# DB2

INTERACTING DB2

1. SPUFI -Sql processing using file inputs
2. QMF -query management facility
3. EMBEDDED SQL PROGRAM( COBOL)

SQL.

Every sql statement will return SQLCODE

Signed 3 digits.

Sqlcode → 0 be happy

Sqlcode is negative → cry for it

Sqlcode is positive → warinings/informations

+ 100 → end of table.

Data types : ref slides.

Char(05) ← sam → 5 bytes→ 'sam  '

Varchar(05) ← sam → 3 bytes.

DDL.

1. CREATE

CREATE TABLE/VIEW/TRIGGER/STOGROUP/UNIQUE INDEX <NAME>

(

OTHER ATTRIBUTES

);

CREATE TABLE TB_DEPT

(

) IN OZAGS1DB.OZAGS1TS;

Every attribute of a table must have

a. The name of the column

b. The data type(size)

c. [Constraints(rules)]

1. UNIQUE → no duplicates values are allowed in the column.
2. NOT NULL → mandatory.

NOT NULL WITH DEFAULT

It stores that column with default values depending on its data type.

NOT NULL WITH DEFAULT 12345

It accepts values from the user, if the user did not mention any values, then the system stores 12345 as the values for that column of the record.

Default → allows null value

3. PK( primary key)

Any unique column can be defined as PK, which acts a pivot access point to the records.

Note; only one PK for a table.

4. Foreign key. Referential constraint

5. Check constraint.

It allows the user to give a range of allowable values for the column.

Eg: gender (m,f,o)

Note: For the Primary key column and for all the UNIQUE columns, UNIQUE INDEX MUST BE CREATED. If not created, then the definition of the table is INCOMPLETE.

SYNTAX: CREATE UNIQUE INDEX <IDX9> ON TABLENAME(COLUMN NAME);

TB_DEPT

DID HOD DNAME

D001

D002

D003

D004

D005

D006

TB_EMPLOYEE

| EID | ENAME | DNO | SALARY | DOJ | GENDER |
|---|---|---|---|---|---|
| CHAR(04) | VARCHAR(15) | CHAR(04) | DECIMAL(7,2) | DATE | CHAR(1) M,F,O |
| | | FK | | | |
| E001 | TOMMY | D009 | 12345.67 | 2001-01-01 | |

REFERENTIAL INTEGRITY/REFERENTIAL CONSTRTAINT.

BUILDING RELATIONSHIP BETWEEN TABLES IS CALLED REFERTIAL CONSTRAINT.
This is done while defining the CHILD Table.
The table that has FOREIGH KEY is the child table.
A foreign key column CAN REFER ONLY TO THE PK column OF THE PARENT TABLE.
WHEN A INSERT SQL Statement is executed on the child table, the referential constraint will check if the value( being inserted in the child table) is in the PT.
If it is there, then the insertion is allowed in the CT.
If not, the insertion is rejected.

WHEN A DELETE sql statement is performed on the PT...
ON DELETE rules are activated.

- Cascade → allow deletion in the PT and delete all the depending records from the CT.
- Set null → allow deletion from the PT, SET NULL value the depending records FK column.
- Restrict → if there at least 1 depending record in the child table, RESTRICT the deletion from the PT.

R

DECIMAL(X,Y)
X→ TOTAL NUMBER OF DIGITS
Y → AFTER DECIMAL POINT

Other ways of creating table with constraints·

```
CREATE TABLE TB_EMP
      (
      EID CHAR(04) NOT NULL
      ,ENAME VARCHAR(15)
      ,DNO CHAR(04)
      ,SALARY DECIMAL(7,2)
      ,DOB DATE
      ,GENDER CHAR(01)
      ,PRIMARY KEY(EID)
      ,FOREIGN KEY(DNO) REFERENCES TB_DEPT(DID) ON DELETE
         SET NULL
      ,CHECK GENDER IN ('M','F','O')
      ) IN OZAGS1DB·OZAGS1TS;
```

COMPLEX SQLS·

PROJ DETAILS?

KNOWN ? NAME

```
SELECT PROJLEAD,PROLOC FROM PROJ WHERE PID =
        (SELECT PROJID FROM EMP WHERE ENAME='TIMMY')
```

SECOND MAX SAL?

```
Select max(sal) from emp where sal < (
                                SELECT MAX(SAL) FROM EMP)
```

third MAX SAL?

```
Select ename,eid,sal from emp where sal = (
        Select max(sal) from emp where sal <(
                Select max(sal) from emp where sal < (
                        SELECT MAX(SAL) FROM EMP)));
```

Write a query to display all the records who are working in the same department for which tommy works·

Write a query to get the HOD whose department is getting the minimum salary·(note: there can be more than one departments, use IN predicate between queries)

Tb_emp

| Eid | ename | pid(FK) | salary |
|---|---|---|---|
| E01 | tommy | p01 | 8098098 |

Tb_projects

| Proj_id | ploc |
|---|---|
| P01 | chennai |
| P02 | new jersy |

TB_EMP A  TB_EMP B

SAL  SAL  5 = 5

10
20
60
30
90
40
50
80

10
20
60
30
90
40
50
80

WRITE A QUERY TO DISPLAY THE ENAME AND EMP ID OF THE EMPLOYEES WHOSE AGE IS GREATER THAN 20.

