Laboratory 2 Table structure configuration constraints

Development and Editing: Z. Garofalaki, A. Tsolakidis, P. Andritsos

Target

Changes to table structure after creation, familiarity with defining constraints.

Tools

A. Table structure configuration – ALTER statements

There is the possibility of configuring the structure of a table, after its initial creation and while it also contains data, without the need to delete and re-create it. This capability is provided by the following variations of the ALTER command.

1. Adding **a column** named NEASTILI and data type varchar(20) to a named table PINAKAS, is done with the command:

```
alter table PINAKAS add NEASTILI varchar(20);

# Check result, display PINAKAS array structure
describe PINAKAS;
```

2. Changing **data type of column** named NEASTILI from varchar(20) to varchar(30), is done with the command:

```
alter table PINAKAS modify NEASTILI varchar(30);
```

3. Deleting **a column** named NEASTILI is done with the command:

```
alter table PINAKAS drop NEASTILI?
```

4. Renaming **a column** named NEASTILI to NEWCLM is done with the following command. It is noted that with this variant of ALTER we can rename or change the data type at the same time in one column.

```
alter table PINAKAS change NEASTILI NEWCLM varchar(30);

# Renaming a column and simultaneously changing its data type
alter table PINAKAS change NEASTILI NEWCLM char(10);
```

5. Renaming **a table** named PINAKAS to PNKS is done with the following command:

```
alter table PINAKAS rename to PNKS?

# Check result, show all tables
show tables?
```

B. Limitations

Constraints are used to limit the data that can be entered into a table, as well as to ensure the accuracy and reliability of the data in the table. If there is a violation of the restriction by any data-related action, h

energy is rejected. Constraints can be column or table level. The most commonly used constraints are:

- NOT NULL: a column cannot have a NULL value in any record
- UNIQUE: all values in a column are unique
- PRIMARY KEY: (combination of NOT NULL and UNIQUE) uniquely identifies each record
- FOREIGN KEY: prevents actions that will destroy connections between tables
- CHECK: the values in a column satisfy a certain condition
- DEFAULT: default value for a column (when no value is set)
- CREATE INDEX: to quickly create/retrieve data from the NW

1. Adding a PRIMARY **KEY constraint** to a table named PINAKAS and defining the PID column of the table as PK is done with the following command. Note that a pk_PID name will be set for the constraint.

```
alter table PINAKAS add constraint pk_PID primary key (PID);
```

2. The removal of PRIMARY **KEY constraint** in a table named PINAKAS and definition as PK of the PID column of the table is done with the following command:

```
alter table PINAKAS drop primary key?

# Check result, display PINAKAS array structure
describe PINAKAS;

# Syntax of the create table command to create the PINAKAS table
show create table PINAKAS;
```

3. Adding a **foreign key constraint** (FOREIGN KEY) to a table named PINAKAS and defining as FK the P2ID column of the table (the PINAKAS.P2ID column will connect the elements of the table with the elements of PINAKAS2), is done with the following command. Note fk_P2ID name will be set for the constraint.

```
alter table PINAKAS add constraint fk_P2ID foreign key(P2ID) references PINAKAS2(P2ID);
```

4. Removing **a foreign key constraint** (FOREIGN KEY) in a table named PINAKAS and defining the P2ID column of the table as FK with name fk_P2ID is done with the following command:

```
alter table PINAKAS drop foreign key fk_P2ID;
```

5. Adding a **default value constraint (DEFAULT)** to a table named PINAKAS, setting a default value of 'UniWA' for the UNI column of the table, is done with the following command:

```
alter table PINAKAS alter UNI set default 'UniWA';
```

6. Removing the **DEFAULT constraint** on a table named PINAKAS for the UNI column of the table is done with the following command:

```
alter table PINAKAS alter UNI drop default?
```

7. The **AUTO_INCREMENT constraint** is applied to a column and is automatically managed with serial number the column value. Note that insert statements cannot change a value in a column with this restriction. The addition **of limitation**

AUTO_INCREMENT on the PID column (the column must be or be set to PK) of a table named PINAKAS, is done with the following command:

```
alter table PINAKAS change PID PID int not null auto_increment primary key;

# Alternatively, add constraint when creating table
create table PINAKAS
(PID int(11) not null auto_increment,
P2ID int(11),
UNI varchar(10),
primary key (PID))
engine=InnoDB default charset=utf8mb4;

# Change the initial value of AUTO_INCREMENT (default is 1)
alter table PINAKAS auto_increment = 100;
```

8. Removing the **AUTO_INCREMENT constraint** may cause error messages, as the constraint is applied to columns that have other constraints (such as the primary key). For this reason, it is necessary to execute the declaration of cancellation, as follows:

```
# Disable foreign key checks
set foreign_key_checks=0;

alter table PINAKAS modify PID int not null;

# Enable foreign key checks
set foreign_key_checks=1;
```

9. The list of restrictions on NW personnel is displayed with the following command:

```
select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where CONSTRAINT_SCHEMA='personnel';
```

Activities

Carry out the following activities. The command or commands required for the implementation of each step, as well as the result of its execution should be included in a deliverable file in text or screenshot format. The file or files with your answers should be compressed into a **xx_ÿÿÿÿÿ_EPONYMO.zip,** where: (a) xx is the number of the section you belong to (e.g. for group [02] MONDAY 12:00-13 :00, **xx = 02)** and (b) YYYYYY your Registration Number This final file will be submitted to the e-class -> DATABASES II -> Assignments.

