საქართველოს ტექნიკური უნივერსიტეტი

რომან სამხარაძე, ნინო ჯოჯუა, ლია გაჩეჩილაძე, მ.ქურდაძე

მეთოდური მითითებები ლაბორატორიული სამუშაოების შესასრულებლად საგანში: Transact-SQL ენა 1

I ნაწილი

საქართველოს ტექნიკური უნივერსიტეტი

რომან სამხარაძე, ნინო ჯოჯუა, ლია გაჩეჩილაძე, მ.ქურდაძე

მეთოდური მითითებები ლაბორატორიული სამუშაოების შესასრულებლად საგანში: Transact-SQL ენა 1

I ნაწილი

დამტკიცებულია
მეთოდურ მითითებად
სტუ-ის სარედაქციო
საგამომცემლო
საბჭოს მიერ

თბილისი - 2013

მეთოდიკური მითითებების პირველი ნაწილი განკუთვნილია იმს ფაკულტეტის ბაკალავრიატის სტუდენტებისთვის ლაბორატორიული სამუშაოების ჩასატარებლად. მასში განხილულია მონაცემთა ბაზების არქიტექტურა, ცხრილები, მონაცემთა და მომხმარებლების ტიპები, ცხრილში მონაცემების ჩამატება, მონაცემების ამორჩევა, ჩადგმული და ბმული მოთხოვნები და ა.შ. თითოეულ ლაბორატორიულ სამუშაოს თან ახლავს მოკლე თეორიული ცნობები და შესაბამისი საკითხის პრაქტიკული გადაწყვეტა.

რეცენზენტები: სრული. პროფესორი მედეა ანდღულაძე
ასოც. პროფესორი ზაურ ჯოჯუა

სარჩევი

ლაბორატორიული სამუშაო N1
SQL Server გარემოს აღწერა

SQL Server Management Studio-ს (სურ.1) გაშვების შემდეგ გამოჩნდება დიალოგური ფანჯარა (სურ.2), სადაც ვირჩევთ სერვერის ტიპს Database Engine და ავტორიზაციის რეჟიმს: windows ან SQL Server. SQL Server-ის შემთხვევაში უნდა გავიართ ავტორიზაცია: login ველში ვუთითებთ სახელს sa, ხოლო password ველში პაროლს.

Database Engine წარმოადგენს ძირითად სერვისულ უზრუნველყოფას, რომელიც შედის SQL Server -ის პაკეტში მონაცემების დამუშავებისა და დაცვისათვის.

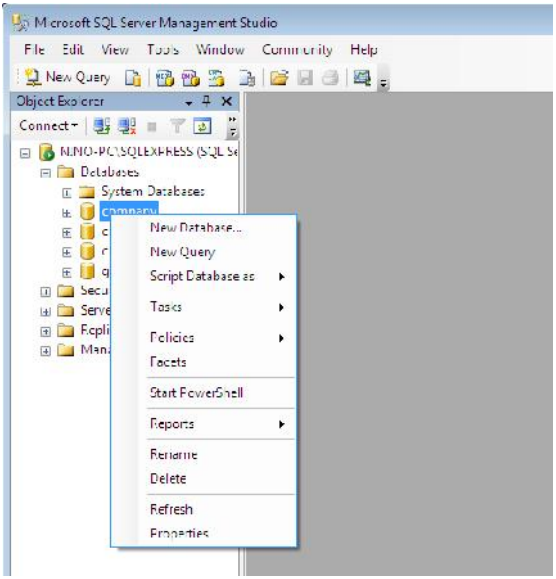
მისი ძირითადი ამოცანაა:

- უზრუნველყოს მონაცემების საიმედო დაცვა
- მონაცემების სწრაფი ამოღების საშუალებებით უზრუნველყოფა
- უზრუნველყოფს მოთხოვნების ადეკვატური დამუშავება
- აკონტროლებს მონაცემებითან წვდომის უსაფრთხოებას
- მონაცემების საიმედოობისა და ადეკვატურობი-სათვის ახდენს მონაცემთა სისრულის წესის დაცვას

სურ.1

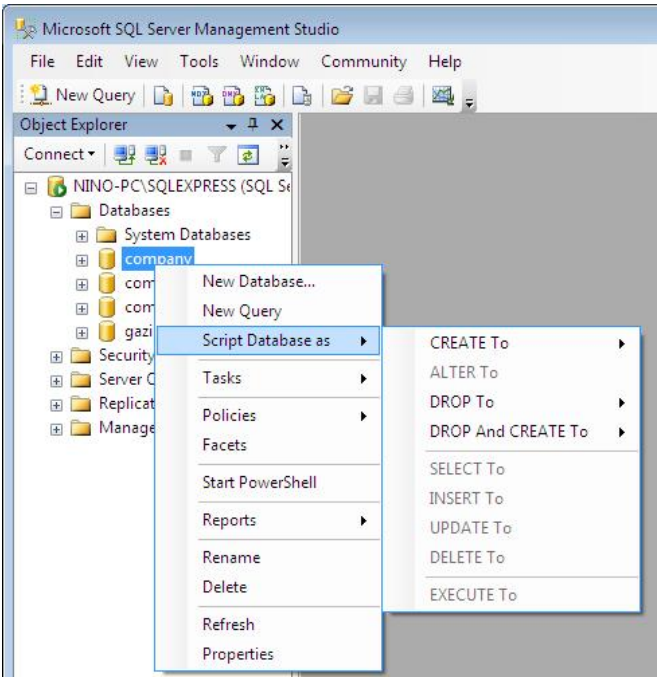
სურ.2

სურ.3-4-ზე მოცემულია კონტექსტური მენიუ საიდანაც შესაძლებელია მონაცემთა ახალი ბაზის, მოთხოვნების და სხვადასხვა ამოცანების, როგორიცაა სახელის შეცვლა, განახლება, წაშლა, მონაცემთა ბაზის სარეზერვო ასლის შექმნა, აღდგენა სარეზერვო ასლიდან, მონაცემთა იმპორტი და ექსპორტი და ა.შ. შესრულება.



სურ.3

სურ.4


Microsoft SQL Server Management Studio
File Edit View Tools Window Community Help
New Query
Object Explorer
Connect
NINO-PC\SQLEXPRESS (SQL Se
Databases
System Databases
company
com
com
gazi
Security
Server O
Replicat
Manage
New Database...
New Query
Script Database as
Tasks
Policies
Facets
Start PowerShell
Reports
Rename
Delete
Refresh
Properties
CREATE To
ALTER To
DROP To
DROP And CREATE To
SELECT To
INSERT To
UPDATE To
DELETE To
EXECUTE To

ლაბორატორიული სამუშაო N2
მონაცემთა ბაზის შექმნა

მონაცემთა ბაზის შესაქმნელად უნდა მივუთითოთ მისი სახელი, მფლობელის სახელი, (ეს იქნება მონაცემთა ბაზის შემქმნელი მომხმარებელი), ზომა, ფაილები და ფაილების ჯგუფები, რომლებისგანაც იქნება შემდგარი შესაქმნელი მონაცემთა ბაზა.

ონაცემთა ბაზის შესაქმნელად გამოიყენება CREATE DATABASE ბრძანება. მისი სინტაქსია:

CREATE DATABASE მონაცემთა_ბაზის_სახელი

[ON [PRIMARY] [<ფაილის_განსაზღვრა> [,...n]] [, <ფაილების_ჯგუფი> [,...n]]]

[LOG ON { <ფაილის_განსაზღვრა> [,...n] }]

[FOR ATTACH]

სადაც,

- მონაცემთა_ბაზის_სახელი შესაქმნელი მონაცემთა ბაზის სახელია.
- ON მიუთითებს, რომ იწყება მონაცემთა ბაზის ფაილების განსაზღვრა.
- PRIMARY მიუთითებს, რომ იწყება მონაცემთა ბაზის პირველადი ფაილის აღწერა. მონაცემთა ბაზაში მხოლოდ ერთი ფაილი შეიძლება იყოს განსაზღვრული როგორც პირველადი. თუ პირველადი ფაილი აშკარად არ არის განსაზღვრული, მაშინ პირველად ფაილად გამოყენებული

იქნება <ფაილის_განსაზღვრა> კონსტრუქციაში მითითებული პირველი ფაილი.

