

SQL/PRO 5.0



Table of Contents

INSTALLATION INSTRUCTIONS	4
STEP #1	4
STEP #2	4
STEP #3	5
STEP #4	5
STEP #5	5
STEP #6	6
STEP #7	6
STEP #8	7
STEP #9	7
STEP #10	8
STEP #11	8
WHAT IS SQL/PRO	9
SQL STATEMENT ENTRY	9
FILE FIELD NAME SELECTION	9
REPORT FORMATTING	10
HISTORY	10
SAVED QUERIES.....	10
EMBED SQL IN CL	10
USING SQL/PRO.....	11
STARTING SQL/PRO.....	11
ENTERING SQL STATEMENTS	11
SUBSETTING THE SQL S TATEMENT HISTORY	12
RUNNING THE SQL STATEMENT	13
SELECTING THE OUTPUT DEVICE	13
SELECTING THE EXECUTION ENVIRONMENT	14
HISTORY	15
SAVED QUERIES.....	16
RUNNING A SAVED QUERY	18
CHANGING A SAVED QUERY	19
COPYING A SAVED QUERY.....	19
DELETING A SAVED QUERY	19
SERVICES	19

RUNNING A SAVE QUERY FROM THE COMMAND LINE.....	20
RUNNING A SAVED QUERY FROM A CL PROGRAM	21
EMBEDDING SQL STATEMENTS IN CL PROGRAMS	21
RETRIEVING A SAVED QUERY INTO A VARIABLE	22
CONVERTING QUERY/400 QUERIES TO SQL/PRO	23
EMBEDDED VARIABLES	25
FORMATTING YOUR REPORTS	26
Advanced Report Formatting Techniques	28
Address Formatting.....	30
SQL/PRO AUTHORITY	31
ACCESSING SQL/PRO AUTHORITY.....	31
CONFIGURING THE SQL AUDIT FUNCTION	33
UPDATING SQL/PRO.....	35
UNINSTALLING SQL/PRO.....	36

INSTALLATION INSTRUCTIONS

STEP #1

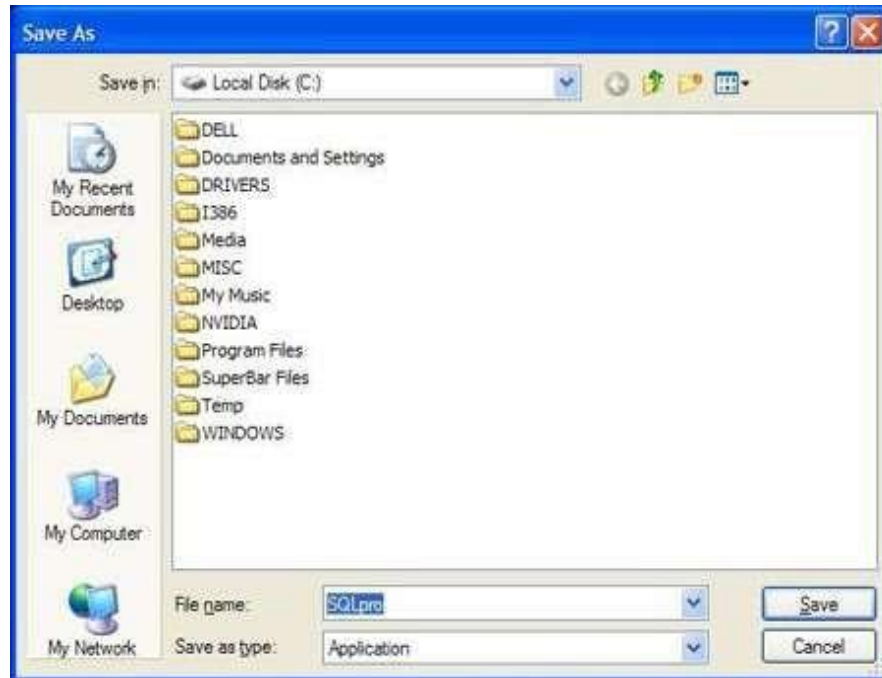
Once you have clicked on the link to start the download process, the following screen will appear.



Click on the save button to save the installation file to your local computer.

STEP #2

Once you have clicked on the Save button, choose the directory C:\Temp on your local computer as the location where the installation file will be saved. Click on the Save button to save the installation file in the selected directory.



STEP #3

Once you have clicked the Save button, the installation file will be downloaded to your local computer. A screen showing the progress will appear. Please be patient during this process.



STEP #4

Once the installation file has been completely downloaded. Open your Windows Explorer, navigate to the C:\Temp directory, Double Click on the installation file to launch the installation process.

STEP #5

A splash screen will appear and a series of notices informing you of the process being performed. After which the following screen should appear. Click on the Next button to continue the installation process.



STEP #6

The installation process requires a connection being established to your IBM i (iSeries, AS/400) host computer. The following notice may appear informing you for the need of a connection. Once you have verified the connection to your IBM i (iSeries, AS/400) host, click the OK button to continue the installation process.

Note: This installation requires that your PC has an active network connection to your iSeries and that the FTP server on the iSeries is active. You can activate the FTP server on your iSeries by running "STRTCPSVR *FTP" from a command line.

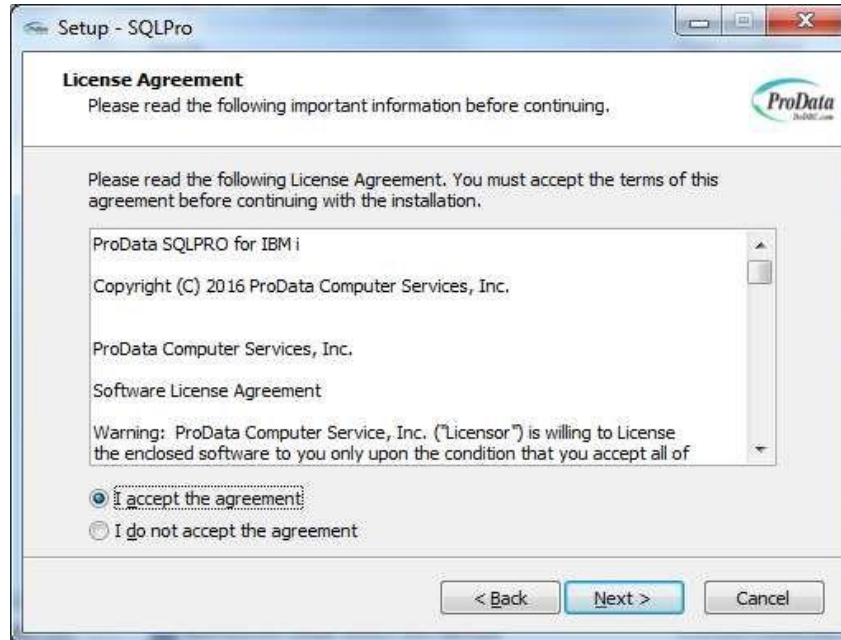
STEP #7

Initial welcome screen. Click next to continue.



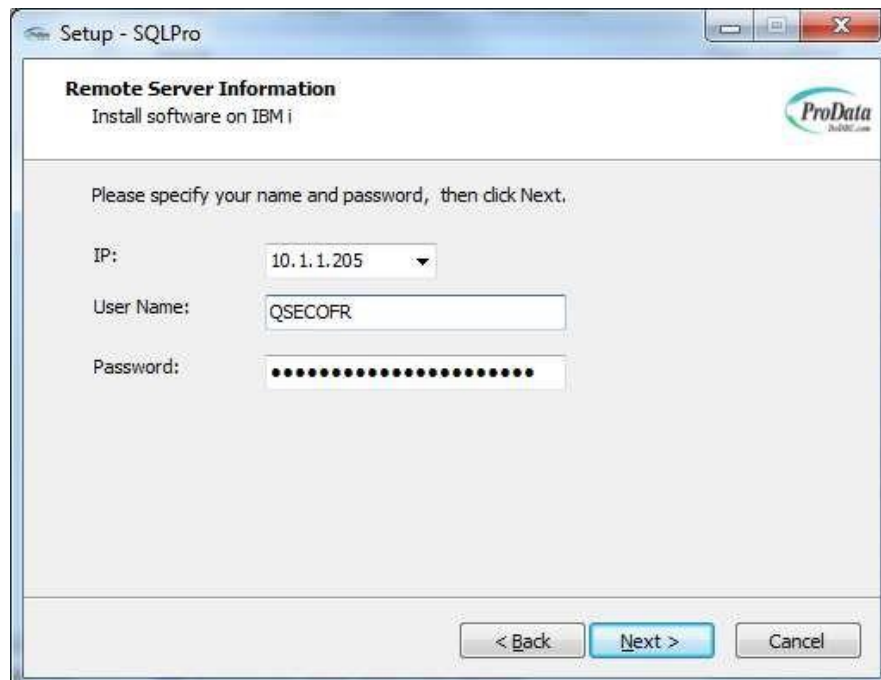
STEP #8

Please read the License Agreement and upon accepting the agreement. Click the Yes button to continue the installation process (see image on next page).



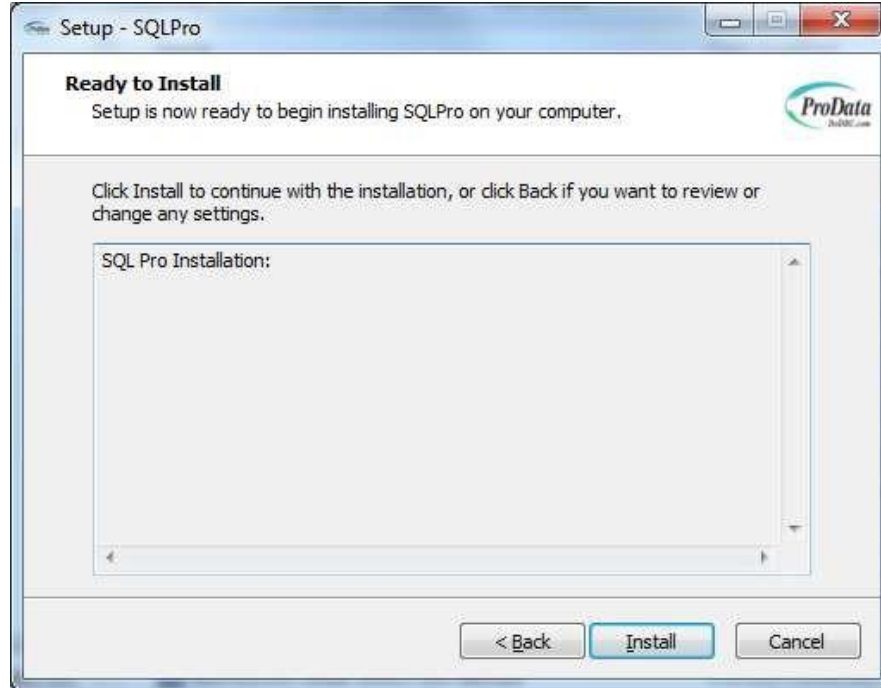
STEP #9

Either select the iSeries (AS/400) host computer where you wish to install SQL/Pro, or type the IP Address in the space provided of the iSeries (AS/400) host computer, the User name and Password. Once you have completed the iSeries (AS/400) host computer selection, click the Next button to continue the installation process.



