***This module is designed for three contact hours of class time***

**Module Objectives:**

* Understand VSAM concepts with emphasis on Key Sequences Data Sets (KSDS)
* Understand elementary relational database concepts
* Be able to create VSAM clusters using IDCAMS
* Be able to create, populate and query DB2 tables
* Be able to write programs that run against VSAM files and DB2

Background: Chapter 4 of zconcepts.pdf in the Reference Module

**VSAM Organization**

VSAM (Virtual Storage Access Method) allows you to create an indexed cluster consisting of a series of indexes to permit sequential and/or nonsequential access to individual records. The Access Methods Services (AMS) utility IDCAMS is used to create a VSAM clusters with either unique or non-unique indexes for both primary indexes and alternate indexes. The following example will illustrate a creation and testing of VSAM cluster with a unique primary index and a non-unique alternate index.

Assume the following data set:

1 2 3 4 5 6 7

1234567890123456789012345678901234567890123456789012345678901234567890

050555500GRAUER RT11450877200E59001181800118050000118145000180

264805298CLARK JS12450879100E53330880222087928000108126000080

300000000MILGRAM IR06130580200E655510814000681480000580

400000000BENJAMIN BL10531073100E73331073 30000108128000180

638972393GRAUER J 11450877200E59001181800118050000118145000180

800000000SMITH BB09430678100E64440678 42000068036000078

900000000BAKER E 06130580200E655510814000681480000580

955000000GRAUER B 11450877200E59001181800118050000118145000180

The numbers on the top of the file with a gray background are the column numbers. The first nine digits of each record (in yellow) are the social security number of the employee. It is a unique ID and is the primary key. For example, the second record has the unique id of 264805298 and Clark begins in column 10. The first part of the task requires creating a VSAM cluster with a unique index—you have to identify the column locations of the records for the unique key to accomplish this task. This is done via identifying the offset from the beginning of the record. In this example, the social security number has an offset of 0 because it begins in the first column. See if you understand offset—what would be the offset for the last name column? What do you notice about the order of the records? Do you think this is by accident? You are correct, it is no accident – unless otherwise sorted in the program, the input records must be sorted in order of the primary index.

The next 15 characters are the last name of the employee. We want to be able to search on the primary index and also on the last name of the employee. Therefore, the last name of the employee will be used as the alternate index. In this case, the alternate index would not be unique (notice the multiple entries for GRAUER).

Although one long step could be used to accomplish this task, this illustration uses two steps—the first step creates the primary index and the second step creates the alternate index. But first, the IDCAMS utility is discussed.

**IDCAMS Utility**

IDCAMS is a utility program for managing VSAM and non-VSAM files. It provides many access method services; some of them are:

* Create a VSAM cluster with a primary index.
* Create a VSAM alternate index.
* List catalog information for a dataset.
* Print the content of a dataset in hex or character form.
* Do backups of VSAM and non-VSAM datasets.
* Delete a VSAM or non-VSAM dataset.

A listing of some of the IDCAMS Control Statements

* BLDINDEX – Build alternate indexes for existing datasets
* DEFINE ALTERNATE INDEX – Define an alternate index
* DEFINE CLUSTER – Define a cluster for KSDS, ESDS, or RRDS
* DEFINE PATH – Define a path directly to a base cluster or an alternate index and its related base cluster
* DELETE – Delete catalogs, VSAM datasets, and non-VSAM datasets
* PRINT – Print contents of VSAM datasets, non-VSAM datasets, and catalogs
* REPRO – Copy VSAM and non-VSAM datasets
* IF – Conditional execution of commands. Used with LASTCC or MAXCC where CC is Condition Code.

**Defining a VSAM Cluster with a Primary Index**

To create a VSAM cluster with a primary indedx, the IDCAMS utility is used. The REPRO statement is used for all copy operations. The code below creates the cluster UOASXXX.VSAM.MASTER from UOASXXX.WORK.DATA(OLDMAST). Notice the KEYS command. The (9 0) specifies that the key is 9 characters long and begins at offset 0 – the first column in the data set above.

You will create acctid.WORK.DATA if not already created and download the input file OLDMAST that is on Blackboard into your PDS.

Download the file named repro on Blackboard into a PDS of type such as COMPILES that accepts 80 character records. Use the appropriate command to replace all instances of UOASXXX with your acctid. Remember to use capital letters!

Type SUBMIT and press enter to run the program.

//REPRO JOB (AMSEXI),'UOASXXX',NOTIFY=UOASXXX,

// CLASS=A,MSGLEVEL=(1,1)

/\*

//STEP0010 EXEC PGM=IDCAMS

//INDATA DD DSN=UOASXXX.WORK.DATA(OLDMAST),DISP=SHR

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

//SYSIN DD \*

DELETE (UOASXXX.VSAM.MASTER) CLUSTER

DEFINE CLUSTER ( -

NAME(UOASXXX.VSAM.MASTER) -

INDEXED -

RECORDSIZE(80 80) -

**KEYS(9 0) -**

FREESPACE(19 5) -

VOLUMES (DB1469)-

RECORDS (50 5)-

SHAREOPTIONS (2 3) -

UNIQUE) -

DATA ( -

NAME(UOASXXX.VSAM.MASTER.DATA) -

CISZ(1024)) -

INDEX ( -

NAME(UOASXXX.VSAM.MASTER.INDEX))

REPRO INFILE(INDATA) -

OUTDATASET(UOASXXX.VSAM.MASTER)

IF LASTCC = 0 -

THEN

PRINT INDATASET(UOASXXX.VSAM.MASTER) CHARACTER

Did you receive a condition code of 8?

**Do you remember how to examine the results of a run?**

* TSO SDSF ST is the command to issue.
* Owner acctid
* Type a ? in front of the last job.
* Enter an S in front of the row with a DDNAME of JESYSMSG and press enter.
* What are the condition codes? (Use FIND COND)

The first time this program is submitted, a condition code of 8 is achieved on the DELETE step because the delete statement is trying to delete a cluster that does not yet exist:

DELETE (acctid.VSAM.MASTER) CLUSTER

IDC3012I ENTRY UOASXXX.VSAM.MASTER NOT FOUND

IDC3009I \*\* VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLA3-42

IDC0551I \*\* ENTRY UOASXXX.VSAM.MASTER NOT DELETED

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

Press F3 to return to the SDSF JOB DATA SET DISPLAY screen.

Type an S in front of the row with a DDNAME of SYSPRINT and press enter.

How many records were processed?

What do you notice about the order of the keys?