- LOG ON განსაზღვრავს ტრანზაქციების ჟურნალის ფაილებს. თუ ეს არგუმენტი არ არის მითითებული, მაშინ სერვერი ავტომატურად ქმნის ტრანზაქციების ჟურნალის ერთ ფაილს, რომლის სახელი იქმნება მონაცემთა ბაზის სახელისათვის _Log სიმბოლოების დამატების გზით.
- FOR ATTACH. გამოიყენება მაშინ, როცა უნდა შესრულდეს სერვერთან მონაცემთა ბაზის მიერთება. ამ შემთხვევაში უნდა არსებობდეს შესაბამისი ფაილები. მონაცემთა ბაზის მისაერთებლად საკმარისია მხოლოდ მონაცემთა ბაზის პირველადი ფაილის ადგილმდებარეობის მითითება.

<ფაილის_განსაზღვრა> კონსტრუქციის სინტაქსია:

<ფაილის_განსაზღვრა> ::=

([NAME = ფაილის_ლოგიკური_სახელი,] FILENAME = 'ფაილის_ფიზიკური_სახელი'

[, SIZE = ზომა] [, MAXSIZE = { მაქსიმალური_ზომა | UNLIMITED }]

[, FILEGROWTH = ნაზრდი]) [,...n]

სადაც,

- NAME = ფაილის_ლოგიკური_სახელი ფაილის ლოგიკური სახელია. ის უნდა იყოს უნიკალური მონაცემთა ბაზის ფარგლებში.
- FILENAME = 'ფაილის_ფიზიკური_სახელი' შეიცავს ფაილის ფიზიკურ სახელსა და გზას, რომელიც შექმნილი იქნება დისკზე.

- SIZE = ზომა. მიუთითებს ფაილის საწყის ზომას მეგაბაიტებში (Mb).
- FILEGROWTH = ნაზრდი განსაზღვრავს ნაზრდის ზომას ფაილისათვის. ნაზრდის ზომა შეიძლება მივუთითოთ მეგაბაიტებში ან პროცენტებში (ფაილის საწყისი ზომიდან).
- MAXSIZE = { მაქსიმალური_ზომა | UNLIMITED }. ფაილის მაქსიმალური ზომა ეთითება მეგაბაიტებში. თუ ფაილის ზომა არ უნდა შეიზღუდოს, მაშინ უნდა მივუთითოთ UNLIMITED მნიშვნელობა ან საერთოდ გამოვტოვოთ MAXSIZE არგუმენტი.

დავალება 1. შექმენით Baza_2 მონაცემთა ბაზა, რომელიც შედგება მონაცემების ერთი ფაილისაგან და ტრანზაქციების ჟურნალის ერთი ფაილისაგან. მონაცემების ფაილის ზომა შეზღუდული არ არის.

```
USE Master
GO
IF DB_ID (N'Baza_2') IS NOT NULL
DROP DATABASE Baza_2;
GO
CREATE DATABASE Baza_2 ON
( NAME = Baza2,
FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Baza_2.mdf',
MAXSIZE = UNLIMITED )
```

```
LOG ON
( NAME = Baza2_log,
FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Baza_2_Log.ldf',
MAXSIZE = UNLIMITED )
GO
```

დავალება 2. შექმენით Baza2 მონაცემთა ბაზა. შექმნის დროს მიუთითეთ ფაილები. მოთხოვნას ექნება სახე:

```
USE Master
GO
IF DB_ID (N'Baza2') IS NOT NULL
DROP DATABASE Baza2;
GO
CREATE DATABASE Baza2
ON
( NAME = Baza2,
FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Baza2.mdf',
SIZE = 15, MAXSIZE = 36, FILEGROWTH = 3 )
LOG ON
( NAME = Baza2_log,
FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Baza2_Log.ldf',
SIZE = 3MB, MAXSIZE = 25MB, FILEGROWTH = 2MB )
```

ლაბორატორიული სამუშაო N3
ცხრილების შექმნა.

ცხრილის შესაქმნელად გამოიყენება CREATE TABLE ბრძანება. მისი სინტაქსია:

CREATE TABLE

[მონაცემთა_ბაზის_სახელი[სქემის_სახელი]ცხრილის_სახელი

({<სვეტის_განსაზღვრა>|გამოთვლადი_სვეტის_სახელი

AS გამოსახულება | <ცხრილის_შეზღუდვა> } [,...n])

განვიხილოთ არგუმენტების დანიშნულება.

- მონაცემთა_ბაზის_სახელი არის იმ ბაზის სახელი, რომელშიც ცხრილი იქმნება.
- სქემის_სახელი არის იმ სქემის სახელი, რომელსაც ცხრილი ეკუთვნის.
- ცხრილის_სახელი არის შესაქმნელი ცხრილის სახელი. ცხრილის სახელისა და სქემის სახელის კომბინაცია უნიკალური უნდა იყოს მონაცემთა ბაზის ფარგლებში.
- გამოსახულება არგუმენტის გამოთვლის შედეგად მიიღება მნიშვნელობა გამოთვლადი სვეტისათვის. დროს.

სვეტის_განსაზღვრა> კონსტრუქციის სინტაქსია:

<სვეტის_განსაზღვრა> ::= სვეტის_სახელი ტიპი

[[DEFAULT გამოსახულება]|[IDENTITY[(საწყისი_მნიშვნელობა, ნაზრდი)]]]

[<სვეტის_შეზღუდვა>] [,...n] [ROWGUIDCOL]

სადაც,

- ტიპი განსაზღვრავს სვეტში მოთავსებული მონაცემების ტიპს.
- DEFAULT განსაზღვრავს ავტომატურ მნიშვნელობას მოცემული სვეტისათვის.
- IDENTITY მიუთითებს, რომ შესაბამისი სვეტი იქნება სვეტი-მთვლელი. საწყისი_მნიშვნელობა არის მთვლელის საწყისი მნიშვნელობა, ნაზრდი კი - მთვლელის ნაზრდი.
- ROWGUIDCOL არგუმენტი მიუთითებს, რომ სვეტში მოთავსებული იქნება გლობალური იდენტიფიკატორი.

<სვეტის_შეზღუდვა> კონსტრუქციაში ეთითება მთლიანობის შეზღუდვები, რომლებიც განსაზღვრული იქნება სვეტისათვის.

დავალება 1. შექმენით ცხრილი, რომლის ერთ-ერთი სვეტი არ უშვებს NULL მნიშვნელობებს.

```
USE Baza_1
GO
-- თუ ცხრილი არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID(N'Cxrili_1',N'U') IS NOT NULL
DROP TABLE Cxrili_1;
GO
-- ცხრილის შექმნა
CREATE TABLE Cxrili_1
(
cxriliID int PRIMARY KEY CLUSTERED,
sveti1 int,
```

```
sveti2 nvarchar(20) NOT NULL,
sveti3 float,
sveti4 datetime
)
```

დავალება 2. შექმენით ცხრილი. მისი ერთ-ერთი სვეტის ბაზაზე შექმენით არაკლასტერული ინდექსი.

```
USE Baza_1
GO
-- თუ ცხრილი არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID(N'Cxrili_2',N'U') IS NOT NULL
DROP TABLE Cxrili_2;
GO
-- ცხრილის შექმნა
CREATE TABLE Cxrili_2
(
cxriliID int PRIMARY KEY CLUSTERED,
sveti1 int,
sveti2 nvarchar(20) NOT NULL UNIQUE NONCLUSTERED,
sveti3 float,
sveti4 datetime
)
```

დავალება 3. შექმენით ცხრილი. განსაზღვრეთ რეჟიმი, როცა მთავარი ცხრილიდან სტრიქონის წაშლისას დამოკიდებულ ცხრილში სტრიქონები წაიშლება.

```
USE Baza_1
GO
-- თუ ცხრილი არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID(N'Cxrili_4',N'U') IS NOT NULL
DROP TABLE Cxrili_4;
GO
-- ცხრილის შექმნა
CREATE TABLE Cxrili_4
(
cxriliID int PRIMARY KEY,
sveti1 int NULL FOREIGN KEY (sveti1) REFERENCES
Cxrili_1(cxriliID) ON DELETE CASCADE,
sveti2 nvarchar(20),
sveti3 float,
sveti4 datetime
)
```

ლაბორატორიული სამუშაო N4
ცხრილში მონაცემების ჩამატება.

