

# Tutorial 4

## DB2

In this tutorial we will create and access a simple OS/390 - DB2 relational database. The database will have a single table and we will populate it with a limited set of data.

We logon under TSO. In the "CUSTOMPAC MASTER APPLICATION MENU" we select the option „P“ to access ISPF. We then call the "Data Set Utility" (see Fig. 1) .

```
Menu RefList Utilities Help
-----
                                Data Set Utility

A Allocate new data set          C Catalog data set
R Rename entire data set        U Uncatalog data set
D Delete entire data set        S Data set information (short)
blank Data set information      M Allocate new data set
                                V VSAM Utilities

ISPF Library:
Project . . . PRAKT20
Group . . . CICSDB2
Type . . . TEST01

Other Partitioned, Sequential or VSAM Data Set:
Data Set Name . . .
Volume Serial . . . (If not cataloged, required for option "C")

Data Set Password . . . (If password protected)

Option ==> A
F1=Help      F3=Exit      F10=Actions  F12=Cancel
```

*Fig. 1 : "Data Set Utility"*

We create three new partitioned data sets for the user "PRAKT20" :

PRAKT20.CICSDB2.TEST01 (see Fig. 1)  
PRAKT20.SPUFI.IN  
PRAKT20.DBRMLIB.DATA

Please use your user ID instead of PRAKT20 when you do the tutorial.

We use the parameters shown in Fig. 2 . The members of "PRAKT20.CICSDB2.TEST01" will store our programs and JCL scripts.

```

Menu  RefList  Utilities  Help
-----
Allocate New Data Set                               More:  +
Data Set Name . . . : PRAKT20.CICSDB2.TEST01
Management class . . . DEFAULT      (Blank for default management class)
Storage class . . . . PRIM90        (Blank for default storage class)
Volume serial . . . . SMS001        (Blank for system default volume) **
Device type . . . . .                (Generic unit or device address) **
Data class . . . . .                (Blank for default data class)
Space units . . . . . KILOBYTE      (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
Average record unit . . . . .        (M, K, or U)
Primary quantity . . 16              (In above units)
Secondary quantity . . 1             (In above units)
Directory blocks . . 5               (Zero for sequential data set) *
Record format . . . . FB
Record length . . . . 80
Block size . . . . . 320
Data set name type : PDS             (LIBRARY, HFS, PDS, or blank) *
                                       (YY/MM/DD, YYYY/MM/DD)
Command ==>
F1=Help      F3=Exit      F10=Actios  F12=Cancel

```

Fig. 2 : Partitioned Dataset Parameter

The members of "PRAKT20.SPUFI.IN" will store our database definitions. The dataset "PRAKT20.DBRMLIB.DATA" is used to store intermediate results.

In addition we need a partitioned dataset "PRAKT20.LIB". During a compile step it will store templates for the next step. We assume "PRAKT20.LIB". has already been created in tutorial 3. If not, you will have to allocate it using the parameters shown in Fig. 2 .

```

CUSTOMPAC MASTER APPLICATION MENU
OPTION ==> DB2                               SCROLL ==> PAGE

IS  ISMF      - Interactive Storage Management Facility
P   PDF       - ISPF/Program Development Facility
ATC ATC       - Application Testing Collection
ART ARTT     - Automated Regression Testing Tool
DB2 DB2      - Perform DATABASE 2 interactive functions
QMF QMF      - QMF Query Management Facility
C   CPSM     - CICSplex/SM
M   MQ       - MQSeries
IP  IPCS     - Interactive Problem Control Facility
OS  SUPPORT  - OS/390 ISPF System Support Options
OU  USER    - OS/390 ISPF User Options
SM  SMP/E    - SMP/E Dialogs
SD  SDSF     - System Display and Search Facility
R   RACF     - Resource Access Control Facility
DI  DITTO    - Data Interfile Transfer, Testing and Operations
HC  HCD      - Hardware Configuration Definition
S   SORT     - DF/SORT Dialogs
BMR BMR READ - BookManager Read (Read Online Documentation)

F1=HELP      F2=SPLIT      F3=END        F4=RETURN     F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN        F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE

```

Fig. 3: Calling ISPF DB2 Subsystem

After having allocated all datasets we return to the "CUSTOMPAC MASTER APPLICATION MENU"-panel by repeatedly pressing the F3 key.