EMBEDDED.

COMPILE

1. PRECOMPILE
   a. The cobol codes are separated from the sql statements.
   b. The sql statement is verified with the db2 catalogue.
   c. Query optimization(DBRM)
   d. The dclgen members are checked and included.
2. COBCOMP – COBOL SYNTAX ERROR
3. LOAD MODULE CREATION
4. BIND
   a. Plan is created.
   A plan is executable form of the best path to access the table for the SQL query mentioned in the program.

RUN

Combine the load module of the cobol program + the PLAN and are executed.

Points to be considered for embedded sql code.

1. SQLCA → SQL COMMUNICATION AREA.
   SQLCODE → SMILIAR TO FILE STATUS VARIABLE. It reflects the status of the sql statement.

   Data division.
   Working-storage section.

```
EXEC SQL
    INCLUDE SQLCA
END-EXEC.
```

   Including the SQLCA makes SQLCODE available with the updated status on the recent sql operation.
2. Dclgen variables. DECLARATION GENERATOR. ( NOT FOR CREATE STATEMENT)

Host variables.

THE LAYOUT aOF THE TABLE.

Make an DCLGEN entry for the table. This will create a list of cobol adaptable variables and deposits the same in a PDS (member).

Include that member in the working storage section of the embedded sql program.

Note: naming conventions → let all the cobol adaptable variables be prefixed by "HV-".

```
EXEC SQL
    INCLUDE <DCLGEN MEMBNAME>
END-EXEC.
```

Note: when a host variable is mentioned inside an SQL delimiter, IT MUST BE PREFIXED BY ':'.

3. DELIMITER.

All the SQL statements MUST BE ENCAPSULED by SQL delimiter.

```
Marigin B
    EXEC SQL
        Stmnts
    END-EXEC.
```

4. DSNTIAR .

Is a built-in sub program. WE need to call this subprogram to capture the sql error message and display the same in the our spool.

```
05 WS05-ERR-MSG.
    10 WS10-ERR-LEN      PIC S9(04) COMP VALUE 800.
    10 WS10-ERR-TXT      PIC X(80) OCCURS 10 TIMES.
05 WS05-ERR-LRECL        PIC  S9(09)  COMP VALUE 80.


    CALL 'DSNTIAR' USING SQLCA WS05-ERR-MSG WS05-ERR-LRECL
    DISPLAY WS05-ERR-MSG
```

5. THE STATUS of sql statement.

```
        EVALUATE TRUE
        WHEN SQLCODE = 0
```

Scenario:
Insert a record into tb_cob_emp

Points to considered·

1· Dclgen
2· Varchar
Handle length part and the text part separately·
3· Null value - null indicator·
User defined variable → pic S9(04) comp
WHILE INSERTING
THE USER HAS TO HANDLE THE NULL INDICATOR
WHILE SELECTING
THE SYSTEM DEPOSITS APPROPRIATE VALUES IN THE NULL INDICATOR

When 0 is deposited in the NULL INDICATOR, then the value in the host variable are considered as a valid value
When -1 is deposited in the NULL INDICATOR, then the value in the host variable are considered NULL, due to set null·
When -2 is deposited in the NULL INDICATOR, then the value in the host variable are considered NULL, due to data truncation error·

Note: Within the SQL statement, attach the null indicator with its host variable with a space in between·

Steps·

Move the values to the host variables·
Move ti001 TO HV-ID
Exec sql
Insert into tb_cob_emp values
(
:HV-ID
,:HV-NAME

```
            )
    End-exec.
    EVALUATE TRUE
    WHEN SQLCODE = 0
```

Scenario:

Selecting ONE record from the table into cobol pgm. SINGLETON SELECT.

Points to be noted:

Will the select statement bring a NULL value?

If yes, attach a null indicator.

Which record( unique) is about to be selected.

1. Sqlca
2. Dclgen mem
3. Select

```
Exec sql
    SELECT
        LIST OF COLUMNS
    INTO
        LIST OF HOST VARIABLES MATCHING THE SEQUENC OF COLUMNS SELECTED
        ALSO ATTACH THE NULL INDICATORS IF NEEDED
    FROM TABLENAME
    WHERE ID = :HV-ID
END-EXEC
```

Note: while displaying the VARCHAR variable, mentioning the TEXT part is enough.

Sqlcode = -305

What will happen if more than 1 record is selected.