INSERT ბრძანება იძლევა ცხრილში ერთი ან მეტი სტრიქონის ჩამატების საშუალებას. მისი სინტაქსია:

INSERT [INTO]{ ცხრილის_სახელი | წარმოდგენის_სახელი } { [(სვეტების_სია)] {VALUES ({DEFAULT |NULL| გამოსახულება})| მიღებული_ცხრილი } } | DEFAULT VALUES

სადაც,

- [INTO] არააუცილებელი არგუმენტია, რომელსაც მოსდევს იმ ცხრილის სახელი, რომელშიც მონაცემების ჩამატება ხდება;
- ცხრილის_სახელი იმ ცხრილის სახელია, რომელშიც სტრიქონების ჩამატება ხდება;
- წარმოდგენის_სახელი იმ წარმოდგენის სახელია, რომელშიც ხდება სტრიქონების ჩამატება.
- (სვეტების_სია) იმ სვეტების სახელების სიაა, რომლებშიც უნდა მოხდეს მონაცემების ჩამატება. სვეტების მნიშვნელობები ეთითება VALUES არგუმენტში. სვეტის მნიშვნელობა შეგვიძლია არ მივუთითოთ მხოლოდ იმ შემთხვევაში, თუ ამ სვეტისთვის განსაზღვრულია IDENTITY თვისება, NULL მნიშვნელობის შენახვის შესაძლებლობა ან timestap ტიპი.
- VALUES ({ DEFAULT | NULL | გამოსახულება }). განსაზღვრავს ცხრილში ჩასასმელ მონაცემებს. მისი არგუმენტების რაოდენობა უნდა ემთხვეოდეს ცხრილში ან სვეტების_სია არგუმენტში მითითებული სვეტების რაოდენობას. DEFAULT არგუმენტი მიუთითებს, რომ

მოხდება ავტომატურად განსაზღვრული მნიშვნელობების ჩასმა. თუ სვეტისთვის ავტომატური მნიშვნელობა განსაზღვრული არ არის და ნებადართულია NULL მნიშვნელობის შენახვა, მაშინ სვეტში NULL მნიშვნელობა მოთავსთავდება.

- მიღებული_ცხრილი არგუმენტი შეიძლება შეიცავდეს SELECT ბრძანებას, რომლის საშუალებითაც მოცემულ ცხრილში ჩასასმელ სტრიქონებს სხვა ცხრილიდან მივიღებთ. ორივე ცხრილს ერთნაირი სვეტები უნდა ჰქონდეს.
- DEFAULT VALUES არგუმენტის მითითების შემთხვევაში თითოეულ სვეტში მოთავსთდება ავტომატურად განსაზღვრული მნიშვნელობა.

დავალება 1. Personali ცხრილს დავუმატოთ სტრიქონი, რომლის ყველა სვეტში ავტომატურად განსაზღვრული მნიშვნელობა მოთავსთდება.

```
USE Shekveta
INSERT INTO Personali DEFAULT VALUES
SELECT * FROM Personali
```

დავალება 2. ცხრილს ჩაუმატეთ ორი სტრიქონი. ჩასამატებელი მნიშვნელობები სხვადასხვა მიმდევრობით შეიტანეთ.

```
USE Shekveta
SET DATEFORMAT dmy
```

INSERT INTO Personali (gvari, xelfasi, asaki, qalaqi, raioni, misamarti, staji, tarigi_dabadebis,

email, mobiluri, teleponi_samsaxuris, teleponi_saxlis, ganyofileba)

VALUES (N'სამხარაძე რომანი', 378, 37, N'თბილისი', N'', N'სტალინის ქ.5', 5, '20.02.1989',

N'romani@geo.net.ge', '899-345012', '23-83-39', '23-21-71', N'სამედიცინო')

INSERT INTO Personali (email, mobiluri, teleponi_samsaxuris, teleponi_saxlis, gvari, xelfasi, asaki,

qalaqi, raioni, misamarti, staji, tarigi_dabadebis, ganyofileba)

VALUES (N'archili@geo.net.ge', '899-012345', '30-80-09', '13-76-78',

N'სამხარაძე არჩილი', 378, 37, N'ქუთაისი', N'', N'წერეთლის ქ.5', 5, '10.02.1959', N'სასპორტო')

SELECT * FROM Personali

ლაბორატორიული სამუშაო N5
ცხრილიდან მონაცემების ამორჩევა. SELECT ბრძანება.

ცხრილებიდან მონაცემების ამოსარჩევად SELECT ბრძანება გამოიყენება. მისი სინტაქსია:

SELECT სია [INTO ახალი_ცხრილის_სახელი] FROM საწყისი_ცხრილის_სახელი [WHERE ძებნის_პირობა]

[GROUP BY დაჯგუფების_გამოსახულება]

[HAVING ძებნის_პირობა]

[ORDER BY დახარისხების_გამოსახულება [ASC | DESC]]

SELECT განყოფილებაში ეთითება გამოსატანი სვეტების სია. მისი სინტაქსია:

SELECT [ALL | DISTINCT] [TOP n [PERCENT]] <სია>

სადაც,

- ALL არგუმენტი მიუთითებს, რომ მოთხოვნის შესრულების შედეგში დასაშვებია გამეორებადი (ერთნაირი) სტრიქონების გამოტანა.
- DISTINCT არგუმენტი კრძალავს ერთნაირი სტრიქონების გამოტანას.
- TOP n [PERCENT] კონსტრუქციის გამოყენების შედეგად გაიცემა პირველი n სტრიქონი. თუ მითითებულია PERCENT, მაშინ n აღიქმება როგორც პროცენტი.
- <სია> კონსტრუქცია შეიცავს გამოსატანი სვეტების სახელებს.

დავალება 1. Personali და Shemkveti ცხრილებიდან გამოიტანეთ თანამშრომლებისა და შემკვეთების გვარები და იმ ქალაქების დასახელება, რომელშიც ისინი ცხოვრობენ.

```
USE Shekveta
SELECT Personali.gvari AS [თანამშრომლის გვარი],
Personali.qalaqi AS [თანამშრომლის ადგილმდებარეობა],
Shemkveti.gvari AS [შემკვეთის გვარი],
Shemkveti.qalaqi AS [შემკვეთის ადგილმდებარეობა]
FROM Personali, Shemkveti
WHERE Personali.personaliID = Shemkveti.personaliID
```

დავალება 2. გვაინტერესებს ინფორმაცია იმ თანამშრომლების შესახებ, რომელთა ასაკი მეტია 50-ზე ან ნაკლებია 30-ზე.

```
USE Shekveta
SELECT * FROM Personali
WHERE asaki > 50 OR asaki < 30
```

ლაბორატორიული სამუშაო N6
SELECT ბრძანება, FROM განყოფილება.

FROM სიტყვის შემდეგ ეთითება იმ ცხრილების ან წარმოდგენების სახელები, საიდანაც ხდება სვეტების ამორჩევა. მისი სინტაქსია:

FROM { ცხრილის_სახელი [[AS] ცხრილის_ფსევდონიმი]

|წარმოდგენის_სახელი [[AS] წარმოდგენის_ფსევდონიმი] |

<ბმული_ცხრილი> }

აქ <ბმული_ცხრილი> კონსტრუქცია გამოიყენება კავ-შირის უზრუნველსაყოფად რამდენიმე ცხრილს შორის. მისი სინტაქსია:

<ბმული_ცხრილი> ::=

<მარცხენა_ცხრილი> <ბმის_ტიპი> <მარჯვენა_ცხრილი> ON <ბმის_პირობა>

| <მარცხენა_ცხრილი> CROSS JOIN <მარჯვენა_ცხრილი> | <ბმული_ცხრილი>

აქ <მარცხენა_ცხრილი> კონსტრუქცია შეიცავს მთავარი ცხრილის სახელს, რომელსაც დაუკავშირდება სხვა ცხრილები. <ბმის_ტიპი> კონსტრუქცია განსაზღვრავს ორ ცხრილს შორის კავშირის ტიპს. მთავარი ცხრილის სახელი ეთითება <ბმის_ტიპი> კონსტრუქციის მარცხნივ, მარჯვნივ კი ეთითება დამოკიდებული ცხრილის სახელი.