We enter "DB2" on the command line and press enter (Fig. 3). This calls an ISPF subsystem that permits to create a new DB2 database.

```
DB2I PRIMARY OPTION MENU          SSID:
COMMAND ==> d

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)

P  DB2 OM             (Performance Monitor)
C  DC Admin           (Data Collector Admin)

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
```

Fig. 4 : "DB2I PRIMARY OPTION MENU"

The "DB2I PRIMARY OPTION" menu appears (Fig. 4). As a first step we have to set the global parameters of our database. We enter "d" on the command line. Press Enter.

```
DB2I DEFAULTS
COMMAND ==>

Change defaults as desired:

1  DB2 NAME ..... ==> DBA1      (Subsystem identifier)
2  DB2 CONNECTION RETRIES ==> 0  (How many retries for DB2 connection)
3  APPLICATION LANGUAGE ==> IBMCOB (ASM, C, CPP, COBOL, COB2, IBMCOB,
FORTRAN, PLI)
4  LINES/PAGE OF LISTING ==> 60  (A number from 5 to 999)
5  MESSAGE LEVEL ..... ==> I    (Information, Warning, Error, Severe)
6  SQL STRING DELIMITER ==> DEFAULT (DEFAULT, ' or ")
7  DECIMAL POINT ..... ==> .    (. or ,)
8  STOP IF RETURN CODE >= ==> 8  (Lowest terminating return code)
9  NUMBER OF ROWS ..... ==> 20  (For ISPF Tables)
10 CHANGE HELP BOOK NAMES?==> NO (YES to change HELP data set names)
11 DB2I JOB STATEMENT: (Optional if your site has a SUBMIT exit)
    ==> //PRAKT20 JOB (ACCOUNT),'NAME'lector Admin)
    ==> /*
    ==> /*
    ==> /*

F1=HELP   F2=SPLIT   F3=END     F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP    F10=LEFT    F11=RIGHT  F12=RETRIEVE
```

Fig. 5 : Subsystem Identifier

A SSID (**S**ubsystem **I**dentifier) is a naming of a database which the OS/390 system uses internally. Our system "jedi.informatik.uni-leipzig.de" uses the identifier "DBA1". (Another OS/390 system may use a different name. The system administrator will know which name to use.)

We enter the two values as indicated in Fig. 5. We accept the remaining default values. Enter

```
COBOL DEFAULTS
COMMAND ==>

Change defaults as desired:

1 COBOL STRING DELIMITER ==> DEFAULT (DEFAULT, ' or ")
2 DBCS SYMBOL FOR DCLGEN ==> G      (G/N - Character in PIC clause)

F1=HELP    F2=SPLIT    F3=END      F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP      F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT  F12=RETRIEVE
```

Fig. 6 : DEFAULTS

No changes to the screen in Fig. 6. Enter

```
DB2I PRIMARY OPTION MENU          SSID: DBA1
COMMAND ==> 1

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)

P DB2 OM        (Performance Monitor)
C DC Admin      (Data Collector Admin)

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
```

Fig. 7: "DB2I PRIMARY OPTION MENU"

The "DB2I PRIMARY OPTION MENU"-Panel reappears (Fig. 7). Different from Fig. 4, the SSID is now listed as "DBA1" .

