**DB2 Notes**

**SITES**

**Tutorials**

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/sample-db2-cobol-compile-jcl.html>

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/db2.html>

<http://www.sql-tutorial.net/SQL-JOIN.asp>

http://mainframealldtime.wordpress.com/category/db2/

**DB2 TOOLS**

<http://www.dbforums.com/db2/1622056-how-dclgen-can-used-create-host-variables-cobol-db2-program.html>

<https://sites.google.com/site/cobolmaterial/dclgen-tutorial>

<http://mainframestutor.in/dclgen-in-db2/>

**Buffers**

<http://publib.boulder.ibm9.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.intro%2Fsrc%2Ftpc%2Fdb2z_bufferpoolsanddatacaching.htm>

**Example Programs**

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/cobol-db2-sample-program.html>

-----------------------------------------------------------------------------------------------------------------------------

**Imp Points**

* **Common DB2 abends are -206, +100, -551, -803, -805, -811, -818, - 305, -904**

**-206 :** column doesn’t exist in any table of select

**-305 :** null Indicator not specified

**-551:** not authorised to access DB2

**-803:** unique index violation. Tried to insert duplicates

**-805:** dbrm or package not found in the plan

**-811:** cursor should be used, when more than one row is returned as a result of singleton select query

**-818:** timestamp mismatch between plan and load moduleority

**-904:** Unsccesful execution caused by unavailable resource. Like table is migrated or Table has no write auth

**+100 :** end of cursor

* **Imp utilities in IKJEFT01, DSNTIAUL(Program) , DSNUTILB**

If you are planning to unload all rows, DB2 UNLOAD will be the best option.However if you want to unload based on some joins, or need to do casting of data types, and want to use IKJEFT01, you can use the below JCL.

 //**SYSTSIN** DD  \*

    DSN SYSTEM(XXX) RETRY(20)

    RUN PROGRAM(**DSNTIAUL**) PLAN(**DSNTIAUL**) PARMS(**'SQL'**) -

       LIB ('SYS.DB2.XXX.RUNXX.LOAD')

    END

DB2 Load

//STEPM        EXEC PGM=DSNUTILB,  
//                   PARM=(SSID,’SKUMAR’),REGION=0M  
//STEPLIB     DD DSN=TEST.PROCLIB,DISP=SHR  
//SYSPRINT  DD SYSOUT=\*  
//SYSUDUMP DD SYSOUT=\*  
//UTPRINT      DD SYSOUT=\*  
//SYSOUT      DD SYSOUT=\*  
//SYSREC01  DD DSN=TEST.TABLE.INPUT.DATA,DISP=SHR  
//SYSCOPY   DD DSN=&&SYSCOPY,DISP=(,DELETE,DELETE),SPACE=(CYL,(10,5))  
//SORTOUT    DD DSN=&&SORTOUT,DISP=(,DELETE,DELETE),SPACE(CYL,(10,5))  
//SYSUT1       DD DSN=&TEMP,DISP=(,DELETE,DELETE),SPACE(CYL,(10,5))  
//SYSMAP     DD DSN=&&SYSMAP,DISP=(,CATLG,CATLG),SPACE(CYL,(10,5))  
//SYSIN DD \*  
LOAD DATA  
INDDN SYSREC01  
REPLACE  
COPYDDN DB2COPY4  
STATISTICS TABLE(ALL) INDEX(ALL) UPDATE ALL  
SORTNUM 12  
WORKDDN(DB2UT1,DB2SORT)  
ERRDDN DB2ERR  
MAPDDN DB2MAP  
LOG NO  
ENFORCE NO  
INTO TABLE DBSSID.TESTTBLE  
( COL\_1 POSITION(1:4) INTEGER  
, COL\_2 POSITION(5:6) CHAR (2)  
, COL\_5 POSITION(7:10) DECIMAL  
NULLIF NULL1=’?’  
, NULL1 POSITION(11:12) CHAR(1)  
)  
/\*

* /\*               <http://www.youtube.com/watch?v=lNkI710RxMc>     - **installation of DB2 Express C**

**DB2 windows -** detailsdb2admin/sandynami

* Good Book - **DBMS by CJ Date**
* **DB2 Certifications**

Different types of data applications, **data warehousing**, and **OLAP**.

* **NAB – National Australian Bank**

The project uses a combination of IMSDB and DB2 for the business requirements.

The DB2 part is mainly used for rules and after checking the rules the required data is fetched from IMSDB.

IMSDB contains segments

**Table structure**

**Srve\_mgmt\_parm\_unit**

MPID   PMPID  EFFDATE  STATUS PERC  INDICATOR

CAMS – Customer savings

CASB  - Business Accounts

* **Fetch last 10 records with DB2**

select \* from PD.AC\_ROLE\_IN\_SRVC

order by ENTRY DESC

FETCH FIRST 10 ROWS ONLY

* The system catalog is updated whenever you Create a DB2 database object.
* DECIMAL(100,2) equivalent of cobol variable ?S9(98)V2
* The DECLARE statement is similar to the CREATE statement, except that it is used to create temporary tables that exist only for the duration of a database connection.
* A table is the only object that can be declared. **The system catalogue is not updated** when you declare a temporary table.

DECLARE GLOBAL TEMPORARY TABLE session.emp1

 LIKE employee

 ON COMMIT PRESERVE ROWS

 NOT LOGGED

 IN tempspace

The rows of the temporary table will be preserved (not deleted) whenever a COMMIT statement is processed.

* Finally, changes to the temporary table are not logged (this is the only option).

 Session is a schema name.

* You cannot alter an index. You must drop it and then create a new one with a different definition
* **Select Syntax**

SELECT  < Column Names > / \* / function(< Column Name >) FROM < TABLE NAME >

  [WHERE < Condition > ....]

* INSERT statement is used to insert record(s) into table / view.
* Inserting a row into view also inserts the row into table.

**Simple Syntax**

INSERT INTO < TABLE NAME / VIEW NAME > [ (<COLUMN NAME1>,<COLUMN NAME2>...) ]

  VALUES ( value1, value2,.... )

* **Inserting the data with full select**

CREATE TABLE emp LIKE EMPLOYEE

  INSERT INTO emp

  SELECT           EMP\_ID,EMP\_FIRST\_NAME,EMP\_M\_NAME,EMP\_LAST\_NAME,SALARY, DEP

  FROM EMPLOYEE

  WHERE DEP = 002

* Eliminate nulls during select

 CREATE TABLE employee (emp\_id INT, savings\_in\_401k INT, total\_salary INT);

INSERT INTO employee VALUES(1, 5000, 40000);

INSERT INTO employee VALUES(2, 0, 40000);

INSERT INTO employee VALUES(3, NULL, 100000);

SELECT emp\_id, savings\_in\_401k AS employer\_match FROM employee WHERE

    CASE WHEN(savings\_in\_401k IS NULL) THEN 0

         ELSE savings\_in\_401k END \* 0.06 > 0;

select 1000, 'Lukas', 'Eder', '1981-07-10', 1981,

    cast(null as varchar(200)) from SYSIBM.DUAL

**COALESCE DB2 function** returns the first non-null value in a list of input expressions (reading from left to right). Each expression is separated from the prior by a comma. All input expressions must be compatible. VALUE is a synonym for this function.

**Example**:

**Input**:

**ID NAME   DEPT JOB YEARS SALARY COMM**

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

 10 Sanders  20 Mgr   7 18357.50 -

 20 Pernal   20 Sales 8 18171.25 612.45

 30 Marenghi 38 Mgr   5 17506.75 -

 40 O’Brien  38 Sales 6 18006.00 846.55

 SELECT ID

 ,COMM

 ,COALESCE(COMM,0)

 FROM STAFF

 WHERE ID < 30

 ORDER BY ID;

**Answer**:

**ID COMM    3**

 10  -      0.00

 20  612.45 612.45

**DB2 Compilation procedure**

Db2 compilation can be explained as

1. PRECOMPILE DB2 PROGRAM (PGM=DSNHPC)

   a.When we precompiled, precompiler will create the DBRM, it will

     be placed in the pds specified here(DBRMLIB  )

   b.We need to give location of DB2COBOL Program as well(SYSIN )

   c.Needs to speficiy DCLGEN member locations(SYSLIB )

2. Compilation(IGYCRCTL)

a.Needs to speficiy DCLGEN member locations(SYSLIB )

3. LINK EDIT(IEWL)

a.Load module location, load module will be created here. this

location needs to be given in run jcl.(SYSLMOD)

4.BIND - BIND THE DB2 PACKAGE(IKJEFT01)

    a. specify the location of DBRM(DBRM)

b.SYSTSIN

//SYSTSIN  DD  \*

DSN SYSTEM (DEVDB )

BIND  MEMBER    (DB2PROG) -

  PACKAGE   (PACKG11) -

  LIBRARY   ('DEV.SURESH.DBRM') -

  ACTION    (REP) -

  ISOLATION (CS) -

VALIDATE  (BIND)-

  RELEASE   (COMMIT) -

  OWNER     (SURESH) -

  QUALIFIER (DEVQUALI)

/\*

* **This is primarily a copybook of host variables and also contains table structure and helps in programming reference.**

Optimising access of data in DB2 can be done by.

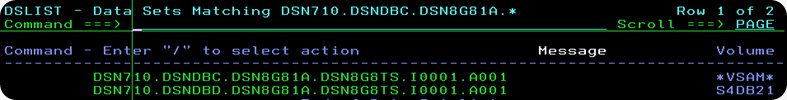
1. RUNSTATS
2. REORG
3. RUNSTATS
4. REBIND packages
5. FLUSH PACKAGE CACHE DYNAMIC (on LUW) or whatever the equivalent is on z/OS.

* DB2 is a complete Software Package that allows you to create, store, update and retrieve Data in a Systematic Manner.   
    
  1. Data Abstraction: It presents a Logical, Structured view of the Data to the User. The user doesn’t need to worry about the details of how the Data is physically stored on the DASD-Disk.   
  2. Data Independence: DB2 provides a layer of Independence, between the application or COBOL Program and the Data. Every-time the structure of the Data changes (say you add a new field to the EMPLOYEE File), you don’t need re-write the entire COBOL Source code again.   
  3. Data Security: DB2 Software controls the Access to the Data. Not everybody can see all the Data. You can see it, only if you've got all the necessary privileges.   
  4. Transaction Management and Concurrency: DB2 manages all the Transactions, and ensures the Data remains consistent. It provides a Locking-Mechanism, for concurrent access to the Data.   
  5. Data Recovery: When there are Crashes, DB2 allows you to recover swiftly, and restore the original Data.
* Database: A Database is a Logical-Grouping of Database Objects, related together by application, or subsystem
* **The information about all the STOGROUP’s (Storage Groups) is stored in the DB2 Catalogue Table SYSIBM.SYSSTOGROUP**
* **The information on all the Databases in DB2 System is stored in the DB2 Catalog Table SYSIBM.SYSDATABASE.**
* STOGROUP: The Set of DASD Volumes that will be used to house the VSAM Files containing the Data.

Table-SPACE: A Table-Space is a VSAM File which stores or houses the DB2 Table-Data Physically

buffer-pool is the place in Main Storage, where DB2 will store Pages fetched from the Table-Space.

The information about all Table-Space Files is stored in the DB2 Catalogue Table **SYSIBM.SYSTABLESPACE**.



-----------------------------------------------------------------------------------------------------------------------------

**Questions and their Explanation**

**Q: Can we declare DB2 HOST variable in COBOL COPY book?  
A:** NO. If we declare DB2 host variable in COBOL COPY book, at the time of Pre-compilation we get the host variable not defined, because pre-compiler will not expand COBOL COPY book. So we declare it either in DCLGEN with EXEC SQL INCLUDE DCLGEN NAME END-EXEC or we directly hardcode it in the working storage section.

**Q: What should be specified along with a cursor in order to continue updating process after COMMIT?  
A:** With Hold option.

**Q: What is the name of the default DB2 catalog database?  
A:** DSNDB06

**Q:**[**CURSORS in DB2**](http://mainframegeek.me/2011/08/17/cursors-in-db2/)

When we are using the DB2 in our applications we can only have one row of data at a time. So what we will do if we don’t know which row exactly we need?, what if we have more than 1 row to work with? well the answer is “CURSORS”.

Cursor is used when more than one row are to be selected.  Cursors has mainly 4 control statements.

1. Declare.

A name will be assigned for particular SQL statement. The name should be unique in the scope of the program. there are no limits for the number of cursors which we can have in one application program. We can declare cursor in Working storage section or Procedure division.  
E.g.  
EXEC SQL                                    
    DECLARE CURREAD1 CURSOR FOR             
        SELECT NAME,SEQ FROM IBMGRP.MYNAM   
END-EXEC.

2. Open.

This statement builds the resultant table.   
E.g.   
EXEC SQL          
   OPEN CURREAD1  
END-EXEC.

3. Fetch.

Fetch statement will returns data from the resultant table (One row at a time) and assigns values to the specified host variables.  
E.g.  
EXEC SQL                                 
   FETCH CURREAD1 INTO :WS-NAME,:WS-SEQ  
END-EXEC

4. Close

Empty all the resources used by the cursor.  
E.g.  
EXEC SQL           
   CLOSE CURREAD1  
END-EXEC.

**All these control statements will throw specific SQLCODES.**

Few Snippets.

Read table.

EXEC SQL                                   
    DECLARE CURREAD1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
END-EXEC.   
Open  
EXEC SQL          
   OPEN CURREAD1  
END-EXEC.

PERFORM UNTIL SQLCODE = 100                            
   EXEC SQL                                            
      FETCH CURREAD1 INTO :WS-NAME,:WS-SEQ             
   END-EXEC                                            
   MOVE SQLCODE             TO WS-SQLCODE              
   DISPLAY ‘SQLCODE FETCH ‘ WS-SQLCODE                 
   IF SQLCODE = 0 THEN                                 
    MOVE SPACES TO TEMP-MSG                            
    STRING                                             
    ‘NAME: ‘ DELIMITED BY SPACE ‘ ‘ DELIMITED BY SIZE  
    WS-NAME DELIMITED BY SPACE ‘,’ DELIMITED BY SIZE   
    ‘SEQ#’ DELIMITED BY SPACE ‘ ‘ DELIMITED BY SIZE    
    WS-SEQ DELIMITED BY SIZE INTO TEMP-MSG             
    DISPLAY TEMP-MSG                                   
   END-IF                                              
END-PERFORM.

Fetch name and SEQ till we hit SQLCODE 100 and display the data.

Update Table

We need to mention FOR UPDATE OF and the field name in declare statement

EXEC SQL                                   
    DECLARE CURUPDT1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
        WHERE SEQ = :WS-SEQ                
        FOR UPDATE OF NAME                 
END-EXEC.        
here I will be updating NAME field of MYNAM table                             
EXEC SQL                                   
   OPEN CURUPDT1   
END-EXEC.  
MOVE ’002′               TO WS-SEQ.          
EXEC SQL                                     
   FETCH CURUPDT1 INTO :WS-NAME,:WS-SEQ-TMP  
END-EXEC                                          
EXEC SQL                                     
   UPDATE IBMGRP.MYNAM                       
   SET NAME = :WS-NAME                       
   WHERE CURRENT OF CURUPDT1                       
END-EXEC.      
**“CURRENT OF CURUPDT1″ statement will pick the current row to update.**      
EXEC SQL                                     
   CLOSE CURUPDT1                            
END-EXEC.

