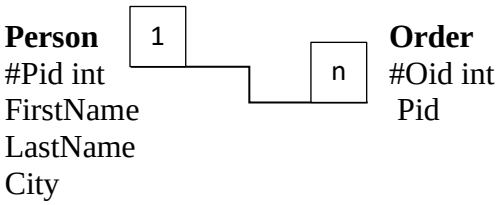
Emilia Pop

# ***Laboratory 3***

Consider 2 tables:




1. Add new column
   ALTER TABLE Person
   ADD Dob date

2. Modify the type of a column
   ALTER TABLE Person
   ALTER COLUMN Dob int NOT NULL

3. Remove a column
   ALTER TABLE Person
   DROP COLUMN Dob

4. Create new table
   CREATE TABLE Person(
   Pidint NOT NULL PRIMARY KEY,
   FirstName varchar(50) NOT NULL,
   LastName varchar(50),
   City varchar(50)
   );

5. Add new column with default constraint
   ALTER TABLE Person
   ADD Dob int DEFAULT 2000;

6. Modify column with default constraint
   ALTER TABLE Person
   ADD DEFAULT 18 FOR Age;

   ALTER TABLE Person
   ADD CONSTRAINT df_18 DEFAULT 18 FOR Age

7. Remove default constraint from a column
   ALTER TABLE Person
   DROP CONSTRAINT df_18;

8. Delete a table
   - Delete all the structure of the table and the records
     DROP TABLE Person

   - Delete only the records (with condition)
     DELETE FROM Person
     [WHERE Dob>2000]

9. Create a foreign key constraint on a new table
   CREATE TABLE Order (
   Oid int NOT NULL PRIMARY KEY,
   Pid int CONSTRAINT fk_Order_Person FOREIGN KEY(Pid) REFERENCES Person(Pid)
   );

10. Create a foreign key as a new add column in a table
    ALTER TABLE Order
    ADD CONSTRAINT fk_Order_Person FOREIGN KEY(Pid) REFERENCES Person(Pid)

11. Remove a foreign key
    ALTER TABLE Order
    DROP CONSTRAINT fk_Order_Person;

12. Create a primary key constraint in a new table
```
CREATE TABLE Order (
Oid INT NOT NULL,
Pid INT CONSTRAINT fk_Order_Person FOREIGN KEY(Pid) REFERENCES Person(Pid)
CONSTRAINT pk_Order PRIMARY KEY(Oid)
);
```

13. Create a primary key constraint as a new add column in a table already created
```
ALTER TABLE Order
ADD CONSTRAINT pk_Order PRIMARY KEY(Oid)
```

14. Remove a primary key constraint
```
ALTER TABLE Order
DROP CONSTRAINT pk_Order
```

15. Create a unique constraint in a new table (candidate key)
```
CREATE TABLE Order(
Oid INT NOT NULL
CONSTRAINT uk_Order UNIQUE(Oid)
);
```

16. Create a unique constraint as a new add column in a table
```
ALTER TABLE Order
ADD CONSTRAINT uk_Order UNIQUE(Oid)
```

17. Remove a unique constraint
```
ALTER TABLE Order
DROP CONSTRAINT uk_Order
```

| **7 procedures do** | **7 procedures undo (reverse)** |
|---|---|
| do_proc_1 – modify the type of the column | undo_proc_1 – modify the type of the column (back) |
| do_proc_2 – add a column | undo_proc_2 – remove a column |
| do_proc_3 – add a default constraint | undo_proc_3 – remove a default constraint |
| do_proc_4 – create a primary key | undo_proc_4 – remove a primary key |
| do_proc_5 – create a candidate key (unique) | undo_proc_5 – remove a candidate key (unique) |
| do_proc_6 – create a foreign key constraint | undo_proc_6 – remove a foreign key constraint |
| do_proc_7 – create a table | undo_proc_7 – remove a table |