We use the "SPUFI" ISPF-Subsystem to create our database. In Fig. 7 we enter "1" in the command line. SPUFI is a primitiv, yet powerful ISPF subsystem that may be used to create and define a data base. Press Enter .

```

          SPUFI                      SSID: DBA1
====>

Enter the input data set name:      (Can be sequential or partitioned)
 1 DATA SET NAME ... ===>
 2 VOLUME SERIAL ... ===>        (Enter if not cataloged)
 3 DATA SET PASSWORD ===>      (Enter if password protected)

Enter the output data set name:    (Must be a sequential data set)
 4 DATA SET NAME ... ===>

Specify processing options:
 5 CHANGE DEFAULTS  ===> YES      (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ===> YES      (Y/N - Enter SQL statements?)
 7 EXECUTE .....   ===> YES      (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ===> YES      (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ===> YES      (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ===>

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN       F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE

```

Fig. 8 : SPUFI-Panel

The screen shown in Fig. 8 appears . We call it the "SPUFI-Panel or the SPUFI-Screen. We will use it quite a few times during this tutorial.

To create a new database we need definitions which define the "what", "how", and „where“. We will store these definitions in members of our dataset "PRAKT20.SPUFI.IN"

We will create 4 definitions to be stored in 4 members of PRAKT20.SPUFI.IN . These definitions contain:

1. Type, location (area on one of many disk drives), size and properties of the disk storage space that will store our database. This storage space is called a Storage Group (STOGROUP) and receives a name ("STOGR020" in our case).

2. Symbolic name of the database. We select "DB020".

A database usually caches a part of its active data within main storage. This cache is usually called a "buffer pool" and receives a symbolic name (BP0 in our case).

3. A relational database consists of at least one table (usually more). We have to reserve disk storage space (called table space) for each table. This table space is part of the storage group and also receives a symbolic name. In our case we reserve space for just one table and name it "TABSP020" .

4. Finally we have to define the table itself in terms of a name ("TAB020") ist structure and the naming of ist fields (columns). Our table will have two colums and the format shown in Fig. 9.

VNAME	NNAME
.....	.....
.....	.....
.....	.....
.....	.....

Fig. 9 : Structure of our Database Table

To summarize, we will generate four members for our partitioned data set "SPUFI.IN" . They are to have the following functions:

Member-Name	Identifier	Function
STOGR1	STOGR020	storage group (space) for our database storing the database itself
DB1	DB020	
TABSP1	TABSP020	storage space for a table
TAB1	TAB020	storing the table itself

The column "Identifier" contains the name of the database, the table, and the associated disk space. These data are stored in members of the partitioned data set SPUFI.IN, whose names are contained in the column "Member-Name".

The names in the two colums could be identical, but it is easier to use different names. We suggest that for the names in the center column you replace the last 3 characters ( "020" ) by the last 3 characters of your user ID.

Using the SPUFI screen we are going to define a storage group member, fill it with data, save this, review the succesful operation, and return to the SPUFI screen. We will go through this loop repeatedly. This loop includes 5 different screen and is shown in Fig. 10 . Also shown are transitions from one screen to the next (either the F3 key or the Enter key).