<ბმის_ტიპი> კონსტრუქციის სინტაქსია:

<ბმის_ტიპი> ::= [INNER | { { LEFT | RIGHT | FULL } [OUTER] }] JOIN

სადაც,

- INNER კავშირის ტიპი ავტომატურად გამოიყენება. შედეგში ჩართული იქნება მარცხენა ცხრილის მხოლოდ ის სტრიქონები, რომლებისთვისაც არსებობენ სტრიქონები ბმულ (მარჯვენა) ცხრილში.
- LEFT [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარცხენა ცხრილის ყველა სტრიქონი, მიუხედავად იმისა, არსებობს თუ არა მათთვის შესაბამისი სტრიქონი მარჯვენა ცხრილში.
- RIGHT [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარჯვენა ცხრილის ყველა სტრიქონი მიუხედავად იმისა, აქვთ თუ არა მათ შესაბამისი სტრიქონები მარცხენა ცხრილში.
- FULL [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარცხენა და მარჯვენა ცხრილების ყველა სტრიქონი.
- JOIN არგუმენტის შემდეგ ეთითება მარჯვენა ცხრილი.
- ON <ბმის_პირობა> არის ორივე ცხრილის დაკავშირების ლოგიკური პირობა.
- CROSS JOIN საკვანძო სიტყვის გამოყენებისას სრულდება მარცხენა ცხრილის თითოეული სტრიქონის დაკავშირება მარჯვენა ცხრილის თითოეულ სტრიქონთან.

დავალება 1. Shemkveti და Xelshekruleba ცხრილებს შორის LEFT OUTER კავშირი დაამყარეთ.

```
USE Shekveta
SELECT Shemkveti.gvari, Shemkveti.qalaqi,
Xelshekruleba.gadasaxdeli_l, Xelshekruleba.vali_l
FROM Shemkveti LEFT OUTER JOIN Xelshekruleba
ON Shemkveti.ShemkvetiID = Xelshekruleba.ShemkvetiID
```

დავალება 2. 4.15. Shemkveti და Xelshekruleba ცხრილებს შორის RIGHT OUTER კავშირი დაამყარეთ.

```
USE Shekveta
SELECT Shemkveti.gvari, Shemkveti.qalaqi,
Xelshekruleba.gadasaxdeli_l, Xelshekruleba.vali_l
FROM Shemkveti RIGHT OUTER JOIN Xelshekruleba
ON Shemkveti.ShemkvetiID = Xelshekruleba.ShemkvetiID
```

დავალება 3. Shemkveti და Xelshekruleba ცხრილებს შორის FULL OUTER კავშირი დაამყარეთ.

```
USE Shekveta
SELECT Shemkveti.gvari, Shemkveti.qalaqi,
Xelshekruleba.gadasaxdeli_l, Xelshekruleba.vali_l
FROM Shemkveti FULL OUTER JOIN Xelshekruleba
ON Shemkveti.ShemkvetiID = Xelshekruleba.ShemkvetiID
```

ლაბორატორიული სამუშაო N7
ჩადგმული და ბმული მოთხოვნები.

ერთ მოთხოვნას შეუძლია მეორე მოთხოვნის მართვა. ეს მიიღწევა ჩადგმული მოთხოვნების გამოყენებით. ჩადგმული მოთხოვნა გვაქვს მაშინ, როცა ერთი მოთხოვნის შიგნით მეორეა მოთავსებული. შიდა მოთხოვნის მიერ გაცემული შედეგები მოწმდება გარე (ძირითადი) მოთხოვნის პირობაში, რომელიც განსაზღვრავს ეს შედეგები ჭეშმარიტია (true) თუ მცდარი (false).

იმისათვის, რომ გარე მოთხოვნა შესრულდეს საჭიროა, რომ შიდა მოთხოვნამ გასცეს ერთი ან მეტი ჩანაწერი. წინააღმდეგ შემთხვევაში, გარე მოთხოვნა არ შესრულდება.

შიდა მოთხოვნებში აკრძალულია SELECT * ტიპის ბრძანების გამოყენება. შიდა მოთხოვნებში დასაშვებია GROUP BY და HAVING საკვანძო სიტყვების გამოყენება.

ბმული მოთხოვნების გამოყენების დროს შიდა მოთხოვნა სრულდება გარე მოთხოვნაში მითითებული ცხრილის თითოეული სტრიქონისათვის. გარე მოთხოვნის სტრიქონს, რომლისთვისაც უნდა შესრულდეს შიდა მოთხოვნა, სტრიქონი-კანდიდატი ეწოდება.

ბმული მოთხოვნების შესრულების ალგორითმია:

1. სტრიქონი-კანდიდატის ამორჩევა.

2. შიდა მოთხოვნის შესრულება.

3. პირობის შეფასება გარე მოთხოვნაში შიდა მოთხოვნის შედეგების საფუძველზე. განისაზღვრება, აირჩევა თუ არა სტრიქონი-კანდიდატი გამოტანისათვის.

4. 1÷3 ბიჯები მეორდება ყველა სტრიქონისათვის.

დავალება 1. დავუშვათ ვიცით ფირმის დასახელება და არ ვიცით მისი კოდი. გვაინტერესებს ინფორმაცია მის მიერ გაფორმებული ხელშეკრულებების შესახებ.

SELECT-მოთხოვნას ექნება სახე:

```
SELECT * FROM Xelshekruleba
WHERE shemkvetiID IN
(
SELECT shemkvetiID FROM Shemkveti WHERE
firmis_dasaxeleba = N'ჯეონეთი'
)
```

დავალება 2. გვაინტერესებს ის თანამშრომლები, რომლებმაც ხელშეკრულება 2002 წლის 12 ოქტომბერს გააფორმეს.

SELECT-მოთხოვნას ექნება სახე:

```
USE Shekveta
SET DATEFORMAT DMY
SELECT * FROM Personali P WHERE '12.10.2004' IN
(
SELECT X.tarigi_dawyebis FROM Xelshekruleba X WHERE
X.personaliID = P.personaliID
)
```

დავალება 3. გვაინტერესებს ის თანამშრომლები, რომლებსაც სამზე ნაკლები შემკვეთი ჰყავს.

SELECT-მოთხოვნას ექნება სახე:

```
USE Shekveta
SELECT personaliID, gvari FROM Personali P WHERE 3 >
(
SELECT COUNT(*) FROM Xelshekruleba X WHERE
X.personaliID = P.personaliID GROUP BY X.personaliID
)
```

ლაბორატორიული სამუშაო N8
GROUP BY განყოფილება და აგრეგირების ფუნქციები.

GROUP BY საკვანძო სიტყვა საშუალებას გვაძლევს ცხრილის სტრიქონები გარკვეული კრიტერიუმის მიხედვით დავაჯგუფოთ. თითოეული ჯგუფისათვის შეიძლება აგრეგირების

ფუნქციების გამოყენება (ისინი ამ თავშია განხილული), რომლებიც შესრულდება ჯგუფის ყველა სტრიქონის მიმართ. GROUP BY საკვანძო სიტყვის სინტაქსია:

[GROUP BY [ALL] სვეტების_სია [,…]]

განვიხილოთ არგუმენტების დანიშნულება.

- სვეტების_სია არგუმენტში ეთითება იმ სვეტების სახელები, რომელთა მიხედვით უნდა შესრულდეს დაჯგუფება.