Even though you had a condition code of 8 (because the cluster to delete did not exist), the VSAM Cluster is created and a listing should appear near the end of the output display similar to this:

LISTING OF DATA SET -UOASXXX.VSAM.MASTER

KEY OF RECORD - 050555500

050555500GRAUER RT11450877200E59001181800118050000118145000180

KEY OF RECORD - 264805298

264805298CLARK JS12450879100E53330880222087928000108126000080

KEY OF RECORD - 300000000

300000000MILGRAM IR06130580200E655510814000681480000580

KEY OF RECORD - 400000000

400000000BENJAMIN BL10531073100E73331073 30000108128000180

KEY OF RECORD - 638972393

638972393GRAUER J 11450877200E59001181800118050000118145000180

KEY OF RECORD - 800000000

800000000SMITH BB09430678100E64440678 42000068036000078

KEY OF RECORD - 900000000

900000000BAKER E 06130580200E655510814000681480000580

KEY OF RECORD - 955000000

955000000GRAUER B 11450877200E59001181800118050000118145000180

IDC0005I NUMBER OF RECORDS PROCESSED WAS 8

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

**Note**:After the first run, a condition code of zero is achieved on all steps including DELETE:

DELETE (UOASXXX.VSAM.MASTER) CLUSTER

IDC0550I ENTRY (D) UOASXXX.VSAM.MASTER.DATA DELETED

IDC0550I ENTRY (I) UOASXXX.VSAM.MASTER.INDEX DELETED

IDC0550I ENTRY (C) UOASXXX.VSAM.MASTER DELETED

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

**Defining an Alternate Key**

After the VSAM cluster has been created (or recreated) using the IDCAMS utility in the JCL code above, the alternative key can be defined. To define the alternate key of last name, the following code is used. Notice that with the RELATE statement, the alternate index is related to the VSAM cluster created above (UOASXXX.VSAM.MASTER). In a simple Cobol program to test the VSAM Cluster alternate index, the name of the alternate index is UOASXXX.VSAM.MASTER.PATH. Notice the KEYS statement with (15 9). The alternate index has a length of 15 and begins at offset 9 in the VSAM cluster. Remember that the first column is really column 0. Columns 0 through 9 contain the social security number. The last name begins in the next column.

To run this program, copy the file named reproai from Blackboard, change all instances of UOASXXX to your acctid, and type SUBMIT.

//REPROAI JOB (AMSEXAI),'UOASXXX',NOTIFY=UOASXXX,

// CLASS=A,MSGLEVEL=(1,1)

/\*

//STEP0010 EXEC PGM=IDCAMS

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

//ASMDUMP DD SYSOUT=\*

//SYSIN DD \*

DEFINE ALTERNATEINDEX ( -

NAME(UOASXXX.VSAM.MASTER.ALTNDX) -

**RELATE(UOASXXX.VSAM.MASTER) -**

VOLUMES (DB1469)-

RECORDS (50 5)-

**KEYS(15 9) -**

RECORDSIZE(80 80) -

NONUNIQUEKEY -

UPGRADE)

DEFINE PATH -

(NAME(**UOASXXX.VSAM.MASTER.PATH**) -

PATHENTRY(UOASXXX.VSAM.MASTER.ALTNDX))

BLDINDEX INDATASET(UOASXXX.VSAM.MASTER) -

OUTDATASET(UOASXXX.VSAM.MASTER.ALTNDX)

IF LASTCC = 0 -

THEN -

PRINT INDATASET(UOASXXX.VSAM.MASTER.PATH)-

CHARACTER

/\*

//

Follow the same steps as before to examine the output.

The output display should be similar to the following:

LISTING OF DATA SET -UOASXXX.VSAM.MASTER.PATH

KEY OF RECORD - BAKER

900000000BAKER E 06130580200E655510814000681480000580

KEY OF RECORD - BENJAMIN

400000000BENJAMIN BL10531073100E73331073 30000108128000180

KEY OF RECORD - CLARK

264805298CLARK JS12450879100E53330880222087928000108126000080

KEY OF RECORD - GRAUER

050555500GRAUER RT11450877200E59001181800118050000118145000180

KEY OF RECORD - GRAUER

638972393GRAUER J 11450877200E59001181800118050000118145000180

KEY OF RECORD - GRAUER

955000000GRAUER B 11450877200E59001181800118050000118145000180

KEY OF RECORD - MILGRAM

300000000MILGRAM IR06130580200E655510814000681480000580

KEY OF RECORD - SMITH

800000000SMITH BB09430678100E64440678 42000068036000078

IDC0005I NUMBER OF RECORDS PROCESSED WAS 8

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

**COBOL Program that uses VSAM cluster and Alternate Index.**

The program first searches on primary index (social security numbers of 955000000 and then 245118095) and then by alternate key (last name of GRAUER and then HUMMER). What do you expect the output to be?

Notice the two JCL statements at the bottom of the program that define the VSAM cluster and alternate inces (repeated here for reference). Can you find acctid.VSAM.MASTER and acctid.VSAM.MASTER.PATH in the two programs above?

**STATEMENT TO REFERENCE THE VSAM CLUSTER:**

//GO.VSAMMAST DD DSN=UOASXXX.VSAM.MASTER,DISP=SHR

**STATEMENT JCL TO REFERENCE THE VSAM ALTERNATE KEY:**

//GO.VSAMMAS1 DD DSN=UOASXXX.VSAM.MASTER.PATH,DISP=SHR

Copy the file vsamex01 on Blackboard to a PDS with a type such as COMPILES that accepts 80 column characters. Change all instances of UOASXXX to your ACCTID. Type Submit and press enter.

**COBOL PROGRAM:**

//AIXEXAMP JOB (AIVSAM),'UOASXXX',NOTIFY=&SYSUID,

// CLASS=A,MSGLEVEL=(1,1),TIME=1,MSGCLASS=A

//STEP1 EXEC PROC=IGYWCLG

//COBOL.SYSIN DD \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* This program was modified from the code in text-- \*

\* The IBM Cobol Environment by Robert T. Grauer, page 120 \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IDENTIFICATION DIVISION.

PROGRAM-ID. ALTINDEX.

AUTHOR. UOAS360.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT INDEXED-FILE

ASSIGN TO DA-VSAMMAST

ORGANIZATION IS INDEXED

ACCESS IS DYNAMIC

RECORD KEY IS INDEX-SOC-SEC-NUMBER

ALTERNATE RECORD KEY IS INDEX-NAME

WITH DUPLICATES.

DATA DIVISION.

FILE SECTION.

FD INDEXED-FILE

LABEL RECORDS ARE STANDARD

RECORD CONTAINS 80 CHARACTERS

DATA RECORD IS INDEXED-RECORD.

01 INDEXED-RECORD.

05 INDEX-SOC-SEC-NUMBER PIC X(9).

05 INDEX-NAME PIC X(15).

05 REST-OF-INDEXED-RECORD PIC X(56).

WORKING-STORAGE SECTION.

01 FILLER PIC X(14)

VALUE 'WS BEGINS HERE'.