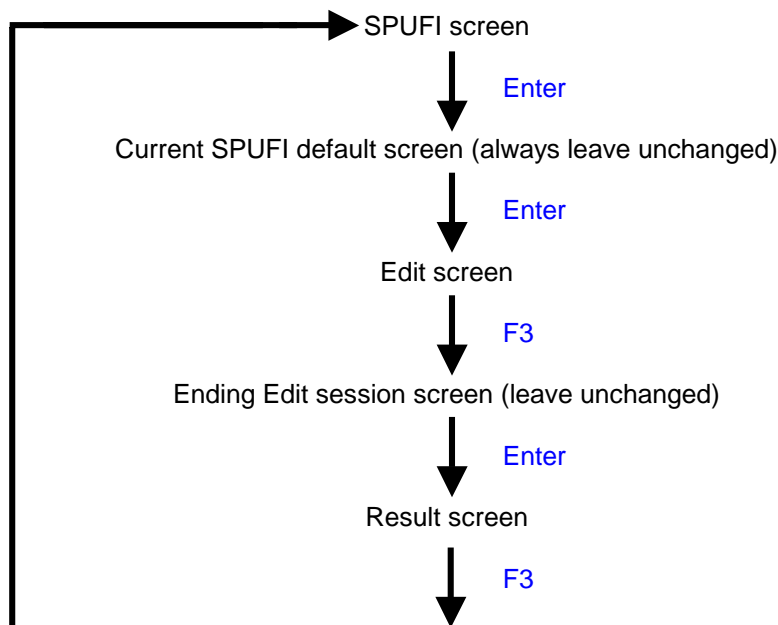


Fig. 10 : The SPUFI Screen Loop

We start defining the storage group member, starting with the SPUFI screen shown in Fig. 8.

```
SPUFI                                SSID: DBA1
===>

Enter the input data set name:        (Can be sequential or partitioned)
 1 DATA SET NAME ... ===> SPUFI.IN(STOGR1)
 2 VOLUME SERIAL ... ===>          (Enter if not cataloged)
 3 DATA SET PASSWORD ===>          (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4 DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5 CHANGE DEFAULTS  ===> YES        (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ===> YES        (Y/N - Enter SQL statements?)
 7 EXECUTE .....   ===> YES        (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ===> YES        (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ===>

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT   F12=RETRIEVE
```

*Fig. 11 : Storage Group Definition*

We want to store the storage group definition in the member "PRAKT20.SPUFI.IN(STOGR1)". SPUFI is going to translate this definition into an internal format which will be stored in a member of a new data set which we decide to name SPUFI.OUT. This data set will be automatically allocated by SPUFI.

We enter the data as shown in Fig. 11 . Enter





SPUFI indicates that editing the member SPUFI.IN(STOGR1) has been completed (Fig. 15). This is what we expected, so it is not much new news. We press Enter. This causes the translation of our definition. The result is stored in SPUFI.OUT .

```

Menu  Utilities  Compilers  Help
-----
BROWSE      PRAKT20.SPUFI.OUT                      Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
CREATE STOGROUP STOGR020                                00010000
  VOLUMES (SCPMV5)                                    00020000
  VCAT DSN510;                                         00030000
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 3
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 15
***** Bottom of Data *****

Command ==>
F1=Help    F3=Exit    F5=Rfind   F12=Cancel

Scroll ==> PAGE

```

Fig. 16 : Translation was successful

Fig. 16 indicates, the translation was successful. We will receive an error message like in Fig. 17 if the disk storage space for the STOGROUP is already available, meaning it has already been assigned in a previous session. In this case we have to free it up. This can be done with the SQL statement

DROP STOGROUP STOGR020

Erasing the STOGROUP is only possible if it contains no table space reservation. Otherwise you will receive an error message as shown in Fig. 18 . The error message indicates the name of the object that needs to be erased, in this case the tablespace "DB020.TABSP020". For this use the SQL-Statement

DROP TABLESPACE DB020.TABSP020

Both SQL-statements may be executed as follows: In the SPUFI-Panel in Fig. 11 enter values in the green field. In place of "STOGR1" use a member name which does not yet exist in the dataset SPUFI.IN" , e.g. "DELTABSP" (DELeTe TABLE Space). This member stores the SQL-Statement , e.g. "DROP TABLESPACE DB020.TABSP020" .

```

Menu Utilities Compilers Help
-----
BROWSE   PRAKT20.SPUFI.OUT                               Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+-----+-----+
CREATE STOGROUP STOGR020                                00010000
  VOLUMES (SCPMV5)                                     00020000
  VCAT DSN510;                                         00040000
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNT408I  SQLCODE = -601, ERROR:  THE NAME OF THE OBJECT TO BE CREATED OR THE
          TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME STOGR020
          OF THE OBJECT TYPE STOGROUP
DSNT418I  SQLSTATE   = 42710 SQLSTATE RETURN CODE
DSNT415I  SQLERRP    = DSNXICSG SQL PROCEDURE DETECTING ERROR
DSNT416I  SQLERRD    = 10 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I  SQLERRD    = X'0000000A' X'00000000' X'00000000' X'FFFFFFFF'
          X'00000000' X'00000000' SQL DIAGNOSTIC INFORMATION
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE618I  ROLLBACK PERFORMED, SQLCODE IS 0
DSNE616I  STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE601I  SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
Command ==>                                         Scroll ==> PAGE
F1=Help   F3=Exit   F5=Rfind  F12=Cancel