- ALL. თუ მოთხოვნაში განსაზღვრულია პირობა, რომელიც ზღუდავს ამოსარჩევი სტრიქონების დაჯგუფების დიაპაზონს, მაშინ GROUP BY განყოფილებაში ALL სიტყვის გამოყენება გამოიწვევს ყველა ჯგუფის გამოტანას. აგრეგირების ფუნქცია არ შესრულდება იმ ჯგუფებისათვის, რომლებიც არ აკმაყოფილებენ მოთხოვნაში მითითებულ პირობას.

აგრეგირების ფუნქციები მონაცემების სტატისტიკურ დამუშავებას ასრულებენ.

აგრეგირების ფუნქციების უმრავლესობა არ ითვალისწინებს NULL მნიშვნელობის შემცველ სვეტებს. ALL საკვანძო სიტყვა მიუთითებს, რომ აგრეგირება უნდა შეეხოს გამეორებად (ერთნაირ) სტრიქონებს. ეს არგუმენტი

ავტომატურად იგულისხმება. DISTINCT საკვანძო სიტყვა მიუთითებს, რომ აგრეგირება შეეხება მხოლოდ უნიკალურ სტრიქონებს.

დავალება 1. გვაინტერესებს შეკვეთის მინიმალური თანხა თითოეული თანამშრომლისათვის. მოთხოვნას ექნება სახე:

```
USE Shekveta
SELECT personaliID, MIN(gadasaxdeli_l) AS [მინიმალური გადასახდელი თანხა]
FROM Xelshekruleba
GROUP BY personaliID
```

დავალება 2. გვაინტერესებს თითოეულ განყოფილებაში მომუშავე თანამშრომლების საშუალო ასაკი და მათი რაოდენობა. მოთხოვნას ექნება სახე:

```
USE Shekveta
SELECT ganyofileba, AVG(asaki) AS [საშუალო ასაკი], [განყოფილება] = COUNT(*)
FROM Personali
GROUP BY ALL ganyofileba
```

ლაბორატორიული სამუშაო N9
HAVING, ORDER BY და UNION განყოფილებები.

ეს განყოფილება გამოიყენება ძებნის პირობის მისათითებლად ჯგუფისთვის. მისი სინტაქსია:

HAVING <ძებნის_პირობა>

თუ HAVING განყოფილება გამოიყენება GROUP BY განყოფილების გარეშე, მაშინ ის იმუშავებს WHERE განყოფილების მსგავსად. თუ HAVING განყოფილება გამოიყენება GROUP BY ALL კონსტრუქციასთან ერთად, მაშინ HAVING ფარავს ALL სიტყვის მოქმედებას.

ORDER BY განყოფილება გამოიყენება ცხრილის დასახარისხებლად ამა თუ იმ სვეტის მნიშვნელობების ზრდის ან კლების მიხედვით. მისი სინტაქსია:

ORDER BY { სვეტების_სია [ASC | DESC] } [,…]

სვეტების_სია არგუმენტი შეიცავს ერთი ან მეტი სვეტის სახელს. ASC საკვანძო სიტყვა გამოიყენება მითითებული სვეტის მონაცემების დასალაგებლად ზრდადობის მიხედევით, DESC სიტყვა კი - კლებადობის მიხედვით. თუ ეს არგუმენტები მითითებული არ არის, მაშინ დახარისხება ზრდადობის მიხედვით შესრულდება.

UNION ბრძანება აერთიანებს რამდენიმე მოთხოვნის შესრულების შედეგს. მისი სინტაქსია:

მოთხოვნა_1

UNION [ALL]

მოთხოვნა_2

[,...n]

მოთხოვნა გასცემს გასაერთიანებელ სტრიქონებს.

UNION ბრძანების გამოყენების დროს ორი წესი უნდა დავიცვათ:

- ყველა მოთხოვნაში სვეტების რაოდენობა და მიმდევრობა ერთნაირი უნდა იყოს.
- სვეტების ტიპები უნდა ემთხვეოდეს.

UNION ბრძანება ავტომატურად გამორიცხავს ერთნაირ სტრიქონებს.

დავალება 1. გვაინტერესებს თბილისელი თანამშრომლები და შემკვეთები. მოთხოვნას ექნება სახე:

```
SELECT personaliID, gvari FROM Personali WHERE qalaqi = N'თბილისი'
UNION
SELECT shemkvetiID, gvari FROM Shemkveti WHERE qalaqi = N'თბილისი'
```

დავალება 2. გვაინტერესებს ვალის მაქსიმალური და მინიმალური მნიშვნელობები.

```
USE Shekveta
SELECT personaliID, vali_l, shesruleba
FROM Xelshekruleba WHERE vali_l > 0
ORDER BY vali_l DESC
COMPUTE MAX(vali_l), MIN(vali_l)
```

დავალება 3. გვაინტერესებს იმ თანამშრომლების რაოდენობა, რომელთა სტაჟი ნაკლებია ბათუმელი თანამშრომლების საშუალო სტაჟზე. მოთხოვნას ექნება სახე:

```
USE Shekveta
SELECT staji, COUNT(DISTINCT PersonaliID) FROM Personali GROUP BY staji
HAVING staji <
(
SELECT AVG(staji) FROM Personali WHERE qalaqi = N'ბათუმი'
)
```

ლაბორატორიული სამუშაო N10
ცხრილის მონაცემების ცვლილება და ცხრილიდან მონაცემების წაშლა. UPDATE და DELETE ბრძანებები.

ცხრილებში მონაცემების შესაცვლელად UPDATE ბრძანება გამოიყენება. მისი სინტაქსია:

UPDATE { ცხრილის_სახელი | წარმოდგენის_სახელი }

SET { სვეტის_სახელი = { გამოსახულება | DEFAULT | NULL }

| @ცვლადის_სახელი = გამოსახულება

|@ცვლადის_სახელი=სვეტის_სახელი =გამოსახულება} [,...n]

{[FROM{<საწყისი_ცხრილი>}[,...]][WHERE<ძებნის_პირობა>] }

სადაც,

- ცხრილის_სახელი იმ ცხრილის სახელია, რომელშიც უნდა შესრულდეს მონაცემების ცვლილება.

- წარმოდგენის_სახელი იმ წარმოდგენის სახელია, რომელშიც უნდა შესრულდეს მონაცემების ცვლილება.

- SET საკვანძო სიტყვა იწყებს ბლოკს, რომელშიც მითითებულია შესაცვლელი სვეტების ან ცვლადების სია.

- სვეტის_სახელი = { გამოსახულება | DEFAULT | NULL }. თითოეული სვეტისათვის უნდა მიეთითოს მისთვის მისანიჭებელი მნიშვნელობა. DEFAULT საკვანძო სიტყვა მიუთითებს, რომ სვეტს უნდა მიენიჭოს ავტომატურად განსაზღვრული მნიშვნელობა. NULL საკვანძო სიტყვა მიუთითებს, რომ სვეტს უნდა მიენიჭოს NULL მნიშვნელობა.

გამოსახულება შეიძლება იყოს მუდმივა, ცვლადი ან გამოსახულება.

- @ცვლადის_სახელი = გამოსახულება არგუმენტი ცვლადს ანიჭებს გამოსახულების მნიშვნელობას.

- @ცვლადის_სახელი = სვეტის_სახელი = გამოსახულება კონსტრუქცია საშუალებას გვაძლევს შევათავსოთ ცვლადებისა და სვეტების სახელების გამოყენება.

ცვლილება ყოველთვის შეეხება ამორჩეულ სტრიქონებს. თუ ფილტრი არ არის მითითებული, მაშინ ცვლილება შეეხება ცხრილის ყველა სტრიქონის მითითებულ სვეტებს.