Delete Record

Like cursor for updating a record we need to mention FOR UPDATE OF in cursor declaration statement.

EXEC SQL                                   
    DECLARE CURDELT1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
        WHERE SEQ = :WS-SEQ                
        FOR UPDATE OF NAME                 
END-EXEC.                                  
EXEC SQL                                   
   OPEN CURDELT1                           
END-EXEC.                                                                 
EXEC SQL                                     
   FETCH CURDELT1 INTO :WS-NAME,:WS-SEQ-TMP  
END-EXEC                                     
EXEC SQL                      
   DELETE FROM IBMGRP.MYNAM   
   WHERE SEQ = :WS-SEQ        
END-EXEC.                     
EXEC SQL                
   CLOSE CURDELT1       
END-EXEC.

**Q. Multi row Fetch Overview in COBOL Db2 program WITH EXAMPLE:**

**Ans.**

Cobol Db2 multi row fetch concept will be used to retrieve multiple rows with a single fetch statement as opposed with a normal cursor which fetches only single record at a time.

**The multi-row fetch capability was introduced to DB2 z/OS in Version 8.**

This can be achieved in two different ways

1.One way is to declare the cursor as normal one and in Fetch by using the row count

2.By using the Row set positioning cursors

**First Method:**

The operations used to define, open, and close a cursor for a multiple-row remains the same. Only the FETCH statement changes to specify the number of rows to retrieve and the storage where the rows are placed.

Need to declare a host structure of array to capture the fetched records. The size of the array depends on your requirement (if you need 10 rows at a time, declare the array with 10 occurrence).

As said earlier there are no changes in declare, open & close statement syntax. Only the fetch syntax differs.

**Ex:**

EXEC SQL FETCH CUR1 FOR 15 ROWS INTO: ww-host-array

Here CUR1 is the name of the cursor

ww-host-array is declared with 15 occurrences.

**Multi-row fetch by using the row-set positioning cursors:**

Row-set is nothing but group of rows returned by a single fetch statement.

The maximum size of row set is 32767

Procedure to use row-set positioning cursors:

1) declare the cursor with row-set option

**Ex:**

EXEC-SQL

     DECLARE cur1 CURSOR WITH ROW-SET POSITIONING

     FOR select \* from ------

END-EXEC

Here we can specify either WITH ROW-SET POSITIONING or WITHOUT ROW-SET POSITIONING. WITHOUT ROW-SET POSITIONING is the default one.

2) Open the cursor, it is same as normal cursor open statement.

3) Fetch the cursor with row-set option

FETCH NEXT-ROWSET FROM cur1 FOR 10 ROWS

       INTO :ww-host-array

Here there are multiple options while fetching

Fetch FIRST-ROWSET – Fetches first 10 rows

Fetch NEXT-ROWSET – fetches next 10 rows

We can also have the PRIOR-ROWSET, CURRENT-ROWSET, LAST-ROWSET & ROWSET STARTING AT options.

By using the rowset starting option we can specify from where to start fetching and how many records to retrieve from that position.

Ex:

FETCH ROWSET STARTING AT ABSOLUTE 15 FOR 10 ROWS.

In this case it retrieves 10 rows starting from the position 15.

Advantages of multi row fetch:

1.reduces the number of program to database calls, thus decreases the burden on DB2.

2.As it also supports positioned updates and deletes, here also the performance gets increased.

**Q. What are Stored Procedures explain in detail?**

**ANS.**

**Q. List various Operating systems where DB2 is being used.**

**Ans**

OS/390, OS/400, Z/OS, UNIX, Intel Platforms.

Internally DB2 on OS/400, VM/VSE and OS/390 differ from DB2 on the UNIZ and Intel platforms but common SQL API is used across all platforms.

Also note that DB2 code base on UNIX and Intel is identical.

**Q. List other RDBMS like DB2.**

**ANS.**

DB2, ORACLE, SYBASE, INGRES, Informix

Q. DB2 Datatypes and COBOL equivalents.

ANS.

|  |  |
| --- | --- |
| **Datatype** | **COBOL Declaration** |
| SMALLINT | 01 WS-VAR PIC S9(4) COMP |
| INTEGER | 01 WS-VAR PIC S9(9) COMP |
| DECIMA(X,Y) | 01 WS-VAR PIC S9(X-Y)V9(Y) COMP-3 |
| CHAR(N) | 01 WS-VAR PIC X(N) |
| VARCHAR(N) | 01 WS-VAR.       49 WS-VAR-LEN PIC S9(4) COMP.       49 WS-VAR-TEXT PIC(N). |
| DATE | 01 WS-VAR PIC X(10) |
| TIME | 01 WS-VAR PIC X(8) |
| TIMESTAMP | 01 WS-VAR PIC X(26) |

Further DB2 datatypes are classified as

**1.Numeric**

Smallint – Binary half word

Integer – Binary Full word

Decimal(X,Y) – X numeric digits of which last Y are fractioned digits

**2.Character**

Char(X) – 0 to 255

VARCHAR – Page Size

LONG VARCHAR

**3.Graphics**

**4.Data/Time**

Date – YYYYMMDD

TIME – HHMMSS

TIME STAMP - YYYYMMDDHHMMSSNNNNNN

**5.User Defined Types**

**Q. What will happen when null with default is coded while creating a table?**

**And some more information about nulls.**

**ANS**

When null with default is coded, in the absence of data default value is populated as per the below table.

|  |  |
| --- | --- |
| **Data Type** | **Default Value** |
| Numeric | 0 |
| Fixed-length String | Spaces |
| Varying-length String | Length of string zero |
| Date | Current date |
| Time | Current time |
| Timestamp | Current Timestamp |

**Note**

**1** NULLS are considered equal when scalar functions like Distinct or COUNT is used in SELECT queries.

2.The default option for columns while creating tables is to allow nulls

3.Avoid nulls in the columns that must participate in arithmetic operations.

AVG, COUNT DISTINCT, SUM, MAX, MIN functions omit column occurrences set to null.

**Q. All about isolation levels**

**Ans**

What are the DB2 Isolation Levels ?

**A brief description of all four isolation levels used by DB2**

1. UR - Uncommitted Read

– For read only queries, no record locking  
– Will see uncommitted changes by other transactions  
– Good for accessing read only tables  
– Statements in UR which modify data are upgraded internally to CS

2. CS - Cursor Stability

– Default isolation level  
– Locks and unlocks each row, 1 at a time (never has 2 locks at once)  
– Guaranteed to only return data which was committed at the time of the  
read

3. RS - Read Stability

– Will keep all qualifying rows locked until the transaction is completed  
– Does release locks on rows that do not satisfy query predicates  
– Use for result set stability or when future actions on returned rows may be taken

4. RR - Repeatable Read

Locks the table within a unit of work. An application can retrieve and operate on rows in the table as many times as needed. However, the entire table is locked, not just the rows that are retrieved. Until the unit of work completes, no other application can update, delete, or insert a row that would affect the table.

DB2 offers four locking isolation levels: Repeatable Read (RR), Read Stability (RS), Cursor Stability (CS), and Uncommitted Read (UR).

Read my previous post on [basics of isolation](https://srinimf.com/2010/08/03/db2-isolation-levels/) levels.

Each of these isolation levels allows the user and application to control the number and duration of read (Share) locks held within a unit of work.

When you set the appropriate isolation level, based on a particular application’s requirement, lock resources can be minimized, and the user/program concurrency can be increased. Take the following example:

SELECT LASTNAME, EMPNO  
FROM EMP  
WHERE LASTNAME LIKE 'S%'  
WITH UR

**Repeatable Read**

With RR means that the same query can be executed multiple times within the same unit of work, and the results of the query will be identical every time (repeatable). A Share lock will be set and will stay on each row or page until the query or logical unit of work has completed.

All accessed rows or pages are locked, even if they do not satisfy the predicate.

For table scans, this would encompass each row/page within the table. For other queries not processing table scans, this would encompass any rows or pages that meet the predicate criteria of the SQL statement.

In the example above, this would be all rows or pages containing last names that begin with S.

All Share locks with RR are held until a commit takes place. These share locks would effectively prevent updates, inserts, or deletes (X locks) from occurring on any of the rows/pages from any other process until a commit is executed.

**This is a common reason for many -911 SQLCODE errors.**

Note Most query tools on the market have their default isolation level set to RR, which is not good. This causes many problems in environments where users, analysts, developers, and others query the data often during the day. Many times users leave their workstations while a query running in the background is applying and holding locks on the data being retrieved.

**Read Stability**

With RS is very much like With RR, except that it will allow inserts from other users.

It can at times lock more rows/pages because locks are taken and held on data, even when it goes to stage 2 processing to further check predicates.

If there is a stage 2 predicate and the data does not fit the predicate criteria, the RS lock is still placed and held.

**Cursor Stability**

With CS sets a Share lock on each row or page processed, and the moment the cursor moves on to another row or page, it releases the lock.

So at any one time, there is only one lock being held either on a row or page of data. This obviously allows good concurrency and some data integrity. Almost all batch COBOL programs in IT shops today are bound with the locking parameter CS. This is because as these programs execute cursor processing, they have no need to reread any data processed. The Share locks get freed up as the query moves through the cursor, and the query has data integrity as it processes each current row or page.

**This bind parameter, along with another bind parameter, Currentdata(No), provides an opportunity for avoiding locks altogether. With these two bind parameters together, DB2 can test whether a row or page has committed data on it, and if it has, DB2 will not have to obtain any lock.**

**Uncommitted Read**

With UR means that no Share locks are placed on any rows or pages processed by this query, and it does not matter if other processes have any locks on any of the data being retrieved.

This can improve efficiency because it reduces overall processing time. But the one issue in using UR is that if some other process has applied updates to data being retrieved, UR will return the updated data from the buffer before the other process has executed a commit.

If for some reason the other process does a rollback of its updates, then this UR process has updated data that was never committed.

Even with the issue of possibly picking up non-committed data, there are definitely times when UR can be used:

**Q. Db2 union and union all**

**Ans**

**UNION – eliminates duplicates**

**UNION ALL – duplicates retained**

**Q. Explain use**

**Ans**

**Explain is used to display access path**

**Q. Complete explanation of Load and unload functionality.**

**Ans**

Sample JCL for UNLOAD & LOAD utility for DB2 TABLE - What is CHECK & COPY PENDING

Below are the Sample JCLs for unload and load utility in DB2.  
**Loading and Unloading of data from DB2 TABLE**  
  
The below JCL shows how to UNLOAD data from one table to a dataset using IKJEFT01.  
  
//STEP02   EXEC PGM=IKJEFT01,DYNAMNBR=50  
//SYSTSPRT DD     SYSOUT=$  
//SYSPRINT DD     SYSOUT=$  
//SYSUDUMP DD  SYSOUT=$  
//SYSTSIN  DD DSN=TEST.PROCLIB(UNLD),DISP=SHR  
//SYSPUNCH DD DSN=TEST.TABLE.STRTCTURE,  
//                                       DISP=(NEW,CATLG,DELETE),  
//                                       UNIT=DISK,SPACE=(CYL,(1,1),RLSE)  
//SYSREC00 DD DSN=TEST.TABLE.LOAD,  
//                                     DISP=(,CATLG,DELETE),  
//                                     UNIT=CART,  
//                                     DCB=(SYS3.DSCB),LABEL=EXPDT=99000  
//SYSIN    DD \*  
SELECT \*  FROM TST.CLIENT ;  
/\*  
//\*  
SYSIN DD\* contains the condition for the SQL query we want to use for fetching the data.  
SYSTSIN  DD DSN=TEST.PROCLIB(UNLD) ->  This will contain the details like DB2 level(test or prod) ie,from where we are dowloading. Also the utility, plan , and the library used to pickup the utility like below.SYSREC00 will contain the actual unloaded records from DB2.

The above JCL  one loads the table CLIENT into TAPE. For loading to DISK use below.  
  
//SYSREC00 DD DSN=TEST.TABLE.LOAD,  
//                                     DISP=(,CATLG,DELETE),  
//                                     UNIT=DISK,SPACE=(CYL,(100,100),RLSE)  
  
**Load the downlaoded data from dataset to test DB2 Table**  
   
This JCL shows how we can upload the data from Flat File back to DB2 Table.  
  
//LOADTB    EXEC DB2LOAD,LDNM=TBLOAD1,DB2=DB2T  
//LOAD.SYSREC00 DD  DSN=TEST.TABLE.LOAD,DISP=SHR  
//\*  
//LOAD.SYSIN    DD  DSN=TEST.TABLE.STRTCTURE,DISP=SHR  
//\*  
  
DB2LOAD is the utility used to execute the load function.  
LBNM=TBLOAD1 ::  We can use any name here inplace of TBLOAD1  
DB2=DB2T:: indicates a test database region.  
TEST.TABLE.LOAD  contains the data to be loaded into the table which was downloaded in the unload step.  
LOAD.SYSIN contains the table structure along with additional parameters to load the table.  
like  
LOAD DATA REPLACE LOG NO ENFORCE NO INDDN SYSREC00  
INTO TABLE TST.CLIENT.

**NOTE:**We Get the table structure in the dataset with DD name SYSPUNCH while we unload the data in previous step  
  
**CHECK PENDING AND COPY PENDING:**  
  
Check pending is a tablespace/index status set when DB2 has reason to believe that an object might be in an inconsistent state. Usually happens as  a result of running LOAD on a tablespace containing a table that has Referential Integrity constraints, but  that was  not enforced during the LOAD OPERATION (ENFORCE NO). (like we used in above example). Recovering and RI tablespace to a previous point in time can also set it. None of this is directly related to the design of the application.If CHECK PENDING is set you can run the **CHECK utility** to instruct DB2 to verify that the data is consistent and switch off the pending status.  
 To remove the Checkpending condition we need to  execute the following commands on tablespace  
for all the affected table space.  
  
REPAIR  OBJECT  LOG NO  
SET  TABLESPACE TBLA00DB.TBLB01TS  
NOCHECKPEND  
REPAIR  OBJECT  LOG NO  
SET  TABLESPACE TBLB00DB.TBLB02TS  
NOCHECKPEND  
  
**COPYPENDING**  
Any table space will be in copy pending status after  load is performed on the DB2 table,  
with log no parameter  
or  
if load is performed without taking an image copy  
or  
if the job abend during load  
or  
if NOCOPYPEND parameter is not specified in syspunch dataset   
To remove the Copypending  condition we need to  execute the following commands on tablespace for all the affected table space.   
  
REPAIR  OBJECT  LOG NO  
SET  TABLESPACE TBLA00DB.TBLB01TS  
NOCOPYPEND  
REPAIR  OBJECT  LOG NO  
SET  TABLESPACE TBLB00DB.TBLB02TS  
NOCOPYPEND   
  
**Is There any way to check which operation has put the table on pending status??**   
  
Yes,We can know this from SYSIBM.SYSCOPY Table.  
If we run a query against this table, the ULITY values can help us in determining the desired operation.  
example:  I want to check what last operation was performed by *my id or any id.*  
So, i Can query the SYSIBM.SYSCOPY Table like below  
SELECT DSNAME SPACE, ICTYPE AS UTILITY, TIMESTAMP  
FROM SYSIBM.SYSCOPY WHERE AUTHID='my-id'  
ORDER BY TIMESTAMP  
If we check the output, check for UTILITY  values

UTILITY Can have these values like 'A', 'S' .. like below.