Sqlcode -811 → multiple row select.

Scenario:

Delete records from cobol program

```
 Move 'e001' to hv-id
Exec sql
          Delete from tb_cob_emp
          Where id = :hv-id
End-exec
```

Note: multiple records can also be deleted.( without where clause)

Scenario:

update records from cobol program

```
 Move 'e001' to hv-id
Exec sql
          update tb_cob_emp
          set salary = salary * 1.1
          Where id = :hv-id
End-exec
```

Note: multiple records can also be updated.( without using where clause)

Task of the day:

Input file: Hlq.db2.ps

1001 tommy Chennai 12345.67 2000-10-10 M

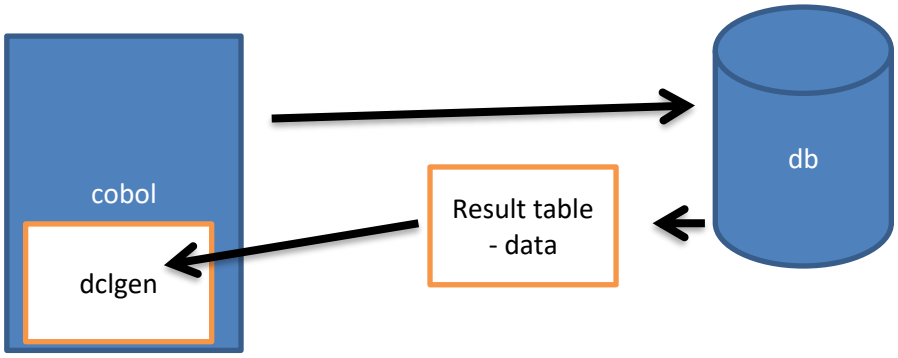
Output is tb_cob-emp.

Write a cobol program to read all records from the ps and insert each of them in the table tb_cob_emp.

The problem.

When a select query brings more than 1 record into the cobol program.

Sqlcode -811




Cursor:

1. Declare ( working-storage section)

Declaring a cursor is when

- a name for the cursor is given.
- Declared FOR a select statement( complex sql too)
- Exec sql

```
DECLARE CURNAME CURSOR
FOR <SELECT>
END-EXEC
```

2. Open(PROCEDURE DIVISION, OPEN-PARA)

```
EXEC SQL
    OPEN CURNAME
END-EXEC.
EVALUATE TRUE
WHEN SQLCODE = 000
```

```
   DISPLAY ' CURSOR OPENED'
WHEN OTHER
   DISPLAY ' CURSOR OPEN FAILED'
   CALL ' DSNTIAR' USING SQLCA WS05-ERR-MSG WS05-ERR-LRECL
   DISPLAY WS05-ERR-MSG
   PERFORM 9000-TERM-PARA
END-EVALUATE.
```

Note: *

- the select statement mentioned in the cursor will be executed.
- A Result Table( RT) is built in the buffer space. Temporary table.
- The structure of the RT is the column that are selected in the select statement mentioned in the declaration of the cursor.
- All the records that gushes out are captured into this RT.
- A pointer is positioned at the first record in the RT.
- Now, the RT is ready to be treated with sequential access.

3. Fetch until sqlcode = +100 ( PROCEDURE DIVISION, FETCH-PARA)

```
EXEC SQL
      FETCH CURNAME
      INTO
      :HV1
      ,:HV2  :NULL-IND
      ,HV3 :NULL-HV2
END-EXEC.
EVALUATE TRUE
WHEN SQLCODE = 000
      IF NULL-IND = 0 AND NULL-HV2 = 0
         DISPLAY DCLTB-COB-EMP
*        PERFORM  3210-WRITE-PARA
*           THRU 3210-WRITE-PARA-EXIT
      ELSE
         DISPLAY ' NULL VALUE IN THE RECORD'
      END-IF
WHEN SQLCODE = +100
```

```
            DISPLAY ' ALL RECORDS PROCESSED'
        WHEN OTHER
            DISPLAY ' CURSOR FETCH FAILED'
            CALL ' DSNTIAR' USING SQLCA WS05-ERR-MSG WS05-ERR-LRECL
            DISPLAY WS05-ERR-MSG
         END-EVALUATE.
```

4. Close (PROCEDURE DIVISION, CLOSE-PARA)

```
            EXEC SQL
                 CLOSE CURNAME
            END-EXEC.
```

```
3000-PROC-PARA.
        PERFORM 3100-OPEN-PARA
            THRU 3100-OPEN-PARA-EXIT
        PERFORM 3200-FETCH-PARA
            THRU 3200-FETCH-PARA-EXIT
            UNTIL SQLCODE = +100
        PERFORM 3300-CLOSE-PARA
              THRU 3300-CLOSE-PARA-EXIT
```

A. IT MUST BE IN THE FETCH PARA AFTER SUCCESSFUL FETCH.