საქართველოს ტექნიკური უნივერსიტეტი

ბადრი მეფარიშვილი, გულნარა ჯანელიძე

Web დანართების პროგრამული რეალიზაცია MySQL-ის გამოყენებით

(ლაბორატორიული პრაქტიკუმი)

რეკომენდებულია სტუ-ის
სარედაქციო- საგამომცემლო საბჭოს
მიერ. 28.10.2015, ოქმი №2

თბილისი
2016

განხილულია მონაცემთა ბაზების მართვის სისტემა MySQL გარემოში ლაბორატორიული სამუშაოების შესრულების აღწერილობა საგნისათვის „Web დანართების პროგრამული რეალიზაცია MySQL-ის გამოყენებით“.

განკუთვნილია ინფორმატიკის სფეროს სტუდენტებისთვის.

რეცენზენტები:

პროფესორი გია სურგულაძე

პროფესორი თინათინ კაიშაური

ISBN 978-9941-20-618-4

http//www.gtu.ge

ლაბორატორიული სამუშაოები	
N	თემის დასახელება და შინაარსი
1	MySQL: ინსტალაცია და კონფიგურირება
2	მუშაობა Tools-თან
3	მონაცემთა განსაზღვრების ენა MySQL -ში
4	მონაცემთა მანიპულირების ენა DML
5	მრავალცხრილიანი მოთხოვნები. JOIN ბრძანება
6	წარმოდგენები
7	მუშაობა dbForge Studio for MySQL-ში
8	პროგრამირება MySQL-ში
9	ბლოკირებები და ტრანზაკციები
10	მონაცემთა ბაზების სარეზერვო ასლების შექმნა-აღდგენა
11	MySQL მონაცემთა ბაზების იმპორტი და ექსპორტი
12	MySQL: ადმინისტრირება და უსაფრთხოების სისტემა
13	მონაცემთა ბაზის პროექტის შემუშავება
14	MySQL მონაცემთა ბაზებთან მუშაობა PHP საშუალებით
15	მუშაობა phpMyAdmin-თან

შინაარსი

ზოგადი მითითებები

ლაბორატორიული სამუშაოების შესრულების თანამიმდევრობა

ლაბორატორიული ციკლის მიზანია წარმოადგენს მონაცემთა ბაზების აგება MySQL-ის, გრაფიკული უტილიტების - GUI Tools-ის, Workbench-ის, dbForge for MySQL-ის და, აგრეთვე, Web დანართების შემუშავება phpMyAdmin-ის გამოყენებით.

ლაბორატორიული სამუშაოების ციკლის დაწყებამდე, საგნის სილაბუსის საშუალებით სტუდენტი ეცნობა სამუშაოთა ჩამონათვალს და ძირითად მიზანს. ყოველ სამუშაო ადგილზე იქმნება მოცემული ჯგუფის საქაღალდე და მასში სტუდენტის ინდივიდუალური საქაღალდე, სადაც განთავსდება შესრულებული ლაბორატორიული სამუშაოების ოქმების ფაილები (doc გაფართოებით) ანუ ელექტრონული ოქმები, რომლებიც სემესტრის ბოლოს E-mail-ით გადაეგზავნება ლაბორატორიული მეცადინეობების წამყვან მასწავლებელს და შემდგომი არქივაციისათვის გადაიტანება CD-ზე.

წინასწარი მომზადება

თითოეული ლაბორატორიული სამუშაოს აღწერა შედგება შემდეგი ნაწილებისაგან:

- სამუშაოს მიზანი;
- დავალება;
- დავალების შესრულების თანამიმდევრობა.

შესასრულებელი დავალება

ყოველი ლაბორატორიული სამუშაოს შესასრულებელი დავალება ფორმულირებულია ნაწილში „დავალება“. სტუდენტი წინასწარ, დამოუკიდებლად უნდა გაეცნოს თავის დავალებას და თუ მას გაუჩნდება შეკითხვები დავალებასთან დაკავშირებით, კითხვები უნდა დასვას სამუშაოს დაწყებამდე.

დავალების შესრულება

დავალების შესრულებამდე მოწმდება სტუდენტის მიერ დავალების წინასწარი მომზადების ხარისხი და გადაწყდება შესრულებისათვის მისი კომპიუტერთან დაშვების საკითხი. შესრულება იწყება ლაბორატორიული სამუშაოს აღწერაში წარმოდგენილ პრაქტიკულ მოქმედებათა თანამიმდევრობის საფუძველზე. მიღებული შედეგების ეკრანზე წარმოდგენის შემდეგ სტუდენტი ქმნის მათ ასლებს ტექსტური ფაილების სახით და ინახავს ინდივიდუალურ საქაღალდეში მოცემული თარიღის მითითებით.

ლაბორატორიული სამუშაოს ანგარიში

ლაბორატორიული სამუშაოს ელექტრონული ანგარიში უნდა შეიცავდეს შემდეგ პუნქტებს:

- ლაბორატორიული სამუშაოს თემა;
- შესასრულებელი ინდივიდუალური დავალება;
- ტექსტურ ფაილში გადატანილი მიღებული შედეგები, ნიმუშების ასლები;
- დასკვნა.

ლაბორატორიული სამუშაო 1
MySQL: ინსტალაცია და კონფიგურირება

სამუშაოს მიზანი:

1. MySQL - ის ინსტალაციის შესწავლა;

2. MySQL-ის კონფიგურაციის შესწავლა, როგორც სერვერის, ისე კლიენტის მხარეზე;

3. MySQL GUI Tools პაკეტის დაყენება.

თეორიული მასალა: MySQL-ის ძირითადი კომპონენტები და მათი შესაძლებლობანი.

1. MySQL-ის ინსტალაცია

MySQL -ის ჩატვირთვა

MySQL-ის Windows-ის საინსტალაციო ვერსია შეიძლება ვსუროთ შემდეგი ვებ-გვერდის http://dev.mysql.com/downloads/mysql/5.0.html ერთ-ერთ განყოფილებაში:

• Windows downloads – აქ არის მიმართვები MySQL დისტრიბუტივებზე 32- თანრიგიანი ოპერაციული სისტემებისათვის: Windows Vista/Server 2003/XP/Millennium Edition/2000/98/95;

• Windows x64 downloads – აქ განთავსებულია მიმართვები MySQL დისტრიბუტივებზე 64-თანრიგიანი ოპერაციული სისტემებისათვის: Windows Vista x64/Server 2003 x64/ XP x64.

ყოველ ამ განყოფილებაში წარმოდგენილია დისტრიბუტივის სამი ვარიანტი:

• Essentials – საბაზო ვარიანტი ოპციური ვარიანტების გარეშე. შეიცავს აწყობის ოსტატს (Configuration Wizard);

• ZIP/Setup.EXE – სრული ვარიანტი ოპციური უტილიტების ჩართვით;

• Without installer – სრული ვარიანტი აწყობის ოსტატის გარეშე.

რეკომენდებულია პირველი და მეორე ვარიანტების გამოყენება. თუ ამჯობინებთ მუშაობას გრაფიკულ ინტერფეისში და არა ბრძანებით სტრიქონში, მაშინ საჭირო იქნება გრაფიკული უტილიტების MySQL GUI Tools პაკეტის გადმოქაჩვა (http://dev.mysql.com/downloads/gui-tools/5.0.html), რომელიც შეიცავს სამ ქვეპროგრამას:

• MySQL Administrator – ადმინისტრირების, კონფიგურირების, მონიტორინგის, MySQL სერვერის გაშვება/შეჩერების, მომხმარებელთა მართვისა და შეერთებების ინსტრუმენტი;

• MySQL Query Browser – გრაფიკულ გარემოში მოთხოვნათა შექმნის, შესრულებისა და ოპტიმიზაციის ინსტრუმენტი;

• MySQL Migration Toolkit – სხვა რელაციური მონაცემთა ბაზებიდან (Oracle, Microsoft SQL Server, Microsoft Access, Sybase და სხვ.) MySQL-ში მონაცემთა გადატანის ინსტრუმენტი.

MySQL სერვერის დაყენება

MySQL-ის Windows-ის საინსტალაციო ვერსიის დაყენებისათვის საჭიროა შემდეგი მოქმედებების შესრულება.

1. ინსტალაციის ოსტატის (Setup Wizard) გაშვება:

• თუ თქვენ გადმოქაჩეთ MySQL დისტრიბუტივის საბაზისო ვარიანტი, მაშინ ორჯერ დააწკაპუნეთ ფაილზე mysql-essential-5.0.xxx-win32.msi.;

• თუ თქვენ გადმოქაჩეთ MySQL დისტრიბუტივის სრული ვარიანტი, მაშინ არქივიდან ამოიღეთ Setup.exe ფაილი და გაუშვით შესრულებაზე ორჯერ დაწკაპუნებით.

2. ინსტალაციის ოსტატის საწყის ფანჯარაში (სურ. 1.1) ვაჭერთ Next ღილაკს.

სურ. 1.1.

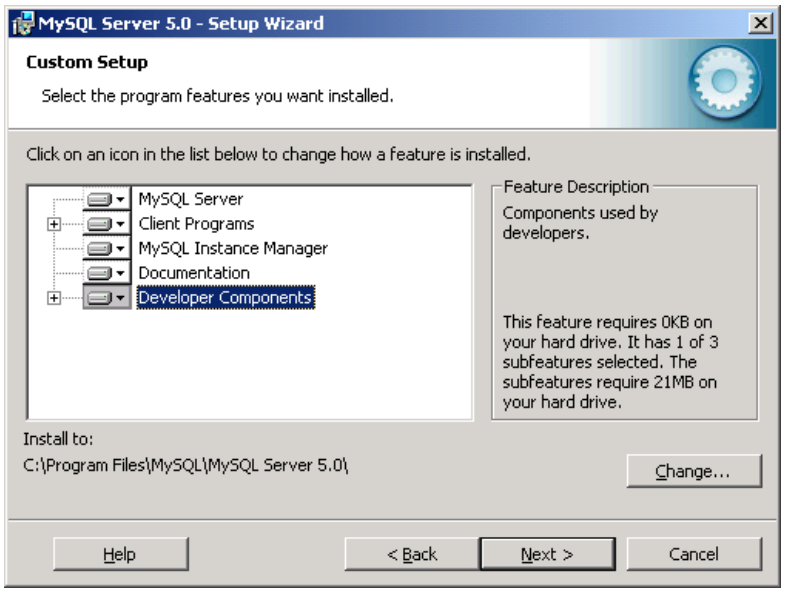
3. ვირჩევთ ინსტალაციის ტიპს (სურ. 1.2):

• Typical – როდესაც დგება მხოლოდ MySQL-ის ძირითადი კომპონენტები: სერვერი და ბრძანებითი სტრიქონის უტილიტები;

• Complete – როდესაც დგება MySQL-ის ყველა კომპონენტი;

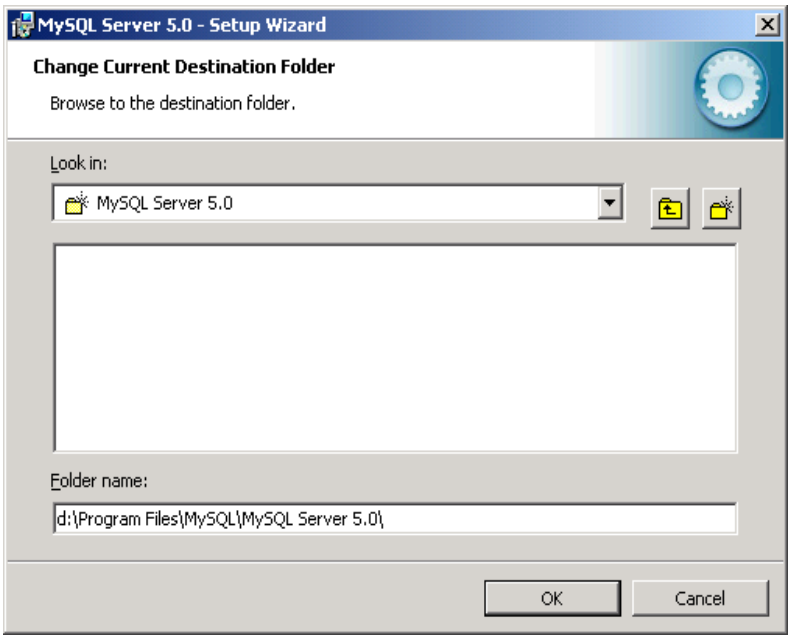
სურ. 1.2.

• Custom – როდესაც მივუთითებთ კატალოგის გზას, რომელშიც დგება MySQL პროგრამა და ვირჩევთ კომპონენტებს (სურ. 1.3).



სურ. 1.3.

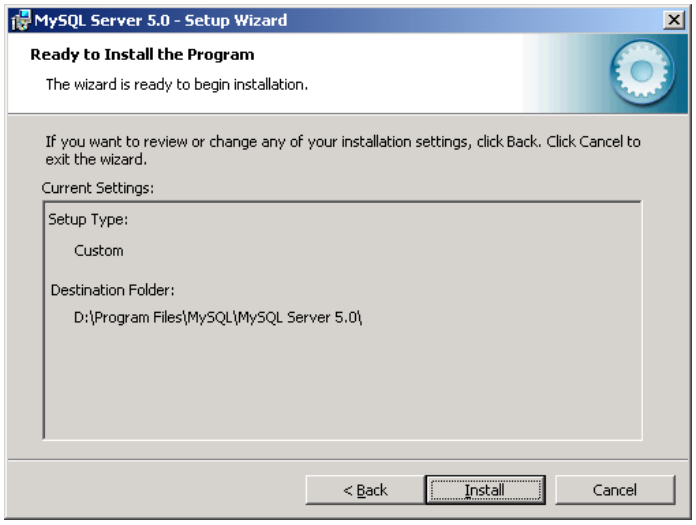
• თუ აუცილებელია ინსტალაციის კატალოგის შეცვლა, ვაჭერთ Change ღილაკს. ჩნდება ფანჯარა (სურ. 1.4), სადაც Folder name ველში შეგვაქვს კატალოგის სახელი ან Look in ველში ვირჩევთ საჭირო დისკს. შემდეგ თანამიმდევრობით ვხსნით ჩასმულ ფოლდერებს, სანამ არ მივალთ სანახველამდე. ვაჭერთ ღილაკს OK.



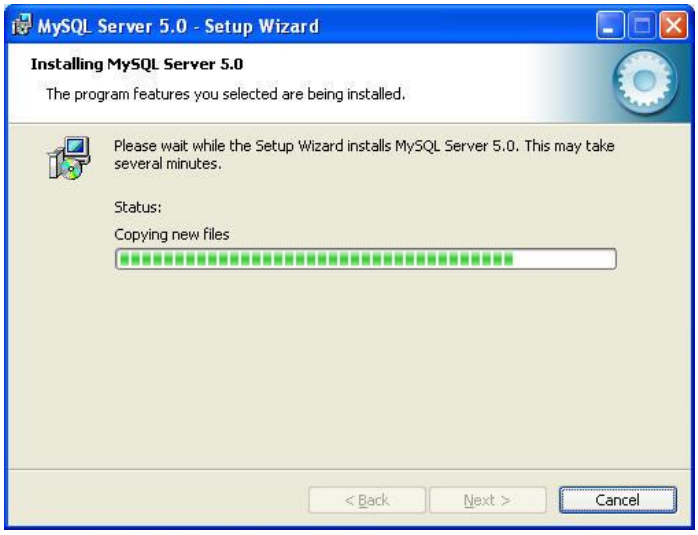
სურ. 1.4.

პარამეტრების აწყობის შემდეგ ვაჭერთ Next ღილაკს.

4. ინსტალაციის ტიპისა და კატალოგის სისწორის შემოწმება ხდება ფანჯარაში (სურ. 1.5). სისწორის შემთხვევაში ვაჭერთ Install ღილაკს.



სურ. 1.5.



სურ. 1.6.

5. ინსტალაციის დამთავრების შემდეგ გამოჩნდება საინფორმაციო ფანჯარა (სურ. 1.7). ამ და მომდევნო ფანჯრებში უბრალოდ ვაჭერთ Next.

სურ. 1.7

6. მივუთითოთ აწყობის ოსტატის გაშვების აუცილებლობა, რისთვისაც ვაყენებთ ალამს Configure the MySQL Server now. ვაჭერთ Finish ღილაკს (სურ. 1.8).

სურ. 1.8.

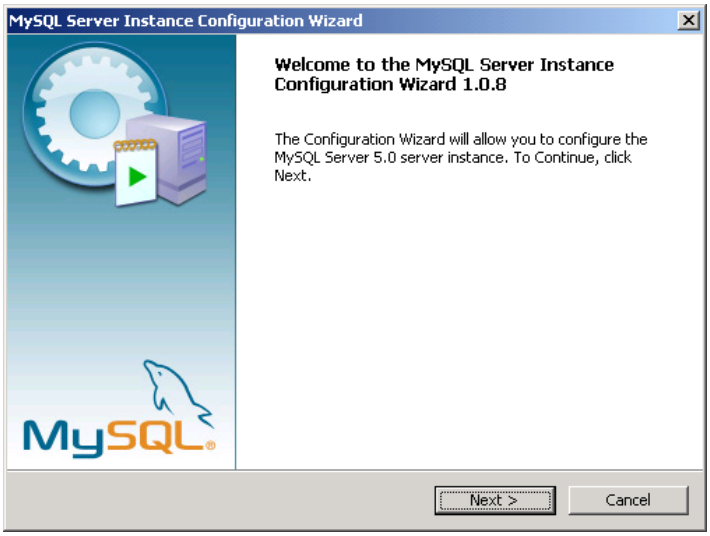
MySQL სერვერის აწყობა

აწყობის ოსტატი ავტომატურად გაიშვება Configure the MySQL Server now -ზე ალამის დაყენებით, თუმცა შესაძლებელია აგრეთვე მენიუდან:

All Programs → MySQL → MySQL Server 5.0 → MySQL Server Instance Config Wizard.

აწყობის ოსტატის გამოყენების შემთხვევაში MySQL სერვერის აწყობისათვის საჭიროა შემდეგი მოქმედებების შესრულება.

1. აწყობის ოსტატის ფანჯარაში (სურ. 1.9) ვაჭერთ Next ღილაკს.

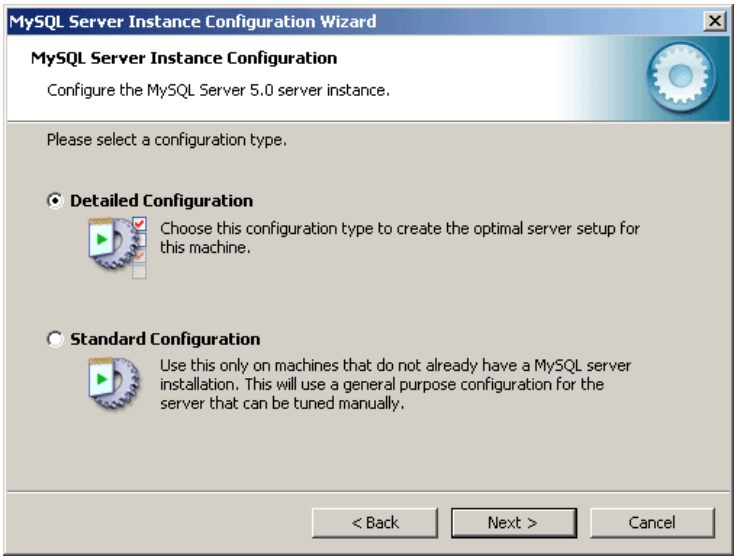


სურ. 1.9.

2. ვირჩევთ სერვერის აწყობის რეჟიმს (სურ. 1.10):

• Detailed Configuration;

• Standard Configuration.



სურ. 1.10.

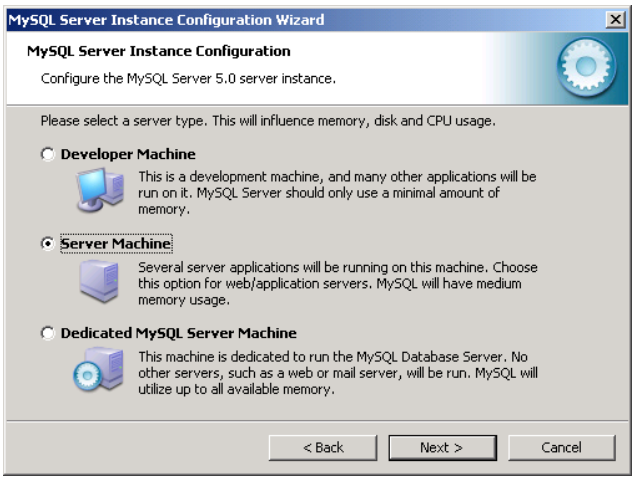
რეკომენდებულია Detailed Configuration ვარიანტი. ვაჭერთ Next ღილაკს. თუ აირჩევთ Standard Configuration ვარიანტს, მაშინ პუნქტები 3–8 უნდა გამოვტოვოთ.

3. ვირჩევთ MySQL კონფიგურაციას, იმის მიხედვით თუ რომელ კომპიუტერზე განთავსდება პროგრამა (სურ. 1.11):

• Developer Machine – მოცემული კონფიგურაცია მისაღებია პერსონალური კომპიუტერისათვის, სადაც ერთდროულად გაიშვება MySQL მრავალი პროგრამა. კონფიგურაცია ითხოვს მინიმალურ სისტემურ რესურსებს;

• Server Machine – მოცემული კონფიგურაცია მისაღებია სერვერისათვის, სადაც ერთდროულად მუშაობს რამდენიმე სერვერული დანართი. მოცემული კონფიგურაცია ითხოვს საშუალო რაოდენობის სისტემურ რესურსებს;

• Dedicated MySQL Server Machine – მოცემული კონფიგურაცია მისაღებია გამოყოფილი სერვერისათვის, სადაც იმუშავებს მხოლოდ MySQL. კონფიგურაცია ითხოვს მაქსიმალური რაოდენობის სისტემურ რესურსებს;

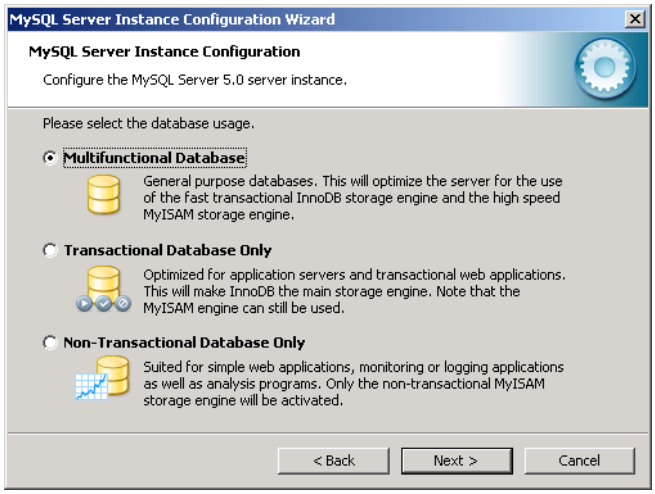


სურ. 1.11.

ვირჩევთ ვარიანტს Developer Machine. ვაჭერთ Next ღილაკს.

4. ცხრილების ძირითად ტიპებს MySQL-ში წარმოადგენს InnoDB და MylSAM, სადაც InnoDB ტიპის ცხრილები მრავალმომხმარებლიან რეჟიმში უზრუნველყოფენ მონაცემთა ცვლილებების ოპერაციათა მაღალ ეფექტურობას. MylSAM ტიპის ცხრილები ვერ უზრუნველყოფენ ტრანზაქციების დამუშავებას, მაგრამ წარმატებულად უზრუნველყოფენ მონაცემთა ძებნისა და წაკითხვის ოპერაციების წარმადობას.

ვირჩევთ მონაცემთა ბაზის ტიპს (სურ. 1.12):

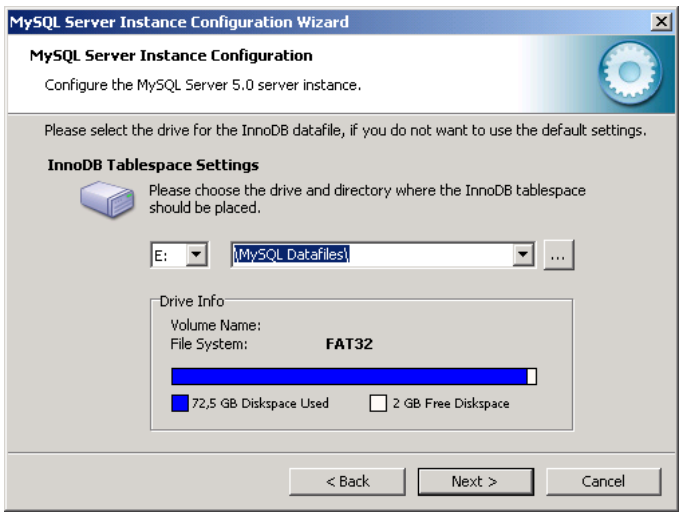


სურ. 1.12.

- Multifunctional Database;
- Transactional Database Only;
- Non-Transactional Database Only.

თუ ვირჩევთ Multifunctional Database-ს. ვაჭერთ Next ღილაკს. თუ ვირჩევთ Non-Transactional Database Only-ის, მაშინ შემდეგი პუნქტი უნდა გამოვტოვოთ.

5. თუ ვირჩევთ ვარიანტს Multifunctional Database ან Transactional Database Only, მაშინ აუცილებელია შევცვალოთ ფაილური სისტემის კატალოგი, სადაც ისურება InnoDB ფაილები.



სურ. 1.13.

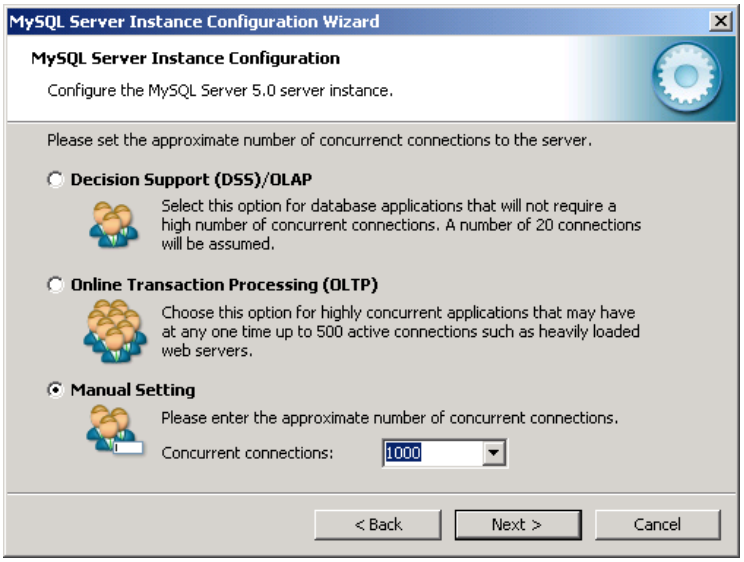
რეკომენდებულია გამოვიყენოთ კატალოგი უსიტყვოდ. ვაჭერთ Next ღილაკს. თუ საჭიროა კატალოგის შეცვლა ვირჩევთ დისკს, შემდეგ ... ღილაკზე დაჭერით - გზას კატალოგისაკენ. დასასრულს, ვაჭერთ Next ღილაკს.

6. მივუთითოთ სერვერთან მიერთებული მომხმარებლების/დანართების სავარაუდო რაოდენობა (სურ. 1.14):

• Decision Support (DSS)/OLAP – დასაშვებია არაუმეტეს 100 ერთდროული შეერთებისა;

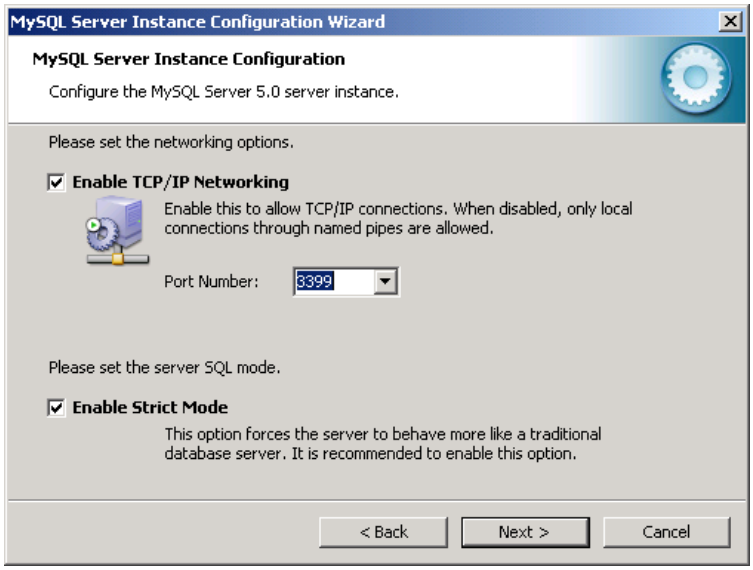
• Online Transaction Processing (OLTP) – დასაშვებია არა უმეტეს 500 აქტიური შეერთებისა;

• Manual Setting – ვირჩევთ ერთდროული შეერთების მაქსიმალურ რაოდენობას. ვაჭერთ Next ღილაკს.



სურ. 1.14.

7. განვსაზღვროთ დაშორებული შეერთებების აუცილებლობა და მკაცრი რეჟიმის გამოყენების შესაძლებლობა (სურ. 1.15).



სურ. 1.15.

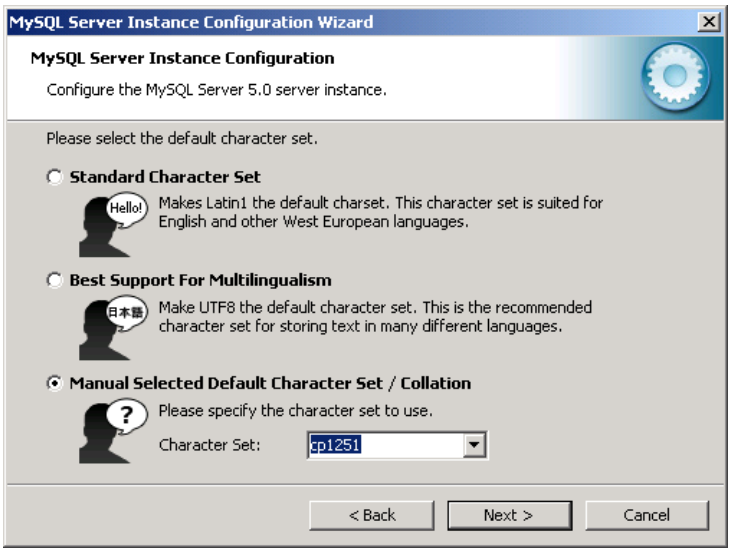
თუ მოსალოდნელია MySQL სერვერთან მომხმარებელის მიერთება დაშორებული კომპიუტერიდან, მაშინ უნდა დავაყენოთ ალამი Enable TCP/IP Networking. ასეთ შემთხვევაში Port Number ველში გამოიყენება 3306.

8. ვირჩევთ კოდირების გვერდს (სურ. 1.16) :

• Standard Character Set – კოდირება Latin1;

• Best Support For Multilingualism – კოდირება Unicode (UTF-8);

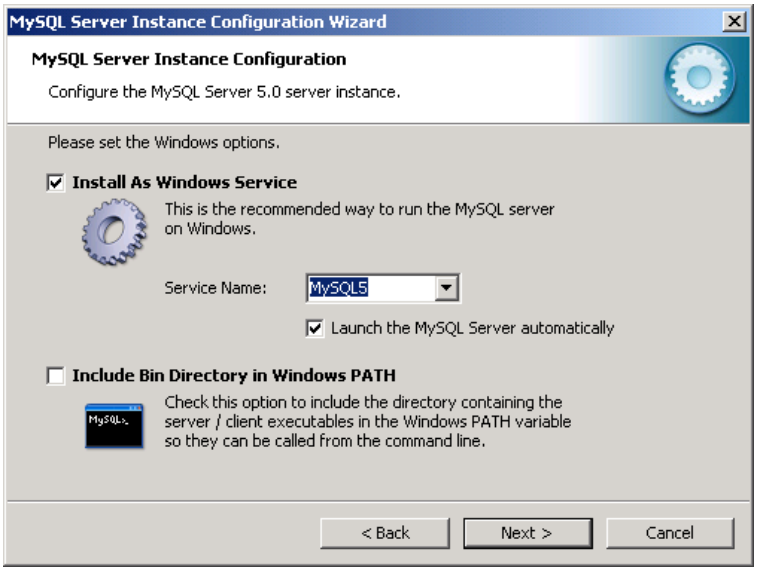
• Manual Selected Default Character Set / Collation.



სურ. 1.16.

რეკომენდებულია Best Support For Multilingualism. ვაჭერთ Next ღილაკს.

9. შევირჩიოთ Windows-ის პარამეტრები, რომლებიც გამოიყენება MySQL პროგრამის მიერ (სურ. 1.17).



სურ. 1.17.

• თუ აუცილებელია MySQL კონფიგურირება, როგორც Windows-ის სერვისი, ვაყენებთ Install As Windows Service ალამს. MySQL სერვერის ავტომატური გაშვებისათვის ვაყენებთ Launch the MySQL Server automatically ალამს.

• ბრძანებითი სტრიქონიდან სერვერისა და უტილიტების გაშვებისათვის ვაყენებთ Include Bin Directory in Windows PATH ალამს.

ვაჭერთ Next ღილაკს.

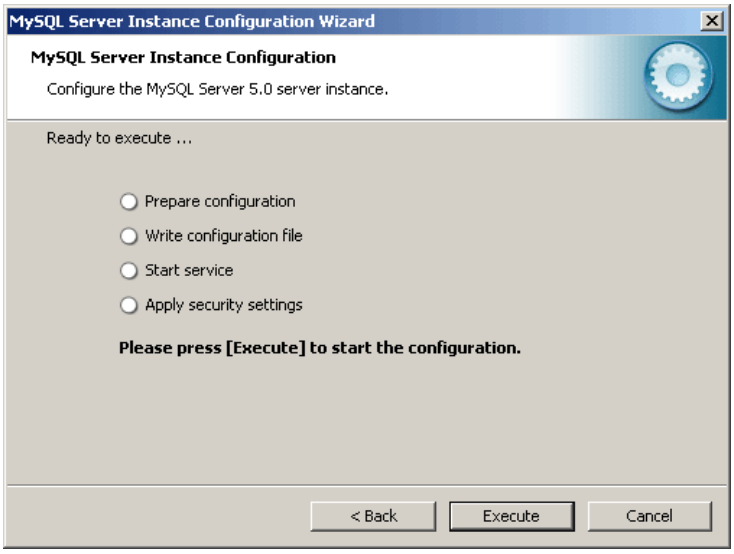
10. ავაწყოთ MySQL უსაფრთხოების პარამეტრები (სურ. 1.18).

სურ. 1.18.

• New root password ველში შევიტანოთ ახალი პაროლი და დავადასტუროთ Confirm.

• თუ აუცილებელია მოხმარებლისათვის ანონიმური წვდომა, მაშინ ვაყენებთ Create An Anonymous Account ალამს. ვაჭერთ Next ღილაკს.

11. კონფიგურირების პროცესის გაშვებისათვის ვაჭერთ Execute ღილაკს (სურ. 1.19).



სურ. 1.19.

12. კონფიგურირების პროცესის დასრულების შემდეგ ვაჭერთ Finish ღილაკს.

3. MySQL GUI Tools დაყენება

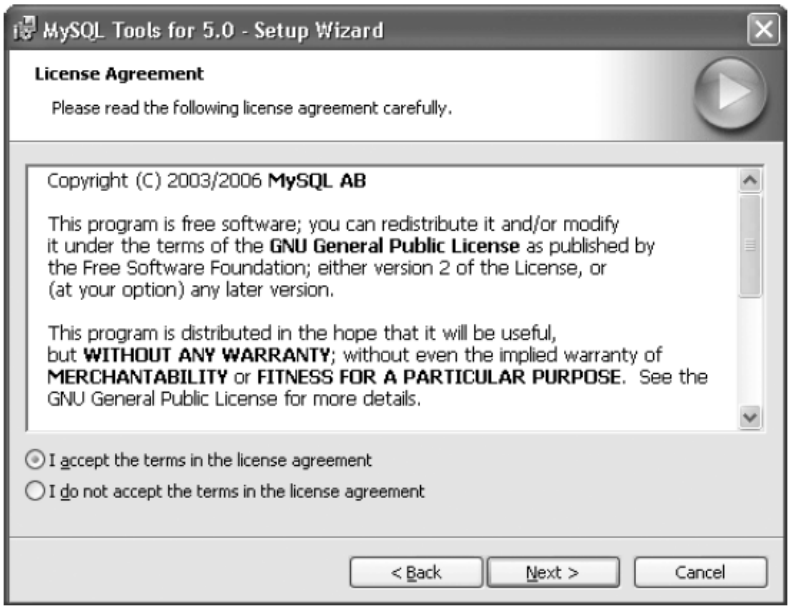
MySQL გრაფიკული უტილიტების დაყენებისათვის შევასრულოთ შემდეგი მოქმედებები:

1. გავუშვათ MySQL GUI Tools დაყენების ოსტატი (Setup Wizard), რისთვისაც ორჯერ დავაწკაპუნოთ mysql-gui-tools-5.0xxx-win32.msi ფაილზე.

2. საწყის ფანჯარაში (სურ. 1.20) ვაჭერთ Next ღილაკს.

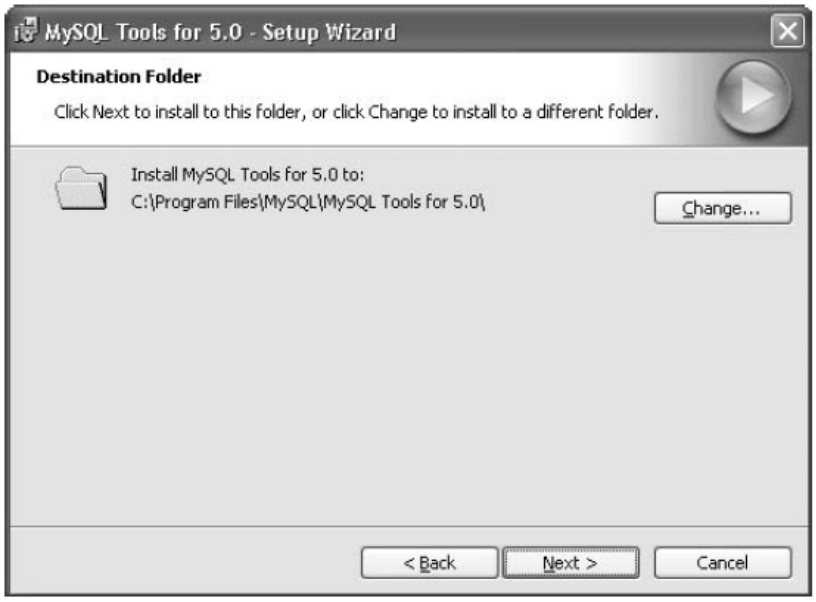
სურ. 1.20.

3. ვირჩევთ გადამრთველის მნიშვნელობას I accept the terms in the license agreement (სურ. 1.21). ვაჭერთ Next ღილაკს.



სურ. 1.21.

4. თუ აუცილებელია უტილიტის დაყენების კატალოგის შეცვლა (სურ. 1.22), ვაჭერთ Change ღილაკს.



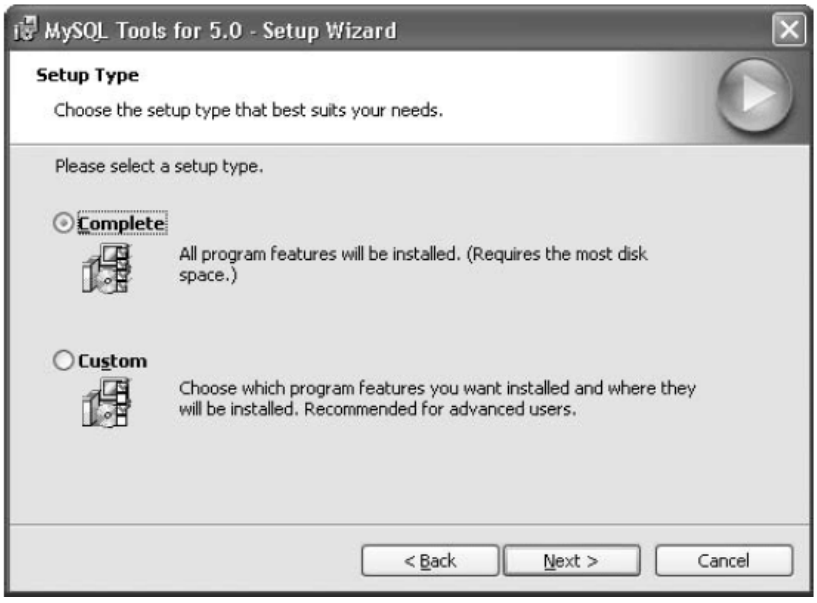
სურ. 1.22.

ფანჯარაში (სურ. 1.4) შევიტანოთ საჭირო კატალოგის გზა Folder name ველში ან Look in ველში სიიდან ვირჩევთ საჭირო დისკს, შემდეგ თანამიმდევრობით ვხსნით ჩასმულ საქაღალდეებს ვიდრე არ ვიპოვით საჭირო საქაღალდეს. ვაჭერთ ღილაკს OK. გაგრძელებისათვის ვაჭერთ Next ღილაკს.

5. ვირჩევთ დაყენების ტიპს (სურ. 1.23):

• Complete – პაკეტის ყველა გრაფიკული უტილიტას დაყენება;

• Custom – პაკეტის ცალკეული გრაფიკული უტილიტას დაყენება.

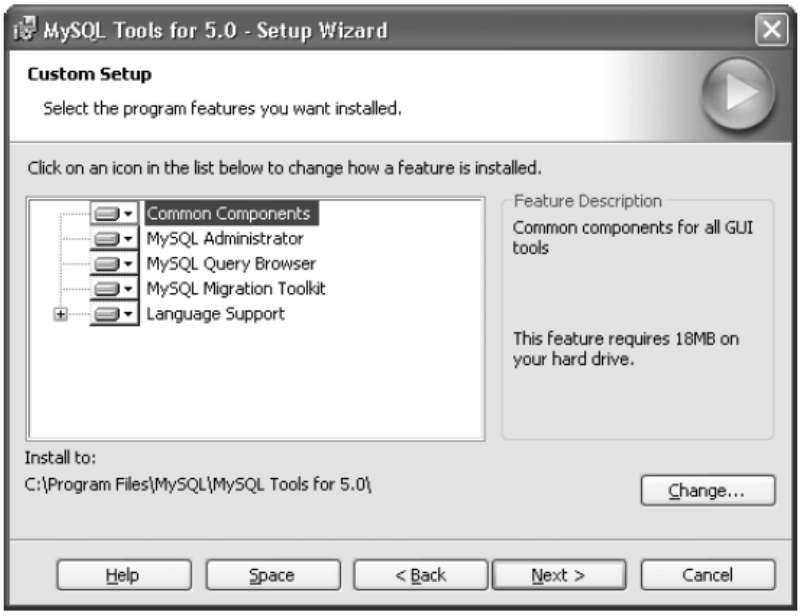


სურ. 1.23.

ვაჭერთ Next ღილაკს.

თუ ვირჩევთ Complete -ს, მაშინ შემდეგ პუნქტს გამოვტოვებთ.

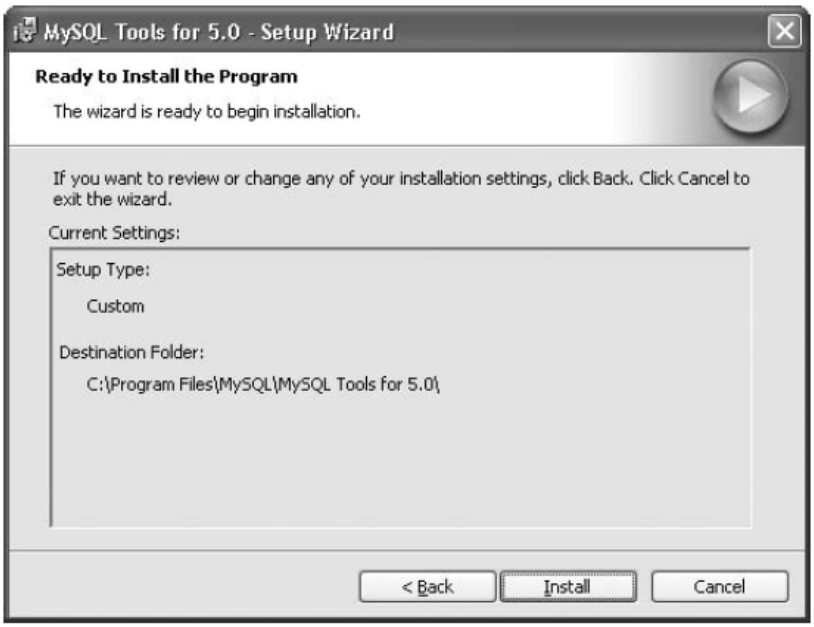
6. თუ ვირჩევთ Custom -ს, მაშინ უნდა მივუთითოთ კომპონენტები (სურ. 1.24).



სურ. 1.24.

ვაჭერთ Next ღილაკს.

7. ვამოწმებთ შერჩეული დაყენების ტიპისა და დაყენების კატალოგის სისწორეს (სურ. 1.25). თუ აუცილებელია პარამეტრების შეცვლა, ვაჭერთ Back ღილაკს. თუ პარამეტრები სწორად არის შერჩეული, მაშინ ვაჭერთ Install ღილაკს.



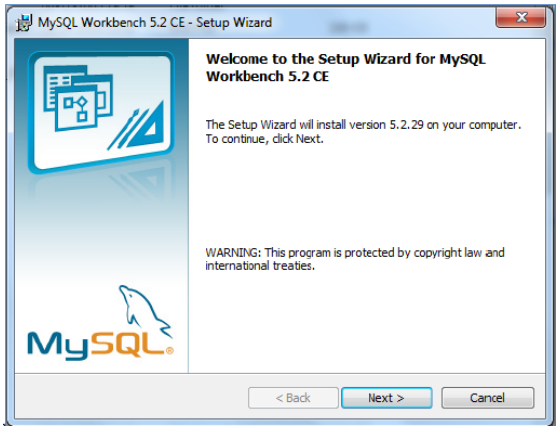
სურ. 1.25.

8. გამოჩნდება ინფორმაციული ფანჯარა (სურ. 1.6). ამ და მომდევნო ფანჯრებში უბრალოდ ვაჭერთ ღილაკს Next. ბოლო ფანჯარაში ვაჭერთ ღილაკს Finish.

MySQL Workbench -ის ინსტალაცია Windows -ში

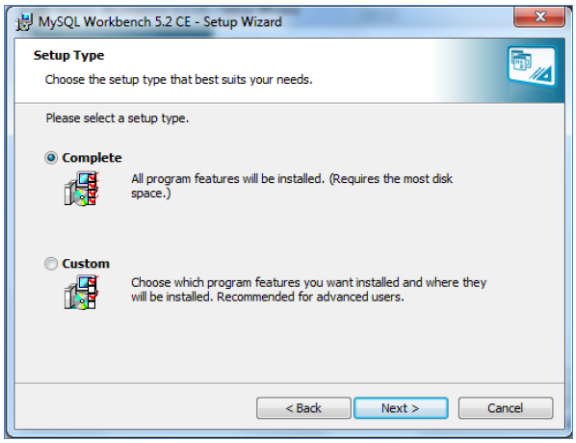
ინსტალაციის ბიჯები (Microsoft Windows პლატფორმისათვის):

1. ჩავტვირთოთ MySQL Workbench შემდეგი საიტიდან:
http://dev.mysql.com/downloads/workbench/5.2.html .
2. გავხსნათ MySQL Workbench საინსტალაციო ფაილი და ვაჭერთ ღილაკს “Next”.



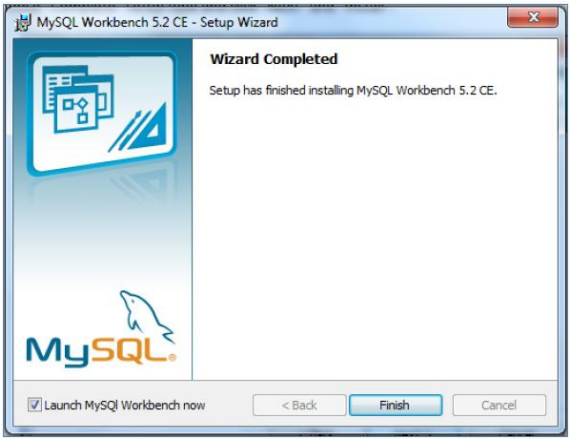
სურ. 1.26.

3. ვირჩევთ სეტაპის “Complete” ტიპს, ვაწკაპუნებთ “Next” და “Install”.



სურ. 1.27

4. ინსტალაციის პროცესის დამთავრების შემდეგ ვაწკაპუნებთ “Finish”.

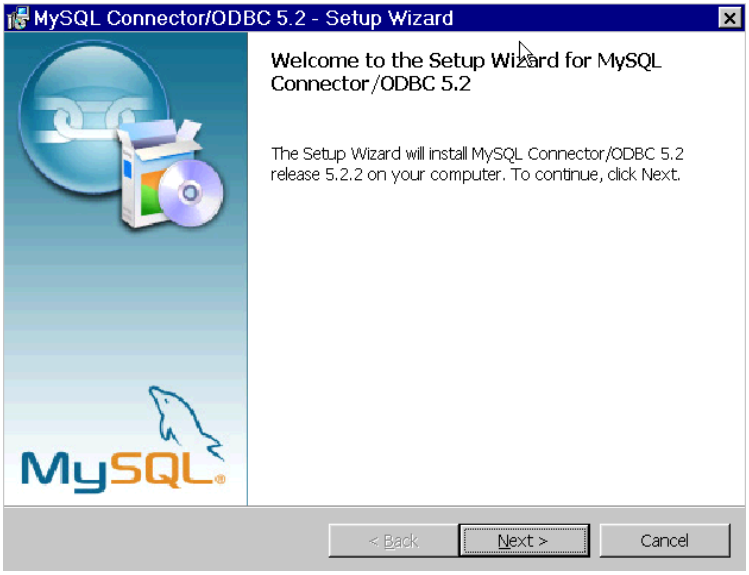


სურ. 1.28

Connector/ODBC -ს ინსტალაცია

Connector/ODBC -ს ინსტალაციის პროცესი შედგება შემდეგი ბიჯებისაგან:

1. ინსტალაციის არქივულ ფაილზე ორმაგი დაწკაპუნებით ვხსნით მას.
2. MySQL Connector/ODBC Setup Wizard იწყებს მოქმედებას. ვაწკაპუნებთ ღილაკზე Next.

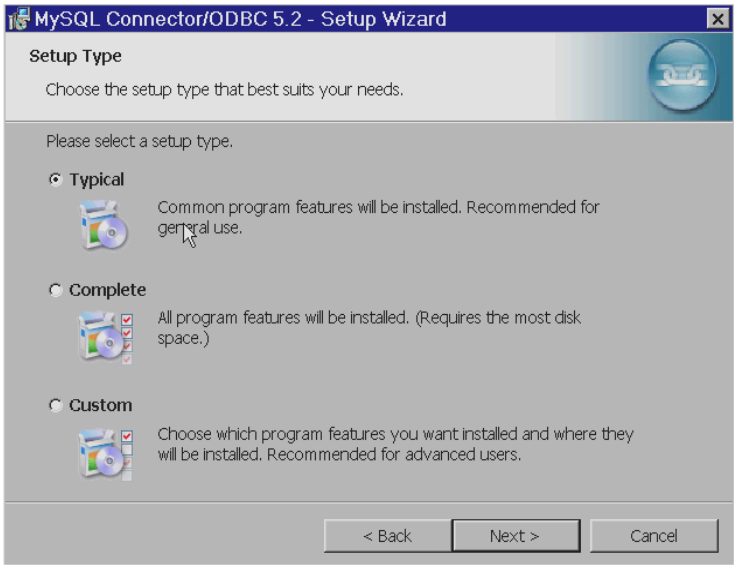


სურ. 1.29

3. ვირჩევთ ინსტალაციის ტიპს.

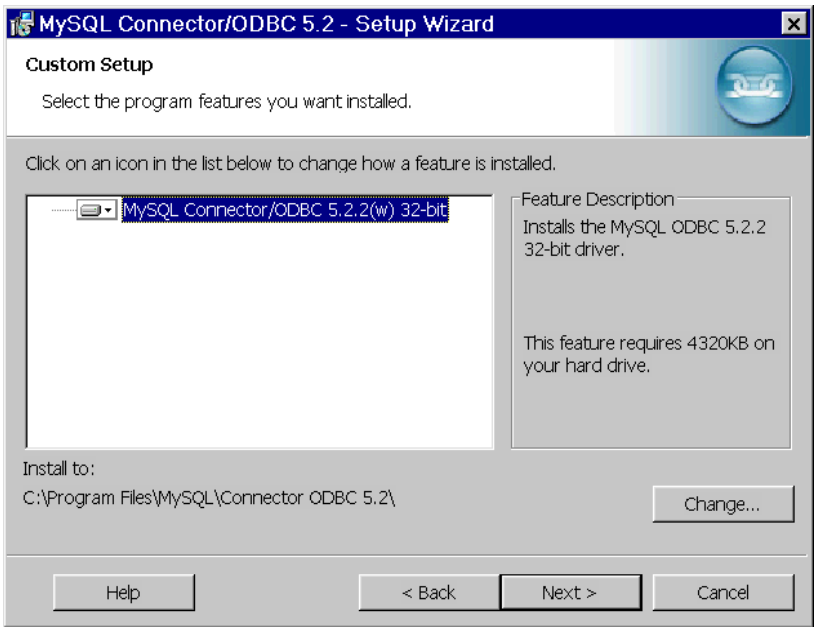
Typical ან **Complete** -ის შემთხვევაში ვაწკაპუნებთ Next ღილაკზე და შემდეგ გადავდივართ 5 ბიჯზე.

თუ ავირჩევთ **Custom** -ს, მაშინ ვაწკაპუნებთ Next ღილაკზე და შემდეგ გადავდივართ 4 ბიჯზე.



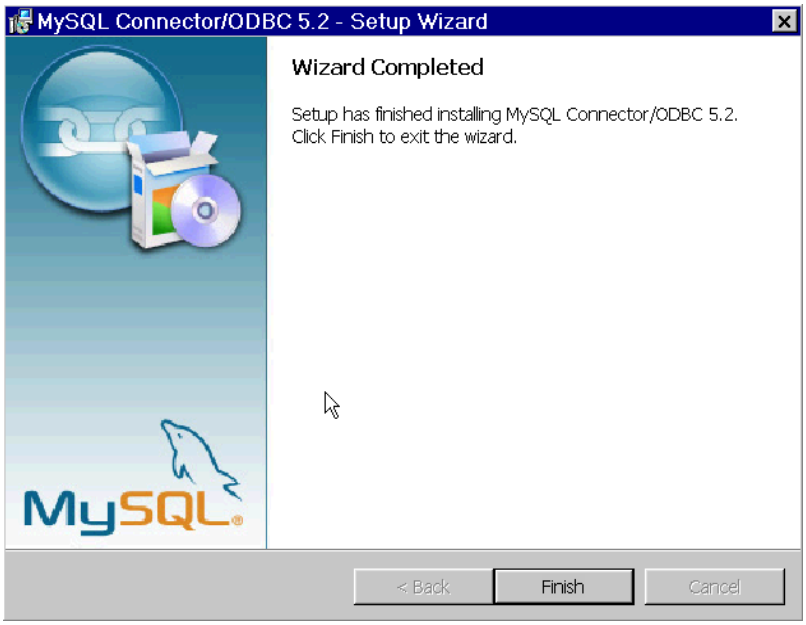
სურ. 1.30

4. გამოვიყენოთ ჩამოშლადი სიები საინსტალაციო კომპონენტების შესარჩევად და საჭირო ფაილების დასაინსტალირებლად ვაწკაპუნებთ Next ღილაკზე.



სურ. 1.31

5. ფაილების კოპირებით სრულდება ინსტალაციის პროცესი, რომლიდანაც გამოსვლისათვის ვაწკაპუნებთ Finish ღილაკზე.

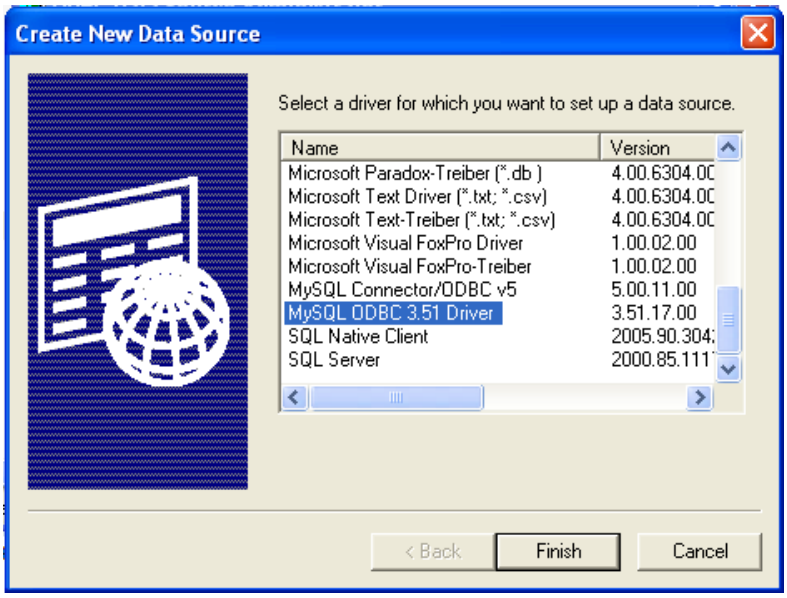


სურ. 1.32

Connector/ODBC DSN -ის კონფიგურირება Windows -ზე

Connector/ODBC 5.x DSN -ს (**data source name**) კონფიგურირებისათვის ვიყენებთ ბრძანებით სტრიქონს ან **ODBC Data Source Administrator** GUI.

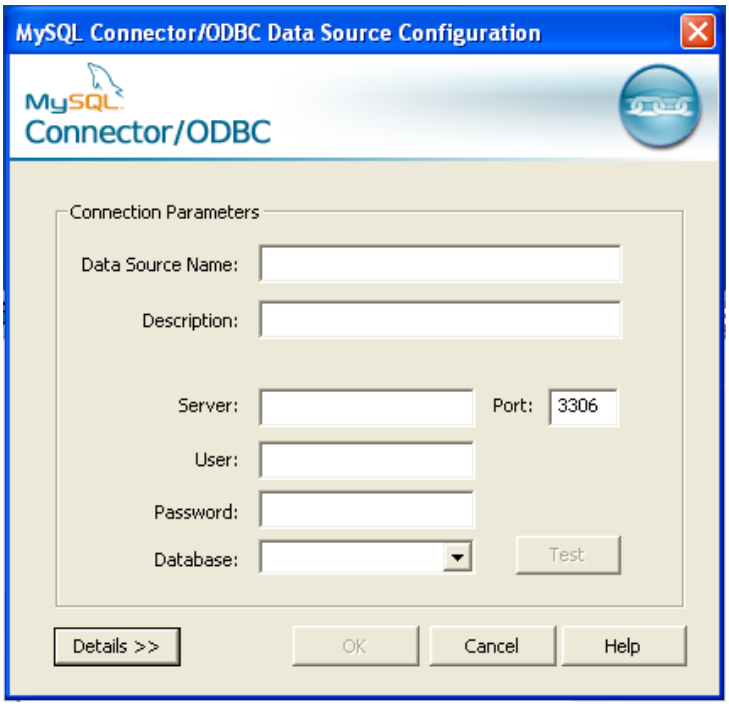
1. გავხსნათ **ODBC Data Source Administrator**.
2. System DSN -ის შესაქმნელად ვირჩევთ ჩანართს **System DSN**, სადაც User DSN-ის შესაქმნელად ვაწკაპუნებთ ADD ღილაკზე.
3. ვირჩევთ ODBC დრაივერს მოცემული DSN-სათვის.



სურ. 1.33

Connector/ODBC-ის შესაბამისი დონისათვის ვირჩევთ **MySQL ODBC 5.*x* Driver** და ვაწკაპუნებთ Finish ღილაკზე.

4. ახლა უკვე საჭიროა DSN-ის სპეციალური ველების კონფიგურაცია **Connection Parameters** ფანჯარაში.



სურ. 1.34

Data Source Name უჯრედში შეგვაქვს მონაცემთა წყაროს დასახელება.

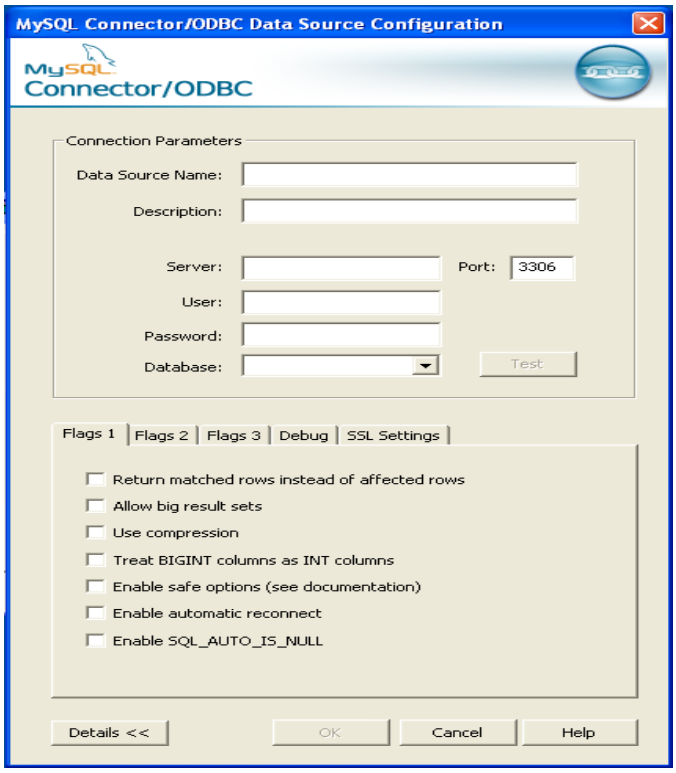
5. **Description** უჯრედში შეგვაქვს მიერთების ტექსტის აღწერილობა.
6. **Server** ველში შეგვაქვს MySQL სერვერის ჰოსტის სახელი (უსიტყვოდ, **localhost**).
7. **User** ველში შეგვაქვს მომხმარებლის სახელი ამ მიერთებისათვის.
8. **Password** ველში შეგვაქვს შესაბამისი პაროლი ამ მიერთებისათვის.
9. **Database** ჩამოშლადი სია იძლევა მონაცემთა ბაზების ჩამონათვალს, რომელთანაც მომხმარებელს აქვს წვდომა.

10. TCP/IP პორტი, რომელიც უსიტყვოდ არის (3306), შევცვალოთ **Port** მნიშვნელობით.

11. DSN-ის შენახვისათვის ვაწკაპუნებთ OK.

მიერთების შემოწმებისათვის ვაწკაპუნებთ Test ღილაკზე. წარმატების შემთხვევაში ვიღებთ შეტყობინებას **Success; connection was made!**

ჩვენ შეგვიძლია აგრეთვე ოპციების რაოდენობის კონფიგურაცია მოცემული DSN -სათვის, რისთვისაც ვიყენებთ ღილაკს Details.

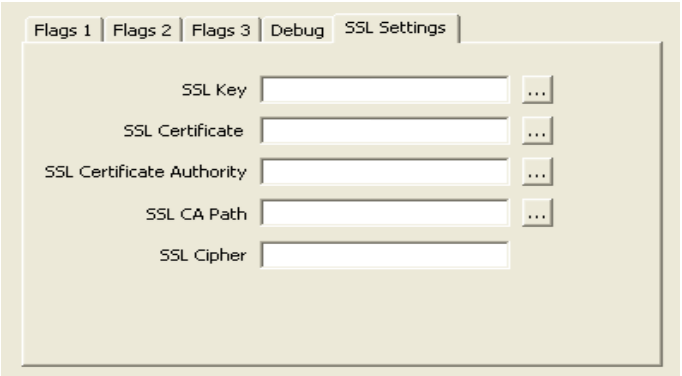


სურ. 1.35

ამასთან, **Details** ღილაკი ხსნის შემდეგ ჩანართებს:

- **Flags 1**, **Flags 2** და **Flags 3** დამატებითი ალამების არჩევისათვის მოცემული DSN -სათვის.

- **SSL Settings** ახდენს, MySQL სერვერთან კომუნიკაციის შემთხვევაში, დამატებითი ოპციების კონფიგურაციას ქვემოთ მოყვანილი პარამეტრების შერჩევის გზით, რომლებიც საჭიროა Secure Sockets Layer (SSL)-სათვის.