STEP #10

A confirmation screen will appear when ready to install SQLPRO on your IBM I (iSeries, AS/400) host computer.



STEP #11

The last screen that should appear is the Completion screen. Click the Finish button to complete the installation process.



WHAT IS SQL/PRO

Structured Query Language (SQL) is a powerful query language that now is supported on virtually every computer made. **SQL/PRO** brings the power of SQL to your IBM i (iSeries, AS/400) for the lowest cost.

SQL provides users - both technical and non-technical - with a simple English-like language that gives them the power to query, print and manipulate any data. You'll be able to select, organize and summarize your information in nearly any way you can imagine - certainly any way you'll ever require! You'll be able to direct your output to the screen, printer or database file. And programmers will appreciate the power of SQL to update files.

How easy is SQL to run? The following statement will alphabetically list all records of the CUSTMAST file where state equals California:

```
SELECT * FROM CUSTMAST  
  
WHERE STATE='CA'
```

```
ORDER BY  
STATE
```

What could be simpler?

Our **SQL/PRO** interface provides truly easy-to use screens that will have you accessing the power of SQL in no time.

But don't let the ease-of-us (or the low price) fool you - **SQL/PRO** is powerful! With field name selection, unlimited history, our unique Saved Queries feature, submission to batch and comprehensive services, **SQL/PRO** is a mighty tool.

SQL STATEMENT ENTRY

The heart of **SQL/PRO** is the interactive SQL statement entry screen, where you key in any SQL statement you wish to run. The insert, split, copy, delete, split, and join function keys make it easy to enter statements. You'll have a full 5,080 characters (40 lines by 127 characters) if using 132 column mode or a full 2,550 (x lines by 74 characters) if using 80 column mode to run your SQL statement. Once the statement is entered you can run it interactively or in batch from this screen.

FILE FIELD NAME SELECTION

SQL statements require that you specify the names of fields with which you will be working. With **SQL/PRO** there's no need to remember field names; just press a function key and you can display the field listing of any file on your system. From this screen, you can select fields to be automatically inserted into your statement.

REPORT FORMATTING

When output is directed to a printer SQL does very little in way of report formatting. However, **SQL/PRO** offers you comprehensive report formatting by creating what is known to QS/400 as a Query Management form (*QMFORM) object.

Report formatting criteria is submitted through a series of prompt screens that are automatically linked to your current SQL SELECT statement. The objects created by **SQL/PRO** are the same objects used by IBM's SQL/400 Query Manager so, if you ever decide to purchase IBM's SQL/400, all your queries are immediately usable by the IBM product.

HISTORY

SQL/PRO remembers every statement you run for as long as you desire. You can browse through the history to look for previous statements and rerun them. The browsing display shows the first 65 characters of the statement. From there, you can select to retrieve the entire SQL statement to optionally modify and re-run. At any time you can purge the history field down to any number of entries.

SAVED QUERIES

The Save Queries feature is unique to **SQL/PRO**. You can name and save any SQL statement you have written. Public access can be indicated for any saved queries making it easy to narrow the users that can use, change or delete query. Queries can be saved directly from the interactive entry screen or indirectly through the history screen. Once saved, you can select and run a Saved Query from the Work With Saved Queries screen interactively or via a job queue. The screen shows all the saved queries alphabetically ordered in a subfile. A special command also lets you run a saved query from a CL program.

EMBED SQL IN CL

A **SQL/PRO** command lets you embed SQL statements in CL, something IBM's SQL/400 won't allow. You can run any SQL statement or an already-created Saved Query from a CL program. Parameters allow full control of the outfile selections. Think about how much easier this is than writing cryptic OPNQRYF commands.

USING SQL/PRO

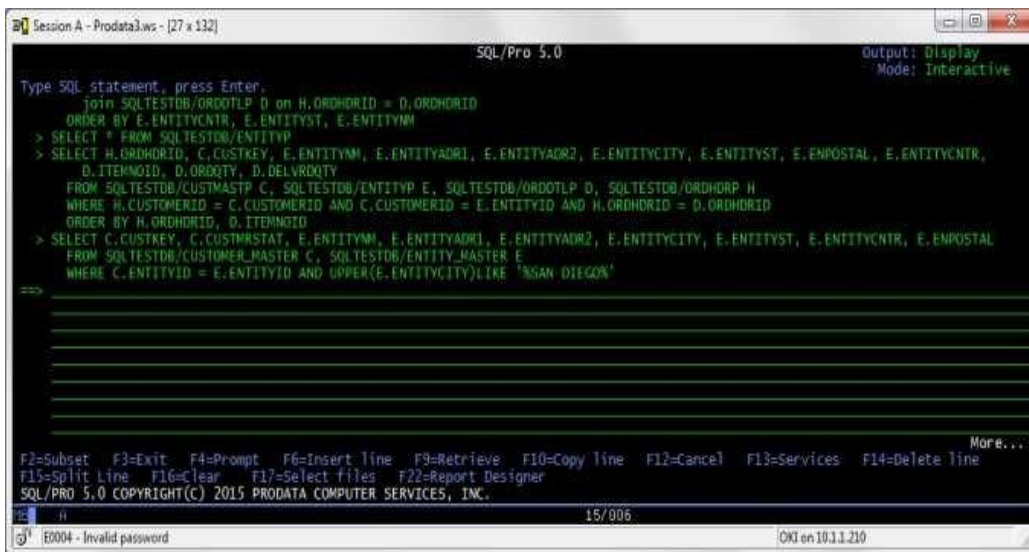
STARTING SQL/PRO

Make sure that you have added the library SQLPRO to your library list. To start **SQL/PRO**, enter STRPDSQL at the command line.

ENTERING SQL STATEMENTS

The following statement entry panel will be displayed once you start **SQL/PRO**. From here you will enter your SQL statements.

132 Column Display

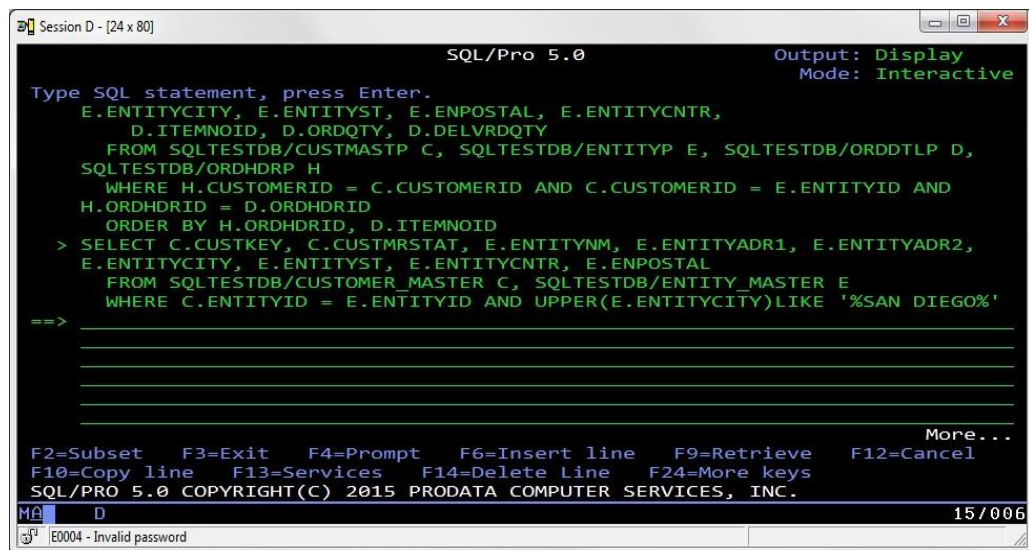


```

Type SQL statement, press Enter.
Join SQLTESTDB/ORDOTLP D on H.ORDHDRID = D.ORDHDRID
ORDER BY E.ENTITYCNTR, E.ENTITYST, E.ENTITYNM
> SELECT * FROM SQLTESTDB/ENTITYP
> SELECT H.ORDHDRID, C.CUSTKEY, E.ENTITYNM, E.ENTITYADR1, E.ENTITYADR2, E.ENTITYCITY, E.ENTITYST, E.ENPOSTAL, E.ENTITYCNTR,
D.ITEMNOID, D.ORDQTY, D.DELVRDQTY
FROM SQLTESTDB/CUSTOMASTP C, SQLTESTDB/ENTITYP E, SQLTESTDB/ORDOTLP D, SQLTESTDB/ORDHDRP H
WHERE H.CUSTOMERID = C.CUSTOMERID AND C.CUSTOMERID = E.ENTITYID AND H.ORDHDRID = D.ORDHDRID
ORDER BY H.ORDHDRID, D.ITEMNOID
> SELECT C.CUSTKEY, C.CUSTMRSTAT, E.ENTITYNM, E.ENTITYADR1, E.ENTITYADR2, E.ENTITYCITY, E.ENTITYST, E.ENTITYCNTR, E.ENPOSTAL
FROM SQLTESTDB/CUSTOMER_MASTER C, SQLTESTDB/ENTITY_MASTER E
WHERE C.ENTITYID = E.ENTITYID AND UPPER(E.ENTITYCITY) LIKE '%SAN DIEGO%'
==>

F2=Subset F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line F12=Cancel F13=Services F14=Delete line
F15=Split Line F16=Clear F17=Select files F22=Report Designer
SQL/PRO 5.0 COPYRIGHT(C) 2015 PRODATA COMPUTER SERVICES, INC.
15/006
E0004 - Invalid password
  