დავალება 1. Xelshekruleba ცხრილში გადასახდელი თანხა შეცვალეთ 8000-ით იმ ხელშეკრულებებისათვის, რომელთა ნომრებია 3 და 9. მოთხოვნას ექნება სახე:

```
USE Shekveta
UPDATE Xelshekruleba SET gadasaxdeli_l = 8000 WHERE ID = 3 OR ID = 9
SELECT * FROM Xelshekruleba
```

დავალება 2. გამოთვალეთ გადასახდელი თანხა და ვალი დოლარებში, ლარებში მათი მნიშვნელობების კურსზე გაყოფის გზით. მოთხოვნას ექნება სახე:

```
USE Shekveta
UPDATE Xelshekruleba SET gadasaxdeli_d = ROUND(gadasaxdeli_l / kursi, 2),
vali_d = ROUND(vali_l / kursi, 2)
SELECT * FROM Xelshekruleba
```

ლაბორატორიული სამუშაო N11
Transact SQL-ის იდენტიფიკატორები, ცვლადები.

იდენტიფიკატორები გამოიყენება კონკრეტულ ობიექტთან. არსებობს იდენტიფიკატორების ორი ტიპი: სტანდარტული იდენტიფიკატორები (regular identifiers), რომელთა განსაზღვრისას დაცულია იდენტიფიკატორების შექმნის წესები, და შეზღუდული იდენტიფიკატორები (delimited identifiers), რომელთა განსაზღვრისას არ არის დაცული იდენტიფიკატორების შექმნის წესები. ასეთი იდენტიფიკატორები უნდა მოვათავსოთ კვადრატულ ფრჩხილებში ([ ]) ან ბრჭყალებში (“ “).

სტანდარტული იდენტიფიკატორის განსაზღვრისას უნდა დავიცვათ შემდეგი წესები:

იდენტიფიკატორის პირველი სიმბოლო უნდა შეესაბამებოდეს Unicode Standard 2.0 სტანდარტს. ამ სტანდარტის მიხედვით ეს სიმბოლო უნდა იყოს ნებისმიერი ლათინური სიმბოლო a-დან z-მდე, აგრეთვე, ეროვნული ანბანის სიმბოლო; შეიძლება იყოს ხაზგასმის სიმბოლო, @ ან # სიმბოლო. იდენტიფიკატორი, რომელიც # სიმბოლოთი იწყება, აღნიშნავს ლოკალურ დროებით ობიექტს, იდენტიფიკატორი, რომელიც ## სიმბოლოებით იწყება, აღნიშნავს გლობალურ დროებით ობიექტს. იდენტიფიკატორი, რომელიც @ სიმბოლოთი იწყება, აღნიშნავს ცვლადს.

იდენტიფიკატორის მომდევნო სიმბოლოები შეიძლება იყოს:

- Unicode Standard 2.0 სტანდარტით განსაზღვრული სიმბოლოები.
- ათობითი ციფრები.

- @, $, _, # სიმბოლოები.

იდენტიფიკატორის შიგნით დაუშვებელია სპეციალური სიმბოლოების გამოყენება: ~, !, %, ^, &, - , (,), {, }, —, ., \, = და ინტერვალი. იდენტიფიკატორი არ უნდა იყოს დარეზერვებული სიტყვა და უნდა იყოს უნიკალური. სახლების განსხვავება რეგისტრის მიხედვით არ ხდება.

ზოგიერთი შეზღუდვის გვერდის ასავლელად შეგვიძლია შეზღუდული იდენტიფიკატორები გამოვიყენოთ. ასეთ შემთხვევაში, ობიექტის სახელში დასაშვებია ინტერვალები და სპეციალური სიმბოლოები, აგრეთვე, დარეზერვებული სიტყვები.

ნებისმიერი ობიექტი გარკვეული მომხმარებლის მიერ იქმნება და ამა თუ იმ მონაცემთა ბაზას ეკუთვნის. თავის მხრივ, მონაცემთა ბაზა მოთავსებულია კონკრეტულ სერვერზე. სერვერის, მონაცემთა ბაზის, სქემისა და ობიექტის სახელის საფუძველზე იქმნება ობიექტის სრული სახელი (complete name) ან სრულად განსაზღვრული სახელი (full qualified name), რომელსაც შემდეგი სინტაქსი აქვს:

[[[სერვერის_სახელი.][მონაცემთა_ბაზის_სახელი].] [სქემის_სახელი].] ობიექტის_სახელი

როგორც სინტაქსიდან ჩანს, აუცილებელია მხოლოდ ობიექტის სახელის მითითება.

იმისათვის, რომ მივმართოთ ცხრილის ან წარმოდგენის რომელიმე სვეტს, აუცილებელია სრულ სახელში სვეტის სახელის დამატება:

[[[სერვერის_სახელი.] [მონაცემთა_ბაზის_სახელი].] [სქემის_სახელი].] ობიექტის_სახელი. სვეტის_სახელი

შემზღუდველების (“ “და [ ] სიმბოლოები) გამოყენების წესები დამოკიდებულია QUOTED_IDENTIFIER პარამეტრის მნიშვნელობაზე. მას მნიშვნელობა შეგვიძლია SET ბრძანებით მივანიჭოთ. მისი სინტაქსია:

SET QUOTED_IDENTIFIER { ON | OFF }

თუ QUOTED_IDENTIFIER პარამეტრი იმყოფება ON მდგომარეობაში, მაშინ შემზღუდველების გამოყენება შემდეგნაირად ხდება:

ბრჭყალები (“ “) და კვადრატული ფრჩხილები ([ ]) გამოიყენება იდენტიფიკატორების შეზღუდვისათვის. ბრჭყალები (“ “) არ გამოიყენება სიმბოლური სტრიქონებისათვის.

სიმბოლური სტრიქონები უნდა მოთავსთდეს აპოსტროფებში (‘ ’). იდენტიფიკატორის შეზღუდვისათვის აპოსტროფების გამოყენება არ შეიძლება. თუ სიმბოლური სტრიქონი აპოსტროფს შეიცავს, მაშინ ამ აპოსტროფის გვერდით მეორე აპოსტროფი უნდა მოვათავსოთ.

ცვლადის გამოყენებამდე საჭიროა მისი გამოცხადება DECLARE ბრძანების საშუალებით მისი სინტაქსია:

DECLARE @ლოკალური_ცვლადი მონაცემთა_ტიპი [,...n]

ცვლადის გამოცხადებისას ეთითება მისი სახელი, რომელიც აუცილებლად უნდა იწყებოდეს @ სიმბოლოთი. სახელის მომდევნო სიმბოლოები უნდა აკმაყოფილებდეს ობიექტების სახელდების წესებს. სახელის შემდეგ ეთითება ცვლადის ტიპი.

@ სიმბოლო ლოკალური ცვლადების სახელების გარდა გამოიყენება, აგრეთვე, ფუნქციებისა და შენახული პროცედურების არგუმენტების სახელების განსაზღვრისათვის.

სინტაქსის [,...n] ნაწილი მიუთითებს, რომ ერთი DECLARE ბრძანებით შეიძლებარამდენიმე ცვლადის გამოცხადება, რომელთა სახელები ერთმანეთისაგან მძიმეებით იქნება გამოყოფილი.

დავალება 1. შეამოწმეთ, მითითებული თანამშრომლის ხელფასი მეტია თუ არა 1000 ლარზე. მოთხოვნას ექნება სახე:

```
USE Shekveta
DECLARE @xelfasi float, @msg nvarchar(50)
SELECT @xelfasi = xelfasi FROM Personali WHERE gvari =
N'სამხარაძე საბა'
IF @xelfasi > 1000
SET @msg = N'ხელფასი მეტია 1000-ზე'
ELSE
SET @msg = N'ხელფასი ნაკლებია 1000-ზე'
SELECT @msg
```

დავალება 2. თანამშრომლების ხელფასი გაზარდეთ მანამ, სანამ მინიმალური ხელფასი 1000-ს არ გადააჭარბებს. მოთხოვნას ექნება სახე:

```
USE Shekveta
DECLARE @ricxvi_1 float
SET @ricxvi_1 = 0
WHILE @ricxvi_1 <= 1000
BEGIN
UPDATE Personali_1 SET xelfasi = xelfasi + 50
SELECT @ricxvi_1 = MIN(xelfasi) FROM Personali_1
END
SELECT gvari, xelfasi FROM Personali_1
```

ლაბორატორიული სამუშაო N12
მმართველი კონსტრუქციები: END...BEGIN, IF...ELSE, CASE...END

BEGIN...END კონსტრუქცია იძლევა ერთ ბლოკში ორი და მეტი ბრძანების დაჯგუფების საშუალებას. BEGIN საკვანძო სიტყვა იწყებს ბლოკს, END კი - ამთავრებს. ასეთი დაჯგუფება შეგვიძლია გამოვიყენოთ განშტოებებში, პირობისა და ციკლის კონსტრუქციებში. ბლოკი შეიძლება იყოს ჩადგმული. ჩადგმულობის დონე შეზღუდული არ არის, თუმცა პრაქტიკულად, ის 5-7-ს არ აღემატება.