სურ. 1.36

ლაბორატორიული სამუშაო 2
მუშაობა Tools-თან

სამუშაოს მიზანი:

1. მუშაობა MySQL - ში;
2. MySQL Workbench - ის ინსტალაცია Windows-ში;
3. მუშაობა MySQL Workbench - ში.

1. მუშაობა MySQL-ში

MySQL-ში მუშაობის დაწყება

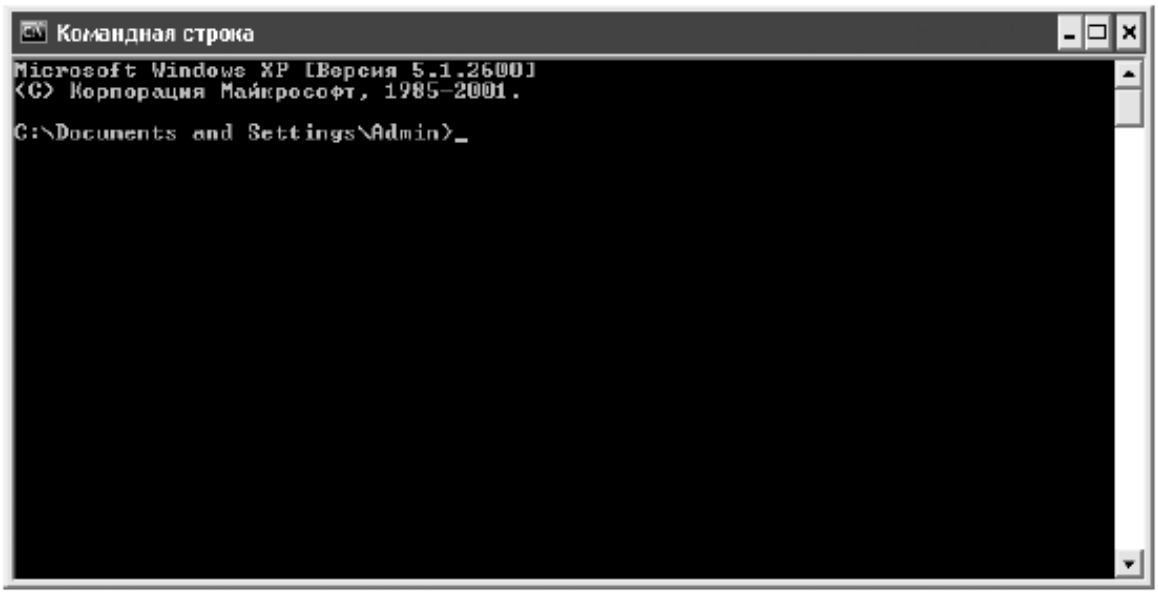
მონაცემთა ბაზასთან მუშაობისათვის საჭიროა MySQL სერვერის გაშვება და მასთან მიერთება. თუ MySQL სერვერი კონფიგურირებული იყო როგორც Windows -ის სერვისი, მაშინ ის ავტომატურად გაიშვება აწყობის შემდეგ. წინააღმდეგ შემთხვევაში, სერვერის გაშვება შესაძლებელია ბრძანებითი სტრიქონიდან ან MySQL Administrator გრაფიკული უტილიტის მეშვეობით.

MySQL სერვერის გაშვება და გაჩერება ბრძანებითი სტრიქონიდან

MySQL სერვერის ხელით გაშვება შესაძლებელია შემდეგი ხერხებით:

• ორჯერ ვაწკაპუნებთ mysqld-nt.exe ფაილზე, რომელიც მოთავსებულია MySQL პროგრამის კატალოგის bin ქვეკატალოგში.

• გაიხსნება ბრძანებითი სტრიქონის Windows ფანჯარა, რისთვისაც ვაჭერთ გაშვების ღილაკს, რომელიც პროგრამის გაშვების ფანჯარაში მენიუს „შესრულება“ პუნქტის არჩევით გამოჩნდება. Open ველში შეგვაქვს cmd ბრძანება და ვაჭერთ ღილაკს OK. ეკრანზე ჩნდება ფანჯარა (სურ. 2.1.).



სურ. 2.1.

ბრძანებით სტრიქონში შეგვაქვს *mysqld-nt* ბრძანება და ვაჭერთ Enter კლავიშს. გაიშვება MySQL სერვერი. თუ სერვერის აწყობის დროს bin ქვეკატალოგისაკენ გზაში არ იყო დამატებული Path სისტემური ცვლადის მნიშვნელობა, მაშინ აუცილებელია სრული გზის შეტანა. მაგალითად: *C: \Program Files\MySQL\MySQL Server 5.0\bin\mysqld-nt*

თუ MySQL სერვერის აწყობის დროს არ იყო მითითებული მომხმარებლის პაროლი root, მაშინ აუცილებელია მისი დაყენება შემდეგი ბრძანების გამოყენებით:

mysqladmin -u root password <პაროლი>

(ან C: \Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin -u root password

პაროლის შეცვლისათვის გამოიყენება იგივე ბრძანება – p ოპციის გამოყენებით:

mysqladmin -u root -p password <ახალი პაროლი>

დასასრულ, თუ აუცილებელია MySQL სერვერის შეჩერება, სრულდება ბრძანება:

mysqladmin -u root -p shutdown

MySQL სერვერის გაშვება და გაჩერება MySQL Administrator-ის დახმარებით

გრაფიკული უტილიტის MySQL Administrator გამოყენების შემთხვევაში სრულდება შემდეგი მოქმედებები.

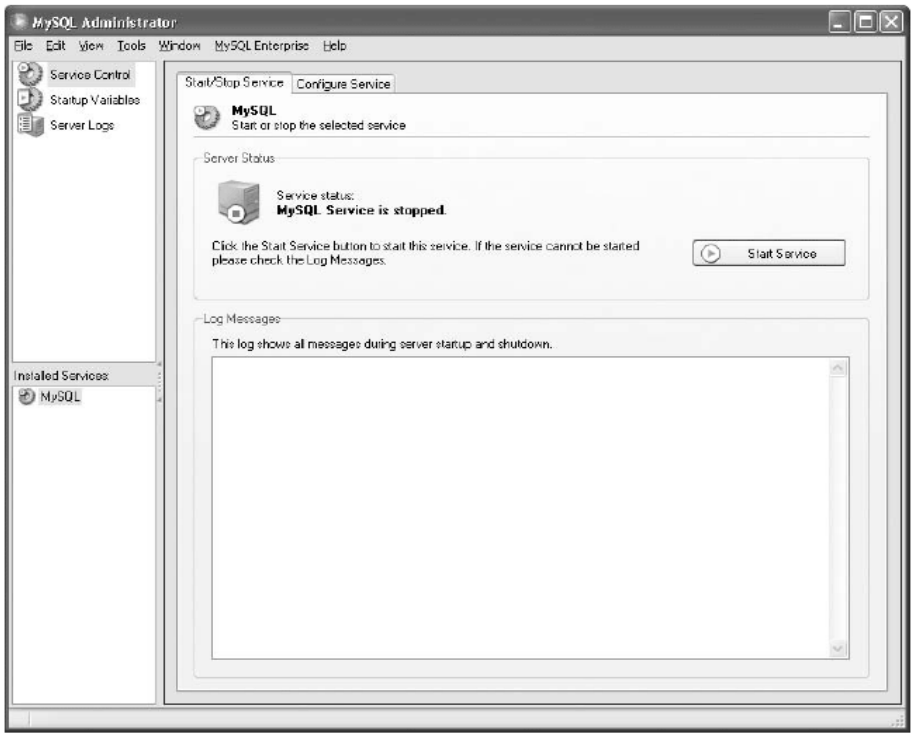
1. გავუშვათ პროგრამა MySQL Administrator:

   Start → All programs → MySQL →MySQL Administrator.

ეკრანზე ჩნდება ფანჯარა(სურ. 2.2).

სურ. 2.2.

2. ვაჭერთ და გვიჭირავს კლავიში Ctrl, ვაწკაპუნებთ Skip ღილაკზე, რომელიც ჩნდება ფანჯრის მარჯვენა ქვედა კუთხეში Cancel ღილაკთან ერთად. ეკრანზე ჩნდება მთავარი ფანჯარა MySQL Administrator (სურ. 2.3.).



სურ. 2.3.

3. ვაწკაპუნებთ პუნქტზე Service Control.

4. თუ MySQL სერვერი არ იყო კონფიგურირებული როგორც Windows სერვისი, მაშინ Start Service ღილაკი მიუწვდომელი იქნება. ამ დროს აუცილებელია შემდეგი მოქმედებების შესრულება:

1) გადავიდეთ Configure Service ჩანართზე. ვაჭერთ Install new Service ღილაკს;

2) შემდეგ დიალოგურ პანელზე მივუთითებთ სერვისის დასახელებას და ვაჭერთ OK ღილაკს;

3) Config Filename ველში შეგვაქვს გზა კონფიგურაციის ფაილისაკენ my.ini (სურ. 2.4.). მაგალითად,

C: \Program Files\ MySQL\MySQL Server5.0\my.ini.

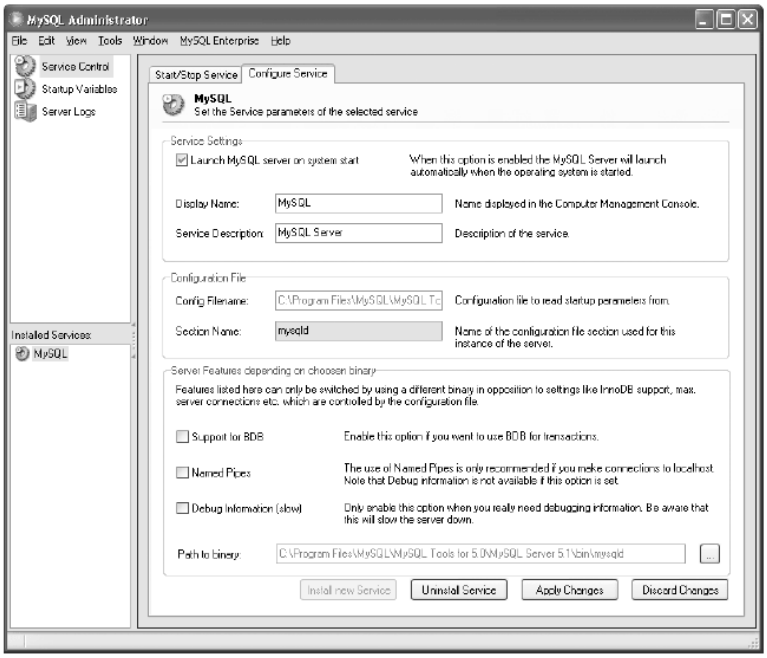
შრიფტის წითელი ფერი მიუთითებს იმაზე, რომ ფაილი არ მოიძებნა;

4) ველში Path to binary შეგვაქვს გზა ფაილისაკენ mysqldnt.exe, მაგალითად,

C: \Program Files\MySQL\MySQL Server 5.0\bin\mysqld-nt;

5) ვაჭერთ ღილაკს Apply Changes;

6) ვბრუნდებით ჩანართში Start/Stop Service.



სურ. 2.4.

5. ვაჭერთ Start Service ღილაკს და MySQL სერვერი გაიშვება.

MySQL სერვერის გაჩერებისათვის MySQL Administrator-ის მეშვეობით უნდა შესრულდეს შემდეგი მოქმედებები.

1. გავუშვათ პროგრამა MySQL Administrator

Start → All programs → MySQL →MySQL Administrator

ეკრანზე გამოჩნდება ფანჯარა (სურ. 2.2.).

2. სერვერთან შეერთების ველში შეგვაქვს შეერთების პარამეტრები:

• Server Host – მნიშვნელობა localhost (ლოკალური კომპიუტერი);

• Port – პორტის ნომერი (უსიტყვოდ – 3306);

• Usename – მომხმარებლის სახელი (root მნიშვნელობა);

• Password – მომხმარებლის პაროლი. ვაჭერთ ღილაკს OK.

3. MySQL Administrator ფანჯრის მარცხენა ნაწილში ვაწკაპუნებთ პუნქტზე Service Control.

4. მარჯვენა ნაწილში ვაჭერთ ღილაკს Stop Service. სერვერი ჩერდება.

MySQL სერვერის გაშვება და გაჩერება მართვის პანელიდან

თუ MySQL სერვერი კონფიგურირებული იყო როგორც Windows-ის სერვისი აწყობის ოსტატის ან MySQL Administrator უტილიტის მეშვეობით, მაშინ MySQL სერვერის გაშვება და გაჩერება შესაძლებელია მართვის პანელიდან, რომლის გამოძახებისათვის ვაჭერთ Star ღილაკს, ხოლო მენიუში ვირჩევთ მართვის პანელს. შემდეგ ორჯერ ვაწკაპუნებთ ადმინისტრირების ნიშანზე და ბოლოს ადმინისტრირების საშუალებათა ფანჯარაში ორჯერ ვაწკაპუნებთ სერვისის ნიშანზე. ეკრანზე ჩნდება სერვისის ფანჯარა სერვისების სიით.

ვაწკაპუნებთ სერვისის დასახელებაზე, ხოლო შემდეგ საჭირო ოპერაციაზე: სერვისის გაშვება, სერვისის გაჩერება და სერვისის ხელმეორედ გაშვება.

სერვერთან მიერთება ბრძანებითი სტრიქონიდან

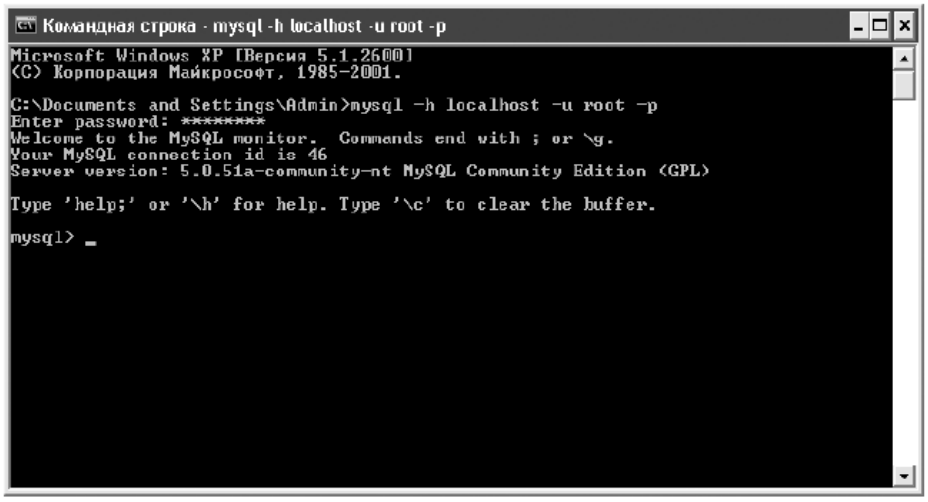
შევასრულოთ შემდეგი მოქმედებები:

1. გავხსნათ ბრძანებითი სტრიქონი.
2. შეგვაქვს ბრძანება:

mysql -h <კომპიუტერის სახელი> -u <მომხმარებლის სახელი> -p

ვაჭერთ Enter კლავიშს. ჩნდება Enter password შემოთავაზება და შეგვაქვს პაროლი. თუ სერვერი იმავე კომპიუტერზე (localhost) მუშაობს, MySQL სერვერთან მიერთებისათვის აღარ არის საჭირო მითითება, მაგალითად *mysql -u root –p.*

სერვერთან მიერთების შემდეგ ჩნდება mysql> (სურ. 2.5.).



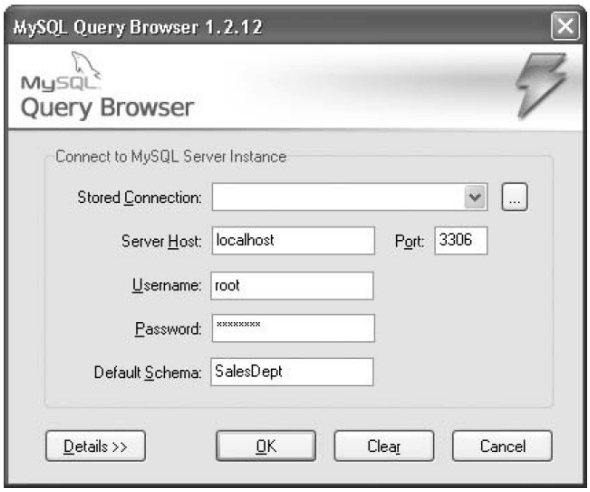
სურ. 2.5.

სერვერის გამორთვისათვის ბრძანებით სტრიქონში უბრალოდ ვკრეფთ *exit* ბრძანებას და ვაჭერთ Enter კლავიშს. სერვერთან მიერთების ალტერნატიულ ხერხს წარმოადგენს გრაფიკული უტილიტა MySQL Query Browser.

სერვერთან მიერთება **MySQL Query Browser** მეშვეობით

შევასრულოთ შემდეგი მოქმედებები:

1. გავუშვათ პროგრამა MySQL Query Browser:
   Start → All Programs → MySQL →MySQL Query Browser.
   ეკრანზე ჩნდება სერვერთან მიერთების ფანჯარა (სურ. 2.6.).



სურ. 2.6.

2. ველებში შევიტანოთ შეერთების პარამეტრები:

• Server Host – კომპიუტერის სახელი, რომელზეც მუშაობს MySQL სერვერი;

• Port – პორტის ნომერი (უსიტყვოდ – 3306);

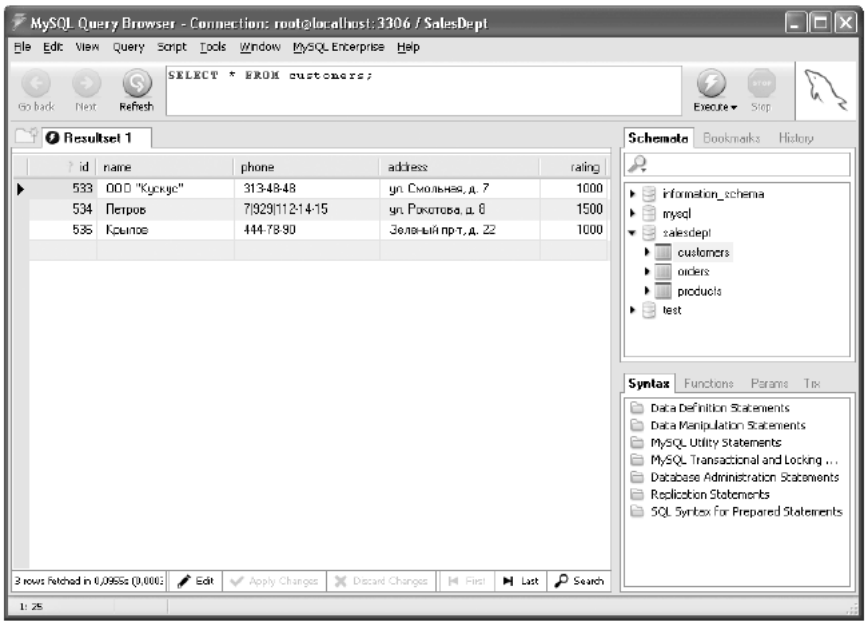
• Username;

• Password;

• Default Schema – მონაცემთა ბაზის სახელი (არსებული ან ახალი), რომელთანაც უნდა ვიმუშაოთ.

3. ვაჭერთ OK ღილაკს. თუ ახალი მონაცემთა ბაზის სახელი შევიტანეთ, მაშინ მისი შექმნისათვის დიალოგურ პანელზე ვაჭერთ Yes ღილაკს.

ეკრანზე ჩნდება MySQL Query Browser -ის მთავარი ფანჯარა (სურ. 2.7.), სადაც უკვე სრულდება მონაცემთა ბაზასთან დაკავშირებული ყველა ოპერაცია.



სურ. 2.7.

2. მუშაობა mySQL Workbench-ში

1. EER- დიაგრამის შექმნა

MySQL Workbench გარემოს დანიშნულებაა მონაცემთა ბაზის ვიზუალური დაპროექტება და MySQL სერვერის მართვა.

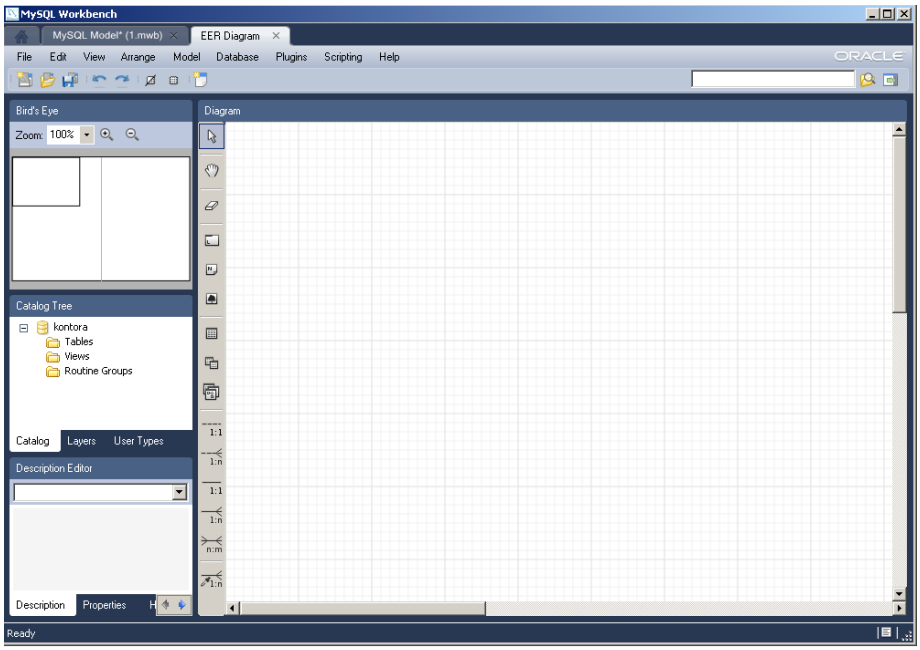
მოდელების ასაგებად გამოიყენება სექცია Data Modeling:

სურ. 2.8.

ვირჩევთ პუნქტს Create new EER Model (Extended Entity-Relationship Model).

უსიტყვოდ შექმნილ მოდელს მიენიჭება myDB სახელი. მოდელის სახელზე ვაწკაპუნებთ თაგუნას მარჯვენა ღილაკზე და მენიუდან ვირჩევთ Edit schema პუნქტს. შემდგომ თვისებების ფანჯარაში შეიძლება მოდელის სახელის შეცვლა. მოდელს ვარქმევთ სახელს. ამავე ფანჯარაში საჭიროა ე.წ. კოდური გვერდების აგება, რისთვისაც სიიდან ვირჩევთ «cp1251- cp1251_general_ci» პუნქტს. ვხურავთ ფანჯარას.

დიაგრამას ვაგებთ ვიზუალური საშუალებების მეშვეობით. ვაწკაპუნებთ პუნქტზე Add diagram, შედეგად ჩაიტვირთება დიაგრამის ცარიელი ფანჯარა.



სურ. 2.9.

ახალი ცხრილის შექმნა შესაძლებელია  პიქტოგრამის მეშვეობით. ამისათვის ვაწკაპუნებთ პიქტოგრამაზე, შემდეგ ვაწკაპუნებთ დიაგრამის სამუშაო არეში. ამ ადგილზე ჩნდება table1 ცხრილი, რომელზეც ორმაგი დაწკაპუნებით შეიძლება შევცვალოთ ცხრილის სახელი და ავაგოთ მისი სტრუქტურა.

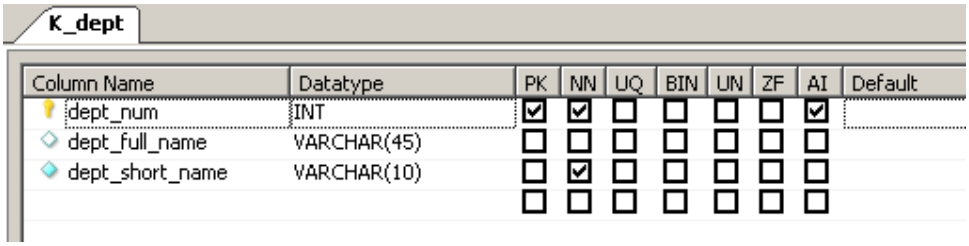
მაგალითისათვის, ავაგოთ ცხრილი განყოფილებები (ანუ k_dept) შემდეგი სვეტებით: განყოფილების_ნომერი, განყოფილების_სრული დასახელება, განყოფილების_მოკლე დასახელება. table1-ს გადავარქვათ k_dept და დავიწყოთ სვეტების შექმნა. ყოველ სვეტს აქვს:

- სახელი,
- მონაცემთა ტიპი. ყველაზე გავრცელებული ტიპებია:
  - INT – მთელი რიცხვი;
  - VARCHAR(მაქსიმალური ზომა) – ცვლადი სიგრძის სიმბოლური მონაცემი;
  - DECIMAL(ზომა, ათობითი ნიშანი) – ათობითი რიცხვი;
  - DATE – თარიღი;
  - DATETIME – თარიღი და დრო.

შემდეგ სვეტების გასწვრივ, ალამის დაყენების მეშვეობით ვაწყობთ ველების დამატებით თვისებებს:

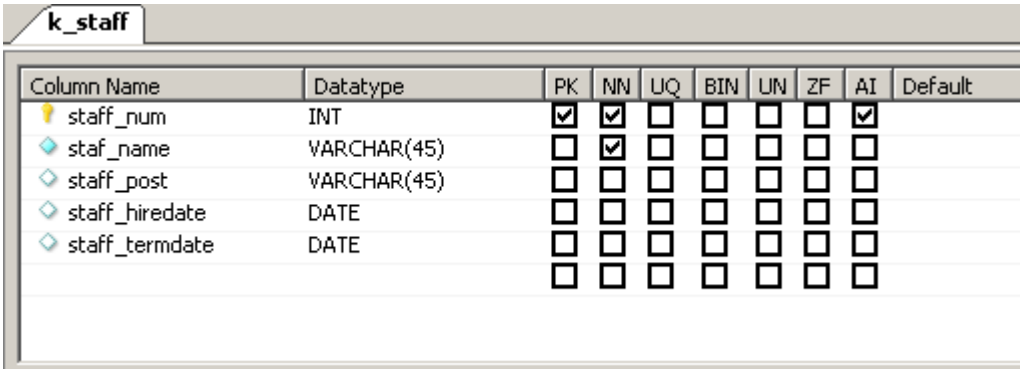
- PK (primary key) – პირველადი გასაღები;
- NN (not null) – დაუშვებელია ცარიელი მნიშვნელობა;
- UN (unique) – სვეტს უნდა ჰქონდეს უნიკალური მნიშვნელობა;
- AI (auto incremental) – პირველადი გასაღების მნიშვნელობათა ავტომატური ზრდა ნატურალური რიცხვებით 1, 2, 3, და ა.შ.;
- DEFAULT – სერვერის მიერ უსიტყვოდ მინიჭებული მნიშვნელობა.

k_dept ცხრილის სტრუქტურას ექნება შემდეგი სახე:



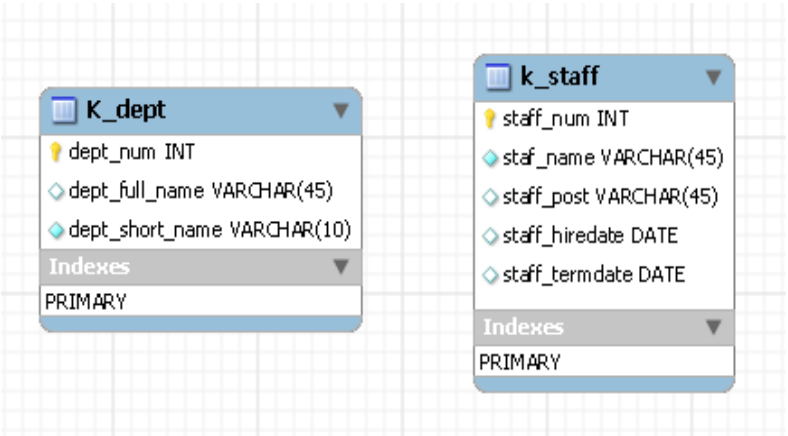
სურ. 2.10.

ანალოგიურად ვქმნით ცხრილს თანამშრომლები (k_staff) შემდეგი სვეტებით:



სურ. 2.11.

შექმნილი ცხრილები ასეთია:



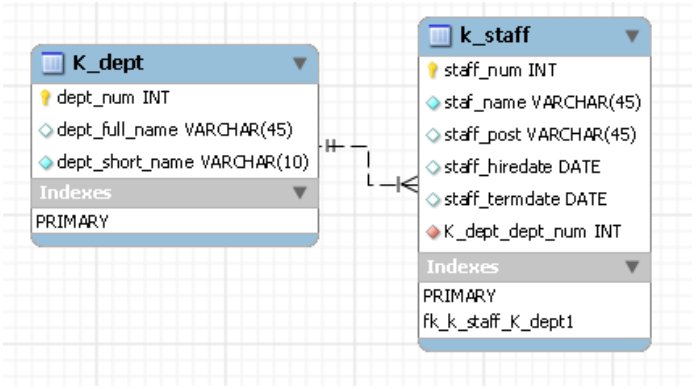
სურ. 2.12.

მივაქციოთ ყურადღება იმას, რომ პირველადი გასაღების შექმნის დროს ავტომატურად იქმნება მისი ინდექსიც, რომლის უპირველესი დანიშნულება არის მონაცემების დაჩქარებული ძებნა და მათი სწრაფი წვდომა.

ახლა დავაკავშიროთ ეს ცხრილები. ამისათვის ჯერ შევქმნათ კავშირი „მუშაობს“ ე.წ. „შვილობილ“ k_staff ცხრილსა და „მშობელ“ k_dept ცხრილს შორის, ხარისხით M:1, რისთვისაც ვიყენებთ პიქტოგრამას 

(პუნქტირით). მისი მეშვეობით იქმნება ე.წ. „არაიდენტიფიცირებული კავშირი“ ანუ ჩვეულებრივი გარე გასაღები. ამასთან, მშობელი ცხრილის პირველადი გასაღები დაემატება შვილობილი ცხრილის სვეტების სიას.

ამრიგად, ვაწკაპუნებთ პიქტოგრამაზე. შემდეგ ვაწკაპუნებთ k_staff შვილობილ ცხრილზე და ამის შემდეგ k_dept მშობელ ცხრილზე.

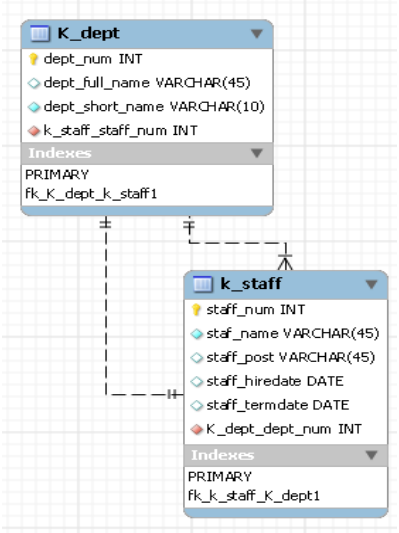


სურ. 2.13.

შედეგად ცხრილებს შორის ჩამოყალიბდა პუნქტირი „ერთი-მრავალთან“ კავშირის შესაბამისად. გარდა ამისა, შვილობილ ცხრილში წარმოიქმნა დამატებითი სვეტი, რომელსაც ავტომატურად მიენიჭა *k_dept_dept_num* სახელი, ხოლო ინდექსების ჯგუფში შეიქმნა ინდექსი გარე გასაღების მიხედვით.

ახლა ამ ცხრილებს შორის დავამატოთ კიდევ კავშირს „ხელმძღვანელობს“ 1:1. ამისათვის, ვირჩევთ 

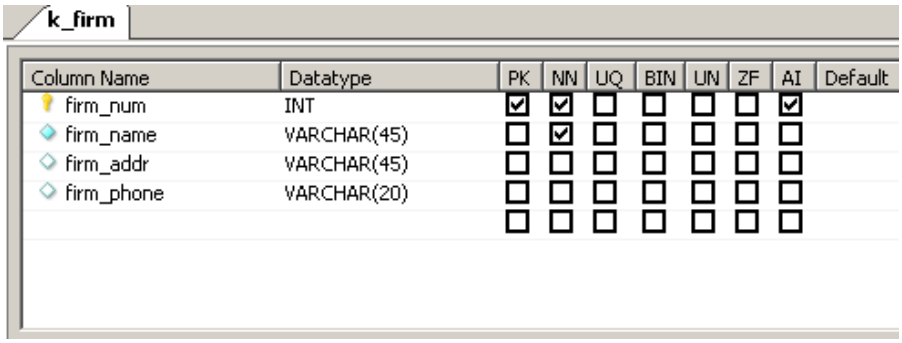
პიქტოგრამას, შემდეგ ვაწკაპუნებთ ჯერ k_dept ცხრილზე, ხოლო შემდეგ k_staff ცხრილზე.



სურ. 2.14.

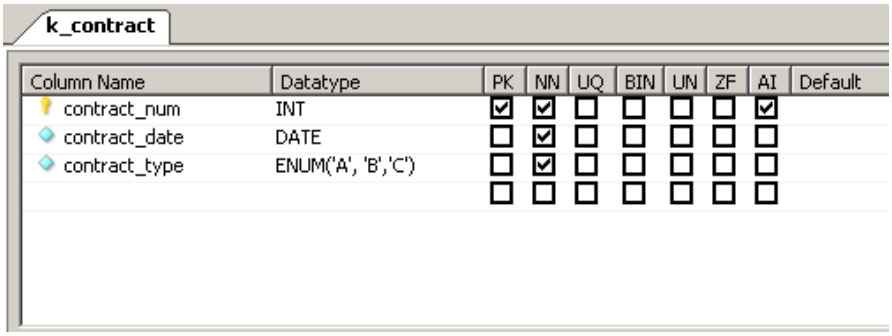
შედეგად k_dept ცხრილში დაემატა *k_staff_staff_num* სვეტი და აგრეთვე ინდექსი გარე გასაღების მიხედვით.

შევქმნათ k_firm ცხრილი:



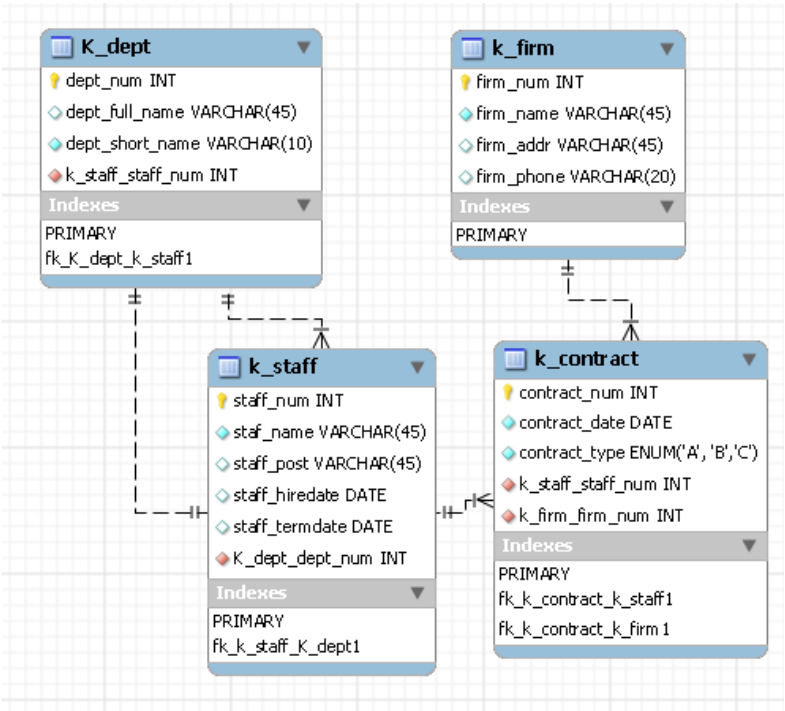
სურ. 2.15.

შევქმნათ k_contract ცხრილი. contract_type სვეტთან შევქმნათ შემდეგი ფორმატი: ასოები ‘A’, ‘B’, ‘C’.



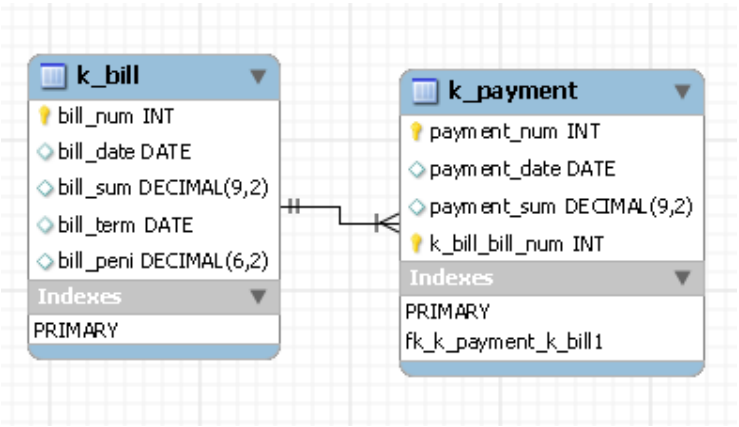
სურ. 2.16.

დავაკავშიროთ ცხრილი k_contract ცხრილებთან k_dept და k_staff კავშირით M:1.



სურ. 2.17.

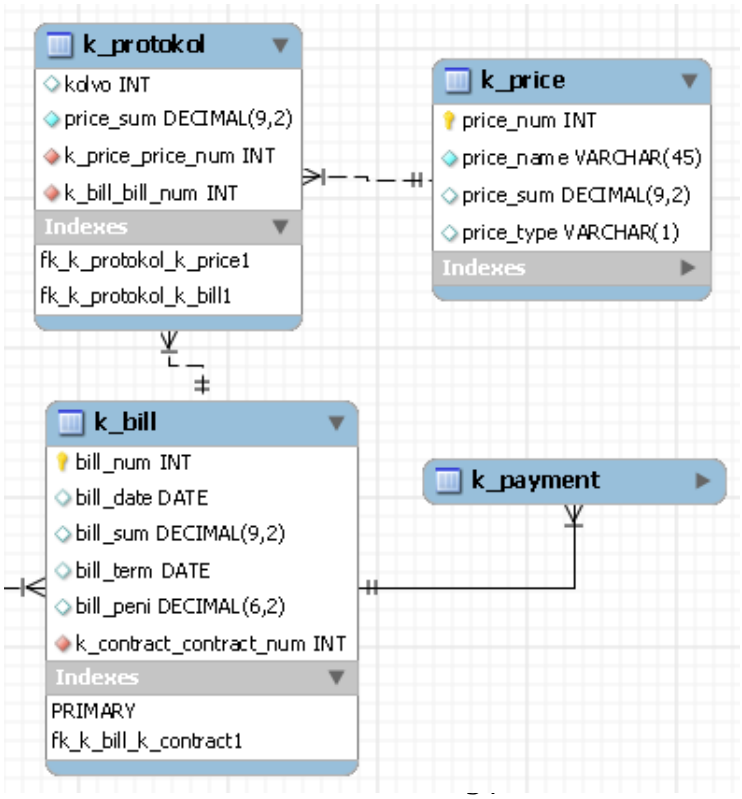
შემდეგ შევქმნათ ცხრილები k_bill (ანგარიში) და k_payment (გადახდა):



სურ. 2.18.

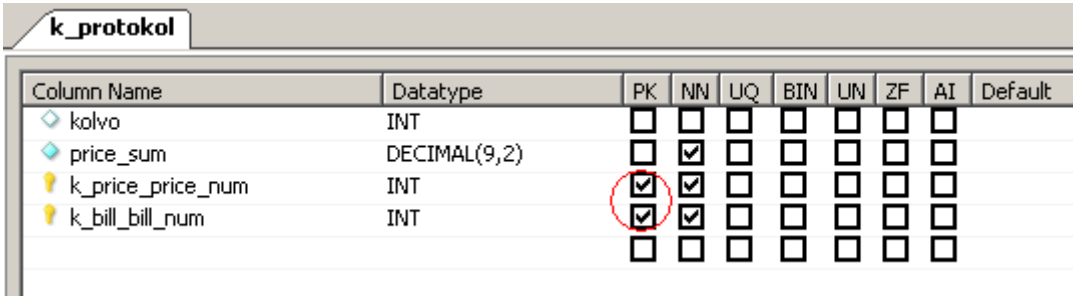
ვინაიდან k_payment არსთა სიმრავლე იყო „სუსტი“, შესაბამისად მას არ გააჩნია სრულფასოვანი პირველადი გასაღები და ყოველი გადახდა ერთმნიშვნელოვნად იდენტიფიცირდება შემდეგი ატრიბუტების ჯგუფთან (bill_num, payment_num). პირველადი გასაღების სახით ავღნიშნოთ ველი payment_num, ხოლო შემდგომ შევქმნათ მაიდენტიფიცირებელი კავშირი k_bill -სა და k_payment-ს შორის, რისთვისაც ვიყენებთ პიქტოგრამას  (მთლიანი ხაზით). ამასთან, ახალი სვეტი *k_bill_bill_num* ხდება არა მხოლოდ გარე გასაღები ცხრილისათვის k_payment, არამედ პირველადი გასაღების ნაწილიც.

შევქმნათ k_price და k_protocol ცხრილები.



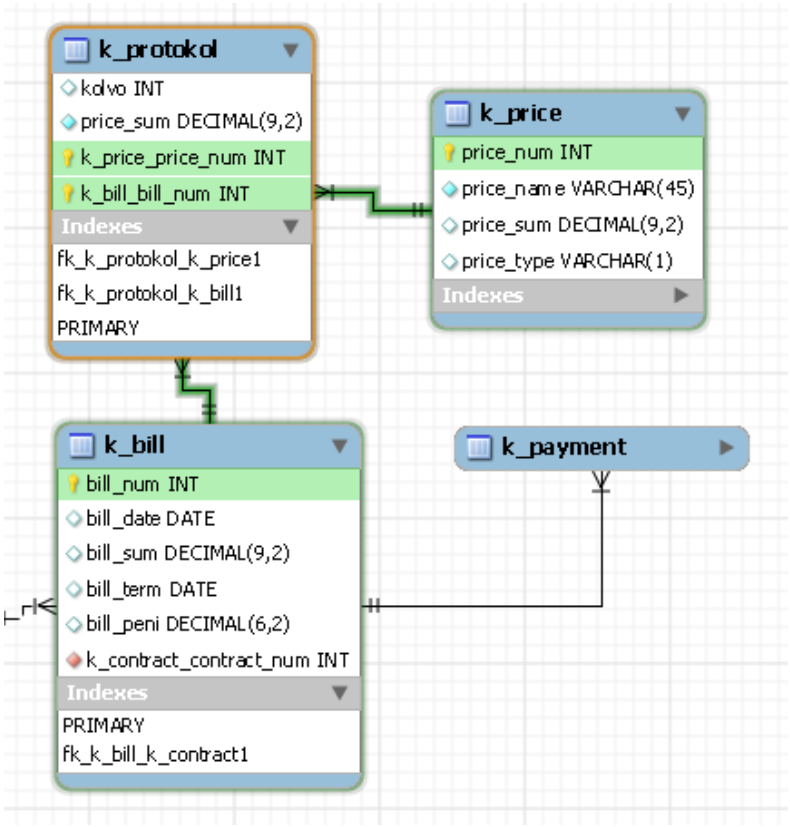
სურ. 2.19.

დავაკავშიროთ k_protocol ცხრილი k_bill და k_price ცხრილებთან. შედეგად გაჩნდება *k_price_price_num* და *k_bill_bill_num* სვეტები. ამ სვეტებიდან უნდა შევქმნათ პირველადი გასაღები, რისთვისაც ცხრილის სტრუქტურაში ვაყენებთ ალმებს PK სვეტში:



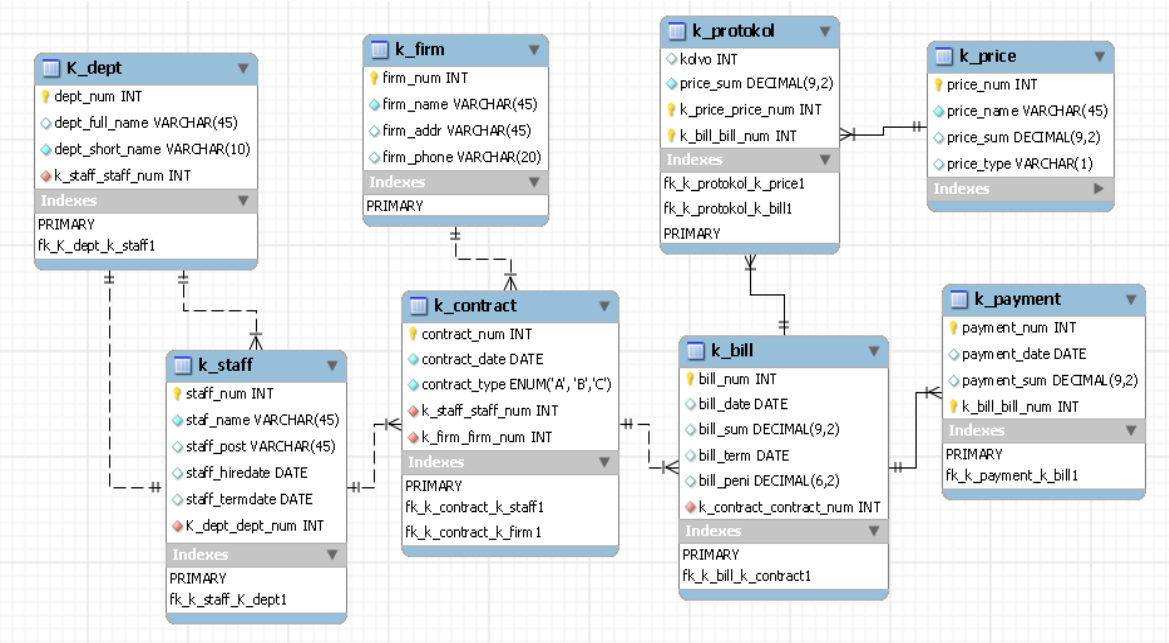
სურ. 2.20.

აღსანიშნავია, რომ k_protocol ცხრილის კავშირები თავისთავად გარდაიქმნენ მაიდენტიფიცირებელ კავშირებად, რადგან ორივე გარე გასაღები ჩავრთეთ პირველადი გასაღების შემადგენლობაში. სურათზე *k_price_price_num* და *k_bill_bill_num* სვეტების ახლოს გაჩნდა შესაბამისი პიქტოგრამები, ხოლო პუნქტირები გარდაიქმნენ მთლიან ხაზებად.



სურ. 2.21.

ამჯერად EER - დიაგრამას ექნება შემდეგი სახე:



სურ. 2.22.

დავალება: შევქმნათ MySQL WORKBENCH -ში ჩვენი ამოცანისათვის EER - დიაგრამა.

ლაბორატორიული სამუშაო 3
მონაცემთა განსაზღვრების ენა MySQL -ში

სამუშაოს მიზანი:

1. მონაცემთა ბაზასთან მუშაობა SQL -ის მეშვეობით;
2. მონაცემთა ბაზის შექმნა და მანიპულირება MySQL GUI Tools -ის გამოყენებით.

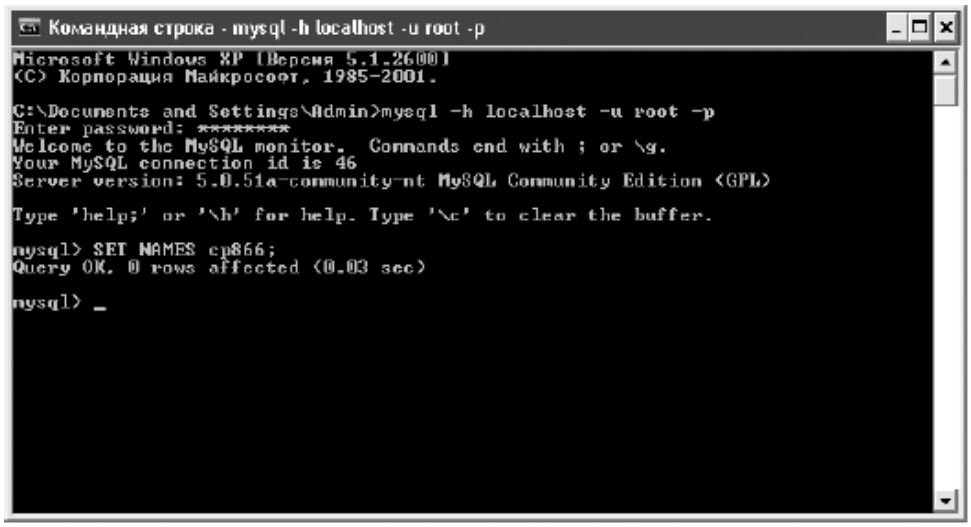
1. მონაცემთა ბაზასთან მუშაობა SQL -ის მეშვეობით

SQL - ბრძანებების შესასრულებლად გამოიყენება MySQL სერვერის მრავალრიცხოვანი კლიენტური დანართები, რომელთა შორის საყურადღებოა MySQL AB კომპანიის მიერ შემუშავებული უტილიტები: ბრძანებითი სტრიქონის უტილიტა mysql და გრაფიკული უტილიტა MySQL Query Browser.

MySQL Query Browser -ის კომპონენტები თვალსაჩინოდ არიან წარმოდგენილი. მონაცემთა ბაზასთან მუშაობის პროცესში ამ უტილიტას დახმარებით უშუალოდ არის შესაძლებელი მონაცემთა რედაქტირება (UPDATE ოპერატორის გარეშე), მოთხოვნებთან მუშაობა, მათი შენახვა ფაილში, შედეგების ექსპორტირება და მრავალი სხვა, მაგრამ მანამდე განვიხილოთ ბრძანებით სტრიქონში მუშაობის საკითხები.

1.1. SQL- ბრძანებების შესრულება

ბრძანებით სტრიქონში მუშაობის დაწყებამდე აუცილებელია MySQL სერვერთან მიერთება. შემდეგ ბრძანებით სტრიქონში შეგვყავს SQL- ბრძანების ტექსტი და ვაჭერთ Enter ღილაკს ბრძანების სერვერზე გასაგზავნად. შედეგი აისახება ეკრანზე (სურ. 3.1).

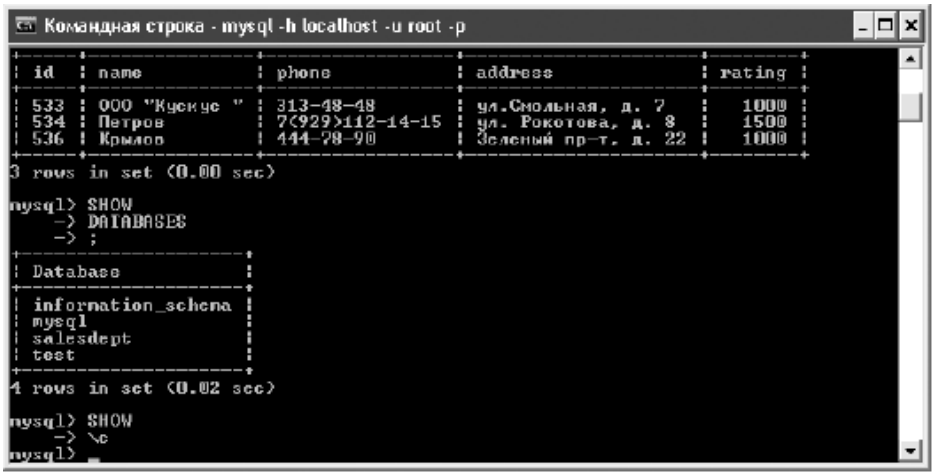


სურ. 3.1.

მონაცემთა ენობრივი კოდირებისათვის გამოიყენება ბრძანება SET NAMES, რომელიც უნდა გავიმეოროთ სერვერთან ყოველ მიერთებაზე. ეს ბრძანება მიუთითებს

სერვერს, რომ კლიენტური დანართი (mysql უტილიტა) გამოიყენებს Windows ბრძანებითი სტრიქონის კოდირებას. ბრძანების შესრულების შედეგების შესახებ შეტყობინება და მოთხოვნილი მონაცემები გამოიტანება ფანჯრის ბრძანებით სტრიქონში.

mysql უტილიტა საშუალებას იძლევა შევიტანოთ მრავალ-სტრიქონიანი ბრძანებები (მაგალითად,SHOW DATABASES), ვიდრე არ იქნება შეტანილი წერტილ-მძიმე ანუ დაბოლოების ნიშანი, რადგან Enter ღილაკის დაჭერით უტილიტა არ აგზავნის სერვერზე ბრძანებას, არამედ მხოლოდ აგრძელებს ბრძანების შეტანას. მრავალსტრიქონიანი ბრძანების შეტანის გაუქმებისათვის ვკრეფთ \c (სურ. 3.2).



სურ. 3.2.

თუ ვიყენებთ MySQL Query Browser, მაშინ კოდირების დაყენება არ არის საჭირო, რადგან პროგრამა მუშაობს UTF-8 კოდირებაში. საკმარისია შრიფტის ერთჯერადი შეცვლაც მოთხოვნათა არეში.

მიმდინარე ბრძანების შესასრულებლად ვაჭერთ Execute ღილაკს, ან Ctrl+Enter კლავიშების კომბინაციას.

1.2. მონაცემთა ბაზის შექმნა

მონაცემთა ბაზის შექმნისათვის სრულდება ბრძანება:

CREATE DATABASE <მონაცემთა ბაზის სახელი>;

მაგალითად,

CREATE DATABASE SalesDept;

თუ ახალი ბაზისათვის გვჭირდება MySQL-ის აწყობის შედეგად არსებული კოდირებისაგან განსხვავებული კოდირების დაყენება, რომელიც უსიტყვოდ იმუშავებს, ვიყენებთ ბრძანებას (character set) და/ან სიმბოლური მნიშვნელობების შედარების (სორტირების) წესებს:

CREATE DATABASE <მონაცემთა ბაზის სახელი>

CHARACTER SET <კოდირების სახელი>

COLLATE <შედარების წესების სახელი>;

მაგალითად, თუ გვინდა ახალ ბაზაში CP-1251 კოდირებაში მყოფი მონაცემების იმპორტირება:

CREATE DATABASE SalesDept

CHARACTER SET cp1251 COLLATE cp1251_general_ci;

კოდირების და/ან სიმბოლური მნიშვნელობების შედარების წესების ცვლილება შესაძლებელია შემდეგი ბრძანებების მეშვეობით

ALTER DATABASE <მონაცემთა ბაზის სახელი>

CHARACTER SET <კოდირების სახელი>

COLLATE <შედარების წესების სახელი>;

ამასთან უკვე არსებულ ცხრილებში კოდირება უცვლელი რჩება.

მონაცემთა ბაზის წაშლისათვის სრულდება ბრძანება

DROP DATABASE <მონაცემთა ბაზის სახელი>;

მოცემულ MySQL სერვერზე არსებული მონაცემთა ბაზების სიის ნახვა შესაძლებელია ბრძანებით:

SHOW DATABASES;

იმ შემთხვევაშიც კი, როცა არ გვაქვს შექმნილი არცერთი მონაცემთა ბაზა, სიაში მაინც დავინახავთ სამ სისტემურ მონაცემთა ბაზას:

• INFORMATION_SCHEMA – საინფორმაციო მონაცემთა ბაზა, სადაც ისურება ცნობები ყველა ბაზების და მათი ობიექტების შესახებ. ეს ბაზა მხოლოდ წაკითხვისათვის არის ხელმისაწვდომი.

• mysql – სამსახურეობრივი მონაცემთა ბაზა, სადაც ისურება ცნობები დარეგისტრირებულ მომხმარებლებზე, მათ უფლებებზე, საცნობარო ინფორმაცია და სხვ.

• test – ცარიელი მონაცემთა ბაზა, რომელიც საჭიროების შემთხვევაში გამოიყენება ან უბრალოდ შლიან.

1.3. მუშაობა ცხრილებთან

ცხრილის შექმნა

ცხრილის შექმნისათვის საჭიროა შემდეგი ბრძანების შესრულება.

ლისტინგი 3.1.

CREATE TABLE <ცხრილის სახელი>

(<1 სვეტის სახელი > <1 სვეტის ტიპი 1> [<1 სვეტის თვისებები>],

<2 სვეტის სახელი > <2 სვეტის ტიპი 1> [<2სვეტის თვისებები>],

...

[<ინფორმაცია გასაღებურ სვეტებზე და ინდექსებზე>])

[<ცხრილის ოპციური თვისებები>];

მაგალითისათვის განვიხილოთ ჩვენ მიერ უკვე შექმნილი მონაცემთა ბაზა SalesDept (გაყიდვების განყოფილება) და ახლა შევქმნათ სამი ცხრილი: Customers (კლიენტები), Products (საქონელი) და Orders (შეკვეთები).

ლისტინგი 3.2.

```
CREATE TABLE Customers
(id SERIAL,
name VARCHAR(100),
phone VARCHAR(20),
address VARCHAR(150),
rating INT,
PRIMARY KEY (id))
ENGINE InnoDB CHARACTER SET utf8;
```

ლისტინგი 3.3.

```
CREATE TABLE Products
(id SERIAL,
description VARCHAR(100),
details TEXT,
price DECIMAL(8,2),
PRIMARY KEY (id))
ENGINE InnoDB CHARACTER SET utf8;
```

ლისტინგი 3.4.

```
CREATE TABLE Orders
(id SERIAL,
date DATE,
product_id BIGINT UNSIGNED NOT NULL,
qty INT UNSIGNED,
amount DECIMAL(10,2),
customer_id BIGINT UNSIGNED,
PRIMARY KEY (id),
FOREIGN KEY (product_id) REFERENCES Products (id)
ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (customer_id) REFERENCES Customers (id)
ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE InnoDB CHARACTER SET utf8;
```

დავალება: სადემონსტრაციოდ შევქმნათ სასწავლო მონაცემთა ბაზა **northwind**

```
DROP DATABASE IF EXISTS northwind;
CREATE DATABASE IF NOT EXISTS northwind;

USE northwind;

CREATE TABLE `Categories` (
  `CategoryID` INTEGER NOT NULL AUTO_INCREMENT,
  `CategoryName` VARCHAR(15) NOT NULL,
  `Description` MEDIUMTEXT,
```

```
   `Picture` LONGBLOB,
   CONSTRAINT `PK_Categories` PRIMARY KEY (`CategoryID`)
);

CREATE INDEX `CategoryName` ON `Categories` (`CategoryName`);


CREATE TABLE `CustomerCustomerDemo` (
   `CustomerID` VARCHAR(5) NOT NULL,
   `CustomerTypeID` VARCHAR(10) NOT NULL,
   CONSTRAINT `PK_CustomerCustomerDemo` PRIMARY KEY (`CustomerID`,
`CustomerTypeID`)
);


CREATE TABLE `CustomerDemographics` (
   `CustomerTypeID` VARCHAR(10) NOT NULL,
   `CustomerDesc` MEDIUMTEXT,
   CONSTRAINT `PK_CustomerDemographics` PRIMARY KEY (`CustomerTypeID`)
);

CREATE TABLE `Customers` (
   `CustomerID` VARCHAR(5) NOT NULL,
   `CompanyName` VARCHAR(40) NOT NULL,
   `ContactName` VARCHAR(30),
   `ContactTitle` VARCHAR(30),
   `Address` VARCHAR(60),
   `City` VARCHAR(15),
   `Region` VARCHAR(15),
   `PostalCode` VARCHAR(10),
   `Country` VARCHAR(15),
   `Phone` VARCHAR(24),
   `Fax` VARCHAR(24),
   CONSTRAINT `PK_Customers` PRIMARY KEY (`CustomerID`)
);

CREATE INDEX `City` ON `Customers` (`City`);
CREATE INDEX `CompanyName` ON `Customers` (`CompanyName`);
CREATE INDEX `PostalCode` ON `Customers` (`PostalCode`);
CREATE INDEX `Region` ON `Customers` (`Region`);

CREATE TABLE `Employees` (
   `EmployeeID` INTEGER NOT NULL AUTO_INCREMENT,
   `LastName` VARCHAR(20) NOT NULL,
   `FirstName` VARCHAR(10) NOT NULL,
```

```
    `Title` VARCHAR(30),
    `TitleOfCourtesy` VARCHAR(25),
    `BirthDate` DATETIME,
    `HireDate` DATETIME,
    `Address` VARCHAR(60),
    `City` VARCHAR(15),
    `Region` VARCHAR(15),
    `PostalCode` VARCHAR(10),
    `Country` VARCHAR(15),
    `HomePhone` VARCHAR(24),
    `Extension` VARCHAR(4),
    `Photo` LONGBLOB,
    `Notes` MEDIUMTEXT NOT NULL,
    `ReportsTo` INTEGER,
    `PhotoPath` VARCHAR(255),
     `Salary` FLOAT,
    CONSTRAINT `PK_Employees` PRIMARY KEY (`EmployeeID`)
);

CREATE INDEX `LastName` ON `Employees` (`LastName`);
CREATE INDEX `PostalCode` ON `Employees` (`PostalCode`);


CREATE TABLE `EmployeeTerritories` (
    `EmployeeID` INTEGER NOT NULL,
    `TerritoryID` VARCHAR(20) NOT NULL,
    CONSTRAINT `PK_EmployeeTerritories` PRIMARY KEY (`EmployeeID`, `TerritoryID`)
);
CREATE TABLE `Order Details` (
    `OrderID` INTEGER NOT NULL,
    `ProductID` INTEGER NOT NULL,
    `UnitPrice` DECIMAL(10,4) NOT NULL DEFAULT 0,
    `Quantity` SMALLINT(2) NOT NULL DEFAULT 1,
    `Discount` REAL(8,0) NOT NULL DEFAULT 0,
    CONSTRAINT `PK_Order Details` PRIMARY KEY (`OrderID`, `ProductID`)
);
CREATE TABLE `Orders` (
    `OrderID` INTEGER NOT NULL AUTO_INCREMENT,
    `CustomerID` VARCHAR(5),
    `EmployeeID` INTEGER,
    `OrderDate` DATETIME,
    `RequiredDate` DATETIME,
    `ShippedDate` DATETIME,
    `ShipVia` INTEGER,
    `Freight` DECIMAL(10,4) DEFAULT 0,
```

```
    `ShipName` VARCHAR(40),
    `ShipAddress` VARCHAR(60),
    `ShipCity` VARCHAR(15),
    `ShipRegion` VARCHAR(15),
    `ShipPostalCode` VARCHAR(10),
    `ShipCountry` VARCHAR(15),
    CONSTRAINT `PK_Orders` PRIMARY KEY (`OrderID`)
);

CREATE INDEX `OrderDate` ON `Orders` (`OrderDate`);
CREATE INDEX `ShippedDate` ON `Orders` (`ShippedDate`);
CREATE INDEX `ShipPostalCode` ON `Orders` (`ShipPostalCode`);


CREATE TABLE `Products` (
    `ProductID` INTEGER NOT NULL AUTO_INCREMENT,
    `ProductName` VARCHAR(40) NOT NULL,
    `SupplierID` INTEGER,
    `CategoryID` INTEGER,
    `QuantityPerUnit` VARCHAR(20),
    `UnitPrice` DECIMAL(10,4) DEFAULT 0,
    `UnitsInStock` SMALLINT(2) DEFAULT 0,
    `UnitsOnOrder` SMALLINT(2) DEFAULT 0,
    `ReorderLevel` SMALLINT(2) DEFAULT 0,
    `Discontinued` BIT NOT NULL DEFAULT 0,
    CONSTRAINT `PK_Products` PRIMARY KEY (`ProductID`)
);

CREATE INDEX `ProductName` ON `Products` (`ProductName`);


CREATE TABLE `Region` (
    `RegionID` INTEGER NOT NULL,
    `RegionDescription` VARCHAR(50) NOT NULL,
    CONSTRAINT `PK_Region` PRIMARY KEY (`RegionID`)
);

CREATE TABLE `Shippers` (
    `ShipperID` INTEGER NOT NULL AUTO_INCREMENT,
    `CompanyName` VARCHAR(40) NOT NULL,
    `Phone` VARCHAR(24),
    CONSTRAINT `PK_Shippers` PRIMARY KEY (`ShipperID`)
);

CREATE TABLE `Suppliers` (
```

```
  `SupplierID` INTEGER NOT NULL AUTO_INCREMENT,
  `CompanyName` VARCHAR(40) NOT NULL,
  `ContactName` VARCHAR(30),
  `ContactTitle` VARCHAR(30),
  `Address` VARCHAR(60),
  `City` VARCHAR(15),
  `Region` VARCHAR(15),
  `PostalCode` VARCHAR(10),
  `Country` VARCHAR(15),
  `Phone` VARCHAR(24),
  `Fax` VARCHAR(24),
  `HomePage` MEDIUMTEXT,
  CONSTRAINT `PK_Suppliers` PRIMARY KEY (`SupplierID`)
);

CREATE INDEX `CompanyName` ON `Suppliers` (`CompanyName`);
CREATE INDEX `PostalCode` ON `Suppliers` (`PostalCode`);

CREATE TABLE `Territories` (
  `TerritoryID` VARCHAR(20) NOT NULL,
  `TerritoryDescription` VARCHAR(50) NOT NULL,
  `RegionID` INTEGER NOT NULL,
  CONSTRAINT `PK_Territories` PRIMARY KEY (`TerritoryID`)
);
```

1.4. გასაღებური სვეტები და ინდექსები

გასაღებური სვეტების განსაზღვრისათვის გამოიყენება შემდეგი კონსტრუქციები:

• [CONSTRAINT <გასაღების სახელი>] PRIMARY KEY (<სვეტების სია>).