```

80 Column Display



```

Type SQL statement, press Enter.
E.ENTITYCITY, E.ENTITYST, E.ENPOSTAL, E.ENTITYCNTR,
D.ITEMNOID, D.ORDQTY, D.DELVRDQTY
FROM SQLTESTDB/CUSTOMASTP C, SQLTESTDB/ENTITYP E, SQLTESTDB/ORDOTLP D,
SQLTESTDB/ORDHDRP H
WHERE H.CUSTOMERID = C.CUSTOMERID AND C.CUSTOMERID = E.ENTITYID AND
H.ORDHDRID = D.ORDHDRID
ORDER BY H.ORDHDRID, D.ITEMNOID
> SELECT C.CUSTKEY, C.CUSTMRSTAT, E.ENTITYNM, E.ENTITYADR1, E.ENTITYADR2,
E.ENTITYCITY, E.ENTITYST, E.ENTITYCNTR, E.ENPOSTAL
FROM SQLTESTDB/CUSTOMER_MASTER C, SQLTESTDB/ENTITY_MASTER E
WHERE C.ENTITYID = E.ENTITYID AND UPPER(E.ENTITYCITY) LIKE '%SAN DIEGO%'
==>

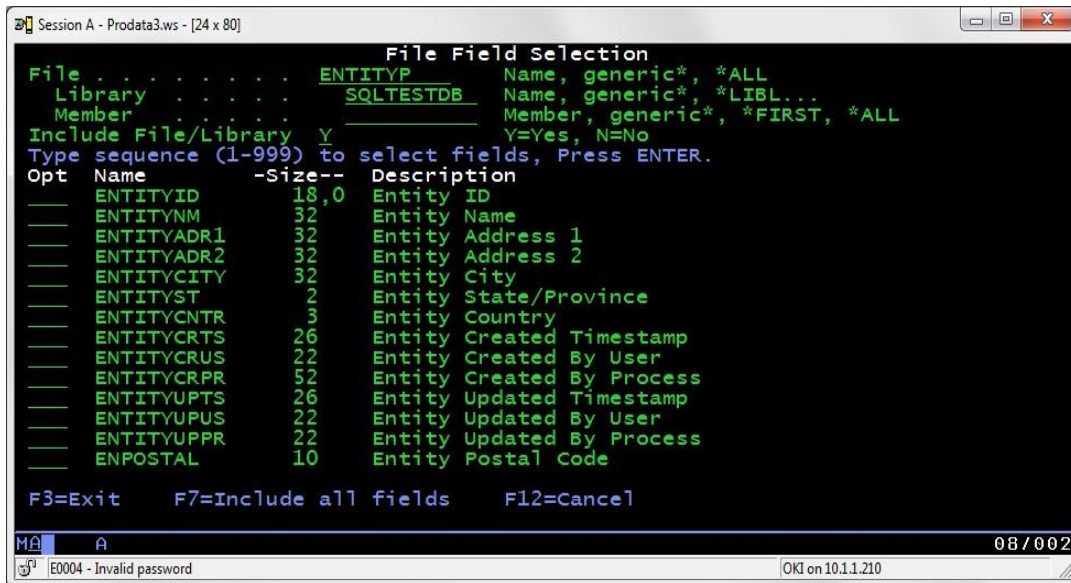
F2=Subset F3=Exit F4=Prompt F6=Insert line F9=Retrieve F12=Cancel
F10=Copy line F13=Services F14=Delete Line F24=More keys
SQL/PRO 5.0 COPYRIGHT(C) 2015 PRODATA COMPUTER SERVICES, INC.
15/006
E0004 - Invalid password
  
```

As you can see from the panel's lower lines, you can access all of the other functions of SQL/PRO by pressing the appropriate function keys.

The cursor is initially located at the first position of the SQL statement entry field. This is where you key your SQL statement. You are allotted up to 32,737 characters (442 lines for 80 column screen and 229 lines for 132 column screen) for your SQL statement. The Page Up and Page Down keys will roll you through all of the available lines.

Four function keys assist you in putting together your SQL statement. The F6 key will insert a blank line at the position of the cursor. F14 will delete the line on which the cursor is placed. F10 will copy a line to the line below it (moving down other lower lines if they contain any data). F15 will split a line at the cursor; the remainder of the line following the cursor will be moved down one line.

There is no need to memorize or write down the field names of files that you will be using in an SQL statement. The F17 key will allow you to view and then insert the field names from any file. It starts by prompting you for the name of the file and the library in which it's found. It then displays a list of the field names, sizes and field descriptions:

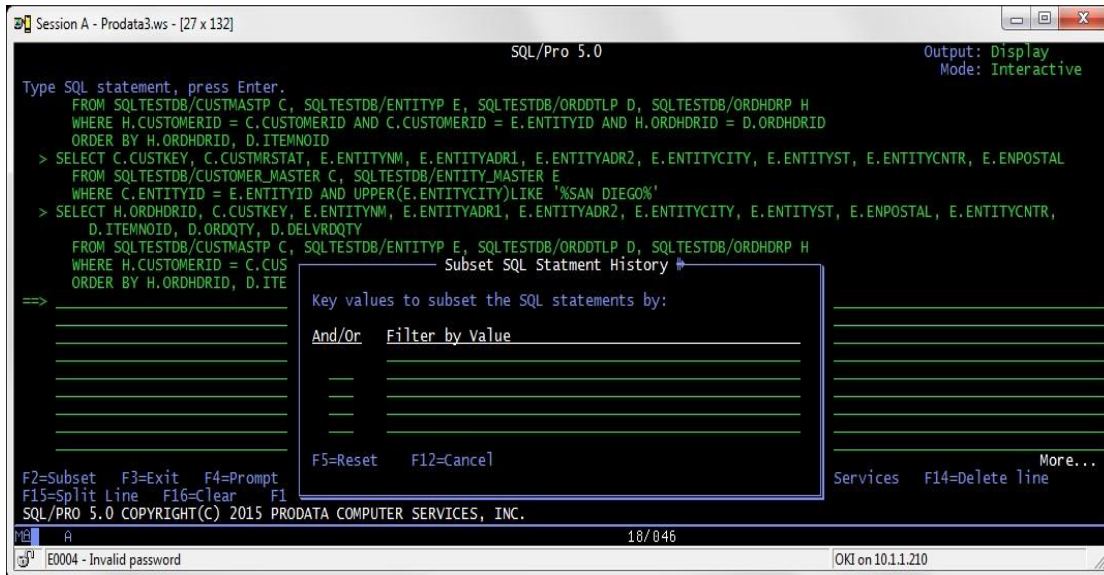


From here, you can select any fields to be included into your SQL statement by placing a '1' to the left of the field name. Selected field names will be placed at the cursor on the SQL statement entry screen. If more than one field is selected, the fields will be separated by commas. If the statement string goes beyond the 80-column screen width, the line will automatically wrap.

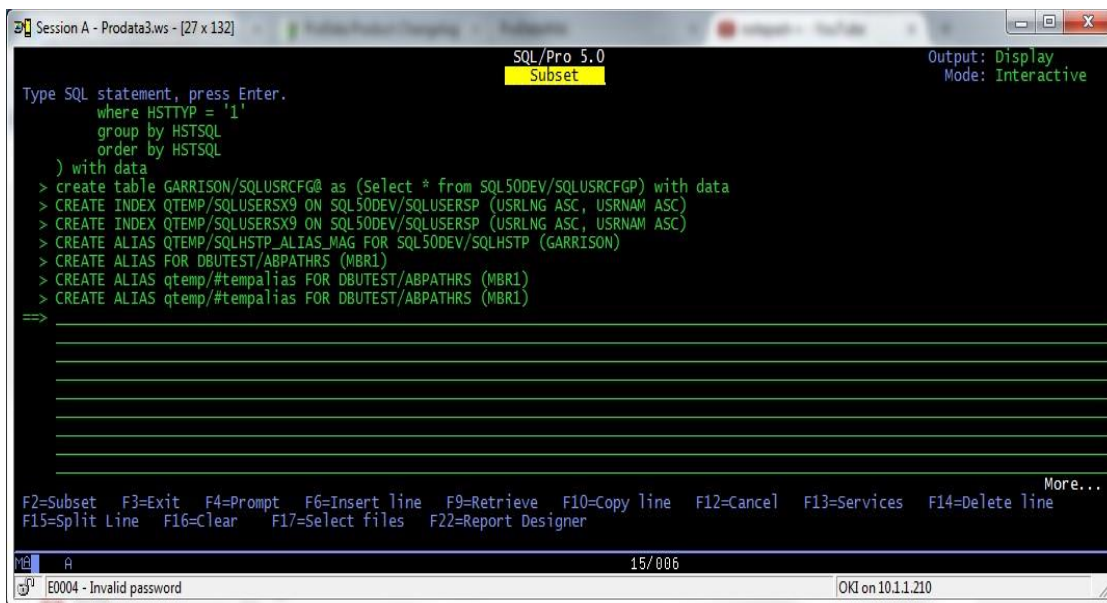
Note: If you use IDDU to create your physical files, the files must be externally described or you will not be able to use them properly with **SQL/PRO**.

SUBSETTING THE SQL S TATEMENT HISTORY

Pressing F2=Subset will allow you to subset (filter) the SQL statement history that is displayed. The following window will be displayed:



By keying a value in the Filter by Value column and pressing ENTER the following screen will be shown with the SQL statement history subsetted based on the keyed Filter by Value as shown below:

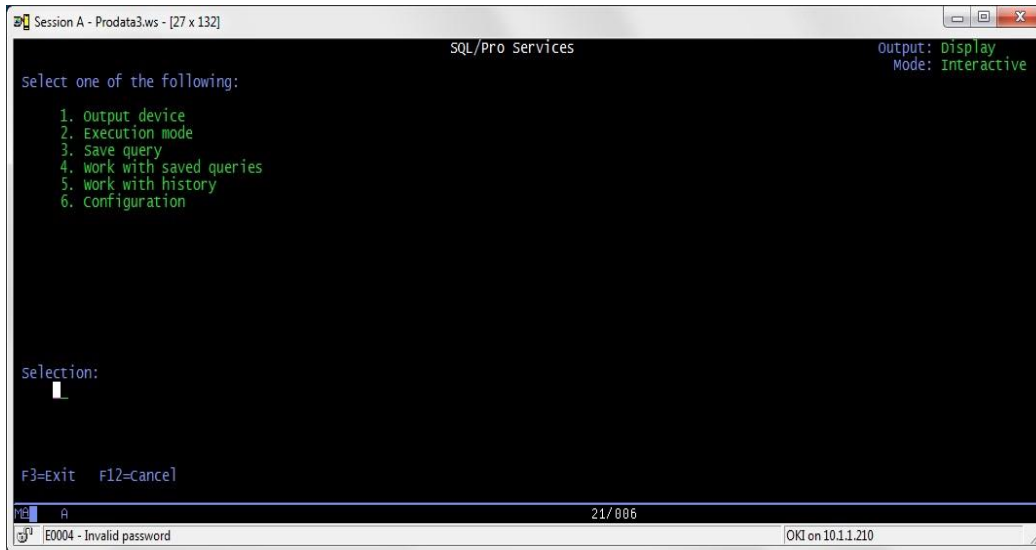


Multiple Subset values can be keyed using the And/Or operator to further customize the filtered SQL statement history displayed.