1. **Connect** to your system's MySQL using any of the above methods you wish.

2. Check if there is a DB with the name **personnel.** If it exists, delete it and recreate it.

```
# Display NW
show databases?

# Delete ND personnel
drop database if exists personnel?

# Create ND personnel
create database personnel?
```

3. Select the NW personnel to **use.**

```
# Select personnel to use
use personnel?
```

4. Ensure that personnel **has no** contained tables. If it has, delete them.

```
# Display tables
show tables?

# Delete table named onoma_pinaka
drop table onoma_pinaka;
```

5. Create the **DEPT, JOB** and **EMP** tables without adding keys. To be taken considering the data types described in Table 1.

```
# Create table DEPT
create table DEPT(DEPTNO numeric(2), DNAME varchar(24), LOC char(23));

# Check result, display DEPT array structure
describe DEPT?
```

6. **Data** is entered into the tables so that their contents are what are shown in Figure 16.

```
# Insert a record into the EMP table
insert into EMP(EMPNO,NAME,JOBNO,DEPTNO,COMM) values (10, 'CODD', 100, 50, NULL);

# Check result, display EMP table contents
select * from DEPT;
```

7. Add **HIREDATE column** with data type DATE ('YYYY-MM-DD') to the EMP table. Redisplay the structure of the EMP table.

```
# Insert HIREDATE column into EMP table
alter table EMP add HIREDATE date;

# Check result, show EMP array structure
describe EMP?
```

8. Update the HIREDATE column of the EMP table with **values** given that the employee hire dates are: CODD 10-Jan-2001, NAVATHE 25-Feb-1999, ELMASRI 17-Mar-2000 and DATE 7-Jun-1989.

```
# Update HIREDATE column with CODD hire date
update EMP set HIREDATE='2001-01-10' where empno=10;

# Check result, display EMP table contents
select * from DEPT;
```

9. Set data type varchar(30) to the JOB_DESCR column of the JOB table.

10. Drop the LOC column of the DEPT table.

11. Rename the DNAME column of the DEPT table to DEPT_NAME. Also set **data type** varchar(25) to column DEPT_NAME. Note that these changes should occur with the execution of a single statement.

12. Add a PRIMARY **KEY constraint** named pk_DEPTNO to table DEPT.

```
alter table DEPT add constraint pk_DEPTNO primary key (DEPTNO);
```

13. Remove the PRIMARY **KEY constraint** on the DEPT table. Show the result of the removal (a) by displaying the structure of the DEPT table and (b) by displaying the create table command for the DEPT table (step B.2).

14. Add PRIMARY **KEY constraint** and **foreigner constraints** (FOREIGN KEY) in all tables.

15. Remove the **FOREIGN KEY constraints** on the EMP table. Show the result of each removal by (a) displaying the structure of the EMP table and (b) displaying the create table command for the EMP table (step B.2).

16. Add **a default value** (DEFAULT) constraint to the DEPT table, setting a default value of 'Development' for the table's DNAME column (step B.5). Show the effect of the constraint by using the following commands and showing the result of executing each command.

```
insert into EMP(EMPNO,NAME,JOBNO,DEPTNO,COMM) values (10, 'CODD', 100, 50, NULL);

insert into DEPT(DEPTNO) values (10), (20);

insert into DEPT values (30, 'Sales');
```

17. Display the list of BD personnel restrictions.

18. Create PROJECT table with P_ID and P_NAME columns. Consider the data types described in Table 1. Define **a primary key** and set **an AUTO_INCREMENT constraint** on the P_ID column of the table.

19. Run at least one **record entry command** to demonstrate the operation of the constraint. Also display the result of executing the command.

20. Set **an initial value of 200** for the AUTO_INCREMENT constraint and repeat previous step 19.

NW personnel

The tables contained in the personnel database should have the following structure and contents:

Columns Data type	
DEPT.DEPTNO, EMP.DEPTNO	numeric(2)
DNAME, JOB_DESCR	varchar(24)
LOC	char(23)
JOBCODE, JOBNO	numeric(3)
SAL, COMM	numeric(10,2)
EMPNO	numeric(4)
PROJECT.P_ID	int
PROJECT.P_NAME	varchar(255)

Table 1. Data types of tables EMP, JOB, DEPT

EMP

EMPNO	NAME	JOBNO	DEPTNO	COMM
10	CODD	100	50	
20	NAVATHE	200	50	450
30	ELMASRI	300	60	
40	DATE	100	50	

JOB

JOBCODE	JOB_DESCR	SAL
100	SALESMAN	2000
200	ANALYST	2000
300	DBA	3000

DEPT

DEPTNO	DNAME	LOC
50	SALES	ATHENS
60	ACCOUNTING	ATHENS
70	PAYROL	VOLOS

Figure 1. EMP, JOB, DEPT table data