შეზღუდვებისა და ინდექსის შესაქმნელად სახელის მითითება აუცილებელი არ არის. ამ შემთხვევაში მათი სახელი ავტომატურად გენერირდება.

• INDEX [<ინდექსის სახელი>] (<სვეტების სია>).

გასაღებური სიტყვის UNIQUE ნაცვლად შეიძლება მათი სინონიმების UNIQUE INDEX ან UNIQUE KEY გამოყენება.

• FULLTEXT [<ინდექსის სახელი>] (<სვეტების სია>).

ქმნის სრულტექსტიან ინდექსს, რომელიც უზრუნველყოფს დაჩქარებულ ძებნას სიმბოლური სვეტების მნიშვნელობების მიხედვით (ტიპები CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT და LONGTEXT) მათი სიგრძისგან დამოუკიდებლად. სრულტექსტიანი ინდექსი შეიძლება შეიქმნას მხოლოდ MyISAM ტიპის ცხრილებში. გასაღებური სიტყვის FULLTEXT ნაცვლად შეიძლება მათი სინონიმების FULLTEXT INDEX ან FULLTEXT KEY გამოყენება.

• SPATIAL [<ინდექსის სახელი>] (<სვეტების სია>).

ქმნის ინდექსს ძემნისათვის სივრცული და გეოგრაფული მნიშვნელობების მიხედვით.

• [CONSTRAINT <გარე გასაღების სახელი>].

FOREIGN KEY [<ინდექსის სახელი>] (<სვეტების სია>)

REFERENCES <მშობელი ცხრილის სახელი>

(<პირველადი გასაღების სვეტების სია მშობელი ცხრილიდან>)

[<კავშირის მთლიანობის მხარდაჭერის წესები>]

სვეტები, რომლებიც შეადგენენ გარე გასაღებს, უნდა იყოს ანალოგური ტიპის, როგორიც გააჩნია მშობელი ცხრილის პირველად გასაღებს. TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB და LONGTEXT ტიპის სვეტები არ შეიძლება შედიოდნენ გარე გასაღებში. ამასთან, გარე გასაღების სახელი და ინდექსის სახელის მითითება არ არის აუცილებელი. ისინი ავტომატურად გენერირდებიან.

მშობელი ცხრილიდან წაშლის და ცვლილებების ოპერაციებისათვის უნდა მიუთითოთ თუ რომელ კავშირის მთლიანობის მხარდაჭერის წესების გამოყენება მოხდება:

• ON DELETE CASCADE;

• ON DELETE SET NULL ;

• ON DELETE RECTRICT ან ON DELETE NO ACTION.

• ON UPDATE CASCADE;

• ON UPDATE SET NULL;

• ON UPDATE RECTRICT ან ON UPDATE NO ACTION - (MySQL-ში ეს გამოსახულებები სინონიმებს წარმოადგენენ.

1.5. ცხრილის ოპციური თვისებები

ცხრილის შექმნის დროს აუცილებელი არ არის ცხრილის ოპციური თვისებების მითითება. მიუხედავად ამისა, განვიხილოთ რამდენიმე ოპციური თვისება, რომელიც ცხრილისათვის მოიცემა.

• ENGINE <ცხრილის ტიპი>.

MySQL - ში ძირითადად განირჩევა InnoDB და MyISAM ტიპები. InnoDB ცხრილები უზრუნველყოფენ ცალკეული სტრიქონების ტრანზაკციებსა და ბლოკირებებს, რაც ამაღლებს მონაცემთა ცვლილებების ოპერაციების წარმადობას მრავალმომხმარებლიან რეჟიმში. MyISAM ცხრილები ვერ აერთიანებენ რამდენიმე ოპერაციას ერთიან ტრანზაკციაში, მაგრამ გააჩნიათ საძიებო მოთხოვნის შესრულების მაღალი სიჩქარე და სისტემური რესურსების მცირე დატვირთვა.

ჩვენი მაგალითის ყველა ცხრილი შექმნილია ENGINE InnoDB პარამეტრით. გასაღებური სიტყვის ENGINE ნაცვლად შეიძლება ვიხმაროთ მისი სინონიმი TYPE.

• AUTO_INCREMENT <საწყისი მნიშვნელობა>.

ეს თვისება ნუმერაციის სანახველი მნიშვნელობიდან დაწყების საშუალებას იძლევა.

• CHARACTER SET <კოდირების სახელი>.

ჩვენი მაგალითის ყველა ცხრილი შექმნილია CHARACTER SET utf8 პარამეტრით. თუ კოდირება არ არის მოცემული, მაშინ მონაცემთა ბაზისათვის გამოიყენება კოდირება, რომელიც ინსტალაციის დროს იყო დადგენილი.

• COLLATE <შედარების წესის სახელი>.

თუ შედარების წესი არ არის მოცემული, მაშინ მონაცემთა ბაზისათვის უსიტყვოდ გამოიყენება შედარების წესის, რომელიც ინსტალაციის დროს იყო დადგენილი.

• CHECKSUM 1.

მოცემული პარამეტრი შეიცავს MyISAM ტიპის ცხრილისათვის სტრიქონების საკონტროლო ჯამის შემოწმების ფუნქციას.

• COMMENT 'კომენტარის ტექსტი'.

ცხრილის ნებისმიერი ტექსტური აღწერა სიგრძით 60 სიმბოლომდე.

1.6. ცხრილის სტრუქტურის შეცვლა

ცხრილის სტრუქტურის შეცვლისათვის გამოიყენება ბრძანება ALTER TABLE.

• სვეტის დამატება:

ALTER TABLE <ცხრილის სახელი>
ADD <სვეტის სახელი> <სვეტის ტიპი> [<სვეტის თვისებები>]
[FIRST ან AFTER <წინამორბედი სვეტის სახელი>];

მაგალითად,

ALTER TABLE Products
ADD id BIGINT AUTO_INCREMENT, ADD PRIMARY KEY (id);

• პირველადი გასაღების დამატება:

ALTER TABLE <ცხრილის სახელი>>
ADD [CONSTRAINT <გასაღების სახელი>]
PRIMARY KEY (<სვეტების სია>);

მაგალითად,

ALTER TABLE Orders ADD PRIMARY KEY (id);

• გარე გასაღების დამატება:

ALTER TABLE <ცხრილის სახელი>
ADD [CONSTRAINT <გარე გასაღების სახელი>]
FOREIGN KEY [<ინდექსის სახელი>] (<სვეტების სია>)
REFERENCES <მშობელი ცხრილის სახელი>
(<მშობელი ცხრილის პირველადი გასაღების სვეტების სია>)
[<კავშირის მთლიანობის მხარდაჭერის წესები>];

მაგალითად,

ALTER TABLE Orders
ADD FOREIGN KEY (customer_id) REFERENCES Customers (id)
ON DELETE RESTRICT ON UPDATE CASCADE);

• ჩვეულებრივი ინდექსის დამატება:

ALTER TABLE <ცხრილის სახელი>
ADD INDEX [<ინდექსის სახელი>] (<სვეტების სია>);

უნიკალური ინდექსის დამატება:

ALTER TABLE <ცხრილის სახელი>
ADD [CONSTRAINT <შეზღუდვის სახელი>]
UNIQUE (<სვეტების სია>);

სრულტექსტიანი ინდექსის დამატება:

ALTER TABLE <ცხრილის სახელი>
ADD FULLTEXT [<ინდექსის სახელი>] (<სვეტების სია>);

• ცხრილის სვეტის შეცვლა:

ALTER TABLE <ცხრილის სახელი>
CHANGE <სვეტის ძველი სახელი >
<სვეტის ახალი სახელი>
<სვეტის ახალი ტიპი> [<სვეტის თვისებები>]
[FIRST ან AFTER <წინამდებარე სვეტის სახელი>];

სვეტისათვის AUTO_INCREMENT თვისების მინიჭებისათვის აუცილებელია ან ერთდროულად ამასთან ან წინასწარ შეიქმნას ამ სვეტის ინდექსი.

გადარქმევის გარეშე სვეტის აღწერის შეცვლა:

ALTER TABLE <ცხრილის სახელი>
MODIFY <სვეტის სახელი>
<სვეტის ახალი ტიპი> [<სვეტის თვისებები>]
[FIRST ან AFTER <წინამდებარე სვეტის სახელი>];

სვეტისათვის მნიშვნელობის უსიტყვოდ დადგენა:

ALTER TABLE <ცხრილის სახელი>
ALTER <სვეტის სახელი>
SET DEFAULT <მნიშვნელობა უსიტყვოდ>;

მაგალითად,
ALTER TABLE Products
ALTER warehouse SET DEFAULT 'Склад № 1';

სვეტისათვის მნიშვნელობის უსიტყვოდ ამოგდება:

ALTER TABLE <ცხრილის სახელი>
ALTER <სვეტის სახელი> DROP DEFAULT;
მაგალითად,
ALTER TABLE Products ALTER warehouse DROP DEFAULT;

• ცხრილიდან სვეტის ამოდგება:

ALTER TABLE <ცხრილის სახელი> DROP <სვეტის სახელი>;

მაგალითად,

ALTER TABLE Products DROP warehouse;

• პირველადი გასაღების ამოგდება:

ALTER TABLE <ცხრილის სახელი> DROP PRIMARY KEY;

მაგალითად,

ALTER TABLE Orders DROP PRIMARY KEY;

• გარე გასაღების ამოგდება:

ALTER TABLE <ცხრილის სახელი>

DROP FOREIGN KEY <გარე გასაღების სახელი>;

მაგალითად,

ALTER TABLE Orders DROP FOREIGN KEY orders_ibfk_1;

• ინდექსის ამოგდება

ALTER TABLE <ცხრილის სახელი> DROP INDEX <ინდექსის სახელი>;

მაგალითად,

ALTER TABLE Customers DROP INDEX name;

• MyISAM ტიპის ცხრილში არაუნიკალური ინდექსების ჩართვა და გამორთვა:

ALTER TABLE <ცხრილის სახელი> DISABLE KEYS;

ALTER TABLE <ცხრილის სახელი> ENABLE KEYS;

• ცხრილის გადარქმევა:

ALTER TABLE <ცხრილის სახელი> RENAME <ცხრილის ახალი სახელი>;

• ცხრილში სტრიქონების დალაგება:

ALTER TABLE <ცხრილის სახელი>

ORDER BY <1 სვეტის სახელი> [ASC или DESC],

[<2 სვეტის სახელი2> [ASC ან DESC],...];

InnoDB ტიპის ცხრილებში, რომლებსაც გააჩნიათ პირველადი გასაღები ან უნიკალური ინდექსი, დაუშვებელია მნიშვნელობა NOT NULL UNIQUE.

• ცხრილის ოპციური თვისებების მოცემა ხდება ბრძანებით:

ALTER TABLE <ცხრილის სახელი> <ცხრილის ოპციური თვისება>;

მაგალითად, ცხრილის ტიპის შეცვლა შესაძლებელია ბრძანებით:

ALTER TABLE <ცხრილის სახელი> ENGINE <ცხრილის ახალი ტიპი>;

• კოდირებისა და სიმბოლური მნიშვნელობების შედარების წესის ცვლილებისათვის გამოიყენება შემდეგი ბრძანება:

ALTER TABLE <ცხრილის სახელი>

CHARACTER SET <კოდირების სახელი>

[COLLATE <შედარების წესის სახელი>];

მაგალითად,

ALTER TABLE Products CHARACTER SET cp1251;

1.7. ცხრილებთან მუშაობის სხვა ბრძანებები

თუ გვინდა მივიღოთ დეტალური ინფორმაცია კონკრეტული ცხრილის შესახებ, მაშინ შეგვიძლია გამოვიყენოთ შემდეგი ბრძანებები:

DESCRIBE <ცხრილის სახელი>;

ან

SHOW CREATE TABLE <ცხრილის სახელი>;

მაგალითად,

DESCRIBE Customers;

შედეგად მივიღებთ ცხრილის სტრუქტურის აღწერას

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
name	varchar(100)	YES		NULL	
phone	varchar(20)	YES		NULL	
address	varchar(150)	YES		NULL	
rating	int(11)	YES		NULL	

სადაც თითოეული სვეტი აისახება შემდეგი მახასიათებლების მეშვეობით:

• Field – სვეტის სახელი;

• Type – სვეტის ტიპი;

• Null – მიუთითებს სვეტის განუსაზღვრელი მნიშნელობის არსებობაზე:

YES – უშვებს, NO – არ უშვებს;

• Key – მიუთითებს სვეტის კუთვნილებას გასაღებების და ინდექსების მიმართ:

– PRI – სვეტი შედის პირველად გასაღებში ან, თუ ცხრილში არ არის პირველადი გასაღები, უნიკალურ ინდექსში, რომელიც არ უშვებს განუსაზღვრელ მნიშვნელობას;

– UNI, MUL – სვეტი არის ინდექსის პირველი სვეტი;

• Default – სვეტის უსიტყვო მნიშვნელობა;

• Extra – დამატებითი მნიშვნელობა.

ბრძანებას SHOW CREATE TABLE გამოაქვს ცხრილის ყველა პარამეტრის შესახებ სრული ინფორმაცია. მაგალითად,

SHOW CREATE TABLE Customers;

შედეგი ნაჩვენებია ცხრილში .

Table	Create Table
Customers	CREATE TABLE `customers` (`id` bigint(20) unsigned NOT NULL auto_increment, `name` varchar(100) default NULL, `phone` varchar(20) default NULL, `address` varchar(150) default NULL, `rating` int(11) default NULL, PRIMARY KEY (`id`), UNIQUE KEY `id` (`id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8

მიმდინარე მონაცემთა ბაზაში ცხრილების სიის ნახვა შესაძლებელია ბრძანებით

SHOW TABLES;

მაგალითად, ჩვენი მონაცემთა ბაზის შემთხვევაში SalesDept, სადაც გვაქვს სამი ცხრილი – Customers, Products და Orders, ბრძანებით SHOW TABLES მივიღებთ შემდეგ ცხრილს.

Tables_in_salesdept
customers
orders
products

ცხრილის წაშლა, როდესაც ხდება ცხრილში არსებული ყველა მონაცემის განადგურება, სრულდება ბრძანებით:

DROP TABLE <ცხრილის სახელი>;

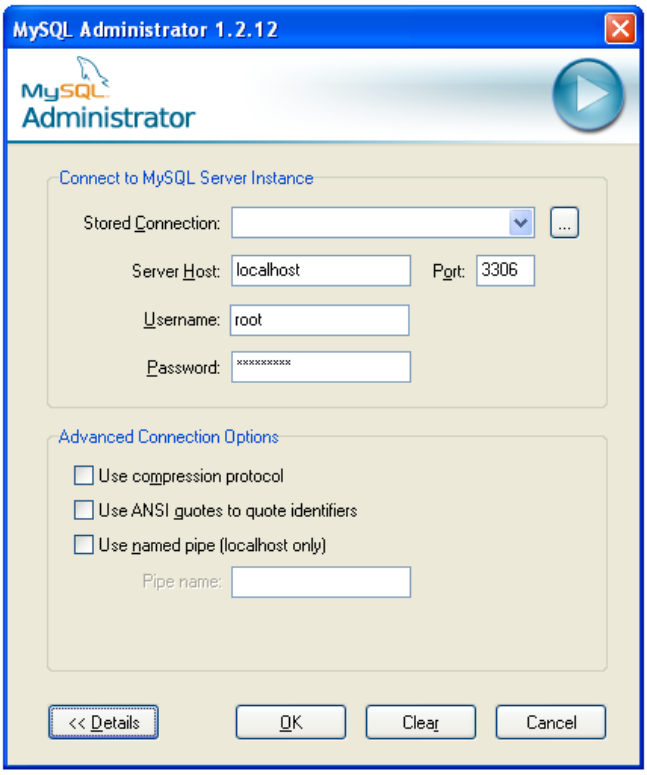
2. მონაცემთა ბაზის შექმნა და მანიპულირება MySQL GUI Tools -ის გამოყენებით

2.1. მუშაობა MySQL Administrator-თან

MySQL Administrator -ის გაშვებისათვის ვასრულებთ Start-მენიუდან ბრძანებებს:

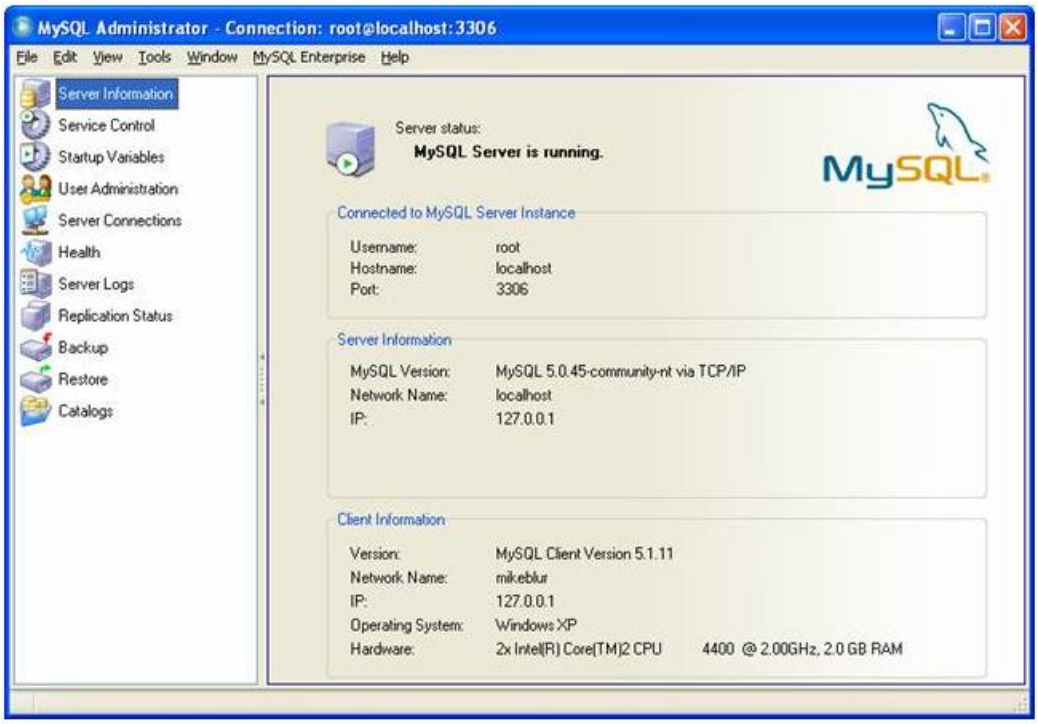
„Start > Program Files > MySQL > MySQL Administrator“

იხსნება MySQL Administrator -ის გაშვების ფანჯარა, სადაც მოთხოვნილი ველების შევსების შემდეგ ვაჭერთ ღილაკს OK.



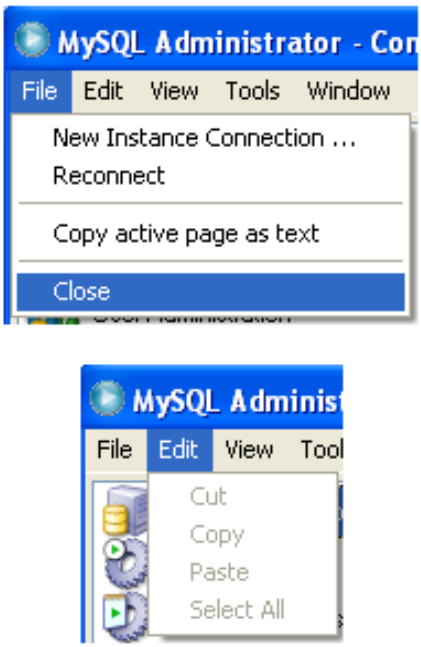
სურ. 3.3.

იხსნება MySQL Administrator -ის ფანჯარა.

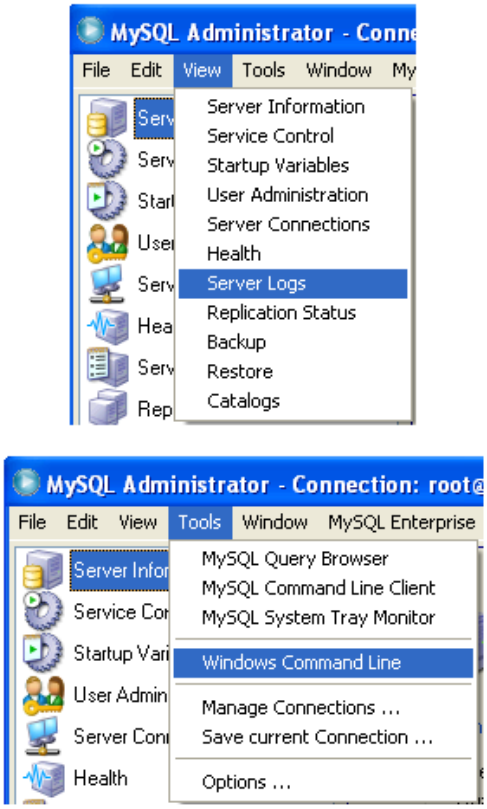


სურ. 3.4.

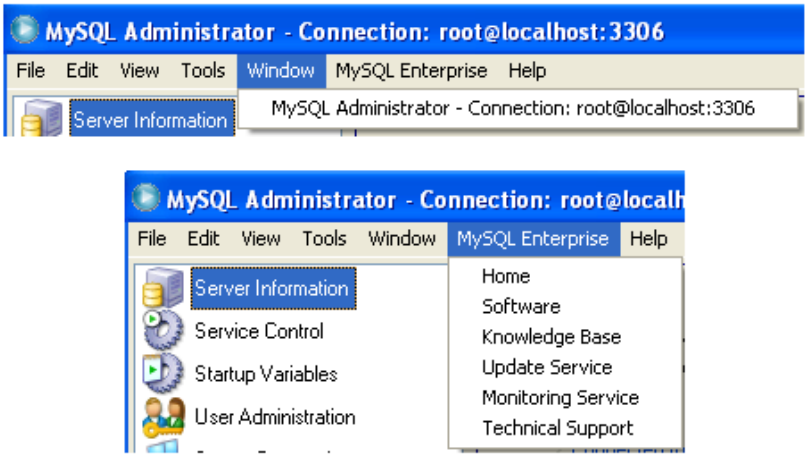
ქვემოთ მოყვანილია MySQL Administrator ფანჯრის მენიუს ბრძანებათა პუნქტები.


MySQL Administrator - Cor
File
Edit
View
Tools
Window
New Instance Connection ...
Reconnect
Copy active page as text
Close
MySQL Adminis
File
Edit
View
Tool
Cut
Copy
Paste
Select All

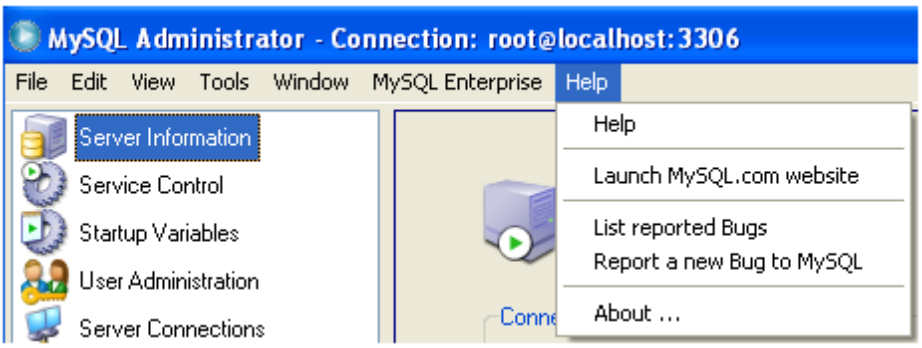
სურ. 3.5.


MySQL Administrator - Conne
File
Edit
View
Tools
Window
My
Server Information
Service Control
Startup Variables
User Administration
Server Connections
Health
Server Logs
Replication Status
Backup
Restore
Catalogs
MySQL Administrator - Connection: root@
File
Edit
View
Tools
Window
MySQL Enterprise
Server Infor
Service Cor
Startup Vari
User Admin
Server Con
Health
MySQL Query Browser
MySQL Command Line Client
MySQL System Tray Monitor
Windows Command Line
Manage Connections ...
Save current Connection ...
Options ...

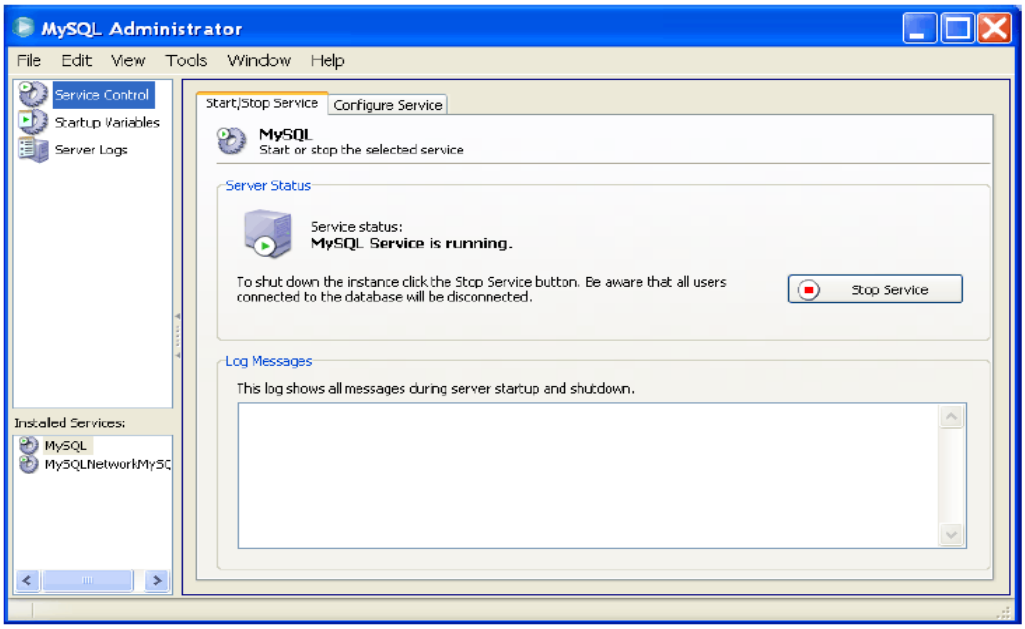
სურ. 3.6.



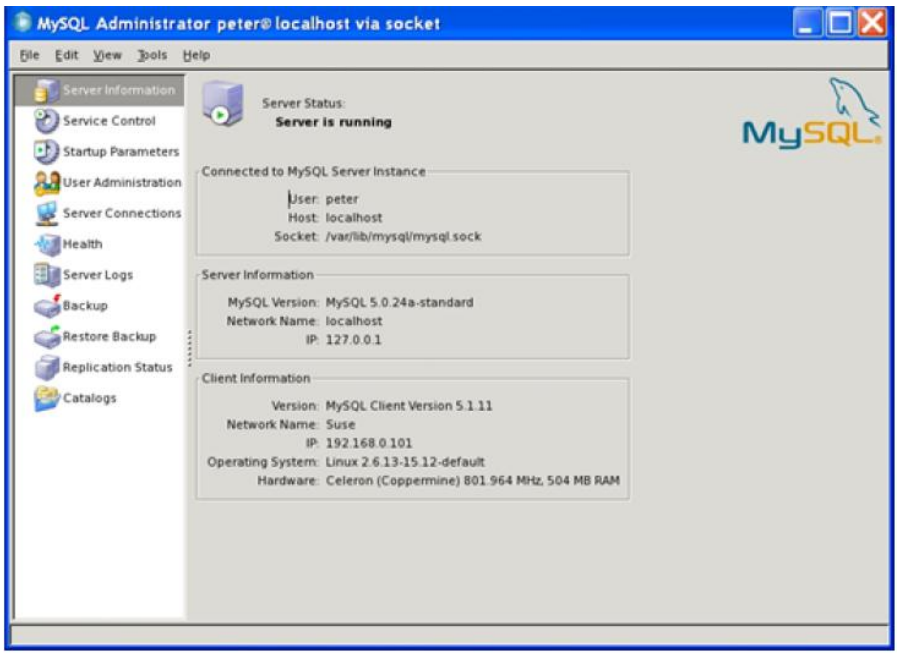
სურ. 3.7.



სურ. 3.8.



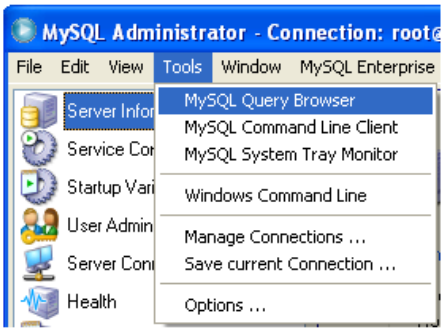
სურ. 3.9.



სურ. 3.10.

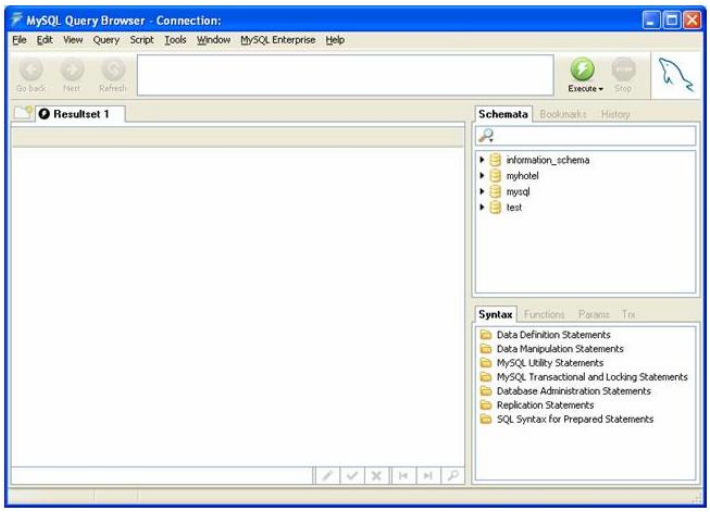
2.2. მუშაობა MySQL Query Browser-თან

MySQL Administrator ფანჯრის მენიუს ბრძანებებით: „Tools > MySQL Query Browser“ იხსნება MySQL Query Browser .



სურ. 3.11.

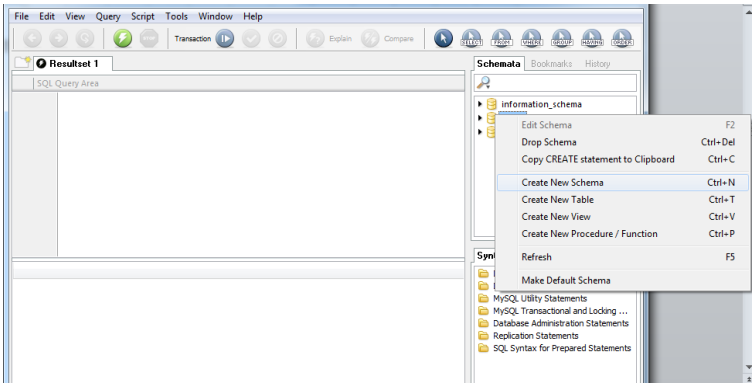
MySQL Query Browser -ის გახსნა შესაძლებელია აგრეთვე Start მენიუდანაც: „Start > Program Files > MySQL > MySQL Query Browser“.



სურ. 3.12.

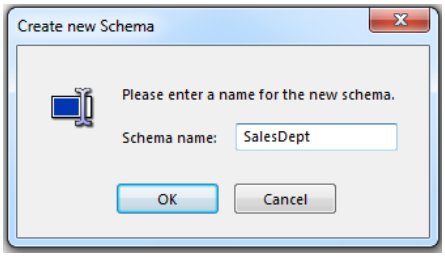
არსებობს მონაცემთა ბაზის შექმნის რამდენიმე მიდგომა.

განყოფილებაში Schemata ცარიელ ადგილას მაუსის მარჯვენა ღილაკზე დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ ბრძანებას „Create New Schema“.



სურ. 3.13.

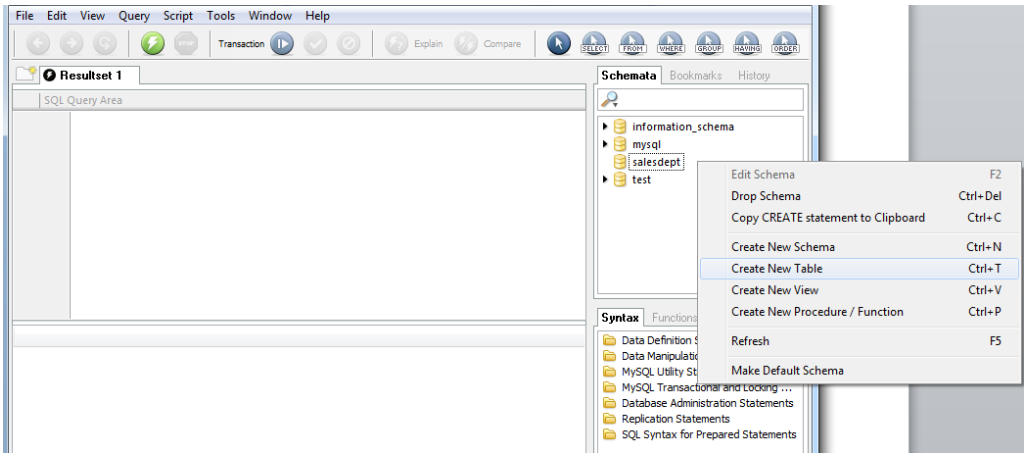
გამოდის ფანჯარა, სადაც მონაცემთა ბაზას ანუ სქემას ვარქმევთ სახელს.



სურ. 3.14.

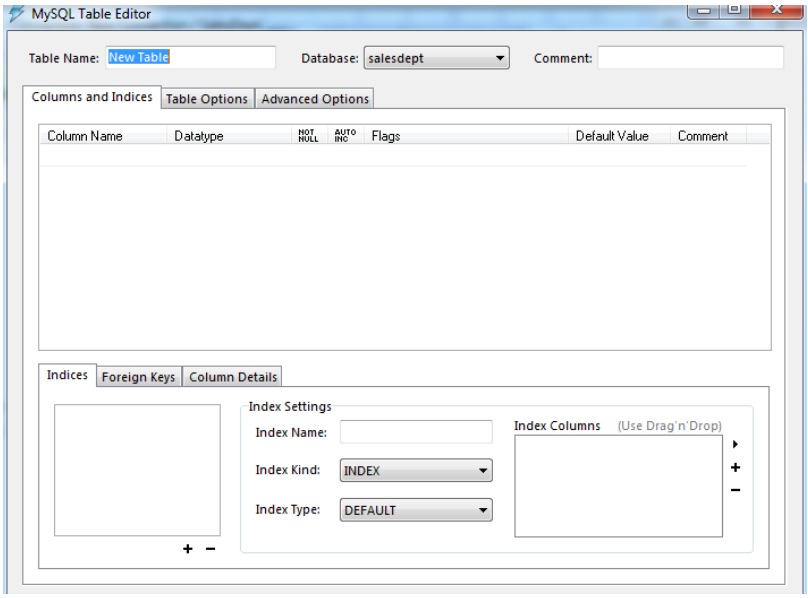
იმისათვის, რომ შექმნილი მონაცემთა ბაზისათვის განმეორებითი ავტორიზაციის გარეშე შევასრულოთ მოთხოვნები, სქემაზე მაუსის მარჯვენა ღილაკზე დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ ბრძანებას „ Make Default Schema".

შემდეგ სქემაზე მაუსის მარჯვენა ღილაკზე დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ ბრძანებას „Create New Table“.



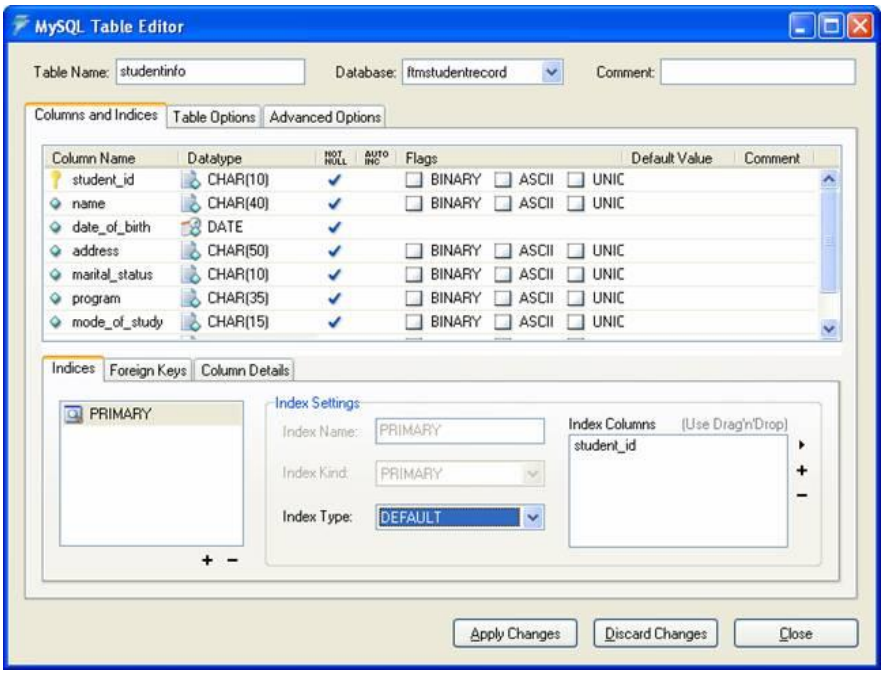
სურ. 3.15.

ფანჯარაში „MySQL Table Editor“ ვქმნით ცხრილის სტრუქტურას, რომელიც აისახება განყოფილებაში Schemata.



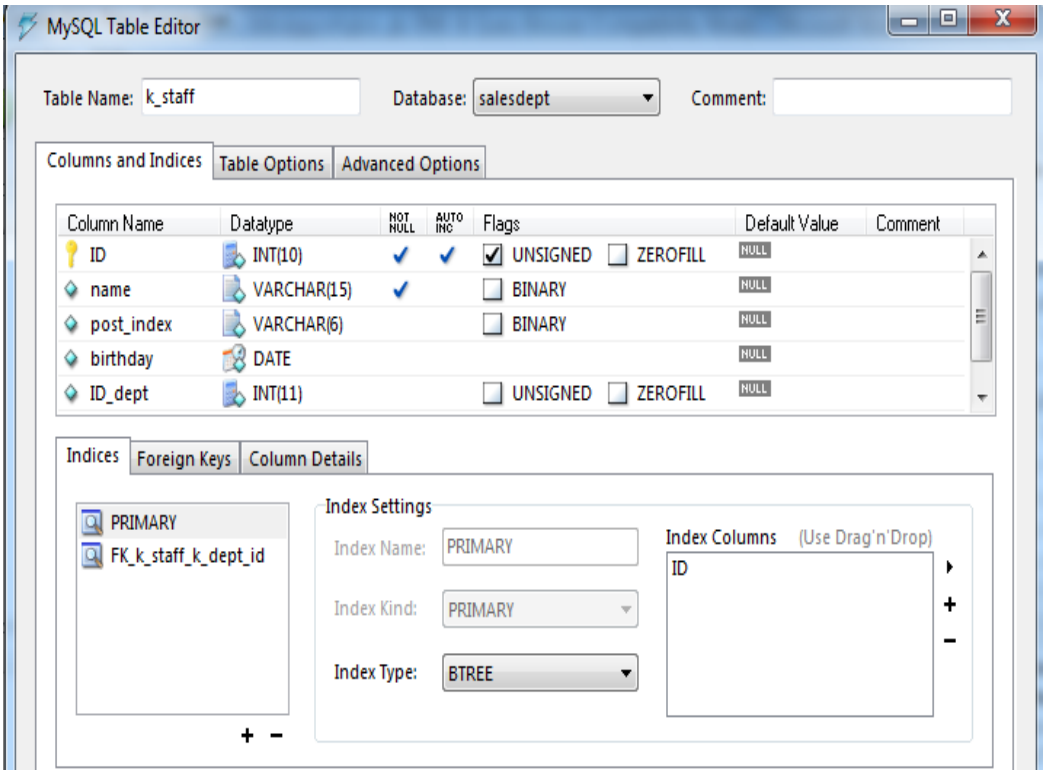
სურ. 3.16.

ცხრილს ვარქმევთ სახელს და ვქმნით ცხრილის ველებს.



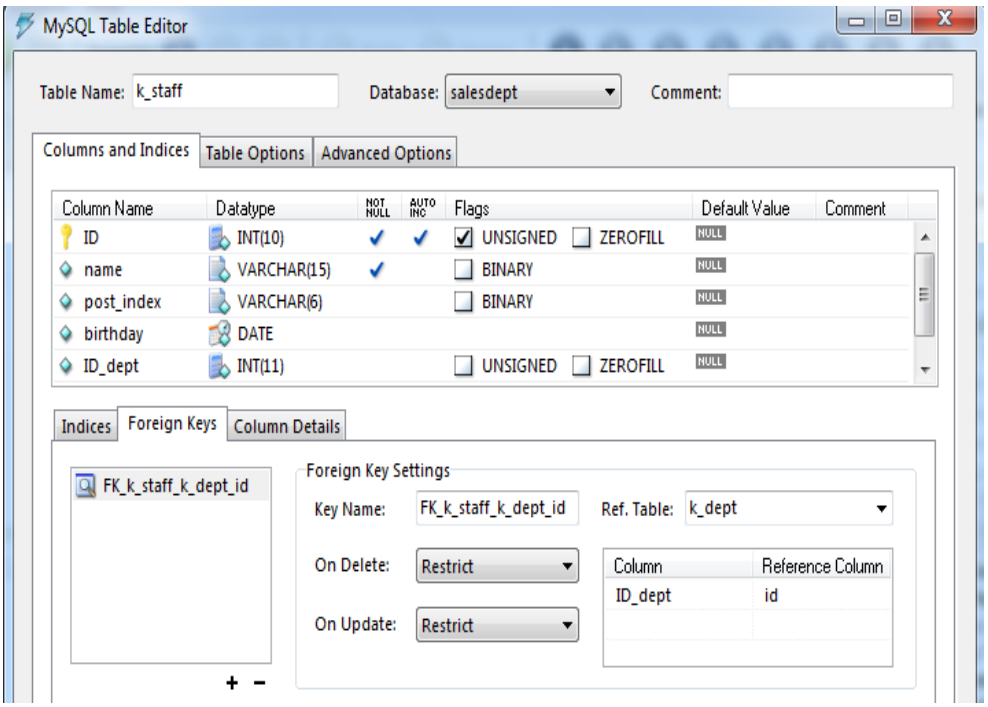
სურ. 3.17.

„MySQL Table Editor“ ფანჯარაში Indices ჩანართში დგება ცხრილის პირველადი და გარე გასაღებური ველები, ხოლო Index Columnes ველში - ინდექსირებული ველები.



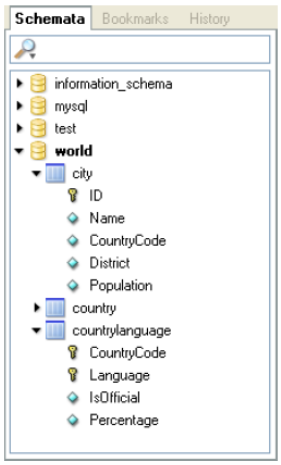
სურ. 3.18.

ფანჯარაში „MySQL Table Editor“ ჩანართში Foreign Keys ველში Ref.Table სვეტებში Column და Reference Column ვაყენებთ კავშირებს ცხრილებს შორის.



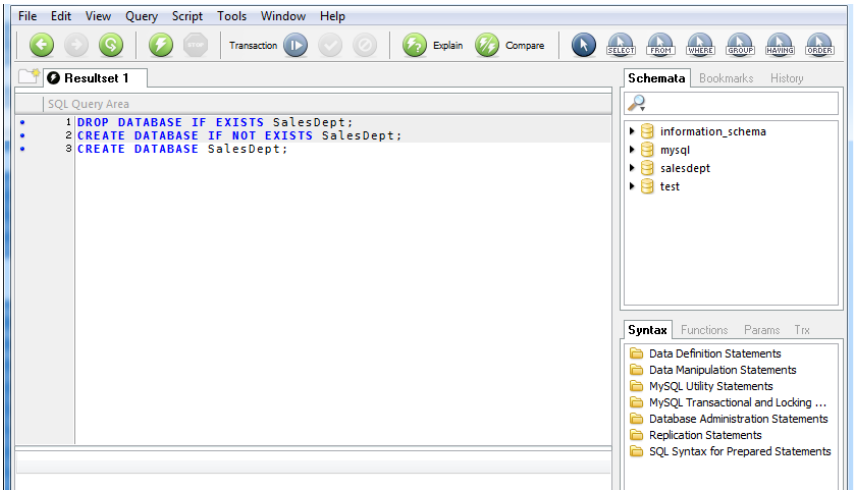
სურ. 3.19.

განყოფილებაში Schemata ვნახულობთ შექმნილ ცხრილებს.



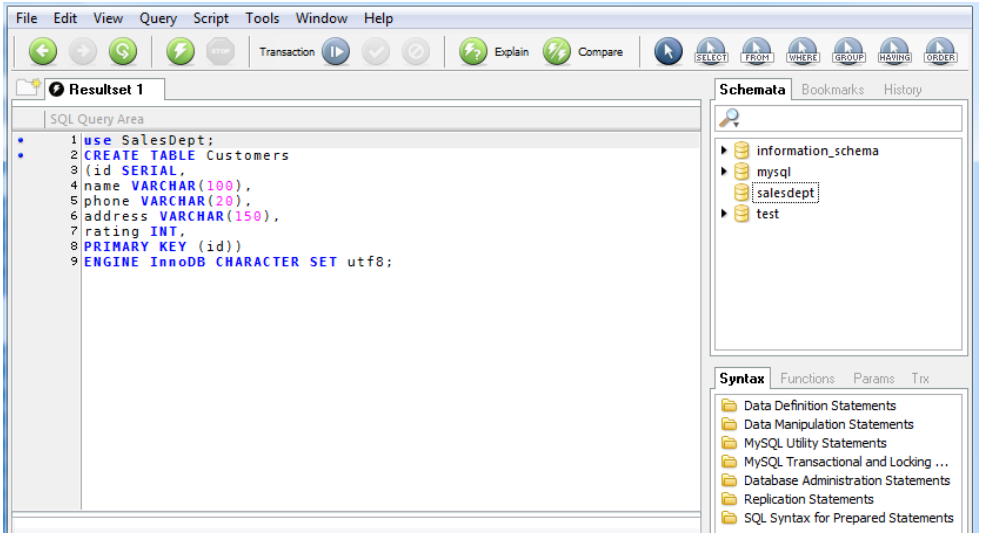
სურ. 3.20.

არსებობს სხვა მიდგომაც. მოთხოვნათა ველში შეგვყავს ქვემოთ მოყვანილი კოდი და ვაჭერთ მწვანე, „Execute“ ღილაკს. იქმნება სქემა „SalesDept“.



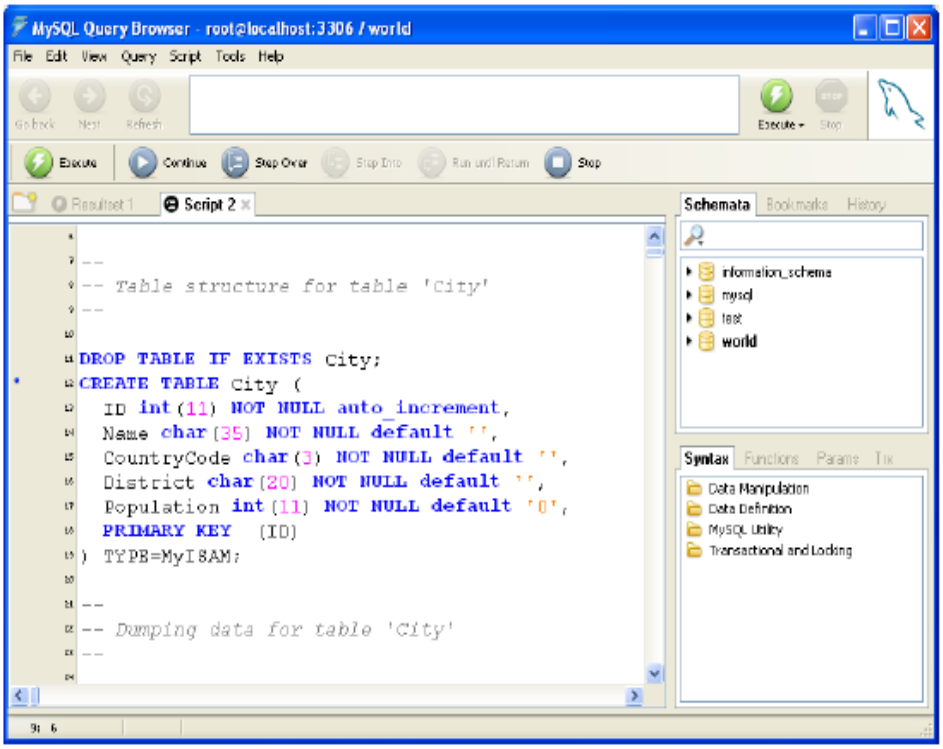
სურ. 3.21.

მას შემდეგ, რაც ჩვენი მონაცემთა ბაზა შეიქმნა, უნდა შევუდგეთ ცხრილების შექმნას, რისთვისაც მოთხოვნათა ფანჯარაში შეგვყავს ამ ცხრილების SQL კოდები და ვაჭერთ მწვანე ღილაკს „Execute“.



სურ. 3.22.

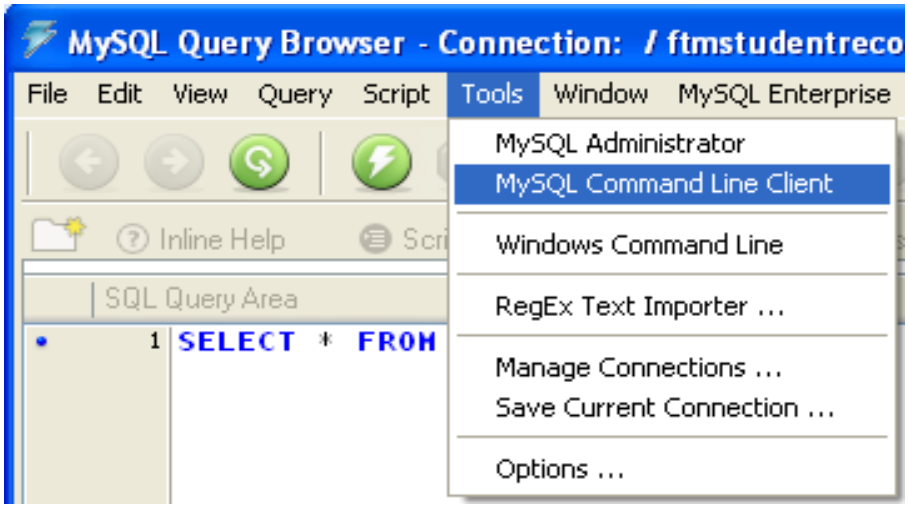
ასე თანამიმდევრულად ვქმნით ჩვენი მონაცემთა ბაზის ცხრილებს.



სურ. 3.23.

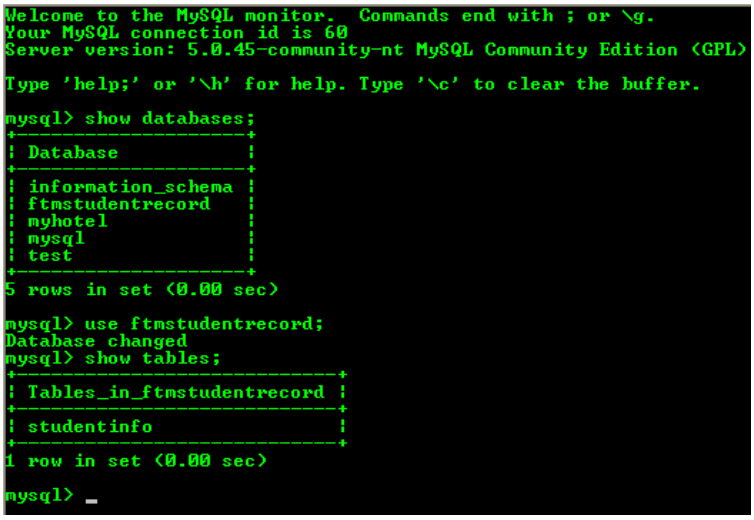
ყოველი ახალი ცხრილის ასახვისათვის განყოფილებაში Schemata მაუსის მარჯვენა ღილაკზე დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ ბრძანებას „Refresh“.

და ბოლოს, MySQL Administrator ფანჯრის მენიუს ბრძანებებით: „Tools > MySQL ბრძანება Line Client“ იხსნება კონსოლი,



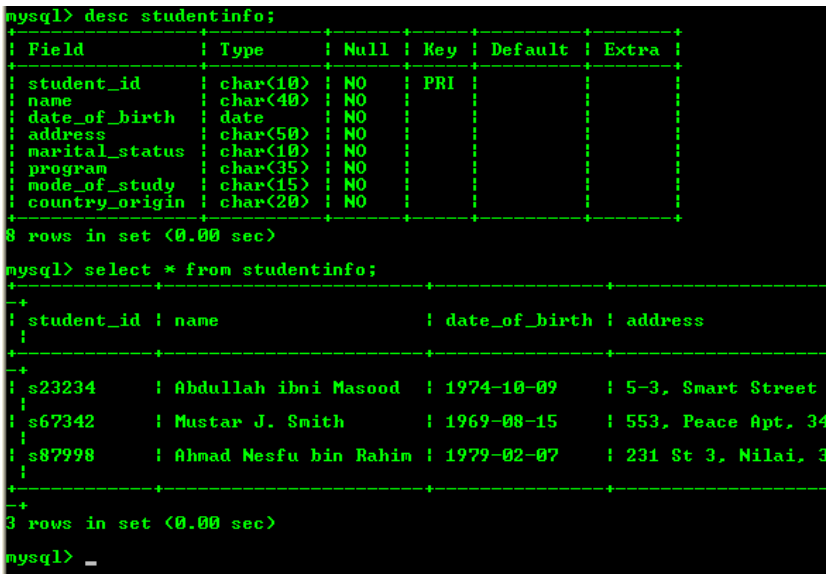
სურ. 3.24.

სადაც ვკრეფთ ბრძანებას show databases;



სურ. 3.25.

შემდეგ ბრძანებით describe შეგვიძლია გამოვიტანოთ ინფორმაცია ამა თუ იმ ცხრილზე.



სურ. 3.26.

ლაბორატორიული სამუშაო 4
მონაცემთა მანიპულირების ენა DML

სამუშაოს მიზანი:

1. ცხრილში მონაცემთა შეტანის ბრძანების შესწავლა;
2. მონაცემთა ცვლილების ბრძანების შესწავლა;
3. მონაცემთა წაშლის ბრძანების შესწავლა;
4. ცხრილიდან მონაცემთა ამოღების ბრძანების შესწავლა.
5. მონაცემების მანიპულირება MySQL Query Browser-ის მეშვებით.

1. მონაცემთა შეტანა ცხრილში

მონაცემთა ჩატვირთვა. ცხრილების შექმნის შემდეგ უნდა მოხდეს მონაცემებით მათი შევსება. არსებობს მონაცემთა ტექსტური ფაილიდან ჩატვირთვის ხერხი, რომელიც მიზანშეწონილია მონაცემთა დიდი მასივების შემთხვევაში. ეს მნიშვნელოვნად უფრო სწრაფია, ვიდრე INSERT ბრძანების მეშვეობით. მაგალითისათვის განვიხილოთ მონაცემთა ჩატვირთვა ცხრილში Customers. ამისათვის უნდა შესრულდეს შემდეგი მოქმედებები:

1. გავუშვათ სტანდარტული პროგრამა Windows Notepad
(Start → All programs →Standards → Notepad).

2. Notepad ფანჯარაში შეგვაქვს მონაცემები, რომლის დროსაც გამოყოფისათვის ვიყენებთ Tab კლავიშს, ხოლო ახალ სტრიქონზე გადასვლისათვის –Enter კლავიშს.

არარსებული ინფორმაციის ნაცვლად ფაილის შევსების დროს აუცილებელია „N“ სიმბოლოს შეტანა, რაც ნიშნავს, რომ მონაცემთა ბაზაში შეიტანება განუსაზღვრელი მნიშვნელობა (NULL).

3. შენახვისათვის ვაჭერთ კლავიშების კომბინაციას Ctrl+S. Windows სტანდარტულ ფანჯარაში ვირჩევთ საქაღალდეს, ფაილს ვანიჭებთ სახელს (მაგალითად, Customers. txt) და ვაჭერთ ღილაკს Save.

4. შექმნილი ტექსტური ფაილიდან მონაცემთა ჩატვირთვისათვის სრულდება ბრძანება

LOAD DATA LOCAL INFILE 'C:/data/Customers.txt'
INTO TABLE Customers
CHARACTER SET cp1251;

თუ საჭიროა მონაცემების ცხრილში ჩატვირთვა სხვა ფორმატის ტექსტური ფაილიდან, შეიძლება სხვა პარამეტრების გამოყენებაც გახდეს საჭირო:

LOAD DATA [LOCAL] INFILE 'გზა და ფაილის სახელი'
[REPLACE ან IGNORE]
INTO TABLE <ცხრილის სახელი>
CHARACTER SET <კოდირების სახელი>
[
FIELDS

[TERMINATED BY <სტრიქონში მნიშვნელობათა გამყოფი>]
[[OPTIONALLY]
ENCLOSED BY <სიმბოლო, რომელშიც ჩადებულია მნიშვნელობები>]
[ESCAPED BY <ეკრანირების სიმბოლო>]
][
LINES
[STARTING BY <სტრიქონის პრეფიქსი>]
[TERMINATED BY <სტრიქონების გამყოფი>]
]
[IGNORE <სტრიქონების რაოდენობა ფაილის სათავეში> LINES]
[(<სვეტების სია>)]
[SET <სვეტის სახელი> = <გამოსახულებ>,...];

ამ ბრძანებაში შეგვიძლია შემდეგი პარამეტრების გამოყენება:

• LOCAL – თუ მონაცემთა ფაილი იმყოფება კლიენტის კომპიუტერზე.

• *'გზა და ფაილის სახელი"* – შეგვაქვს სრული გზა ფაილისაკენ, მაგალითად C:/Data/ mytable.txt.

• REPLACE ან IGNORE – მივუთითოთ MySQL პროგრამას ერთ-ერთი ეს პარამეტრი, თუ როგორ დაამუშაოს ჩატვირთული სტრიქონი.

• CHARACTER SET <კოდირების სახელი> – მივუთითოთ მონაცემთა კოდირება.

მონაცემთა ჩატვირთვა UTF-8 კოდირებაში შეიძლება არაკორექტული აღმოჩნდეს სიმბოლოზე ბაიტების ცვლადი რაოდენობის გამო. ამიტომ რეკომენდებულია ჩატვირთვის წინ მონაცემთა ფაილი გარდაიქმნას ერთბიტიან კოდირებად.

ვინახავთ როგორც Windows სტანდარტულ ფანჯარაში. კოდირება აირჩევა ANSI სიიდან და ვაჭერთ Save ღილაკს . შემდეგ ბრძანებით LOAD DATA ჩავტვირთავთ ფაილს პარამეტრით CHARACTER SET cp1251.

მაგალითად:

LOAD DATA INFILE 'C:/DATA/t1.txt'
INTO TABLE t1 (@var1)
SET c1 = IF(@var1< = 1000,@var1,NULL), c2 = CURRENT_TIMESTAMP;

ცალკეული სტრიქონების ჩასმა

ერთი ან რამდენიმე სტრიქონის ცხრილში დამატებისათვის სრულდება ბრძანება:

INSERT [INTO] <ცხრილის სახელი>
[(<სვეტების სია>)]
VALUES
(<მნიშვნელობათა სია 1>),
(<მნიშვნელობათა სია 2>),
...
(<მნიშვნელობათა სია N>);

დრო და თარიღი შეიტანება შემდეგ ფორმატში: „YYYY-MM-DD“ და „HH:MM:SS“;
განვიხილოთ მონაცემების შეტანა ცხრილში Orders.

ლისტინგი 4.1.

```
INSERT INTO Orders
VALUES
(1012,'2007-12-12',5,8,'4500',533),
(1013,'2007-12-12',2,14,'22000',536),
(1014,'2008-01-21',5,12,'5750',533);
```

2. მონაცემთა ამოღება ცხრილიდან

მარტივი მოთხოვნები

მარტივი მოთხოვნა აბრუნებს პასუხს მხოლოდ ერთი ცხრილის მონაცემებიდან გამომდინარე:

*SELECT * FROM <ცხრილის სახელი>;*

მაგალითად,

*SELECT * FROM Customers;*

შედეგად მივიღებთ სრულ ინფორმაციას კლიენტების შესახებ.

ვარსკვლავის ნაცვლად შეიძლება მივუთითოთ კონკრეტული სვეტები.

მაგალითად,

SELECT name,phone,rating FROM Customers;

მოთხოვნის მიხედვით შესაძლებელია მივიღოთ გამოთვლითი მნიშვნელობები. მაგალითად,

SELECT name,phone,rating/1000 FROM CUSTOMERS;

განმეორებადი მნიშვნელობების გამორიცხვისათვის გამოიყენება გასაღებური სიტყვა DISTINCT. მაგალითად,

SELECT DISTINCT rating FROM Customers;

მოთხოვნის შედეგად გამოტანილი მონაცემების მოწესრიგებისათვის გამოიყენება ბრძანება

ORDER BY <სვეტის სახელი> [ASC ან DESC]

მაგალითად,

```
SELECT name,phone,rating FROM Customers
ORDER BY rating DESC, name;
```

შერჩევის პირობა

თუ გვინდა ცხრილიდან ამოვირჩიოთ სტრიქონები გარკვეული პირობით, მაშინ გამოიყენება შერჩევის კრიტერიუმი

WHERE <შერჩევის პირობა>

მაგალითად,

SELECT name,phone,rating FROM Customers WHERE rating = 1000;

შერჩევის კრიტერიუმის შედგენის დროს შეიძლება გამოყენებულ იქნას ლოგიკური ოპერატორები AND ან OR . მაგალითად,

```
SELECT name,phone,rating FROM Customers
WHERE name LIKE 'OOO%' OR rating>1000
ORDER BY rating DESC;
```

ცხრილების გაერთიანება

რამდენიმე ცხრილიდან გარკვეული სვეტების ერთობლივად გამოტანისათვის გვექნება:

SELECT <სვეტების სია> FROM <ცხრილების სია>

WHERE <შერჩევის პირობა>;

მაგალითად,

```
SELECT name,address,product_id,qty
FROM Customers, Orders
WHERE Customers.id = customer_id AND date = '2007-12-12';
```

ორი და მეტი ცხრილიდან მოთხოვნების გაერთიანებისათვის ასევე შეიძლება გამოყენებულ იქნას ბრძანება WHERE. მაგალითად,

```
SELECT L.name,R.name FROM Customers L, Customers R
WHERE L.rating = R.rating;
```

მოთხოვნათა გაერთიანების დროს შეიძლება გამოყენებულ იქნას ლოგიკური ოპერატორები AND ან OR . მაგალითად,

```
SELECT L.name,R.name FROM Customers L, Customers R
WHERE L.rating = R.rating AND L.name<R.name;
```

ჩასმული მოთხოვნები

ხშირად მოთხოვნის შედეგი, რომელიც წარმოადგენს მონაცემთა მასივს ცხრილის სახით, შეიძლება გამოყენებულ იქნას სხვა მოთხოვნაში, რასაც ჩასმულ მოთხოვნებს უწოდებენ. მაგალითად:

```
SELECT name FROM Customers
WHERE id IN
(SELECT DISTINCT customer_id FROM Orders
WHERE product_id = 5);
```

ასეთივე შედეგი მიიღება გაერთიანების გამოყენების შემთხვევაში:

```
SELECT DISTINCT name FROM Customers, Orders
WHERE Customers.id = customer_id AND product_id = 5;
```

მოთხოვნათა შედეგების გაერთიანება

რამდენიმე მოთხოვნის ერთ SQL- ბრძანებაში გაერთიანებისათვის გამოიყენება გასაღებური სიტყვა UNION. ამისათვის მონაცემები სვეტებში თავსებადი უნდა იყოს.