01 WS-NDX-MAST-RECORD.

05 NDX-SOC-SEC-NUMBER PIC X(9).

05 NDX-NAME.

10 NDX-LAST-NAME PIC X(15).

10 NDX-INITIALS PIC XX.

05 FILLER PIC X(54).

01 WS-ACTIVE-NAME PIC X(15).

01 WS-BALANCE-LINE-SWITCHES.

05 WS-RECORD-KEY-ALLOCATED-SWITCH PIC X(3).

05 WS-END-INDEX-FILE PIC X(3).

PROCEDURE DIVISION.

0010-PROCESS-NAME-FILE.

OPEN INPUT INDEXED-FILE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* RETRIEVE RECORDS BY SOCIAL SECURITY NUMBER \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MOVE '955000000' TO INDEX-SOC-SEC-NUMBER

PERFORM 0040-READ-INDEX-FILE-BY-NUMBER

IF WS-RECORD-KEY-ALLOCATED-SWITCH = 'YES'

DISPLAY WS-NDX-MAST-RECORD

ELSE

DISPLAY ' '

DISPLAY 'NO MATCH FOUND FOR: 955000000'

END-IF

MOVE '245118095' TO INDEX-SOC-SEC-NUMBER

PERFORM 0040-READ-INDEX-FILE-BY-NUMBER

IF WS-RECORD-KEY-ALLOCATED-SWITCH = 'YES'

DISPLAY WS-NDX-MAST-RECORD

ELSE

DISPLAY ' '

DISPLAY 'NO MATCH FOUND FOR: 245118095'

END-IF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* RETRIEVE RECORDS BY NAME - THE ALTERNATE RECORD KEY \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MOVE 'GRAUER ' TO INDEX-NAME

WS-ACTIVE-NAME

PERFORM 0020-READ-INDEX-FILE-BY-NAME

IF WS-RECORD-KEY-ALLOCATED-SWITCH = 'NO'

DISPLAY ' '

DISPLAY 'NO MATCH FOUND FOR: ' WS-ACTIVE-NAME

ELSE

PERFORM 0030-READ-DUPLICATES

UNTIL WS-ACTIVE-NAME NOT = NDX-LAST-NAME

OR WS-END-INDEX-FILE = 'YES'

END-IF

MOVE 'HUMMER ' TO INDEX-NAME

WS-ACTIVE-NAME

PERFORM 0020-READ-INDEX-FILE-BY-NAME.

IF WS-RECORD-KEY-ALLOCATED-SWITCH = 'YES'

DISPLAY WS-NDX-MAST-RECORD

ELSE

DISPLAY ' '

DISPLAY 'NO MATCH FOUND FOR: ' WS-ACTIVE-NAME

END-IF

CLOSE INDEXED-FILE

STOP RUN.

0020-READ-INDEX-FILE-BY-NAME.

MOVE 'YES' TO WS-RECORD-KEY-ALLOCATED-SWITCH

READ INDEXED-FILE INTO WS-NDX-MAST-RECORD

KEY IS INDEX-NAME

INVALID KEY

MOVE 'NO' TO WS-RECORD-KEY-ALLOCATED-SWITCH.

0030-READ-DUPLICATES.

DISPLAY WS-NDX-MAST-RECORD

READ INDEXED-FILE NEXT RECORD

INTO WS-NDX-MAST-RECORD

AT END

MOVE 'YES' TO WS-END-INDEX-FILE.

0040-READ-INDEX-FILE-BY-NUMBER.

MOVE 'YES' TO WS-RECORD-KEY-ALLOCATED-SWITCH

READ INDEXED-FILE INTO WS-NDX-MAST-RECORD

INVALID KEY

MOVE 'NO' TO WS-RECORD-KEY-ALLOCATED-SWITCH.

//\*

//GO.VSAMMAST DD DSN=UOASXXX.VSAM.MASTER,DISP=SHR

//GO.VSAMMAS1 DD DSN=UOASXXX.VSAM.MASTER.PATH,DISP=SHR

/\*

**OUTPUT from the Program**

Follow the same steps as before to examine the output.

Are the condition codes all 0 when you show the row with the DDNAME of JESYSMSG?

To see your output, enter an s in front of the row with the ProcStep equal to GO.

The output from the program you should see is:

\*\*\*\* END OF MESSAGE SUMMARY REPORT \*\*\*\*

955000000GRAUER B 11450877200E59001181800118050000118145000180

NO MATCH FOUND FOR: 245118095

050555500GRAUER RT11450877200E59001181800118050000118145000180

638972393GRAUER J 11450877200E59001181800118050000118145000180

955000000GRAUER B 11450877200E59001181800118050000118145000180

**NO MATCH FOUND FOR: HUMMER**

Did your expected output match the actual output?

**Review Questions:**

* Does the order of the records matter? Explain.
* Must the alternate index have unique values?
* Can you explain the KEYS statement?
* What if the original data file were changed to the following – what would be used for the KEYS statement in each of the two programs to generate the VSAM cluster and the alternate index? For your reference, the first row is included so that you can count the column.

1 2 3 4 5 6 7

1234567890123456789012345678901234567890123456789012345678901234567890

050500GRAUER RT11450877200E59001181800118050000118145000180

264898CLARK JS12450879100E53330880222087928000108126000080

300000MILGRAM IR06130580200E655510814000681480000580

400000BENJAMIN BL10531073100E73331073 30000108128000180

638993GRAUER J 11450877200E59001181800118050000118145000180

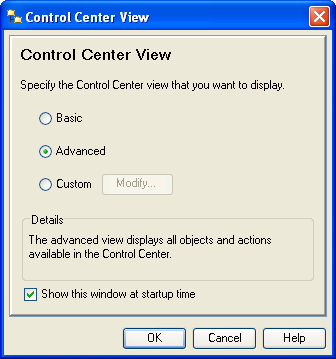
800000SMITH BB09430678100E64440678 42000068036000078

900000BAKER E 06130580200E655510814000681480000580

955000GRAUER B 11450877200E59001181800118050000118145000180

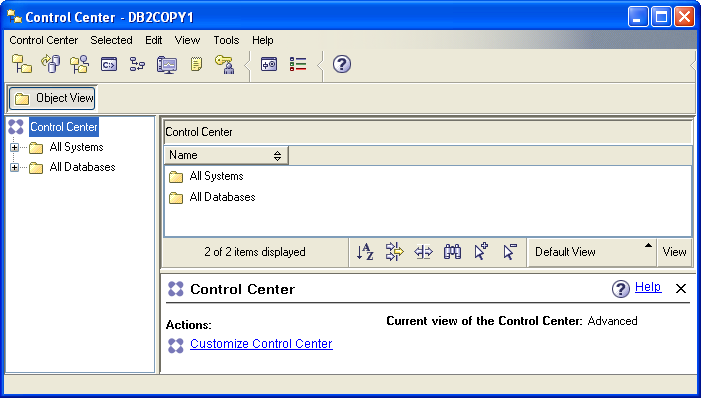
**Creating a table in your ACCTID using the Control Center**

After logging onto the IBM terminal server, click the Control Center icon to open the DB2 control center dialog.