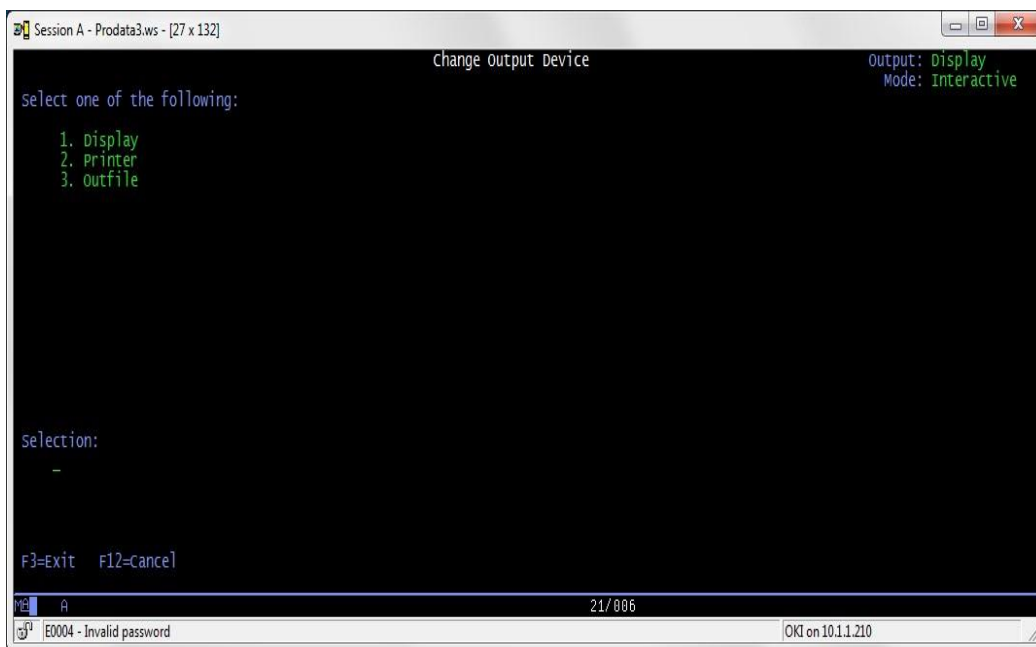
RUNNING THE SQL STATEMENT

SELECTING THE OUTPUT DEVICE

You have the ability to select an output device of either display, printer, or database file. This is performed by selecting option 1 from the Services menu (F13). You'll be presented a panel that provides for the selection:



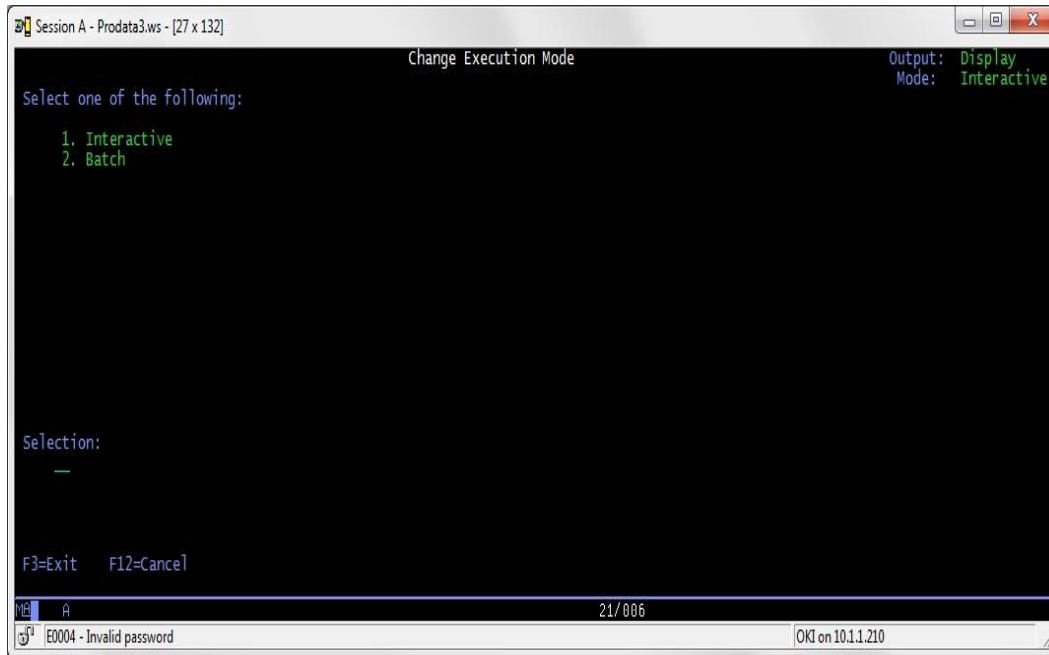
Enter the number of the selection you wish. The default output device is display. Each time you start **SQL/PRO** it will begin with this setting. If you select number 3, Database file, you'll be presented with the addition panel to define your file criteria:



When output is directed to the printer, **SQL/PRO** uses the print file QPQXPRTF. The default page width is 80 columns. This can be changed to 132 by enter the CHGPRTF command and specifying 132 on the Page Size keyword (width- positions per line).

SELECTING THE EXECUTION ENVIRONMENT

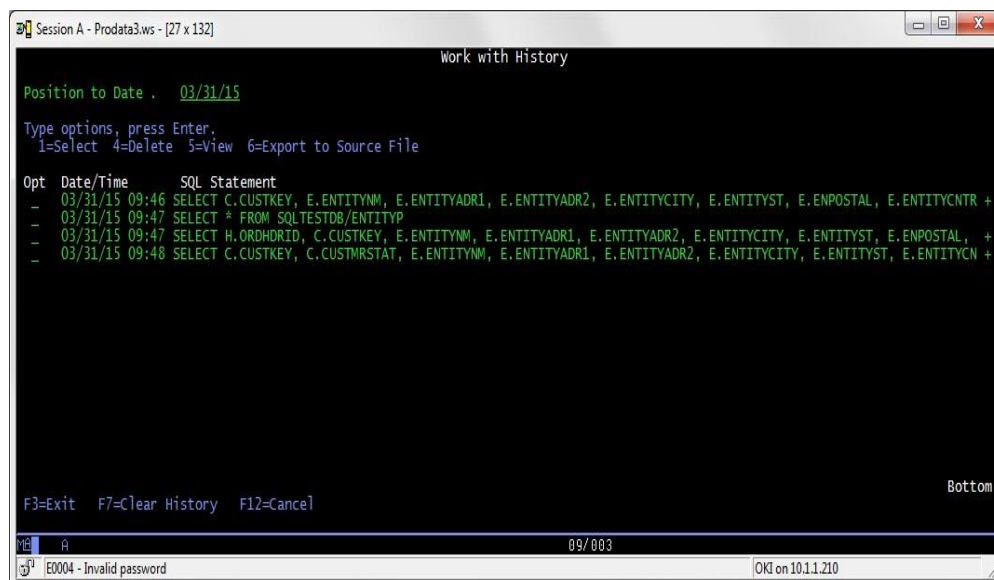
You are able to run your SQL statement either interactively or in batch, depending upon how you have set it. To set the execution environment, take option 2 of the Service menu (F13). Enter a "B" for batch or "I" for interactive. The default is Interactive: each time you start **SQL/PRO**, it will begin with this setting.



HISTORY

SQL/PRO has the ability to retain a history of executed SQL statements with the only size limitation being the capacity of your disk. Each time you hit enter to execute an SQL statement from the interactive **SQL/PRO** screen, the statement is logged into the **SQL/PRO** history file. This holds true even if the statement fails, as long as you hit enter, the statement is logged. If you key a statement and hit F22 to go into the report formatter but do not actually hit enter, the statement will not be logged.

You can review the history by selecting option 5 of the Services Menu (F13). This will give you a summary listing of **SQL/PRO** statements, with each line showing the SQL statement and the date and time it was run. This display is in chronological order with the initial display showing the most recent page of statements. You can page back and forth through the statements with the roll keys.



You can pull up the entire SQL statement by entering a '1' in the option field. A selected statement will be displayed in the original main screen. From here, you can do anything that you are able to do from the main SQL statement entry screen.

You can retain as much or as little history as you like. The history file will continue to grow until you run the purge history step from the services menu. ON this step you tell it the number of statements you wish to end up with after the purge.

SAVED QUERIES

A unique feature of SQL/PRO is Saved Queries. This function allows you to name and save any SQL statement that you have successfully run. You'll find this invaluable for those queries that you run time and time again. Rather than rewrite the SQL statement each time, just pull up the tried and true query. For example, if you frequently run a list of all customers with a balance due, just develop it once and create a Saved Query entry with a meaningful name like BALDUE. From that point on, you only need to select and run BALDUE from the Work With Saved Queries display. In fact, you can even reference the Saved Query in a CL program with the RUNSQLSQ command (more on that later).

You can create a Saved Query from any SQL statement you have on the SQL/PRO Statement Entry screen. Go to the Services menu (F13) and select option 3 (Create Saved Query). From here you will get the following panel:



Figure 2.7

Enter the name of the Saved Query (Choose an appropriate name since the list of Saved Queries displays in alphabetical order), a library and a meaningful description. The library defaults to the library specified in your configuration. Then enter the public authority which can be *ALL, *CHANGE, *USE or *EXCLUDE. This parameter default value is part of your configuration.

For each of the operations listed, the public requires the corresponding authorities:

- » Select or execute a saved query: *USE

- » Copy or replace a saved query: *CHANGE
- » Delete a saved query: *ALL

If the public is excluded from a saved query it will not be included in the list of saved queries presented by the Work With Saved Queries panel.

You can save an existing saved query by specifying 'Y' value for Replace. The default is no ('N') which will cause an error message if you attempt to save a query under a name that already exists.

The current output device information, including any specific database information, will be saved with the Saved Query. If you desire an output device other than the current one, you can change this. Later, when you run a Saved Query, you will be able to override the set output device if necessary. When you save a query a Query Management (*QMQR) object is created. If you have created a report format, a Query Management Form (*QMFORM) object is also created. This can take several minutes. The save process can be submitted to batch or run interactively by specifying Y or N in the submit to batch parameter. A default value of Y or N can be stored by using the configuration service option from the Services panel.