მაგალითად,

```
SELECT * FROM Orders
WHERE amount = (SELECT MAX(amount) FROM Orders)
UNION
```

*SELECT * FROM Orders*
WHERE amount = (SELECT MIN(amount) FROM Orders)
ORDER BY 1;

3. მონაცემთა ცვლილება

განვიხილოთ ცხრილებში ერთ ან რამდენიმე სტრიქონში მონაცემთა ცვლილების ბრძანება UPDATE, რომლის სინტაქსისი შემდეგია:

UPDATE <ცხრილის სახელი>
SET <1 სვეტის სახელი> = <მნიშვნელობა 1>,
...,
<N სვეტის სახელი> = <მნიშვნელობა N>
[WHERE <შერჩევის პირობა>]
[ORDER BY <სვეტის სახელი> [ASC ან DESC]]
[LIMIT <სტრიქონების რაოდენობა>];

მაგალითად,

UPDATE Customers SET phone = '444-25-27' WHERE id = 536;

ცვლილება გულისხმობს აგრეთვე მათემატიკური ოპერაციების გამოყენებასაც:

*UPDATE Customers SET rating = rating*2;*

შემდეგი ბრძანება არის – REPLACE, რომელიც ახორციელებს ცხრილში სტრიქონის ან დამატებას ან ჩანაცვლებას. მას გააჩნია იგივე პარამეტრები, რაც INSERT ბრძანებას:

REPLACE [INTO] <ცხრილის სახელი>
[(<სვეტების სია>)]
VALUES
(<მნიშვნელობათა სია 1>),
(<მნიშვნელობათა სია 2>),
...
(<მნიშვნელობათა სია N>);

ბრძანებაში REPLACE, განსხვავებით ბრძანებიდან UPDATE, არ შეიძლება ახალი მნიშვნელობების მინიჭება გამოთვლითი ოპერაციების მეშვეობით.

4. მონაცემთა წაშლა

და ბოლოს, სტრიქონების წაშლის ბრძანება:

DELETE FROM <ცხრილის სახელი>
[WHERE <შერჩევის პირობა>]
[ORDER BY <სვეტის სახელი> [ASC ან DESC]]
[LIMIT <სტრიქონების რაოდენობა>];

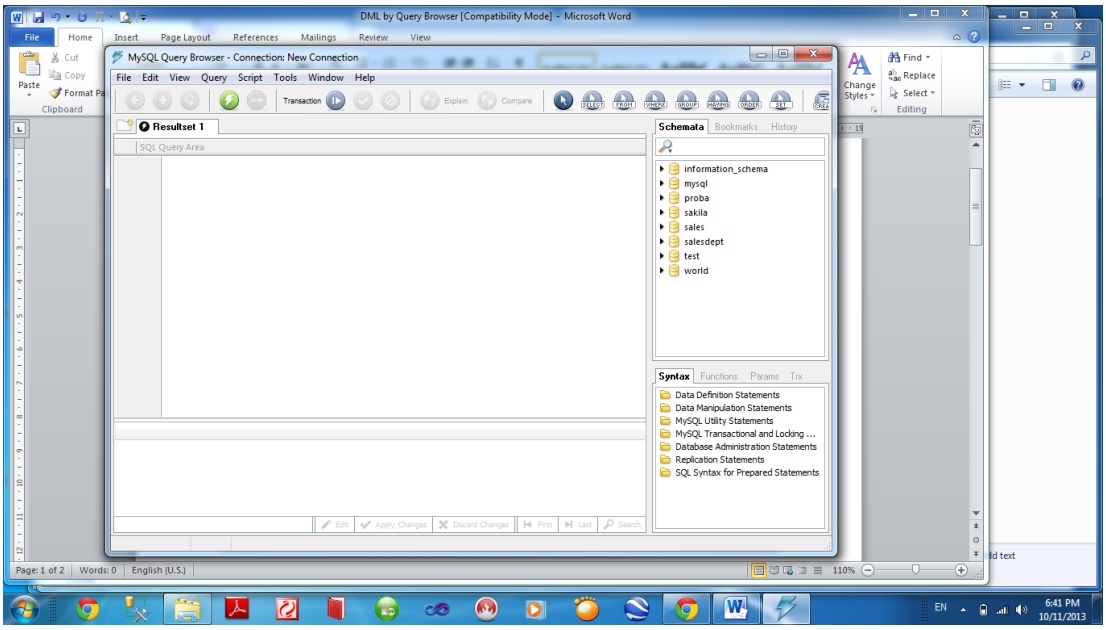
მაგალითად,

DELETE FROM Customers WHERE id = 534;

5. მონაცემთა მანიპულირება MySQL Query Browser -ის მეშვეობით

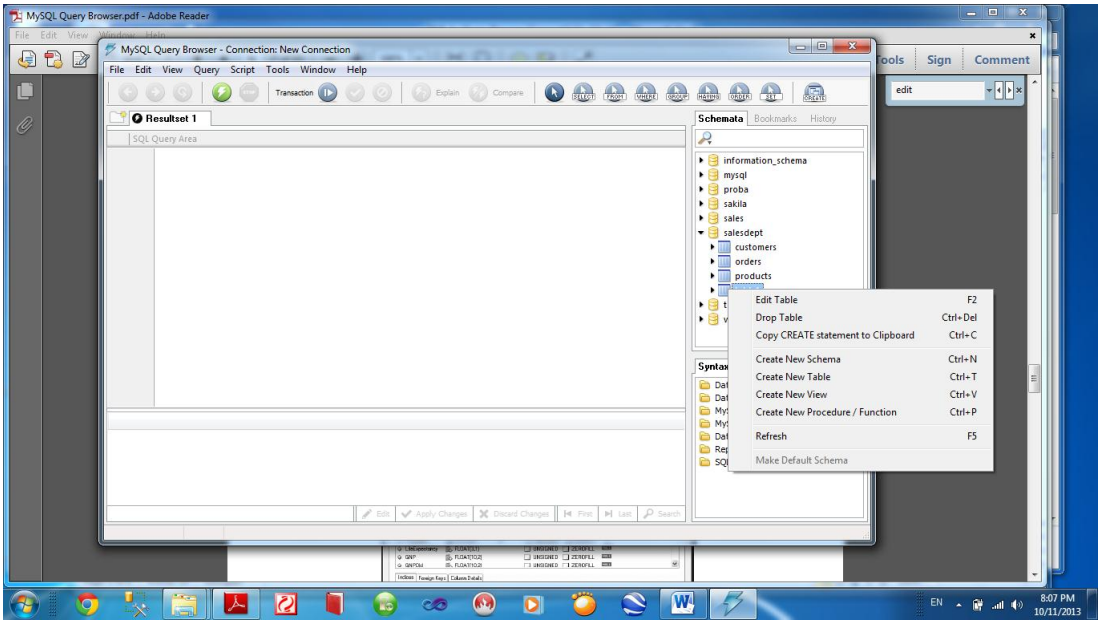
ცხრილის სტრუქტურის რედაქტირება

გავხსნათ MySQL Query Browser -ის მთავარი ფანჯარა.



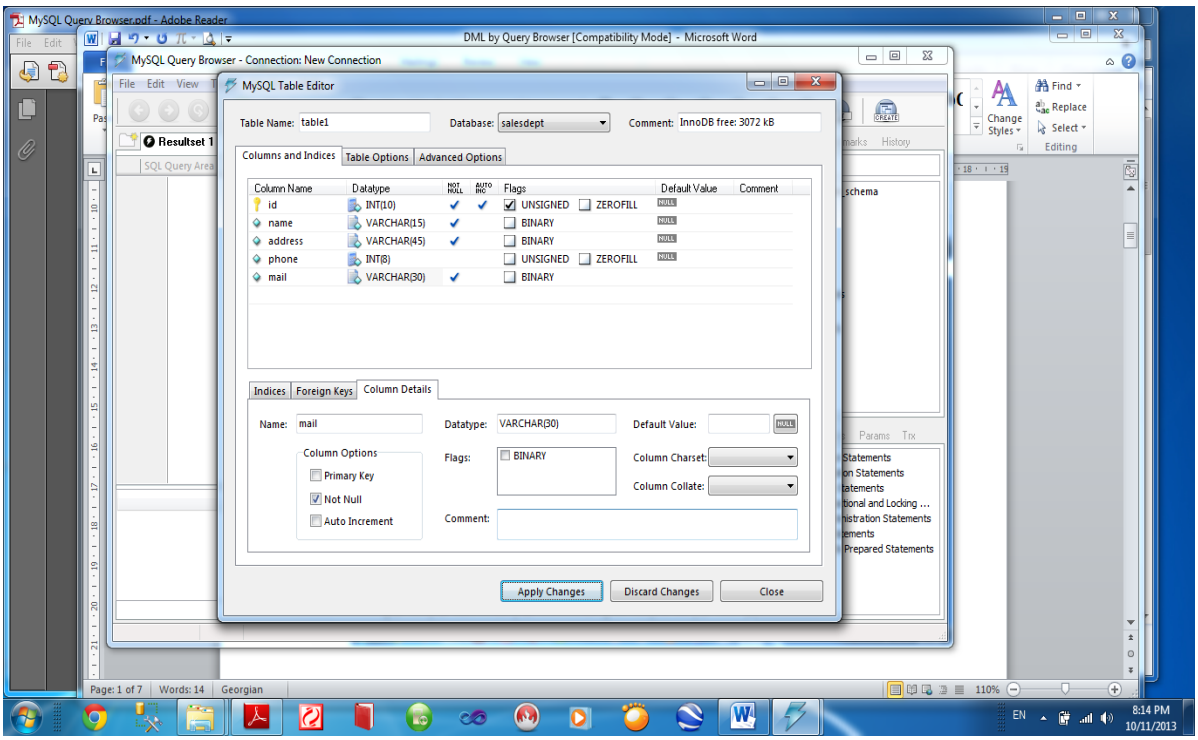
სურ. 4.1

Schemata არეში მაუსის მარჯვენა ღილაკზე დაწკაპუნებით იხსნება კონტექსტური მენიუ, სადაც ვირჩევთ Edit Table.



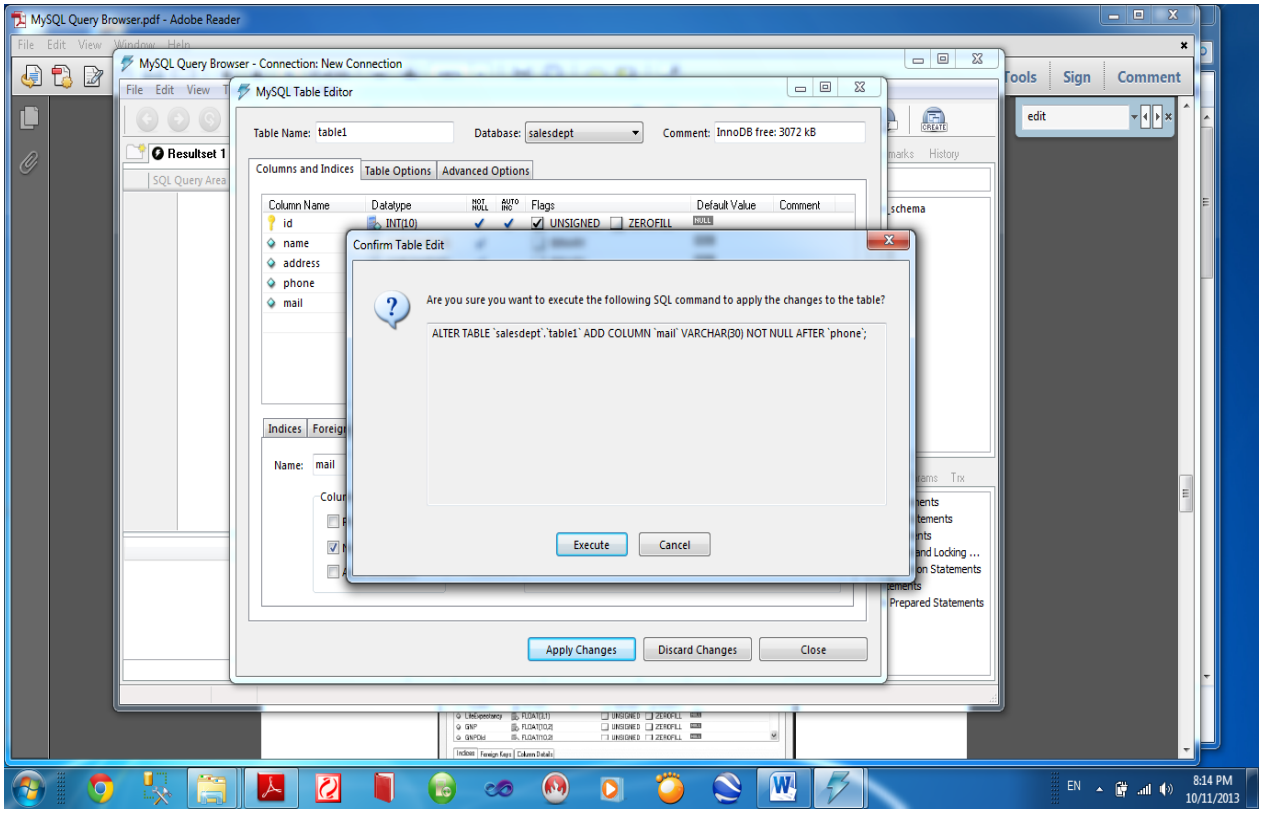
სურ. 4.2

იხსნება Edit Table Editor ფანჯარა ცხრილის სტრუქტურით, სადაც შეგვაქვს სანახველი ცვლილებები და ვაჭერთ ღილაკს Apply Changes.



სურ. 4.3.

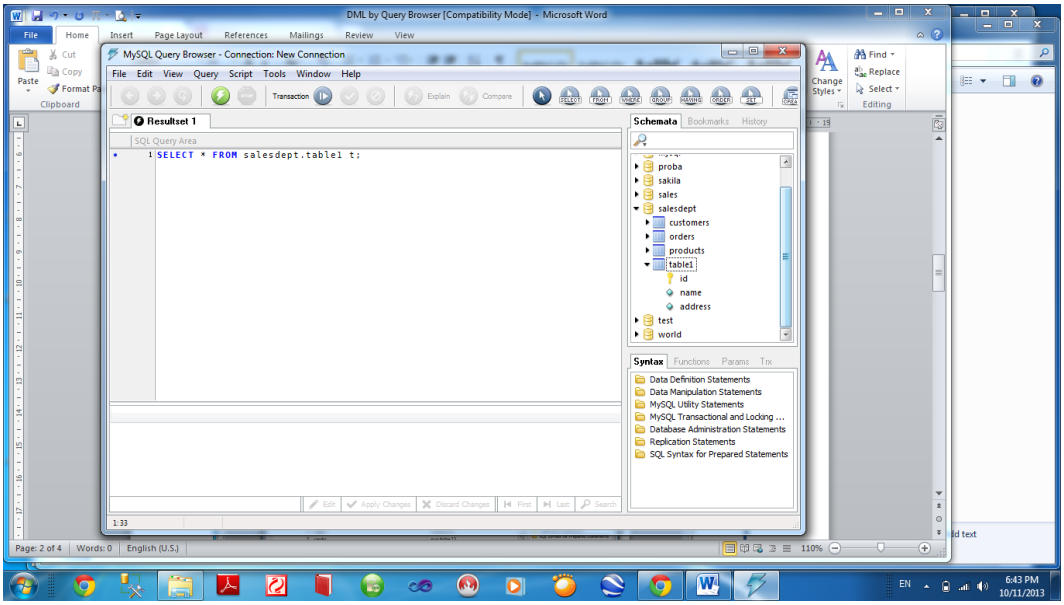
იხსნება ფანჯარა Confirm Table Edit, სადაც ვაჭერთ ღილაკს Execute.



სურ. 4.4.

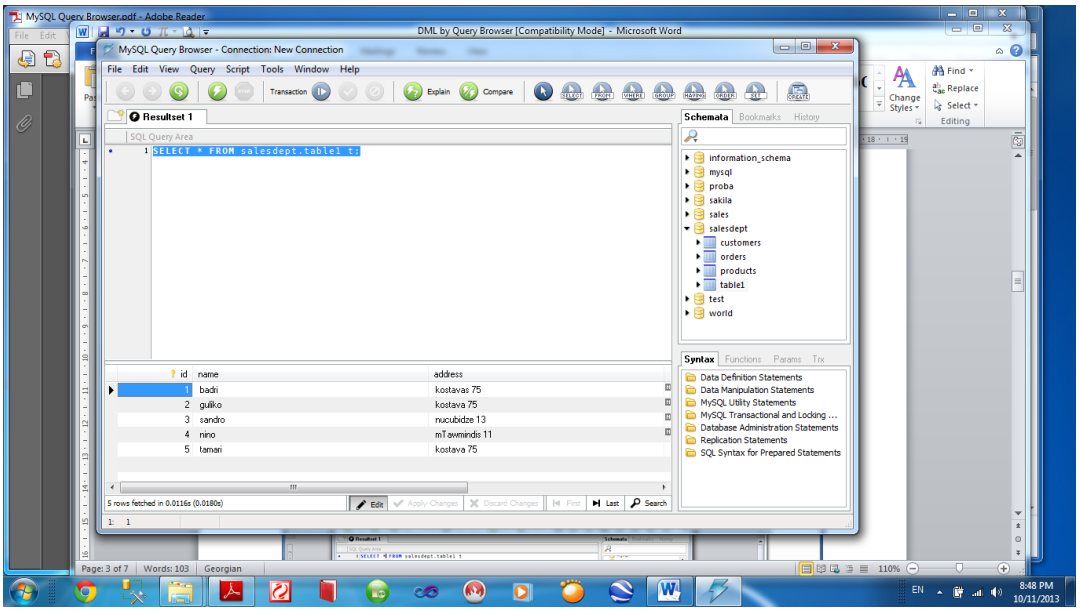
მონაცემების შეტანა

Schemata არეში ცხრილზე ორჯერ ვაწკაპუნებთ. Resultat არეში შესრულდება SELECT ბრძანება ჩვენი ცხრილის ყველა ჩანაწერის ჩვენების მოთხოვნით.



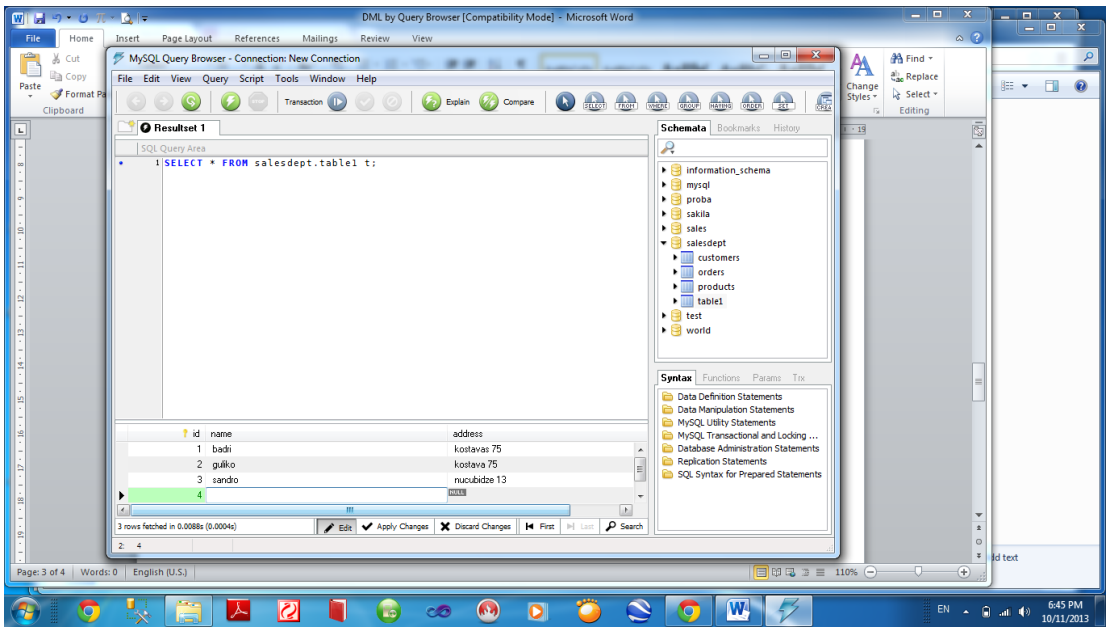
სურ. 4.5.

შემდეგ განმეორებით ორჯერ ვაწკაპუნებთ ცხრილზე. ფანჯრის ქვედა ნაწილში ჩნდება ცხრილის სტრუქტურა. ვაჭერთ ღილაკს Edit, რომლის შედეგად შესაძლებელი ხდება მონაცემების შეტანა. ბოლო ცარიელ ჩანაწერზე გადასვლის შემდეგ ჩნდება ღილაკი Apply Changes, რომლითაც ვაფიქსირებთ ცვლილებას.



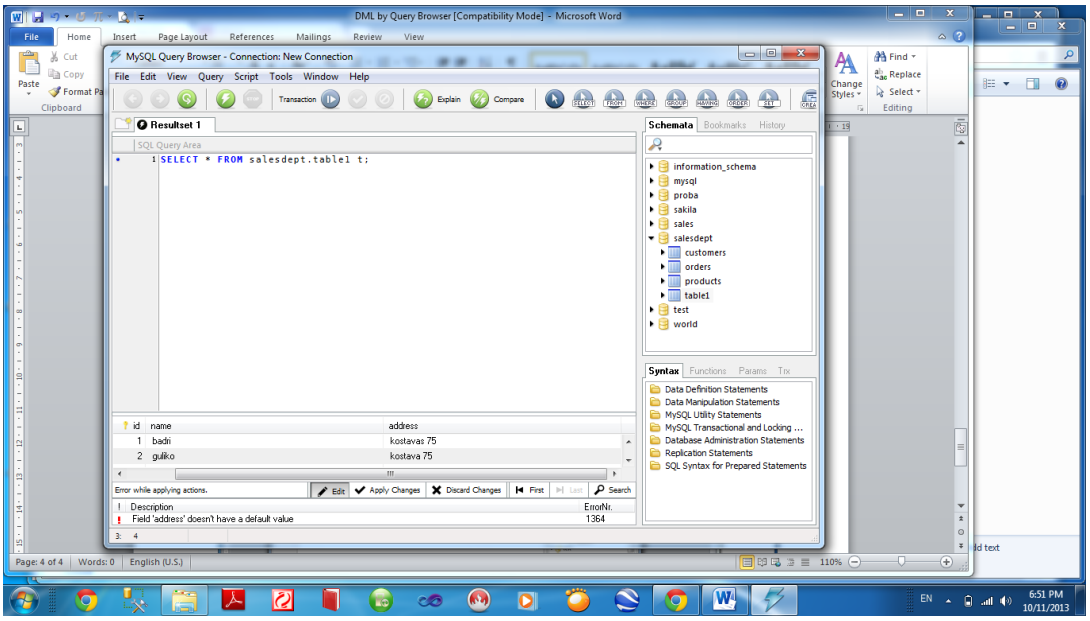
სურ. 4.6.

ცხრილში მონაცემთა ცვლილებები შესაძლებელია Edit ღილაკზე დაჭერით. ცვლილებების (ან ახალი ჩანაწერის) შეტანის შემდე ვაჭერთ ღილაკს Apply Changes.



სურ. 4.7.

შემდეგ ვაჭერთ Execute ღილაკს.



სურ. 4.8.

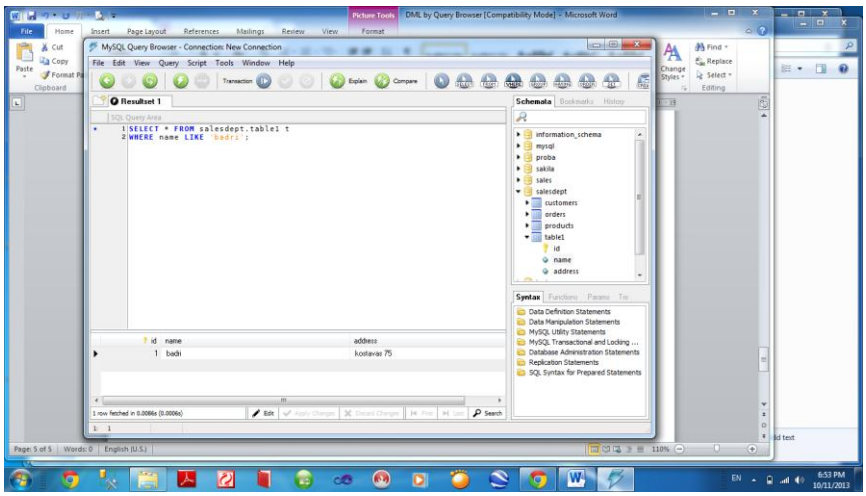
მოთხოვნათა შესრულება

არსებობს მოთხოვნათა შესულების ორი გზა: ხელით და ფუნქციური ღილაკების გამოყენებით. ფუნქციური ღილაკების შემთხვევაში ჯერ ვაწკაპუნებთ ფუნქციურ ღილაკზე, ხოლო შემდეგ მოცემული ცხრილის შესაბამის სვეტზე, რომელიც ჩართული იქნება მოთხოვნაში.

ვთქვათ გვაქვს მოთხოვნა:

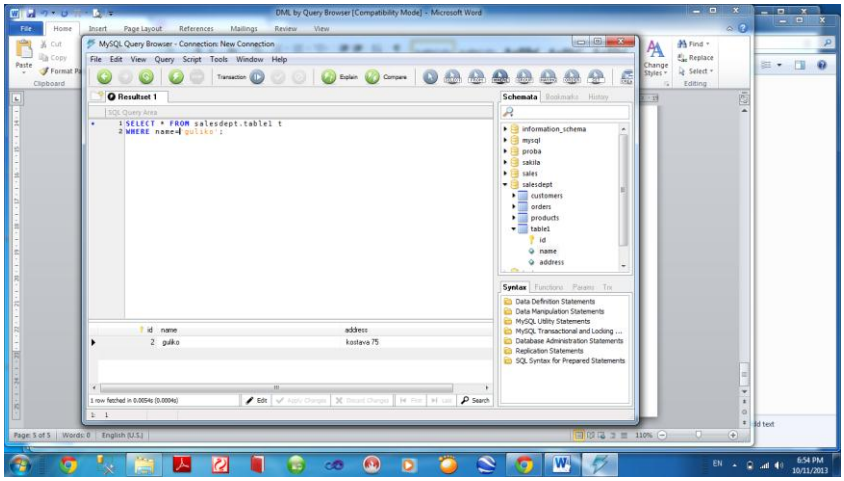
SELECT * FROM salesdept.table1 t;

WHERE name LIKE ‘badri’;



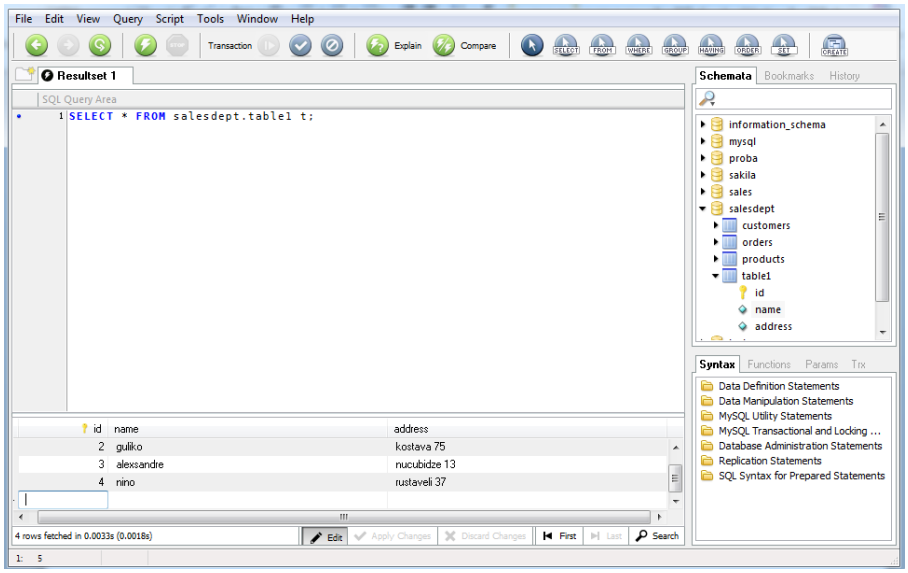
სურ. 4.9.

SELECT * FROM salesdept.table1 t;
WHERE name=’sandro’;



სურ. 4.10.

მოთხოვნის შესასრულებლად ვაჭერთ ღილაკს Execute.



სურ. 4.11.

შედეგი მიიღება ფანჯრის ქვედა ნაწილში.

ლაბორატორიული სამუშაო 5
მრავალცხრილიანი მოთხოვნები. JOIN ბრძანებები

სამუშაოს მიზანი:

1. JOIN / INNER JOIN ბრძანების შესწავლა;
2. LEFT OUTER JOIN ბრძანების შესწავლა;
3. RIGHT OUTER JOIN ბრძანების შესწავლა;
4. FULL JOIN ბრძანების შესწავლა;
5. CROSS JOIN ბრძანების შესწავლა.
6. JOIN ბრძანების შესრულება MySQL Query Browser -ის გამოყენებით.
7. JOIN ბრძანების შესრულება Workbench -ის გამოყენებით.

გასაღებური სიტყვა JOIN გამოიყენება გარკვეული სვეტებით დაკავშირებული ორი ან მეტი ცხრილიდან მონაცემების ამოსარჩევად. ცხრილებს შორის კავშირის მეშვეობით მიიღება ე.წ. ბმული ცხრილი.

- INNER კავშირის ტიპი ავტომატურად გამოიყენება. შედეგში ჩართული იქნება მარცხენა ცხრილის მხოლოდ ის სტრიქონები, რომლებისთვისაც არსებობენ სტრიქონები ბმულ (მარჯვენა) ცხრილში. ჩართული იქნება, აგრეთვე, მარჯვენა ცხრილის მხოლოდ ის სტრიქონები, რომლებისთვისაც არსებობენ შესაბამისი სტრიქონები მარცხენა ცხრილში.

- LEFT [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარცხენა ცხრილის ყველა სტრიქონი, მიუხედავად იმისა, არსებობს თუ არა მათთვის შესაბამისი სტრიქონი მარჯვენა ცხრილში. მარჯვენა ცხრილის შესაბამის ველებს NULL მნიშვნელობა ექნებათ.

- RIGHT [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარჯვენა ცხრილის ყველა სტრიქონი მიუხედავად იმისა, აქვთ თუ არა მათ შესაბამისი სტრიქონები მარცხენა ცხრილში. მარცხენა ცხრილის შესაბამის ველებს NULL მნიშვნელობა ექნებათ.

- FULL [OUTER] არგუმენტის გამოყენების შედეგად შედეგში ჩართული იქნება მარცხენა და მარჯვენა ცხრილების ყველა სტრიქონი.

- CROSS JOIN საკვანძო სიტყვის გამოყენებისას სრულდება მარცხენა ცხრილის თითოეული სტრიქონის დაკავშირება მარჯვენა ცხრილის თითოეულ სტრიქონთან. თუ ცხრილებს შორის კავშირი არ არის მითითებული, მაშინ ავტომატურად სრულდება მარცხენა ცხრილის თითოეული სტრიქონის დაკავშირება მარჯვენა ცხრილის თითოეულ სტრიქონთან.

შესაბამისად, განირჩევა JOIN ბრძანების შემდეგი ვარიაციები:

- JOIN / INNER JOIN.
- LEFT OUTER JOIN.
- RIGHT OUTER JOIN.

- FULL JOIN.
- CROSS JOIN.

თვალსაჩინოებისათვის განვიხილოთ მაგალითი. გვაქვს ორი ცხრილი: „employee“ და „department“. თუ დავაკვირდებით ცხრილში employee გვაქვს ჩანაწერი Jasper, რომელიც მუშაობს 36-ე დეპარტამენტში, მაგრამ ეს დეპარტამენტი არ ფიგურირებს ცხრილში department, ხოლო ამ უკანასკნელში გვაქვს ჩანაწერი marketing, სადაც ჩამოთვლილი თანამშრომლებიდან არცერთი არ მუშაობს.

Table "employee"

LastName	DepartmentID
Smith	34
Jones	33
Robinson	34
Jasper	36
Steinberg	33
Rafferty	31

Table "department"

DepartmentName	DepartmentID
Sales	31
Engineering	33
Clerical	34
Marketing	35

1. INNER JOIN ბრძანება

INNER JOIN ანუ იგივე JOIN ბრძანება ორივე ცხრილიდან საერთო ჩანაწერებს. INNER JOIN ბრძანების სინტაქსისი შემდეგია:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

ჩვენი მაგალითის შემთხვევაში გვექნება:

```
SELECT *
  FROM employee
       INNER JOIN department
       ON employee.DepartmentID =
             department.DepartmentID
```

```
+-----------+--------------+----------------+--------------+
| LastName  | DepartmentID | DepartmentName | DepartmentID |
+-----------+--------------+----------------+--------------+
| Smith     |           34 | Clerical       |           34 |
| Jones     |           33 | Engineering    |           33 |
| Robinson  |           34 | Clerical       |           34 |
| Steinberg |           33 | Engineering    |           33 |
| Rafferty  |           31 | Sales          |           31 |
+-----------+--------------+----------------+--------------+
```

2. LEFT OUTER JOIN ბრძანება

LEFT OUTER JOIN ანუ იგივე LEFT JOIN ბრძანება აბრუნებს მარცხენა ცხრილის (table_name1) ყველა ჩანაწერს, ხოლო მარჯვენა ცხრილიდან (table_name2) მხოლოდ იმ ჩანაწერებს, რომლებიც აკმაყოფილებენ შესაბამისობის პირობას. LEFT JOIN სინტაქსისი შემდეგია:

```
SELECT column_name(s)
FROM table_name1
```

LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
ჩვენი მაგალითის შემთხვევაში გვექნება:

```
SELECT *
  FROM employee
       LEFT OUTER department
       ON employee.DepartmentID =
             department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31

3.RIGHT OUTER JOIN ანუ RIGHT JOIN ბრძანება

RIGHT OUTER JOIN ბრძანება აბრუნებს მარჯვენა ცხრილის (table_name2) ყველა ჩანაწერს, ხოლო მარცხენა ცხრილის (table_name1) მხოლოდ იმ ჩანაწერებს, რომლებიც აკმაყოფილებენ შესაბამისობის პირობას. ჩვენი მაგალითის შემთხვევაში გვექნება:

```
SELECT *
  FROM employee
       RIGHT OUTER department
       ON employee.DepartmentID =
             department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

4. FULL OUTER JOIN ანუ იგივე FULL JOIN ბრძანება

FULL OUTER JOIN ბრძანება ფაქტობრივად წარმოადგენს LEFT OUTER JOIN და RIGHT OUTER JOIN ბრძანებების გაერთიანებას. მისი სინტაქსისი შემდეგია:

```
SELECT *
FROM A LEFT JOIN B ON A.id = B.id
UNION
SELECT *
FROM A RIGHT JOIN B ON A.id = B.id
WHERE A.id IS NULL
```

ჩვენი მაგალითის შემთხვევაში გვექნება:

```
SELECT *
 FROM employee
      FULL OUTER JOIN department
      ON employee.DepartmentID =
      department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

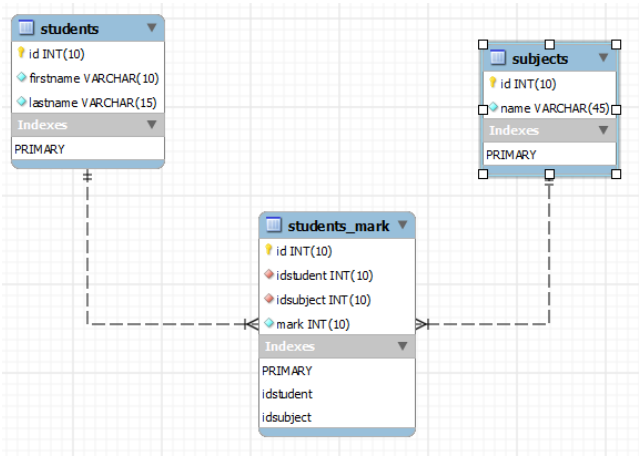
5. CROSS JOIN ბრძანება

ფაქტობრივად CROSS JOIN ბრძანება წარმოადგენს ამ ორი ცხრილის ჩანაწერების დეკარტულ ნამრავლს. ჩვენი მაგალითის შემთხვევაში გვექნება:

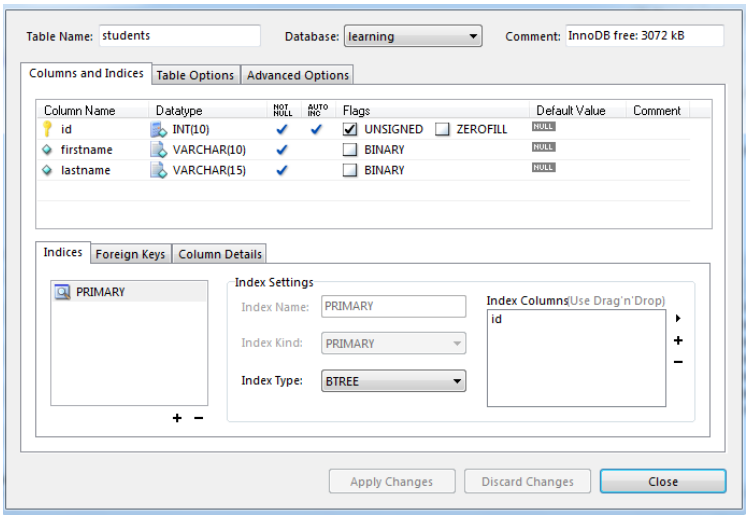
```
SELECT *
 FROM employee
     CROSS JOIN department;
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Sales	31
Smith	34	Engineering	33
Smith	34	Clerical	34
Smith	34	Marketing	35
Jones	33	Sales	31
Jones	33	Engineering	33
Jones	33	Clerical	34
Jones	33	Marketing	35
Robinson	34	Sales	31
Robinson	34	Engineering	33
Robinson	34	Clerical	34
Robinson	34	Marketing	35
Jasper	36	Sales	31
Jasper	36	Engineering	33
Jasper	36	Clerical	34
Jasper	36	Marketing	35
Steinberg	33	Sales	31
Steinberg	33	Engineering	33
Steinberg	33	Clerical	34
Steinberg	33	Marketing	35
Rafferty	31	Sales	31
Rafferty	31	Engineering	33
Rafferty	31	Clerical	34
Rafferty	31	Marketing	35

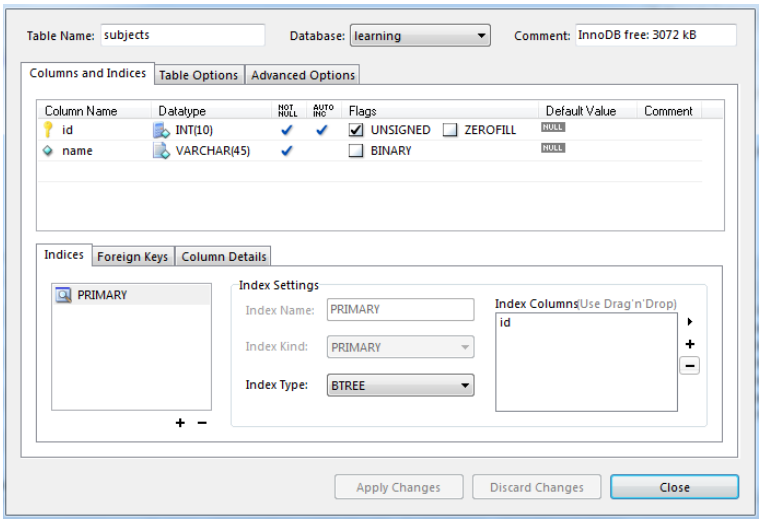
6. JOIN ბრძანების შესრულება MySQL Query Browser-ის გამოყენებით



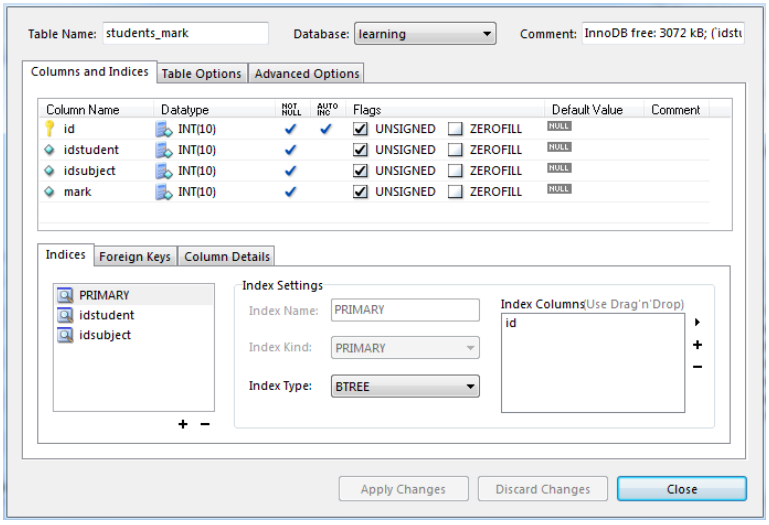
სურ. 5.1



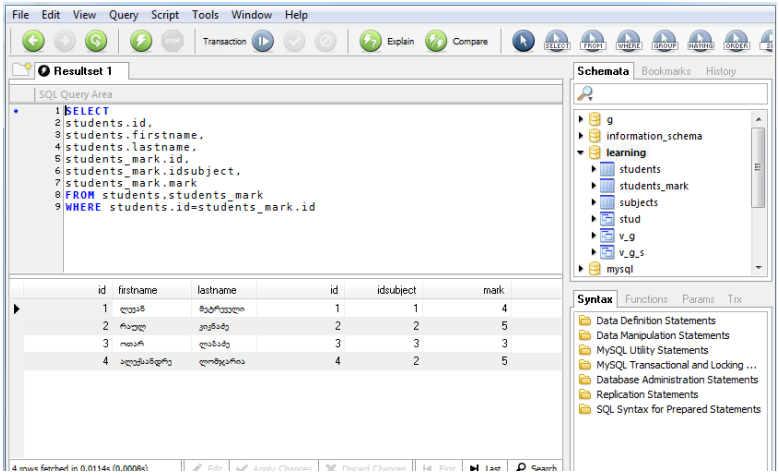
სურ. 5.2



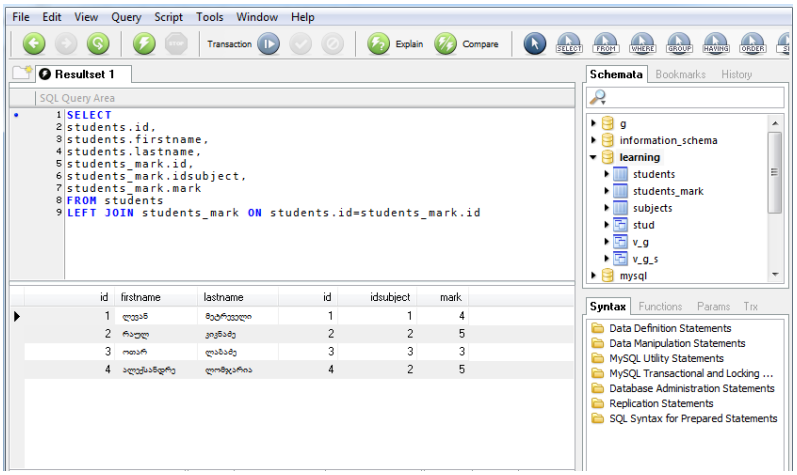
სურ. 5.3



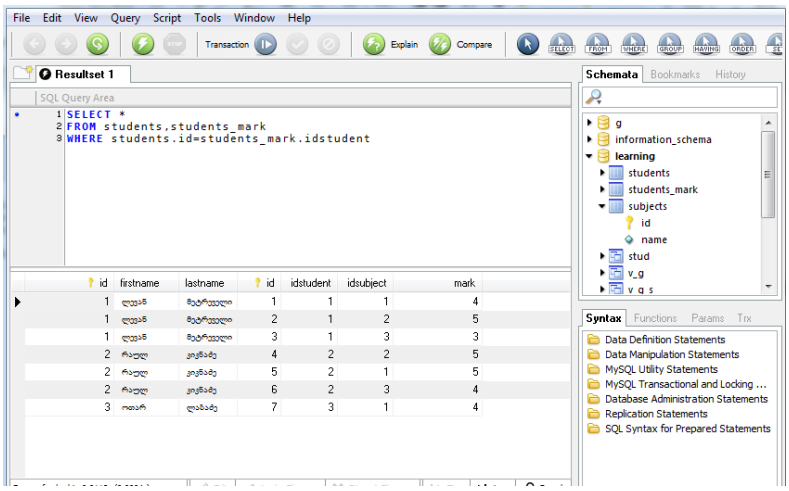
სურ. 5.4.



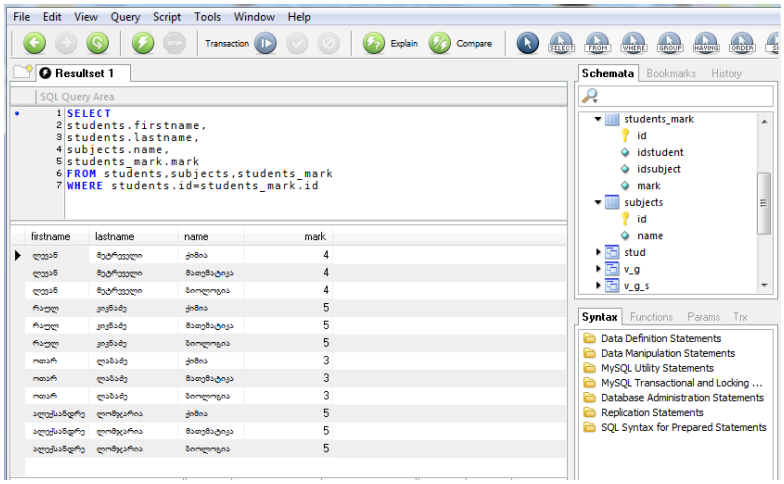
სურ. 5.5.



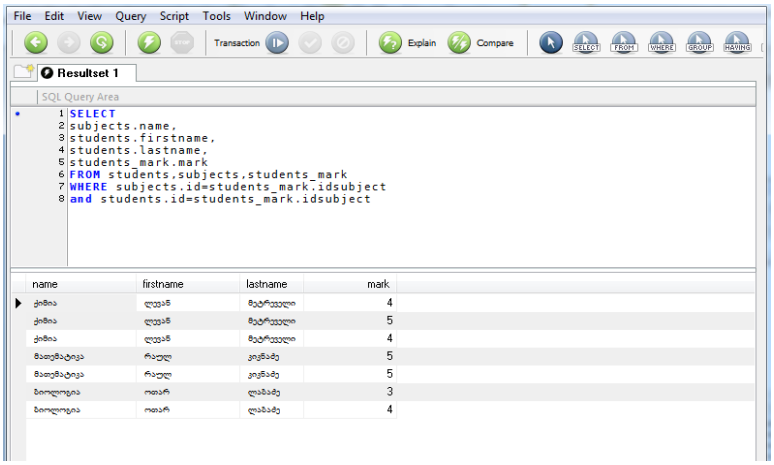
სურ. 5.6.



სურ. 5.7.



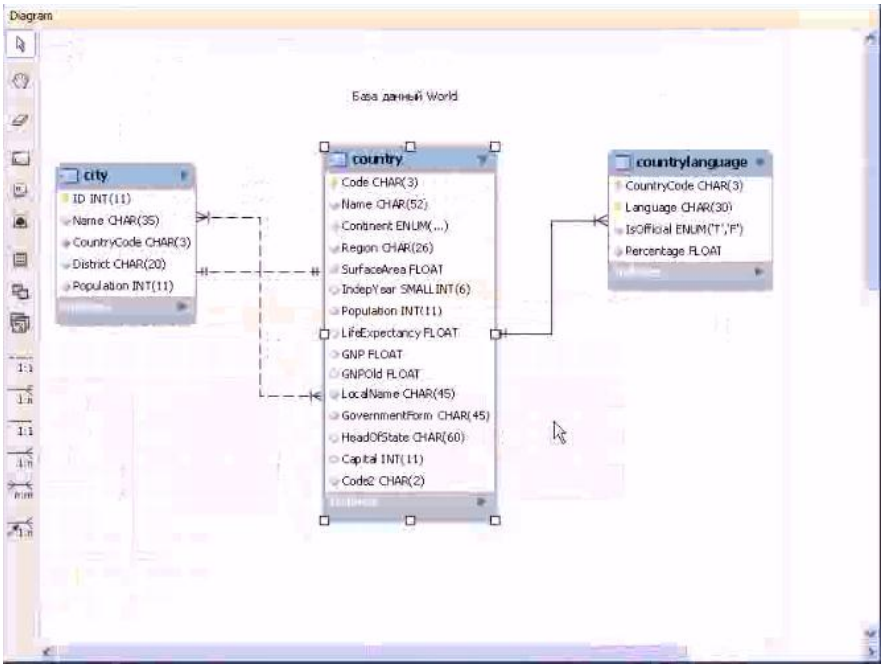
სურ. 5.8.



სურ. 5.9.

7. JOIN ბრძანების შესრულება Workbench -ის გამოყენებით

ვთქვათ გვაქვს მონაცემთა ბაზა World, რომელიც შედგება შემდეგი სამი ცხრილისაგან: city, country და countrylanguage.

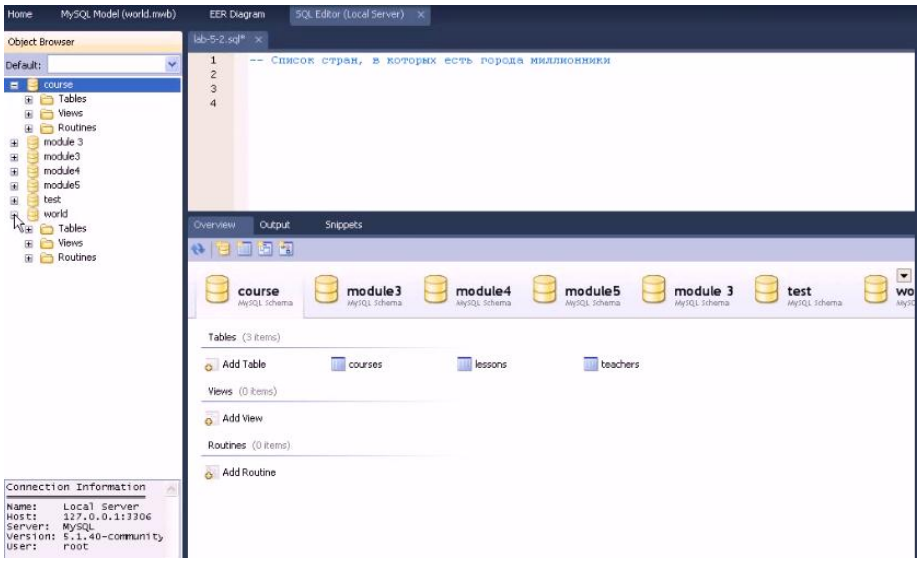


სურ. 5.10.

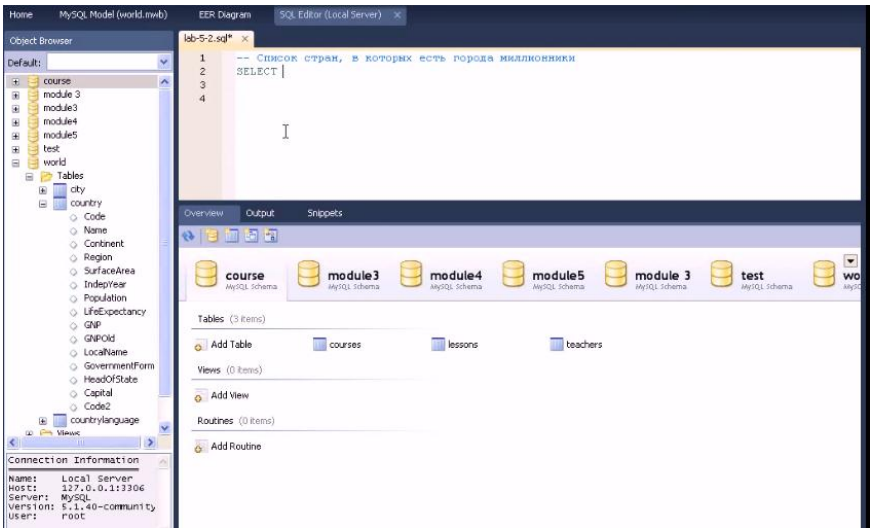
მაგალითის სახით შევასრულოთ შემდეგი მოთხოვნა:

„გამოვიტანოთ იმ ქვეყნების სია, რომელთაც გააჩნიათ ქალაქები 1 მლნ.-ზე მეტი მცხოვრებით“.

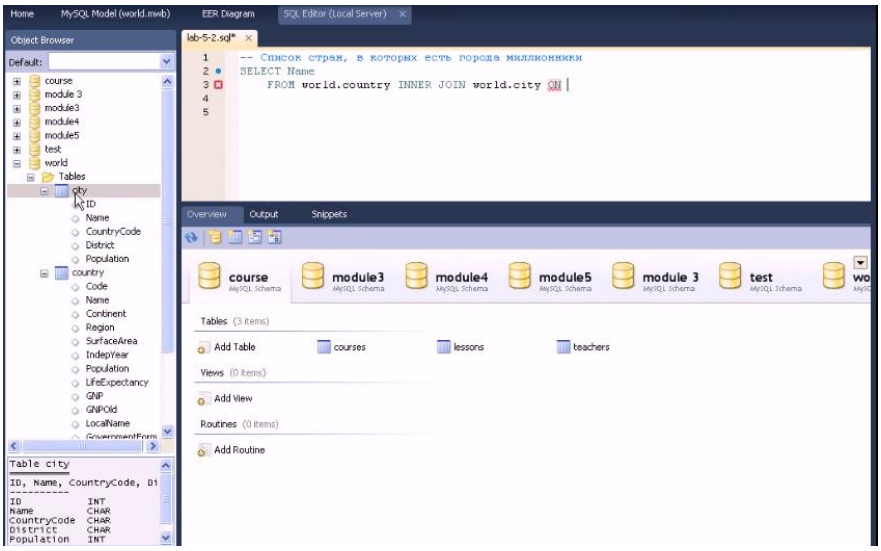
Workbench -ში გავხსნათ SQL Editor-ის ფანჯარა და შევუდგეთ მოთხოვნის ფორმირებას.



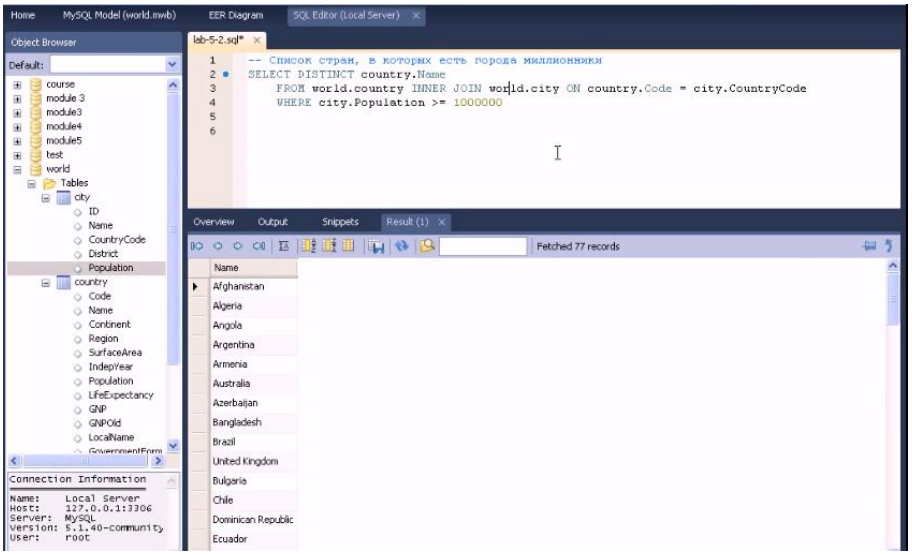
სურ. 5.11.



სურ. 5.12.



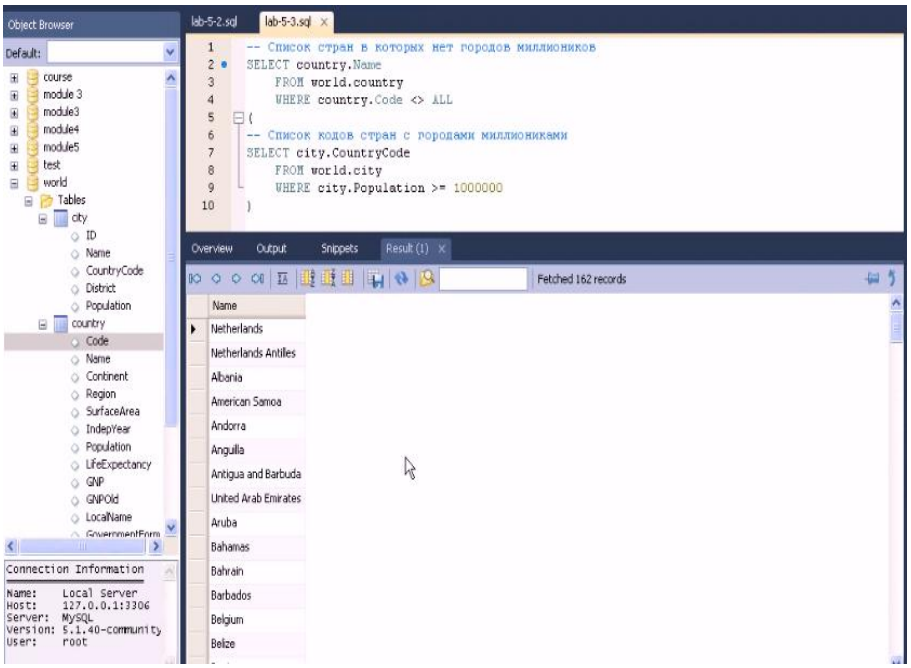
სურ. 5.13.



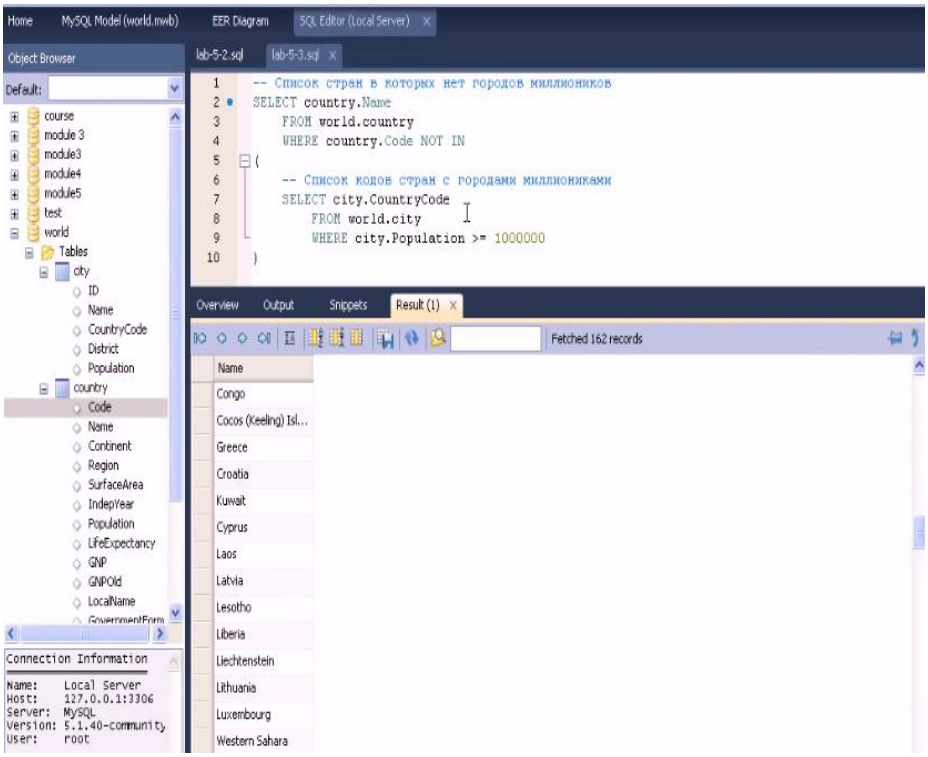
სურ. 5.14.

ახლა შევასრულოთ შემდეგი მოთხოვნა:

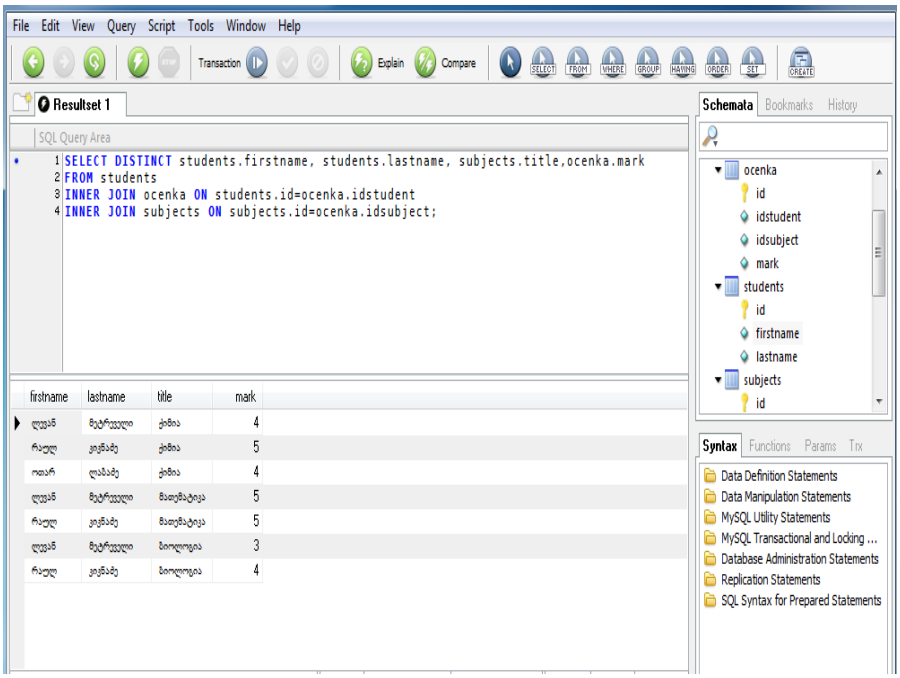
„გამოვიტანოთ იმ ქვეყნების სია, რომელთაც არ გააჩნიათ ქალაქები 1 მლნ.-ზე მეტი მცხოვრებით“.



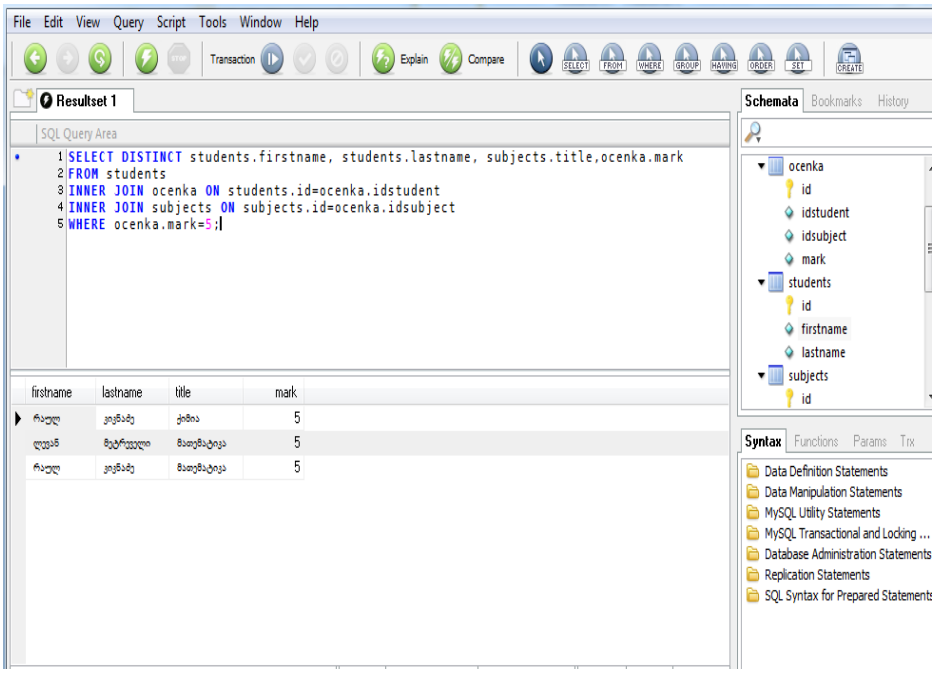
სურ. 5.15.