Alternate: Start 🡪All Programs🡪IBM DB2🡪General Administration Tools🡪Control Center

This is IBM’s interface to DB2 and is common for DB2 on all platforms. A dialog window may appear allowing the user to select a desired Control Center view. The DB2 dialog window shown here defaults to Advanced--note that the user can uncheck the Show this window at startup time checkbox to always accept the default.

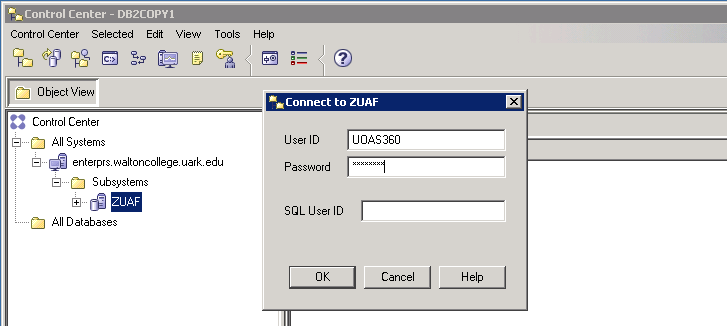
Click the OK button to accept the default Advanced setting to reach the Control Center. Although one can go directly to the Command Editor (a command line interface), the Control Center is probably the better choice. It allows the user to poke around to see what instances of DB2 are available without having to logon to a system. However, one does have to log on and connect to access a particular DB2 instance. Further, after login, reviewing data is possible without queries.



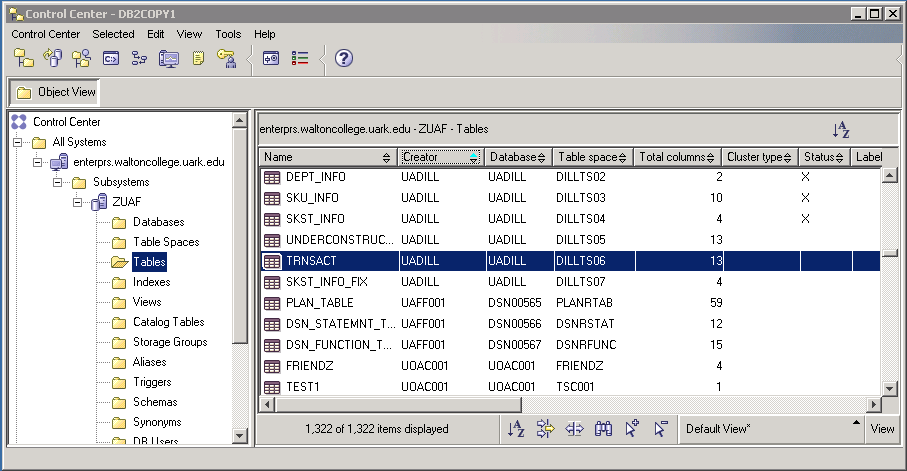
The icons under the menu allow the user to select different types of tasks (Views) to be accomplished. Moving the mouse over the icons displays a tool tip indicating its View. For our purposes here, we will initially work directly from the Control Center View.

The Object View and the expandable tree underneath it provide information about the systems and databases that are accessible by the user. Note that it will change over time. For example, the UA adds student and faculty accounts as needed.

Expanding the tree demonstrates that the UA has three DB2 instances. The ZUAF instance is running on the IBM z900. Trying to expand an instance results in a Connect dialog window that prompts for a user id and password; for the ZUAF instance in this case.



After logging in, expand the ZUAF DB2 instance and click on Tables. Note that the columns in the right pane are sorting by simply clicking the header. Multiple clicks reverse the sort on each click—the first click being in ascending order. In the image shown below, the Database column was clicked twice to get a descending order of databases—an existing database that may be of interest is UADILL. Note that it has six tables.

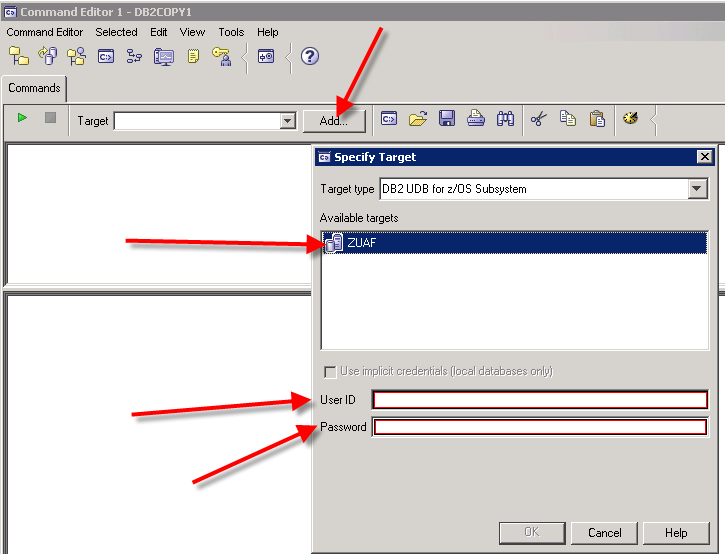


**Creating a table**

Before working with the UADILL database, lets create a table for an ACCTID. There are a number of ways to do this but one easy way if one knows the SQL to create a table is to use the command editor. Click the icon  shown below to open the command editor.



Click the “Add” button to open a dialog that will provide a connection string to the desired database, ZUAF in this case. Enter your sign on credentials.



If the login is successful, you should see your account in the Target drop down textbox and the bottom pane of the Command Editor should indicate a successful connection.



The example table for this illustration is an “Order” table. The SQL statements to create this table are shown below—note that a unique index and primary key are included. Also note that this is a very complete definition and many of the features could be allowed to default.

ALTER TABLE UOAS360.ORDER DROP CONSTRAINT ORDER\_ID;

DROP INDEX UOAS360.ORDER\_ID\_NDX;

DROP TABLE UOAS360.ORDER;

CREATE TABLE UOAS360.Order (

Order\_ID INTEGER NOT NULL,

Item\_ID INTEGER NOT NULL,

Item\_Desc VARCHAR(25) NOT NULL,

Item\_Cost DEC(7,2) NOT NULL,

Item\_Price DEC(7,2) NOT NULL,

Item\_NbrOrdered INTEGER NOT NULL,

Item\_Supplier\_ID INTEGER NOT NULL

);

CREATE UNIQUE INDEX UOAS360.ORDER\_ID\_NDX ON UOAS360.ORDER (ORDER\_ID ASC)