A-Alter  
B-REBUILD IX  
D-CHECKDATA LOG(NO)  
F-full IC  
I-Incremental IC  
P-partial recovery point  
Q-Quiesce  
R-LOAD REPLACE LOG(YES)  
S-LOAD REPLACE LOG(NO)  
T-terminated utility  
W-REORG LOG(NO)  
X-REORG LOG(YES)  
Y-LOAD RESUME LOG(NO)  
Z-LOAD RESUME LOG(YES)

Thus we can check the operation last performed.   
  
**Common errors encountered while loading a table**  
  
UTILITY BATCH MEMORY EXECUTION ABENDED, REASON=X'0B37'  
  
Check the spool for which dataset it is abending with SB37.  
Then we need to change the UNIT parameter.  
If it is using SYSDA, we can use multivolume coding the SYSDA like  
UNIT=(SYSDA,59),  
SPACE=(CYL,(200,200),RLSE)  
  
DSNURSIX - KEY COUNT INCONSISTENT FOR SORT-RD PHASE,NUMBER OF KEYS EXPECTED =72379826, NUMBER OF KEYS PROCESSED=8681906  
  
DSNUGSOR - SORT PHASE STATISTICS -   
NUMBER OF RECORDS=0  
ELAPSED TIME=00:00:00  
 DSNURSIX - KEY COUNT INCONSISTENT FOR SORT-RD PHASE, NUMBER OF KEYS  
PROCESSED=0  
DSNUGBAC - UTILITY BATCH MEMORY EXECUTION ABENDED, REASON=X'00E4030A  
  
Resolution:  
1. Increase the SORT datafiles used internally in the load utility.  
2. Terminate the UTLITY  
**How to terminate?**  
use the command -TERM UTIL(UTIL-NAME)  
**What is this UTIL-NAME?**  
Its just the LDNM name we use in our load job. In the above example, the utility name is  
TBLOAD1   
**where to use this command?**  
Go to  DB2I PRIMARY OPTION MENU -> option 7(DB2 COMMANDS)  
3. Restart the load job  
  
ERROR LOADING INDEX, ERROR = X'00E40322' INDEX = Bth00.D4xx30X1  
DSNUGBAC -RESOURCE UNAVAILABLE  
      REASON 00D70014  
      TYPE 00000220  
      NAME TEX2.DSNDBC.DXXXXDB.D890000X1.I0001.A001  
UTILITY DATA BASE SERVICES MEMORY EXECUTION ABENDED, REASON=X'00E40322'  
  
Explnation: The load phase of the load job worked, but the rebuilding index failed.  
00D70014 - An attempt to extend a data set failed, because a problem was detected in media manager services.  
  
Resolution:  
1.Check the allocations for the index of the table.It must be too small.  
We need to increase it, Both primary and seconday quantity for all the index available for the table.  
2. Terminate the utility using the same command  -TERM UTIL(UTIL-NAME)  
3. restart the load job again.

**Q. Expain about the types of SQL?**

**ANS.**

**Static SQL-** is the one embedded in the application program written in high level language. (access path is known before the execution of the query)

**EG** – a full select cursor written to fetch the records from DB2

**Dynamic SQL-** is either typed in the terminal for real time execution(QMF) or constructed in an application program at the run time(Prepare statement).

**Q. All about access path and performance optimisation.**

**Ans**

The access paths formulated by the Db2 optimizer during the BIND process are critical to application performance. It is these access paths that determine how efficiently Db2 data is read and delivered to an organization’s programs.

The stability of access paths is important. And with customers executing thousands of distinct SQL statements millions of times per day, there can be a disruption to their system performance if the optimizer is not choosing the most effective access path for any single SQL.

There are many events that can occur that might cause Db2 to change from a good access path to the bad access path, or where a suboptimal path is initially chosen. Customers desire 100% perfection in access path selection, which has eluded commercial DBMSs. With the mission critical nature of Db2 for z/OS customers, any single poor optimizer access path choice is undesirable.

Based on customer engagements our own distinguished engineer estimates that 25–30% of customer DBA time is spent managing access paths — where even minor performance regressions can cause disruption to their business and increase the impact to IBM support. Getting applications to run efficiently can help reduce CPU costs and meet the service level agreements expected by the business. This can be a constant battle that requires DBAs to possess acute analysis and tuning skills.

When you apply any of the following methods to influence access path selection, *Db2* uses information that you provide during access path selection. For static SQL statements, *Db2* validates and uses the information when you rebind the package that contains the statements. For dynamic SQL statements, *Db2* applies, validates, and uses the information when the statements are prepared.

You can use the BIND QUERY command to influence access path selection at the statement level. When you use these methods, *Db2* applies the information for SQL statements that match the statement text you specify, in any of the following contexts:

1. System-wide
2. From any version of particular collection and package
3. From a particular version of a collection and package

You can also insert values in a PLAN\_TABLE instance to specify access paths.

After running the JCL that performed a REORG and RUNSTATS on the tablespace (C), I re-bound the affected PACKAGE. This is the most important step; not re-binding the affected package will not force DB2 to re-evaluate the access path.

I re-measured the application after the changes and now my results are much more in line with what I would hope for this application.

Scenario – your query is performing poorly.  Checking an Explain, you see DB2 picked an access path which you think is less than optimal.  First, you make sure there are appropriate indexes.  Then you look to see if the statement is written in a way that enables DB2 to use those indexes (e.g., expressions on indexed columns prevent index usage - until DB2 for z/OS V11, which can rewrite some of those).  The code looks good to you.  So, what else could be causing the poor choice of access path?

DB2 has a cost based Optimizer, which calculates the estimated cost of each possible access path.  One of the major components of cost is the number of rows that are qualified at each step.  This in turn is determined by the Filter Factor of each predicate.

DB2 filters out rows that do not meet the predicates, but the filter factor is the fraction of rows that *satisfy* that predicate.  The term Selectivity is also used to mean the same thing, and different parts of the DB2 documentation use each term.

1. SELECT …. FROM CUST WHERE LASTNAME=‘PURCELL’
2. If there are 1,000,000 rows in the CUST table and 100 are named PURCELL, then we are selecting
3. 100/1,000,000 = .0001 of the rows  (that is 1/10,000)
4. The Filter Factor is .0001

If DB2’s estimate of the filter factor is incorrect, that can clearly lead to an inaccurate cost estimate and therefore DB2 may pick a different access path.  Looking at the explain (Visual Explain will clearly show you the filter factors for every predicate), you can see if they match the actual data in the tables.  If they don’t, you have to find a way to give DB2 more accurate information.

**Combining Predicates**

1. AND
   1. If the predicates are independent, you multiply the filter factors
      1. Sex = ‘M’ AND Salary\_grade = 23
         1. ½ \* 1/20  =  1/40  (.025)  - 25,000 rows
   3. If they are dependent (such as city and state)
      1. If DB2 knows they are dependent,  then it does not multiply.  DB2 will know that columns are not independent if the predicate columns are the leading columns of an index  or if you have collected colgroup statistics on those columns.
      2. LUW – index stats have firstkeycard, first2keycard, first3keycard, first4keycard, fullkeycard
      3. z/OS – index stats have firstkeycardf and fullkeycardf, but all intermediate combinations are gathered too and stored in SYSCOLDIST
      4. BETWEEN – even though between is equivalent to a >= predicate AND a <= predicate, DB2 does not treat them as independent
2. OR
   1. If the predicates are exclusive, then the filter factors are added
      1. Salary\_grade = 23 OR Salary\_grade = 17
         1. 1/20 + 1/20 = .1     100,000 rows
   2. If the predicates are inclusive, then the filter factor is less than the sum of the individual FFs.  For different columns, DB2 assumes the predicates are not exclusive
      1. Sex = ‘M’ OR Salary\_grade = 23
         1. ½ + 1/20  =  .55     - 550,000 rows , but the actual FF will be somewhat less than .55 and the estimated cardinality will be between 500,000 and 550,000  In fact, for this example it is .525   ½ + 1/20 – (1/2\*1/20)

**Matchcols or Filter Factors – Which is more important?**

Matchcols (start keys in LUW) are the number of leading columns of an index that are used to position within the index. This means there are predicates (usually equality predicates) on each of these columns. We have long been told that higher Matchcols is good for performance.  But is that always true?  Within a single index, more matching columns **is**better, but when DB2 has a choice of indexes, then it depends.  Consider this example:

A table has 2 indexes:

* IX1 on cola, colb, colc
* IX2 on cold

A query has this Where clause  – Where cola=? and colb=? and colc=? and cold=?

1. IX1 -> 3 matchcols    IX2 -> 1 matchcols
2. Which one is better?

It depends on the filter factors:

* The better index is the one that examines fewer entries and qualifies fewer rows
* Let’s say the table was the people in the world
* If IX1 were country, state, city and the predicates were
  + country=‘US’ and state=‘NY’ and city=‘New York’
  + 7 million rows would qualify
* If IX2 were SSN (social security number) and the predicate was
  + SSN = 123-04-0567
  + 1 row would qualify.
* IX2 is much better

**RUNSTATS**

The first and best step is to make sure that you have gathered as much relevant statistics (through Runstats) as possible.

1. Run Runstats on a regular basis.  It really is important to keep your statistics up to date.  Equally important is that the statistics of related objects are in synch.  If you create a new index, you should run Runstats on the table and all of its indexes, not just on the new index.  And, you should gather statistics on related tables if they are not recent.
2. Column cardinality statistics.   This is the default for LUW, but not for z/OS.  This is the key statistic for accurate filter factors.
3. Distribution statistics.  If a column has a non-uniform distribution and if you have a predicate on that column and the predicate uses a literal value, then column distribution statistics can be invaluable.
4. COLGROUP statistics.  If two columns are dependent on each other (such as city and state), then the filter factor should be very different than if they were independent.  The way for DB2 to know they are related is to gather colgroup statistics for the set of columns.

**Literals vs Host Variables (parameter markers)**

Generally speaking, using variables in your SQL statements (especially dynamic SQL) is a good technique.  For static SQL, it allows you to use one statement with different values passed in.  For dynamic SQL it fosters statement reuse.  DB2 has a cache of recently executed statements.  If the exact same dynamic statement is executed and found in the cache, DB2 does not have to do a prepare and determine the access path.  This can save substantial CPU time on each query.  However, if a column has a highly skewed distribution, then using a literal in the predicate can enable DB2 to use the distribution statistics (if you have collected them) and may produce a much better access path than it would for a uniform distribution (which DB2 has to assume if you use a variable).  Similarly, range predicates can benefit from using a literal in some cases.

A rule of thumb is to use variables most of the time, but use literals where it could make a difference.  You should always use variables for ID columns.  Otherwise every execution will be a different statement and there will be no reuse.

**REOPT**

* Problem – program uses variables for static SQL or parameter markers for dynamic SQL because different values are passed in. But, some columns have highly skewed data. Different values would benefit from different access paths. You don’t want to convert to dynamic SQL.

1. Solution – REOPT.  REOPT is a bind option with 3 choices:
   1. NONE – process as normal
   2. ALWAYS – determine the access path for each execution.
   3. ONCE – determine the access path on the first execution (the assumption here is that the first value is typical)

* Is this good? Well of course that depends
  + PLUS – customized access paths
  + MINUS – execution time overhead.  Access Path selection done for every execution if REOPT ALWAYS is used

**Statistical Views**

On DB2 LUW, there is a mechanism to gather statistics for more complicated relationships.  A statistical view is:

1. A view you create and collect statistics on
2. It is used to capture statistics for more complex relationships – such as joins, subqueries, expressions
3. The statistics are used by the Optimizer when you issue a query that has a similar structure and can be matched by the Optimizer
4. Create VIEW schema.v1 (cols…) AS ……
5. ALTER VIEW schema.v1 enable query optimization;
6. Runstats on view schema.v1 with distribution on all columns
7. Examples covering a number of situations can be found in this article -<http://www.ibm.com/developerworks/data/library/techarticle/dm-1305leverage/>
8. Statistical Views can be quirky and difficult for more complex cases with multiple query blocks.  You may need multiple stat views to cover subsets of the query

**Selectivity Overrides**

Sometimes it is impossible to provide DB2 with enough information through Runstats.  Range predicates with host variables are particular problems.  How big a range are you asking for?  DB2 has no way of knowing.  Other cases include:

1. Temp tables
2. Expressions and subqueries (DB2 for z/OS does not have Statistical Views, and even though LUW does, they can get complicated to use).
3. Joins – It is hard for DB2 to know how many rows will match between the outer and inner tables.  RI helps, and statistical views can sometimes provide enough information, but not always.  Each join key from the outer table may match a different number of inner rows.
4. Highly correlated columns, but colgroup stats have not been collected.

When the access path is not what you want (and of course, the access path that you want may not actually be a better access path), you can tell DB2 what access path to use (access path hint on z/OS or optimization guideline on LUW).  Alternatively (and more simply) you can tell DB2 what the filter factor of one or more predicates should be.  With this type of hint (or override), you are giving DB2 more information, but DB2 will still pick the access path.

DB2 for z/OS and DB2 LUW have implemented selectivity overrides in completely different fashions.  On z/OS, the overrides are done externally to the program and are put in the (fairly new) Access Path Repository.  On LUW, the override goes right in the program as part of the SQL statement.

**LUW**

To enable the use of the selectivity clause, a registry variable has to be set:

* DB2\_SELECTIVITY=ALL

The manuals actually only list values of YES or NO for this registry variable.  All is the same as YES, but enables the use of selectivity in more places.

The selectivity override is a clause in the statement.  It goes right after the predicate.

SELECT LASTNME, FIRSTNAME FROM EMPLOYEE

    WHERE WORKDEPT  = ?

DB2 doesn’t know the distribution for WORKDEPT = ?       Is it 10% or 90% of the rows?

SELECT LASTNME, FIRSTNAME FROM EMPLOYEE

    WHERE WORKDEPT  = ? SELECTIVITY .4

Now DB2 knows that it is 40% (actually DB2 has been told by the programmer that it is 40%)

The Selectivity clause cannot be used for all predicates.  For example, it can’t be used on a BETWEEN.  You need to convert a BETWEEN to  >= and <= predicates – adding Selectivity to each.

**z/OS  – The Access Path Repository**

The Access Path Repository for access path hints was introduced in V10.  Selectivity overrides via the Access Path Repository (APR) came in V11.

APR allows statement level hints based on matching the statement text.  This can be done at two levels:  Global – any statement in the system that matches; Package based – limited to a specific package.

There are 3 types of hints:

1. Access path hints (as was done via the plan\_table)
2. Optimization hints (bind parameters such as REOPT)
3. Selectivity overrides

This article will only be discussing the selectivity overrides.

Two sets of tables are used in the process.

1. SYSIBM tables
   1. SYSQUERY
   2. SYSQUERYPREDICATE
   3. SYSQUERYSEL

1. User tables to feed into the APR tables
   1. DSN\_USERQUERY\_TABLE
   2. DSN\_PREDICAT\_TABLE
   3. DSN\_PREDICATE\_SELECTIVITY

There are additional tables used for the other types of hints.