სურ. 5.16.



სურ. 5.17.



სურ. 5.18.

ლაბორატორიული სამუშაო 6
წარმოდგენები

სამუშაოს მიზანი:

1. წარმოდგენებთან მუშაობა კონსოლში;
2. წარმოდგენებთან მუშაობა MySQL Query Browser - ში;
3. წარმოდგენებთან მუშაობა Workbench -ში.

წარმოდგენა (view) არის შენახული მოთხოვნა ე.წ. ვირტუალური ცხრილი, რომელიც ფიზიკურად არ არსებობს, არამედ დინამიურად იწარმოება ერთი ან მეტი ცხრილიდან და/ან სხვა წარმოდგენიდან. View-ს საშუალებით შეიძლება მონაცემების ცვლილება იმ ცხრილებში, რომლებიც წარმოდგენაში მონაწილეობენ.

1. მუშაობა კონსოლში

View -ს შექმნა და ნახვა

<u>სინტაქსისი :</u>

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

მაგალითი :

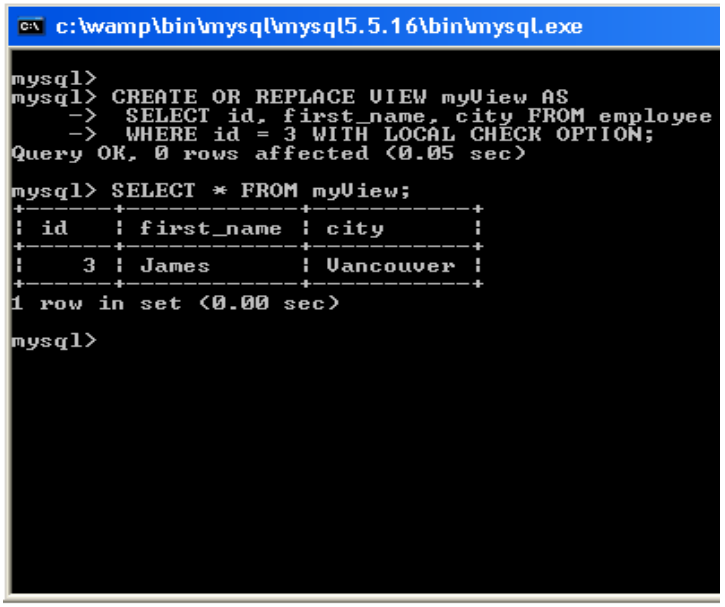
```
CREATE TABLE Employee(
id int,
first_name VARCHAR(15),
last_name VARCHAR(15),
start_date DATE,
end_date DATE,
salary FLOAT(8,2),
city VARCHAR(10),
description VARCHAR(15)
);
```

მონაცემების შეტანის შემდეგ ვქმნით **view** -ს:

```
CREATE OR REPLACE VIEW myView AS
SELECT id, first_name, city FROM employee
WHERE id = 3 WITH LOCAL CHECK OPTION;
```

ვნახულობთ მიღებულ წარმოდგენას როგორც ცხრილს:

```
SELECT * FROM myView;
```



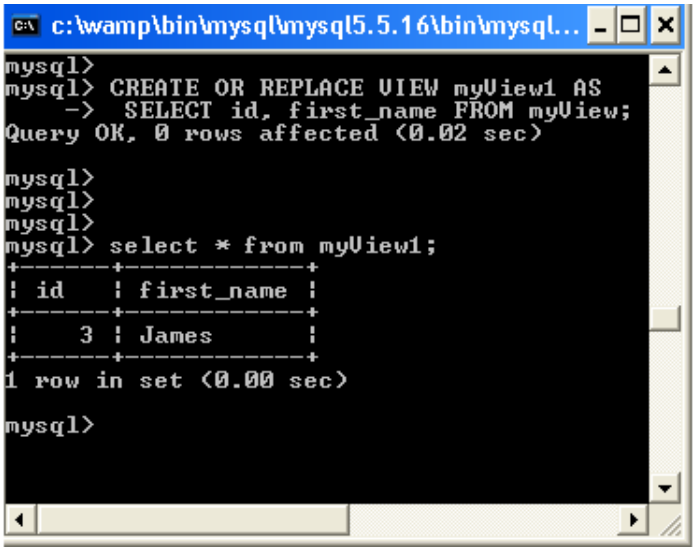
სურ. 6.1.

მიღებული წარმოდგენიდან შევქმნათ ახალი წარმოდგენა:

```
mysql> CREATE OR REPLACE VIEW myView1 AS
-> SELECT id, first_name FROM myView;
```

ვნახულობთ მიღებულ ახალ წარმოდგენას:

```
select * from myView1;
```



სურ. 6.2.

წარმოდგენის ნახვა შესაძლებელია ბრძანებით:

```
SHOW CREATE VIEW databasename.viewname
```

ან

```
EXEC sp_helptext 'databasename.viewname'

mysql> SHOW CREATE VIEW myView\G
```

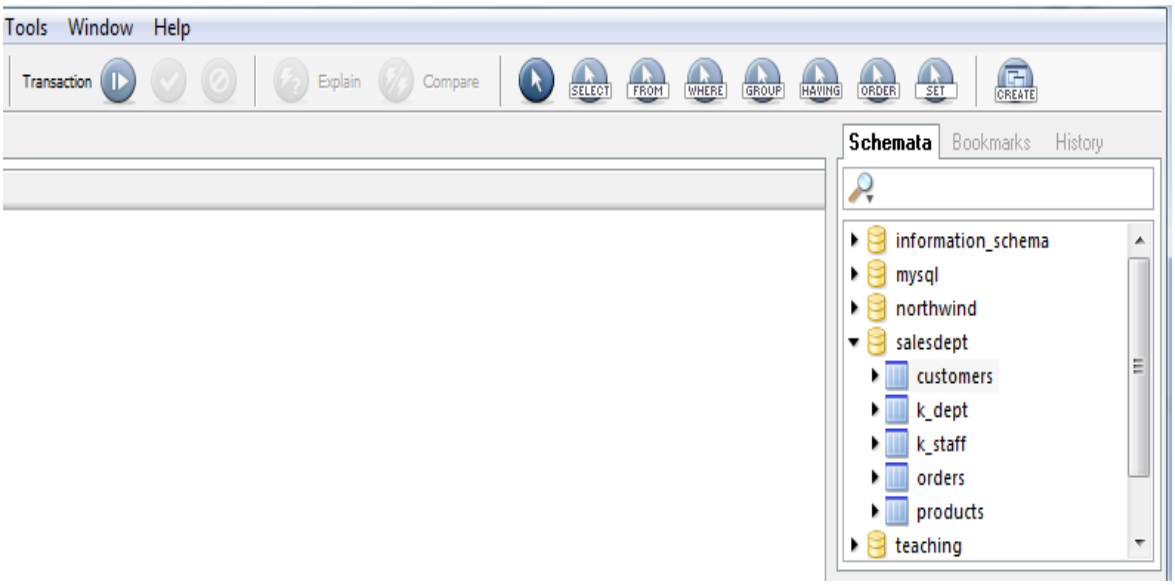
View -ს ამოგდება :

```
mysql> drop view myView1;
```

2. მუშაობა MySQL Query Browser -ში

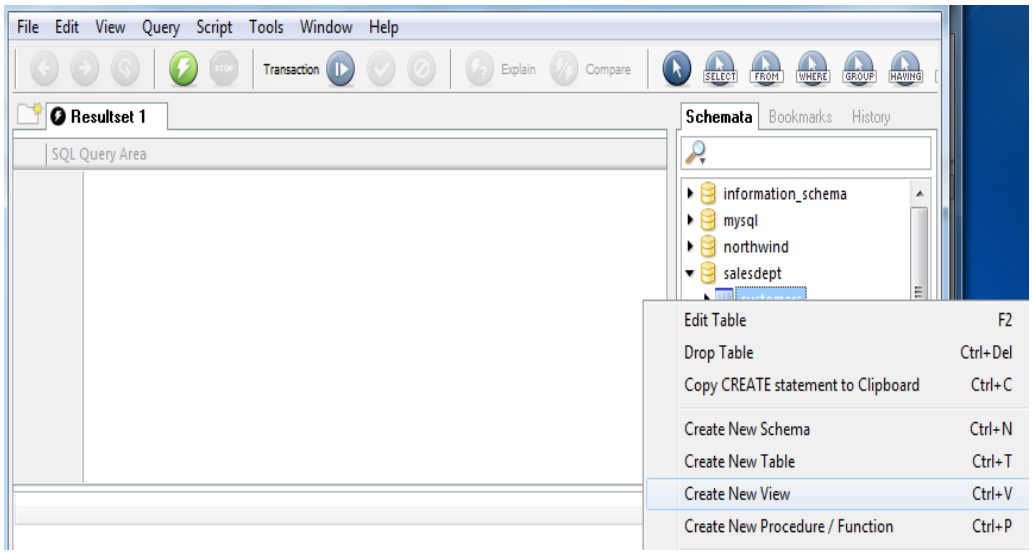
View-ს შექმნა

MySQL Query Browser-ის მეშვეობით წარმოდგენის შექმნის უმარტივესი გზა არის Create View ღილაკის გამოყენება. ამისათვის ჯერ ვქმნით მოთხოვნას, რომლის მიხედვითაც გვინდა წარმოდგენის შექმნა. მოთხოვნის შესრულების შემდეგ ვაწკაპუნებთ Create View ღილაკზე და ვანიჭებთ სახელს.



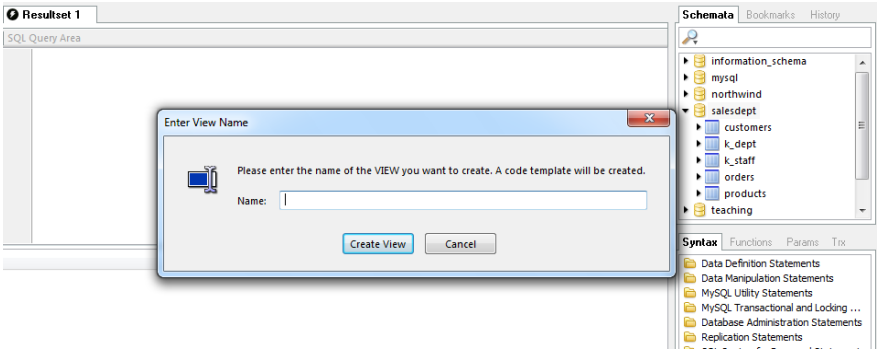
სურ. 6.3.

არსებობს მეორე გზაც. database browser-ში მონაცემთა ბაზაზე მარჯვენა დაწკაპუნებით გაიხსნება კონტექსტური მენიუ, სადაც ვირჩევთ **Create New View** ოპციას.



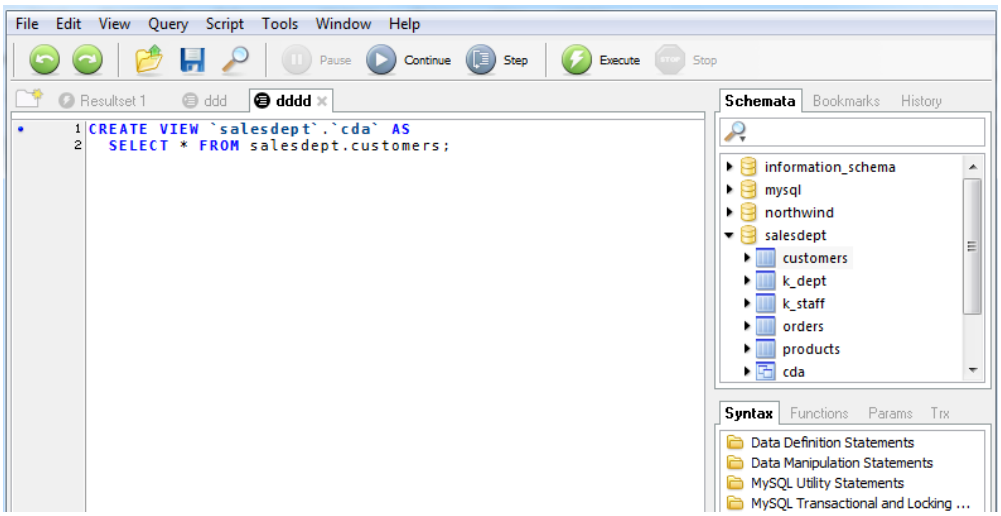
სურ. 6.4.

შემდეგ წარმოდგენას ვანიჭებთ სახელს option.



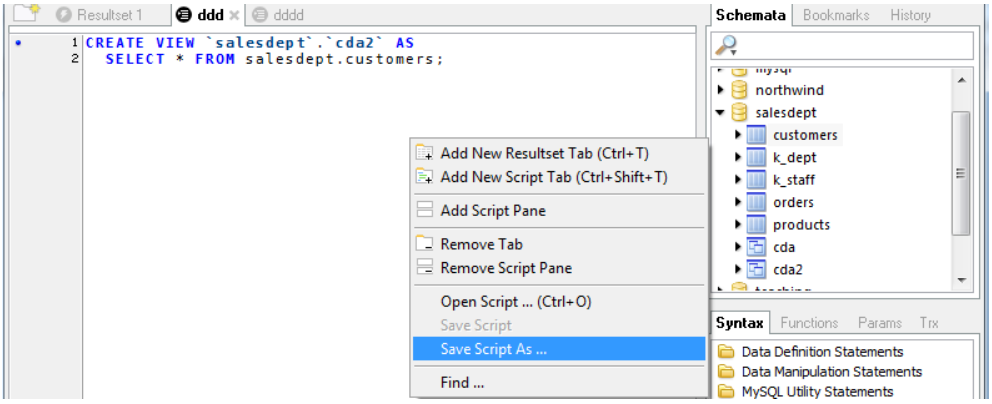
სურ. 6.5.

წარმოდგენის ბრძანების კოდი ჩნდება Script Editor-ში, რომელსაც Execute ღილაკზე დაჭერით ვუშვებთ შესრულებაზე. შედეგად მიიღება წარმოდგენა, რომელიც აისახება Schemata არეში.



სურ. 6.6.

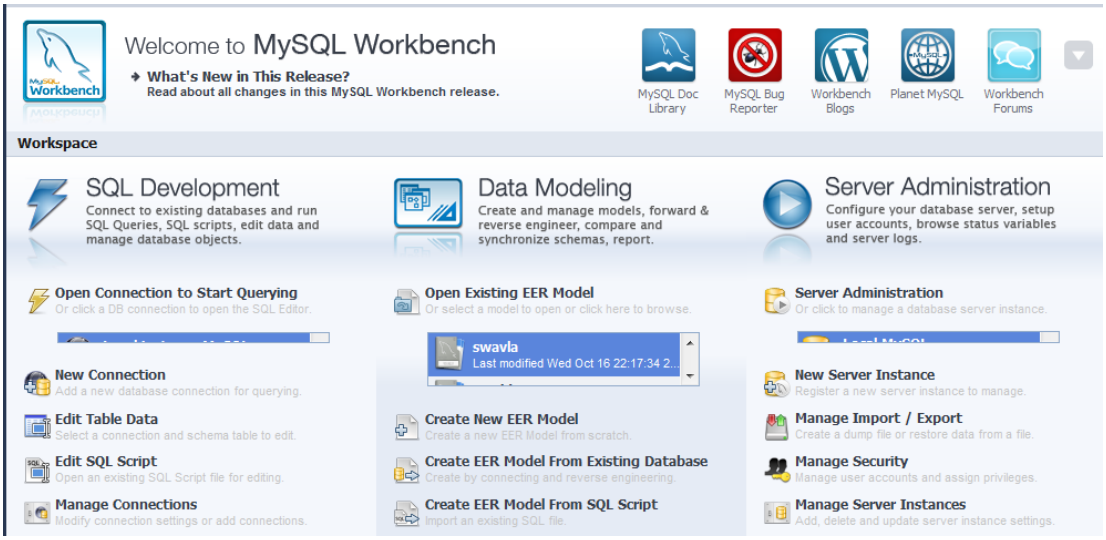
და ბოლოს, წარმოდგენას ვინახავთ კონტექსტური მენიუს მეშვეობით.



სურ. 6.7.

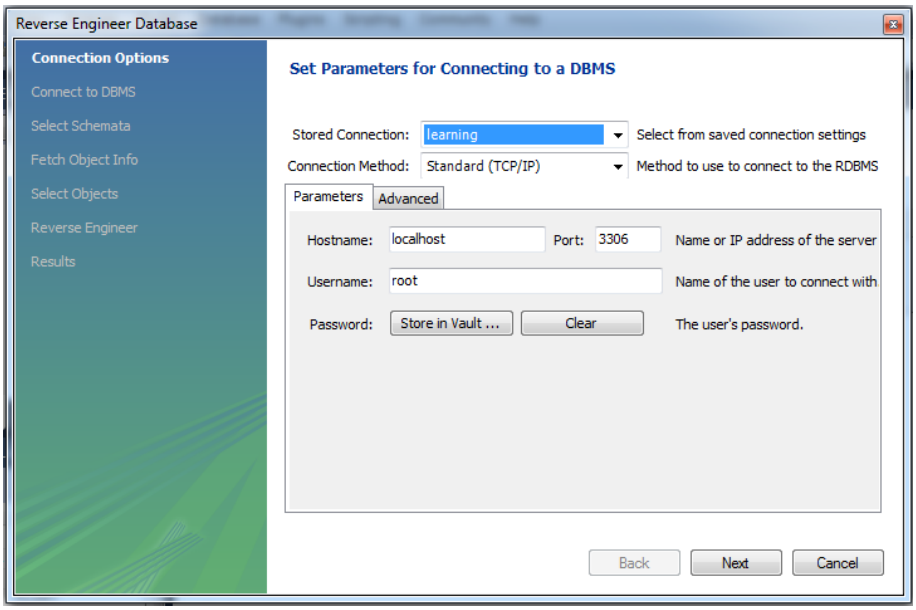
3. მუშაობა Workbench -ში

განვიხილოთ Workbench-ის მეშვეობით წარმოდგენის შექმნის თავისებურებანი. თუ წარმოდგენას ვქმნით მონაცემთა ბაზისათვის, რომლის EER დიაგრამა არ იყო შექმნილი, მაშინ ჯერ ეს საკითხი უნდა მოვაგვაროთ. ამისათვის Workbench-ის ფანჯრის Data Modeling-ის განყოფილებაში ვირჩევთ ბრძანებას Create EER Model From Existing Database.



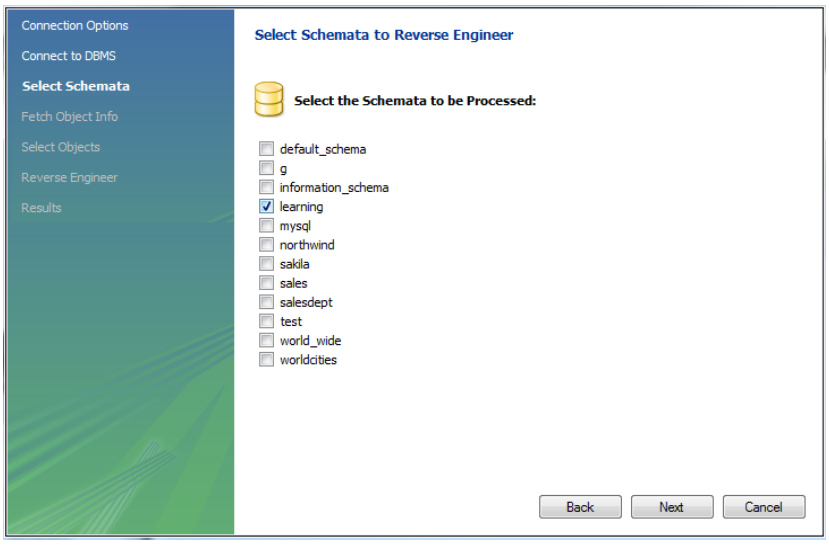
სურ. 6.8.

იხსნება Set Parameters for Connecting to a DBMS ფანჯარა, სადაც ჩამოშლადი სიიდან ვირჩევთ მონაცემთა ბაზას. შემდეგ ვაჭერთ ღილაკს Next.



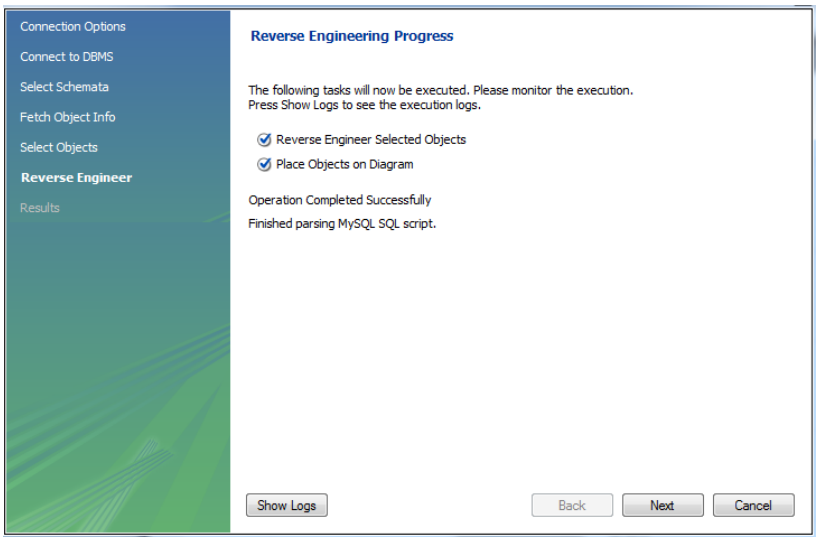
სურ. 6.9.

შემდეგ ფანჯარაში Select Schemata to Reverse Engineer ვაყენებთ ალამს მონაცემთა ბაზაზე და ვაჭერთ ღილაკს Next.



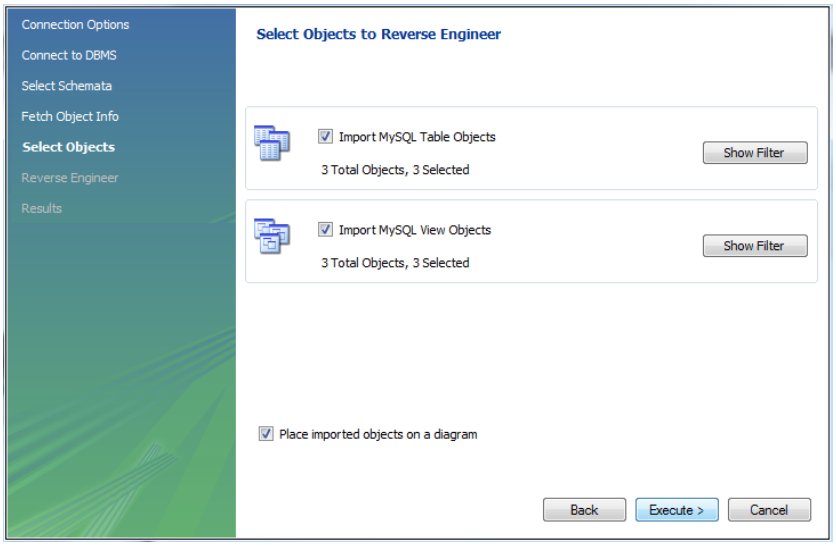
სურ. 6.10.

არაფერს არ ვცვლით. ვაჭერთ ღილაკს Next.



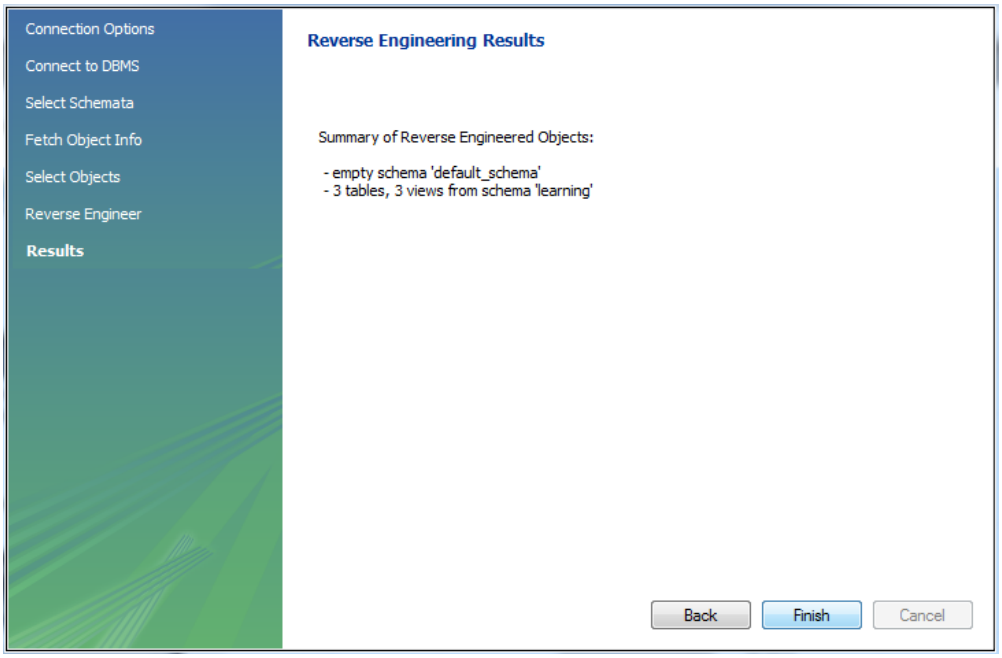
სურ. 6.11.

ვაჭერთ ღილაკს Execute.



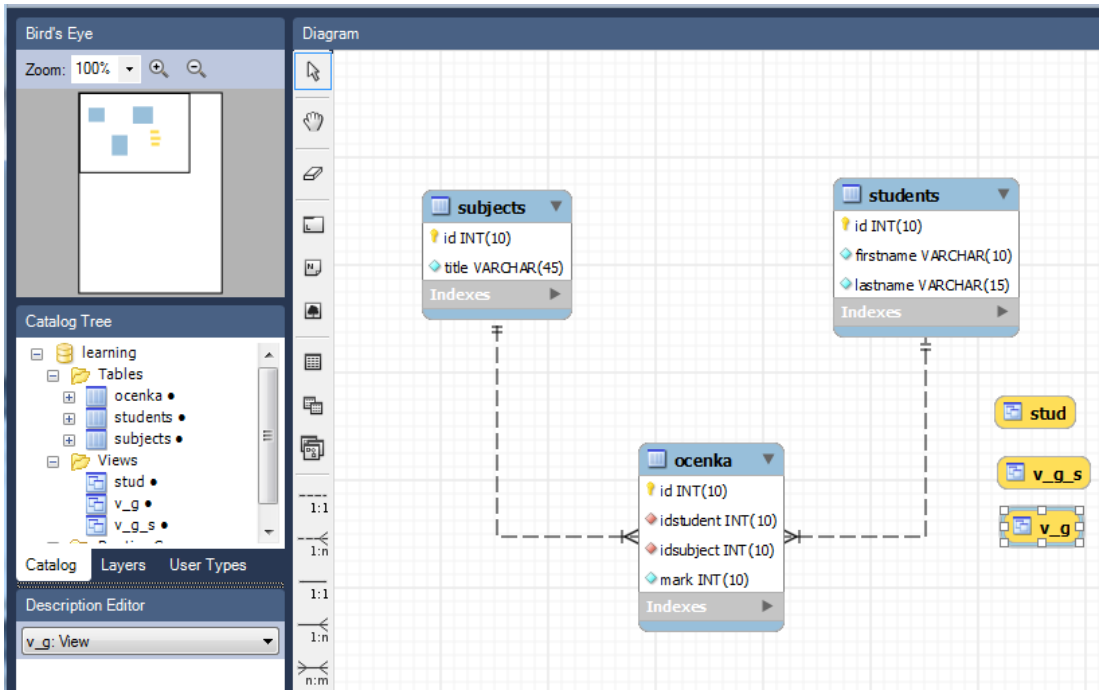
სურ. 6.12.

ვაჭერთ ღილაკს Finish.

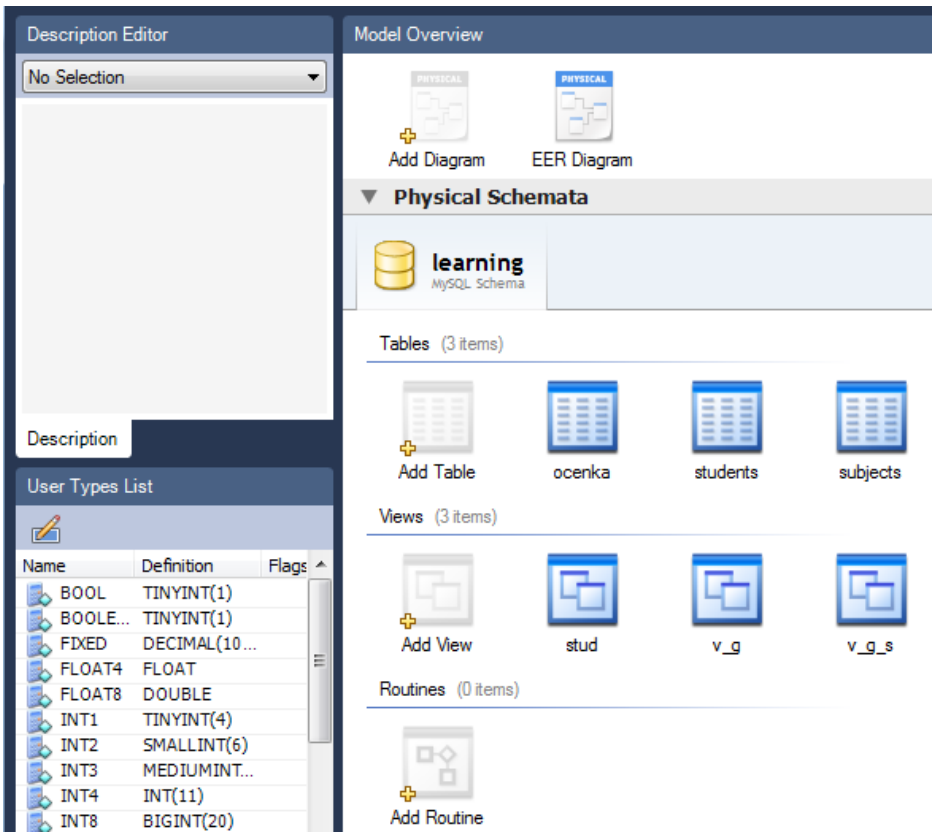


სურ. 6.13.

ამით სრულდება მონაცემთა ბაზისათვის დიაგრამის აგების პროცედურა. Catalog Tree -ში იხსნება მონაცემთა ბაზა ცხრილებით და არსებული წარმოდგენებით.

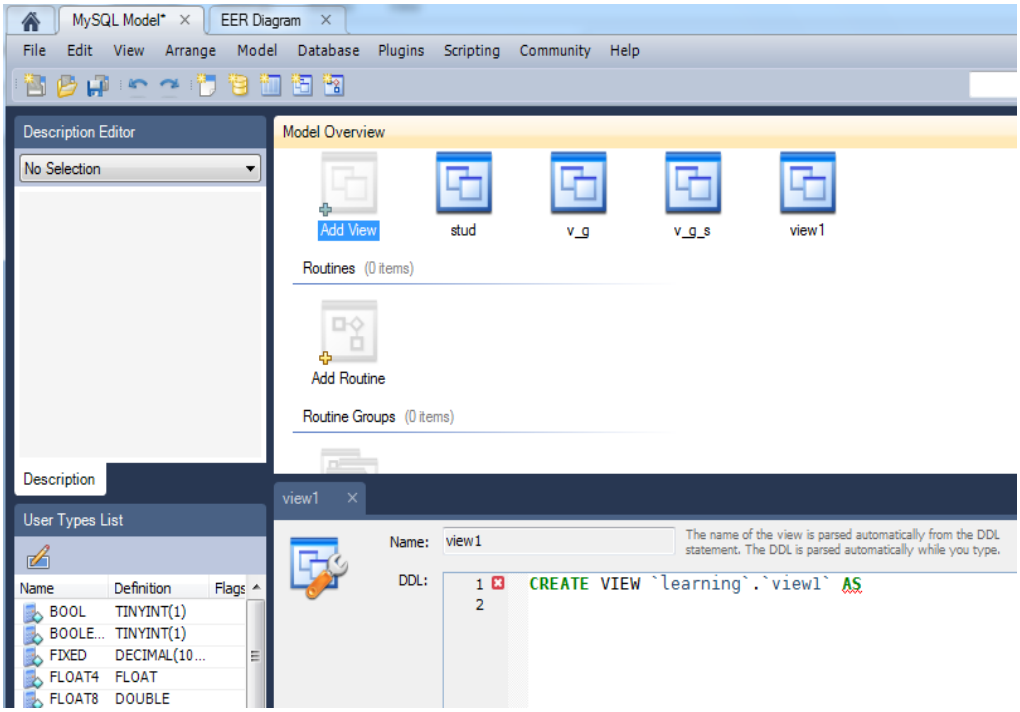


სურ. 6.14.



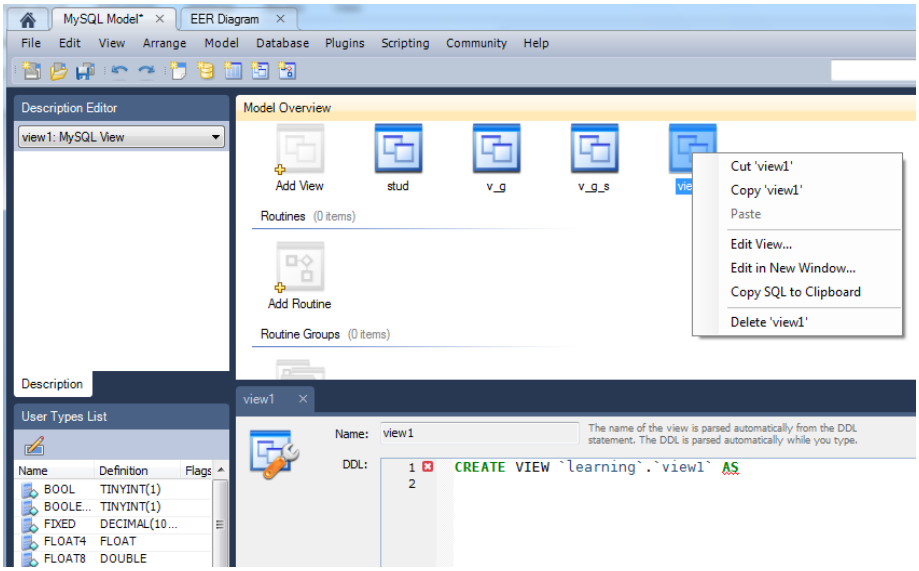
სურ. 6.15.

Physical Schemata-ში წარმოდგენის დამატებისათვის **MySQL Model** გვერდზე ორჯერ ვაწკაპუნებთ **Add View** იკონაზე **Physical Schemata** სექციაში. ახლადშექმნილ წარმოდგენას უსიტყვოდ ენიჭება სახელი **view1**. თუ ასეთი სახელით უკვე არსებობს წარმოდგენა, მაშინ - **view2**. ავტომატურად იხსნება წარმოდგენის რედაქტორი.



სურ. 6.16.

view1-ზე მარჯვენა დაწკაპუნებით იხსნება pop-up მენიუ შემდეგი პუნქტებით:



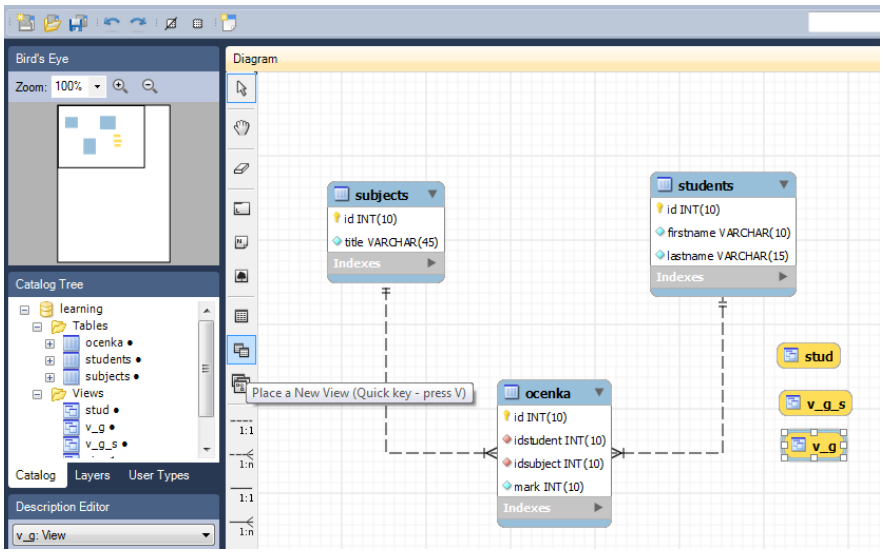
სურ. 6.17.

- Cut '***view_name***'
- Copy '***view_name***'
- Paste
- Edit View...
- Edit in New Window...
- Copy SQL to Clipboard
- Delete '***view_name***': შლის როგორც EER დიაგრამიდან, ისე სქემიდან.
- Remove '***view_name***': შლის EER დიაგრამიდან, მაგრამ არა სქემიდან.

Cut და Copy ძალიან სასარგებლოა სხვადასხვა სქემებს შორის წარმოდგენების კოპირებისათვის.

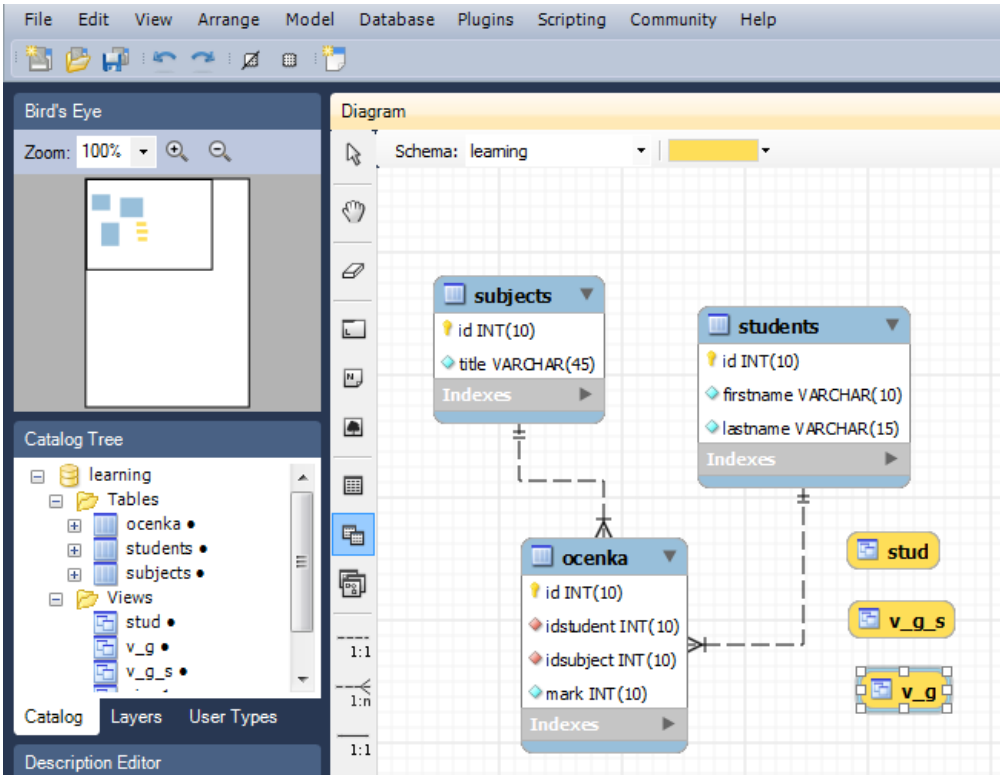
EER დიაგრამაზე წარმოდგენის დამატება

ამისათვის ჯერ უნდა დავრწმუნდეთ, რომ EER Diagram ჩანართი გახსნილია. შემდეგ ვერტიკალურ ინსტრუმენტების პანელზე ვირჩევთ და ვაწკაპუნებთ Viewtool - ზე.

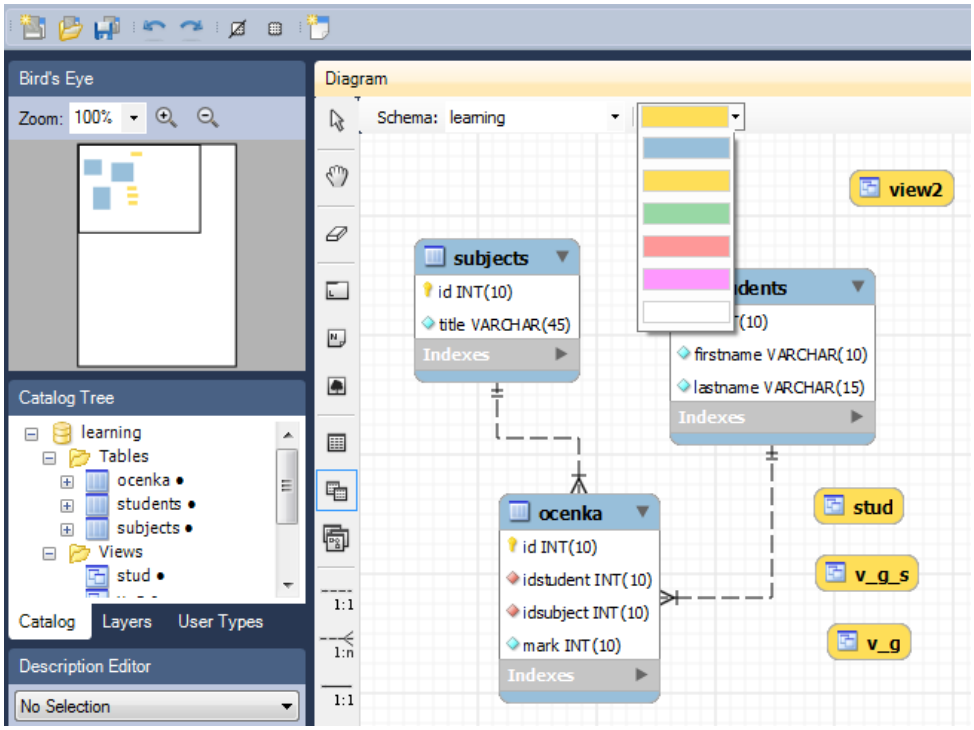


სურ. 6.18.

დიაგრამაზე ჩნდება მართკუთხედი.

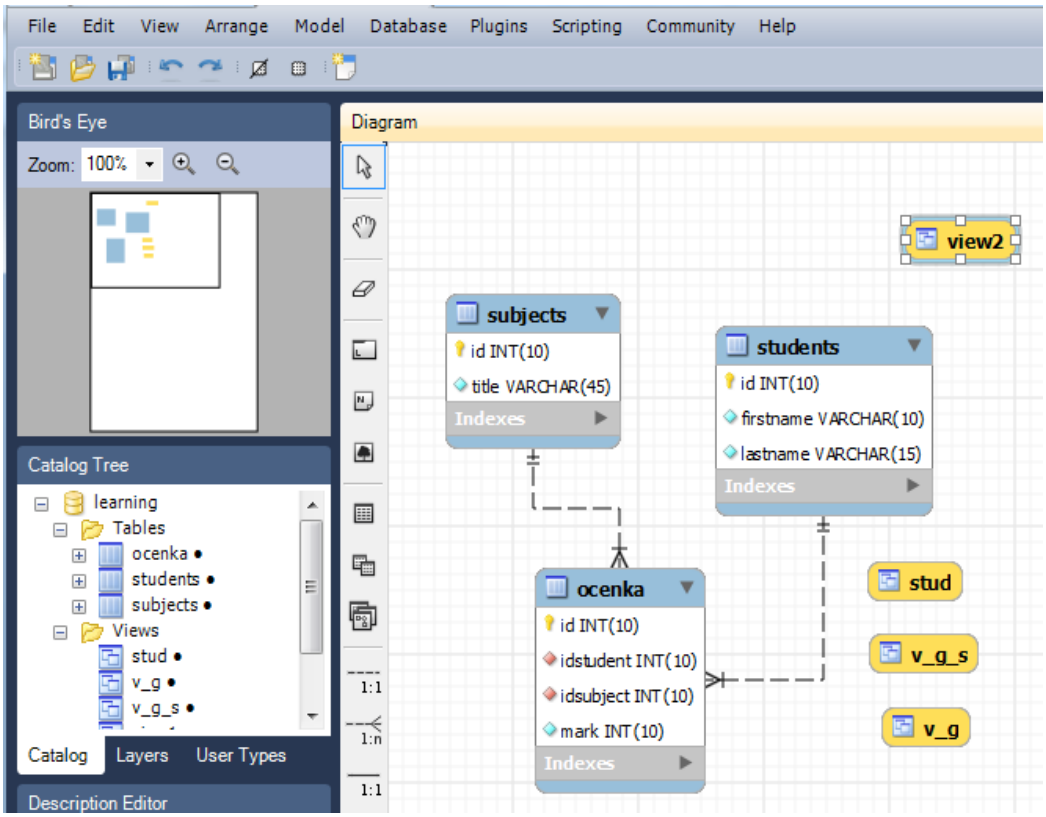


სურ. 6.19.



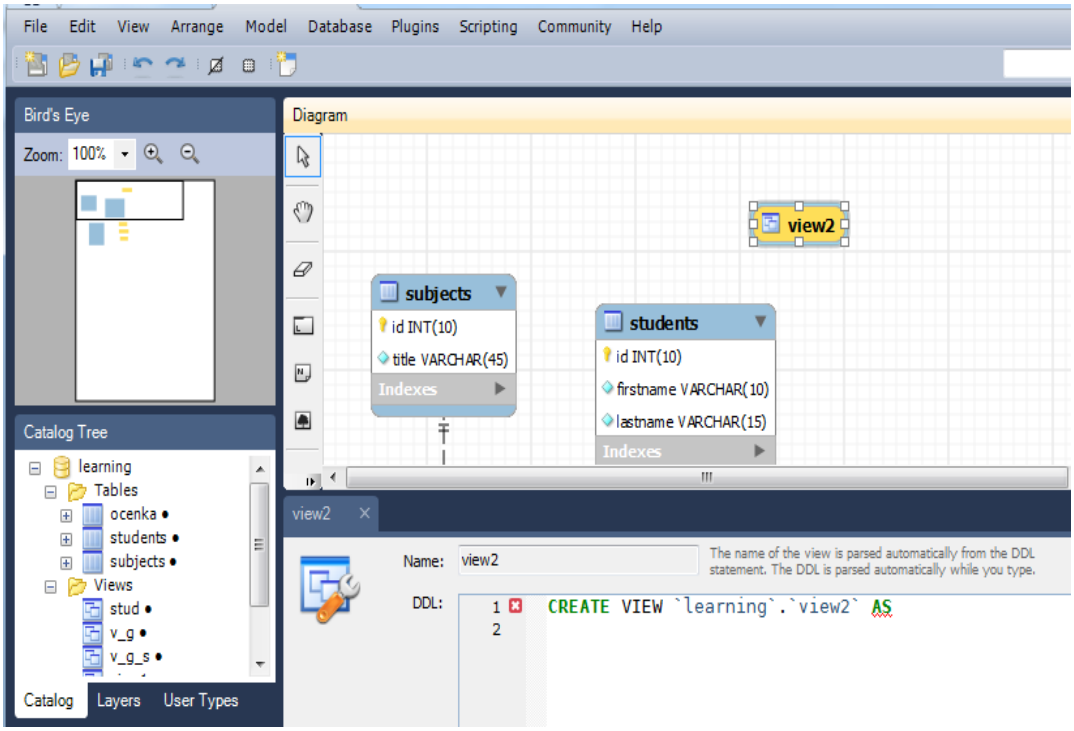
სურ. 6.20.

ვაწკაპუნებთ EER Diagram-ის ნებისმიერ ადგილას. ჩნდება ახალი წარმოდგენა სახელწოდებით **view1**.



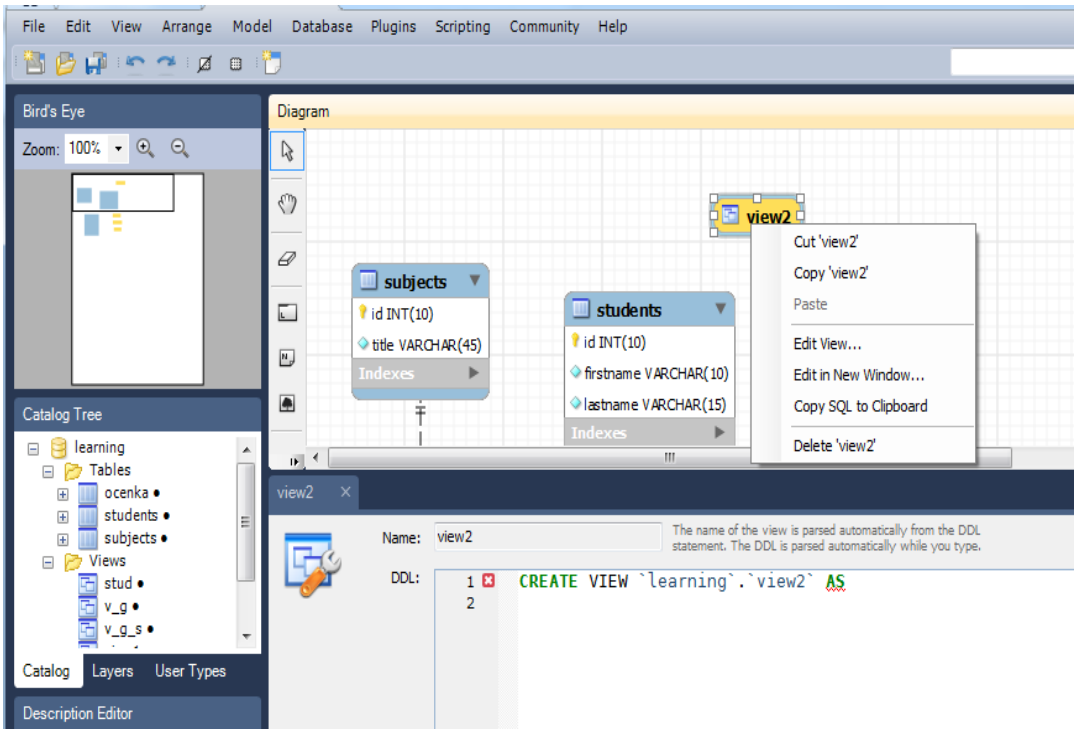
სურ. 6.21.

ორჯერ ვაწკაპუნებთ წარმოდგენაზე. ქვედა ნაწილში იხსნება DDL ბრძანების შაბლონი.



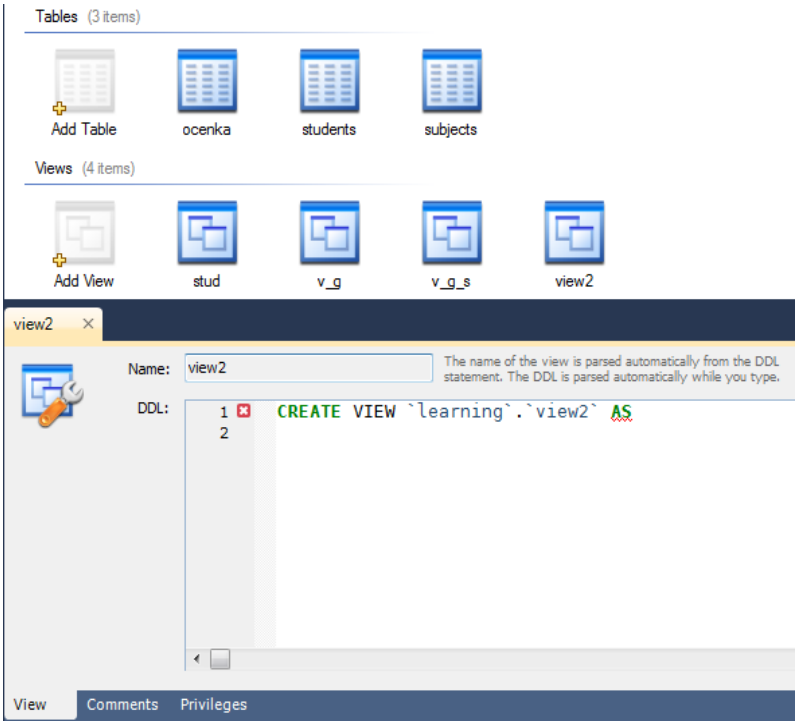
სურ. 6.22.

წარმოდგენაზე მარჯვენა დაწკაპუნებით იხსნება pop-up მენიუ ზემოჩამოთვლილი პუნქტებით.

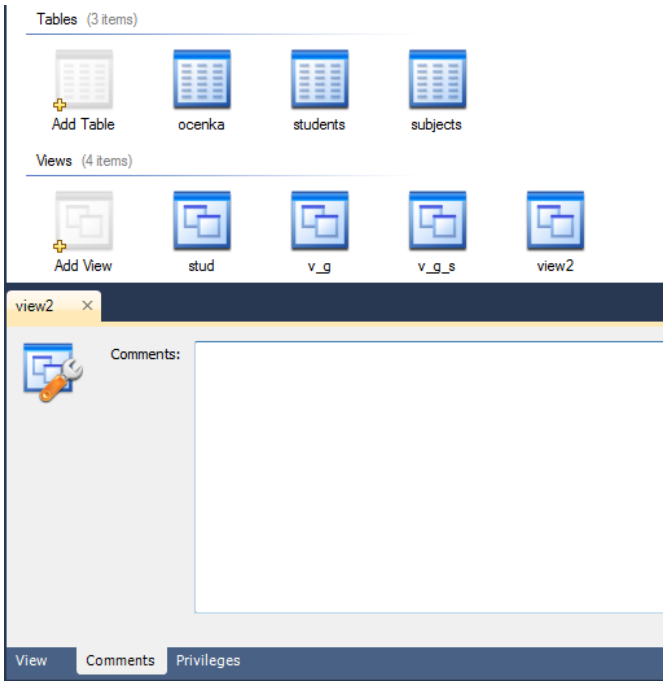


სურ. 6.23.

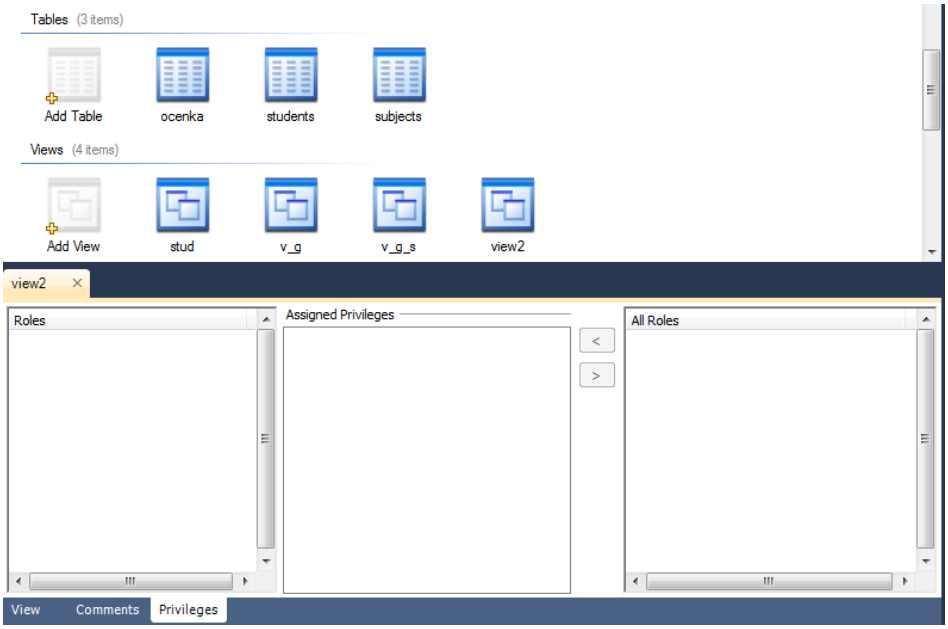
წარმოდგენის რედაქტორი (View Editor) ფანჯრის ქვედა ნაწილში შეიცავს სამ ჩანართს: **View**,**Comments**, და **Privileges**. მათ შორის გადაადგილება შესაძლებელია როგორც მაუსით, ისე კლავიატურიდან შემდეგი კომბინაციის დაჭერით: **Control+Alt+Tab**.



სურ. 6.24.



სურ. 6.25.



სურ. 6.26.

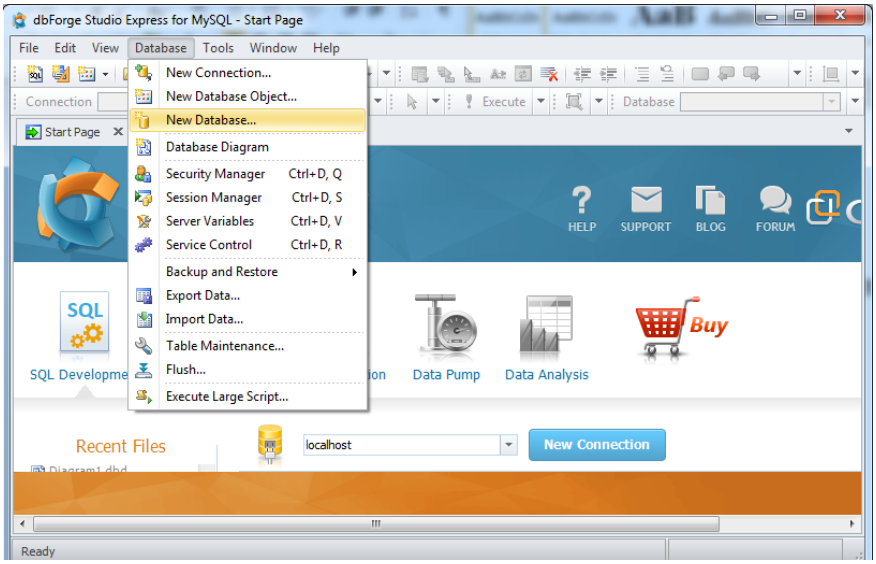
ლაბორატორიული სამუშაო 7
მუშაობა dbForge Studio for MySQL-ში

სამუშაოს მიზანი:

1. DDL: მონაცემთა ბაზის შექმნა
2. DML: მონაცემთა ამორჩევა dbForge Studio -ში

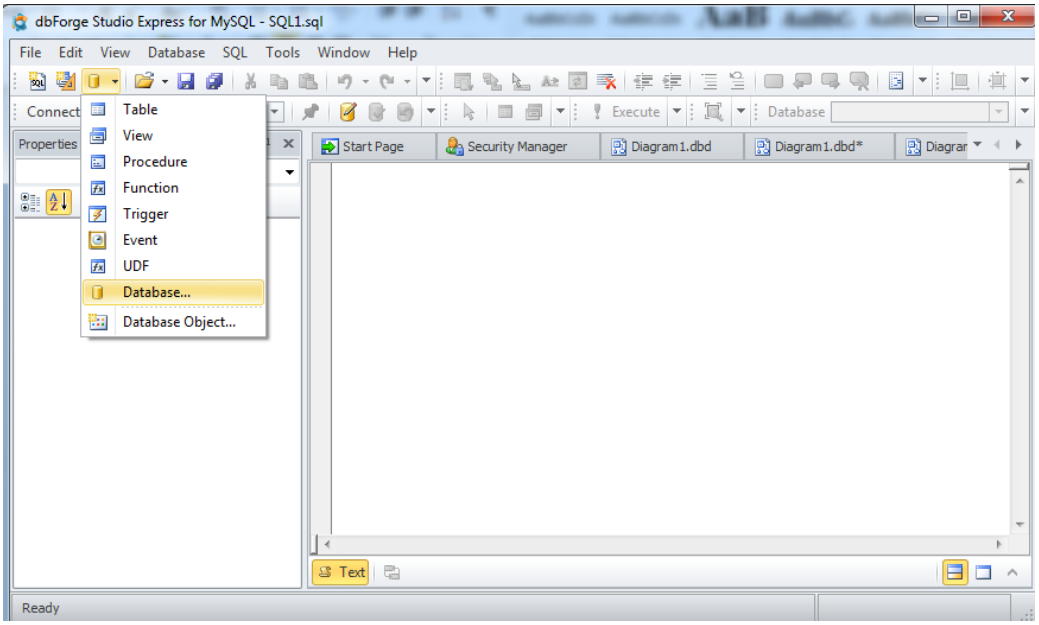
1. DDL: მონაცემთა ბაზის შექმნა

dbForge Studio for MySQL - ის მთავარ ფანჯარაში, მენიუს Database ბრძანებაში ვირჩევთ New Database - ს.



სურ. 7.1

ან ინსტრუმენტების პანელზე New Database Object პიქტოგრამას ჩამოვშლით და ვირჩევთ Database ობიექტს.



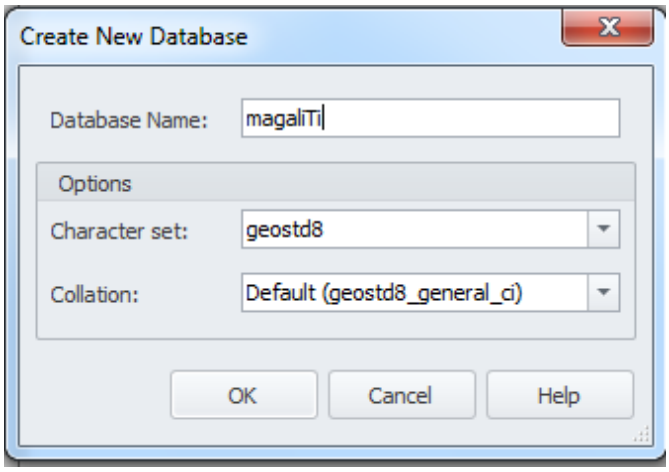
სურ. 7.2

იხსნება ფანჯარა Create New Database, სადაც ვავსებთ ველებს:

Database Name: მონაცემთა ბაზის სახელი;

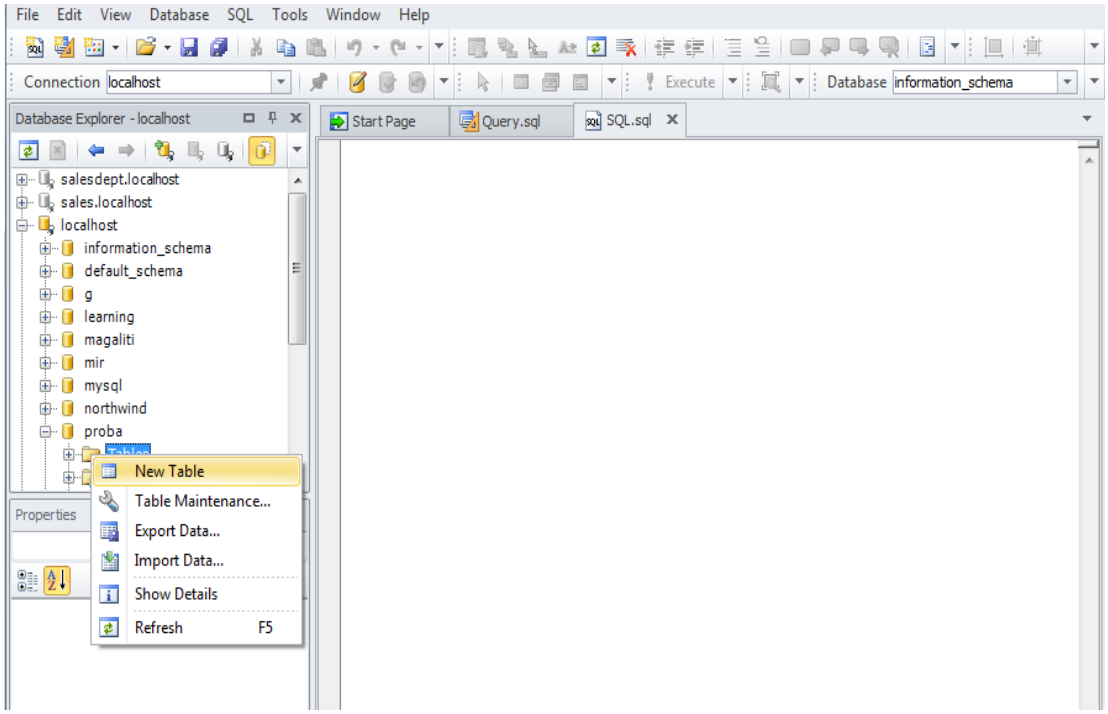
Character set: ვთქვათ geostd8;

Collation: ვთქვათ geostd8_general_ci.