PAY ATTENTION to the name of the procedures – because with their names you work in the main procedure (but please don't use the ones from up).
A table Version will contains the version of the database (version 0 – the first one – the one it is now). It is kept in an integer column, from where can be extracted using the Select instruction.

main 4 – will take the database from version 0 to version 4 (crossing through versions 1, 2, 3, 4)
- version 1 – will be given by executing do_proc_1
- version 2 – will be given by executing do_proc_2
- version 3 – will be given by executing do_proc_3
- version 4 – will be given by executing do_proc_4

main 2 – will take the database from version 4 (the actual one that should be is kept in the table Version) to version 2 (crossing version 4, 3)

version 4 – will be given by executing undo_proc_4
version 3 – will be given by executing undo_proc_3

After each execution, or change of the version of the database, the new version will be updated in the table Version.

main 8 – will return an error message, because that version does not exist

STORED PROCEDURES can be found in the Database (your database) -> Programmability -> Stored Procedures -> (right click) Stored Procedures.

Examples of a stored procedure name without parameter:

| | |
|---|---|
| CREATE PROCEDURE do_proc_1<br>AS<br>BEGIN<br>-- the code<br>SELECT * FROM Produs<br>END<br>/* EXECUTE (to create the stored procedure and find it in the list of stored procedures */ | Run the procedure (in a new query):<br>EXECUTE do_proc_1 /<br>EXEC do_proc_1 /<br>do_proc_1 |

Examples of a stored procedure with parameters:

| | |
|---|---|
| CREATE PROCEDURE main<br>@vers int,<br>@t varchar(50)<br>AS<br>BEGIN<br>IF @vers>5<br>BEGIN<br>SELECT * FROM Produs<br>END<br>IF @t='Cluj'<br>BEGIN<br>PRINT ' DONE'<br>END<br>END | Run the procedure (in a new query):<br>EXEC main 6, 'Alba' /<br>EXEC main 1, 'Cluj' /<br>EXEC main 7, 'Cluj' |

Each stored procedure will have a different name and after EXECUTE it will appear in the list of the stored procedures (at Refresh). This means that the procedure was created and can be used (in main procedure or wherever you want).

To run the procedure: open a New Query and write EXECUTE procedure_name [parameters].
EXECUTE main 3 / EXEC main 4 / EXEC main @vers=3

Instructions:

| | |
|---|---|
| 1. WHILE condition<br>BEGIN<br>….<br>END | 2. IF condition<br>BEGIN<br>…<br>END<br>[ELSE<br>BEGIN<br>…<br>END] |

3. PRINT 'Your message.'
4. DECLARE @a INT -- to declare a variable
5. SET @a=@a +1 -- to modify the value of a variable
6. EXEC @text -- to execute (run) the instruction saved in variable @text

Suggestion: One can use 'PRINT' instruction in each stored procedure to specify what operation was performed or the version of the database.
Also, 'PRINT' instruction can be use to return the error messages (OR RAISERROR).

The operations must be executed in the order in which appear on the requirement!!!

Steps in solving the requirement:

1. **Create the version table** (Version(versionNo int), OR Version (oldVersion int, newVersion int, ModificationDate date) OR Version(versionNo int, StoredProceduresNames varchar(20)), OR as you want to have it, with the possibility of finding out the current version of the database.
2. **Create the stored procedures** (7 direct and 7 reverse = 14 stored procedures)
3. **Create the main stored procedure – with one parameter**
   - Validate the parameter (number, text, …) – if it is not ok, print an error message and finish the program (e.g. if the value of the parameter is 8 and you have only 7 versions, print message and finish the program).
   - Extract the current version of your database from the version table.
   - Compare the current version of the database to the version in which you want to take your own database (version given by the parameter).
     - If current_version<new_version then execute the direct stored procedures (as many as you need to get to the new_version)
     - If current_version>new_version then execute the reverse stored procedures (as many as you need to get to the new_version)
     - If current_version=new_version the print a corresponding message
   - Update the new version in the version table