NOT CLUSTER

BUFFERPOOL BP3

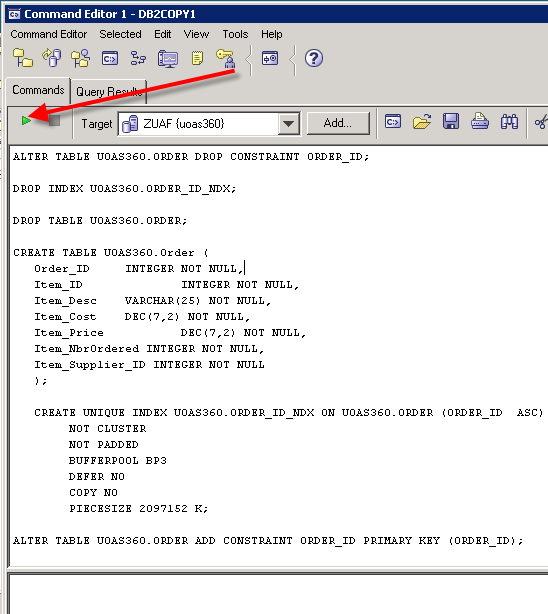
DEFER NO

COPY NO

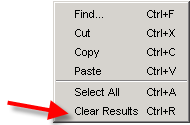
PIECESIZE 2097152 K;

ALTER TABLE UOAS360.ORDER ADD CONSTRAINT ORDER\_ID PRIMARY KEY (ORDER\_ID);

Copy all of the statements above and paste into the top pane of the Command Editor and then click the green run icon .



The results of the run will appear in the lower pane—note that if this is the first time you run the SQL statements, then errors will occur as you are asking it to delete the primary key, an index and a table that are all non-existent. Also, the creation of the unique index will throw a warning; but is ok. You need to look for “The SQL command complete successfully” for each SQL statement except for the unique index which throws a warning.



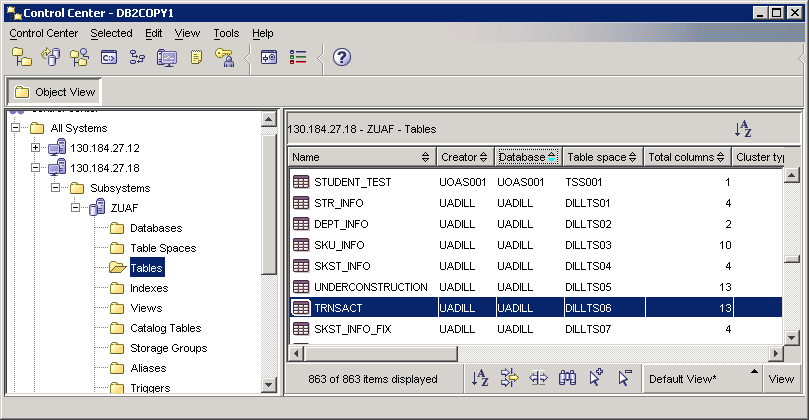
***Note1: To clear contents in either the upper pane or the lower pane of the Command Editor, right-click in an open space and select Clear Results***

***Note2:*** **Because you have added a table, you will need to right-click on ZUAF and select Refresh for it to appear in the Tables list**

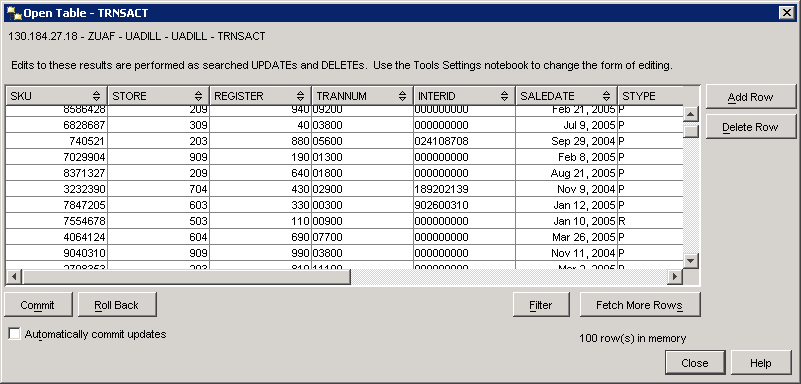
To populate the “Order” table, you can stay in the Command Editor and use INSERT statements or return to the Control and populate the table. Note—when returning to the Command Editor, you may have to select a Target in the drop-down box. Return to the Control Center by clicking the leftmost icon on the toolbar. In the Control Center, sort the Creator column by clicking on it—as indicated it changes from ascending to descending on each click. Find your account id and double-click it. You can then enter data.

**Example Queries using the Dillard’s Dataset**

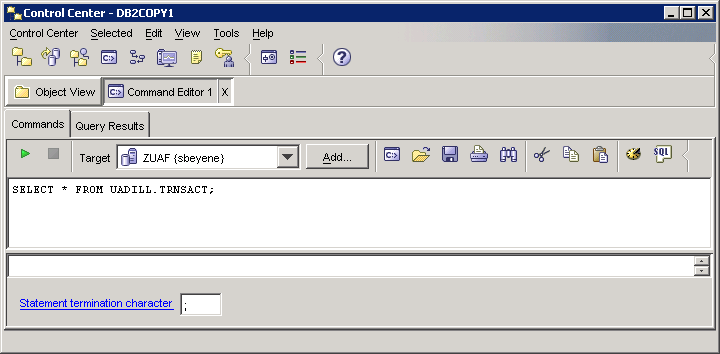
When in the Object View, expand the ZUAF DB2 instance and click on Tables. Note that the columns in the right pane are sorting by simply clicking the header. Multiple clicks reverse the sort on each click—the first click being in ascending order. In the image shown below, the Database column was clicked twice to get a descending order of databases—the database of interest for this demonstration is UADILL. Note that it has six tables with the names that match the ERD presented above.



Double-click the TRNSACT table to view the data.



To create queries, right-click on any table and select Query--the Control Center’s Query View opens which is an instance of the Command Editor View. For this example, right-click on the TRNSACT table and select Query.