სურ. 7.3

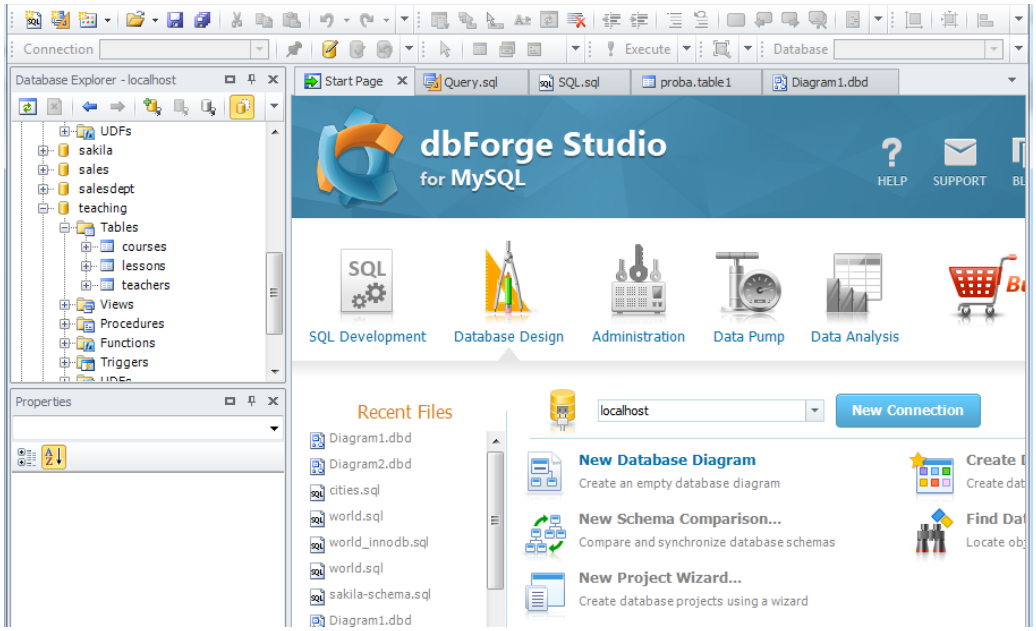
შემდეგ გადავდივართ ცხრილების შექმნაზე (დაწვრილებით ქვემოთ იქნება აღწერილი).



სურ. 7.4

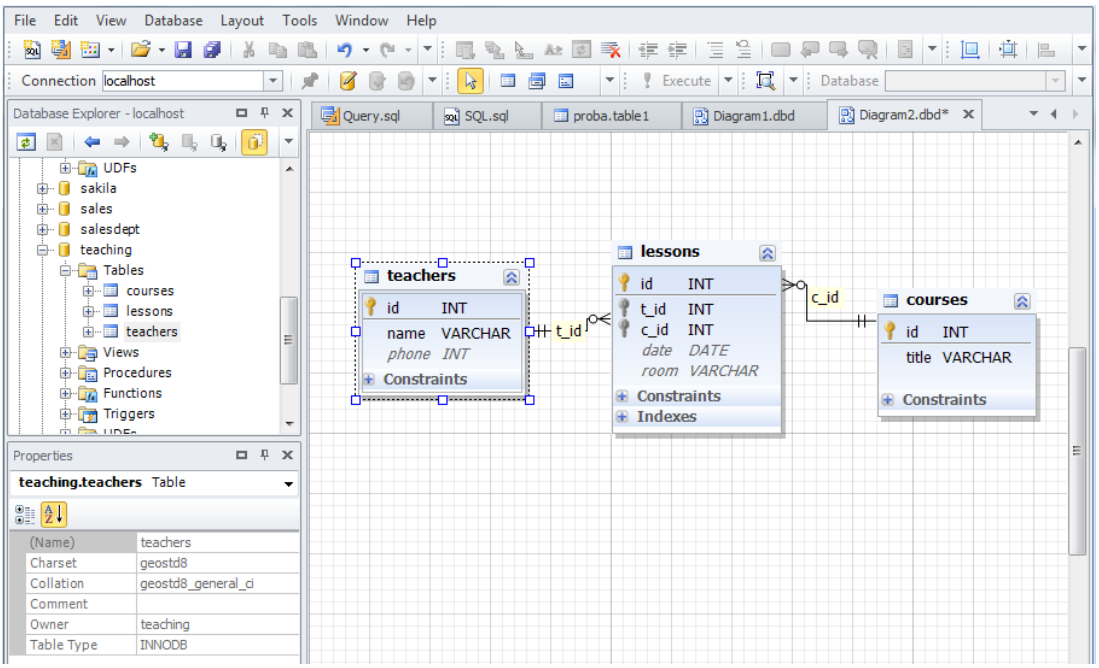
მონაცემთა ბაზის დიაგრამის აგება

მონაცემთა ბაზის დიაგრამის ასაგებად მთავარი ეკრანის Database Design მენიუში ვირჩევთ ბრძანებას New Database Diagram.



სურ. 7.5

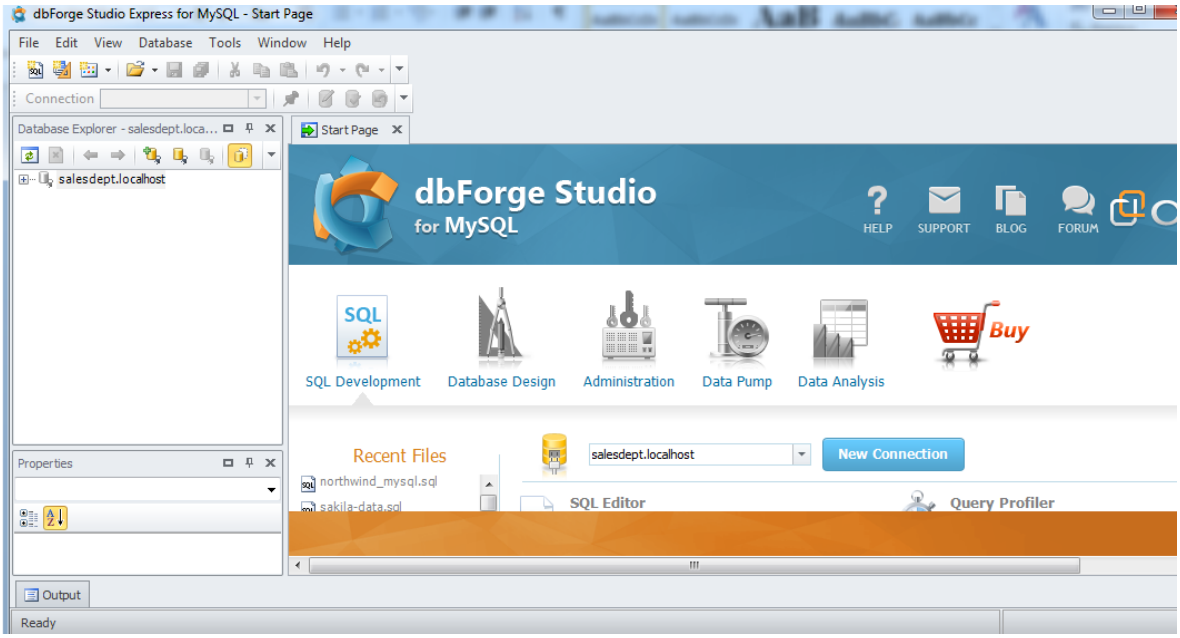
იხსნება დიაგრამის ფანჯარა. Database Explorer-ის ხისებრ სტრუქტურაზე ვპოულობთ ჩვენს მონაცემთა ბაზას, რომლის ცხრილები მონაცვლეობით გადმოგვაქვს დიაგრამის ნაწილში. ყველა ცხრილის გადმოტანის შემდეგ დიაგრამას ვარქმევთ სახელს და ვინახავთ.



სურ. 7.6

მონაცემთა ბაზასთან მიერთება

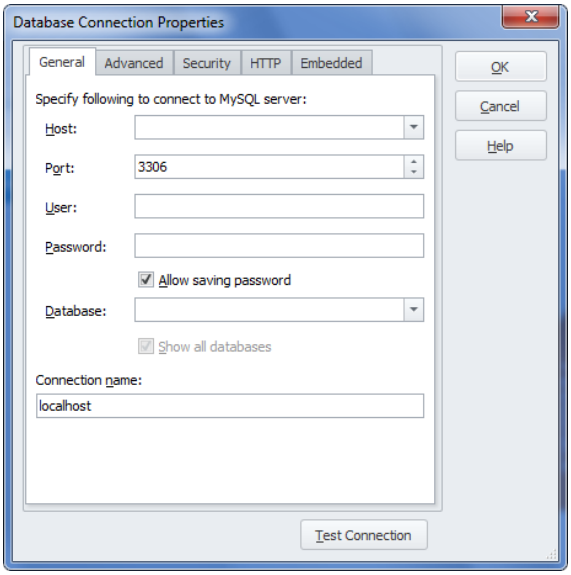
მონაცემთა ბაზასთან მიერთებამდე აუცილებელია სერვერის მიერთება.



სურ. 7.7

შეერთების შექმნა:

1. **Start** გვერდზე ვაწკაპუნებთ  **SQL Development** და შემდეგ ვაწკაპუნებთ ღილაკზე  **New Connection**. იხსნება **Database Connection Properties** დიალოგური ფანჯარა.



სურ. 7.8

2.**Host** ველში შეგვაქვს host-ის სახელი.

3.შეგვაქვს პორტის შესახებ ინფორმაცია. უსიტყვოდ პორტის ნომერი 3306.

4.შეგვაქვს მომხმარებლის login და პაროლი.

5. **Database** - ში ავკრიფოთ ან დავაწკაპუნოთ მონაცემთა ბაზაზე, რომელთანაც გვინდა მიერთება.

6. **Connection name** - ში ავტომატურად გენერირდება host - ის სახელი. ჩვენ შეგვიძლია განსხვავებული სახელის შექმნაც ახალი მიერთებისათვის.

7.ჩანართში **Advanced** შეგვიძლია დავაწკაპუნოთ მიერთების დამატებითი თვისებების კონფიგურირებისათვის.

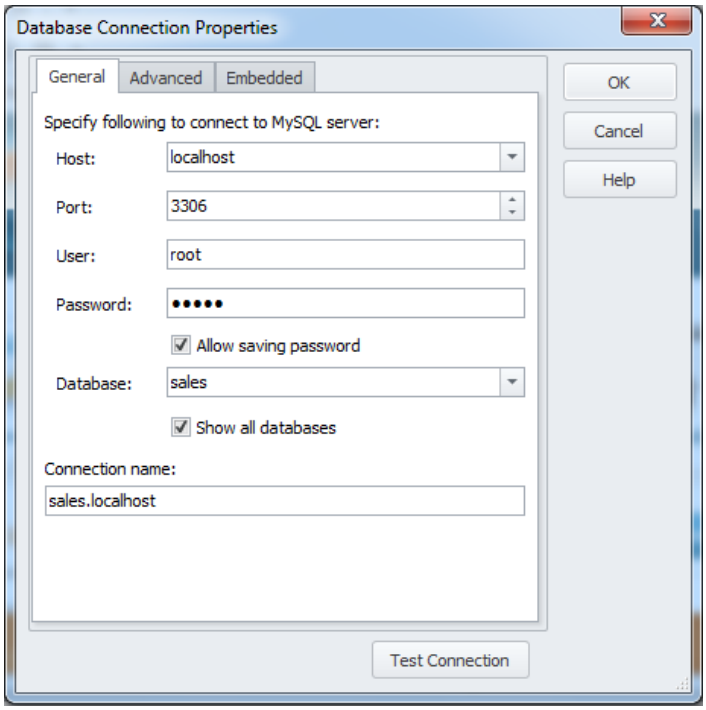
8. ჩანართში **Security** შეგვიძლია დავაწკაპუნოთ უსაფრთხოების თვისებების კონფიგურირებისათვის.

9. ჩანართში **HTTP** შეგვიძლია დავაწკაპუნოთ თვისებებზე tunnel properties.

10. ჩანართში **Embedded** ვაწკაპუნებთ to configure embedded server properties.

11. ჩანართში **Test Connection** ვაწკაპუნებთ მონაცემთა ბაზასთან მიერთების შესამოწმებლად.

12. მიერთების შესაქმნელად ვაწკაპუნებთ **OK** .



სურ. 7.8

მონაცემთა ბაზასთან მიერთების თვისებებთან წვდომის ალტერნატიული გზა:

Alternative ways to access the Database Connection Properties window:

1. მენიუში **Database** ვაწკაპუნებთ **New Connection.**

2. Database Explorer -ში ვაწკაპუნებთ **New Connection. Database Explorer** ეკრანზე გვიჩვენებს მონაცემთა ბაზების სიას.

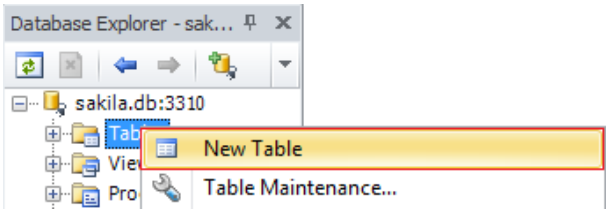
ცხრილის შექმნა Table Editor - ში

ცხრილის შესაქმნელად საჭიროა:

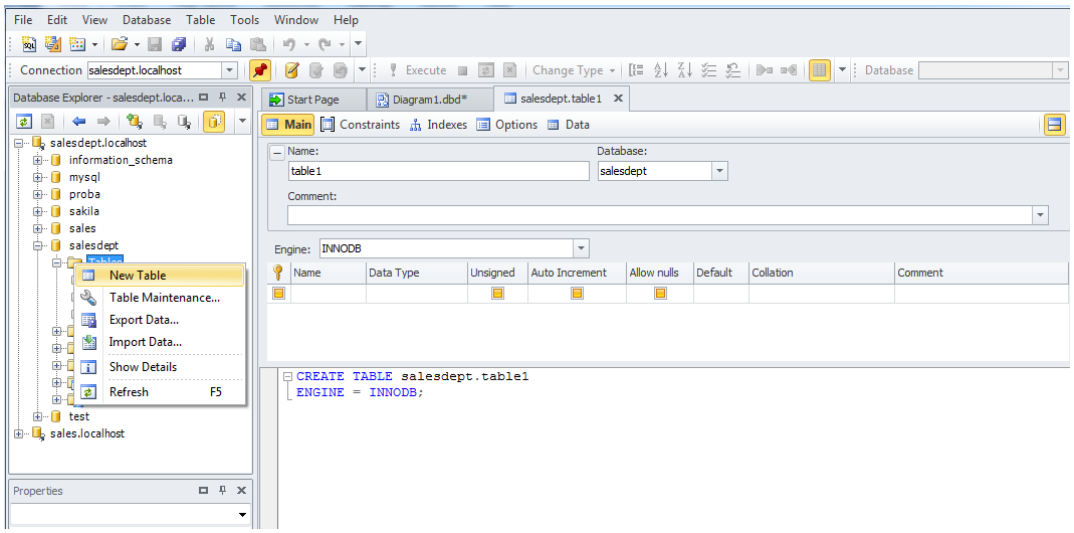
1.მონაცემთა ბაზასთან მიერთება.

2.**Database Explorer** -ში ორჯერ დავაწკაპუნოთ იმ მონაცემთა ბაზაზე, რომელშიც გვინდა ცხრილის შექმნა.

3.ვდგებით Tables საქაღალდეზე და მაუსის მარჯვენა ღილაკით გამოსულ მენიუში ვირჩევთ **New Table** -ს. გაიხსნება **Table Editor.**

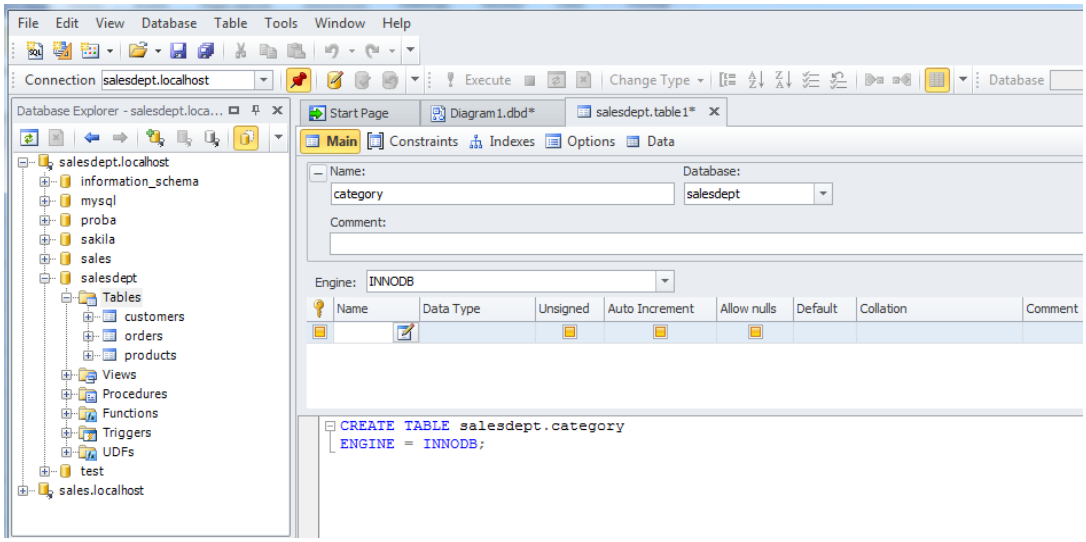


სურ. 7.9



სურ. 7.10

4.ველში **Name** შეგვაქვს ცხრილის სახელი.

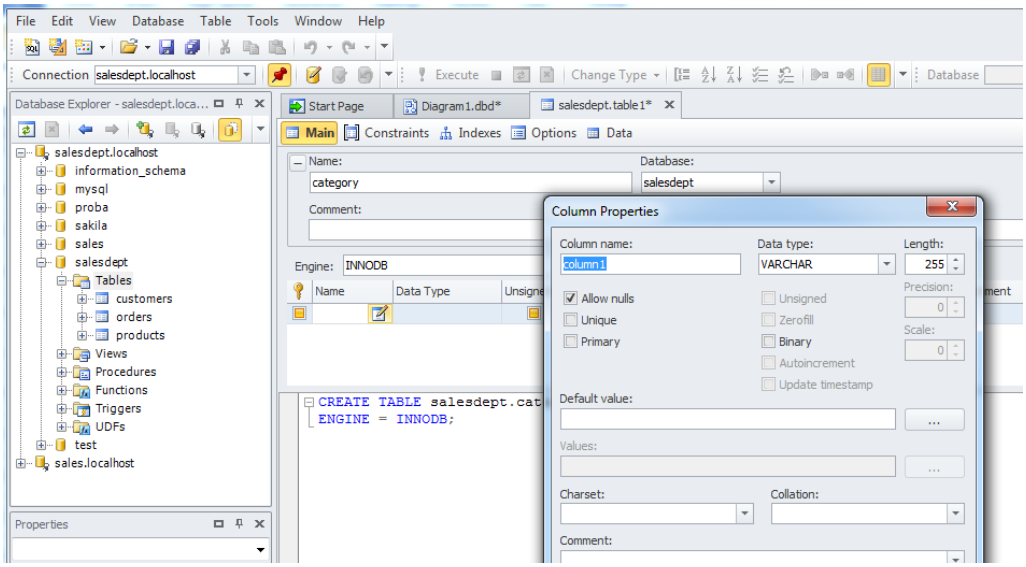


სურ. 7.11

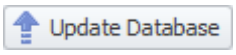
5. ქვემოთ გახსნილ ბადეზე ვაწკაპუნებთ პირველივე ცარიელ უჯრედზე და შეგვყავს სვეტის სახელი. შესაბამისად, განვსაზღვრავთ მის მონაცემთა ტიპსა და მას კი როგორც პირველად გასაღებს (primary key).

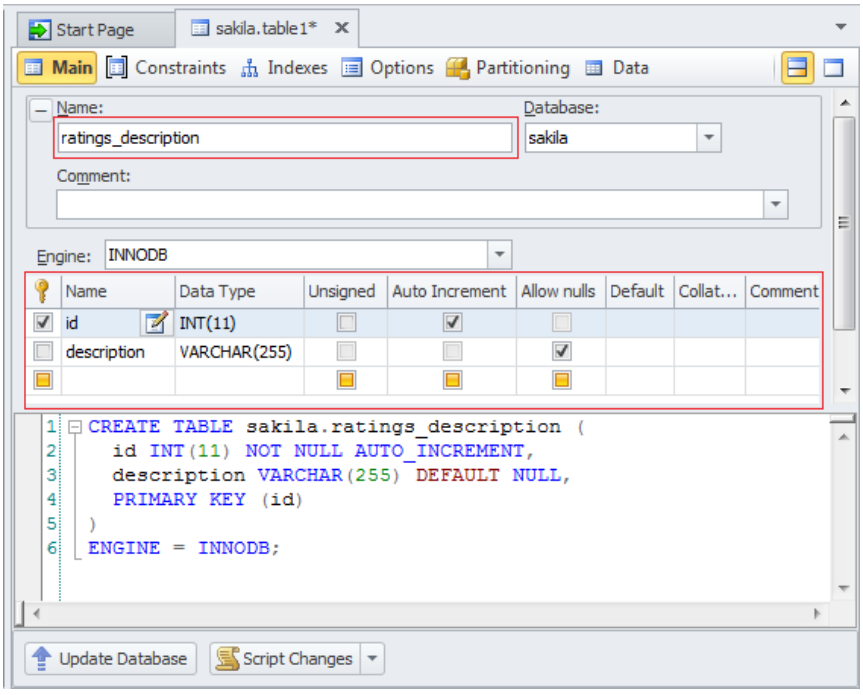
6. ვაწკაპუნებთ შემდეგ ცარიელ უჯრედზე და შეგვყავს სვეტის სახელი. განვსაზღვრავთ მის მონაცემთა ტიპს.

ამგვარად ვავსებთ მოცემული ცხრილისათვის ყველა სვეტის მახასიათებლებს.



სურ. 7.12

7. ვაწკაპუნებთ ღილაკზე  **Update Database**.

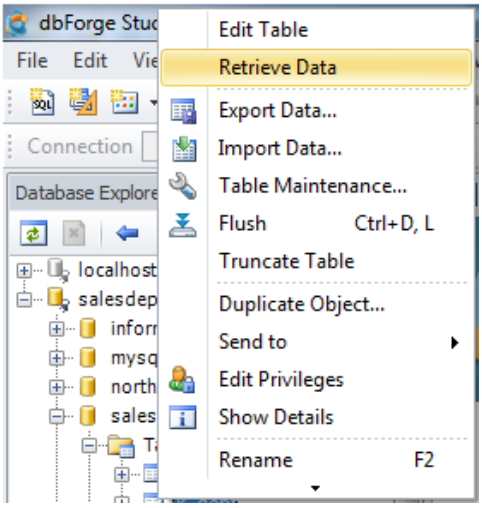


სურ. 7.13

ახალი ცარიელი ცხრილი შეიქმნა, რომელიც გამოჩნდება **Database Explorer** -ში **Tables** საქაღალდეში .

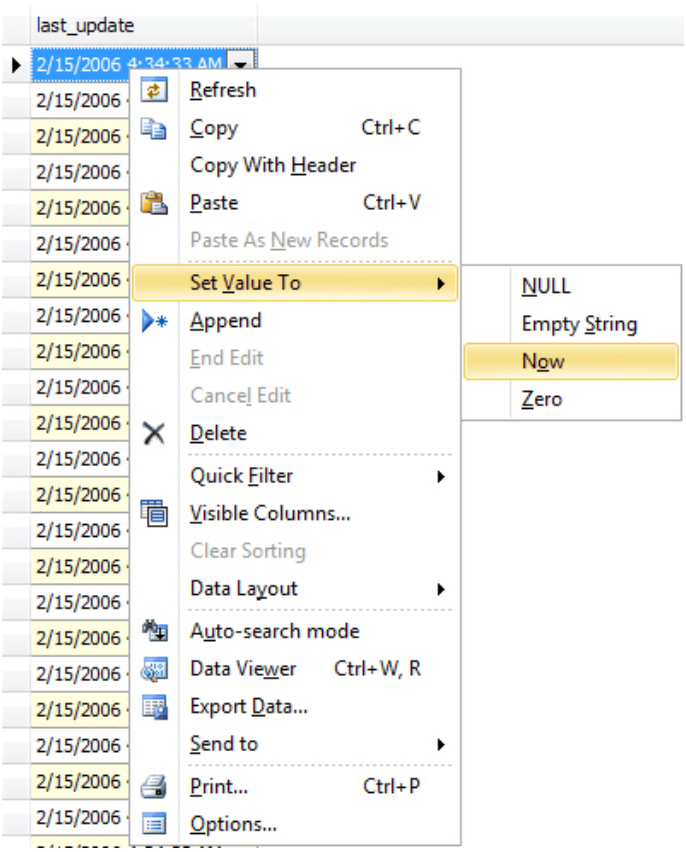
მონაცემთა შეტანა, წაშლა და კოპირება

- გამოვიყენოთ popup მენიუ ან შესაბამისი ღილაკი ბადის ქვეშ. იხსნება popup მენიუ, სადაც ვირჩევთ Retrieve Data-ს.



სურ. 7.14

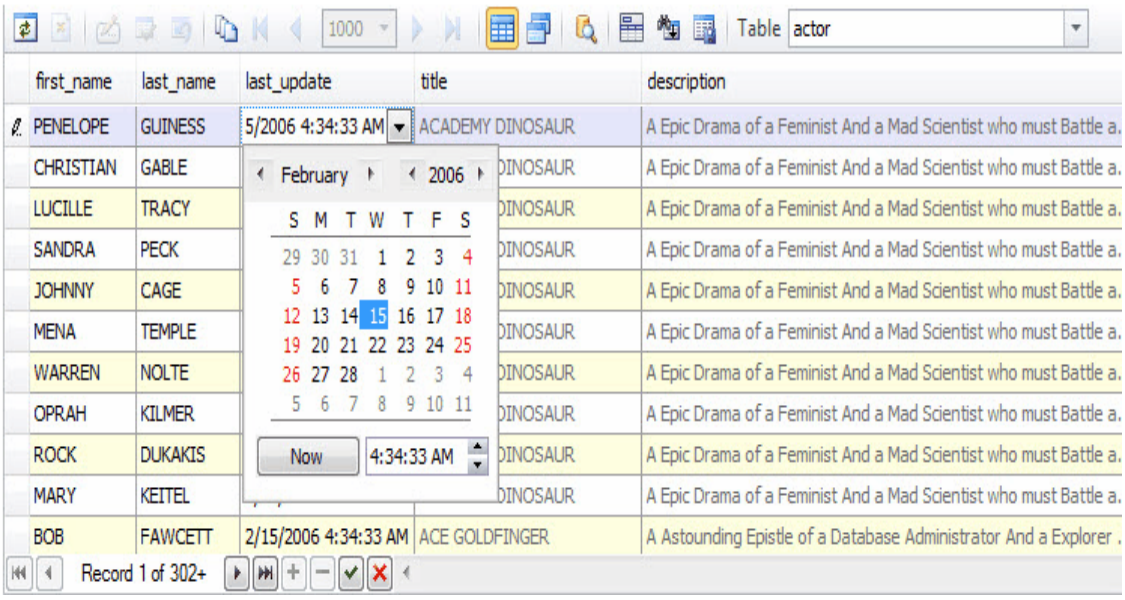
- იხსნება ბადე. ვაწკაპუნებთ ბადის ქვეშ. გაიხსნება მენიუ შემდეგი ბრძანებებით:



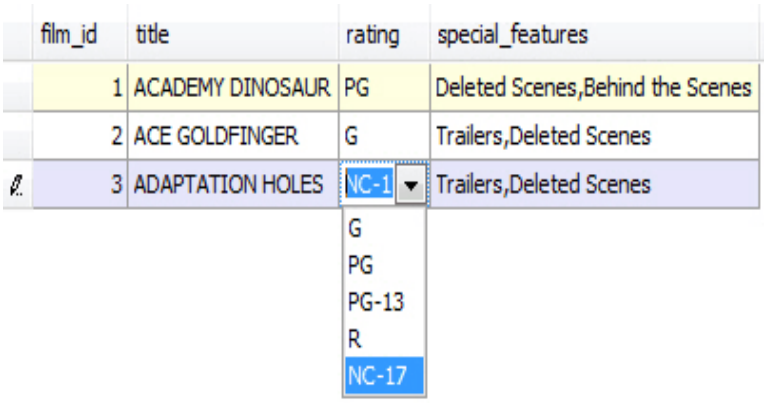
სურ. 7.15

• ახალი ჩანაწერის დამატებისათვის ვირჩევთ ბრძანებას Append ან ვაჭერთ ღილაკს Append ბადის ქვეშ.

• ჩნდება ახალი ცარიელი სტრიქონი, სადაც ვიწყებთ მონაცემთა შეტანას. DATE ან DATETIME ტიპის ველებისათვის იხსნება კალენდარი, რომლის მეშვეობით შესაძლებელია დროითი მონაცემის აწყობა.



სურ. 7.16



სურ. 7.17

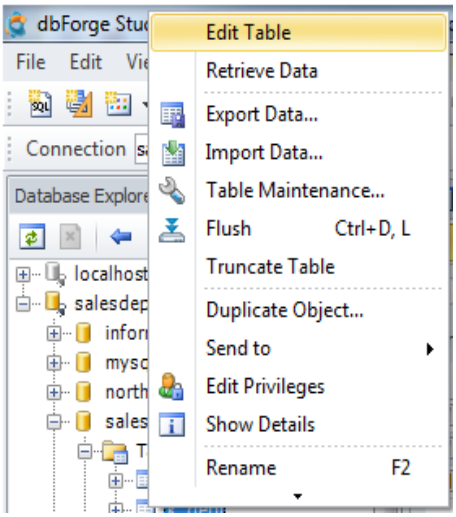
• მონაცემთა რედაქტირება ხდება იმავე ფანჯარაში, ოღონდ რედაქტირების შემდეგ დადასტურებისათვის ვაჭერთ ღილაკს **End Edit** ბადის ქვევით ან ვირჩევთ ოპციას **End Edit** popup მენიუდან.

• ჩანაწერის ბადიდან წაშლისათვის ვირჩევთ ოპციას Delete popup მენიუდან ან ვაწკაპუნებთ ღილაკზე '-' ბადის ქვემოთ ან ვაჭერთ გასაღებურ კლავიშებს CTRL+DEL.

• უჯრედის მნიშვნელობის კოპირებისა (copy) და ჩასმისათვის (paste) უნდა გამოვიყენოთ შესაბამისი ოპციები popup მენიუდან.

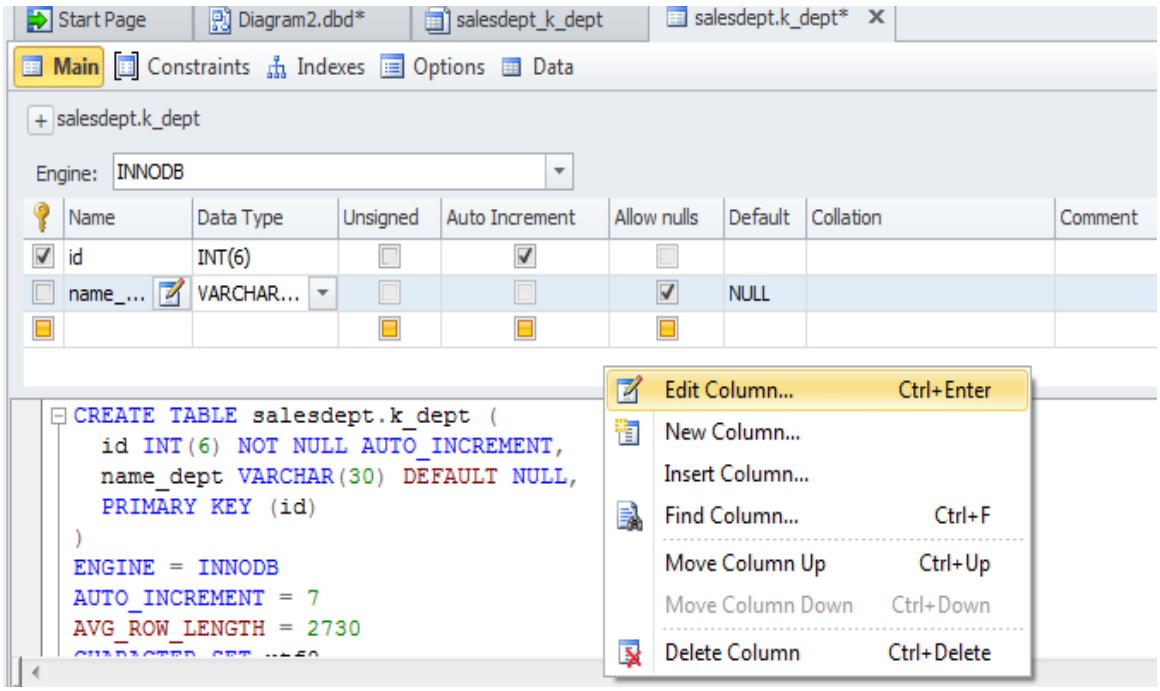
მონაცემთა სტრუქტურის რედაქტირება

- გამოვიყენოთ popup მენიუ ან შესაბამისი ღილაკი ბადის ქვეშ. იხსნება popup მენიუ, სადაც ვირჩევთ Edit Table-ს.



სურ. 7.18

- ბადის ქვევით ვხსნით popup მენიუს, სადაც ვირჩევთ Edit Column -ს.



სურ. 7.19

- ვახდენთ ცხრილის სტრუქტურაში საჭირო ცვლილებებს.
- რედაქტირების შემდეგ დადასტურებისათვის ვაჭერთ ღილაკს **End Edit** ბადის ქვევით ან ვირჩევთ ოპციას **End Edit** popup მენიუდან.

2. DML: მონაცემთა ამორჩევა dbForge Studio -ში

მოთხოვნათა შექმნა და რედაქტირება

მოთხოვნის შექმნისათვის საჭიროა:

1. დამყარდეს სერვერთან მიერთება.

2. **Start** გვერდზე ჯერ ვაწკაპუნებთ  **SQL Development** -ზე და შემდეგ  **SQL Editor** -ზე. სხვაგვარად, ვაწკაპუნებთ **Standard** ინსტრუმენტების პანელის toolbar **New SQL** -ზე. იხსნება ცარიელი SQL დოკუმენტი.

3. ვკრეფთ მოთხოვნას მონაცემთა ბაზისათვის.

4. ვაწკაპუნებთ **Execute** -ზე ან ვაჭერთ **Ctrl+F5** მოთხოვნის შესრულების შედეგების ნახვის მიზნით.

შენიშვნა: მოთხოვნის აკრეფის პროცესში გასაღებური სიტყვები, ცხრილები და მათი სვეტები იხსნებიან ჩამოშლადი სიების სახით.

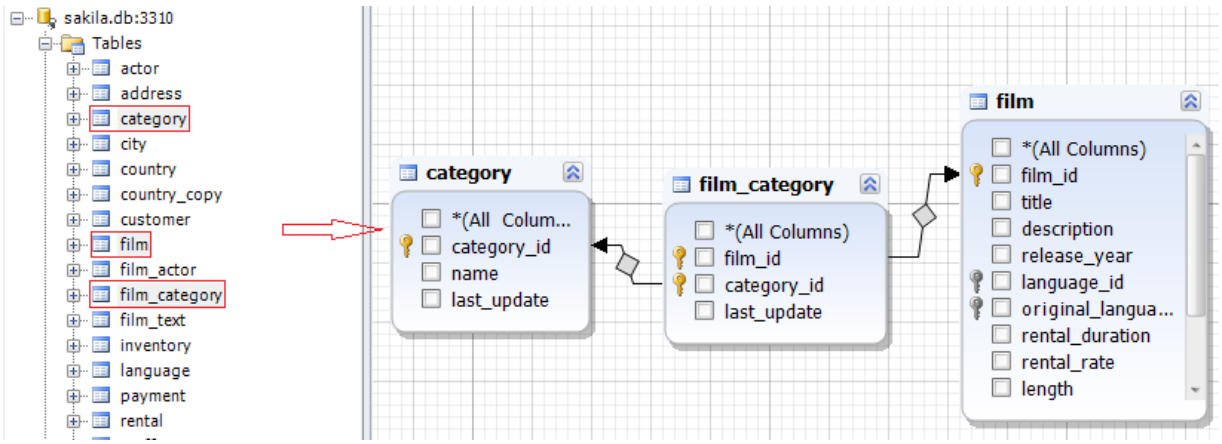
მოთხოვნის შექმნა Query Builder -ში

ამ შემთხვევაში მოთხოვნა იქმნება ვიზუალურად:

1. დამყარდეს სერვერთან მიერთება.

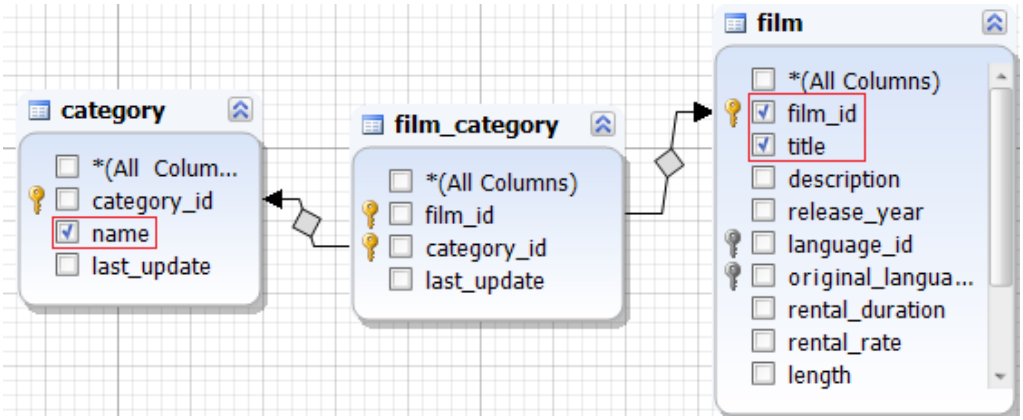
2.**Start** გვერდზე ჯერ ვაწკაპუნებთ  **SQL Development** -ზე და შემდეგ  **Query Builder** ზე. ახლა უკვე შესაძლებელია **Database Explorer** -დან ცხრილების რედაქტორში „გადათრევა“ (drag-and-drop).

განვიხილოთ მაგალითი:



სურ. 7.19

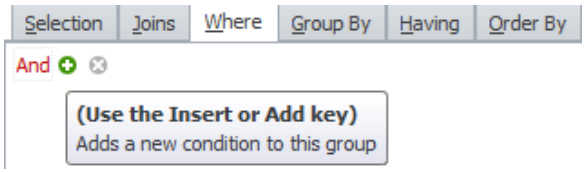
3. ვირჩევთ სვეტებს **film_id** და **title** ცხრილში **film,** ხოლო სვეტს **name** ცხრილში **category**.



სურ. 7.20

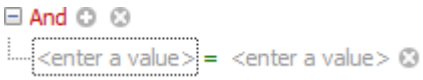
ჩვენი მაგალითისათვის მოთხოვნა მზად არის შესასრულებლად.

4. თუ გვჭირდება **WHERE** პირობის დამატება ვაწკაპუნებთ მწვანე **plus** იკონაზე.



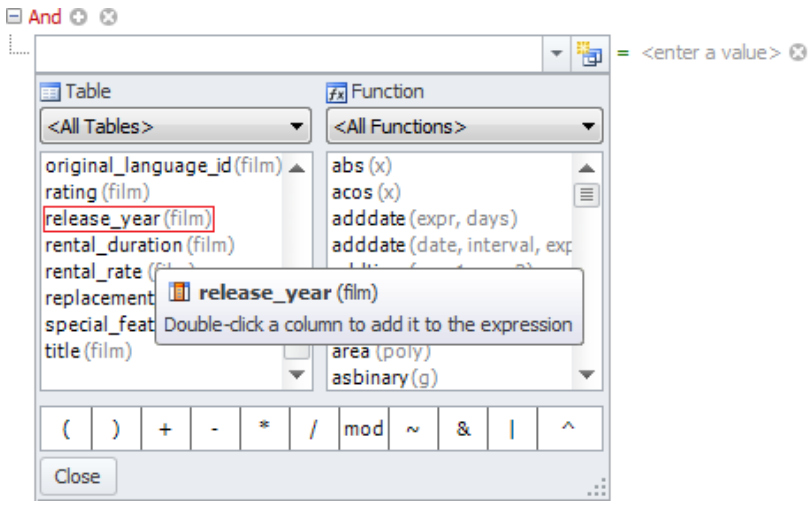
სურ. 7.21

5. ვაწკაპუნებთ **enter value** ზე.



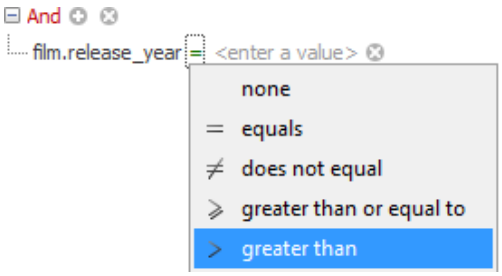
სურ. 7.22

6.სიიდან ვირჩევთ **release_year** -ს.



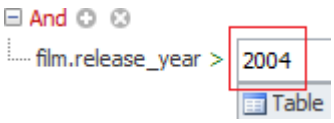
სურ. 7.23

7. ვაწკაპუნებთ ნიშანზე **equals** და ვირჩევთ **greater than** -ს.



სურ. 7.24

8. ვაწკაპუნებთ **enter a value** და ვკრეფთ **2004**.



სურ. 7.25

9. ვაწკაპუნებთ **Execute.** შედეგი აისახება ეკრანზე.

film_id	title	name
19	AMADEUS HOLY	Action
21	AMERICAN CIRCUS	Action
29	ANTITRUST TOMATOES	Action
38	ARK RIDGEMONT	Action
56	BAREFOOT MANCHURIAN	Action
67	BERETS AGENT	Action
97	BRIDE INTRIGUE	Action

სურ. 7.26

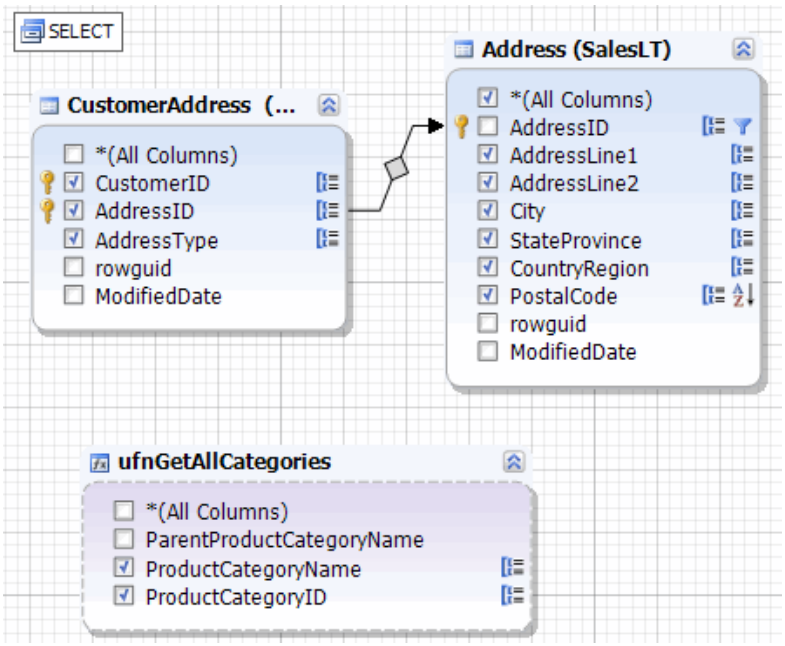
მოთხოვნაში ცხრილის დამატება

ცხრილის და წარმოდგენის (view) მოთხოვნაში დამატების უმარტივესი ხერხია მათი უბრალოდ „გადათრევა“ (drag-and-drop) **Database Explorer** ხიდან მოთხოვნის დიაგრამაში.

ჩვენ შეგვიძლია აგრეთვე საჭირო ცხრილების ამორჩევა **Database Explorer** -ში, ამ ცხრილებზე მარჯვენა ღილაკზე დაწკაპუნებით და შემდეგ ჯერ ვაწკაპუნებთ **Send To** ზე და მერე **Query Builder**-ზე. ცხრილის ფორმები ჩნდება სვეტების სიაში, რომელიც უნდა იქნას მონიშნული ჩართვისათვის **SELECT** სიაში. ვრთავთ "**All Columns**" ოპციას, თუ გვინდა ცხრილის ყველა ველის არჩევა.

ცხრილების დამატების შემდგომ ჩვენ შეგვიძლია ამორჩევის რედაქტორის გამოყენება მოთხოვნის საჭიროებისამებრ დაზუსტებისათვის.

გარდა ცხრილებისა, ჩვენ შეგვიძლია აგრეთვე წარმოდგენების დამატება დიაგრამაზე. ისინი ეკრანზე გამოჩნდებიან სვეტების სიაში, რომლის მონიშვნაც შეგვიძლია SELECT სიაში ჩართვისათვის.



სურ. 7.27

ცხრილის ამოგდება მოთხოვნიდან

მოთხოვნიდან ცხრილის ამოგდებისათვის ვატარებთ შემდეგ მოქმედებებს:

• დიაგრამაზე ცხრილის დასახელებაზე მარჯვენა ღილაკზე ვაწკაპუნებთ და popup მენიუდან ვირჩევთ Remove from Diagram.

• ვდგებით ცხრილის დასახელებაზე, მარცხენა ღილაკზე ვაწკაპუნებთ და ვაჭერთ **DELETE** -ს.

სვეტის ჩართვა მოთხოვნაში

მოთხოვნაში ცხრილის სვეტების ჩართვისათვის ვატარებთ შემდეგ მოქმედებებს:

• დიაგრამაზე ვპოულობთ ცხრილის ფორმას. თუ გვინდა ცხრილიდან ყველა ველის ჩართვა მოთხოვნაში, მაშინ იქნება select *(All Columns) ოპცია.

• გადავდივართ Selection ჩანართში. დიაგრამის ქვეშ რედაქტორში. Column ველის ახალ სტრიქონში ვირჩევთ საჭირო სვეტს.

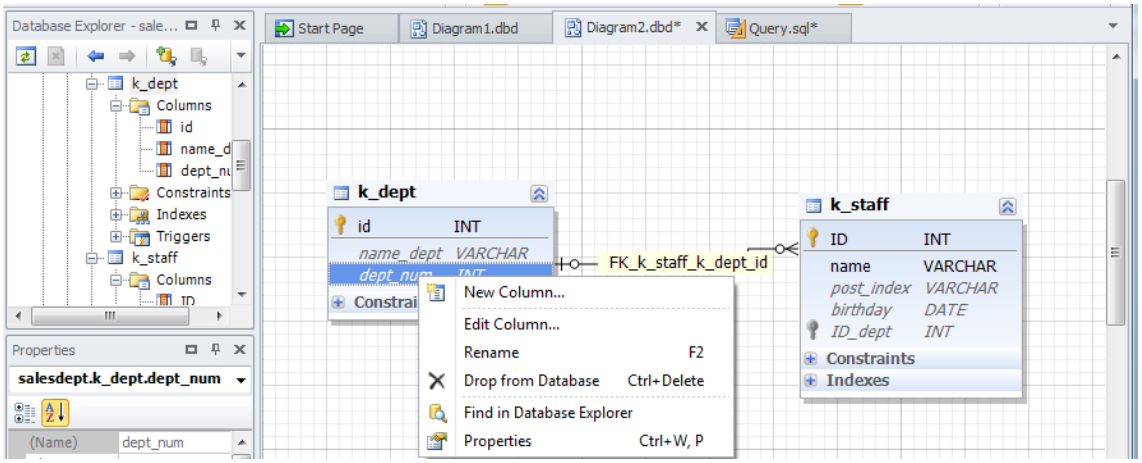
მარტივი **WHERE** ოპერატორის ჩართვისათვის შევდივართ WHERE ჩანართში, ვაწკაპუნებთ And- ზე და შემდეგ „პლუსზე“. გამოდის ფორმა:

<enter a value> ოპერატორი < enter a value >.

ოპერატორი შეიძლება იყოს: >, <, =, <>, >=, <=. მარცხენა ნაწილში ვირჩევთ ცხრილის სვეტს და ვაჭერთ ღილაკს Close, ხოლო მარჯვენა ნაწილში ასევე სვეტს, ფუნქციას ან მნიშვნელობას.

მოთხოვნიდან სვეტის ამოგდებისათვის უნდა შესრულდეს შემდეგი მოქმედებები:

- დიაგრამაზე ვპოულობთ ცხრილის იმ სვეტს, რომლის ამოგდებაც გვინდა.
- ვირჩევთ ბრძანებას Drop from Database.

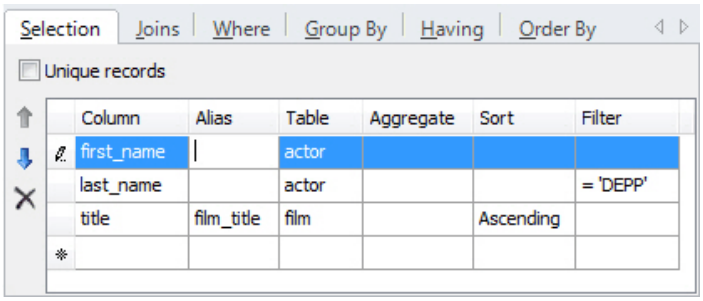


სურ. 7.28

ან კიდევ:

• ცხრილის ფორმაზე მოვნიშნავთ სვეტს.

• ვაჭერთ ღილაკს ✕ Remove.



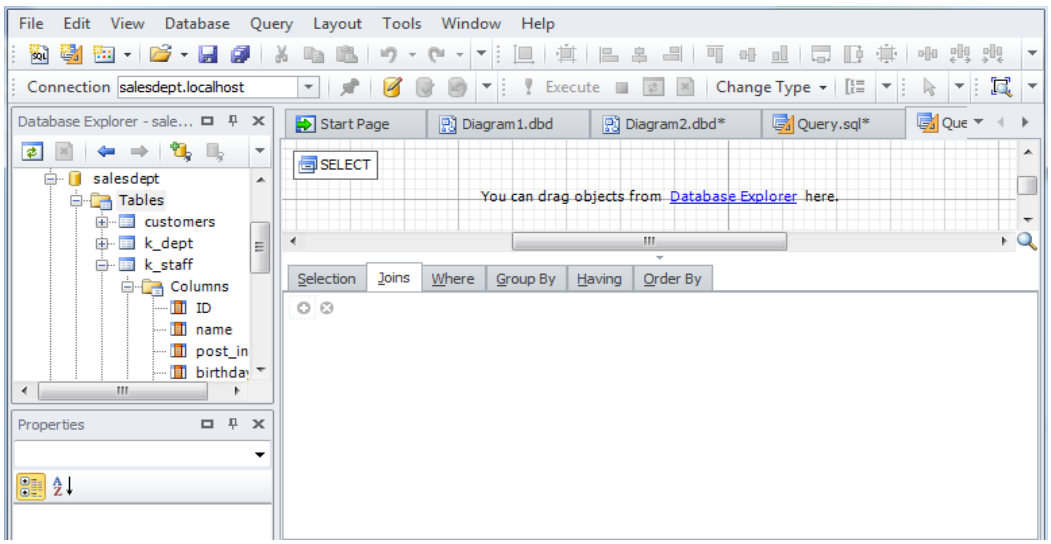
სურ. 7.29

ბრძანება Joins ცხრილების შეერთებისათვის

ცხრილების შეერთებისათვის საჭიროა შემდეგი მოქმედებების შესრულება:

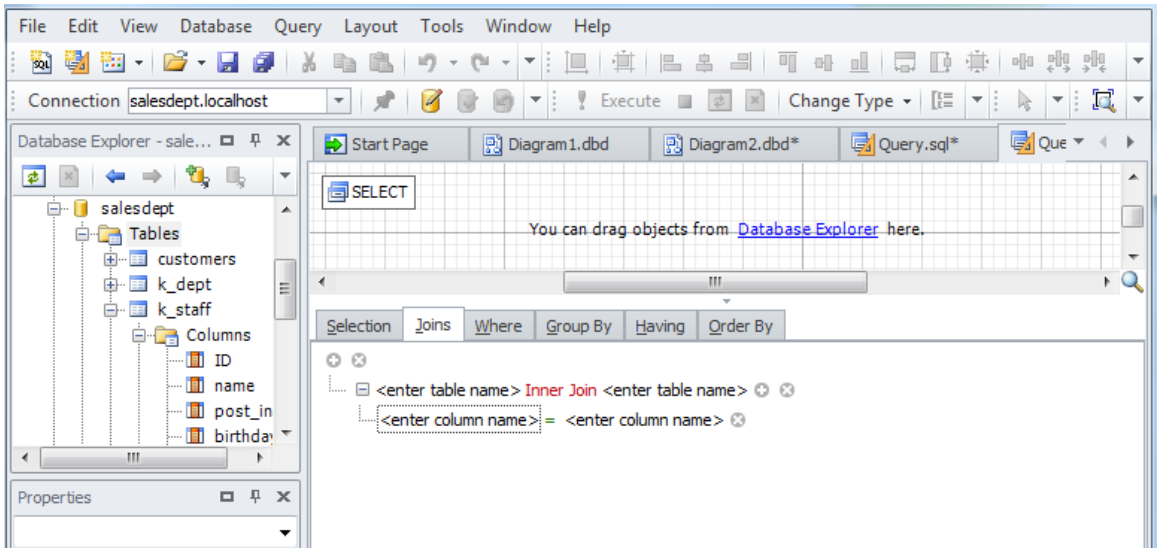
• დიაგრამის გამოყენება: სვეტის Drag-and-drop ერთი ცხრილიდან მეორე ცხრილისაკენ.

• Join ბრძანების გამოყენება: join ჩანართში ვაწკაპუნებთ ღილაკზე ✪ ფესვური ხის სათავეში.



სურ. 7.30

ჩნდება ცარიელი join ცარიელ პირობებთან ერთად.



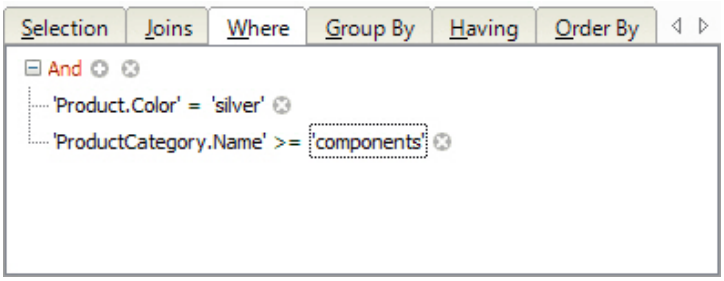
სურ. 7.31

ვაწკაპუნებთ <ცხრილის სახელი>ველი და join-ის სახეობა (Inner Join, Outer Join). მათი არჩევა შეგვიძლია ჩამოშლადი სიიდან.

join -ის ამოგდებისათვის ვაწკაპუნებთ ღილაკზე .

ასევე დიაგრამიდან მარჯვენა ღილაკზე დაწკაპუნებით ვირჩევთ **Remove from Diagram** ან **DELETE** .

WHERE, HAVING და **ORDER BY** პირობების შექმნისათვის ვიყენებთ შესაბამის ჩანართებს. ვაწკაპუნებთ ღილაკზე . შემდგომ ვაყენებთ პარამეტრებს. პირობების ამოგდებისათვის კი ვაწკაპუნებთ ღილაკზე . მაგალითად:



სურ. 7.32

მუშაობა ქვემოთხოვნებთან

ახალი ქვემოთხოვნის შექმნისათვის:

• დიაგრამაზე მარჯვენათი ვაწკაპუნებთ და popup მენიუდან ვირჩევთ **Create Subquery.**

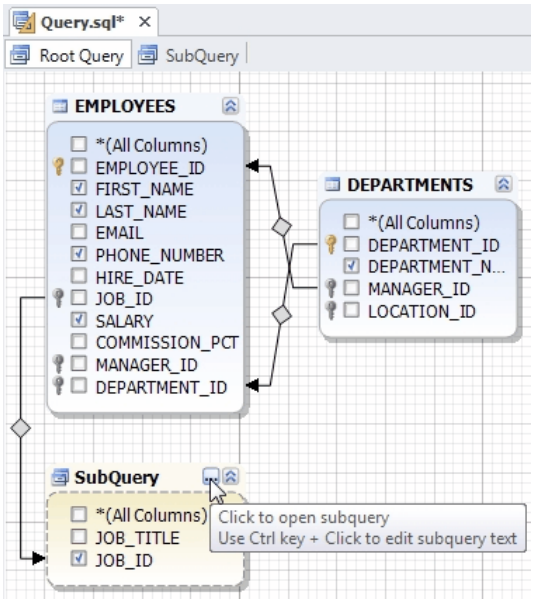
• **Query** ინსტრუმენტების პანელზე ვაწკაპუნებთ ღილაკზე**Create Subquery.**

• **Query** მენიუდან ვირჩევთ **Create Subquery** -ს.

ჩნდება ჩანართი **SubQuery**, სადაც შეგვიძლია ქვემოთხოვნის შექმნა -

ცხრილის დამატება, საჭირო სვეტების შერჩევა, პირობების განსაზღვრა და ა.შ.

მაგალითად:



სურ. 7.33

ქვემოთხოვნის რედაქტირებისათვის:

მოთხოვნის სათაურის სტრიქონში ვაწკაპუნებთ ღილაკზე ... ან იქვე მარჯვენა დაწკაპუნებით popup მენიუდან ვირჩევთ **Edit Subquery** -ს.

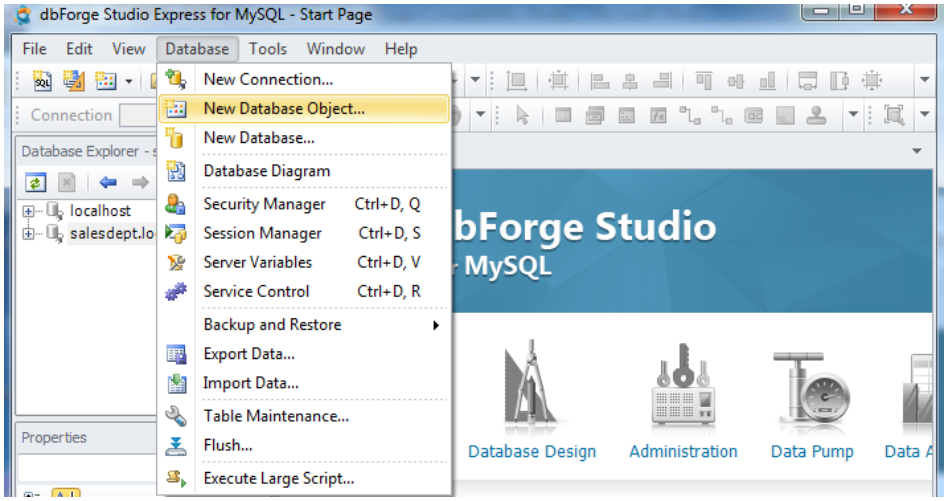
და ბოლოს, **Visual Query Builder** არის მძლავრი ინსტრუმენტი, რომელიც რთული მოთხოვნების სწრაფად და კოდის გარეშე აგების საშუალებას იძლევა.

ხდომილობებთან (Event) მუშაობა

dbForge Studio დავალებათა შექმნის, მოდიფიცირებისა და ამოგდების საშუალებას იძლევა.

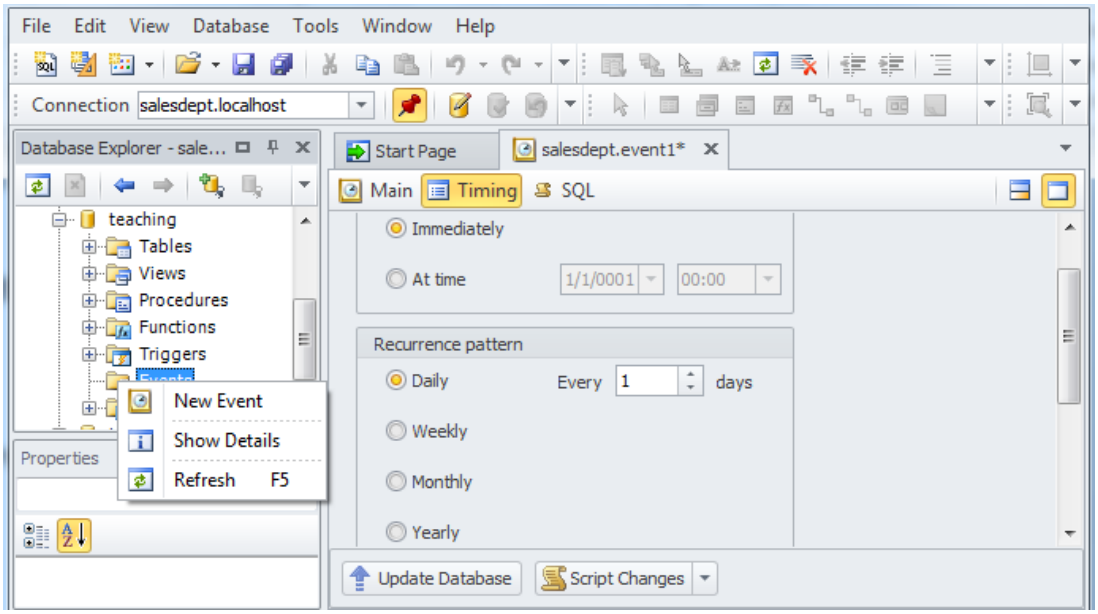
დავალების შექმნა

1.მენიუში ვირჩევთ **New Database Object** ღილაკს. ვირჩევთ "Event".



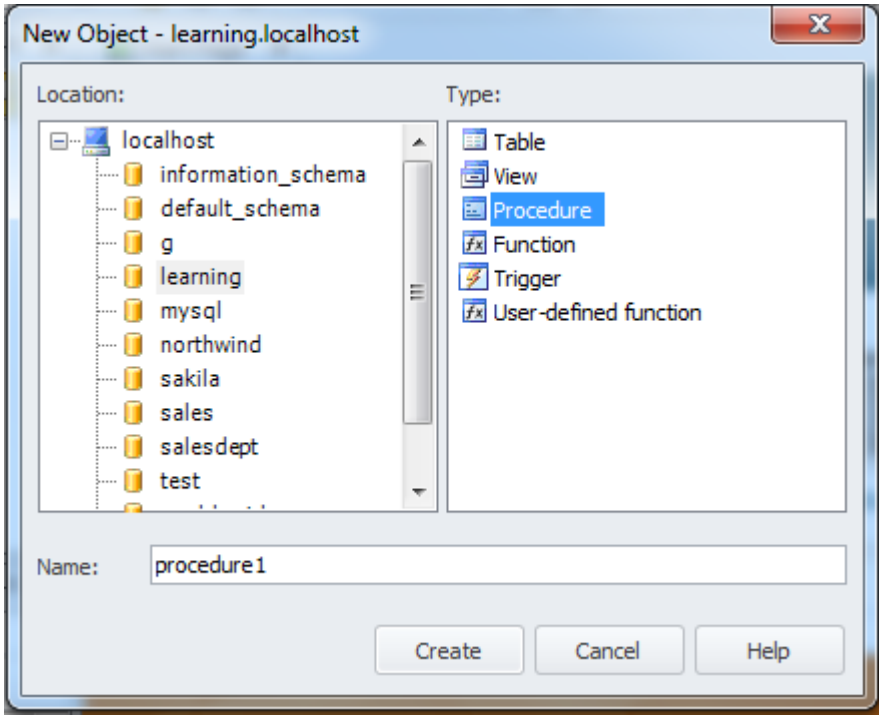
სურ. 7.34

ან **Database Explorer** -ში მარჯვენა დაწკაპუნებით **Events**-ზე და ვირჩევთ**New Event**.



