

Command line processor DB2

Task 1. Connecting to the database SAMPLE.

1. start the command line processor (CLP)
2. start DB2 database manager (db2start)
3. start the interactive input mode (db2)
(in the interactive input mode we have the db2 => input prompt)
4. view currently active databases (list active databases)
5. connect to the db SAMPLE as STUDENT (you will be prompted for the password)
connect to sample user student
or connect with the default user ID, with
connect to sample
6. list the table spaces for db SAMPLE (list tablespaces)
to see more information (the number and size of pages) add option show detail
7. list the tables for database SAMPLE (list tables)
8. get the id of the currently logged in user (values current user)
9. check, if you are connected to a database (get connection state)
10. list information about the applications connected to the SAMPLE (list applications) (the id of the current application one can read with the command values application_id)
11. open new CLP window and connect to SAMPLE
12. in the second CLP window, list all applications currently connected to SAMPLE
13. force one of the applications to be disconnected from the database (force the first CLP)
force application (id) (use the application-handle as id)
14. in the second CLP window, check all applications currently connected to SAMPLE
15. quit the interactive mode using terminate (it closes also the database connection) and close this window
16. in the first CLP window, check your connection state and disconnect from database, if you are connected; next, quit the interactive mode and close the window

Task 2. Working with database objects: tables

1. open CLP and start the interactive mode
2. check all active databases and the connection state; if you are connected then disconnect
3. create new database with the name testdb: create database testdb
4. list all active databases
5. connect to the database testdb
6. list information about table spaces in db testdb and their page sizes and numbers
7. in db testdb create new table employee:
create table employee (empid integer, name varchar(50), dept integer)
list info about this table: describe table employee
8. change the data type of field dept to char(9)
alter table employee alter column dept set data type char(9)
list info about table employee
9. try to insert new row to table employee (using insert into)
10. if this fails, read the error code; use help to read what this code means and how to manage it (print ?SQLcode); do suggested operations
11. insert three rows to table employee (it can be done with one insert statement)
12. change the value of field dept in one of the rows (statement update)
13. delete one row (statement delete)
14. delete all rows from table employee: truncate table employee immediate
15. after each modification, check what is currently stored in employee (use select)

Task 3. Working with database objects: Schemas.

A **schema** is a collection of objects; it gives a way to group those objects logically. A schema can contain tables, views, nicknames, triggers, functions, packages, and other objects.

Database object names can be made up of a single identifier or they might be *schema-qualified objects* made up of two identifiers. In such a case, the schema name provides a means to classify or group objects in the database and the name of an object has two parts: schema_name.object_name. Hence we can use the same name for different objects in different schemas.

When an object (such as a table, view, alias, distinct type, function, index, package or trigger) is created, it is assigned to a schema. This assignment is done either **explicitly** (the object is specifically qualified with a schema name when created, then the object is assigned to that schema) or **implicitly** (the default schema name is used, i.e., the current schema, which is usually the same as the user ID). Schemas can be also created implicitly by users whenever they create an object with a schema name that does not already exist or explicitly (with the command CREATE SCHEMA schema-name [AUTHORIZATION schema-owner-name]).

1. find out in what schema is the table employee in db testdb.
2. was this schema created explicitly or implicitly?
3. list all schemas in db testdb, use system view schemata in schema syscat (connect to db testdb first in an interactive mode): select schemaname from syscat.schemata
4. in db testdb create new schema: myschema and create in this schema table department with fields dept char(9), dept_name varchar(30), budget numeric(9,2);
(to create the table and assign it to the schema, use schema name as the qualifier of the table name in the create statement: schema-name.table-name. Table names must be all distinct in one schema.
5. list all tables in schema myschema using list tables for schema schema-name
(use option for all to list tables of all schemas)
6. list all schemas from db testdb, for each of them list information about owner, definer and creation time (first check the structure of the view syscat.schemata)
7. in schema myschema create table employee with the same structure, as the table employee created in task 2.7; answer, if this can be done and why
8. list all tables in db testdb for the current schema and for all schemas.

To remove an unused schema, use drop schema schema-name restrict
(the schema can be dropped only if it does not have any objects).