```

Fig. 17 : Error Message, Object exists already

```

Menu Utilities Compilers Help
-----
BROWSE   PRAKT20.SPUFI.OUT                               Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DROP STOGROUP STOGR020                                00010000
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNT408I  SQLCODE = -616, ERROR:  STOGROUP STOGR020 CANNOT BE DROPPED BECAUSE IT
          IS REFERENCED BY TABLESPACE DB020.TABSP020
DSNT418I  SQLSTATE   = 42893 SQLSTATE RETURN CODE
DSNT415I  SQLERRP    = DSNXIDSG SQL PROCEDURE DETECTING ERROR
DSNT416I  SQLERRD    = 60 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I  SQLERRD    = X'0000003C' X'00000000' X'00000000' X'FFFFFFFF'
          X'00000000' X'00000000' SQL DIAGNOSTIC INFORMATION
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE618I  ROLLBACK PERFORMED, SQLCODE IS 0
DSNE616I  STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE601I  SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I  NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I  NUMBER OF INPUT RECORDS READ IS 1
DSNE622I  NUMBER OF OUTPUT RECORDS WRITTEN IS 18
Command ==>                                         Scroll ==> PAGE
F1=Help   F3=Exit   F5=Rfind  F12=Cancel

```

Fig. 18 : Error Message, STOGROUP contains a Table Space

With the F3 key we return to the SPUFI screen (Fig. 19).

In many cases the step we just executed (reserving disk storage space for the database) will have to be done by the system administrator, because the student users may not have the necessary rights. The following steps however do not require any special authorisation.

```

SPUFI                                SSID: DBA1
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:        (Can be sequential or partitioned)
 1 DATA SET NAME ... ===> SPUFI.IN(STOGR1)
 2 VOLUME SERIAL ... ===>          (Enter if not cataloged)
 3 DATA SET PASSWORD ===>         (Enter if password protected)

Enter the output data set name:      (Must be a sequential data set)
 4 DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5 CHANGE DEFAULTS  ===> YES       (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ===> YES       (Y/N - Enter SQL statements?)
 7 EXECUTE .....   ===> YES       (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ===> YES       (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ===> YES       (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ===>

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP     F10=LEFT    F11=RIGHT   F12=RETRIEVE

```

Fig. 19 : SPUFI-Screen

As a next step we define a database within the reserved disk storage space "STOGR020" .

```

SPUFI                                SSID: DBA1
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:        (Can be sequential or partitioned)
 1 DATA SET NAME ... ===> SPUFI.IN(DB1)
 2 VOLUME SERIAL ... ===>          (Enter if not cataloged)
 3 DATA SET PASSWORD ===>         (Enter if password protected)

Enter the output data set name:      (Must be a sequential data set)
 4 DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5 CHANGE DEFAULTS  ===> YES       (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ===> YES       (Y/N - Enter SQL statements?)
 7 EXECUTE .....   ===> YES       (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ===> YES       (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ===> YES       (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ===>

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP     F10=LEFT    F11=RIGHT   F12=RETRIEVE

```

Fig. 20: Definition for the Database DB1

As shown in Fig. 20 we enter the name of the member which will store the database definition. Enter



```

SPUFI                                SSID: DBA1
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:        (Can be sequential or partitioned)
 1 DATA SET NAME ... ===> SPUFI.IN(TABSP1)
 2 VOLUME SERIAL ... ===>          (Enter if not cataloged)
 3 DATA SET PASSWORD ===>         (Enter if password protected)

Enter the output data set name:      (Must be a sequential data set)
 4 DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5 CHANGE DEFAULTS ===> YES        (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ===> YES       (Y/N - Enter SQL statements?)
 7 EXECUTE ..... ===> YES          (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ===> YES       (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ===> YES      (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ===>

F1=HELP      F2=SPLIT      F3=END      F4=RETURN    F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN      F9=SWAP    F10=LEFT    F11=RIGHT   F12=RETRIEVE

```

Fig. 23 : Tablespace Definition

Our example assumes a database that consists of a single table. We have to reserve disk storage space for the table, called "tablespace". This is a piece of the disk storage space (storage group) we already reserved for the database. We store the definition of the tablespace in a member named "TABSP1" of our partitioned data set PRAKT20.SPUFI.IN .