IF...ELSE კონსტრუქცია საშუალებას გვაძლევს ბრძანება ან ბლოკი შევასრულოთ მხოლოდ მითითებული ლოგიკური პირობის შესრულების შემდეგ. მისი სინტაქსია:

IF ლოგიკური_გამოსახულება
{ sql_ბრძანება | ბრძანებების_ბლოკი }
[ELSE
{ sql_ბრძანება | ბრძანებების_ბლოკი }]

ლოგიკური_გამოსახულება არის ლოგიკური პირობა, რომელიც იღებს true (ჭეშმარიტი) ან false (მცდარი) მნიშვნელობას. თუ ის იღებს true მნიშვნელობას, მაშინ შესრულდება პირველი sql_ბრძანება ან ბრძანებების_ბლოკი. თუ პირობა იღებს false მნიშვნელობას, მაშინ შესრულდება ELSE სიტყვის შემდეგ მოთავსებული sql_ბრძანება ან ბრძანებების_ბლოკი.

CASE END არის მრავალგანშტოებიანი კონსტრუქცია. მისი სინტაქსია:

CASE case_გამოსახულება

WHEN{when_გამოსახულება|ლოგიკური_გამოსახულება} [,...n]

THEN then_გამოსახულება [,...n]

[ELSE else_გამოსახულება]

END

case_გამოსახულება შეიძლება შეიცავდეს ცვლადის სახელს ან ფუნქციას. when_გამოსახულება იღებს case_გამოსახულება არგუმენტის მნიშვნელობებიდან ერთ-ერთს. თუ ამ ორი არგუმენტის მნიშვნელობა ერთმანეთს დაემთხვევა, მაშინ გაიცემა then_გამოსახულება არგუმენტის მნიშვნელობა. შეიძლება მიეთითოს რამდენიმე WHEN...THEN სტრიქონი, რომლებიც შეიცავენ case_გამოსახულება არგუმენტის შესაძლო მნიშვნელობებს. თუ საწყისი მნიშვნელობა არ ემთხვევა when_გამოსახულება არგუმენტის არც ერთ მნიშვნელობას, მაშინ გაიცემა else_გამოსახულება არგუმენტის მნიშვნელობა.

დავალება 1. თანამშრომლების ხელფასი გაზარდეთ მანამ, სანამ მინიმალური ხელფასი 1000-ს არ გადააჭარბებს. მოთხოვნას ექნება სახე:

```
USE Shekveta
DECLARE @ricxvi_1 float
SET @ricxvi_1 = 0
WHILE @ricxvi_1 <= 1000
BEGIN
UPDATE Personali_1 SET xelfasi = xelfasi + 50
```

```
SELECT @ricxvi_1 = MIN(xelfasi) FROM Personali_1
END
```

დავალება 2. განვსაზღვროთ ამჟამად თვის პირველი ნახევარია, თუ მეორე. მოთხოვნას ექნება სახე:

```
DECLARE @@striqoni nvarchar (20)
IF DAY(GETDATE()) < 15 SET @@striqoni = N'პირველი'
ELSE SET @@striqoni = N'მეორე'
SELECT N'ახლა არის თვის ' + RTRIM(@@striqoni) + N' ნახევარი'
```

დავალება 3. შევადგინოთ პროგრამა, რომელიც გასცემს იმ თანამშრომლების გვარებს, რომელთა იდენტიფიკატორია 1’3. მოთხოვნას ექნება სახე:

```
USE Shekveta
GO
SELECT PersonaliID,
CASE PersonaliID
WHEN 1 THEN N'სამხარაძე საბა'
WHEN 2 THEN N'სამხარაძე ანა'
WHEN 3 THEN N'გაჩეჩილაძე ლია'
ELSE N'უცნობი'
END
FROM Personali
```

ლაბორატორიული სამუშაო N13
მმართველი კონსტრუქციები: WHILE...BREAK&CONTINUE

WHILE...BREAK & CONTINUE კონსტრუქცია გამოიყენება ციკლების ორგანიზებისათვის. მისი სინტაქსია:

```
WHILE ლოგიკური_გამოსახულება
{ sql_ბრძანება | ბრძანებების_ბლოკი }
[ BREAK ]
{ sql_ბრძანება | ბრძანებების_ბლოკი }
[ CONTINUE ]
{ sql_ბრძანება | ბრძანებების_ბლოკი }
```

თუ ლოგიკური_გამოსახულება იღებს true მნიშვნელობას, მაშინ შესრულდება ციკლის ტანი (კოდი). ციკლის შესრულება შეწყდება, როგორც კი ლოგიკური-გამოსახულება false მნიშვნელობას მიიღებს.

BREAK ბრძანების შესრულება იწვევს ციკლის შესრულების იძულებით შეწყვეტას და ციკლიდან გამოსვლას. CONTINUE ბრძანების შესრულება იწვევს ლოგიკური პირობის შემოწმებაზე გადასვლას. ამ ბრძანების შემდეგ მოთავსებული ბრძანებები არ შესრულდება.

დავალება 1. გამოვთვალოთ 1-დან 10-მდე რიცხვების კვადრატი.

```
DECLARE @ricxvi_1 int
SET @ricxvi_1 = 1
WHILE 10 = 10
BEGIN
```

```
    PRINT STR(@ricxvi_1) + N'-ის კვადრატი ='
+STR(SQUARE(@ricxvi_1))
    SET @ricxvi_1 = @ricxvi_1 + 1
    IF @ricxvi_1 <= 10 CONTINUE
    BREAK
    END
```

დავალება 2. თუ თანამშრომლების საშუალო ხელფასი ნაკლებია 600 ლარზე, მაშინ WHILE ციკლი აორმაგებს თანამშრომლების ხელფასს და შემდეგ ირჩევს მაქსიმალურ ხელფასს. თუ მაქსიმალური ხელფასი ნაკლებია ან ტოლი 1000 ლარის, მაშინ ისევ შესრულდება ციკლის კოდი (CONTINUE), წინააღმდეგ შემთხვევაში ციკლის შესრულება წყდება (BREAK).

```
    USE Shekveta;
    GO
    WHILE ( SELECT AVG(xelfasi) FROM Personali_1 ) < 600
    BEGIN
    UPDATE Personali_1 SET xelfasi = xelfasi * 2
    SELECT MAX(xelfasi) FROM Personali_1
    IF ( SELECT MAX(xelfasi) FROM Personali_1 ) > 1000
    BREAK
    ELSE
    CONTINUE
    END
```

ლაბორატორიული სამუშაო N14
ლოგიკის ოპერატორები: ALL, IN, LIKE

ALL ოპერატორი სკალარულ სიდიდეს ადარებს ქვემოთხოვნის მიერ დაბრუნებულ თითოეულ მნიშვნელობას. თუ ლოგიკური პირობა სრულდება ყველა დაბრუნებული მნიშვნელობისათვის, მაშინ პირობა ჩაითვლება შესრულებულად. მისი სინტაქსია:

გამოსახულება { = | <> | != | > | >= | !> | < | <= | !< } ALL (ქვემოთხოვნა) სადაც,

- გამოსახულება - ნებისმიერი დასაშვები გამოსახულებაა.

- ქვემოთხოვნა - ქვემოთხოვნაა, რომელიც გასცემს ერთი სვეტის მნიშვნელობებს. მნიშვნელობების ტიპი უნდა ემთხვეოდეს გამოსახულების ტიპს.