The process:

1. Populate yourschema.DSN\_USERQUERY\_TABLE with the statement text
2. Do an Explain of the query.  This  populates the tables DSN\_PREDICAT\_TABLE & DSN\_PREDICATE\_SELECTIVITY (with the filter factor that DB2 determined)
3. Update the selectivities of the predicates you want to change
   1. The set of selectivities is a selectivity instance. You can create additional selectivity instances with different overrides for various predicates by inserting additional rows. Each instance is given a weight representing the percentage that will occur at execution time. The full set of instances is called a selectivity profile
4. Issue BIND QUERY
   1. This will populate the SYSIBM tables (the APR) with the data from all queries in the DSN… tables. It is recommended to clear out these user tables of old entries before working on a new override to prevent inadvertently making an unintended change.
5. Do an Explain of your query (or bind your package with EXPLAIN(YES) to verify the selectivity overrides were used (i.e. a match for the statement was found) and to check the access path to see if you accomplished what you intended
6. Execute the query and monitor the performance to see if your new access path actually is better than what you had before

You can get the statement text from your source code, but IBM recommends getting it from SYSIBM.SYSPACKSTMT (for static SQL) or from the dynamic statement cache for dynamic SQL.

Creating multiple instances with different weights allows very detailed modeling of the occurrence rate of execution.  However, the optimizer can still only select one access path, regardless of how many different instances are specified.  This access path may be the best for 1 instance or another, or it may not be the best for any one combination of filter factors.  It may not be worth this much effort.

All the details can be found in

<http://www-01.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc.perf/src/tpc/db2z_influenceaccesspaths.dita>

**Access Path Repository or Selectivity Clause**

**Which Do You Prefer?**

1. APR Advantages:
   1. External to program. Can modify values without changing and redeploying the code
   2. Can affect all occurrences of the matching statement, or just those for a particular package
   3. Can be more detailed

1. Selectivity Clause Advantages:
   1. Internal to the program. Developer is aware of its existence
   2. If the statement changes, APR must be kept in synch – redo the query in DSN\_USERQUERY\_TABLE and BIND QUERY (will someone notice?).  Information hiding may be a good attribute for inter-module communication, but it is not a good attribute when the information is hidden from the developer.
   3. Much simpler process

**Limitations and Drawbacks to Selectivity Overrides**

Selectivity can be a moving target. As the table grows and the data distribution changes, the filter factors will change too.  If the table is growing, but the number of qualifying rows stays constant over time, the % is really going down.  For example, if data is not purged, so the table size keeps increasing, but the daily volume stays the same, the number of  “open” orders stays the same, but the percentage has changed.  The reverse can also happen – data **is** purged on a regular basis, but the business is growing and the number of open orders is increasing and may now be a higher percent of the total rows.

Join predicates are a little trickier to handle.  If you want to indicate that an outer row will match 1 row on average, then the selectivity will be 1/cardinality of the inner table  (this assumes you know which is the inner table).  The default filter factor is 1/max(colcard(t1.joincolcard),colcard(t2.joincolcard))

Any access path hint, whether you are just giving DB2 more information (selectivity) or telling DB2 what to do (access path hint) can actually hurt performance if not done with care and with performance testing.  You may think you are smarter than DB2, but you may not be.

**Q. What is predicate in sql.**

**Ans**

A predicate is a condition expression that evaluates to a boolean value, either true or false.

Predicates can be used as follows:

1. In a [SELECT](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_select) statement's [WHERE](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_where) clause or [HAVING](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_having) clause to determine which rows are relevant to a particular query. Note that not all predicates can be used in a **HAVING** clause.
2. In a [JOIN](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_join) operation’s **ON** clause to determine which rows are relevant to the join operation.
3. In an [UPDATE](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_update) or [DELETE](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_delete) statement's **WHERE** clause, to determine which rows are to be modified.
4. In a [WHERE CURRENT OF](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_wherecurrentof) statement's **AND** clause.
5. In a [CREATE TRIGGER](https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_createtrigger) statement's **WHEN** clause to determine when to apply triggered action code.

**Q. Explain SQL?**

**ANS.**

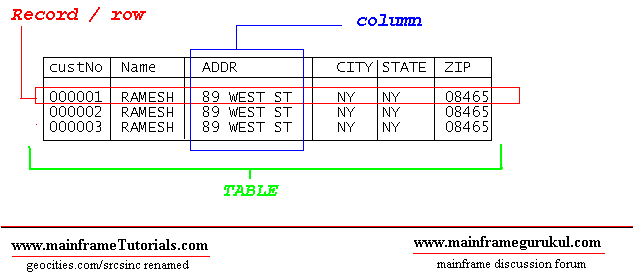
SQL (structured query language) is used to acccess, manipulate or control the

 access of relational database.  Databases like **DB2, ORACLE, SQL SERVER 2000**

 support the SQL (Structured query language).

In relational database data represented as tables. Tables contains the data

 in rows and column. Rows are called as records, columns called as fields.



**PRIMARY KEY**:  PRIMARY KEY is one column or combination columns.  This primary key is to identify the records uniquely in the table. Applications/programs normally

 provide PRIMARY KEY value to get the record values from the table. PRIMARY KEY

 columns have the unique index (unique indexes wont allow duplicate values) defined

 on them, so that processing time of SQL will be very less compare than the non-index columns.

**FOREIGN KEY:**  FOREIGN KEY is one or more columns in a table that refers to the

 primary key of another table. **Through this key we can establish a relation between**

**two tables**. If table B foreign key refers table A's primary key, DB2 / ORACLE wont

 allow to enter the records into table B if the foreign key value in the input data

 is not present in the table A's primary key list of values.

DB2 SQL (Structured query language) commands can be classified into 3 main categories

**DDL, DML and DCL.**

**DDL ( Data Definition Language ):** These statements usaully used by DBA's or Senior Programmers to create table spaces, tables , indexes etc...

**COMMANDS :**

 CREATE

DECLARE

 DROP

ALTER

**DML  (Data Manipulatoin Language )**: using these statements we can select, insert,  update

    and delete the data in the database.

**COMMANDS :**

 SELECT,

 INSERT,

 UPDATE,

 DELETE

**DCL  (Data Control Language )**: Using these statements DBA / user who has authority to

    grant can grant access or revoke the authority of accessing DB2 database.

**COMMANDS :**

GRANT

REVOKE

**Q. Explain DDL Statements in DB2?**

**ANS.**

Following are the **DDL ( Data Definition Language )** statements

**CREATE**

**DECLARE**

**ALTER**

**DROP**

**CREATE**

CREATE statement is used to create following DB2 database objects. The system catalog is updated whenever you Create a DB2 database object.

1. Buffer pools
2. Event monitors
3. Functions
4. Indexes
5. Schemas
6. Stored procedures
7. Tables
8. Table spaces
9. Triggers
10. Views

**EG-**

CREATE TABLE employee(

Emp\_id smallint not null,

Emp\_first\_name varchar(30),

Emp\_m\_name varchar(1),

EMP-LAST\_NAME VARCHAR(30),

SALARY DECIMAL(100,2),

DEP smallint )

**DECLARE**

The DECLARE statement is similar to the CREATE statement, except that it is used to

 create temporary tables that exist only for the duration of a database connection.

 Temporary tables are useful when you are working with intermediate results. Declared

 tables can be referenced like any other table, and they can be altered or dropped like

 any other table. A table is the only object that can be declared. The system catalog

 is not updated when you declare a temporary table. You can declare a temporary table

 by using the **DECLARE GLOBAL TEMPORARY TABLE** statement.

**DECLARE GLOBAL TEMPORARY TABLE session.emp1**

**LIKE employee**

**ON COMMIT PRESERVE ROWS**

**NOT LOGGED**

**IN tempspace**

 In this example, the DECLARE GLOBAL TEMPORARY TABLE statement is used to

 declare a temporary table named emp1, located in an existing user temporary table

 space named TEMPSPACE. The columns in this table will have the same names and

 definitions as the columns in the EMPLOYEE table. The rows of the temporary table

 will be preserved (not deleted) whenever a COMMIT statement is processed.

 Finally, changes to the temporary table are not logged (this is the only option).

 session is a schema name.

**ALTER**

The ALTER statement is used to change some of characterstics of following DB2 objects.

1. Buffer pools
2. Tables
3. Table spaces
4. Views

 Example - In the following example, we can alter the table defined.

         ALTER TABLE EMPLOYEE

ADD DATE\_OF\_JOIN   DATE

**Note:** you cannot alter an index. You must drop it and then create a new one with a different definition

**DROP**

In DB2, You can drop any db2 objects created using CREATE or DECLARE statement

 You can drop any of the following db2 objects

1. Buffer pools
2. Event monitors
3. Functions
4. Indexes
5. Schemas
6. Stored procedures
7. Tables
8. Table spaces
9. Triggers
10. Views

The DROP statement will delete the object definition of catalogue. Following is an

**Ex:**

**DROP TABLE EMPLOYEE**

**Q. Explain DML Statements in DB2?**

**ANS.**

You can use following statements to manipulate data on db2 database.

1. SELECT
2. INSERT
3. UPDATE
4. DELETE

**SELECT**

  SELECT statement is used to get all data or required data  from the table.

**Simple Syntax**

**SELECT  < Column Names > / \* / function(< Column Name >) FROM < TABLE NAME >**

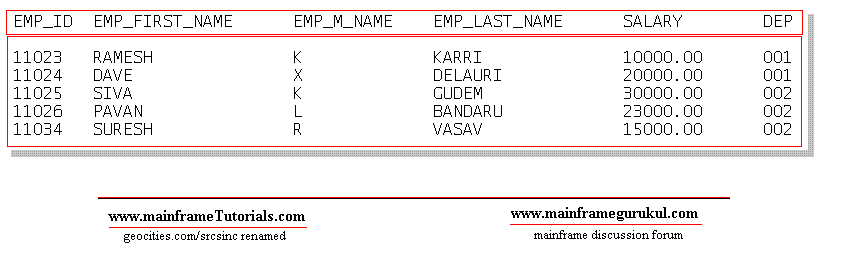
**[ WHERE < Condition > .... ]**

  Example (s)

  1. To get all data from EMPLOYEE table , issue

  SELECT \* FROM EMPLOYEE

  Following is a the result set returned by the query



**INSERT**

  INSERT statement is used to insert record(s) into table / view. Inserting a row into

  view also inserts the row into table.

**Simple Syntax**

**INSERT INTO < TABLE NAME / VIEW NAME > [ (<COLUMN NAME1>,<COLUMN NAME2>...) ]**

**VALUES ( value1, value2,.... )**

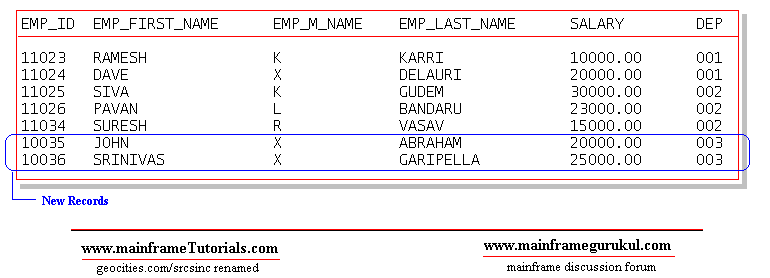
**Example 1:**

**INSERT INTO EMPLOYEE ( EMP\_ID, EMP\_FIRST\_NAME,EMP\_M\_NAME,EMP\_LAST\_NAME, SALARY, DEP)**

**VALUES   ( 10035, 'JOHN', 'X', 'ABRAHAM', 20000.00, 003 ) ,**

**( 10036, 'SRINIVAS', 'X', 'GARIPELLA', 25000.00, 003 )**

Above insert statemet, insert two records into EMPLOYEE Table, Now table contains following records.



**Example2.**

 Specify a fullselect to identify data that is to be copied from other tables or views. A

  fullselect is a statement that generates a result table. For example:

**CREATE TABLE emp LIKE EMPLOYEE**

**INSERT INTO emp**

**SELECT EMP\_ID,EMP\_FIRST\_NAME,EMP\_M\_NAME,EMP\_LAST\_NAME,SALARY, DEP**

**FROM EMPLOYEE**

**WHERE DEP = 002**

**Q.DB2 compile program**

ANS.

//DB2COMP (XXX,XXX),'COMPILE JCL',

//             CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* COMPILATION, LINK EDIT AND THE BIND STEP FOR A COBOL DB2 PROGRAM \*

//\* WILL BE DONE BY SUBMITTING THIS JOB.                             \*

//\* THE DB2 REGIONS AND CORRESPONDING PARAMETERS NEEDS TO BE CHANGED \*

//\* WITH RESPECT TO THE PROGRAM                                      \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*                 PRECOMPILE DB2 PROGRAM                           \*

//\*--------------  LOCATION OF DBRM LIBRARY -------------------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//PC       EXEC PGM=**DSNHPC**,

//             PARM='HOST(COB2),APOST,SOURCE',

//             REGION=4096K

//**DBRMLIB**  DD  DISP=SHR,

//             DSN=**DEV.SURESH.DBRM(DB2PROG)**<------------------------ (1)

//STEPLIB  DD  DISP=SHR,

//             DSN=SYSX.DB2.XXX.XXXXX

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*SYSIN -----------INPUT COBOL DB2 PROGRAM LOCATION-----------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSIN** DD  DISP=SHR,

//             DSN=**DEV.SURESH.SRC(DB2PROG)**<----------------------  (2)

//SYSCIN   DD  DISP=(MOD,PASS),

//             DSN=&&TEMP,

//             SPACE=(800,(500,500)),

//             UNIT=SYSDA

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*                          DCLGEN MEMBER LOCATION                  \*

//\*SYSLIB-----------------INPUT SOURCE LIBRARY FOR SQL---------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSLIB** DD  DISP=SHR,

//             DSN=**DEV.SURESH.DCL**<---------------------- (3)

//         DD  DISP=SHR,

//             DSN=**DEV.SURESH.CPY**

//SYSPRINT DD  SYSOUT=T

//SYSTERM  DD  SYSOUT=T

//SYSUDUMP DD  SYSOUT=\*

//SYSUT1   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT2   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*                         COMPILATION                              \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//COB      EXEC PGM=**IGYCRCTL**,

//             COND=(4,LT,PC),

//             PARM=('SIZE(4000K),BUFSIZE(32760),LIST,LIB,MAP,OBJECT',

//             'DATA(31),XREF,RENT'),

//             REGION=4M

//STEPLIB  DD  DISP=SHR,

//             DSN=XXXX.XXXXXX

//SYSIN    DD  DISP=(OLD,DELETE),

//             DSN=&&TEMP

//SYSLIN   DD  DISP=(MOD,PASS),

//             DSN=&&LOADTMP,

//             SPACE=(800,(500,500)),

//             UNIT=SYSDA

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*--------------SOURCE LIBRARIES FOR COBOL DB2 CODE (COPY LIBRARIES)\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSLIB** DD  DISP=SHR,

//             DSN=**DEV.SURESH.DCL**<-----------------  (4)

//         DD  DSN=**DEV.SURESH.CPY**,DISP=SHR

//SYSPRINT DD  SYSOUT=\*

//SYSUDUMP DD  SYSOUT=\*

//SYSUT1   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT2   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT3   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT4   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT5   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT6   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//SYSUT7   DD  SPACE=(800,(500,500),,,ROUND),

//             UNIT=SYSDA

//\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*                         LINK EDIT                                \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//LKED     EXEC PGM=**IEWL**,

//             COND=((4,LT,COB),(4,LT,PC)),

//             PARM='XREF'

//SYSLIB   DD  DISP=SHR,

//             DSN=SXXX.SXXXXXXX

//         DD  DISP=SHR,

//             DSN=XXXX.DB2.XXX.XXXXLOAD

//         DD  DISP=SHR,

//             DSN=SYS1.VSCLLIB

//SYSLIN   DD  DISP=(OLD,DELETE),

//             DSN=&&LOADTMP

//\*        DD  DDNAME=SYSIN

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*----------------LOCATION OF LOAD LIBRARY--------------------------\*

//SYSLMOD  DD  DISP=SHR,

//             DSN=**DEV.SURESH.LOADLIB(DB2PROG)**<------------  (5)

//SYSPRINT DD  SYSOUT=\*

//SYSUDUMP DD  SYSOUT=\*

//SYSUT1   DD  SPACE=(1024,(50,50)),

//             UNIT=SYSDA

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*               BIND - BIND THE DB2 PACKAGE                        \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//BIND     EXEC PGM=**IKJEFT01**,

//             COND=(4,LT),

//             REGION=4096K

//STEPLIB  DD  DISP=SHR,

//             DSN=XXX4.DB2.XXXX.XXXXLOAD

//**DBRMLIB**  DD  DISP=SHR,

//             DSN=**DEV.SURESH.DBRM(DB2PROG)**<---------------  (6)

//SYSPRINT DD  SYSOUT=\*

//SYSTSPRT DD  SYSOUT=\*

//SYSUDUMP DD  SYSOUT=\*

//SYSTSIN  DD  \*

**DSN SYSTEM (DEVDB )**

**BIND  MEMBER    (DB2PROG) -**

**PACKAGE   (PACKG11) -**

**LIBRARY   ('DEV.SURESH.DBRM') -** <---------------- (7)

**ACTION    (REP) -**

**ISOLATION (CS) -**

**VALIDATE  (BIND)-**

**RELEASE   (COMMIT) -**

**OWNER     (SURESH) -**

**QUALIFIER (DEVQUALI)**

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bottom of Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(1)  -  When we precompiled, precompiler will create the DBRM, it will

be placed in the pds specified here.

