variables) in at compile time, we can declare a data structure based on a externally described table:

DCL-DS CUSTOMER EXT END-DS;

This one line of code at compile time will bring in all the fields in a table and define them properly. You'll get the right field name, type, size. Once we do this, we can refer to all the fields in the table referenced as host variables in our SQL statements.

## Meanings

* DCL-DS -> declare data structure
* CUSTOMER -> external file name we want to reference
* EXT -> specifying that the name CUSTOMER is in a fact an externally described file.
* END-DS -> End data structure.

## Indicator variables

Used in situations where we have the possibility of the retrieved value(s) from a record could be null. If The value retrieved from a specific column happens to be null, **The assignment cannot happen** Thus, an indicator variable is required to know when we get a null value (and attempted to assign to our host variable(s))

We define standalone indicator variables for this.  
When defining an indicator variable, it must be of type binary and have decimal size of 4 (no precision).

DCL-S SHIPCITYNULL BINDEC(4:0);

DCL-S DISCOUNTNULL BINDEC(4:0);

When an attempt to store a value of NULL into a host variable, the indicator variable was take on the value of -1 (0 otherwise).

### Definition of NULL

**NULL means unknown value**

### Example of embedded SQL

Reminder: because we declared a CUSTOMER data structure, we get all the host variables defined ready to use.

EXEC SQL // SQL precompiler

SELECT CUSTID, // all selected columns must have a corresponding

NAME, // variable.

SHIPCITY,

DISCOUNT

INTO :CUSTID,

:NAME,

:SHIPCITY :SHIPCITYNULL, // Note: the space between the

:DISCOUNT :DISCOUNTNULL // host vars and indicator vars

FROM BCI433LIB/CUSTOMER

WHERE CUSTID = :CUSTID

The host variables are in the same order as the column names.

## SQL Communications Area

When you have an embedded SQL statement in an RPGLE program, you should always check for possible errors that could occur. We get feedback from a recent SQL statement from the SQL Communications Area. It provides fields that we can check to find what happened with the recent SQL statement that had executed.

A lot can go wrong with the execution of SQL statements.

### SQLCODE

* < 0 - ERROR
* = 0 - SUCCESSFUL execution (may have a warning, and thus we should still check for warnings).
* 0 - SUCCESSFUL execution with a definite warning

examples:

* 100 - Row not found -> **common and used in the lab**
* 811 - More than one row (happends with a SELECT ... INTO ... statement)

### SQLWN

One digit follows SQLWN to specify the warning. It could be any digit from 0-9.

SQLWN0 is a catch all for all other SQL warn fields (that is, if any other SQLWN? field has a value of 'W', then SQLWN0 with have a value of 'W' as well)

#### Sample code to check for error (SQLCODE and SQLWRN)

IF (SQLCODE <> 0) OR (SQLWN0 = 'W'); EXSR SQLPROBLEM; ENDIF;

### SQLSTATE -> modern technique to get feedback

SQLSTATE returns 5 characters with the meanings:

* '00000' - SUCCESSFUL EXECUTION
* '02000' - ROW NOT FOUND
* '01???' - WARNING
* ANYTHING ELSE - some other ERROR

#### Example of SQLSTATE

SELECT

WHEN SQLSTATE = '00000';

/\*

\* When SQLSTATE is all 0's, we want to get out of this SELECT

\* routine. One technique is to do an assignment to a DUMMY

\* field.

\*/

DUMMY = 1;

WHEN SQLSTATE = '02000';

EXSR ROWNOTFOUND;

WHEN %SUBST(SQLSTATE:1:2) = '01'; // '01' as the first two digits

EXSR WARNING; // denotes some kind of warning

OTHER;

EXSR SQLERROR;

ENDSL;

Lab9B

# Week 10 Lecture Notes

## General Concepts

We can't use a SELECT ... INTO ... statement when we want to retrieve more than one row.

SQL allows for this with a **cursor**.

With a Cursor you must perform the following tasks:

1. Declare the Cursor - Like a DCL-F (declare file) statement This statement can be anywhere before the OPEN statement
2. Open the Cursor  
   This produces a temporary result table (that is, the SELECT statement specified in the DECLARE CURSOR statement is actually executed.
3. Fetch rows from the Cursor (in a loop) Every iteration we retrieve a row from the temporary result table
4. Optionally update, delete, or report on the most recent fetched row (in a loop)
5. Close the cursor When finished processing all the rows in the temporary result table, the cursor must be close to free the temporary result table from memory.

Declare file (DCL-F) vs DECLARE CURSOR:

Unlike the DCL-F statement, A cursor declaration can specify row selection, derived columns, union and other data manipulations.

AND:

When you use DCL-F, you cannot include the library name, you must ensure that the library is in your library at compile time. When you declare a cursor, you can have a qualified name so you don't need to worry about the library in your library list

# Lab 9B Notes

## Sample snippets managing a cursor at each step

Step 1. Declare the cursor

EXEC SQL // inform precompiler that this is SQL

DECLARE CustomerCusor CURSOR // CURSOR keyword

FOR // indicates a cursor

SELECT \*

FROM BCI433LIB/CUSTOMER

ORDER BY CUSTID;

Step 2. Open the cursor

EXEC SQL

OPEN CustomerCursor;

/\*

\* This is something procedurally executed and happens

\* strategically at a specific point in my program which may not

\* work properly. Thus we must check the SQL communications area

\*/

IF (SQLCODE <> 0) OR (SQLWN0 = 'W')

EndOfFile = \*ON;

ENDIF;

**NOTE:** ENDOFFILE is a custom indicator variable declared like so:

DCL-S ENDOFFILE IND;

Named indicator variables can make code easier to understand so long as they have a meaningful name.