You may wish to create a Saved Query from a statement in your history. To do this, select the statement from the history display to bring it up in the **SQL/PRO** Statement Entry screen. Then proceed as discussed above as if you had just keyed it. Now to view the Saved Queries, select option 4 (Work With Saved Queries) from the Services menu (F13). Here is a sample of the Work With Saved Queries screen:

```

Work with Saved Queries
Position to . . . . . _____ Library . . . . . QGPL
Type options, press Enter.
  1=Select  3=Copy  4=Delete  6=Print  8=Run in batch  9=Run

Opt  Name                Description                Owner
--  -
  -  #OFCPUS              NUMBER OF DBU COPIES SOLD TO DATE      QPGMR
  -  DSMSALES             DSM PURCHASES TO DATE                QPGMR
  -  DSTCLIENTS          DISTRIBUTOR CLIENTS UNDER THEIR INITIALS  QPGMR
  -  DSTLIST              DBU Variable Customer Listing          QPGMR
  -  ILESALES            ILE PURCHASES TO DATE                QPGMR
  -  LEASES              YEAR200 & OTHERS LICENSE THAT ARE LEASED  QPGMR
  -  LEDGER              Ledger entries for May 1998            QPGMR
  -  LIC40               Lic w/4.0rel and maint not paid        QPGMR
  -  LIC40MAINT          LICENSE W/REL 4.0 MAINT PD NEED TO SEND OUT 4.1  QPGMR
  -  MOCLLPRMPT          MONTHLY BREAK DOWN OF TRIALS SENT AND WHO SENT  QPGMR
  -  MONTHCLL           MONTHLY BREAK DOWN OF TRIALS SENT AND WHO SENT  QPGMR
  -  NOTPURCH           TRIALS NOT PURCHASED IN A CERTAIN MONTH      QPGMR
  -  NOTPURYEAR        TRIALS NOT PURCHASED IN A YEARS TIME        QPGMR
  -
F3=Exit  F5=Refresh  F7=Change description  F12=Cancel
  
```

Figure 2.8

From here you can run a job, select it, delete it, copy it or print the statement by entering the appropriate number in the option field to the left of the query name. Queries can be saved to libraries of your choice therefore the “Work with Saved Queries” panel allows you to choose the library from which to display saved queries. The library defaults to the target library that has been configured by the user from the Configuration function panel.

With the WRKSQ command you can directly access the “Work with Saved Queries” display for saved queries within the library you specify. The format is:

WRKSQ LIB (library_name)

All options will be available except Select therefore, you cannot edit a saved query from the WRKSQ command.

RUNNING A SAVED QUERY

You can run a saved query either in batch or interactively. Option 9 runs it interactively and 8 runs it in batch. You can also run a Saved Query by selecting it with option 1 which will copy it into the SQL Statement Entry Screen.

Then follow the normal run procedures from there. With either of these three methods, you can run the job using the output device stored in the Saved Query, unless you choose to override the output device on the following screen (which is presented automatically prior to execution):

```

Override Saved Query Output Device

PDSQL Query . . . TEST
Description . . . Sample for Manual

Saved Query Output Parameters:
Override settings for this run, press Enter.

Device . . . . . OUTFILE   DISPLAY, PRINTER, OUTFILE
File . . . . . _____   Name
Library . . . . . _____   Name
Member . . . . . _____   Name, #FILE
Output option . . . _____   1=Create, 2=Replace
Authorization . . . _____
File text . . . _____

F3=Exit   F5=Refresh   F12=Cancel

```

Figure 2.9

Changing the output parameters will only affect running this current statement. Once completed, the session values that were in effect prior to selecting the Saved Query will be in effect once again.

You can also override any value of a Saved Query by selecting it with option 1 and then changing the statement itself, the output device and the execution environment before running the statement.

CHANGING A SAVED QUERY

Changing a Saved Query begins the same as overriding a Saved Query. You select with option 1 to bring the statement into the SQL Statement Entry screen. Make any changes to the statement you want. Then, go to the Services menu and run option 3 to save the query. You will be presented with the name of the query, the description, the public or private authorization and the output device that you specified when you last saved this query. At this point you can change any value before pressing enter. Obviously if you are changing a Saved Query, keep the Query name the same. However, you can change the name to save the query to a new Saved Query name.

COPYING A SAVED QUERY

The easiest way to copy a Saved Query is with option 3. This will provide you with the following screen from which you can copy to another Saved Query of another name.

```

Copy Saved Query

Copy from:  Name      Description
           TEST      Sample for Manual
Output Device . . . . :  OUTFILE          Public authority:  #ALL
Output File . . . . . :  QSQLSELECT
Library . . . . . :    QTEMP
Member . . . . . :    #FILE
Option . . . . . :    2
File authority . . . . :  #LIBCRTAUT
Text . . . . . :
Copy to . . TEST
Library . . QGPL      Sample for Manual
Output device . . . . :  OUTFILE          Public authority:  #ALL
Output file . . . . . :  QSQLSELECT
Library . . . . . :    QTEMP
Member . . . . . :    #FILE
Option . . . . . :    2
File authority . . . . :  #LIBCRTAUT
Text . . . . . :
F3=Exit    F5=Refresh  F12=Cancel

```

Figure 2.10

DELETING A SAVED QUERY

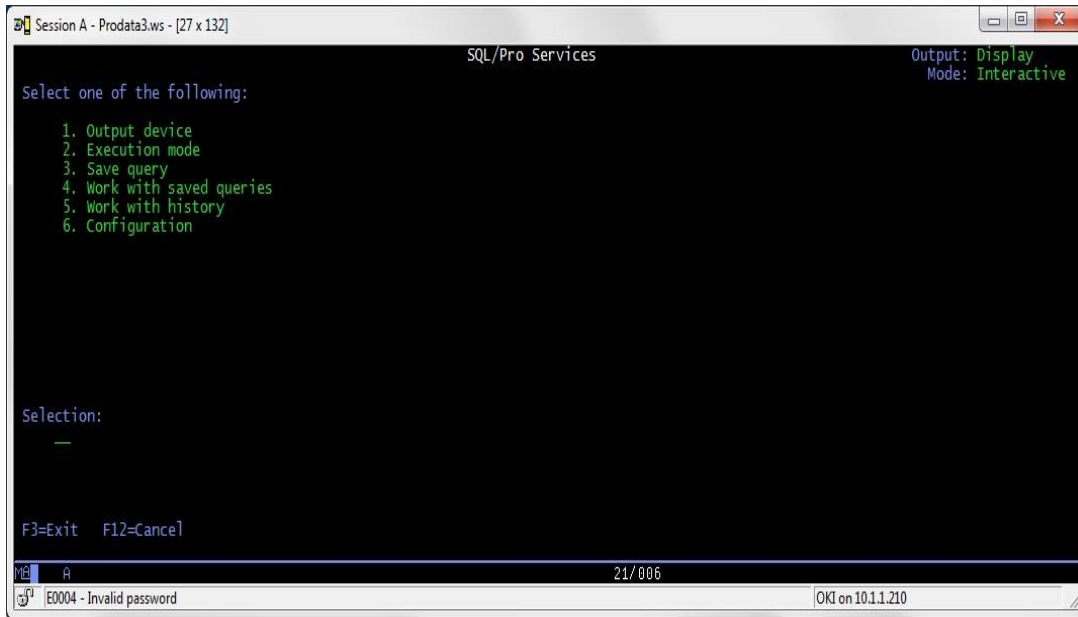
You can delete a Saved Query with option 4. You will be presented with a screen to confirm the deletion. You can also use the DLTSQ command described later in this section.

PRINTING A SAVED QUERY

You can print a Saved Query with option 6.

SERVICES

In previous sections of this chapter we've pointed you to the Services menu. This is it:



You've already seen how you can select the output device, set the execution options, work with history and create and work with Saved Queries. Option 6, Work with History, allows you to reorganize your history file. **SQL/PRO** stores history records for each user in a separate member of the history file. Eventually you will want to reorganize your history file to remove old, unwanted entries.

RUNNING A SAVE QUERY FROM THE COMMAND LINE

The command STRQMQRV will allow you to run a Saved Query from the command line. By keying STRQMQRV and pressing F4, you'll get this panel:

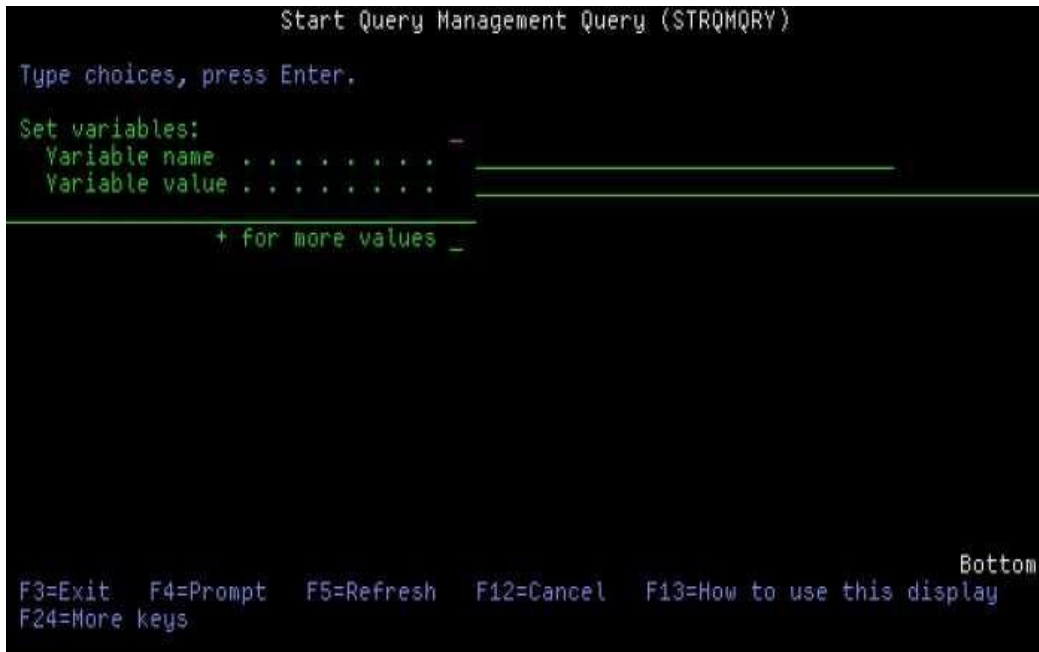


Figure 2.13

```

Start Query Management Query (STRQMQR)

Type choices, press Enter.

Query management query . . . . . Name
Library . . . . . *LIBL Name, *LIBL, *CURLIB
Output . . . . . * PRINT, *OUTFILE
Query management report form . . . . . Name, *SYSDFT, *QMQR
Library . . . . . Name, *LIBL, *CURLIB

Additional Parameters

Relational database . . . . . *NONE
Connection Method . . . . . *DUW *DUW, *RUW
User . . . . . *CURRENT Name, *CURRENT
Password . . . . . Character value, *NONE
Naming convention . . . . . *SYS *SYS, *SAA
Allow information from QRYDFN . . . . . *NO *NO, *YES, *ONLY

More...
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

Figure 2.12

In this example we are running the Saved Query LSTCSTMST, printing the output and substituting the value 'AZ' for the variable STATE. (Refer to chapter 5 for more information about variables.) Just key the name of the Saved Query and any other necessary information and press enter. This will begin running the Saved query as it is set up with **SQL/PRO**.

RUNNING A SAVED QUERY FROM A CL PROGRAM

You can also execute a Saved Query from a CL program with the STRQMQR command. The previous example where we used the command prompter could easily be embedded in a CL program like this: STRQMQR QMORY (LSTCSTMST) +

OUTPUT (*OUTFILE) +

SETVAR ((STATE 'AZ'))

QMQR: The name of the Saved Query you wish to run.

OUTPUT: The output device (* = Display, *PRINT = Printer, *OUTFILE = database file) SETVAR: Optional parameter used to substitute values into variables embedded in your queries. Maximum allowed is 50.

EMBEDDING SQL STATEMENTS IN CL PROGRAMS

Another unique feature of SQL/PRO over SQL/400 is its ability to execute an SQL statement from a CL program. This is somewhat similar to running a Saved Query from a CL program, only you can construct the SQL statement dynamically in the

CL Program. This is done with the RUNPDSQL command, which has the following format:

RUNPDSQL STM ('SQL-statement') +

OUTPUT (* or *OUTFILE or *PRINT) +
OUTFILE (qualified outfile name) +
OUTMBR (member_name or *FIRST *REPLACE OR *ADD)

STM: The SQL statement, in parentheses

OUTPUT: *For display (changes to *PRINT if run in batch), *OUTFILE to create an outfile and *PRINT to print.

OUTMBR: This parameter has two elements in the list. The first is either the specific outfile member name or *FIRST. The second element specifies *REPLACE (to replace the member if it already exists) or *ADD (to add the new member). Here is an example of running a dynamic SQL statement from within a CL program that creates an outfile PGM

```
RUNPDSQL STM ('SELECT + FROM CUSTOMER') +  
OUTPUT (*OUTFILE) +  
OUTFILE (QTEMP/CUSTLIST) +  
OUTMBR (*FIRST *REPLACE)  
  
CALL  
MYPROG  
END PGM
```

Of course you can construct the SQL statement dynamically in your CL program and then place it into STM through a variable: RUNPDSQL STM (&DYNSTM) ...

RETRIEVING A SAVED QUERY INTO A VARIABLE

You may have a reason to retrieve a Saved Query statement into a variable. For example, you may want to retrieve one, modify it in a CL program, and then run the modified statement with our RUNPDSQL command. You can do this with the RTVSQSQ command:


```

Retrieve Query Into Variable (RTVSQLSQ)

Type choices, press Enter.

Label . . . . . _____ Name
Enter saved query name . . . . . _____ Name, #LIBL
Library . . . . . #LIBL _____ Name, #LIBL
SQL statement (2720) _____ Character value
Output device (8) _____ Character value
Output file name (10) _____ Character value
Output file library (10) _____ Character value
Return code (1) _____ Character value
Comment . . . . . _____

Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

Figure 2.14

CONVERTING QUERY/400 QUERIES TO SQL/PRO

For those of you who have Query/400 queries **SQL/PRO** provides the Convert QRYDFN to **SQL/PRO** (CVTQRYSQL) command which will convert any QUERY/400 query to an **SQL/PRO** saved query. The converted query will execute through SQL/PRO just as it did through QUERY/400. However, if you need to edit the report format, the report formatter will force you to create your report format from scratch.

```

Convert QRYDFN to SQL/4LESS (CVTQRYSQL)

Type choices, press Enter.

Query/400 QRYDFN object . . . . . _____ Name
Library . . . . . #LIBL _____ Name, #LIBL, #CURLIB
SQL/4LESS saved query . . . . . #QRYDFN _____ Name, *QRYDFN
Library . . . . . _____ Name, *QRYDWNPRF, *USRPRF...
Text description . . . . . #QRYDFN _____

Public authority . . . . . #QRYDFN _____ *QRYDFN, *ALL, *CHANGE...
Replace SQL/4LESS saved query . . . . . #YES _____ *YES, *NO
Output device . . . . . #DISPLAY _____ *DISPLAY, *PRINTER, *OUTFILE

Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

Figure 2.15

A powerful feature of **SQL/PRO** is the ability to use variables within your SQL statements. What this means is that you can let the system either prompt you for a value or pass a value through a parameter and its value will be substituted into your SQL statement at execution time.

For example, you want your users to be able to create a list of customers for any state they choose. Instead of creating 50 different saved queries (one for each state) you can simply create one query that uses a variable in place of the state value. A variable is defined by starting a string with an '&' character. The following SQL statement would be all you would need to create the query for the above example: `SELECT * FROM CUSTMAST WHERE CMSTE = &STATE`

A variable can be placed anywhere within an SQL statement (in place of a column name, a file name, an operator or a compare value just to mention a few), but no more than 50 can be used. The user gains the added benefit of being able to run the query in batch.

At execution time the system will prompt the user for a value for variable STATE. Since CMSTE is a character data type it is required that the value be submitted within apostrophes.

If an SQL statement containing variables is saved, values can be passed to it through the STRQMQRV command.

This way the system doesn't need to prompt you for the value/s at execution time. To illustrate, here is the STRQMQRV command necessary to run the above statement using AZ as the state:

```
STRQMQRV QMQRV (LSTCSTMST)
          OUTPUT (*PRINT)
          SETVAR ((STATE "'AZ'"))
```

Since the command processor removes the outer apostrophes and converts embedded double apostrophes to single the SQL statement resolves to look like this: `SELECT * FROM CUSTMAST`

```
WHERE CMSTE = 'AZ'
```

 A CL variable could be used in place of the character constant. The format for the CHGVAR command could look like this:

```
DCL VAR(&QUOTE) TYPE(*CHAR) LEN(2) VALUE('')
DCL VAR(&STATE) TYPE(*CHAR) LEN(4)
```

EMBEDDED VARIABLES

This method also allows you to replace the character constant with a CL variable so that different values can be entered at run time from a display file or another program.

Passing numeric values is much easier since apostrophes aren't required. To pass a variable to the following SQL statement:

```
SELECT * FROM CUSTOMAST
      WHERE CMPDUE = &PDCTR
```

This segment of CL code could be used:

```
CHGVAR VAR (&PASTDUECTR) VALUE (6)
STRQMQR Y QMQR Y (LSTCSTMST)
      OUTPUT (*PRINT)
SETVAR ((PDCTR &PASTDUECTR))
      CHGVAR VAR (&STATE) VALUE (&QUOTE |< 'AZ' |< &QUOTE)
```

NOTE: “|<” is a concatenation symbol

The CHGVAR command (as well as any command that accepts character strings) strips the outermost single apostrophes and converts each inner set of two apostrophes to one. Therefore, the &STATE variable ends up with “AZ” as we desire.

The STRQMQR Y command could now be coded as:

```
STRQMQR Y QMQR Y (LSTCSTMST)
      OUTPUT (*PRINT)
SETVAR ((STATE &STATE))
```

FORMATTING YOUR REPORTS

Without the Report Designer, **SQL/PRO** only creates a simple listing using a default report format, which prints the values of the columns of the SELECT statement in the order they occur in the statement. The column names are used as heading constants above each column. This is the extent of the default report format.

Now with **SQL/PRO**' report Designer you can quickly and easily create sophisticated reports. The **SQL/PRO** Report Designer allows you to format the results of any SQL SELECT statement. You can also create summary information and perform several arithmetic operations. The Report Designer supports most of the requirements for a standard production report:

- » The order of the columns
- » The horizontal and vertical position of the information
- » Meaningful column and page headings
- » Control group summary operations
- » Separators to highlight headings and summary data
- » Text and summary data for the end of the report

With some advance techniques you can produce reports that are much more sophisticated than you might imagine.

For example:

- » Character edits that keep long strings on multiple lines of a report but in the same column
- » First and last summary functions
- » SQL column data in headers and footers

When a query is saved the report formatting for the query is also saved.

The following sample report gives you an idea of what can be done with **SQL/PRO** report formatting. We will lead you through each step of the format process that would create this report so you can get an overview of how it's done. The format function is identified by the labels to the right of the report. As we go through each of the functions in detail, refer to this report.

To begin designing a report, select F22 (Edit report format) from the main **SQL/PRO** screen and you will be transferred to the Report Designer main screen where you can specify your report functions and format. Initially **SQL/PRO** will create a default form for you. A report format record will be created for each column specified in your SELECT statement. If you use the '*' in your SELECT statement, a record will be created for every column in the table. On subsequent requests for report format editing (F22), within the same session, **SQL/PRO** will present a window informing you that a form already exists. Online help is available from anywhere within the Report Designer. Press the help key or F1 to access the online help. The online help is cursor sensitive, so position your cursor on the phrase in question before accessing help.

You can choose to change the existing form or re-create a new default form. Typically you will change an existing form unless you change the file from which you are retrieving your information in which case you would want to recreate a default form.

The Edit report format function presents the Report Designer panel.

In this example we are creating a control group for District and summarizing the total order amount for each group.

To add a column use F6. A blank column will be inserted after the column in which the cursor is located. To delete a column use F14. The column in which the cursor is located will be deleted. There must be at least one column in a

report format (the system will not let you delete a column if it is the last one in the report format).

F7 displays the Design Report Header window and allows you to enter up to five lines of header text. You can see the prompt at the top of the window requesting for the number of lines to skip before and after the header text prints. If these are left blank the header text will simply print on line one. For every text line entered, a “Justify” option of LEFT, CENTER or, RIGHT must be selected to specify the text alignment on the page.

F8 presents the Design Report Footer screen. This screen is exactly the same as the report header except that the text lines entered will print at the bottom of the report. Here, we are placing the name of the saved query.

To create control group header text use F10. You will be prompted for break header text for each column with a control break specified in the USASGE column (a maximum of six groups can be created, BREAK1 to BREAK6). Again you are prompted for the number of blank lines to skip before and after the text prints, but you also have two new options: Print on new page and repeat column text. These both default to NO but entering YES in the first option will cause a page break and a YES in the second will reprint the column heading text immediately following the break header. Typically, if YES is specified in Print on new page you would leave the repeat column text at NO since the column text is printed automatically on every new page anyway.

Take particular notice of the ‘&1’ in the break text. The ampersand followed by a number tells the report format to print the value of the nth data column from the SQL SELECT statement. In this case the first data column contains the break field DISTRICT. Any number of data columns may be inserted in this way and is not limited to the break fields.

F11, like F10, will prompt you for footer text for each break field. Again, the example has an embedded variable, ‘&1’, to pull in the DISTRICT. The justification LEFT tells the report format to print the break footer text to the immediate left of the first summary column. RIGHT, then, will print to the right of the last summary column.

The report final text (F9), again, works essentially the same as the other report sections except that the final text allows for ten lines of text instead of the header and footer limitation of five. Function key 22 brings up the Modify Standard Report Defaults screen. You will rarely use this screen if you are happy with the defaults specified. Review the previous report example to see what the standard defaults look like. Now that you know how to create report headers and footers, let’s go over the formatting of the detail data. The main screen has a record for each column in the SQL SELECT statement. Most of the editing can be done from the main report designer screen except that the column header text entry field allows for only 27 characters to be entered.

Whenever a column heading exceeds 27 characters or is to list on multiple lines of text, use the Extended Column Edit screen accessible with a 2 option.

The extended column edit allows you to enter up to four lines of column heading per data column. There is, however, a limitation of a total of 62 characters per column heading.

The Usage edit is the real power option to the report designer. Here is where you specify the break fields (which should correspond to the ORDER BY fields on the SQL SELECT statement) and the summary functions. The usage options are: OMIT, AVG, COUNT, FIRST, LAST, MAX, MIN, SUM, BREAK1 - BREAK6 Most of these are obvious and there is help text available if you position the cursor in the usage field and press F1 or HELP. I will however mention two of the more obscure options: OMIT and COUNT. Omit simply specifies that a column is not to appear on the report. COUNT lists the total number of detail lines that were a part of the summary data group break. Notice also that only one usage can be specified per column.

The edit code is where you definitely will use the F1 Help function. These SQL edit codes are not the same ones that AS/400 programmers are familiar with from SDA, RPG, and Query/400, but the help text is very straightforward. The edit codes include numeric, date, time and several interesting character edits.

The field length and decimal position represents the width to display on the report - it does not have to correspond to the actual physical data length in the file. If a numeric value exceeds the width specified, the report will display all

Asterisks in that data column.

The column indent is the number of spaces from one data column to the previous data column.

The column sequence allows you to override the placement of a data column on a report. You may have keyed a long SQL SELECT statement and no longer want the report column sequence to default to the column placement from the SQL. With the sequence number you can easily move columns on a report without changing the SQL statement. The data type is set on the automatic build of the form but on column adds (F6) you need to specify the appropriate type. Don't worry about whether not the field is packed or zoned - the report designer sees all numbers as NUMERIC. Version 2 enthusiasts will also be happy to know that DATE and TIME data types are available with **SQL/PRO**' Report Designer. The extended column edit screen will automatically be displayed when you select the add function (F6). Remember: The number and data type of report format columns must always agree with that of the SQL SELECT statement. This is where the Display SQL Function (F3) comes in handy. Again, if you add or delete a column from the SQL statement you must also add (F6) or delete (F14) from the report form.

Advanced Report Formatting Techniques

Multi-level reports Many databases have files with one-to-many relationships. An example of a one-to-many relationship is a sales order header and sales order detail. Formatting a report that joins two such files is a problem since the header information is duplicated for every detail record. Forget trying to create a clean report of this nature with Query/400, but with **SQL/PRO**'s Report Designer and a few simple techniques it can be fairly easy. Using a Sales Order Picklist as an example the following steps describe how to create a multi-level report. On the SQL SELECT:

- » Duplicate the break fields.
- » Concatenate a blank to the beginning of each break field. If the break is numeric use the SQL digits() function to convert it to character.