(2)  -  Location of COBOL-DB2 program

(3)  -  Needs to speficiy DCLGEN member locations

(4)  -  Needs to specify DCLGEN and COPYBOOK locations here

(5)  -  Load module location, load module will be created here. this

location needs to be given in run jcl.

(6) & (7) - specify the location of DBRM, ( same location used in step1 ).

**Q. Extract data from DB2 table directly using IKJEFT01 SQL JCL**

**ANS.**

//STEP002  EXEC PGM=**IKJEFT01**,

//             **DYNAMNBR**=20,COND=(0,NE)

//**STEPLIB**  DD  DISP=SHR,

//             DSN=SYS.DB2.XXX.RUNXX.LOAD

//         DD  DISP=SHR,

//             DSN=SYS.DB2.XXX.SDSNEXIT

//         DD  DISP=SHR,

//             DSN=SYS.DB2.XXX.SDSNLOAD

//**SYSIN**DD  \*

**SELECT  \***

**FROM EMP\_TABLE**

**WITH UR;**

//\*

//**SYSTSIN** DD  \*

    DSN SYSTEM(XXX) RETRY(20)

    RUN PROGRAM(**DSNTIAUL**) PLAN(**DSNTIAUL**) PARMS(**'SQL'**) -

       LIB ('SYS.DB2.XXX.RUNXX.LOAD')

    END

/\*

//\*\*\*UNLOAD FILE

//**SYSREC00** DD  DISP=(NEW,CATLG,DELETE),

//             DSN=DEV.UNLOAD.FILE,

//             UNIT=SYSDA,SPACE=(CYL,(50,50),RLSE)

//\*

//\*\*\*TABLE STRUCTURE

//SYSPUNCH DD  DUMMY

//\*

//SYSTSPRT DD  SYSOUT=\*

//SYSPRINT DD  SYSOUT=\*

//SYSUDUMP DD  SYSOUT

**Q.Binding DB2 with IKJEFT01**

**ANS.**

//\*  DB2 BIND JCL

//\*

//**BIND**     EXEC PGM=IKJEFT01,

//             COND=(4,LT),

//             REGION=4096K

//**STEPLIB**  DD  DISP=SHR,

//             DSN=XXX4.DB2.XXXX.XXXXLOAD

//**DBRMLIB**  DD  DISP=SHR,

//             DSN=DEV.SURESH.DBRM(DB2PROG)    <---------------  (1)

//SYSPRINT DD  SYSOUT=\*

//SYSTSPRT DD  SYSOUT=\*

//SYSUDUMP DD  SYSOUT=\*

//**SYSTSIN**  DD  \*

**DSN SYSTEM (DEVDB)**

BIND  MEMBER    (DB2PROG) -

      PACKAGE   (PACKG11) -

      LIBRARY   ('DEV.SURESH.DBRM') -         <---------------- (2)

      ACTION    (REP) -

      ISOLATION (CS) -

VALIDATE  (BIND)-

      RELEASE   (COMMIT) -

      OWNER     (SURESH) -

      QUALIFIER (DEVQUALI)

END

/\*

**Q. Running Cobol DB2 program using IKJEFT01**

**ANS.**

//\*\* COBOL DB2 RUN JCL

//STEP01 EXEC PGM=**IKJEFT01**,

//         DYNAMNBR=20

//\*

//**STEPLIB**  DD  DISP=SHR,

//             DSN=SYSL.DB2.DEV.XXXXLOAD

//         DD  DISP=SHR,

//             DSN=SYSL.DB2.DEV.XXXXYYYY

//\*

//**SYSTSPRT** DD  SYSOUT=\*

//SYSTSIN  DD  \*

DSN SYSTEM(DEVDB)

  RUN PROGRAM(DB2PROG)              -

PLAN(PLAN11111)                      -

LIBRARY('DEV.SURESH.LOADLIB')

      END

/\*

//SYSOUT   DD  SYSOUT=\*

//SYSIN    DD  \*

//\*

//\* COBOL DB2 RUN JCL ENDS

**Q. Writing a COBOL DB2 Program.**

**ANS.**Let us assume we are writing a cobol program to read EMPLOYEE

table and get the details of employee with the name XXXXXXX.

Let us go in step wise..

create the following table in db2 or assume it is there in db2 database.

**EMPLYEE**

**EMPID   EMPNAME     DEPARTMENT   SALARY   DESIGNATION**

1000         XXXXXXX           XX                 10000                 SE

1001         YYYYYYY           YY                      9000                   SE

1002         ZZZZZZZ              ZZ                       20000                 MA

**STEP 1.**  We need to declare the table structure in the

**WORKING-STORAGE SECTION or LINKAGE SECTION.**

**EXEC SQL**

               DECLARE DSNXXX.EMPLOYEE

( EMPID          CHAR(10)   NOT NULL,

                         EMPNAME        CHAR(30)   NOT NULL,

                         DEPARTMENT     CHAR(2)    NOT NULL,

                         SALARY         DECIMAL(10,2)   NOT NULL,

                         DESIGNATION    CHAR(4)    NOT NULL  )

**END-EXEC.**

we can use DB2 tool called DCLGEN to generate this declaration

for us and can include that copy book here.

if you create a copybook using DCLGEN. Use following sntax to include

**EXEC SQL**

**INCLUDE<copybookname>**

**END-EXEC.**

**STEP 2.**   Declare host variables in WORKING-STORAGE SECTION.

          HOST VARIABLES - A host variable is a data item declared in cobol to use

it in embedded SQL.

          For EMPLOYEE table, host variable declaration is look like as follows...

01  EMPOYEE-RECORD.

05  HV-EMPID PIC X(10).

05  HV-EMPNAME PIC X(30).

05  HV-DEPARTMENT       PIC X(2).

**05  HV-SALARY           PIC S9(8)V99 COMP-3.** (SALARY- DECIMAL(10,2))

05  HV-DESIGNATION      PIC CHAR(4).

         If you use db2 tool DCLGEN, it will automatically creates this structure also

along with table declaration specified in step1.

**STEP 3.**  Include SQLCA as follows in WORKING-STORAGE SECTION.

EXEC SQL

         INCLUDE SQLCA

END-EXEC.

SQLCA  - SQL communication area.

                  When a SQL statement executes, DB2 places a value in SQLCODE AND SQLSTATE host variables or any other fields of SQLCA.  based on the values in these variables we can know whether sql ran sucessfully or not.

                  SQLCA contains a declartion of fields like SQLCODE,SQLSTATE and

SQLERRD etc....

STEP 4.  Add a SQL statement in procedure division to get the details of employee

with the name XXXXXXX.

         DISPLAY ' PROGRAM STARTED .... '

         .........

         EXEC SQL

              SELECT   SALARY

**INTO  :HV-SALARY**

              FROM EMPLOYEE

              WHERE EMPNAME = 'XXXXXXX'

END-EXEC.

         IF SQLCODE = 0

              DISPLAY ' SQL EXECUTED SUCESSFULLY '

              DISPLAY ' EMPLOYEE SALARY IS       '   HV-SALARY

         ELSE

              DISPLAY ' SQL FAILED '

              DIAPLY  ' SQL CODE   '  SQLCODE

         END-IF.

          ....

          ....

         DISPLAY ' PROGRAM ENDED'.

Here SQLCODE = 0 means, sql ran sucessfully without any issues. Hence

we are displaying the HV-SALARY into the spool.

If SQLCODE NOT = 0, there is issue in executing the sql statement.

Now we have compeleted coding a cobol-db2 program. our next step is to

compile the program.

**Q. Using  RunJCl for DB2**

**ANS.**

**Q.Joins**

**ANS.** http://www.sql-tutorial.net/SQL-JOIN.asp

Customers:

CustomerIDFirstNameLastName Email DOB Phone

1 John Smith John.Smith@yahoo.com 2/4/1968 626 222-2222

2 Steven Goldfish goldfish@fishhere.net 4/4/1974 323 455-4545

3 Paula Brown pb@herowndomain.org 5/24/1978 416 323-3232

4 James Smith jim@supergig.co.uk 20/10/1980 416 323-8888

Sales:

**CustomerIDDateSaleAmount**

2 5/6/2004 $100.22

1 5/7/2004 $99.95

3 5/7/2004 $122.95

3 5/13/2004 $100.00

4 5/22/2004 $555.55

There are 2 types of SQL JOINS – INNER JOINS and OUTER JOINS. If you don't put INNER or OUTER keywords in front of the SQL JOIN keyword, then INNER JOIN is used. In short "INNER JOIN" = "JOIN" (note that different databases have different syntax for their JOIN clauses).

If the Sales table has the following rows:

Sales:

CustomerID Date SaleAmount

2 5/6/2004 $100.22

1 5/6/2004 $99.95

The INNER JOIN will select all rows from both tables as long as there is a match between the columns we are matching on. In case we have a customer in the Customers table, which still hasn't made any orders (there are no entries for this customer in the Sales table), this customer will not be listed in the result of our SQL query above.

Even though Paula and James are listed as customers in the Customers table they won't be displayed because they haven't purchased anything yet.

But what if you want to display all the customers and their sales, no matter if they have ordered something or not? We’ll do that with the help of SQL OUTER JOIN clause.

The second type of SQL JOIN is called SQL OUTER JOIN and it has 2 sub-types called LEFT OUTER JOIN and RIGHT OUTER JOIN.

As you can see we have selected everything from the Customers (first table). For all rows from Customers, which don’t have a match in the Sales (second table), the SalesPerCustomer column has amount NULL (NULL means a column contains nothing).

The RIGHT OUTER JOIN or just RIGHT JOIN behaves exactly as SQL LEFT JOIN, except that it returns all rows from the second table (the right table in our SQL JOIN statement).

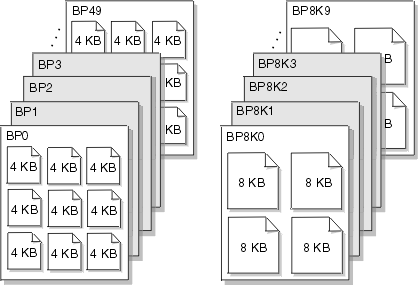
**Q. Explain DB2 buffers?**

ANS.*Buffer pools* are a key element of DB2® performance, and help you to avoid delays when retrieving data.

DB2 can retrieve a page from a buffer pool faster than it can from disk. When data is already in a buffer, an application program avoids the delay of waiting for DB2 to retrieve the data from disk.

DB2 lets you use up to 50 buffer pools that contain 4 KB pages and up to 10 buffer pools each that contain 8 KB, 16 KB, and 32 KB pages.

Figure 1. Buffer pools with 4 KB and 8 KB pages



At any time, pages in a virtual buffer pool can be in use, updated, or available.

* In-use pages are currently being read or updated. The data that they contain is available for use by other applications.
* Updated pages contain data that has changed but is not yet written to disk.
* Available pages are ready for use. An incoming page of new data can overwrite available pages.

To avoid disk I/O, you can use updated and available pages that contain data.

When data in the buffer changes, that data must eventually be written back to disk. Because DB2 does not need to write the data to disk right away, the data can remain in the buffer pool for other uses. The data remains in the buffer until DB2 decides to use the space for another page. Until that time, applications can read or change the data without a disk I/O operation.

The key factor that affects the performance of buffer pools is their size. The method that DB2 uses to access buffer pools also affects performance.

**Buffer pool size**

The size of buffer pools is critical to the performance characteristics of an application or a group of applications that access data in those buffer pools.

Tuning your buffer pools can improve the response time and throughput for your applications and provide optimum resource utilization. For example, **applications that do online transaction processing are more likely to need large buffer pools because they often need to re access data. In that case, storing large amounts of data in a buffer pool enables applications to access data more efficiently.**

By making buffer pools as large as possible, you can achieve the following benefits:

* Fewer I/O operations result, which means faster access to your data.
* I/O contention is reduced for the most frequently used tables and indexes.
* Sort speed is increased because of the reduction in I/O contention for work files.

You can use the ALTER BUFFERPOOL command to change the size and other characteristics of a buffer pool at any time while DB2 is running. Use the DISPLAY BUFFERPOOL and ALTER BUFFERPOOL commands to gather buffer pool information and change buffer pool sizes.

**Q. Expalain DCLGEN**

**ANS.**DCLGEN stands for **DeCLarationGENerator.**It is an IBM provided function which generates INCLUDE members for DB2 tables for use in COBOL and PL/1 programs. These INCLUDE members contain [SQL](http://www.geekinterview.com/question_details/11191) table declarations and working storage structures.

In COBOL DB2 programs the host variables are needed for the column names of the tables. This can be done by a DB2 utility called DCLGEN Declaration Generator. **This is primarily a copybook of host variables and also contains table structure and helps in programming reference.** We can code this manually but using DCLGEN we can eliminate the mistakes by replacing all underscores in column names into (hyphens).

DCLGEN is a declaration generator used to generate DB2 equivalent COBOL variables called Host variables.

Which can be used to generate host variables with equivalent data types of DB2 columns.

DB2 Table  <>  DCLGEN <> COBOL

**Host variables:**

1)  Can be used to pass the data from cobol program to db2 table or DB2 table to COBOL

2)  When host variables are coded with sql statements it must be prefixed with : like :hv-cname.

3) Table name must be supplied as input to DCLGEN & partition dataset should be as output.