We can use these named indicator variables anywhere in the program, akin a boolean flag in c/c++.

In this particular case, we use ENDOFFILE to control how long we will stay in our loop where we are to process the data.

Step 3. Fetch rows (inside loop)

DOW NOT(ENDOFFILE);

// ...

EXEC SQL

FETCH NEXT

FROM CustomerCursor

INTO :CustId, :Name, :ShipCity, :ShipCityNull,

:Discount, :DiscountNull;

IF (SQLCODE <> 0) OR (SQLCODE = 'W')

ENDOFFILE = \*ON; // If there was a problem, I want

ENDIF; // to exit the loop

// ...

ENDDO;

Step 4. Once you fetch a row from a cursor, you process the info in some way.

Step 5. Free the temporary result table from memory.

EXEC SQL

CLOSE CustomerCursor;

IF (SQLCODE <> 0) OR (SQLWN0 = 'W')

/\*

\* failed to close cursor

\* Handle in some way, such as informing the user (by showing it

\* in a display that something went wrong) or display an error

\* in the report.

\*/

ENDIF;

### Code snippets used in the program

It's always a good idea to test select statements you write interactively  
(STRSQL to open the interactive menu).

After some testing, this is the select statement we have created,

SELECT custid, trim(FName) || ' ' || LName AS FullName, // triming to remove

city, purchase\_, pdate, // trailing whitespace

'ontario' AS Province // we want the province of the customer to show

FROM BCI433LIB/ONTARIO // which we use in the report (main routine logic)

WHERE Purchase\_ > 400;

UNION ALL // combines the tables without ordering, to help create the report

SELECT custid, trim(FName) || ' ' || LName AS FullName,

city, purchase\_, pdate, 'Quebec' AS Province

FROM BCI433LIB/QUEBEC

WHERE Purcahse > 400;

UNION ALL

SELECT custid, trim(FName) || ' ' || LName AS FullName,

city, purchase\_, pdate, 'Manitobs' AS Province

FROM BCI433LIB/Manitoba

WHERE Purchase\_ > 400;

**Note**: A variable instead of the hard coded 400 will be present in the final version to be more robust.

Procedure interface

// Procedure interface

// LowLimitIn passed to the program

DCL-PI Main extPgm('PROVSQL'); // PROVSQL is the program name

LowLimitIn Packed(15:5);

END-PI;

When you pass a number to a program from the command line, you need to declare  
it as 15 digits with 5 decimal positions. If you don't the program will crash.

**Data Structure that's used**

**The order of the fields in the data structure should match the order of the col names in the SELECT statement.**

DCL-DS CustomerRecord;

CustID Char(6);

FullName Char(31);

City Char(20);

Purchase\_ Packed(7:2);

PDate Date;

Province Char(10);

END-DS CustomerRecord;

We store the inputted number from the command line into another variable of a  
smaller type that's defined more realistically. We narrow/truncate the value.

C/Eject

/Free

LowLimit = LowLimitIn;

//...

This subroutine uses a particular strategy of declaring the cursor and then opening it right after.

BEGSR PREPAREFILES;

EXEC SQL

DECLARE ALLPROVCURSOR CURSOR

FOR

//...

EXEC SQL

OPEN ALLPROVCURSOR;

// checking to see if there was a problem opening the cursor

IF (SQLCODE <> 0) OR (SQLWN0 = 'W');

ENDOFFILE = \*ON;

ENDIF;

ENDSR;

Since the fields in CUSTOMERRECORD data structure are in the same order as the column names in the SELECT statement, we can specify the data structure name itself and it will properly fill in all the data members.

If the order of the fields in the data structure did not match the columns in  
the SELECT statement, we would have had to define each member individually.

BEGSR GETROW;

EXEC SQL

FETCH NEXT

FROM ALLPROVCURSOR

INTO :CUSTOMERRECORD;

IF SQLCODE <> 0 OR SQLCODE = 'W';

ENDOFFILE = \*ON;

ENDIF;

ENDSR;

In the main routine:

DOW NOT (ENDOFFILE)

//...

IF IN01;

// ...

ENDIF;

IF PROVINCEH = PROVINCE; // province is the same

Write Detail;

ELSE; // new province detected ...

ProvinceH = Province;

Write totals;

write newprov;

write detail;

ENDIF;

Ending the program by cleaning up and obtaining some aggregated data.

EXEC SQL

CLOSE ALLPROVCURSOR;

// ... approx. 28 more lines here

ENDSR;

Lab10

# Lab Question

1. DayNumName is user-defined and thus the RPGLE compiler does not recognise it. What do you have to do to let the compiler know that it's a legitamite name? **A. We needed to create a prototype**