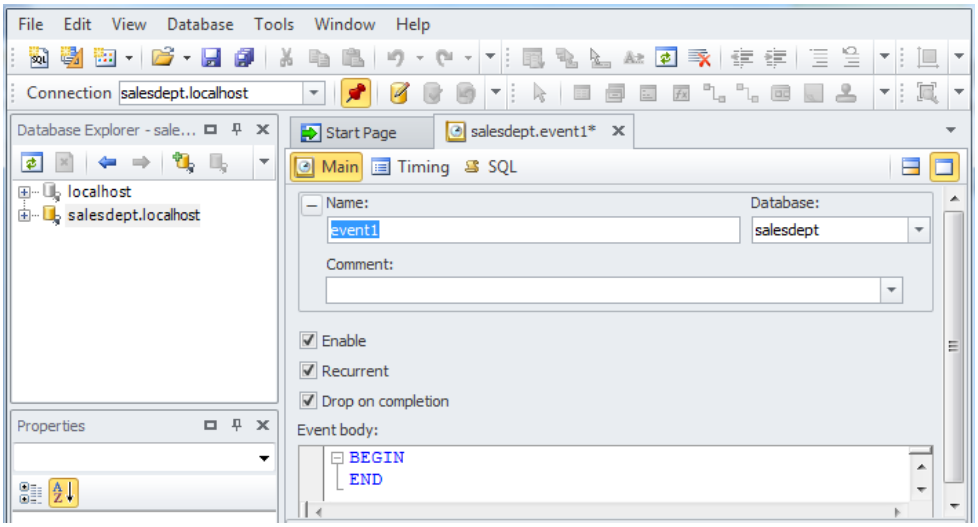
სურ. 7.35

2. შეგვაქვს დავალების სახელი.
3. ვაწკაპუნებთ ღილაკზე **Create**. მივუთითებთ სახელს.



სურ. 7.36

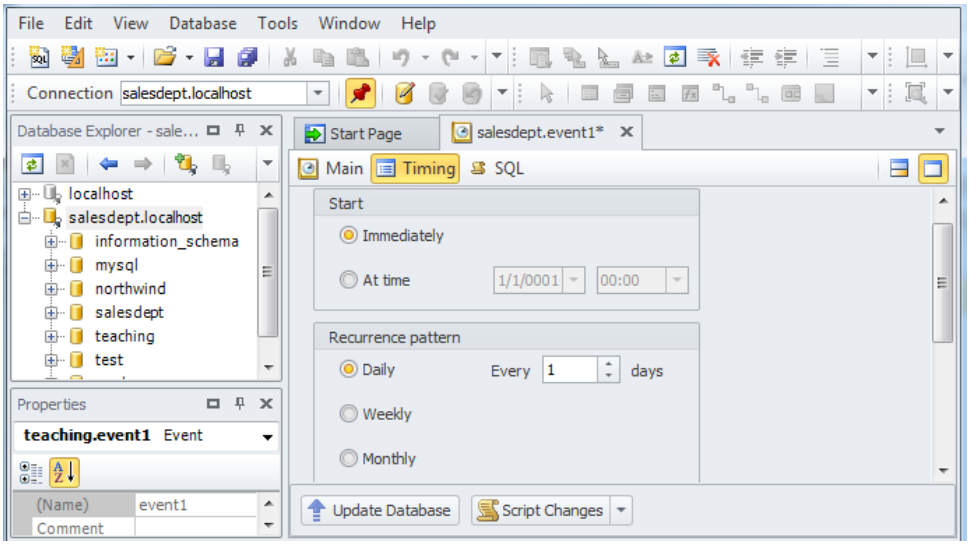
4. ვირჩევთ **Enable**, **Recurrent** და **Drop on Completion** ოპციებს.



სურ. 7.37

5.ჩვენ შეგვიძლია შევცვალოთ სახელი ან მონაცემთა ბაზა (მფლობელი), რომელშიც უნდა შევქმნათ ხდომილობა.

6. გადავდივართ **Timing** -ზე და ვაყენებთ ხდომილობის დამგეგმავის პარამეტრებს.



სურ. 7.38

7. ვინახავთ დოკუმენტს **Save**. თუ რაიმე შეცდომა იქნა დაშვებული, მაშინ იგი უნდა შესწორდეს.

ხდომილობის რედაქტირება

ხდომილობის რედაქტირებისათვის **Edit Event** აირჩევა Database Explorer კონტექსტური მენიუდან. აღსანიშნავია, რომ ხდომილობა არ შეინახება, მისი დაწყების დღე უკვე ჩავლილია.

ხდომილობის ამოგდება

ხდომილობის ამოგდებისათვის Database Explorer კონტექსტური მენიუდან აირჩევა **Delete**.

ლაბორატორიული სამუშაო 8
პროგრამირება MySQL-ში

სამუშაოს მიზანი:

1. MySQL-პროგრამირება კონსოლში. შენახული პროცედურები
2. კონსტრუქციები
3. ტრიგერები
4. კურსორები
5. პროცედურებისა და ფუნქციების შექმნა MySQL Query Browser გამოიყენებით
6. პროცედურებისა და ფუნქციების შექმნა dbForge Studio for MySQL გამოიყენებით
7. ტრიგერების შექმნა dbForge Studio for MySQL გამოიყენებით

1.MySQL-პროგრამირება კონსოლში. შენახული პროცედურები

სერვერზე ხშირად გვიწევს ერთი და იგივე ოპერაციების შესრულება, რისთვისაც უმჯობესია მათი გაფორმება ე.წ. შენახული პროცედურის (stored procedure) სახით, რომელიც წარმოადგენს მონაცემთა ბაზის დამოუკიდებელ ობიექტს. შენახული პროცედურები, რომელთა დახმარებით იქმნება ქვეპროგრამები, ისურება და მუშაობენ უშუალოდ სერვერზე. ნაცვლად პროგრამირების სტანდარტული ენების (C, Visual Basic და სხვ.) გამოყენებისა, MySQ-ის მონაცემთა ბაზებში ოპერაციათა შესასრულებლად შეიძლება შენახული პროცედურების გამოყენება. მათი გამოძახება შესაძლებელია კლიენტის პროგრამის, სხვა შენახული პროცედურის ან ტრიგერის მიერ.

პროცედურა არის მოქმედებები, რომელიც SQL ბრძანებათა ერთობლიობის შესაბამისად სრულდება და შედეგად აბრუნებს ცხრილს. პროცედურის გარდა არსებობს შენახული ფუნქციები.

ორივე ოპერაცია იქმნება ბრძანებების *CREATE FUNCTION* ან *CREATE PROCEDURE* გამოყენებით. ქვემოთ მოყვანილია ამ ბრძანებების სინტაქსისი:

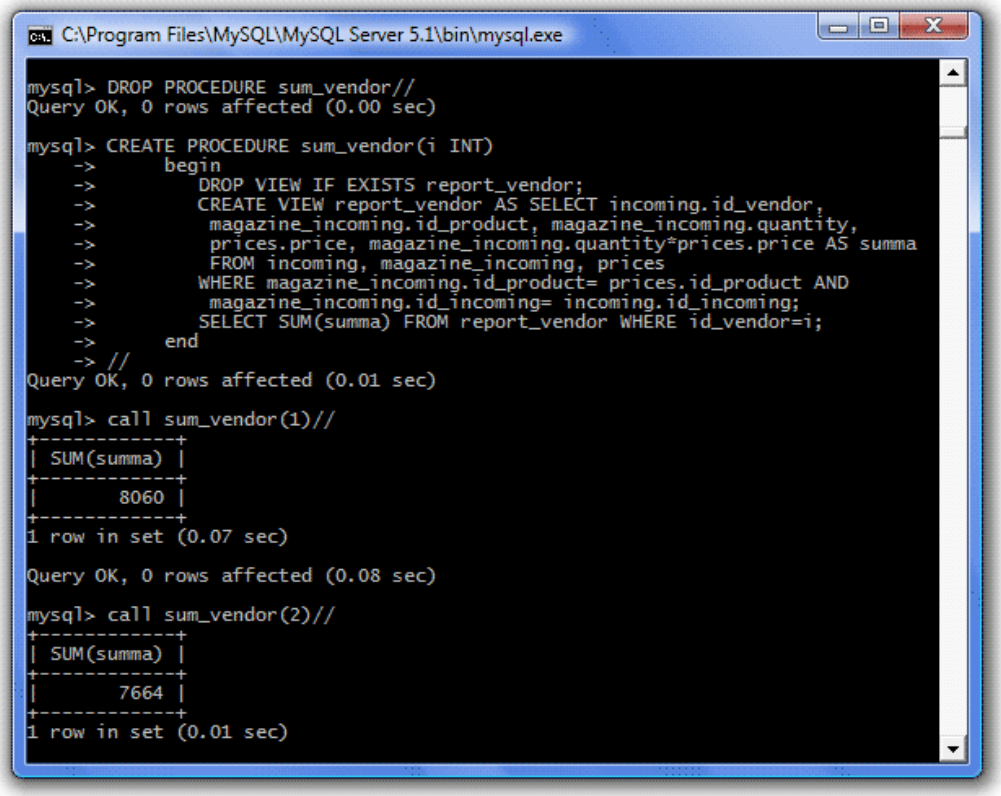
პროცედურების შექმნისათვის გამოიყენება ბრძანება *CREATE PROCEDURE* სინტაქსისით:

CREATE PROCEDURE *name* ([*parameterlist*])
[*options*] *sqlcode*

ფუნქციის შექმნისათვის გამოიყენება ბრძანება *CREATE FUNCTION* სინტაქსისით:

CREATE FUNCTION name
([*parameterlist*]) *RETURNS datatype*
[*options*] *sqlcode*

ქვემოთ მოყვანილია პროცედურის კონსოლში შექმნის მაგალითი.



სურ.8.1

ფუნქციებსაც და პროცედურებსაც შეიძლება ჰქონდეთ ერთიდაიგივე სახელები, რადგან მათ გააჩნიათ გამოძახების განსხვავებული მექანიზმები. ფუნქცია მთავრდება ბრძანებით RETURN და აბრუნებს სკალარულ მნიშვნელობას, რომლის ტიპიც მითითებულია კოდში.

```
DELIMITER ;
DROP FUNCTION IF EXISTS reduce;
DELIMITER $$
CREATE FUNCTION reduce(str VARCHAR(255), len INT)
  RETURNS VARCHAR(255)
BEGIN
  IF ISNULL(str) THEN RETURN NULL; END IF;
  IF len = 0 OR CHAR_LENGTH( str ) = 0 OR CHAR_LENGTH( str ) <= len THEN RETURN
str;
  END IF;
    IF len < 15 THEN
    RETURN LEFT( str, len );
  ELSE
    RETURN CONCAT(  LEFT( str, len - 10 ), ' ___ ', RIGHT( str, 5 ) );
  END IF;
END$$
DELIMITER ;
```

$$ გამყოფის ცვლილება წერტილ-მძიმისათვის ფუნქციის შიგნით მოქმედების

საშუალებას იძლევა: delimiter $$. ორი პარამეტრი გამოიყენება *reduce( )* ფუნქციის შიგნით: სტრინგი *str* და მთელრიცხვა *len*, რომელიც წარმოადგენს დაბრუნებული მნიშვნელობის მაქსიმალურ სიგრძეს. უკან დაბრუნება შემოსაზღვრულია წერტილ-მძიმით: delimiter ;

პროცედურების გამოძახება

პროცედურების გამოძახება ბრძანებით *CALL*.

CALL *sp_name*([*parameter*[,...]])
CALL *sp_name*[()]

ფუნქციის წაშლა

ფუნქციის (ასევე პროცედურის) წაშლისათვის გამოიყენება ბრძანება DROP:

DROP FUNCTION [IF EXISTS] name
DROP PROCEDURE [IF EXISTS] name

ფუნქციის ცვლილება

ALTER FUNCTION/PROCEDURE name
[*NAME newname*]
[*SQL SECURITY DEFINER/INVOKER*]
[*COMMENT* '*newcomment*']

არსებული ფუნქციების სიის ნახვა

ამისათვის გამოიყენება ორი ბრძანება *SHOW PROCEDURE STATUS* და *SHOW FUNCTION STATUS* :

SHOW FUNCTION STATUS;
SHOW PROCEDURE STATUS;

SHOW FUNCTION STATUS LIKE 'repeat%';
SHOW PROCEDURE STATUS LIKE 'film%';

ფუნქციის და პროცედურის შესახებ ინფორმაციის გამოტანისათვის გამოიყენება ოპერატორი SHOW STATUS, ხოლო პუნქტი LIKE ასრულებს ფილტრის როლს. უფრო მეტი ინფორმაციით არის აღჭურვილი ცხრილი *information_schema.routines,* რომელიც შეიცავს ყველა მონაცემს *mysql.proc* ცხრილის შესახებ, თუმცა ის შეიძლება შეიცავდეს სვეტების სხვადასხვა სახელებს:

```
SELECT routine_name, routine_type, created
  FROM information_schema.routines
 WHERE routine_schema='sakila';
```

SP -ის კოდის განსაზღვრა

თუ ცნობილია SP -ის სახელწოდება , მაშინ შესაძლებელია მისი კოდის წვდომა *SHOW CREATE FUNCTION/PROCEDURE name* ბრძანების შესრულებით:

ბრძანებები BEGIN-END

ყოველი პროცედურა ან ფუნქცია მოქცეულია *BEGIN-END* ბრძანებების ბლოკის შიგნით შემდეგი თანამიმდევრობით:

BEGIN
DECLARE variables;
DECLARE cursors;
DECLARE conditions;
DECLARE handler;
other SQL ბრძანებები;
END;

BEGIN- ის წინ მოცემულია ბლოკის სახელწოდება, რომელიც მოხსენიებული იქნება აგრეთვე *END*- ის შემდეგაც. ბლოკიდან გასვლა შესაძლებელია, ნაადრევი გასვლა *LEAVE* - ით:

blockname: *BEGIN*
ბრძანებები;
IF condition *THEN LEAVE* blockname; *END IF*;
further ბრძანებები;
END blockname;

წერტილ-მძიმე გამოყოფს ცალკეულ ბრძანებას SP -ის შიგნით.

ცვლადები

განირჩევა ცვლადების ორი ტიპი MySQL პროგრამირებაში:

- *Ordinary SQL variables;*
- *Local variables and parameters.*

ცვლადები არიან ლოკალური *BEGIN-END* ჯგუფის შიგნით, რომელშიც ისინი იქნენ დეკლარირებული.

DECLARE varname1, varname2, ... datatype [*DEFAULT* value];

ლოკალური ცვლადები შეიცავენ *NULL* უსიტყვოდ.

```
DELIMITER ;
DROP PROCEDURE IF EXISTS `3blocks`;
DELIMITER $$
CREATE PROCEDURE `3blocks`(n INT)
BEGIN
 DECLARE newn INT DEFAULT n;
 BEGIN
  DECLARE newn INT DEFAULT n * 2;
  IF TRUE THEN
    BEGIN
      DECLARE newn INT DEFAULT n * 3;
      SELECT n 'Orig', 3 'Run', newn 'New Factor';
    END;
  END IF;
  SELECT n 'Orig', 2 'Run', newn 'New Factor';
 END;
 SELECT n 'Orig', 1 'Run', newn 'New Factor';
```

```
END$$
DELIMITER ;
```

პროცედურის გამოძახება მოხდება ბრძანებით:

```
CALL `3blocks`(10);
```

აქ გვაქვს სამი ცვლადი სახელწოდებით *newn*, რომლებიც არიან დეკლარირებული კოდის სამ დონეზე და არიან ერთმანეთისაგან დამოუკიდებელი. პროცედურა აბრუნებს 1, 2, 3 -ზე გამრავლებულ სამ შედეგს.

ცვლადების მნიშვნელობის განსაზღვრისათვის გამოიყენება *SET* ან *SELECT INTO*. ფუნქციებში მხოლოდ *SET* შეიძლება გამოყენებულ იქნას მანამ, სანამ *SELECT* და სხვა SQL ბრძანებები არ არის დაშვებული.

```
DELIMITER ;
DROP PROCEDURE IF EXISTS `assign_vars`;
DELIMITER $$
CREATE PROCEDURE `assign_vars`(n INT)
BEGIN
   DECLARE varlength SMALLINT;
   DECLARE pgcount SMALLINT;
   DECLARE var1 SMALLINT;
   DECLARE var2 SMALLINT;
   DECLARE var3 SMALLINT;
   SET var1 = n * 5, var2 = n * 10, var3 =  n * 15;
   SELECT @var := CONCAT_WS( ',', var1, var2, var3 );
   SELECT CHAR_LENGTH(@var) INTO varlength;
   SELECT COUNT(film_id) FROM film WHERE rating = 'PG' INTO pgcount;
   SELECT @var, varlength 'Length of @var', pgcount 'PG Rated Film Count';
   SELECT title, rating FROM film WHERE film_id = 101 INTO @vtitle, @vrating;
   SELECT 101 'Film ID', @vtitle 'Title', @vrating 'Rated As';
END$$
DELIMITER ;

CALL `assign_vars`(10);
```

2.კონსტრუქციები

შენახული პროცედურების შესრულების მართვისათვის გამოიყენება Transact - SQL-ის შემდეგი კონსტრუქციები:

IF-THEN-ELSE Branching

IF condition *THEN*
ბრძანებები;
[*ELSE IF* condition *THEN*
ბრძანებები;]

[*ELSE*
ბრძანებები;]
END IF;

მაგალითად:

```
DELIMITER ;
DROP FUNCTION IF EXISTS enclose;
DELIMITER $$

CREATE FUNCTION enclose(str VARCHAR(255), leftstr VARCHAR(3), rightstr
VARCHAR(3) )
   RETURNS VARCHAR(255)
BEGIN

   DECLARE returnstr VARCHAR(255) DEFAULT '';

   IF ISNULL(str) THEN RETURN NULL; END IF;
   IF leftstr IS NULL OR rightstr IS NULL OR CHAR_LENGTH( str ) = 0 THEN
     RETURN str;
   END IF;

   IF LEFT( str, CHAR_LENGTH( leftstr ) ) != leftstr THEN
      SET returnstr = concat( leftstr, str );
   END IF;

   IF RIGHT( str, CHAR_LENGTH( rightstr ) ) != rightstr THEN
      SET returnstr = concat( str, rightstr );
   END IF;

   IF returnstr = '' THEN SET returnstr = str;
   END IF;
      RETURN returnstr;
END$$

DELIMITER ;
SELECT enclose( 'abc', '[', ']' );
SELECT enclose( '(abc', '(', ')' );
SELECT enclose( 'abc)', '(', ')' );
SELECT enclose( '(abc)', '(', ')' );
SELECT enclose( '[abc)', '(', ')' );
```

განშტოება CASE -ის გამოყენებით

CASE არის *IF* მსგავსი კონსტრუქცია, როდესაც არჩეული ვარიანტი მხოლოდ ერთი გამოსახულების მნიშვნელობაზე იქნება დამოკიდებული.

CASE expression

```
WHEN value1 THEN
 ბრძანებები;
[WHEN value2 THEN
 ბრძანებები;]
[ELSE
 ბრძანებები;]
END CASE;
```

REPEAT-UNTIL ციკლი

კოდი *REPEAT* და *UNTIL* -ს შორის სრულდება სანამ პირობა, რომელიც მოწმდება ციკლის ბოლოს, იქნება ჭეშმარიტი.

[loopname:] *REPEAT*
ბრძანებები;
UNTIL condition
END *REPEAT* [loopname];

მაგალითი :

აქ repeat_plus ფუნქცია აბრუნებს (data type *VARCHAR(255)*) ტიპის სტრიქონს, რომელიც *n* -ჯერ შეიცავს + სიმბოლოს .

```
DELIMITER ;
DROP FUNCTION IF EXISTS repeat_plus;
DELIMITER $$
CREATE FUNCTION repeat_plus(n INT) RETURNS VARCHAR(255)
BEGIN
  DECLARE i INT DEFAULT 0;
  DECLARE returnstr VARCHAR(255) DEFAULT '';
  DECLARE c VARCHAR(1) DEFAULT '+';

  char_add_loop: REPEAT
    SET i = i+1;
    SET returnstr = CONCAT( returnstr, c );
  UNTIL i >= n END REPEAT;
    RETURN returnstr;
END$$
DELIMITER ;
SELECT repeat_plus(5);
```

WHILE

ინსტრუქციები *DO* და *END WHILE* -ს შორის სრულდებასანამ პირობა იქნება დაკმაყოფილებული. სინტაქსისს ექნება შემდეგი სახე:

[loopname:] *WHILE* condition *DO*
ბრძანებები;
END WHILE [loopname];

LOOP

ინსტრუქციები *LOOP, END LOOP* და *LEAVE loopname*. მისი სინტაქსისია:

loopname: *LOOP*
ბრძანებები;
END LOOP loopname;

მაგალითი *LOOP*-ზე:

```
DELIMITER ;
DROP FUNCTION IF EXISTS cut_number;
DELIMITER $$

CREATE FUNCTION cut_number(startnum INT, cutby INT, times INT)
RETURNS INT
BEGIN

  DECLARE i INT DEFAULT 0;
  DECLARE s TEXT DEFAULT '';
  DECLARE cut INT DEFAULT 0;

  SET cut = startnum;
    cutloop: LOOP
    IF i >= times THEN LEAVE cutloop; END IF;
    SET cut = cut - cutby;
    SET i = i + 1;
  END LOOP cutloop;
  RETURN cut;
END$$
DELIMITER ;
SELECT cut_number( 80, 16, 4 );
SELECT cut_number( 100, 20, 5 );
SELECT cut_number( 120, 4, 6 );
```

LEAVE და ITERATE

ბრძანება *LEAVE loopname* ახდენს ბლოკის ციკლიდან გასვლას. *LEAVE* -ბრძანება შეიძლება გამოყენებულ იქნას აგრეთვე *BEGIN-END* ბლოკის რიგგარეშე გასვლისათვის.

ITERATE loopname ბრძანებას აქვს შედეგი, როდესაც ციკლის ტანის ნაწილი იგნორირებულია და ციკლი ხელახლა სრულდება. *ITERATE* მუშაობს მხოლოდ ციკლებთან და არ შეიძლება გამოყენებულ იქნას ჩვეულებრივ *BEGIN-END* ბლოკში.

მაგალითი *WHILE* -ზე.

```
DELIMITER ;
DROP FUNCTION IF EXISTS cut_number_except3;
DELIMITER $$
```

```
CREATE FUNCTION cut_number_except3(startnum INT, cutby INT, times INT)
RETURNS INT NO SQL
BEGIN

  DECLARE i INT DEFAULT 0;
  DECLARE s TEXT DEFAULT '';
  DECLARE cut INT DEFAULT 0;

  SET cut = startnum;
    cutloop: WHILE ( i < times ) DO
    SET i = i + 1;
    IF i % 3 = 0 THEN ITERATE cutloop; END IF;
    SET cut = cut - cutby;
  END WHILE;
  RETURN cut;
END$$
DELIMITER ;

SELECT cut_number_except3( 60, 10, 6 );
SELECT cut_number_except3( 80, 16, 4 );
SELECT cut_number_except3( 100, 20, 5 );
SELECT cut_number_except3( 120, 4, 6 );
```

3. ტრიგერები

ტრიგერი (trigger) არის სპეციალური ტიპის შენახული პროცედურა, რომელიც სრულდება კონკრეტულ ცხრილზე კონკრეტული მოქმედებების განხორციელებისათვის. ტრიგერი სამი ძირითადი ნაწილისაგან შედგება: სახელი, ქმედება და შესრულება. ტრიგერები ავტომატურად ასრულებს SQL ბრძანებეს ან შენახულ პროცედურებს *INSERT, UPDATE,* ან *DELETE* ბრძანებების წინ ან შემდეგ. ტრიგერების შექმნისათვის გამოიყენება ბრძანება *CREATE TRIGGER*. მხოლოდ MySQL მომხმარებლებს აქვთ ამ ბრძანების გამოყენების *Super* პრივილეგია. მისი სინტაქსისი შემდეგია:

CREATE TRIGGER name BEFORE|AFTER INSERT|UPDATE|DELETE
ON tablename FOR EACH ROW sql-code

ჩვენ შეგვიძლია განვსაზღვროთ ექვსზე მეტი ტრიგერი თითოეული ცხრილისათვის, რომლის კოდი შესრულდება თითოეული *INSERT, UPDATE,* ან *DELETE* ბრძანებების წინ ან შემდეგ.

ტრიგერის კოდით შესაძლებელია მიმდინარე ჩანაწერთან წვდომა:

- *OLD.columnname* returns the content of an existing record before it is changed or deleted (*UPDATE, DELETE*).
- *NEW.columnname* returns the content of a new or altered record (*INSERT, UPDATE*).

• You may change *NEW.columnname* in *BEFORE INSERT* triggers and *BEFORE UPDATE* triggers.

ტრიგერის წაშლა

ტრიგერის წასაშლელად გამოიყენება DROP TRIGGER ბრძანება. მისი სინტაქსია: DROP TRIGGER [სქემის_სახელი .] ტრიგერის_სახელი [,...n]. როგორც სინტაქსიდან ჩანს შესაძლებელია რამდენიმე ტრიგერის ერთდროულად წაშლა.

მაგალითი:

```
CREATE TABLE trig_test (idcol SERIAL, rate FLOAT, status
ENUM('low','high','orig')
DEFAULT 'orig' );
DELIMITER $$

DROP TRIGGER IF EXISTS trig_test.trig_test_before_insert;
DROP TRIGGER IF EXISTS trig_test.trig_test_before_update;

CREATE TRIGGER trig_test_before_insert
BEFORE INSERT ON trig_test FOR EACH ROW
BEGIN
  IF NEW.rate < 0.0 THEN
    SET NEW.rate = 0.0;
    SET NEW.status = 'low';
  ELSEIF NEW.rate > 5.0 THEN
    SET NEW.rate = 5.0;
    SET NEW.status = 'high';
  END IF;
END$$

CREATE TRIGGER trig_test_before_update
  BEFORE UPDATE ON trig_test FOR EACH ROW
BEGIN
  IF NEW.rate < 0.0 THEN
    SET NEW.rate = 0.0;
    SET NEW.status = 'low';
  ELSEIF NEW.rate > 5.0 THEN
    SET NEW.rate = 5.0;
    SET NEW.status = 'high';
  END IF;
END$$
DELIMITER ;

INSERT INTO trig_test (rate) VALUES (-1), (0.3), (5.5), (3.5), (-0.2), (4.2);
SELECT * FROM trig_test;
```

```
UPDATE trig_test SET rate = 1.7 WHERE idcol = 2;
SELECT * FROM trig_test;
```

4. კურსორები

კურსორი (Cursors – Current Set of Records) კლიენტის პროგრამას საშუალებას აძლევს იმუშაოს, ნაცვლად ასობით ან ათასობით სტრიქონისა, ერთ სტრიქონთან ან სტრიქონების მცირე ბლოკთან. კურსორთან მუშაობის დროს შეგვიძლია ხუთი ძირითადი ოპერაცია შევასრულოთ:

- კურსორის შექმნა. კურსორი უნდა შევქმნათ მის გამოყენებამდე.
- კურსორის გახსნა. შექმნილი კურსორი მონაცემებს არ შეიცავს. გახსნის ოპერაცია კურსორს მონაცემებით ავსებს.
- სტრიქონების ამორჩევა კურსორიდან და მათი შეცვლა კურსორის საშუალებით. სტრიქონებით კურსორის შევსების შემდეგ შეგვიძლია მათთან მუშაობა. კურსორის ტიპზე დამოკიდებულებით შეგვეძლება სტრიქონების ან მხოლოდ წაკითხვა ან წაკითხვა და შეცვლა.
- კურსორის დახურვა. კურსორთან მუშაობის დამთავრების შემდეგ ის უნდა დავხუროთ. ამ დროს სერვერი ათავისუფლებს სივრცეს tempdb მონაცემთა ბაზაში, რომელიც კურსორს გამოეყო შექმნის დროს.
- კურსორის გათავისუფლება. ამ დროს კურსორი იშლება.

კურსორის შექმნა. კურსორის შესაქმნელად გამოიყენება DECLARE CURSOR ბრძანება.

კურსორის გახსნა. კურსორის გასახსნელად და მონაცემებით შესავსებად გამოიყენება OPEN ბრძანება.

მონაცემების წაკითხვა. კურსორიდან მონაცემების წასაკითხად გამოიყენება FETCH ბრძანება. მისი სინტაქსია:

```
FETCH [ [ NEXT | PRIOR | FIRST | LAST
| ABSOLUTE { n | @nvar } | RELATIVE { n | @nvar } ]
FROM
]
{ { [ GLOBAL ] კურსორის_სახელი } | @cursor_variable_name }
[ INTO @variable_name [,...n] ]
```

განვიხილოთ არგუმენტების დანიშნულება.

- FIRST. მისი მითითების შემთხვევაში გაიცემა კურსორის შედეგობრივი ნაკრების პირველი სტრიქონი, რომელიც მიმდინარე გახდება.
- LAST. მისი მითითების შემთხვევაში გაიცემა კურსორის შედეგობრივი ნაკრების უკანასკნელი სტრიქონი, რომელიც მიმდინარე გახდება.
- NEXT. მისი მითითების შემთხვევაში გაიცემა კურსორის შედეგობრივი ნაკრების მიმდინარე სტრიქონის შემდეგ მოთავსებული სტრიქონი, რომელიც მიმდინარე გახდება.

- PRIOR. მისი მითითების შემთხვევაში გაიცემა კურსორის შედეგობრივი ნაკრების მიმდინარე სტრიქონის წინ მოთავსებული სტრიქონი, რომელიც მიმდინარე გახდება.
- ABSOLUTE { n | @nvar }. გასცემს სტრიქონს კურსორის მთლიან შედეგობრივ ნაკრებში მისი აბსოლუტური რიგითი ნომრის მიხედვით.
- RELATIVE { n | @nvar }. გასცემს სტრიქონს, რომელიც მდებარეობს მიმდინარე სტრიქონიდან n სტრიქონის შემდეგ, თუ n დადებითია და n სტრიქონით წინ, თუ n უარყოფითია. გაცემული სტრიქონი ხდება მიმდინარე. თუ მითითებულია ნულოვანი მნიშვნელობა, მაშინ მიმდინარე სტრიქონი გაიცემა.
- INTO @variable_name [,...n]. ეს არგუმენტი მიუთითებსს იმ ცვლადების სიას, რომლებიც შეიცავენ დასაბრუნებელი სტრიქონის სვეტების შესაბამის მნიშვნელობებს. ცვლადების მითითების მიმდევრობა უნდა შეესაბამებოდეს კურსორში სვეტების მიმდევრობას. ამასთან, ცვლადის ტიპი უნდა ემთხვეოდეს სვეტის ტიპს. თუ ეს არგუმენტი მითითებული არ არის, მაშინ მონაცემები ეკრანზე გამოიტანება.

მონაცემების შეცვლა. კურსორის საშუალებით მონაცემების შესაცვლელად UPDATE ბრძანება გამოიყენება. მისი სინტაქსია:

UPDATE ცხრილის_სახელი SET { სვეტის_სახელი = { DEFAULT | NULL | გამოსახულება } } [,..n]
WHERE CURRENT OF კურსორის_სახელი

კურსორის დახურვა. კურსორის დახურვა ათავისუფლებს მისთვის გამოყოფილ რესურსებს და შლის კურსორში მოთავსებულ სტრიქონებს. დახურვის დროს მოიხსნება კურსორის მუშაობის დროს დაყენებული ბლოკირებები. კურსორი, რომელიც დაიხურა, მაგრამ არ გათავისუფლდა, შეიძლება განმეორებით გაიხსნას. კურსორის დასახურად გამოიყენება CLOSE ბრძანება, რომლის სინტაქსია:

CLOSE { { [GLOBAL] კურსორის_სახელი } | @cursor_variable_name }

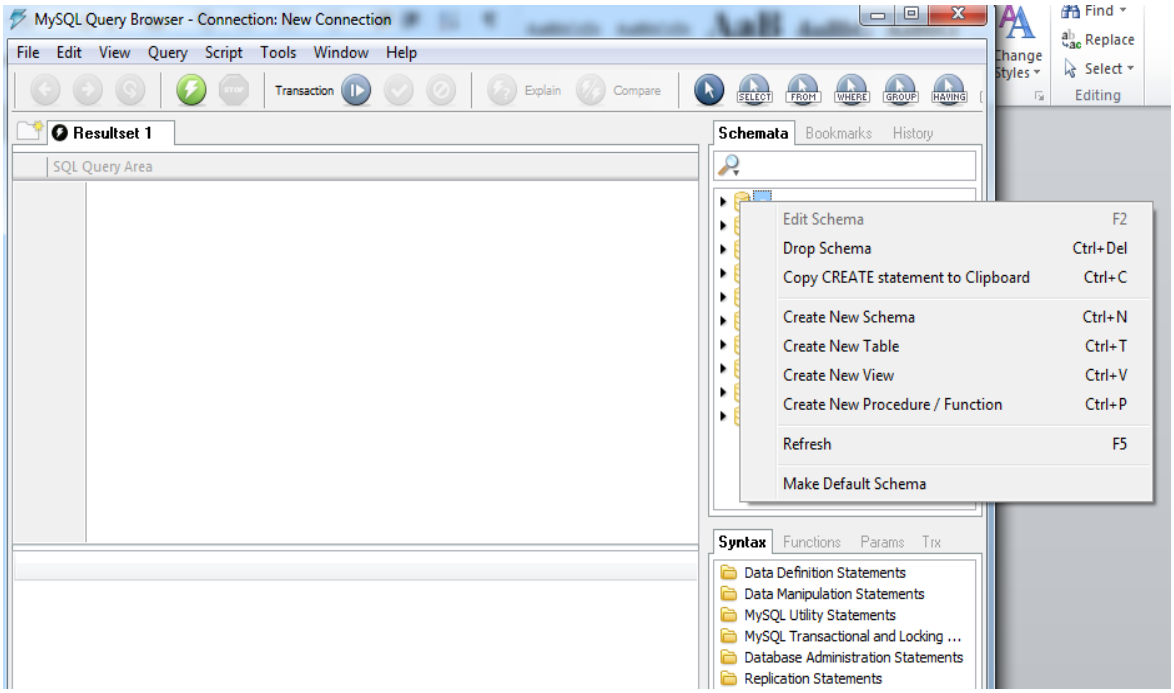
მაგალითი:

```
CREATE PROCEDURE curdemo()
BEGIN
DECLARE done INT DEFAULT 0;
DECLARE a CHAR(16);
DECLARE b,c INT;
DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
OPEN cur1;
OPEN cur2;
REPEAT
FETCH cur1 INTO a, b;
FETCH cur2 INTO c;
IF NOT done THEN
```

```
IF b < c THEN
INSERT INTO test.t3 VALUES (a,b);
ELSE
INSERT INTO test.t3 VALUES (a,c);
END IF;
END IF;
UNTIL done END REPEAT;
CLOSE cur1;
CLOSE cur2;
END
```

5. პროცედურებისა და ფუნქციების შექმნა MySQL Query Browser გამოყენებით

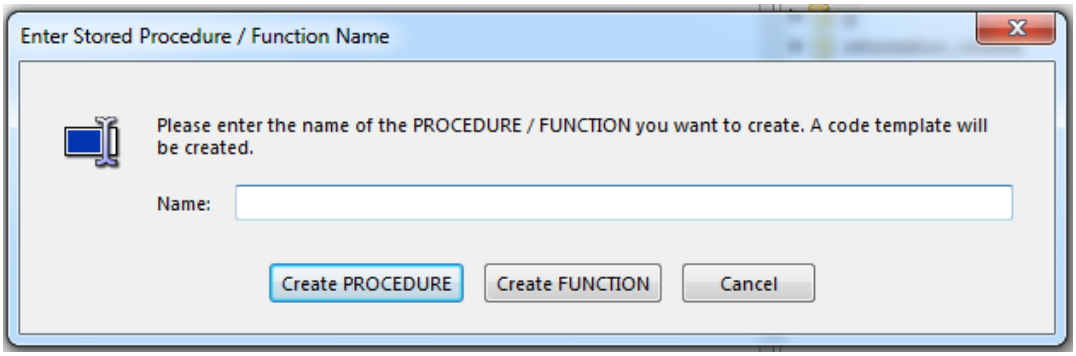
პროცედურებისა და ფუნქციების შექმნისათვის გამოიყენება MySQL Query Browser. ამისათვის ჩვენს მონაცემთა ბაზაზე მარჯვენა დაწკაპუნებით ვხსნით popup მენიუს, სადაც ვირჩევთ Create New Procedure/Function ბრძანებას.



სურ.8.2

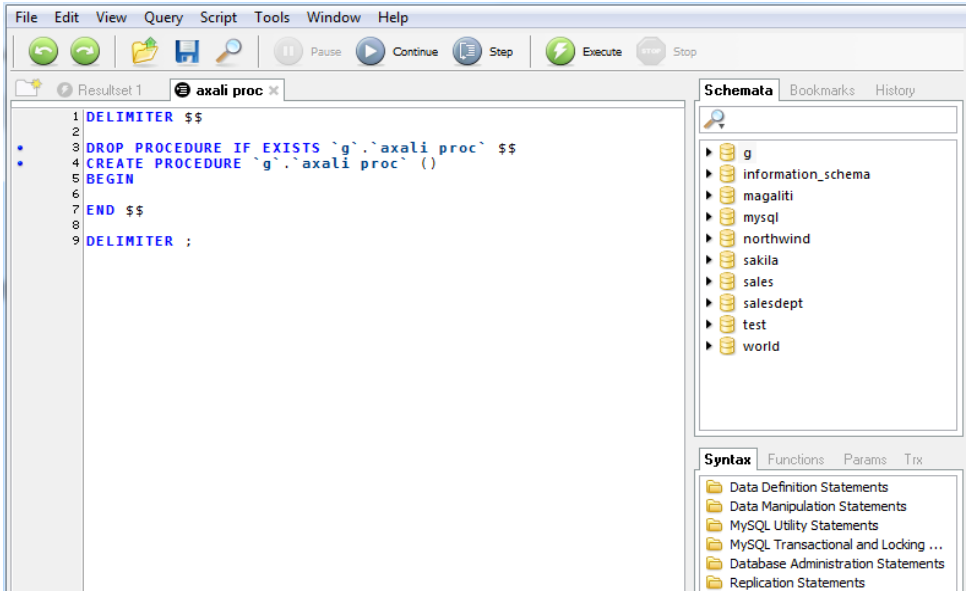
იხსნება ფანჯარა Enter Stored Procedure/Function Name.

შეგვაქვს პროცედურის ან ფუნქციის სახელი და შესაბამისად ვაჭერთ ღილაკს Create PROCEDURE ან Create FUNCTION.



სურ.8.3

იხსნება ფანჯარა პროცედურის მზა სასტარტო კოდით, რომელშიც უნდა შევიტანოთ ჩვენი პროცედურის კოდის ტანი.



სურ.8.4

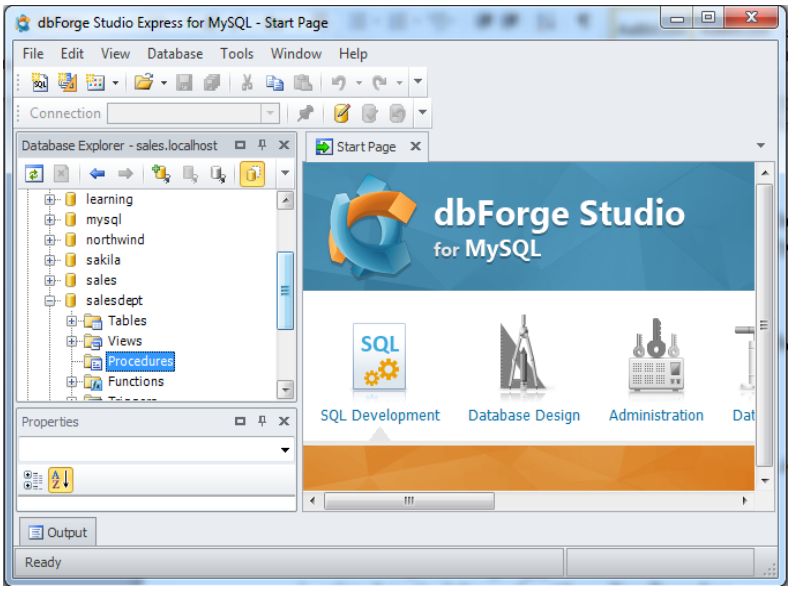
შემდეგ ვუშვებთ შესრულებაზე Execute ღილაკზე დაჭერით. წარმატებით შესრულების შემთხვევაში ვინახავთ პროცედურას, რომელსც გამოვიძახებთ საჭიროების მიხედვით.

6. პროცედურებისა და ფუნქციების შექმნა **dbForge Studio** for MySQL გამოიყენებით

dbForge Studio -ს გამოყენებით შესაძლებელია შენახული პროცედურების შექმნა, რედაქტირება და წაშლა.

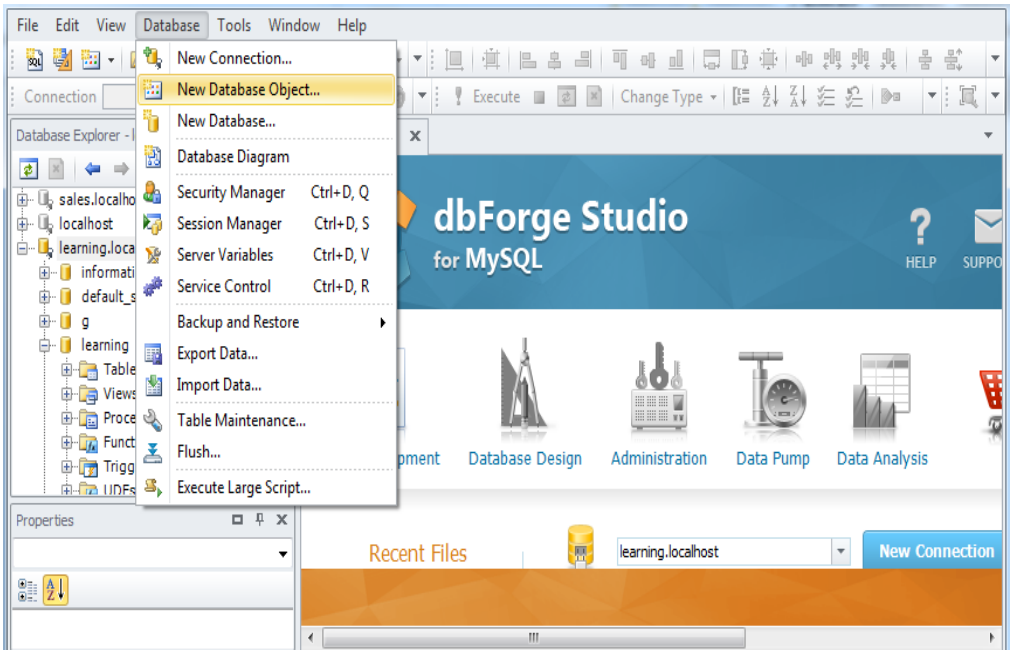
შენახული პროცედურების შექმნა

1. Database Explorer -ში ვპოულობთ Procedure-ს.



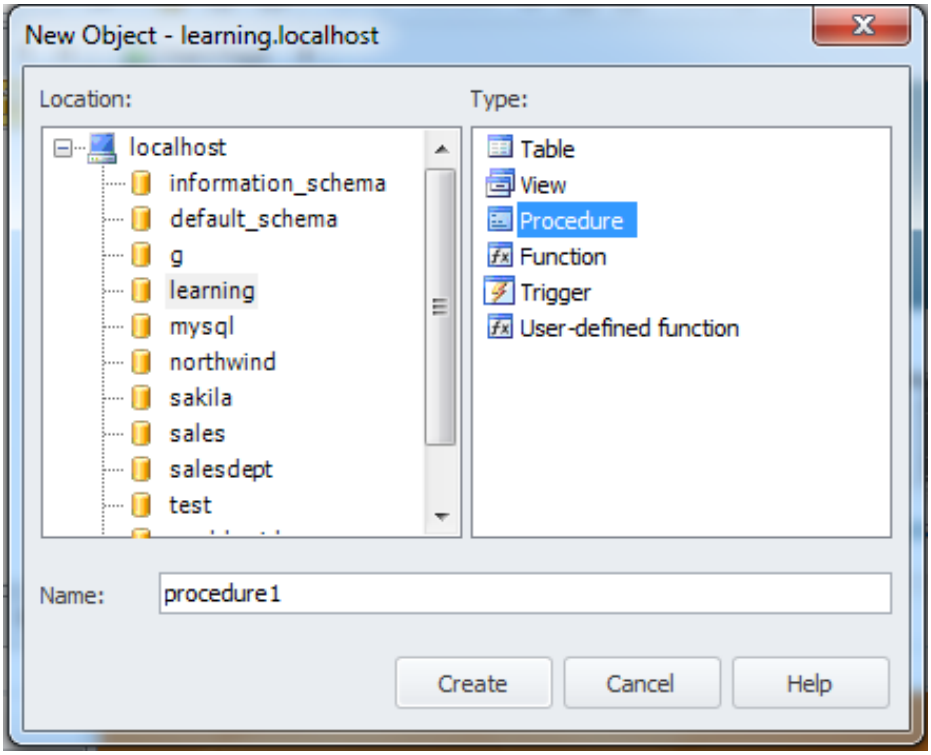
სურ.8.5

2. კონტექსტური მენიუდან ვირჩევთ **New Procedure/New Database Object.**



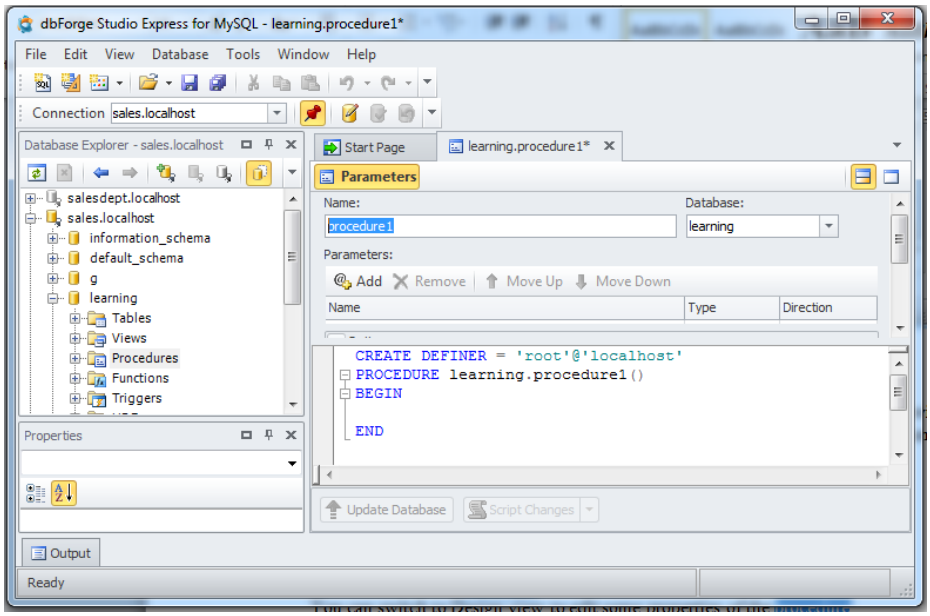
სურ.8.6

3. შეგვაქვს Procedure-ის სახელი და გახსნილ დიალოგურ ფანჯარაში ვაწკაპუნებთ **Create**.



სურ.8.7

4. ჩნდება ახალი დოკუმენტი ტექსტით. ვავსებთ პროცედურის თვისებებს და ვინახავთ დოკუმენტს.



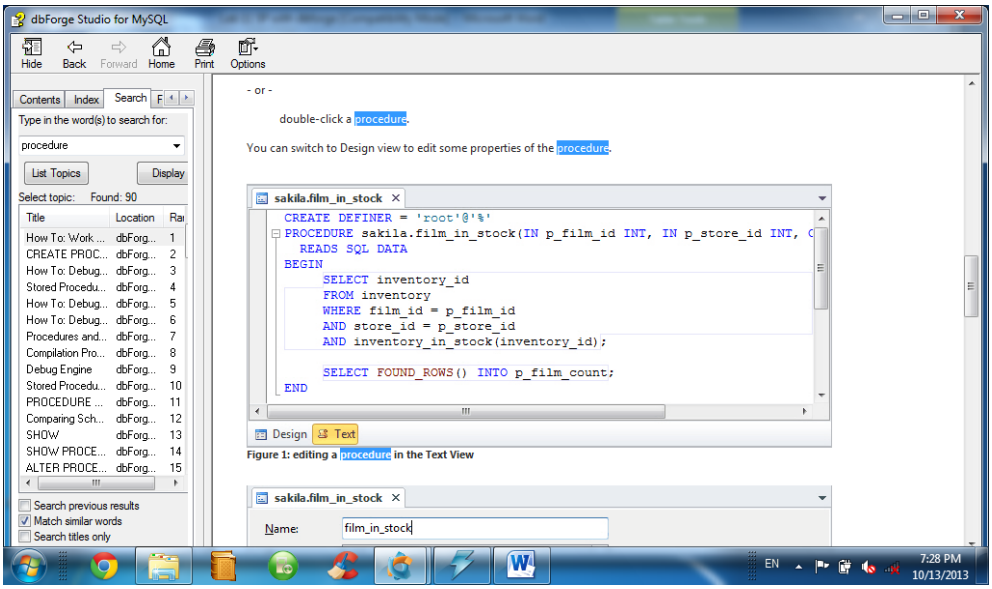
სურ.8.8

შენახული პროცედურის რედაქტირება

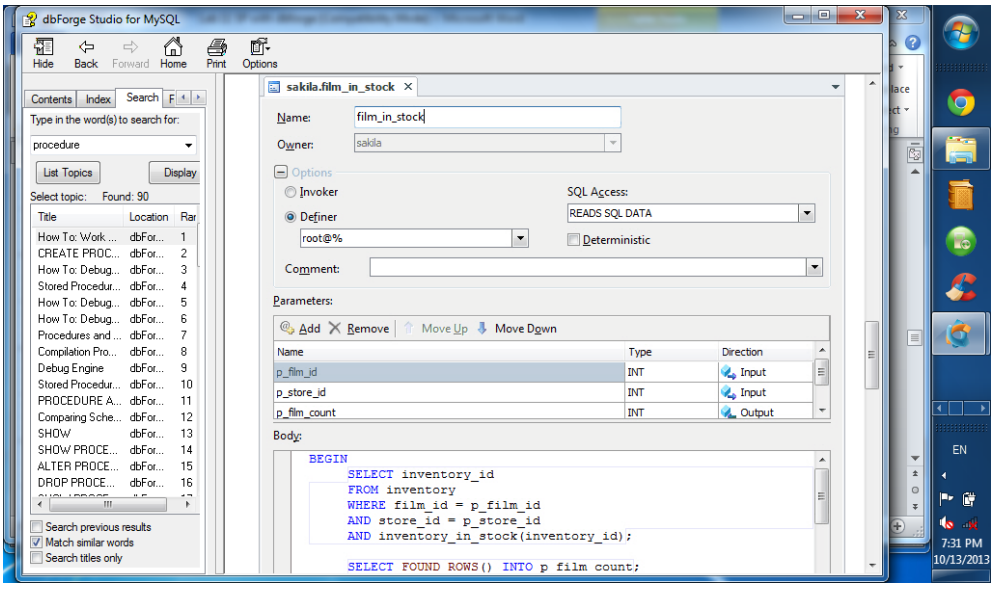
1. Database Explorer -ში ვპოულობთ Procedure-ს.

2. Procedure-ზე მარჯვენა დაწკაპუნებით იხსნება კონტექსტური მენიუ, სადაც ვირჩევთ **Open**.

ან ორჯერ ვაწკაპუნებთ Procedure-ზე. გახსნილ ფანჯარაში შეგვიძლია Procedure - თვისებების რედაქტირება.



სურ.8.9



სურ.8.10

შენახული პროცედურის შესრულება

პროცედურის შესრულებისათვის საჭიროა Database Explorer -ში Procedure-ზე მარჯვენა დაწკაპუნება და შემდგომ ვირჩევთ **Execute**.

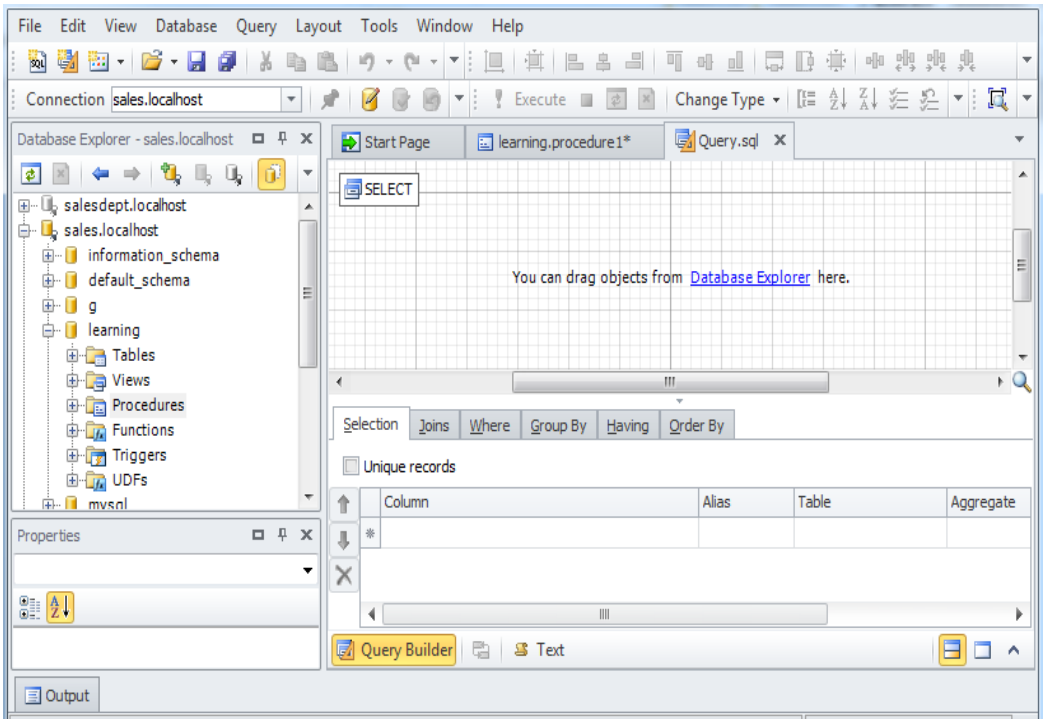
შენახული პროცედურის შესრულების შეჩერება

ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Stop Execution**.

შენახული პროცედურის წაშლა

ამისათვის საჭიროა Database Explorer -ში Procedure-ზე მარჯვენა დაწკაპუნება დაა

შემდგომ ვირჩევთ **Delete**.

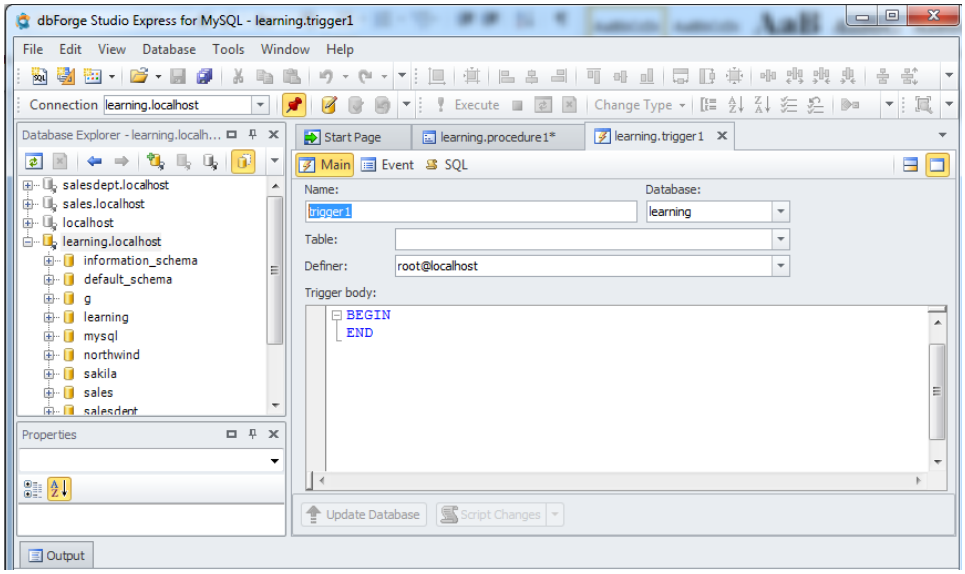


სურ.8.11

7. ტრიგერების შექმნა dbForge Studio for MySQL გამოყენებით

ტრიგერის შექმნა

1. Database Explorer -ში ვპოულობთ **Triggers** და ვაფართოებთ მას.
2. კონტექსტურ მენიუში ვირჩევთ **New Trigger**. იხსნება ახალი დოკუმენტი.



სურ.8.12

3. შეგვაქვს ტრიგერის სახელი. შესაძლებელიაობიექტისათვის სქემის შეცვლა.

4. შევავსოთ ტრიგერის ტანი.

5. ჩავრთოთ **Event** იარლიყი და განვსაზღვროთ ხდომილობა ტრიგერისათვის.

6.შევინახოთ დოკუმენტი.

ტრიგერის რედაქტირება

1. Database Explorer -ში ვპოულობთ **Triggers** და ვაფართოებთ მას.

2. კონტექსტურ მენიუში ვირჩევთ **Edit Trigger** from context menu.

Trigger Editor -ში არის ორი იარლიყი: **Main** და **Event**. **Main** -ში ჩვენ ვირჩევთ ტრიგერის ტიპს და ვავსებთ მის ტანს. **Event** -ში ჩვენ შეგვიძლია ხდომილობის გაშვება ტრიგერის შესრულებისათვის.

ტრიგერის წაშლა

ტრიგერის კონტექსტური მენიუდან ვირჩევთ **Delete**.

ლაბორატორიული სამუშაო 9
ბლოკირებები და ტრანზაკციები

სამუშაოს მიზანი:

1. ბლოკირებები
2. ტრანზაკციები

1. ბლოკირება

MySQL მონაცემთა ბაზის ცხრილის ბლოკირებისათვის გამოიყენება ბრძანება LOCK TABLES შემდეგი სინტაქსისით:

LOCK {TABLE | TABLES}
<table name> [AS <alias>] {READ [LOCAL] | [LOW_PRIORITY] WRITE}
[{, <table name> [AS <alias>] {READ [LOCAL] | [LOW_PRIORITY] WRITE}}...]

მაგალითი:

CREATE TABLE table1 (who VARCHAR(30) NOT NULL);
LOCK TABLE table1 READ;
INSERT INTO table1 VALUES ('სესია1');

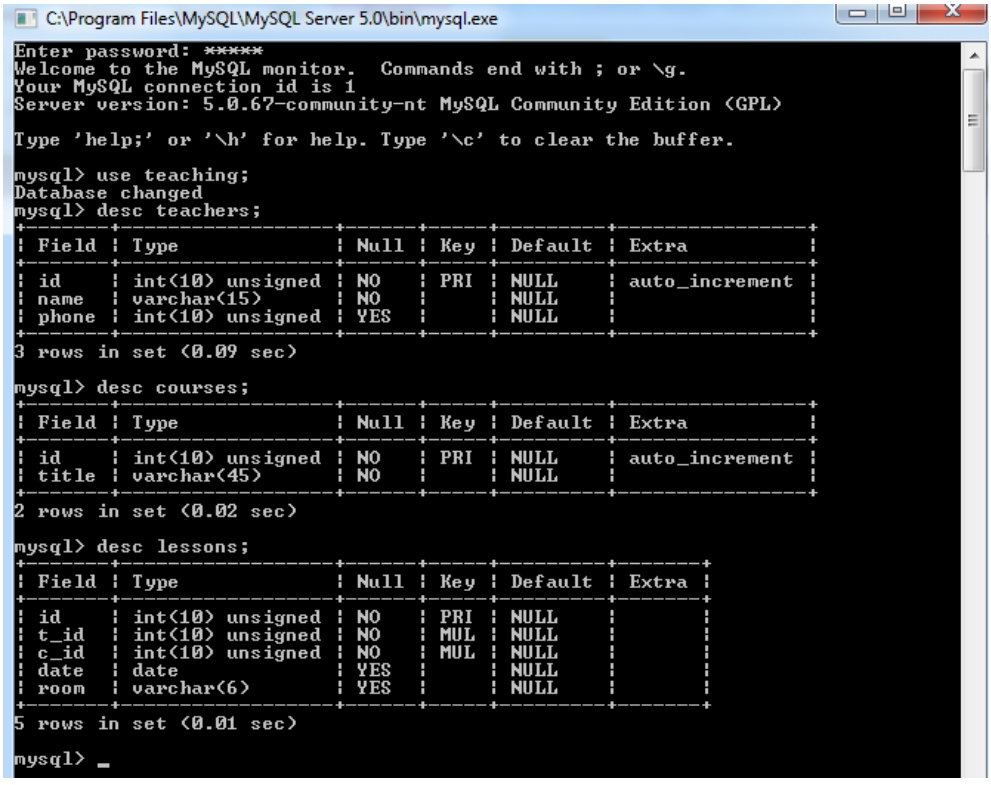
ცხრილის განბლოკვისათვის გამოიყენება ბრძანება UNLOCK TABLES შემდეგი სინტაქსისით:

UNLOCK {TABLE | TABLES}

ბლოკირება კონსოლის მეშვეობით

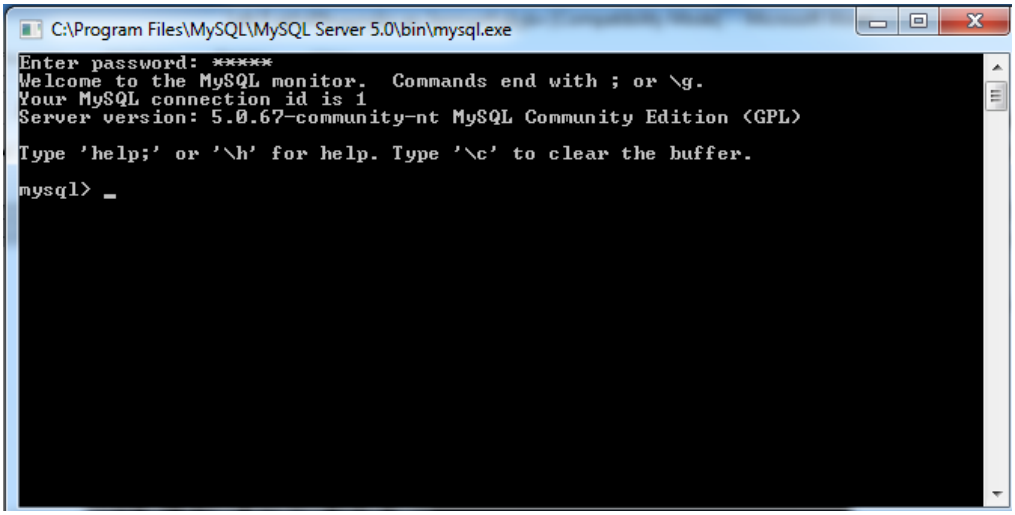
განვიხილოთ ბლოკირება კონკრეტულ მაგალითზე.

თავდაპირველად ვქმნით მონაცემთა ბაზას სამი ცხრილით:

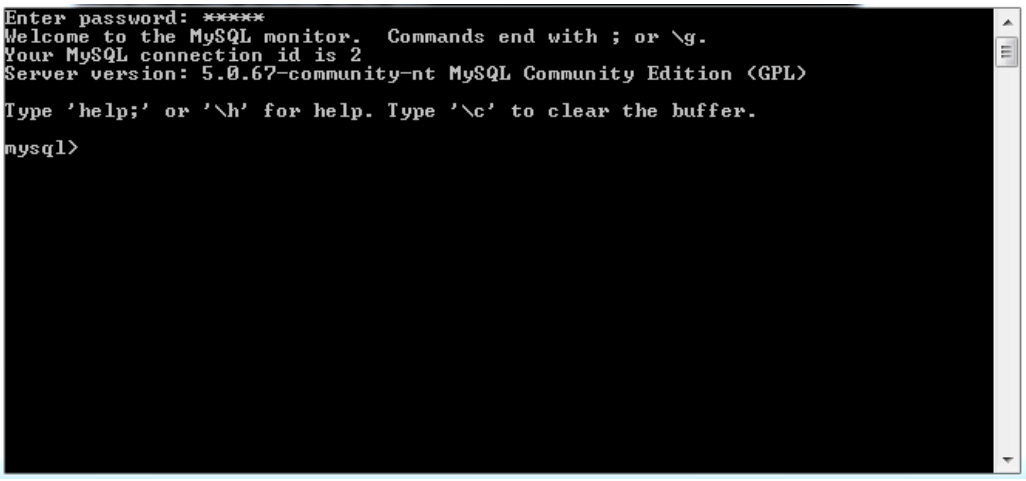


სურ. 9.1

ვთქვათ, ასევე გვაქვს ორი მომხმარებელი, ერთიდაიგივე Account-ით, ოღონდ სხვადასხვა Connection - ით.