4) After creating DCLGEN variables which must be copied to application program in WORKING-STORAGE SECTION by using include command i.e.

Exec SQL

INLCUDE DCLGEN

End-EXEC.

5) INCLUDE & COPY have the same functionality

**Q. What is REORG in DB2?**

**ANS:** REORG is used to help DB2 point to accurate data (ie, indexes should become aware of fresh data and no longer include deleted data), as well as "collapse" empty page space created by deletion of data and/or indexes. It can also help move data that is related closer to each other for more efficient access (especially true in the case of a cluster index).

RUNSTATS is used to help gather updated statistics on the volume and/or distribution of data within tables and indexes. This information is stored in the system tables and is used by many things including the optimizer to determine the optimal path to query the data. This is especially true as data grows. The data may distribute in a way that causes the optimizer to now include an index that it didn't before in its access path (or perhaps vice versa and choose not to use an index). RUNSTATS can also affect how a REORG runs - at least in older versions of DB2.

Based on what I have read from DB2 DBA's you generally want to run

1. RUNSTATS
2. REORG
3. RUNSTATS

The first RUNSTATS helps the REORG to work appropriately and efficiently. The RUNSTATS after is more to make sure that stats are now accurate given all the data movement around on pages. Depending on where you read you may see DBA's mention the last RUNSTATS is no longer needed with newer versions of DB2. Since we haven't seen definitive answers on that and since most DB2 DBA's I've heard of follow the above order, our company has chosen to use that order (for the record we are on DB2 LUW 9.7 FP4 and we plan on migrating to V 10.1 sometime next year).

Also, just for the sake of completeness, generally when you update the statistics and you affect the optimizer you want all applications that call into DB2 to make sure they are taking advantage of those new optimizations. **So a REBIND of packages will make sure that static packages know of the updated statistics and optimized query paths**, and a FLUSH PACKAGE CACHE DYNAMIC cleans out dynamic queries from things like Hibernate, so that way the queries will be rebound with the updated query paths. (**NOTE**: on z/OS I see you have BIND and REBIND options. Perhaps the FLUSH PACKAGE CACHE DYNAMIC is under those or has a different name on z.)

So ultimately I would do the following:

1. RUNSTATS
2. REORG
3. RUNSTATS
4. REBIND packages
5. FLUSH PACKAGE CACHE DYNAMIC (on LUW) or whatever the equivalent is on z/OS.

**Q. What is clustered index in DB2?**

**ANS: In other words, a clustered index stores the actual data, where a non-clustered index is a pointer to the data**. In most DBMSs, you can only have one clustered index per table, though there are systems that support multiple clusters (DB2 being an example).   
  
Like a regular index that is stored unsorted in a database table, a clustered index can be a composite index, such as a concatenation of first name and last name in a table of personal information.

Oracle likes heaps, but SQL Server likes clusters, how do you design for both? I’d say you can’t.

**Q.What is cluster ratio in db2**

**Q.What is clustering in db2**

**Q.What is member Cluster in db2**

**Q. What is error code for end of record/record not found in DB2?**

A. 100

**Q. Then How you will find that, Error is for end of record or record not found?**

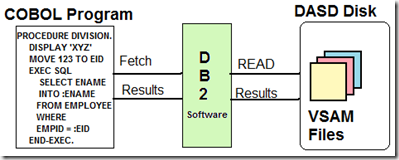
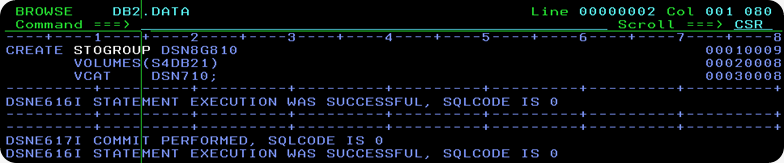
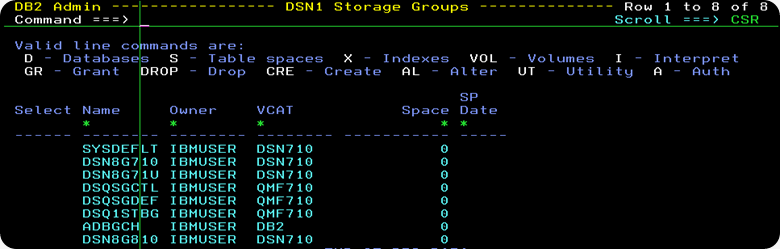
A. If we are using select statement, sql error code 100 represents record not found

   If we are using cursor , in that situation, sql error code 100 represents end of records

**Q. What is DB2? How does it compare with VSAM Files?**

DB2 is the Data-Management System from IBM. DB2 is a newer technology, whilst VSAM is old. VSAM (which stands for Virtual Sequential Access Method) is just a Data Access-Method. You can browse a VSAM Dataset in Sequential Mode – Record-by-record. You can do random access – jump (hop), fly directly to a particular Record. Skip-Sequential is like the best of both worlds – Jump to 1st record, and then start browsing the subsequent records one-by-one. Sounds cool, isn’t it, but hold on and have a look at what DB2 Software offers!   
  
DB2 is a complete Software Package that allows you to create, store, update and retrieve Data in a Systematic Manner.   
  
1. Data Abstraction: It presents a Logical, Structured view of the Data to the User. The user doesn’t need to worry about the details of how the Data is physically stored on the DASD-Disk.   
2. Data Independence: DB2 provides a layer of Independence, between the application or COBOL Program and the Data. Every-time the structure of the Data changes (say you add a new field to the EMPLOYEE File), you don’t need re-write the entire COBOL Source code again.   
3. Data Security: DB2 Software controls the Access to the Data. Not everybody can see all the Data. You can see it, only if you've got all the necessary privileges.   
4. Transaction Management and Concurrency: DB2 manages all the Transactions, and ensures the Data remains consistent. It provides a Locking-Mechanism, for concurrent access to the Data.   
5. Data Recovery: When there are Crashes, DB2 allows you to recover swiftly, and restore the original Data.

**Q. What are the DB2 Data Structures?**

DB2 Software physically stores the Data in VSAM Files on the DASD Disk.   
  
    
STOGROUP: The Set of DASD Volumes that will be used to house the VSAM Files containing the Data. In the example below, I've created a STOGROUP called DSN8G810 with the DASD Volume S4DB21.   
  
  
  
**The information about all the STOGROUP’s (Storage Groups) is stored in the DB2 Catalogue Table SYSIBM.SYSSTOGROUP**. The snapshot of all the STOGROUP's is shown in the picture below.   
  


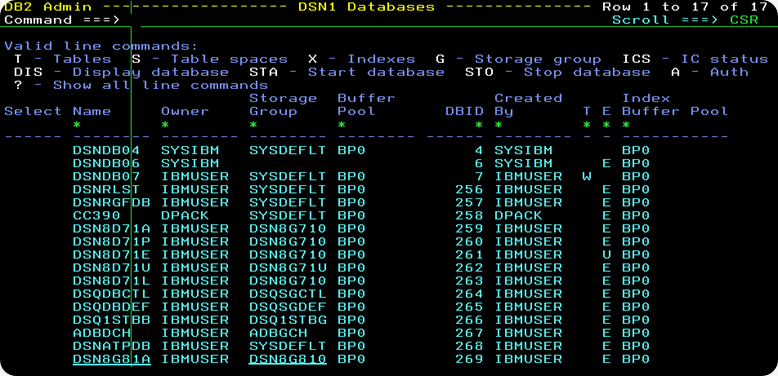
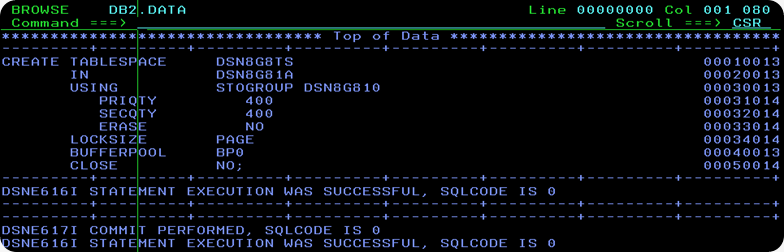
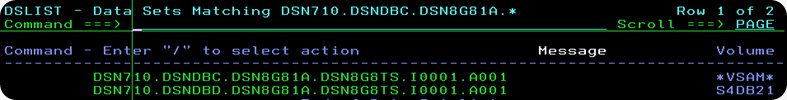
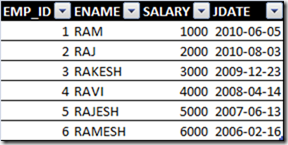
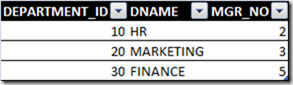
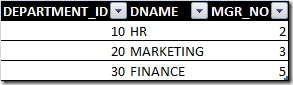
Database: A Database is a Logical-Grouping of Database Objects, related together by application, or subsystem. A Database contains no Data of its own, but it merely groups like DB2 Objects. To create a New Database, use the CREATE DATABASE Statement. In the example below, I have created a New Database called DSN8G81A on the STOGROUP DSN8G810.   
  
  
  
**The information on all the Databases in DB2 System is stored in the DB2 Catalog Table SYSIBM.SYSDATABASE.** The below picture is a snapshot of it.   
  


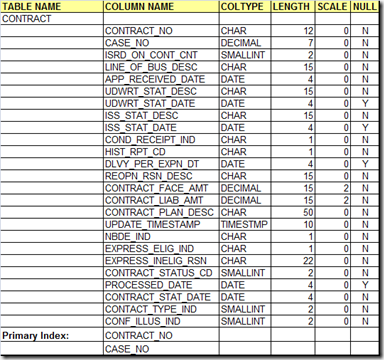
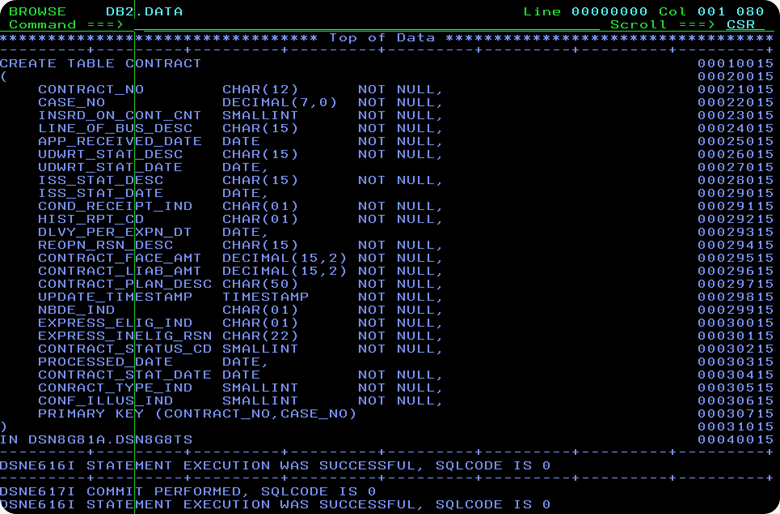
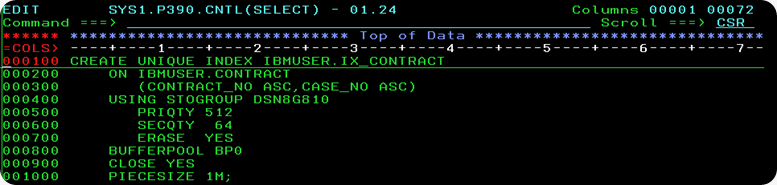
Table-SPACE: A Table-Space is a VSAM File which stores or houses the DB2 Table-Data Physically. You can create a Brand-New Table-space File, by using the CREATE TABLESPACE Statement.   
  
  
  
In the picture above, I have created a new Table-space File called DSN8G8TS, which has a Primary-Space of 400KB and Secondary Space of 400KB. Just like the contents in a Book, the data in a Table-Space File is stored in Fixed-Size Pages. DSN8G8TS Table-Space is created under the DSN8G81A Database Umbrella. The newTable-Space File will be stored on DSN8G810 Storage-Group. A buffer-pool is the place in Main Storage, where DB2 will store Pages fetched from the Table-Space.   
  
  
  
The information about all Table-Space Files is stored in the DB2 Catalogue Table SYSIBM.SYSTABLESPACE. A snapshot of this is shown in the above picture.

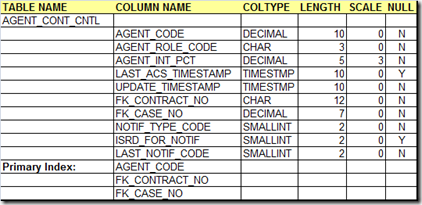
Once the Table-Space File has been newly created, you can actually see it using DSLIST Utility in ISPF Menu 3.4. The VSAM Dataset-name has the format vcat.DSNDBC.dbname.tsname.\*.   
  


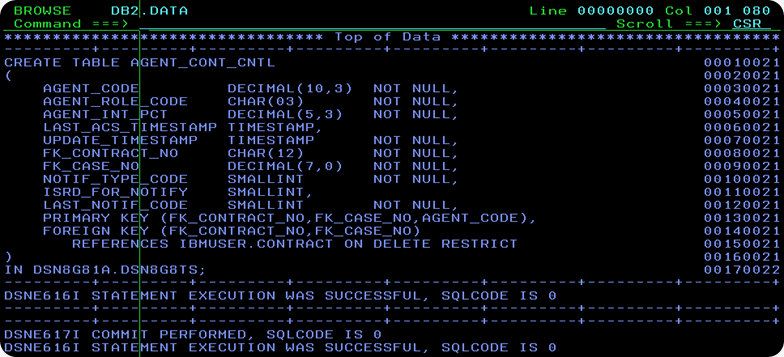
**Q. What are Tables, Primary Keys and Foreign Keys?**

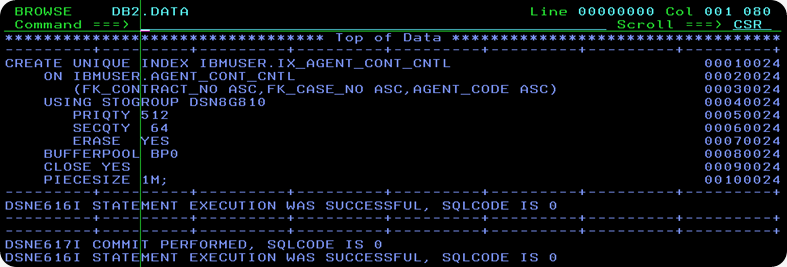
A Table stores Data. Tables look like a spreadsheet, they consist of Rows and Columns, as shown in the picture below.    
  
        
The Employee-ID Number is unique for each Employee. No two Employees have the same Employee-ID in the Company. Similarly, the Department-ID is unique for each Department. Employee-ID Column is said to be the Primary-Key or   
Primary-Index in the EMPLOYEE Table. Department-ID is said to be the Primary-Key or Primary-Index in the DEPARTMENT Table.   
  
Suppose you want to establish a Relationship, and store the information about, which Employee works for which Department? The EMPLOYEE Table should be expanded and an additional column is added to it, called FK\_DEPT\_ID. EMPLOYEE.FK\_DEPT\_ID Column refers to DEPARTMENT\_ID Primary-key column of the DEPARTMENT Table. FK\_DEPT\_ID is called Foreign-Key. Here, is how it works. RAJ has FK\_DEPT\_ID=10. Look-up Department-No 10 in the DEPARTMENT Table, and it is the HR Department. It means RAM works in HR Department.   
  