IN ოპერატორის საშუალებით შეგვიძლია შევამოწმოთ ემთხვევა თუ არა გამოსახულების მნიშვნელობა ქვე-მოთხოვნის მიერ დაბრუნებული მნიშვნელობებიდან ან ჩამოთვლილი სიდიდეებიდან ერთ-ერთს. მისი სინტაქსია:

გამოსახულება [NOT] IN (ქვემოთხოვნა | გამოსახულება [,...n])

LIKE ოპერატორის საშუალებით შესაძლებელია გამოსახულების შედარება მოცემულ შაბლონთან. LIKE ოპერატორის გამოყენება მოხერხებულია მაშინ, როცა ზუსტად არ ვიცით სტრიქონის მნიშვნელობა. მისი სინტაქსია:

გამოსახულება [NOT] LIKE შაბლონი [ESCAPE escape_სიმბოლო]

სადაც, გამოსახულება არგუმენტი განსაზღვრავს შესამოწმებელ გამოსახულებას. შაბლონი შეიცავს სიმბოლო_შემცვლელებს.

დავალება 1. გვაინტერესებს ინფორმაცია სასოფლო და სავაჭრო განყოფილების თანამშრომლების შესახებ.

```
USE Shekveta
SELECT * FROM Personali WHERE ganyofileba IN
(N'სასოფლო', N'სავაჭრო')
```

დავალება 2. 5.7. გვაინტერესებს ის ფირმები, რომელთა სახელები 'G' ასოთი იწყება.

```
USE Shekveta
SELECT firmis_dasaxeleba, gvari, qalaqi FROM Shemkveti
WHERE firmis_dasaxeleba LIKE N'G%'
```

დავალება 3. გვაინტერესებს ყველა ხელშეკრულება შესრულდა თუ არა. თუ არა, მაშინ რომელი ხელშეკრულებები არ შესრულდა.

```
USE Shekveta
IF N'კი' = ALL ( SELECT shesruleba FROM Xelshekruleba )
SELECT N'ყველა ხელშეკრულება შესრულდა'
```

```
ELSE
BEGIN
SET NOCOUNT ON
SELECT Xelshekruleba.shemkvetiID AS [არ შესრულდა შემდეგი ხელშეკრულებები],
Personali.gvari AS [შემსრულებელი], Shemkveti.gvari AS [შემკვეთი],
Shemkveti.firmis_dasaxeleba AS [ფირმის დასახელება]
FROM Xelshekruleba, Personali, Shemkveti
WHERE shesruleba = N'არა' AND
Personali.personaliID = XelShekruleba.personaliID
AND Shemkveti.shemkvetiID = Xelshekruleba.shemkvetiID
ORDER BY Xelshekruleba.shemkvetiID
END
```

ლაბორატორიული სამუშაო N15
მონაცემების მასობრივი გადაწერა

მონაცემების იმპორტისა და ექსპორტისათვის BCP (Bulk Copy Program) უტილიტი გამოიყენება. ის ასრულებს მონაცემების გაცვლას სერვერსა და ტექსტურ ფაილებს შორის. BCP უტილიტის მუშაობის შედეგები არ აისახება ტრანზაქციების ჟურნალში. ეს იმას ნიშნავს, რომ თუ გადაწერის ოპერაცია წარუმატებლად დამთავრდა, მაშინ საწყისი მონაცემების აღდგენას ვერ შევძლებთ. BCP უტილიტი მუშაობს ავტომატურ და ინტერაქტიურ რეჟიმში. თუ ფორმატირების გასაღებები მითითებულია საბრძანებო სტრიქონში ან ფორმატირების ფაილში, მაშინ BCP უტილიტი ავტომატურ რეჟიმში იმუშავებს. თუ ფორმატირების პარამეტრები მითითებული არ არის, მაშინ გაიცემა შესაბამისი შეკითხვები. BCP ბრძანების შესასრულებლად ის უნდა შევიტანოთ Command Prompt ფანჯრის (Start All Programs Accessories Command Prompt) საბრძანებო სტრიქონში და დავაჭიროთ Enter კლავიშს.

BCP ბრძანების სინტაქსია:

bcp { [[მონაცემთა_ბაზის_სახელი.] [მფლობელი] .]
{ცხრილის_სახელი | წარმოდგენის_სახელი } | 'მოთხოვნა' }
{ in | out | queryout | format } მონაცემების_ფაილი
[-e შეცდომების_ფაილი]
[-F პირველი_სტრიქონი] [-L უკანასკნელი_სტრიქონი]
[-c] [-w] [-N]
[-q] [-t სვეტების_გამყოფი] [-r სტრიქონების_გამყოფი]

[i შესასვლელი_ფაილი] [-o გამოსასვლელი_ფაილი] [-a პაკეტის_ზომა]

[-Sსერვერის_სახელი] [-U სააღრიცხვო_ჩანაწერის_სახელი] [-P პაროლი]

[-T] [-v] [-R] [-k] [-E]

სადაც,

- მონაცემთა_ბაზის_სახელი, იმ მონაცემთა ბაზის სახელია, რომელშიც ის ცხრილი ან წარმოდგენა ინახება, საიდანაც უნდა შესრულედეს ასლის აღება. თუ სახელი მითითებული არ არის, მაშინ გამოყენებული ის მონაცემთა ბაზა, რომელიც ეკუთვნის იმ მომხმარებელს, რომლის სააღრიცხვო ჩანაწერის ქვეშ მუშაობს ეს უტილიტი.
- მფლობელი. იმ ცხრილის ან წარმოდგენის მფლობელის სახელია, რომელთანაც სრულდება მუშაობა. თუ ის მითითებული არ არის, მაშინ ცხრილი ან წარმოდგენა ეკუთვნის იმ ფლობელს, რომლის სააღრიცხვო ჩანაწერის ქვეშ მუშაობს ეს უტილიტი.
- ცხრილის_სახელი | წარმოდგენის_სახელი } | 'მოთხოვნა'. ცხრილის ან წარმოდგენის სახელია, რომელთანაც შესრულდება მუშაობა. მოთხოვნა არგუმენტი შეიცავს SELECT-მოთხოვნას.

დავალება 1. Shekveta მონაცემთა ბაზის Personali ცხრილიდან ტექსტურ ფაილში გადავწეროთ ყველა სტრიქონი. გამოვიყენოთ SELECT მოთხოვნა და Windows NT-ის სააღრიცხვო ჩანაწერი.

bcp "SELECT * FROM Shekveta.dbo.Personali"

queryout "C:\Documents and Settings\romani\My Documents\Romani\Books\DB\
DB_Wignis_Programebi\Tavi 6\BCP\Personali4.txt"
-w –T

დავალება 2. Shekveta მონაცემთა ბაზის Personali ცხრილიდან ტექსტურ ფაილში გადავწეროთ ყველა სტრიქონი. მონაცემები უნდა დამუშავდეს Unicode ფორმატში. გამოვიყენოთ Windows NT-ის სააღრიცხვო ჩანაწერი.

bcp "Shekveta.dbo.Personali"
out "C:\Documents and Settings\romani\My Documents\Romani\Books\DB\DB_Wignis_Programebi\
Tavi 6\BCP\Personali1.txt"
-w –T

ლიტერატურა

1. SQL სერვერი. რომან სამხარაძე. საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი‖ , თბილისი.2009. ISBN 978-9941-14-190-4 . http://www.gtu.ge/publishinghouse

2. V i s u a l C#.N E T. რომან სამხარაძე. საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“, თბილისი.2009. ISBN978-9941-14-593. http://www.gtu.ge/publishinghouse

რედაქტორი
ტექნიკური რედაქტორი
კორექტორი
კომპიუტერული უზრუნველყოფა ნ. ჯოჯუასი

წარმოებას გადაეცა . ხელმოწერილია დასაბეჭდად . ქაღალდის ზომა პირობითი ნაბეჭდი თაბახი . სააღრიცხვო-საგამომცემლო თაბახი . ტირაჟი 20 ეგზ.

გამომცემლობა “ტექნიკური უნივერსიტეტი”, თბილისი კოსტავას 77