```
SELECT ORDNR, ' ' || ORDNR, ORDDTE, LINITM, LINQTY, LINPRC, LINQTY * LINPRC
FROM ORDHDR, ORDDTL
WHERE ORDHDR.ORDNR = ORDDTL.ORDNR
ORDER BY ORDNR
```

On the Report Design:

- » Specify the concatenated ORDER BY duplicate as the break field and set the column width of the break field to 1 so that only the blank portion of the break will list on the report (you can't omit a break column)
- » Set USAGE to OMIT for all other header fields
- » Design a Break header with the header columns specified to print in the header.

Date Edits Dates are a bit of a problem with Report Designer. We do not have the old six digit numeric date edit of Y available to produce a formatted date. But there are two other methods of editing a date: **Method One** The first way to edit a date is to substring it into its individual month, day, and year fields in the SQL SELECT

statement and concatenate. For instance, to edit a year-month-day date to MM/DD/YY use:

```
substr (datefld, 3,2) || '/' || substr (datefld, 5, 2) || '/' || substr (datefld, 1,2)
```

Method Two

Another way is to use the date edits provided with the Report designer. These edits, though only work with the new OS/400 Version 2 date types. But you don't have to wait for date and time fields to be in your database. What you do is convert your old format date into a Version 2 date field by combining Method One to create a date that matches your system date format and then enclose the manually edited date with the SQL DATE () function. Then select any of myriad of date edits available in the Report Designer (see the F1 field text on the main screen).

Why go to all this trouble since it uses Method One anyway? By converting your database dates into Version 2 date data type you can then use date math capabilities. This makes calculating such things as the number of days between dates and the maturation of notes a breeze.

Address Formatting

A neat trick that the Report Designer will do for you is to format an address. How often have you done lists where a full address was required? But two or three address lines; a name; and the city, state, and zip code takes up too many columns on the report. This leaves too few columns available on the report to print anything else. How would you like to print something like this:

This technique requires the following steps:

On the SQL SELECT

- » Find the longest address line. For example: address line one may be 40 characters but the City, State, and Zip might exceed 50 with the embedded blanks.
- » In the SQL concatenate all fields together adding blank strings so each address line is the length of the longest line.

For instance, if the name, address line 1, and address line 2 are 40 bytes and the city/state/zip combination is 50 including the concatenated comma and blanks: `Select name ||' '|| ad1||' '|| ad2||' '|| strip (city) ||','||state||' '||zip, purch, amtpaid, amtdue`

(Notice the column function we use with the city: `strip(city)`. This strips the trailing blanks from the city.)

On the Report Format

- » Change the column heading for the concatenated column to something meaningful since by default it will contain concatenation characters. Here, we change it to Customer address.
- » Set the field width to longest field address line (50 in the example).
- » Specify the CT character edit code to concatenated address.
- » The CT character edit will then wrap the columns to the next line of the report when the column width has been exceeded (50 in the example).

Standard Headers The Report Designer defaults to automatically printing the current date, time, and page number on the bottom of the report. This works quite well but many users are so used to having the date, time and page at the top of the report

that they now expect it. To do this with the Report Designer is easy; simply enter `&DATE`, `&TIME` and `&PAGE` (all caps) in

the report header. The problem with the report formatter though, is that the justify option prints to the left, right, or the center of the report. Users want the date, time, and page at opposite ends of the report. So - fake it. Put the `&DATE` and `&TIME` on the first line of the report header left justified. Then put the `&PAGE` on the second line of the report right justified. Finally, put your report title on line three and center it.

Remember, if you use `&DATE`, `&TIME` and `&PAGE` on a report for a saved query and later run the query with the

STRQMQR command, you could end up with date, time and/or page on the report twice (wherever you place it and at the bottom of each page). This is because the STRQMQR command will place the date, time, and page number at the bottom of every page of any query report if the DATETIME and PAGE parameter values are set to *YES (the default).

Dynamic Reports One of the most powerful features of **SQL/PRO** is its support of variables allowing us to create dynamic reports. All that is required is to create an SQL SELECT statement that has variable selection criterion in the WHERE clause.

```
SELECT CUSNBR, NAME, ADDRESS  
  
FROM CUSMAS  
WHERE &WHERE
```

Any word prefixed by an ampersand is seen as a variable to **SQL/PRO**. The Variable is then set at run time in the STRQMQR command by keying the variable name in the VAR parameter (make sure to use the same case as is specified in the SQL but without the ampersand) and the dynamic selection in the VALUE parameter.

Note that triple quotes are required for character strings; numeric data should have NO quotes or else the system will see the value as a string.

```
STRQMQR QMQR (YOURQR) QMFORM (YOURQR) VAR (WHERE) +
```

```
VALUE ('"AMTDUE > 100.00 "')
```

This powerful feature allows you to create reports where the user can dynamically set the selection criterion. But the report header should contain the selection criterion or else the user will not know what it is. The solution to this problem is to list the selection criterion on the report by putting the variable on the report header or footer. To do this you need to add the variable to the column list of the SQL SELECT since the Report Designer only supports variable data from the SELECT list.