      

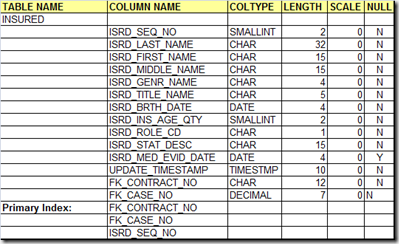
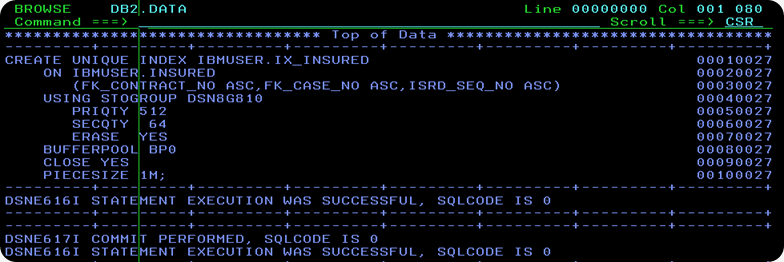
**Q. How do you write the DDL to create Tables and Index Key-Columns?**

As an example, let us build a toy Database-System, called New Business Tracking (NBT), which stores the information about new Insurance Policies (New Business) of an Agent for the first 90 days. This Database System will have three DB2 Tables – CONTRACT, AGENT\_CONT\_CNTL and INSURED.    
  
CONTRACT Table Schema  
The CONTRACT Table stores the basic Details of the Contract, what is the Contract-No, how many Insured's does this Contract cover (ISRD\_ON\_CONT\_CNT), is it a Life or Annuity Policy (LINE\_OF\_BUS\_DESC), when the Application was received, when was the Policy under-written, when the Policy got Issued, what is the Face Amount (Sum-Assured) and many other details.   
   
  
DDL For CONTRACT Table  
You write the CREATE TABLE Statement listing all the Columns followed by their SQL-types, along with whether the Column accepts NULL Values or is it NOT NULL. The List of Columns is a Comma-Separated List.   
  
Primary Index For CONTRACT Table   
Here's, how I have created the Primary-Index on the CONTRACT\_NO, CASE\_NO Columns of the CONTRACT Table. I have created a UNIQUE INDEX on the CONTRACT Table.   
    


AGENT\_CONT\_CNTL Table Schema   
AGENT\_CONT\_CNTL Table stores the basic data about the Insurance Agent, and the Insurance Policies sold by him.   
 

DDL for AGENT\_CONT\_CNTL Table  
The AGENT\_CONT\_CNTL Table stores Agent Details. How do you know, which Policy belongs to which Agent-ID? For each Agent, the Policy-No sold by him is stored in FK\_CONTRACT\_NO Column, and this refers to the CONTRACT\_NO Column in CONTRACT Table. Similarly, FK\_CASE\_NO refers to the CASE\_NO in the CONTRACT Table. Thus, the foreign-key (FK\_CONTRACT\_NO, FK\_CASE\_NO) refers to the   
Primary-Key (CONTRACT\_NO, CASE\_NO) in the CONTRACT Table. One needs to understand, that you cannot create a Primary-Key without defining a UNIQUE INDEX.  
  


Primary Index For AGENT\_CONT\_CNTL Table   
I have created a New Unique-Index IX\_AGENT\_CONT\_CNTL on the columns FK\_CONTRACT\_NO, FK\_CASE\_NO and AGENT\_CODE of the AGENT\_CONT\_CNTL Table.   
    


INSURED Table Schema  
INSURED Table stores information about the Insured, or Policy-Holder. In case of a Join-Policy, a Contract may have two or three Insured's. In that case, each insured for that Policy has a different ISRD\_SEQ\_NO. The Insured's Name, Date-of-Birth etc. is stored in this Table.   
      
  
DDL for INSURED Table   
the CREATE TABLE Statement for the INSURED Table is shown in the picture below. How do you know, who is the Policy-Holder for a particular Contract-No? This information is stored in the Columns FK\_CONTRACT\_NO and FK\_CASE\_NO, which is the Foreign-Key, and refers to the corresponding columns in the CONTRACT Table.   
  
  
Primary Index For INSURED Table   
I have created a Primary Unique-Index on the ISRD\_SEQ\_NO,FK\_CONTRACT\_NO and FK\_CASE\_NO Columns of the INSURED Table.   
  


**DB2 SQL Error Codes**

**Retrieving SQL Return Code Information & Messages**

COBOL programs executing SQL statements communicate with DB2 via a Working Storage area called the SQL Communications Area (SQLCA).

When DB2 executes SQL statements, it returns the results of the operation into the SQLCODE and SQLSTATE fields in the SQLCA. SQLCODE provides key information about the success or failure of SQL statement execution.

If the **SQLWARNO** field in the SQLCA contains 'W', DB2 has set at least one of the SQL warning flags (SQLWARN1 through SQLWARNA). These flags provide additional info about execution of specific types of SQL.

Prior to DB2 V8, COBOL programs could call a subroutine called DSNTIAR that would convert a SQLCODE in the SQLCA into more a detailed text message with diagnostics about the return code.

New with DB2 V8, COBOL programs can now execute a GET DIAGNOSTICS statement that will return all previous SQLCA values and provide additional information about new DB2 V8 extended object names and new SQL functions.

The new GET DIAGNOSTICS function replaces existing SQLCA processing now found in most DB2 COBOL programs. GET DIAGNOSTICS also passes a text message about SQLCODE directly to programs.

**SQLCODE Overview**

* If SQLCODE = 0, execution was successful.
* If SQLCODE > 0, execution was successful with a warning.
* If SQLCODE < 0, execution was not successful.
* If SQLCODE = 100, "no data" was found. For example, a FETCH statement returned no data because the cursor was positioned after the last row of the result table.
* New with DB2 V8, when DB2 processes a multiple row FETCH statement, the contents of SQLCODE is set to +100 if the last row in the table has been returned with the set of rows.

**SQLCODE - Successful SQL Execution**

**SQL Return Code +100** ROW NOT FOUND FOR FETCH, UPDATE OR DELETE, OR THE RESULT OF A QUERY IS AN EMPTY TABLE.  
**Suggestion:** If expecting data, verify WHERE clause for accuracy and completeness.

**SQL Return Code +117** THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS.  
**Suggestion:** Correct SQL statement to provide only one value for each column in the table.

**SQL Return Code +231** CURRENT POSITION OF CURSOR cursor-name IS NOT VALID FOR FETCH OF THE CURRENT ROW.  
**Suggestion:** Be certain to FETCH to position on a row after opening a cursor. If cursor is declared SENSITIVE STATIC SCROLL, the row may be a hole, from which no values can be fetched.

**SQL Return Code +304** A VALUE WITH DATA TYPE data-type1 CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION position-number WITH DATA TYPE data-type2.  
**Suggestion:** Verify DCLGEN host variable definitions are current with DB2 catalog table/view attributes.

**SQL Return Code +347** THE RECURSIVE COMMON TABLE EXPRESSION name MAY CONTAIN AN INFINITE LOOP.  
**Suggestion:** Verify predicate in the SQL WHERE clause of the form "counter\_col < constant" or "counter\_col < :hostvar".

**SQL Return Code +802** EXCEPTION ERROR exception-type HAS OCCURRED DURING operation-type OPERATION ON data-type DATA, POSITION position-number.  
**Suggestion:** Check arithmetic operation for divide by zero or result to exceed size of host variable.

**SQLCODE - Unsuccessful SQL Execution**

**SQL Error Code -117** THE NUMBER OF VALUES ASSIGNED IS NOT THE SAME AS THE NUMBER OF SPECIFIED OR IMPLIED COLUMNS.  
**Suggestion:** Provide one value for each column in the table.

**SQL Error Code -150** THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW, SYSTEM-MAINTAINED MATERIALIZED QUERY TABLE, OR TRANSITION TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED.  
**Suggestion:** Be certain to specify base DB2 table/view names for INSERT statements.

**SQL Error Code -180** THE DATE, TIME OR TIMESTAMP VALUE value IS INVALID.  
**Suggestion:** Verify the data value is in the correct range and value type.

**SQL Error Code -181** THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE.  
**Suggestion:** Verify data format with the SQL Reference Guide.

**SQL Error Code -204** name IS AN UNDEFINED NAME.is returned when the object name(DB2 table) is not defined in the subssytem. Please check the table is existing or not.  
**Suggestion:** Correct DB2 CREATOR or OBJECT NAMEs located in SQL statements.

**SQL Error Code -227** FETCH fetch-orientation IS NOT ALLOWED, BECAUSE CURSOR cursor-name HAS AN UNKNOWN POSITION (sqlcode,sqlstate).  
**Suggestion:** CLOSE and re-OPEN the cursor; For scrollable use (FIRST, LAST, BEFORE, AFTER, or ABSOLUTE) to establish valid position.

**SQL Error Code -305** THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER position-number BECAUSE NO INDICATOR VARIABLE IS SPECIFIED.  
**Suggestion:** Add null indicator variable to SELECT statement in the format of "column:hostvarind".

**SQL Error Code -501** THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN.  
**Suggestion:** Correct logic in application program to OPEN the cursor before the FETCH or CLOSE statement.

**SQL Error Code -502** THE CURSOR IDENTIFIED IN AN OPEN STATEMENT IS ALREADY OPEN.  
**Suggestion:** Correct logic in application program to CLOSE the CURSOR before the OPEN statement.

**SQL Error Code -503** A COLUMN CANNOT BE UPDATED BECAUSE IT IS NOT IDENTIFIED IN THE UPDATE CLAUSE OF THE SELECT STATEMENT OF THE CURSOR.  
**Suggestion:** Use FOR UPDATE statement in your cursor.

**SQL Error Code -530** THE INSERT OR UPDATE VALUE OF FOREIGN KEY constraint-name IS INVALID.  
**Suggestion:** Ensure that INSERT row for DB2 PARENT table is completed before INSERT row in CHILD table

**SQL Error Code -532** THE RELATIONSHIP constraint-name RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'.  
**Suggestion:** Change the program to DELETE CHILD table row before DELETE of row on PARENT table.

**SQL Error Code -551** auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation ON OBJECT object-name.  
**Suggestion:** Contact the support DBA to GRANT the needed privilege.

**SQL Error Code -803** AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE indexspace-name CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS Xrid.  
**Suggestion:** Verify DB2 INDEX and, if needed, change the statement to an UPDATE.

**SQL Error Code -805** DBRM OR PACKAGE NAME location-name.collection-id.dbrm-name.consistency-token NOT FOUND IN PLAN plan-name. REASON reason.  
**Suggestion:** Ensure COLLECTION name is in DB2 PLAN. Recompile and BIND the DB2 program. Verify correct LOAD library is being used.

**SQL Error Code -811** THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE.  
**Suggestion:** -811 is often detected after program check for DB2 data existence. Consider using new DB2 V8 FETCH FIRST ROW ONLY feature instead.

**SQL Error Code -818** THE PRECOMPILER-GENERATED TIMESTAMP x IN THE LOAD MODULE IS DIFFERENT FROM THE BIND TIMESTAMP y BUILT FROM THE DBRM z.  
**Suggestion:** Recompile and BIND the DB2 program. Verify correct LOAD library is being used.

**SQL Error Code -904** UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.  
**Suggestion:** -904 is usually caused because a database utility job has started the desired DB2 object in utility mode. Check DB2 Master Log for more details on the resource name – contact DBA.

**SQL Error Code -911** THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.  
**Suggestion:** Review DB2 Master Log to find process holding DB2 locks. Consider adding additional COMMITs to program holding the DB2 resource.

**SQL Error Code -913** UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.(No rollback done)  
**Suggestion:** Review DB2 Master Log to find process holding DB2 locks. Consider adding additional COMMITs to program holding the DB2 resource.

**SQL Error Code -922** AUTHORIZATION FAILURE: error-type ERROR. REASON reason-code.  
**Suggestion:** Connection to DB2 has failed due authority for USER or PLAN. Contact DBA to check DB2 authorizations.

**SQL Error Code -927** THE LANGUAGE INTERFACE (LI) WAS CALLED WHEN THE CONNECTING ENVIRONMENT WAS NOT ESTABLISHED. THE PROGRAM SHOULD BE INVOKED UNDER THE DSN COMMAND.

**SQLCODE  -918, Error:**THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE A CONNECTION HAS BEEN LOST

**Interview Questions**

**Q: What is JOIN and what are the different types of JOIN.  
A:** The ability to join rows and combine data from two or more tables is one of the most powerful features of relational system. There are three type of joins:1. Equi-join; 2.Non-equijoin; 3.self-join

**Q: Which is the most widely used batch performance monitor for DB2?  
A:** DB2PM

**Q: Can I alter a table (like adding a column) when other user is selecting some columns or updating some columns from the same table?**   
**A:** Yes. It is possible until the updating or selection is committed, DB2 table will not be restructured. New column definition will be there but it will not be included until all the tasks on the table are committed.

**Q: What are the different methods of accessing DB2 from TSO? How is the connection established between TSO & DB2? A:**  
There are three ways in establishing TSO/DB2 connection 1. **SPUFI** 2. QMF 3. CATALOG VISIBILITY; A thread between TSO & DB2 is established while attempting to make connection between TSO & DB2.

**Q: How many buffer bools are available in DB2?  
A:** Ten 32K size bufferpools and fifty 4K size buffer pools (BP0 to BP49). Default buffer pools are BP0, BP1, BP2 & BP32

**Q: What is B37 abend during SPUFI?  
A:** The B37 ABEND in the SPUFI is because of space requirements. The query has resulted in so many rows that the SPUFI.OUT file is not large enough to handle it; One possible solution is to increase the space allocation of SPUFI.OUT file.

**Q: What is the command used by TSO users to invoke DB2?  
A:** DSN RUN

**Q: What is the error code -803 ?  
A**: Unique Index violation

**Q: How do you filter out the rows retrieved from a DB2 table?  
A:** One way is to use the SQL WHERE clause.

**Q: What is a collection?  
A:** A collection is something that every programmer should assign/specify for every package. This is about 1-18 characters long.

**Q: What is Skeleton Cursor Table (SKCT)?  
A:** The Executable form of a Plan. This is stored in SYSIBM.SCT02 table.

**Q: What is the equivalent Cobol Data type for Decimal (x,y) in DB2? What does the current SQLID register contain?  
A:** PIC S9(X-Y)V9(Y) COMP-3; The current SQLID contains the current authorization ID.

**Q.Can u please tell me the corelation b/w Plan, Package and collection.**

The DB2 Precompiler splits the program into two parts: a COBOL and a DB2 part. The embedded SQL is stripped out of the program and put into a partitioned data set (PDS) member, called a **DBRM**. Just as the COBOL part has to be compiled, the **DBRM part has to go through BIND process to create the run-time executable code for the DB2 portion of the COBOL program**. To help the COBOL and DB2 part to find each other later at run time, the precompiler engraves each with identical timestamps called **consistency tokens**.   
  