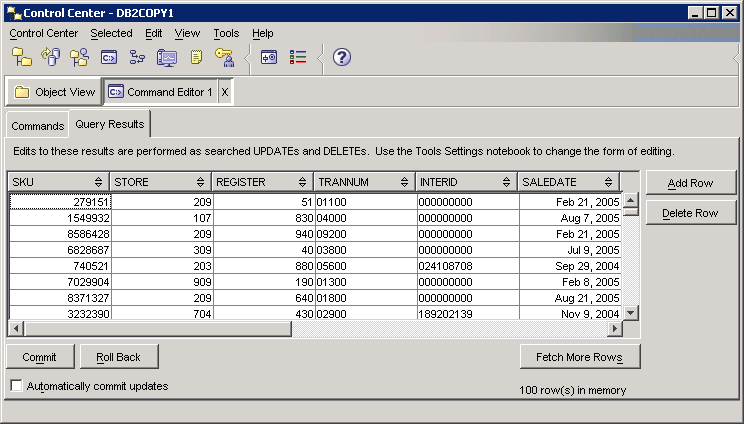
Icon to invoke SQL Assist

Enter SQL Statements Directly

Run button

Command Editor Instance

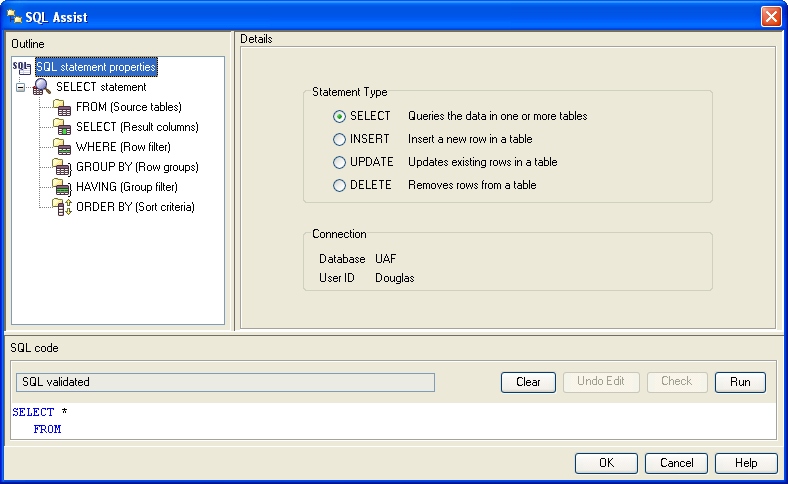
The default query is to select all columns and rows from the table on which the user clicked—in this case, the TRNSACT (transaction table). Note the run button—click the run button to run the query. **Note—the Commands tab is active. When the query is executed, the results will show up on the Query Results tab.** Click the Run button.

****

Click the Commands tab to return from the Query Results to the Commands tab. Then click the SQL icon to open the SQL Assist window. SQL Assist, a query by example—QBE, is shown below.

SQL Statement Type

Guide for creating the SQL statement



Clear and Run buttons

Beginning SQL Template and SQL entry panel

SQL Statement Status

SQL Assistant is designed to guide the user through creating queries. Of course, the user can simply enter the SQL statement directly. Also, note that initially, the Check button is not available—greyed out—but may become available as the SQL statement is created. The user can alter generated SQL at any point and then click the Check button to validate the SQL syntax.

Building the SQL statement using the SQL Assist interface is relatively straight forward. The upper left pane steps the user through creating the SQL statement by first identifying the tables, then the SELECT statement, etc. The Details part of the interface changes based on the context. For example, after selecting the tables, the user can join them and then select the columns.

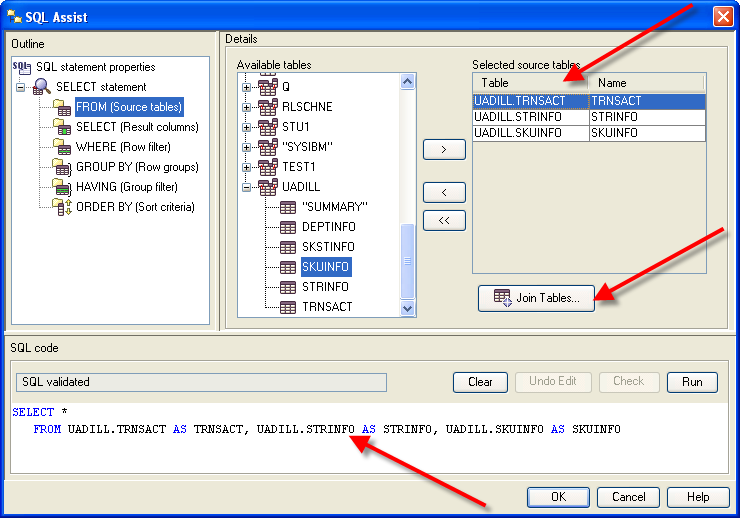
Note that three tables have been selected to illustrate using SQL Assist—TRNSACT, STRINFO and SKUINFO. The following is a possible management scenario using these three tables.

**Management Scenario**

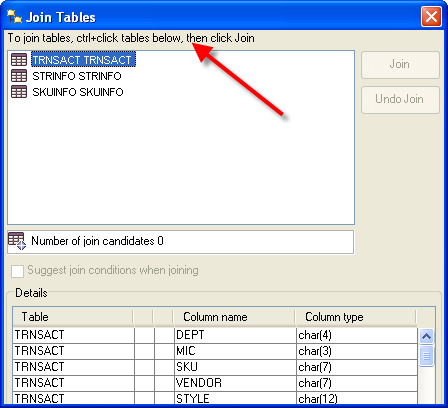
Management wishes to know the best performing stores, by state, city and department number, in terms of total sales for the first six months of the year 2005. The output should be in descending order of total sales.

Note that this SQL statement can be entered directly into the SQL code window if the user is proficient enough in SQL to enter the data. The SQL Assist can also be used to create the query and its use for generating the above query will be illustrated below. Again, note that the user can work with SQL Assist as much or little as desired because the user can alter the SQL in the SQL Assist query pane at any time.

Click the SQL icon in the Command Editor instance to open SQL Assist as shown above. Again, note the default SQL type command is Query. From the upper left outline pane, click FROM as shown below. Also, scroll down if needed to find the UADILL database in the **Available table** pane in the Details part of the window and expand UADILL so the tables in this database will be displayed. Recall in this example, three tables – TRNSACT, STRINFO and SKIINFO will be used. Therefore, select each of these tables and click the > button to add each of the three tables to the **Selected source tables** pane as shown below.

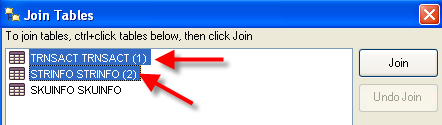


As shown, the three tables have been selected and have been added to the SQL statement in the Query pane. Click the Join Tables... button to join the tables.



Notice the instructions on how to join tables. As indicated, using conventional Windows techniques, select the tables you wish to join. In this example, the TRNSACT table needs to be joined to the STRINFO table and then the TRNSACT table needs to be joined to the SKUINFO table. Thus, it will take two steps to complete the joins.

Note that because two tables have not yet been selected, the Join and Undo Join buttons are not available.

First, select the TRNSACT and STRINFO tables and note that the Join button will now available as shown below.

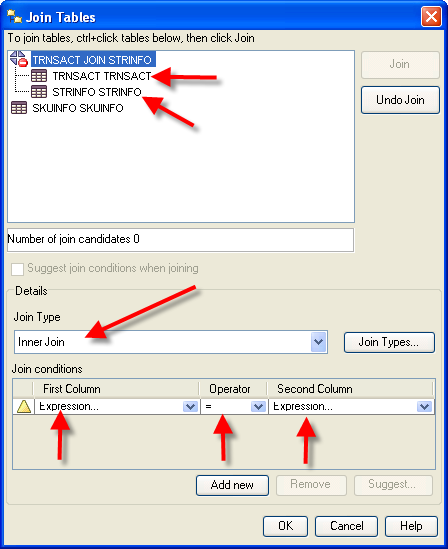
Click the Join button.