We enter the name of this member as shown in fig. 23 . Pressing the Entry key two times moves us to the "Edit Entry Panel" .

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAKT20.SPUFI.IN(TABSP1) - 01.00          Columns 00001 00072
*****      ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
'''''' CREATE TABLESPACE TABSP020
''''''      IN DB020
''''''      USING STOGROUP STOGR020
''''''      PRIQTY 20
''''''      SECQTY 20
''''''      ERASE NO
''''''      BUFFERPOOL BP0
''''''      CLOSE NO;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>
F1=Help      F3=Exit      F5=Rfind     F6=Rchange   F12=Cancel

```

Fig. 24 : Defining the Tablespace for our Table

We create a tablespace named "TABSP020" (see Fig. 24). It will store a table TAB020 which is a part (here the only part) of the database "DB020", which is stored in the storage group "STOGR020". The associated buffer pool is named "BP0". We save the screen content by pressing F3, followed by pressing Enter.

```
Menu Utilities Compilers Help
-----
BROWSE      PRAKT20.SPUFI.OUT                      Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+
CREATE TABLESPACE TABSP020                        00010000
  IN DB020                                         00020000
  USING STOGROUP STOGR020                          00030000
  PRIQTY 20                                        00040000
  SECQTY 20                                        00050000
  ERASE NO                                         00060000
  BUFFERPOOL BP0                                   00070000
  CLOSE NO;                                        00080000
-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
                                                00090000
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
Command ==>
F1=Help    F3=Exit    F5=Rfind   F12=Cancel          Scroll ==> PAGE
```

Fig. 25 : Definition was successful

Successful definition. Pressing the F8 key we can scroll forward to view the rest of the message. We scroll backward using the F7 key.

We press F3 to leave the screen.



```

Menu Utilities Compilers Help
-----
BROWSE      PRAKT20.SPUFI.OUT                      Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+-----+-----+
CREATE TABLE TAB020                                00010000
(                                                    00020000
  VNAME CHAR(20) NOT NULL,                          00030000
  NNAME CHAR(20) NOT NULL                            00040000
)                                                    00050000
  IN DB020.TABSP020;                                00060000
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 6
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 18
Command ==>                                         Scroll ==> PAGE
F1=Help      F3=Exit      F5=Rfind    F12=Cancel

```

Fig. 28 : Successful Definition

Definiton was successful. We leave the screen pressing F3.

```

SPUFI                      SSID: DBA1
==>

Enter the input data set name:      (Can be sequential or partitioned)
1  DATA SET NAME ... ==> SPUFI.IN(ININSERT)
2  VOLUME SERIAL ... ==>      (Enter if not cataloged)
3  DATA SET PASSWORD ==>      (Enter if password protected)

Enter the output data set name:     (Must be a sequential data set)
4  DATA SET NAME ... ==> SPUFI.OUT

Specify processing options:
5  CHANGE DEFAULTS ==> YES      (Y/N - Display SPUFI defaults panel?)
6  EDIT INPUT ..... ==> YES    (Y/N - Enter SQL statements?)
7  EXECUTE ..... ==> YES      (Y/N - Execute SQL statements?)
8  AUTOCOMMIT ..... ==> YES   (Y/N - Commit after successful run?)
9  BROWSE OUTPUT ... ==> YES   (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ==>

F1=HELP      F2=SPLIT      F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
F7=UP        F8=DOWN       F9=SWAP     F10=LEFT    F11=RIGHT    F12=RETRIEVE

```

Fig. 29: Definition for INSERT

OK, we have created a database. At this point it is still empty. There are many ways to enter data into a data base. We select a primitive approach using the SPUFI subsystem.

We create (in addition to the existing 4 members) an additional member "PRAKT20.SPUFI.IN(ININSERT)" (Fig. 29) . Press Enter two times.