```
SELECT CUSNBR, NAME, ADDRESS, PHONE
```

```
FROM CUSMAS
```

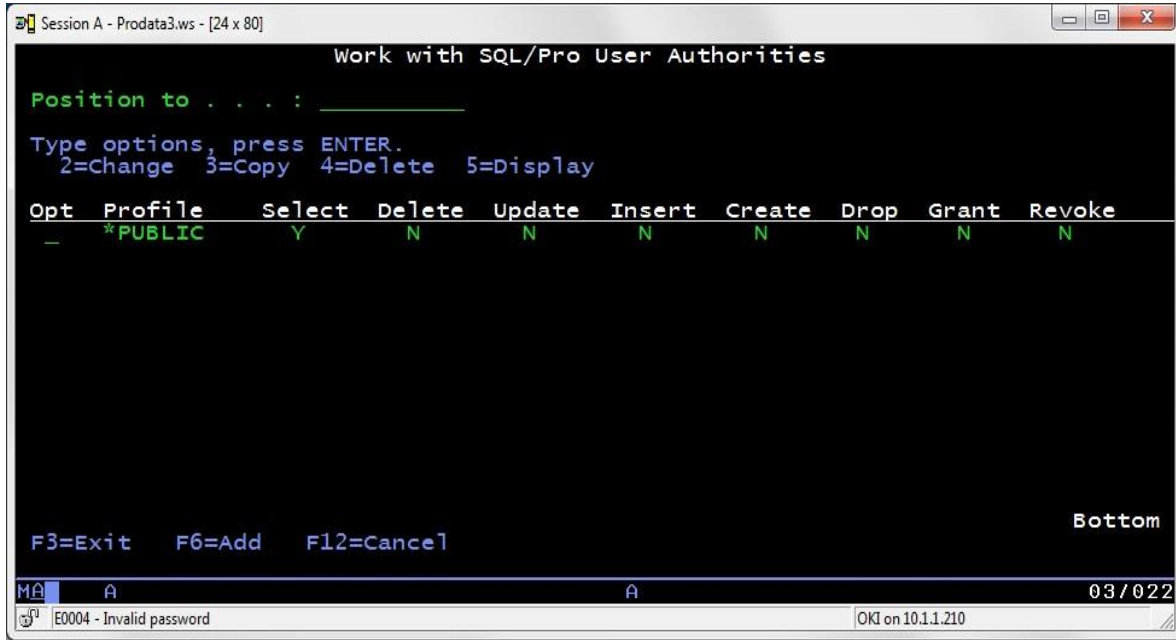
```
WHERE & PHONE
```

Then put that data column in the report header or footer and OMIT in the USAGE column in the detail data. The dynamic selection will then always print on the report. You can make slick use of this technique by coding a command or HLL screen with a wide range of prompts to assist the user through the entry of the selection logic.

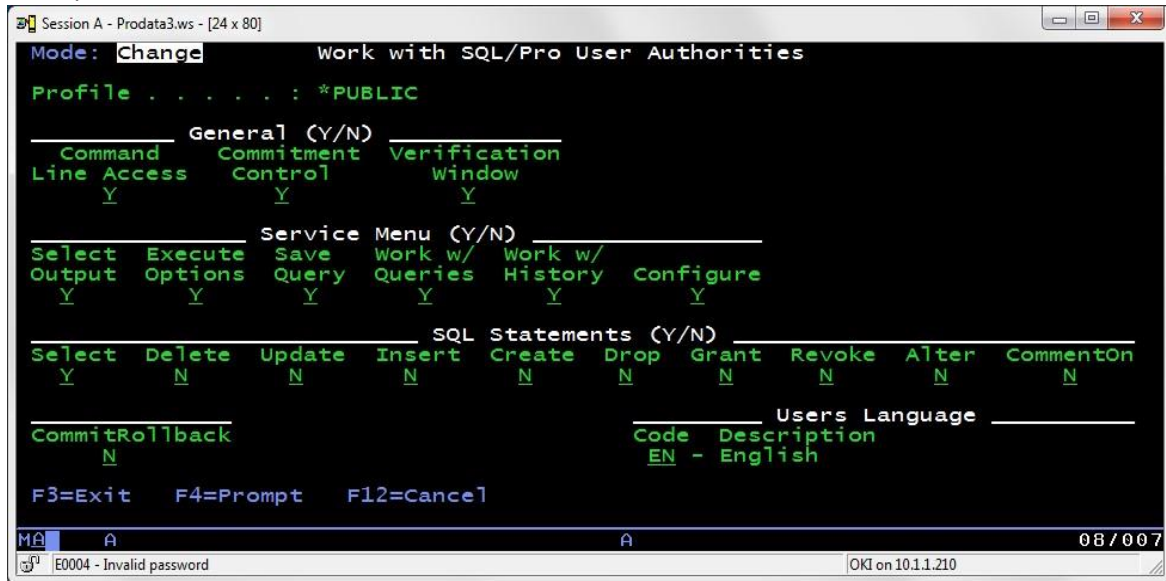
SQL/PRO AUTHORITY

ACCESSING SQL/PRO AUTHORITY

Make sure that you are signed on as security officer (QSECOFR) and have added library SQLPRO50 to you library list. Enter the command SQLAUT and press ENTER. The following screen will be presented:

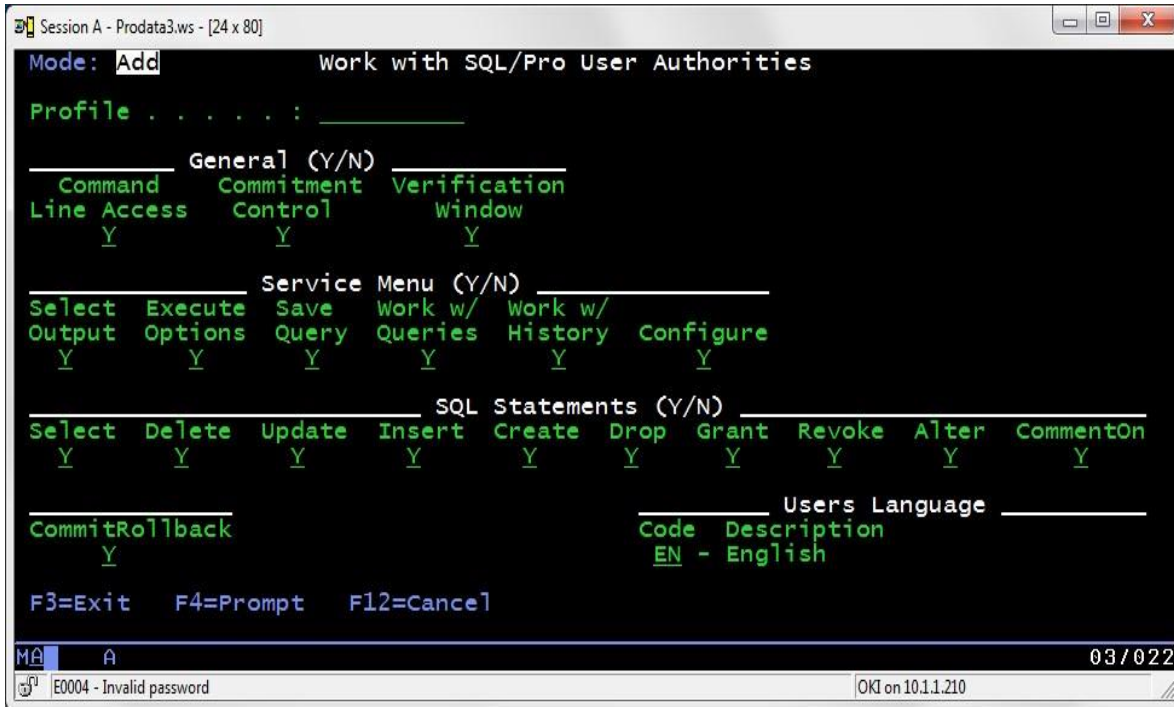


Keying an option of 2=Change and pressing ENTER will present the following screen to allow the **SQL/PRO** authority for a user to be updated.



Pressing F6=Add will present the following screen to allow a **SQL/PRO** authority for a new user to be added.

When you elect to do an Add or Change the next screen appears. Place a “Y” or “N” in each field to denote the functions that the profile will have access to.



*PUBLIC is the default user profile. It is initially set up to have no authorities. This profile is used as the default if a user’s individual or group profile is not found.

CONFIGURING THE SQL AUDIT FUNCTION

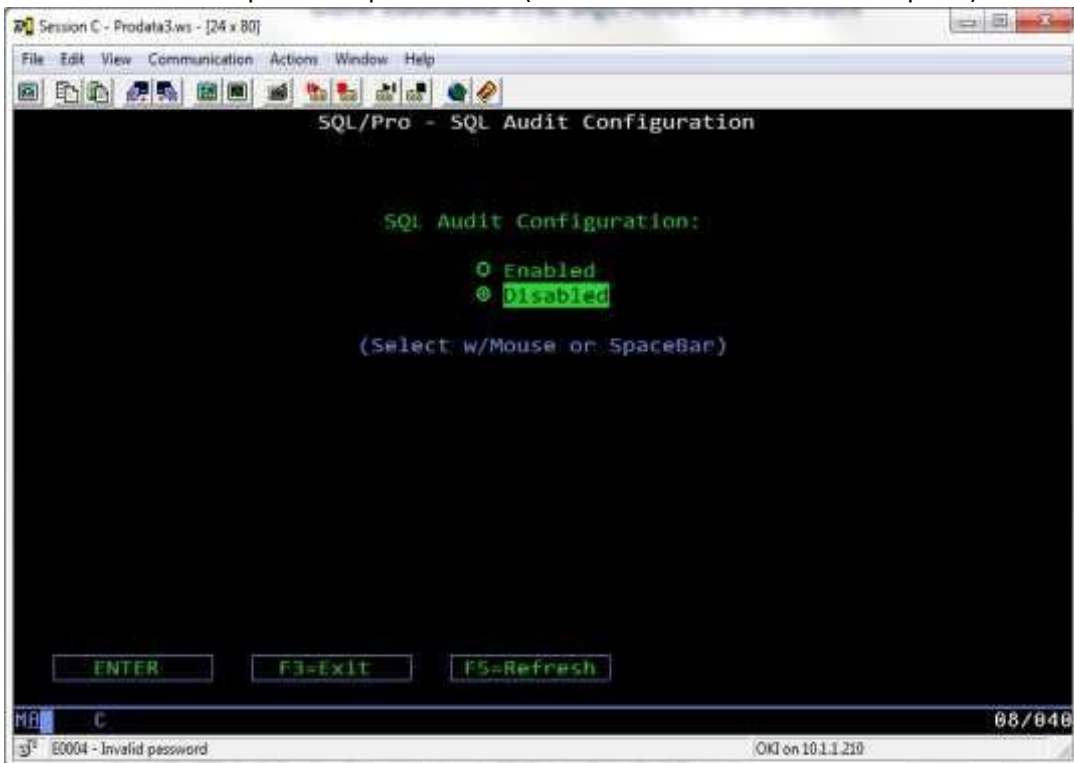
To set the SQL Audit function for your system to Enable then perform the following:

1. Sign on the system as QSECOFR or a user with *SECADM Special authority.
2. Execute the command SQLAUDCFG.
 - a. Select the “Enabled” option and press ENTER (or double click the “Enabled” option).



To set the SQL Audit function for your system to Disabled then perform the following:

1. Sign on the system as QSECOFR or a user with *SECADM Special authority.
2. Execute the command SQLAUDCFG.
 - a. Select the "Disabled" option and press ENTER (or double click the "Disabled" option).



UPDATING SQL/PRO

If your system has a connection to the Internet you may easily update **SQL/PRO** with the latest updates, fixes, and enhancements. Your IBM I (iSeries, AS/400) needs access to the Internet and port 2809 needs to be available to perform the update. To update **SQL/PRO** via the Internet connection perform the following steps.

1. Execute command WRKOBJLCK OBJ(SQLPRO50) OBJTYPE(*LIB) to verify that there are NO locks on the SQL/PRO library. If you continually experience locks then you might want to consider running this process at a time when there are no users on the system.
2. Execute command SQLPTF LIB(SQLPRO50) to download and install the latest updates, fixes, and enhancements from the ProData Computer Services, Inc. **SQL/PRO** servers.

UNINSTALLING SQL/PRO

If you ever decide to de-install **SQL/PRO** , you should follow these steps:

1. Remove SQLPRO50 from your library list.
2. Ensure that there are no object locks on library SQLPRO50,
3. Delete the library SQLPRO50 from your system.

Note: If you are only temporarily removing **SQL/PRO**, make sure to make a backup copy first. Then reinstall **SQL/PRO** from the backup copy. If you do not, you will lose your Saved Queries and history.