SQL Assist then indicates a TRNSACT JOIN on STRINFO and lists the two tables indented underneath the join heading. The Undo Join button now becomes active.

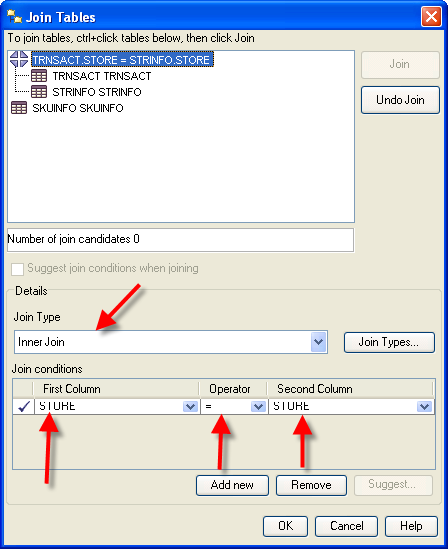
Note the drop down list box for the type of join—the default is Inner Join. This is the type of join needed for this example so the default will be retained.

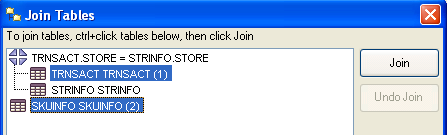
The join conditions allow the user to select the appropriate column from each table via a drop down list box and then select the operator. In this case, the STORE column will be selected from each table to be joined. The operator drop down list box has equal as the default operator; which should be used for this example.

Click the drop down list box for the First Column (the table is TRNSACT) and select Store. Click the drop down list box from the Second Column (this will be the STRINFO table) and select STORE. This will complete the join for these two tables.



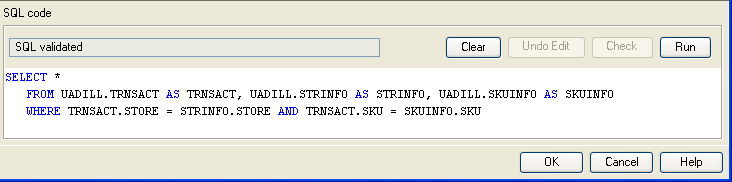
The result of the join of the TRNSACT and STRINFO tables



To join the TRNSACT and SKUINFO tables, select the TRNSACT table and the SKUINFO tables. The Join button will now be active.

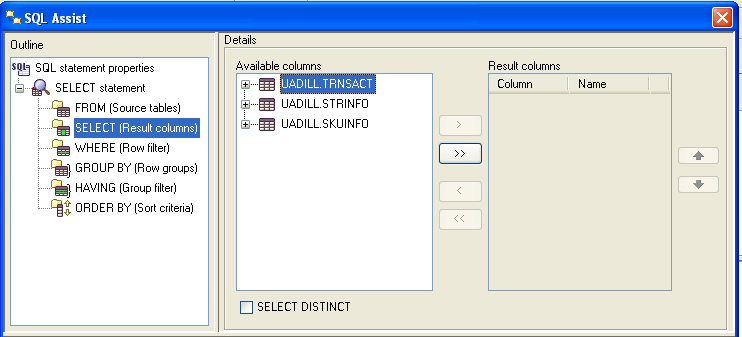
Click the Join button and repeat this process to join the TRNSACT table and the SKUINFO table. The join type should be **Inner Join** and SKU should be the value from each of the columns of the two tables. Click the OK button to exit the Join Tables Dialog Window.

Note the generated SQL in the SQL Code pane. The WHERE clause joins the TRNSACT table to both the STRINFO and SKUINFO tables.

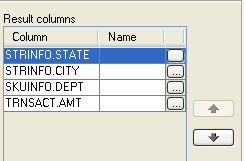


Next the desired columns from the three tables need to be selected. The desired columns are State, City, Department and sales amount—noting that sales will be accumulated for the departments within cities within states.

Click the SELECT entry in the Outline pane of SQL Assist.



Expand the TRNSACT table, select the AMT column and click the > button to move it from the Available columns to the Result columns. Repeat this to move CITY, STATE and DEPT from the Available columns to the Result Columns. Then using, the **UP** or **DOWN** buttons to the right of the Result columns, reorder the Results columns to be STATE, CITY, DEPT and AMT as shown below.

Each of the columns from the table has a corresponding Name column and an ellipsis button. The name column allows the user to enter a more descriptive name than the column name in the table. The ellipsis button opens the Expression Builder – Columns Dialog Window that can be used to create SELECT clause options. In this example, we want a SUM of the AMT so click the ellipsis button for the AMT column. The Expression Builder – Columns is shown below.

Ellipsis button

Up/Down buttons

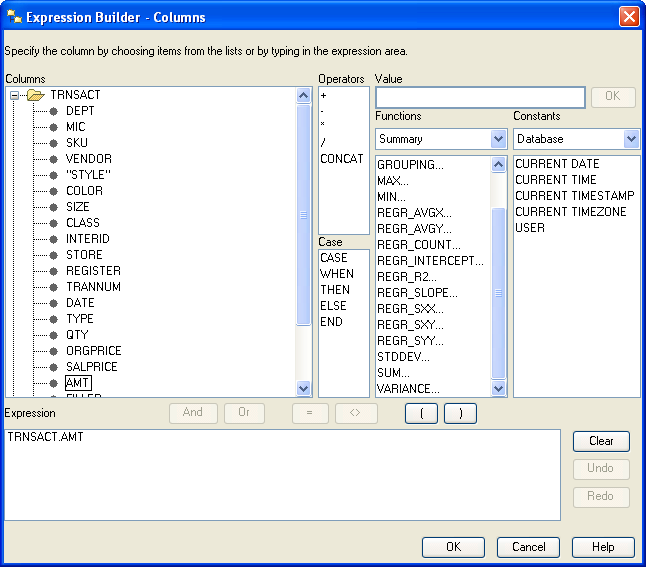
The Expression Builder – Columns allows you to create any legal SELECT clause entries. The Columns pane lists all the tables and columns in an expandable tree. The Operators pane in the top/middle lists all the legal SELECT clause operators. Underneath the Operators pane is the Case pane which allow creating different displayed values for a column based on conditions. The right of the Expression Builder – Columns provides for the Value, Functions and Constants for building the expression. The default value for Functions is All and for Database for the Constants.