**You can BIND the DBRM into a PLAN (the old way), or you can BIND the instructions into a PACKAGE**.   
  
A PLAN is an executable module containing the access path logic produced by the DB2 optimizer**. The DBRMs of more than one program or PACKAGES can be bound into a PLAN.**  
A PACKAGE is a single, bound DBRM with optimized access paths. The DBRM of a single program is bound into a PACKAGE. To execute a PACKAGE, it should be included in the package list of a PLAN. **PACKAGEs are not directly executed, they are only indirectly executed when the PLAN in which they are contained executes.**  
  
**The relationship between a DBRM and a PLAN is one-to-many, the relationship between a DBRM and a PACKAGE is always one-to-one.**  
  
As the number of DBRMs bound to a PLAN increases, binding the DBRM into a PLAN is not recommended. If we need to precompile and bind a new program or one of the programs changes and it is to be precompiled and bound again, all the programs (not just the modified/added program) will be rebound into the PLAN again. Then the BIND process could take hours to complete.   
  
**On the other hand, if a DBRM is bound to a PACKAGE and if the program is modified, only that PACKAGE would have to be rebound.**  
  
A collection is simply a way of grouping PACKAGEs into meaningful groups. You could use COLLECTIONs to separate programs for different application areas, such as payroll and inventory. Another use might be to have customized set of BIND parameters associated with different COLLECTIONs.   
  
**At run time, the load module starts up and eventually hits a paragraph containing a CALL to DB2. Then the COLLECTIONs named in the PLAN are searched for the PACKAGE with the same name and consistency token. If you don't find it anywhere in DB2, you get an -805 error. If you're using the older technique of binding DBRMs directly into PLANs, then an unsuccessful search will result in an -818 error code.**

Q.What is difference between 911 and 913 error code in DB2 ?

911 - THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO        
DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE   
resource-type, AND RESOURCE NAME resource-name

913-UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT.    
REASON reason-code, TYPE OF RESOURCE resource-type, AND  
RESOURCE NAME resource-name

\*\*\*\*\*\*\*\*\*\*\*\*END of questions & and answers \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DB2 TOOLS**

 Go to **DB2I** PRIMARY OPTION MENU

4.  DB2I PRIMARY OPTION MENU

                             DB2I PRIMARY OPTION MENU          SSID: DSN          
 COMMAND ===>                                                                     
                                                                                  
 Select one of the following DB2 functions and press ENTER.                       
                                                                                  
  1  SPUFI                  (Process SQL statements)                              
  2  **DCLGEN**                 (Generate SQL and source language declarations)       
  3  PROGRAM PREPARATION    (Prepare a DB2 application program to run)            
  4  PRECOMPILE             (Invoke DB2 precompiler)                              
  5  BIND/REBIND/FREE       (BIND, REBIND, or FREE plans or packages)             
  6  RUN                    (RUN an SQL program)                                  
  7  DB2 COMMANDS           (Issue DB2 commands)                                  
  8  UTILITIES              (Invoke DB2 utilities)                                
  D  DB2I DEFAULTS          (Set global parameters)                               
  X  EXIT                   (Leave DB2I)                                          
                                                                                  
                    
                                                                                  
                                                                                  
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE     
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE

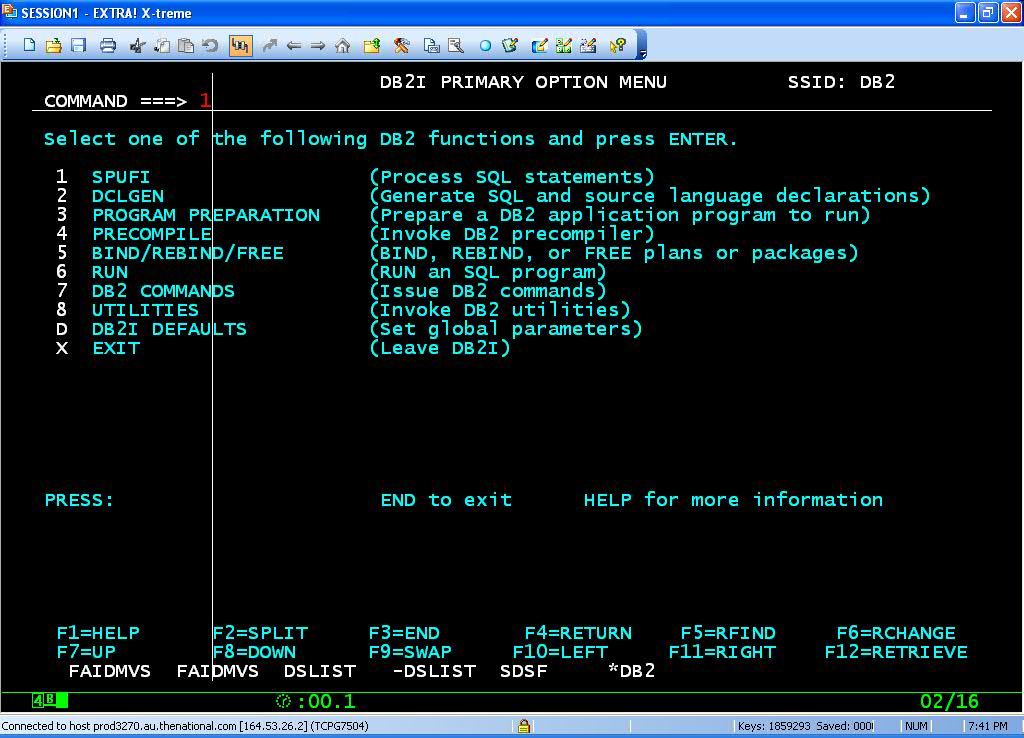
Fill the table name and the required options

                            DCLGEN                             SSID: DSN          
 ===>                                                                             
                                                                                  
 Enter table name for which declarations are required:                            
  1  **SOURCE TABLE NAME** ===> IND\_SUPP\_TAB                     (Unqualified)        
  2  TABLE OWNER ..... ===>                                     (Optional)        
  3  AT LOCATION ..... ===>                                     (Optional)        
                                                                                  
 Enter destination data set:          (Can be sequential or partitioned)          
  4  **DATA SET NAME** ... ===> 'INDU77.INDUS.DCLGENS(SUPPTAB)'                       
  5  DATA SET PASSWORD ===>           (If password protected)                     
                                                                                  
 Enter options as desired:                                                        
  6  ACTION .......... ===> ADD       (ADD new or REPLACE old declaration)        
  7  COLUMN LABEL .... ===> YES       (Enter YES for column label)                
  8  STRUCTURE NAME .. ===>                                     (Optional)        
  9  **FIELD NAME PREFIX** ===> HS-                                 (Optional)        
 10  DELIMIT DBCS .... ===> YES       (Enter YES to delimit DBCS identifiers)     
 11  COLUMN SUFFIX ... ===> YES       (Enter YES to append column name)           
 12  INDICATOR VARS .. ===> YES       (Enter YES for indicator variables)         
                                                                                  
                                                                                  
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE     
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE

|  |  |
| --- | --- |
| **Sample DCLGEN copy book :**      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* DCLGEN TABLE(IND\_STU)                                           \*        LIBRARY(INDU777.INDUS.COPYLIB(STUCPY))  \*        ACTION(REPLACE)                                          \*        LANGUAGE(COBOL)                                         \*        NAMES(HS-)                                              \*        STRUCTURE(STU-REC)  \*        QUOTE  \*        LABEL(YES)                                               \*        COLSUFFIX(YES)                                          \*        INDVAR(YES)                                              \* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      EXEC SQL DECLARE IND\_STU TABLE                                     ( STU\_NO                         CHAR(5),                            STU\_NAME                       CHAR(20),                           STU\_ADDR                       CHAR(20)       ) END-EXEC.                                                   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* COBOL DECLARATION FOR TABLE IND\_STU                             \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  01  STU-REC.                                                      \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_NO                                          10 HS-STU-NO            PIC X(5).                             \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_NAME                                        10 HS-STU-NAME          PIC X(20).                            \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_ADDR                                        10 HS-STU-ADDR          PIC X(20).                            \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* INDICATOR VARIABLE STRUCTURE                                    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  01  IIND-STU.       10 INDSTRUC           PIC S9(4) USAGE COMP OCCURS 3 TIMES.    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 3       \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |  |

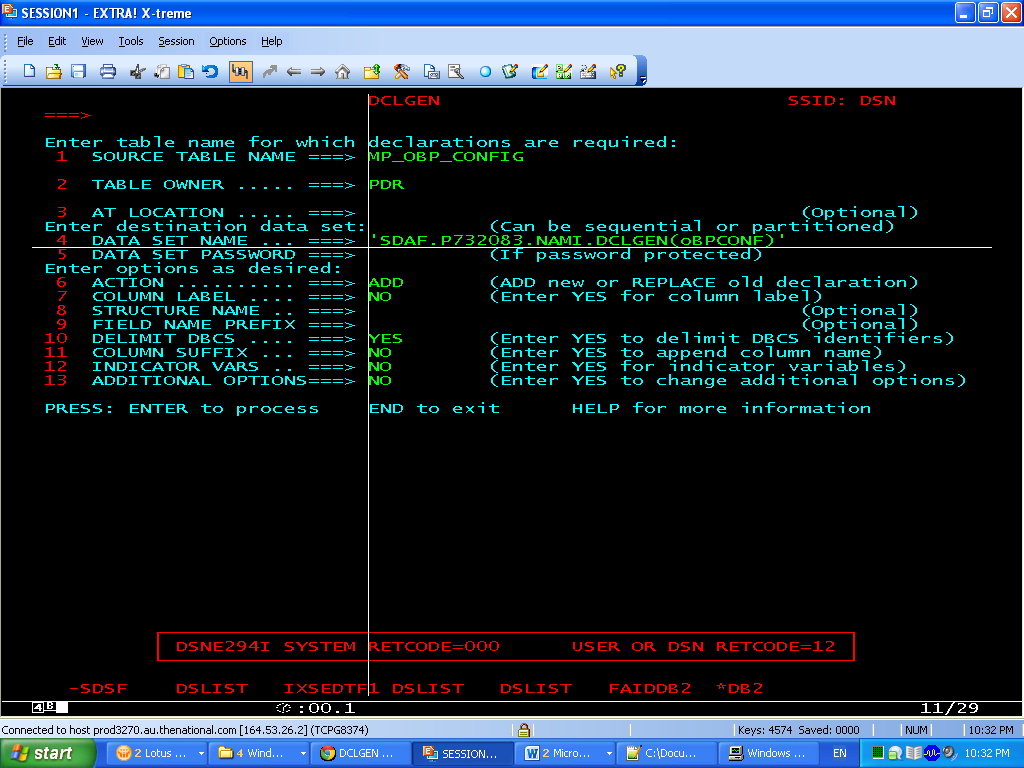
**DB2 Defaults settings**

D ( DB2I)



**DCLGEN**

**Go to D;;**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* DCLGEN TABLE(IND\_STU)                                          \*  
\*        LIBRARY(INDU777.INDUS.COPYLIB(STUCPY))                   \*  
\*        ACTION(REPLACE)                                         \*  
\*        LANGUAGE(COBOL)                                         \*  
\*        NAMES(HS-)                                              \*  
\*        STRUCTURE(STU-REC)                                      \*  
\*        QUOTE                                                   \*  
\*        LABEL(YES)                                              \*  
\*        COLSUFFIX(YES)                                          \*  
\*        INDVAR(YES)                                             \*  
\* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS   \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     EXEC SQL DECLARE IND\_STU TABLE                                
     ( STU\_NO                         CHAR(5),                     
       STU\_NAME                       CHAR(20),                    
       STU\_ADDR                       CHAR(20)

     ) END-EXEC.                                                   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* COBOL DECLARATION FOR TABLE IND\_STU                            \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 01  STU-REC.                                                      
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_NO                                     
     10 HS-STU-NO            PIC X(5).                             
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_NAME                                   
     10 HS-STU-NAME          PIC X(20).                            
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_ADDR                                   
     10 HS-STU-ADDR          PIC X(20).                            
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* INDICATOR VARIABLE STRUCTURE                                   \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**01  IIND-STU.**

**10 INDSTRUC           PIC S9(4) USAGE COMP OCCURS 3 TIMES.**    
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 3       \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ABENDS**

**+100**  ->  End of cursor

**-180**  ->String representation of DATE, TIME, TIMESTAMP is invalid

**-204**  ->  Object not defined to DB2

**-205**  ->  Column name not in specified DB2 table

**-206**  ->  Column does not exist in any table in the SELECT

**-551**  ->  Not authorized to access DB2

**-803**  ->  UnIque index violation. Try to insert duplicate record.

**-805**  ->  DBRM or PACKAGE not found in PLAN

**-811**  ->  Cursor should be used , when more than one row returned as an result

          of an singleton SELECT query

**-818**  ->  Timestamp mismatch between plan  and load module

**DB2 Commands**

**CLP** - Command Line Processor

Command Center

Control Center

**DB2CMD** - Open DB2 Command Window

**DB2EVA** - Event Analyzer

**DB2EVMON** - Event Monitor Productivity Tooldb2

**EXFMT** - Explain Table Format Tool

**DB2CKBKP** - Check Backup

**DB2CC** - Start Control Center

**DB2ADMIN** - DB2 Administration Server

**DB2ICRT** - Create Instance

**DB2IDROP** - Remove Instance

**DB2ILIST** - List Instances

**DB2TRC** - Tracedb2start - Start DB2

**DB2STOP** - Stop DB2

**DB2SYNC** - Start DB2 Synchronizer

**DB2TBST** - Get Tablespace State

The db2 command starts the command line processor (CLP). The CLP is used toexecute database utilities, SQL statements and online help. Common commandsinclude:

**RESTART DATABASE**

**CREATE DATABASE**

**DROP DATABASE**

**GET DATABASE CONFIGURATION**

**RESET DATABASE CONFIGURATION**

**UPDATE DATABASE CONFIGURATION**

**LIST DATABASE DIRECTORY**

**LIST ODBC DATA SOURCES**

**BACKUP DATABASE**

**RESTORE DATABASE**

**TERMINATE**

**QUIT**

**Launch Control Center for:**

Privileges, Authorities and Access Rights

**SQL: (Connect to a Database).**

**CONNECT TO**<dbname>**USER**<username>**USING**<userpswd>

EG:CONNECT TO TEST USER DB2ADMIN USING DB2ADMIN

**View a tables structure**

Connect to the DB first.

**DESCRIBE TABLE**<schema>.<tablename>

**Backup and Restore (also Copy a DB).**

Create a directory to backup to

Ex: C:\db2\mybackup

Backup your DB:

Ex: db2 =>**BACKUP DATABASE** SAMPLE **TO** c:\db2\mybackup

Restore the DB:

db2 =>**RESTORE DATABASE** SAMPLE **FROM** C:\db2\mybackup

**Restore the DB to another name (ie: Copy DB)**

db2 =>**RESTORE DATABASE** SAMPLE **FROM** C:\db2\mybackup **INTO** MYSAMPLE

**Run SQL Commands**

db2=> (examples)

**LIST DATABASE DIRECTORY** - lists databases on DB2 server

**CONNECT TO** sample **USER** db2admin **USING** db2admin - connects to a db.

**LIST TABLES** - lists tables in the DB.

**DESCRIBE SELECT** \* **FROM** emp\_act - lists the structure of a table.

**SELECT \* FROM** EMP\_ACT - list the contents in the table.

**DISCONNECT**<database name> - clears connection

**Reference**

<http://users.sdsc.edu/~jrowley/db2/howto.html>