```

Menu Utilities Compilers Help
-----
BROWSE      PRAKT20.SPUFI.OUT                      Line 00000019 Col 001 080
-----+-----+-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 3
DSNE621I NUMBER OF INPUT RECORDS READ IS 6
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 27
***** Bottom of Data *****

Command ==>
F1=Help    F3=Exit    F5=Rfind   F12=Cancel

Scroll ==> PAGE

```

Fig. 32 : 2nd Part of the Fig. 29 Screen

We believe we have entered data into our data base. Lets test this. We call the SPUFI screen by pressing F3.

```

SPUFI                      SSID: DBA1
==>
DSNE800A NO DEFAULT VALUES WERE CHANGED. PRESS ENTER TO CONTINUE
Enter the input data set name:      (Can be sequential or partitioned)
 1 DATA SET NAME ... ==> SPUFI.IN(SELECT)
 2 VOLUME SERIAL ... ==>          (Enter if not cataloged)
 3 DATA SET PASSWORD ==>         (Enter if password protected)

Enter the output data set name:     (Must be a sequential data set)
 4 DATA SET NAME ... ==> SPUFI.OUT

Specify processing options:
 5 CHANGE DEFAULTS ==> *          (Y/N - Display SPUFI defaults panel?)
 6 EDIT INPUT ..... ==> YES      (Y/N - Enter SQL statements?)
 7 EXECUTE .....   ==> YES      (Y/N - Execute SQL statements?)
 8 AUTOCOMMIT ..... ==> YES      (Y/N - Commit after successful run?)
 9 BROWSE OUTPUT ... ==> YES     (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ==>

F1=HELP    F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP      F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT   F12=RETRIEVE

```

Fig. 33 : SELECT Definition

We can use SPUFI to view the database content. We create another SPUFI-Member "PRAKT20.SPUFI.IN (SELECT)" (Fig. 33) Press Enter two times.



OK, success .Maybe you want to experiment entering additional data into the database. Press F3 to call the SPUFI screen, enter "INSERT" , press Enter 2 times to access the "Edit Screen" . Now you may enter additional data. Press F3 to save them. Then press Enter to return to the "Browse Screen".

```

Menu Utilities Compilers Help
-----
BROWSE   PRAKT20.SPUFI.OUT                               Line 00000000 Col 001 080
***** Top of Data *****
-----+-----+-----+-----+-----+-----+-----+-----+
SELECT * FROM PRAKT20.TAB020;                               00010000
-----+-----+-----+-----+-----+-----+-----+
VNAME          NNAME
-----+-----+-----+-----+-----+-----+
HANS           BAUER
FRED           MAYER
JORG           WAGNER
HEINZ         BAUER
FRITZ         MAYER
RICHARD       SCHULTE
JORG          MEISTER
HANS          BERG
FRITZ         MEIER
RICHARD       SCHMITZ
MARTIN        WAGNER
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
Command ==>
F1=Help   F3=Exit   F5=Rfind  F12=Cancel
                                           Scroll ==> PAGE

```

Fig. 36 : Database with additional Content

The result might look like shown in Fig. 36. Use the F8 key to scroll to the next output screen (Fig. 37).

```

Menu Utilities Compilers Help
-----
BROWSE   PRAKT20.SPUFI.OUT                               Line 00000019 Col 001 080
-----+-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 1
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 27
***** Bottom of Data *****
-----+-----+-----+-----+-----+-----+-----+
Command ==>
F1=Help   F3=Exit   F5=Rfind  F12=Cancel
                                           Scroll ==> PAGE

```

Fig. 37 : 2nd Part of the Screen Output

OK, we have successfully created a data base and entered data into it. We will use it in the next tutorial (tutorial 5).

Please understand we are explaining basic principles. There are many other procedures, some highly automated, that are being used in real-life situations. Several of the following tutorials will access DB2 from a Java program, using JDBC, SQLJ and DB2Connect.

This ends Tutorial 4 .