Functions and Constants

Operators and Case Structures

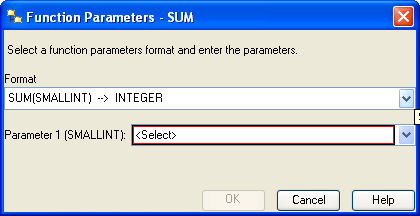
Columns for the

TRNSACT table

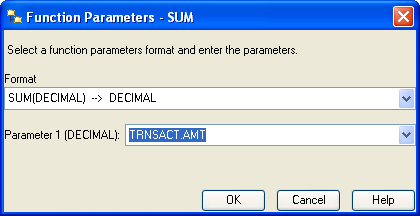


Expression pane and Clear button

For this example, the AMT needs to be summed and given a name. Thus, click the Clear button to remove any entries and then click the SUM function.

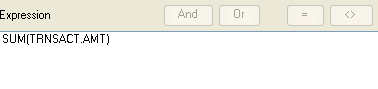


This opens the Function Parameters – Sum Dialog Window. Note that there are two drop down list boxes. The first one is the Format type of the desired output and the default is for integer values (database type of SMALINT). The second drop down list box will have a list of all the columns with a database numeric data type such as SMALLINT, DECIMAL, etc.

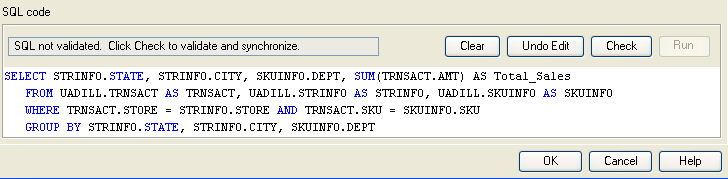


AMT has a database data type of DECIMAL—thus, click the first drop down list box and select DECIMAL. Then, click the second drop down list box and select AMT. The result should be as shown at the right.

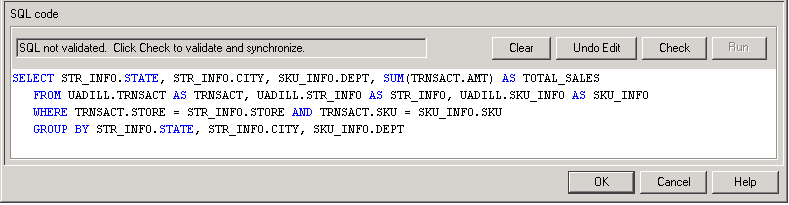
Click the OK button to complete this part of building the expression.

The Expression pane will have the sum part of the expression as shown. The remaining task is to assign a name to be displayed for the sum of the AMT values. For this example, name it Total\_Sales.

Click the OK button to exit the Expression Builder – Columns and simply add AS Total\_Sales to the right of SUM(TRNSACT.AMT) in the SQL Code pane as shown below.

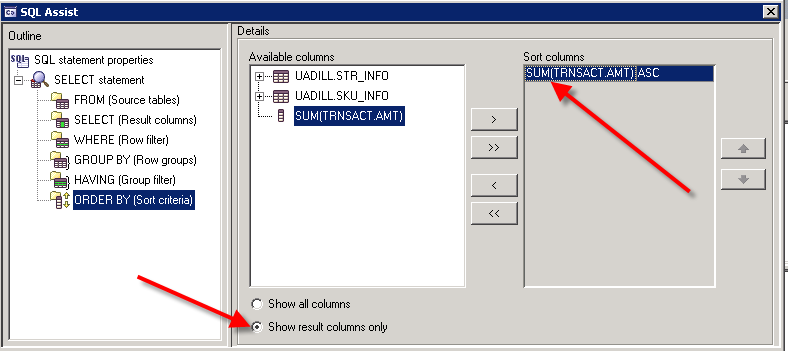


Directly entering SQL in the SQL code pane may result in turning on a warning and making the Check button active so the user can validate the syntax of the entered SQL. That is the case in this example, see above. Click the Check button and the status should return to SQL validated from SQL not validated.



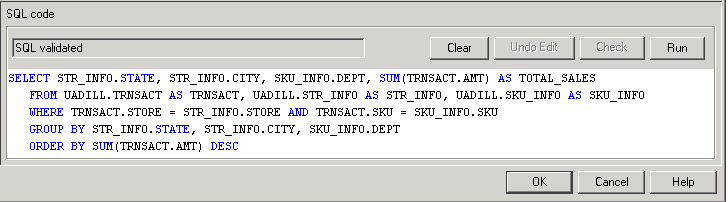
Note also that SQL Assist was smart enough to add the GROUP BY clause which is necessary in this example. Because individual column values and an aggregate value (SUM) are requested to be displayed, then a GROUP BY will be required for all the columns except the aggregate value.

To complete the query for the management scenario, a descending sort on the Total\_Sales is needed. Thus, click ORDER BY in the Outline pane.



After clicking ORDER BY, click the option Show result columns only and then click the SUM(TRNSACT.AMT) entry in Available columns. This places it in the Sort columns – however, the default sort order is ascending (ASC). Via the dropdown list box to the right of the Sort Columns, change the sort from ASC to DESC for the SUM(TRNSACT.AMT) entry.

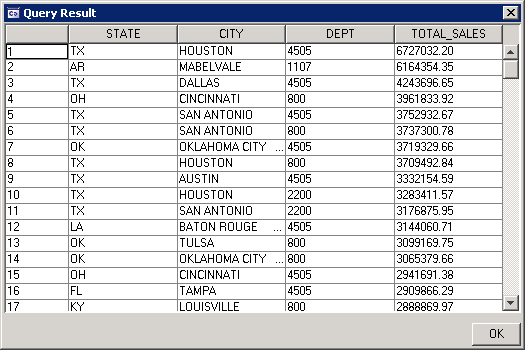
The SQL statement can be completed by adding the range of Dates for the query—can be done directly or with the Expression Builder. The final SQL statement is shown in the SQL code pane and its status is SQL validated. Thus, the syntax for the SQL statement is correct.



Now, of course, this SQL statement could have been entered directly and then its syntax checked via the Check button. Note that alternative equivalent SQL statements are possible. For example, the ORDER BY clause has an equivalent of

ORDER BY 4 DESC or ORDER BY Total\_Sales DESC

which is saying to sort on the 4th column in descending order. Some SQL systems will not allow the use of a user-defined name, Total\_Sales in this case, in the ORDER BY clause. However, DB2 will allow Total\_Sales in the ORDER BY clause. Click the Run button. The results are shown below.

****

The SQL statement for the above output is shown below:

**SELECT STR\_INFO.STATE, STR\_INFO.CITY, SKU\_INFO.DEPT, SUM(TRNSACT.AMT) AS TOTAL\_SALES**

**FROM UADILL.TRNSACT AS TRNSACT, UADILL.STR\_INFO AS STR\_INFO, UADILL.SKU\_INFO AS SKU\_INFO**

**WHERE TRNSACT.STORE = STR\_INFO.STORE AND TRNSACT.SKU = SKU\_INFO.SKU**

**GROUP BY STR\_INFO.STATE, STR\_INFO.CITY, SKU\_INFO.DEPT**

**ORDER BY SUM(TRNSACT.AMT) DESC**