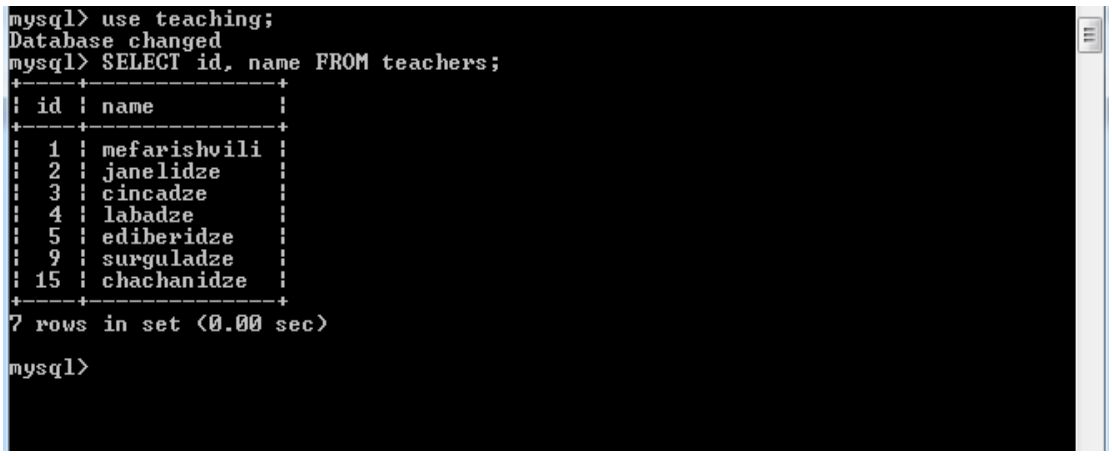
სურ. 9.2



სურ. 9.3

ორივე მომხმარებელს თავისუფლად შეუძლიათ ცხრილის მონაცემების დათვალიერება ბრძანებით:

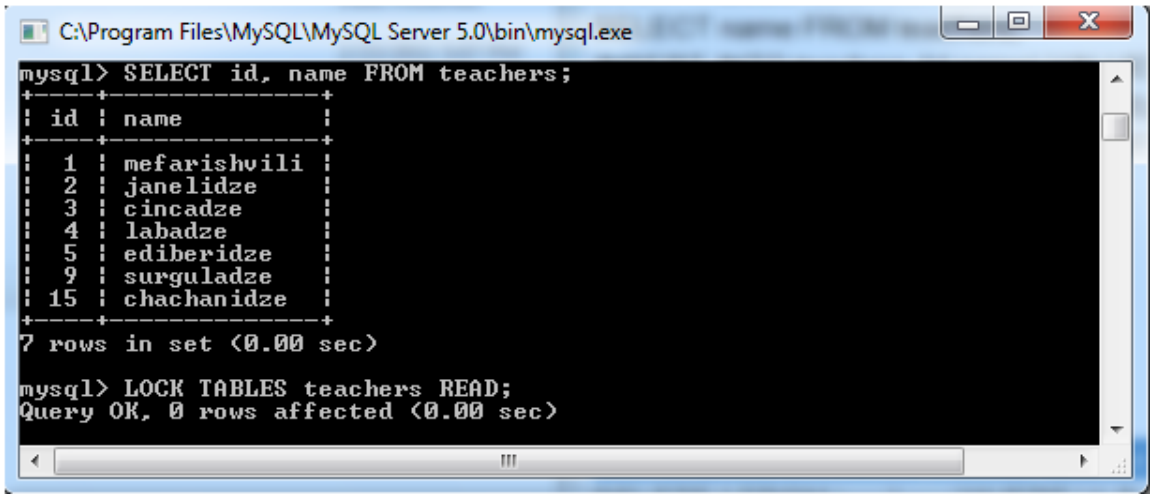
SELECT id, name FROM teachers;



სურ. 9.4

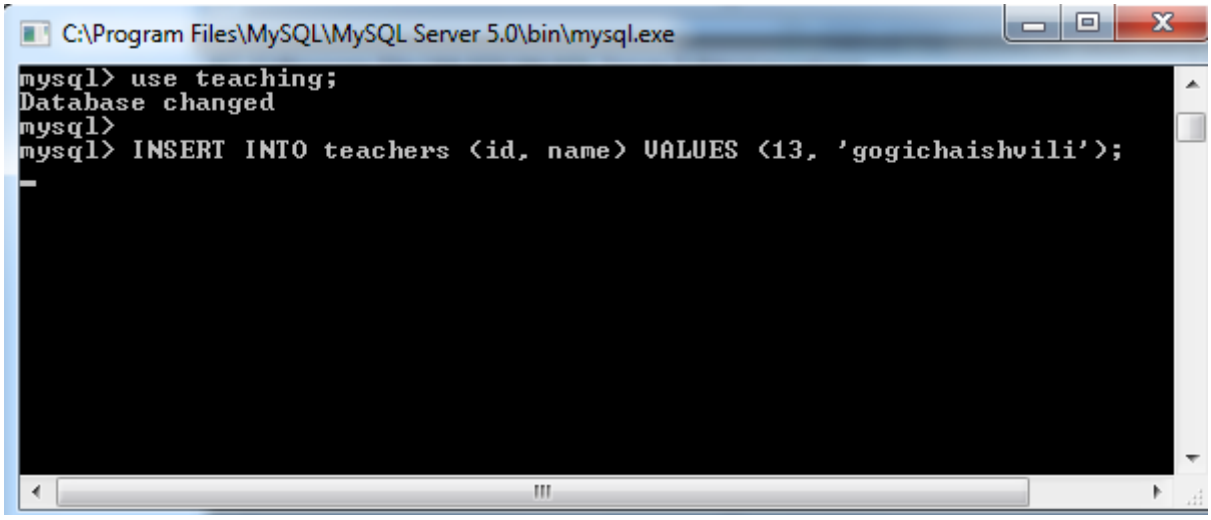
დავბლოკოთ ერთი ცხრილი წაკითხვაზე:

LOCK TABLES teachers READ;



სურ. 9.5

თუ მეორე მომხმარებელი, რომელსაც წარმოდგენა არ აქვს პირველის მიერ ცხრილის ბლოკირებაზე, შეეცდება ცხრილში ახალი ჩანაწერის გაკეთებას მცდელობა უშედეგო იქნება ანუ იგი „ჩამოეკიდება“.



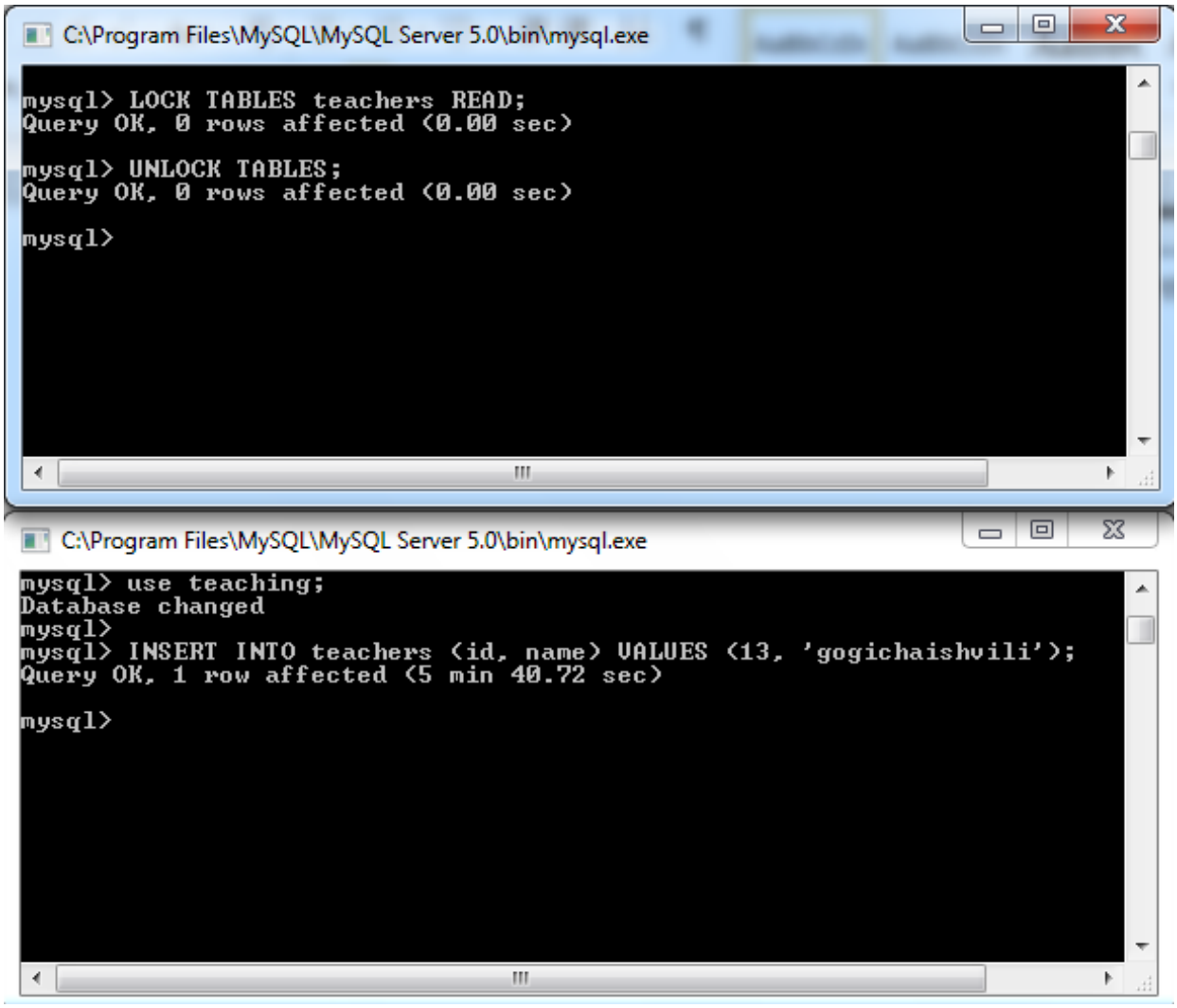
სურ. 9.6

პირველი მომხმარებელი კი უპრობლემოდ გააგრძელებს მუშაობას. ბუნებრივია, რომ ამ ხნის განმავლობაში სხვა მომხმარებლებიც ვერ განახორციელებენ წვდომას ამ ცხრილთან.

ადრე თუ გვიან, მუშაობის დამთავრების შემდეგ პირველი მომხმარებელი ახორციელებს განბლოკვას ბრძანებით:

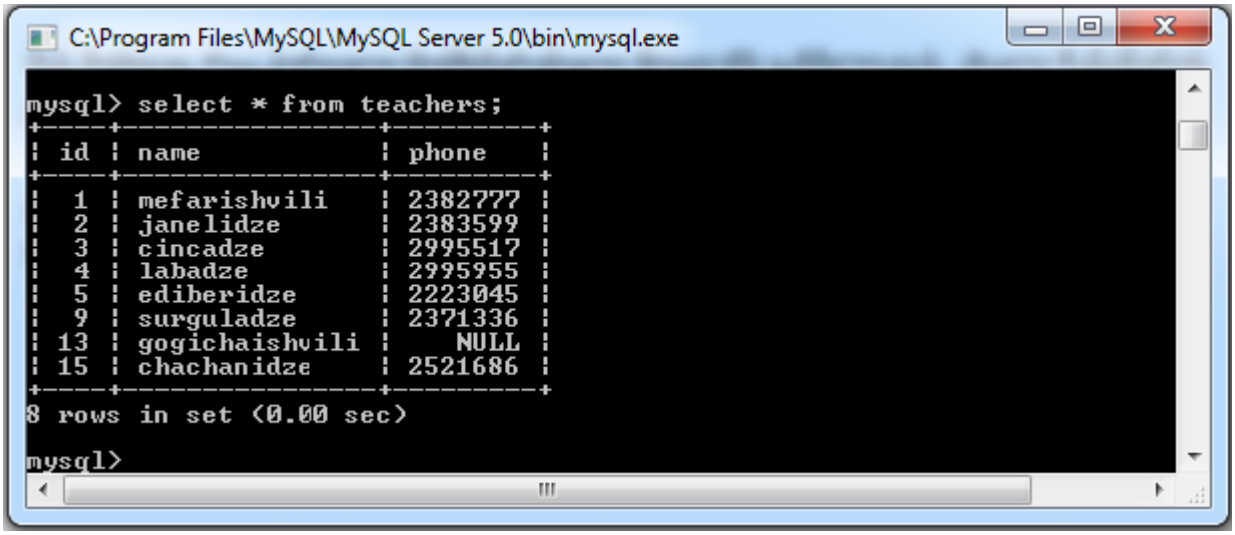
UNLOCK TABLES;

მხოლოდ ამის შემდეგ იმავდროულად ავტომატურად მოხდება დაწყებული ბრძანების შესრულება ანუ მეორე მომხმარებელი შეძლებს ახალი ჩანაწერის გაკეთებას.



სურ. 9.7

შესაბამისად, ცხრილში შესრულდა ტრანზაკცია.

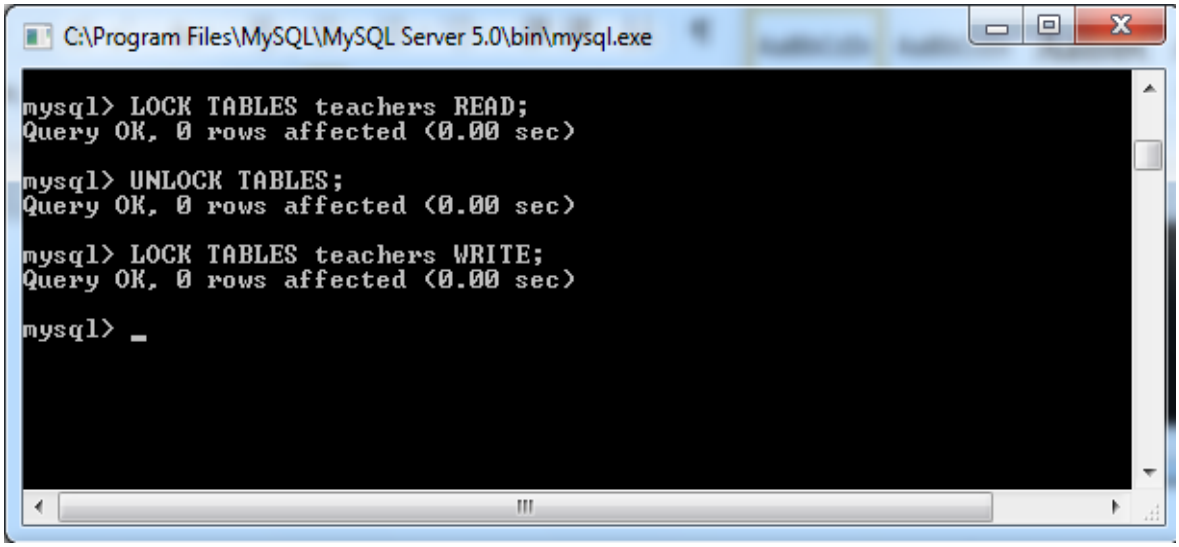


სურ. 9.8

ანალოგიურად, როდესაც პირველ მომხმარებელს ჯერ კიდევ არ აქვს დასრულებული მონაცემებზე მუშაობა ანუ ნაწილ-ნაწილ ამზადებს მონაცემებს,

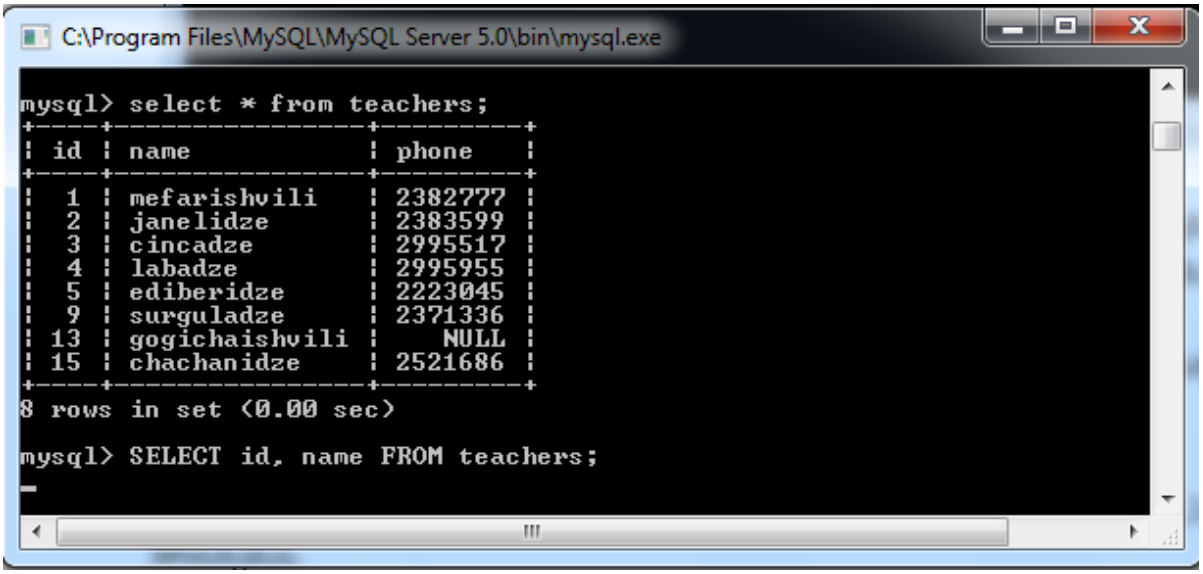
საჭირო ხდება ბლოკირება ჩაწერაზე ბრძანებით:

```
LOCK TABLES teachers WRITE;
```



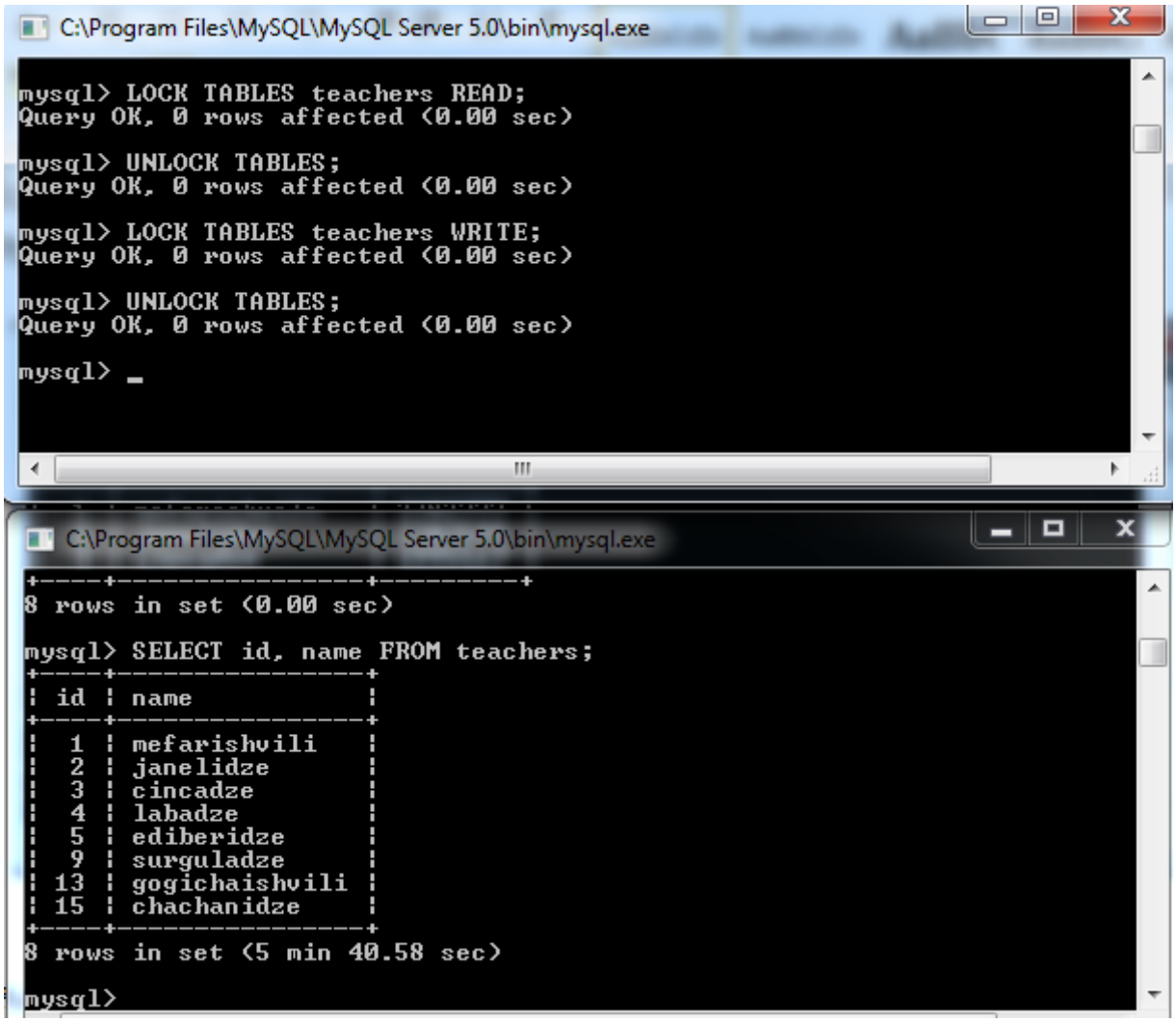
სურ. 9.9

რათა ასეთ შემთხვევაში სხვა მომხმარებლებისათვის არა თუ ჩაწერა ხდება შეუძლებელი, არამედ მონაცემთა ამორჩევაც და მათი მოთხოვნა „ეკიდება“ .



სურ. 9.10

მას შემდეგ, რაც პირველი მომხმარებელი მოახდენს განბლოკვას, ავტომატურად მეორე მომხმარებლისათვის იმავდროულად გახდება შესაძლებელი ახალი ჩანაწერის გაკეთება.



სურ. 9.11

ცალკეული ჩანაწერის ბლოკირების შემთხვევაში ვიყენებთ SELECT GET_LOCK ბრძანებას. აქ ფიგურირებს ფუნქცია GET_LOCK(სტრიქონი, ტაიმაუტი), რომელიც ახდენს ბლოკირებას სტრიქონში მოცემული სახელისა და დროის ხანგრძლიობის მიხედვით. ფუნქცია გვიბრუნებს 1, თუ ბლოკირება შედგა, ხოლო თუ დროის დაყოვნება გადააჭარბებს ტაიმაუტს (მაგალითად, სხვა კლიენტის მიერ ამავე სახელის დაბლოკვის გამო) – 0 ან NULL, თუ ადგილი ჰქონდა მტყუნებას (მეხსიერების გადავსების ან mysqladmin kill ბრძანების მიერ ნაკადის განადგურების გამო). თუ სახელი ბლოკირდება ერთი კლიენტის მიერ, მაშინ სხვა კლიენტის იმავე სახელის მქონე ყველა მოთხოვნა დაიბლოკება.

GET_LOCK() -ის მიერ მიღებული ბლოკირება მოიხსნება RELEASE_LOCK() -ის მეშვეობით, რომელიც გამოიყენება GET_LOCK() -ის ხელმეორედ გამოძახებისათვის.

```
mysql> SELECT GET_LOCK('lock1',10) ;
```

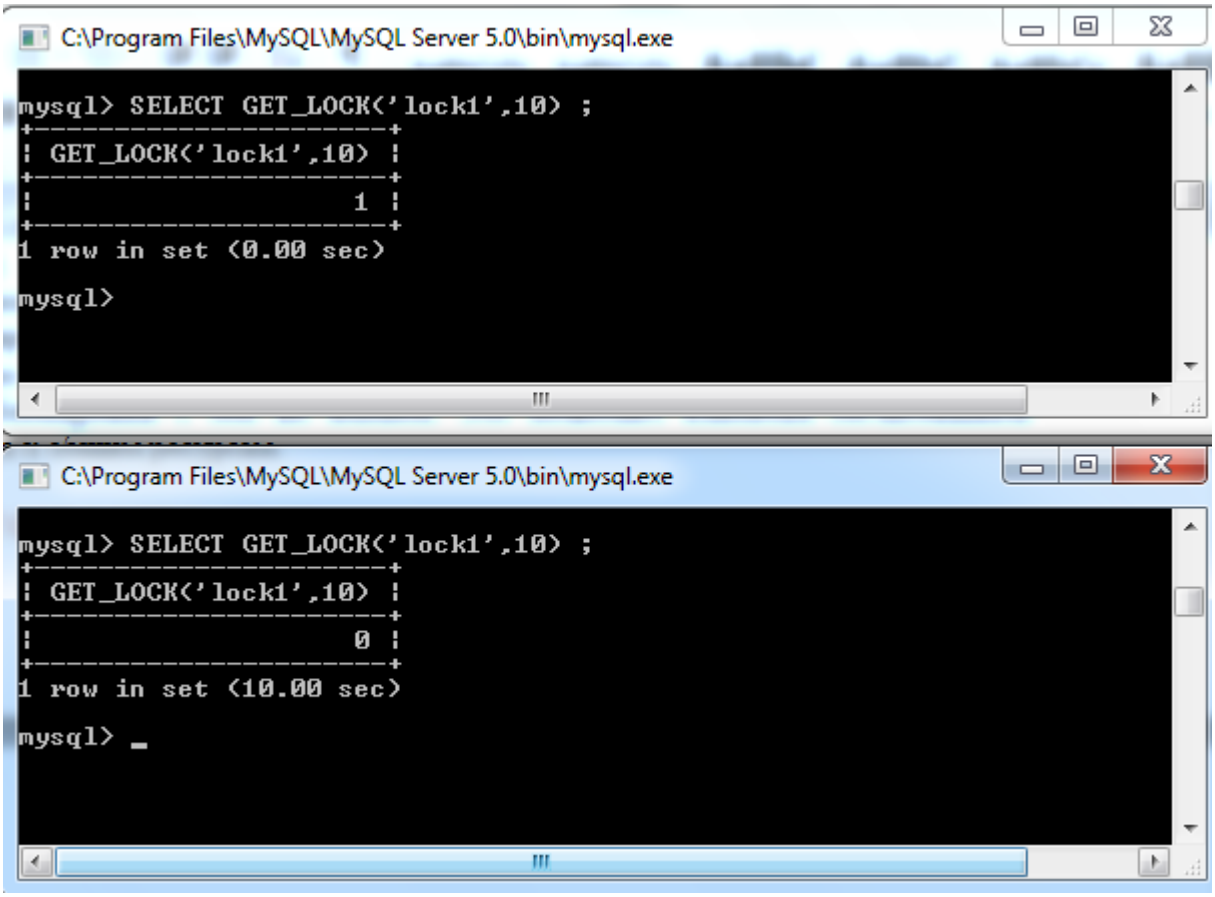
იმისათვის, რომ გავიგოთ მოცემული სტრიქონი ბლოკირებულია თუ არა, გამოიყენება ფუნქცია IS_FREE_LOCK:

```
mysql> SELECT IS_FREE_LOCK('lock2');
```

თუ ფუნქცია აბრუნებს 0-ს, მაშინ ხაზი დაკავებულია. ვაბრუნებთ ამას ციკლში,

ვიდრე 0-ს არ დაგვიბრუნებს.

```
mysql> SELECT GET_LOCK('lock2',10);
```

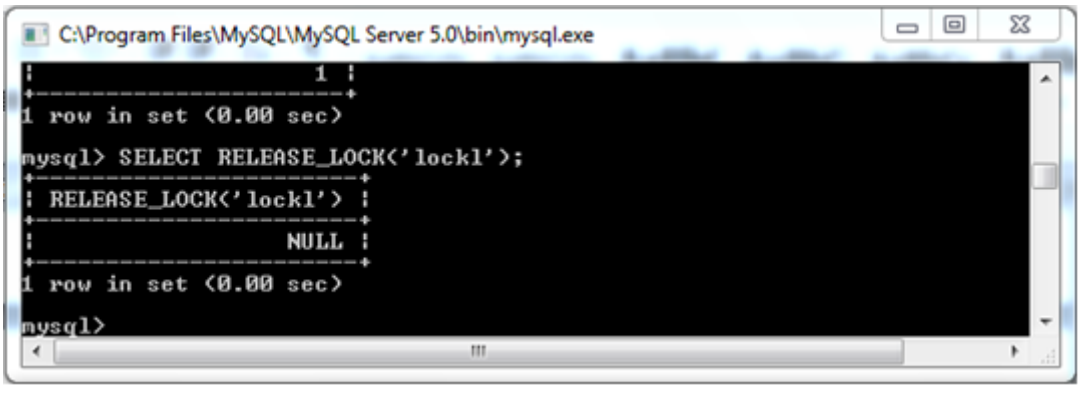


სურ. 9.12

ჩანაწერის განბლოკვისათვის ვიყენებთ SELECT RELEASE_LOCK ბრძანებას.

```
mysql> SELECT RELEASE_LOCK('lock2');
mysql> SELECT RELEASE_LOCK('lockl');
```

აღსანიშნავია, რომ RELEASE_LOCK() ფუნქციის განმეორებითი გამოძახება აბრუნებს NULL-ს, რადგან ბლოკირება 'lock1' ავტომატურად იქნა მოხსნილი GET_LOCK()ფუნქციის განმეორებითი გამოძახებით.



სურ. 9.13

2.ტრანზაკციები

ტრანზაკცის მართვისათვის გამოიყენება ბრძანებები:

SET AUTOCOMMIT = 0 | 1;

START TRANSACTOIN;

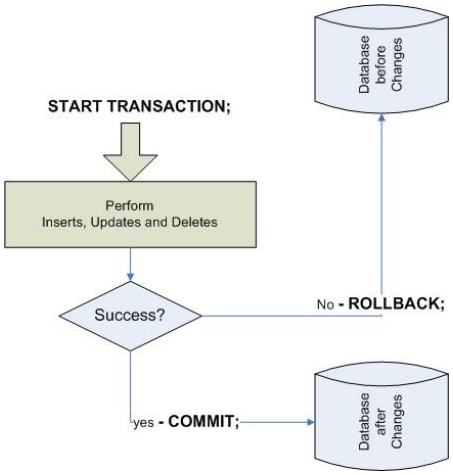
COMMIT;

ROLLBACK;

ამასთან:

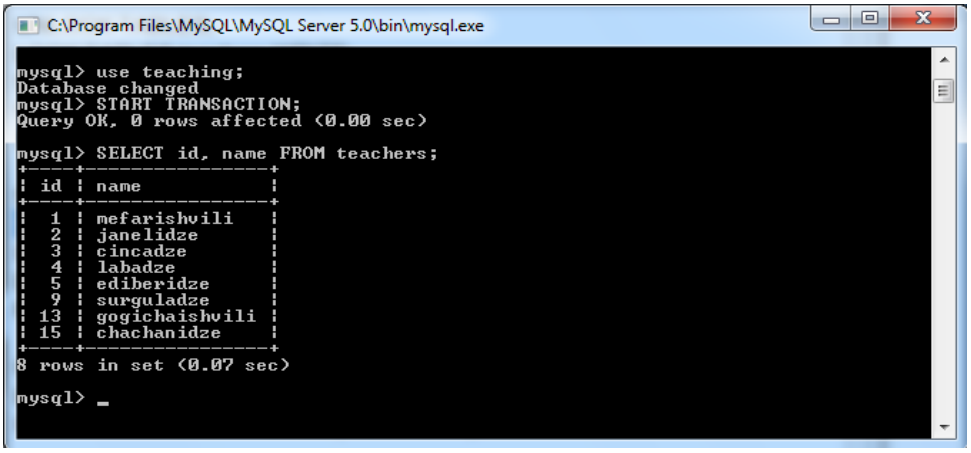
- SET AUTOCOMMIT = 1, რთავს ოპციას auto-commit option. იგი აგრეთვე ასრულებს მიმდინარე ტრანზაკციას.
- SET AUTOCOMMIT = 0 თიშავს ოპციას auto-commit option. იგი აგრეთვე იწყებს ახალ ტრანზაკციას. It also starts a new transaction

START TRANSACTION, COMMIT და ROLLBACK. ბრძანება COMMIT განსაზღვრავს ტრანზაქციის დასასრულს. თუ ტრანზაქციის შესრულების დროს ადგილი არ ჰქონდა შეცდომებს, მაშინ შესრულებული ცვლილებები დაფიქსირდება. ROLLBACK ბრძანების შესრულება იწვევს ტრანზაქციის შეწყვეტას და უკუქცევას (roll back). უკუქცევის დროს, შესრულებული ცვლილებები უქმდება და აღდგება სისტემის პირვანდელი მდგომარეობა.



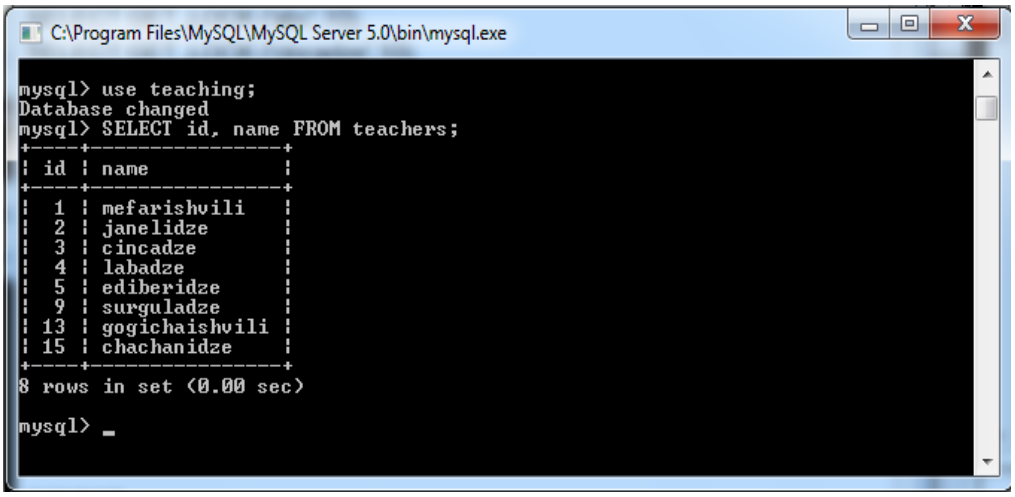
სურ. 9.14

მაგალითად: პირველი მომხმარებელი იწყებს ტრანზაკციას.



სურ. 9.15

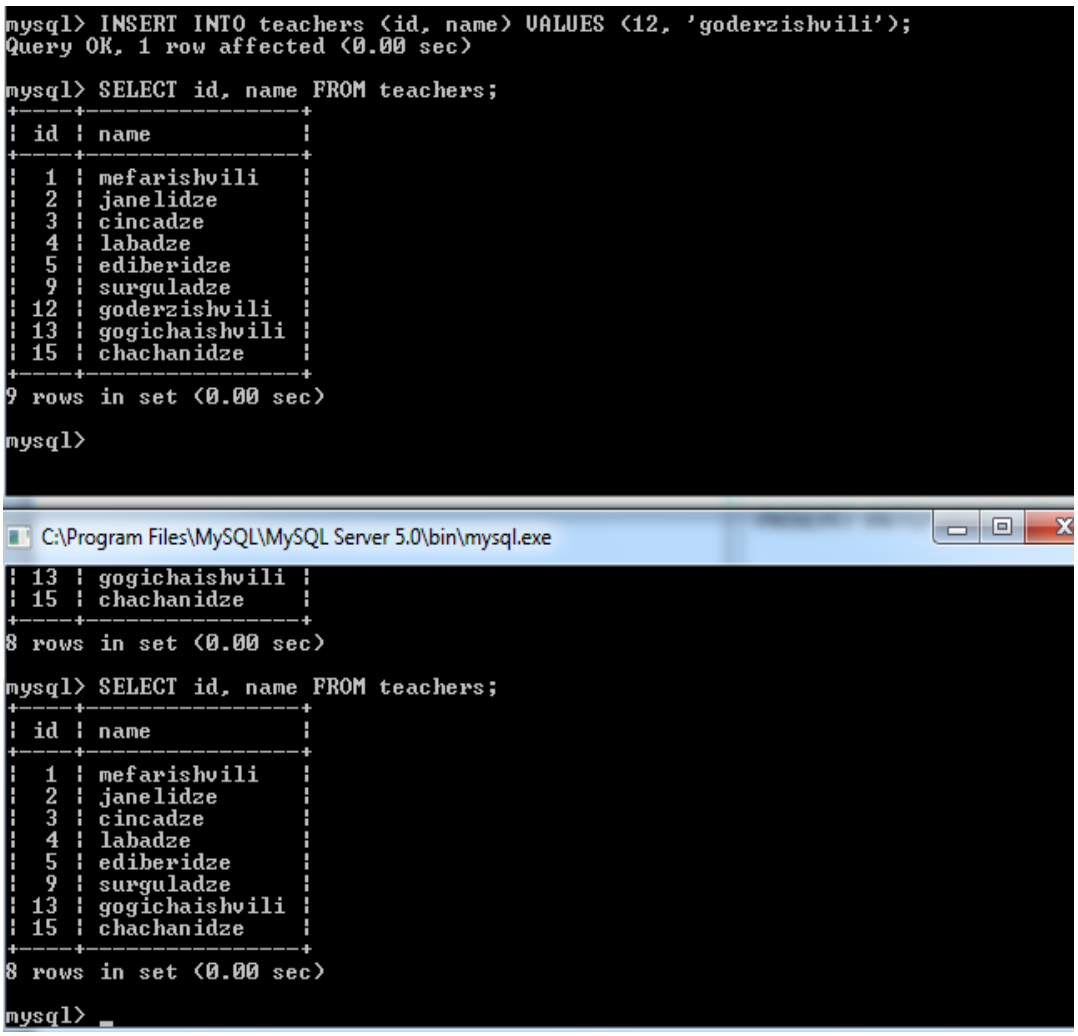
ამ დროს მეორე მომხმარებელს წარმოდგენაც არ აქვს ამის თაობაზე.



სურ. 9.16

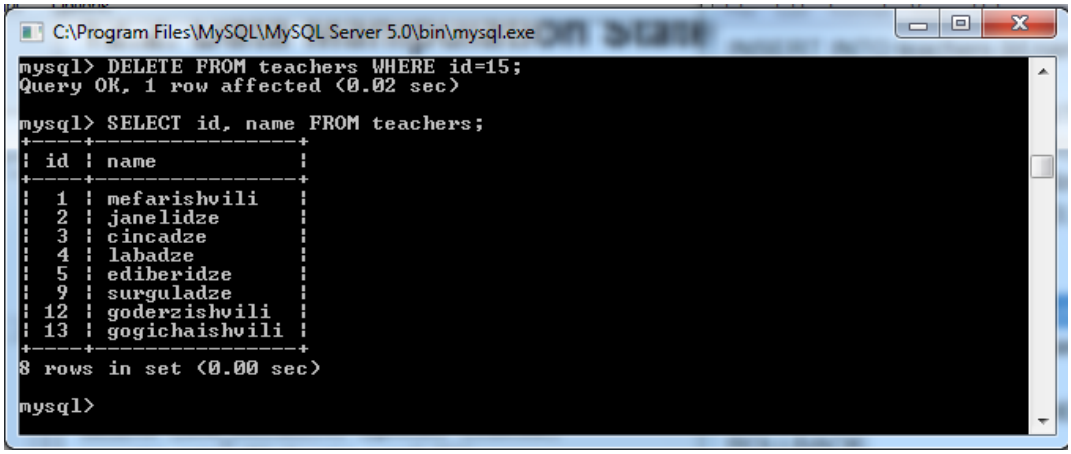
პირველ მომხმარებელს შეაქვს ახალი ჩანაწერი.

ტრანზაკცია შესრულდა პირველი მომხმარებლისათვის, მაშინ როცა მეორე მომხმარებლისათვის ყველაფერი უცვლელად დარჩა.



სურ. 9.17

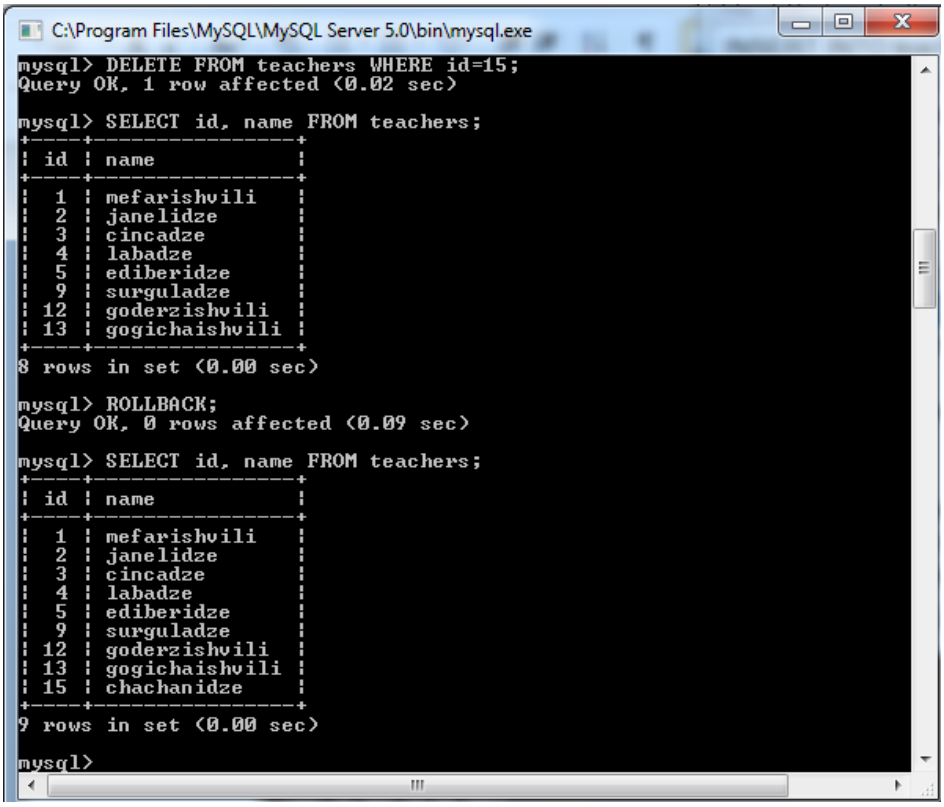
ვთქვათ პირველი მომხმარებელი შლის ერთ ჩანაწერს.



სურ. 9.18

ამ დროის განმავლობაში მეორე მომხმარებლისათვის არაფერი არ შეცვლილა. როცა უკვე მან გადაწყვიტა ახალი ჩანაწერის შეტანა, ტრანზაკცია აღარ შესრულდა.

შემდეგ პირველმა მომხმარებელმა საჭიროდ ჩათვალა გაეუქმებინა წინა გადაწყვეტილება ბრძანებით ROLLBACK.



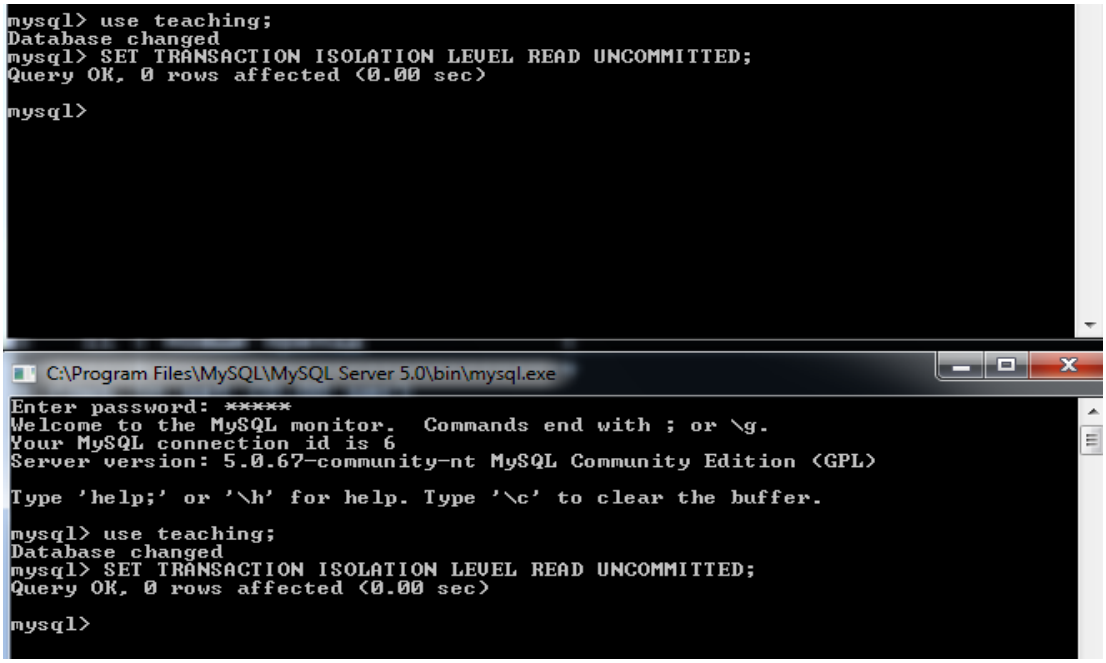
სურ. 9.19

თუ პირველი მომხმარებელი გააუქმებს თავის ტრანზაკციას, მაშინ მეორე მომხმარებლისათვის ავტომატურად შესრულდება ტრანზაკცია ანუ დაემატება ახალი ჩანაწერი. ხოლო თუ პირველი მომხმარებელი ასრულებს თავის ტრანზაკციას

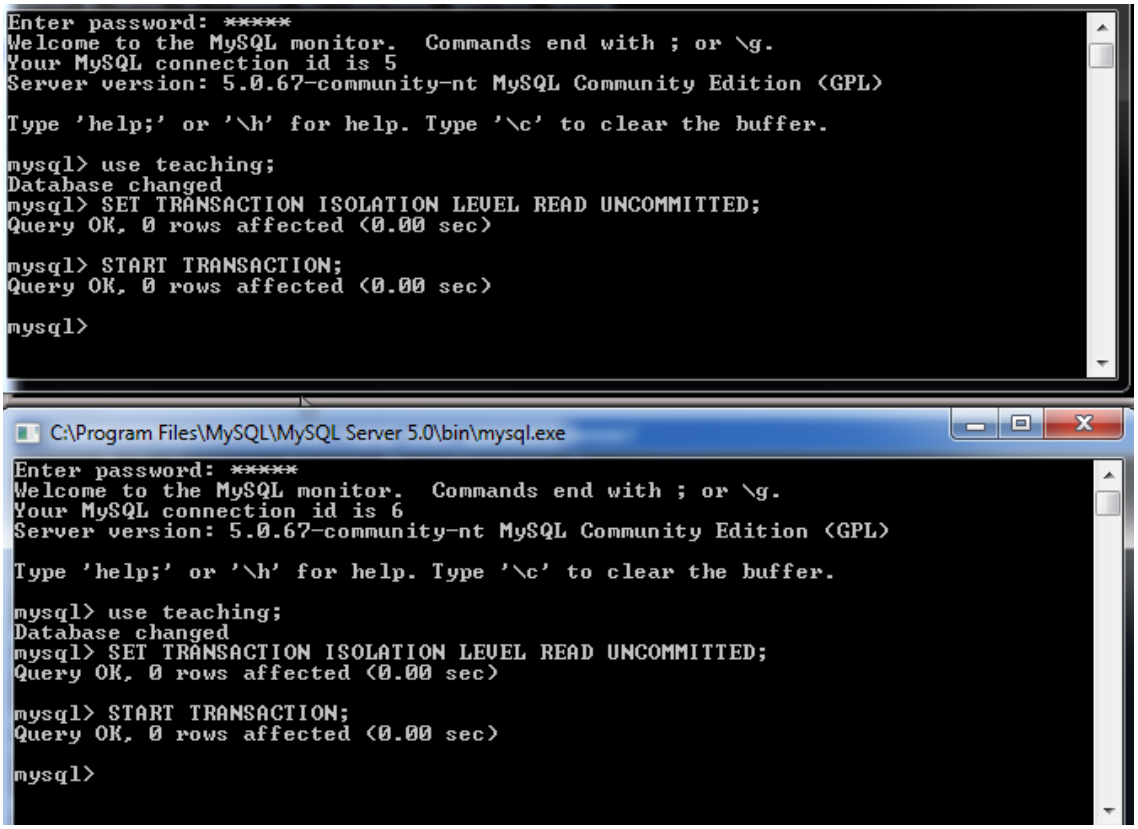
ბრძანებით COMMIT, მაშინ მეორე მომხმარებელსაც შეეძლება ტრანზაკცირებული ცხრილის ნახვა. ბრძანება COMMIT-ის შემდეგ ბრძანება ROLLBACK უკვე ვეღარ შესრულდება.

ტრანზაკციის იზოლაციის დონეები

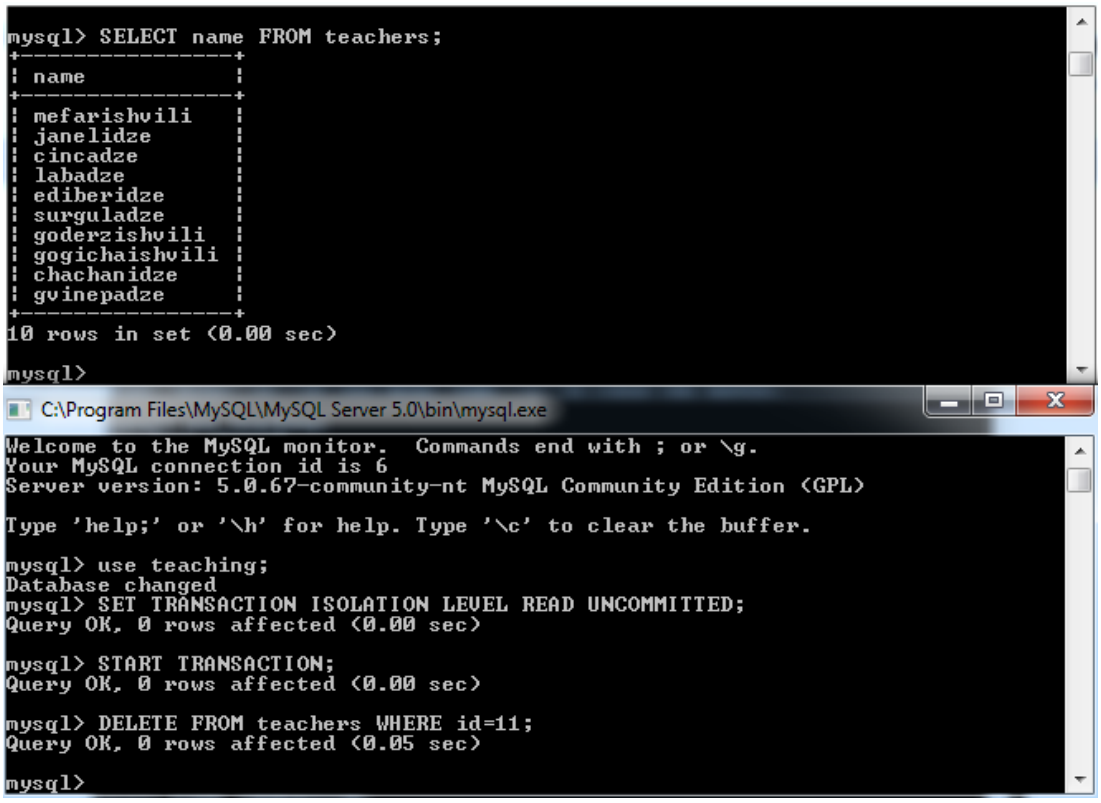
ტრანზაკციის იზოლაციის დონეების განსაზღვრისათვის გამოიყენება ბრძანება SET TRANSACTION.



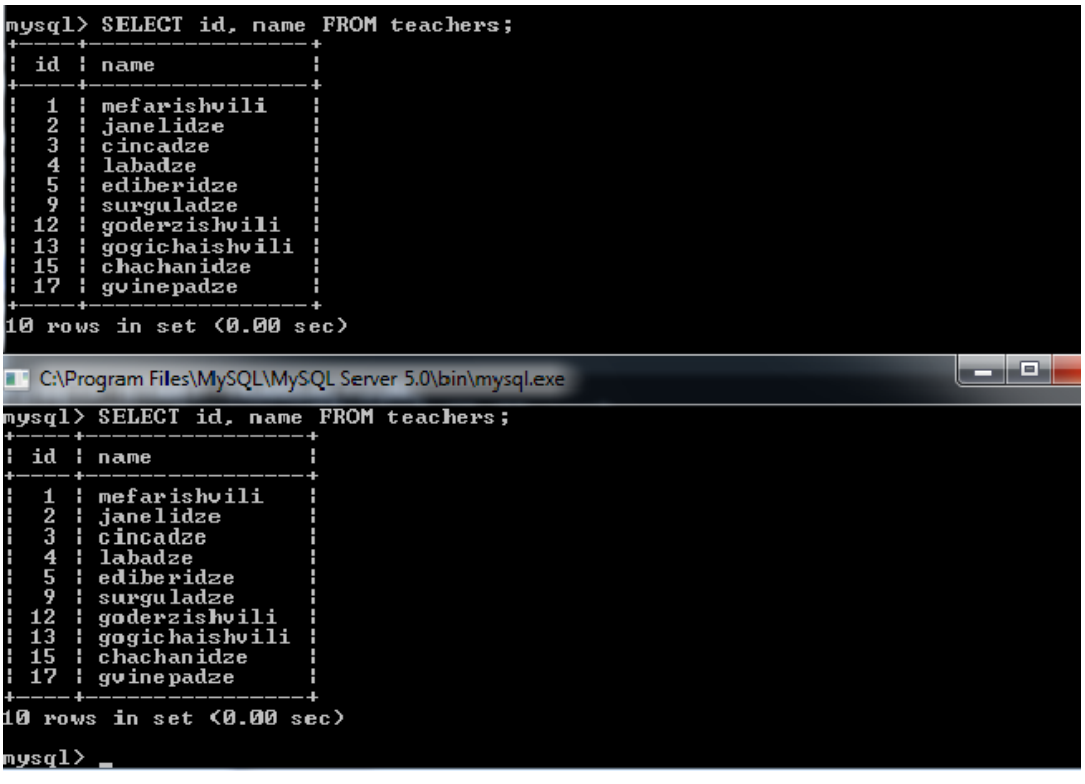
სურ. 9.20



სურ. 9.21



სურ. 9.22



სურ. 9.23

```
mysql> SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM teachers WHERE id=12;
Query OK, 1 row affected (0.07 sec)

mysql> SELECT id, name FROM teachers;
+----+--------------+
| id | name         |
+----+--------------+
|  1 | mefarishvili |
|  2 | janelidze    |
|  3 | cincadze     |
|  4 | labadze      |
|  5 | ediberidze   |
| 15 | chachanidze  |
+----+--------------+
6 rows in set (0.13 sec)
```

სურ. 9.24

```
mysql> SELECT id, name FROM teachers;
+----+--------------+
| id | name         |
+----+--------------+
|  1 | mefarishvili |
|  2 | janelidze    |
|  3 | cincadze     |
|  4 | labadze      |
|  5 | ediberidze   |
+----+--------------+
5 rows in set (0.00 sec)

mysql>
```

სურ. 9.25

```
mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT id, name FROM teachers;
+----+--------------+
| id | name         |
+----+--------------+
|  1 | mefarishvili |
|  2 | janelidze    |
|  3 | cincadze     |
|  4 | labadze      |
|  5 | ediberidze   |
| 15 | chachanidze  |
+----+--------------+
6 rows in set (0.00 sec)

mysql>
```

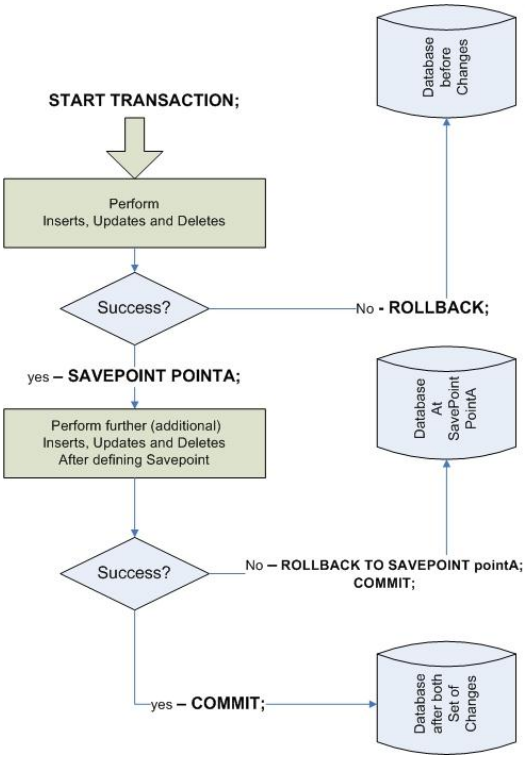
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

```
mysql> SELECT id, name FROM teachers;
+----+---------------+
| id | name          |
+----+---------------+
|  1 | mefarishvili  |
|  2 | janelidze     |
|  3 | cincadze      |
|  4 | labadze       |
|  5 | ediberidze    |
| 12 | goderzishvili |
+----+---------------+
6 rows in set (0.00 sec)
```

სურ. 9.26

ოპერატორი SAVEPOINT

ოპერატორი SAVEPOINT ადგენს იმ ტრანზაკციის დაწყების სახელდებულ წერტილს, რომელსაც გააჩნია გარკვეული იდენტიფიკატორი. ბრძანებები SAVEPOINT და ROLLBACK TO SAVEPOINT განაცალკევებენ ტრანზაკციის ნაწილებს. SAVEPOINT ბრძანება განსაზღვრავს მარკერს ტრანზაკციაში, ხოლო ბრძანება ROLLBACK TO SAVEPOINT წინასწარ განსაზღვრულ მარკერში (savepoint) ტრანზაკციის უკუქცევის (roll back) საშუალებას იძლევა.



სურ. 9.27

SAVEPOINT -ის სინტაქსისი შემდეგია:

SAVEPOINT <savepoint-name>

განვიხილოთ მაგალითი:

```
mysql> CREATE TABLE Books
    -> (
    ->     BookID SMALLINT NOT NULL PRIMARY KEY,
    ->     BookTitle VARCHAR(60) NOT NULL,
    ->     Copyright YEAR NOT NULL
    -> )
    -> ENGINE=INNODB;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql>
mysql> START TRANSACTION;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Books VALUES (103, 'Opera', 1966);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Books VALUES (104, 'Sql Server', 1932);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT sp1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Books VALUES (105, 'C', 1996);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Books VALUES (106, 'Pascal', 1980);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT sp1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Books VALUES (107, 'Postcards', 1992);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Books VALUES (108, 'Oracle', 1993);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> drop table Books;
Query OK, 0 rows affected (0.00 sec)
```

ლაბორატორიული სამუშაო 10
MySQL მონაცემთა ბაზების სარეზერვო ასლების შექმნა და აღდგენა

სამუშაოს მიზანი:

1. ბაზების სარეზერვო ასლების შექმნა და აღდგენა Mysqldump ბრძანების გამოყენებით
2. სარეზერვო ასლების შექმნა და აღდგენა MySql Administrator -ის გამოყენებით
3. სარეზერვო ასლების შექმნა და აღდგენა dbForge Studio for MySQL -ის გამოყენებით

1.ბაზების სარეზერვო ასლების შექმნა და აღდგენა Mysqldump ბრძანების გამოყენებით

MySQL-ს გააჩნია ბრძანებითი სტრიქონის უტილიტა სარეზერვო ასლების შექმნისა და აღდგენისათვის. **mysqldump** ბრძანებითი სტრიქონი MySQL ინსტალაციასთან ერთად არის bin დირექტორიაში და შეიძლება გამოყენებულ იქნას საჭიროებისამებრ.

1. ბრძანებით სტრიქონში შეგვაქვს ბრძანებები შემდეგი სინტაქსისით:

mysqldump –-user [user name] –-password=[password] [database name] > [dump file]

ან

mysqldump –u[user name] –p[password] [database name] > [dump file]

მაგალითად:

mysqldump –-user root –-password=myrootpassword db_test > db_test.sql

ან

mysqldump –uroot –pmyrootpassword db_test > db_test.sql

2. MySQL-ში მრავალი მონაცემთა ბაზის სარეზერვო ასლების შექმნისათვის შეგვაქვს ბრძანებები შემდეგი სინტაქსისით:

mysqldump –u[user name] –p[password] [database name 1] [database name 2] .. > [dump file]

მაგალითად:

mysqldump –-user root –-password=myrootpassword db_test db_second db_third > db_test.sql

3. MySQL-ში ყველა მონაცემთა ბაზის სარეზერვო ასლების შექმნისათვის შეგვაქვს ბრძანებები შემდეგი სინტაქსისით:

shell> mysqldump –u[user name] –p[password] –all-databases > [dump file]

4. MySQL-ში განსაზღვრული ცხრილის სარეზერვო ასლების შექმნისათვის ვკრეფთ შემდეგ ბრძანებებს:

shell> mysqldump --user [username] --password=[password] [database name] [table name] \
> /tmp/sugarcrm_accounts_contacts.sql

მაგალითად:

shell> mysqldump --user root --password=myrootpassword db_test customers \
> db_test_customers.sql

5. MySQL მონაცემთა ბაზის აღდგენა.

mysqldump უტილიტა გამოიყენება ასლის მისაღებად მხოლოდ MySQL dump -ის გამოყენებით. მონაცემთა ბაზის აღდგენა dump ფაილიდან, რომელიც წინა ბიჯზე შეიქმნა, შეიძლება **mysql** ბრძანების გამოყენებით.

```
shell> mysql --u [username] --password=[password] [database name] < [dump file]
```

მაგალითად:

```
shell> mysql --user root --password=myrootpassword new_db < db_test.sql
```

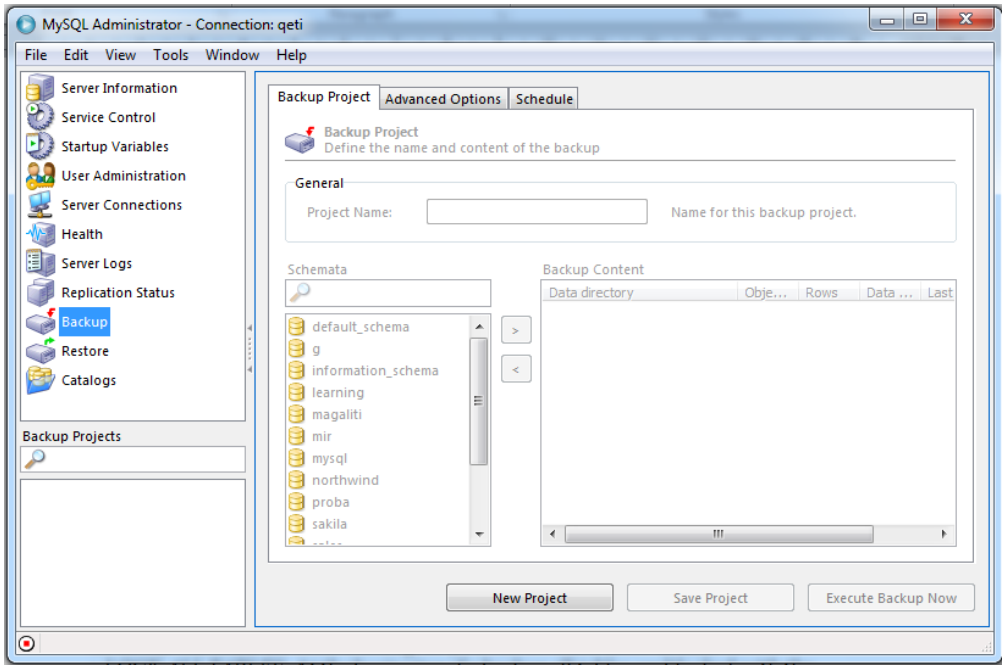
2.სარეზერვო ასლების შექმნა და აღდგენა MySql Administrator - ის გამოყენებით

MySql Administrator უტილიტის გამოყენების შემთხვევაში სრულდება შემდეგი ბიჯები:

1) გავხსნათ MySql Administrator.

2) ვირჩევთ **Backup** -ს მენიუს მარცხენა ნაწილში.

ამ მოდულთან ერთად შეგვიძლია მონაცემთა ბაზების სარეზერვო ასლების შექმნა.



სურ. 10.1

3) ვაწკაპუნებთ ღილაკზე "**New Project**" მარჯვენა ნაწილში.

4) სიიდან ვირჩევთ მონაცემთა ბაზას და ვაწკაპუნებთ ღილაკზე ">".

ჩანართში *ADVANCED OPTIONS* მოცემულია შემდეგი ოპციები:

- LOCK ALL TABLES.
- SINGLE TRANSACTION.
- NORMAL BACKUP.
- COMPLETE BACKUP.
- NO CREATES.
- NO EXTENDED INSERTS.
- ADD DROP TABLE.
- COMPLETE INSERTS.
- ANSI QUOTES.
- DISABLE KEYS.

ჩანართში *SCHEDULE* უტილიტა Administrator სარეზერვო ასლების შექმნის გეგმიურად შესრულების საშუალებას იძლევა.

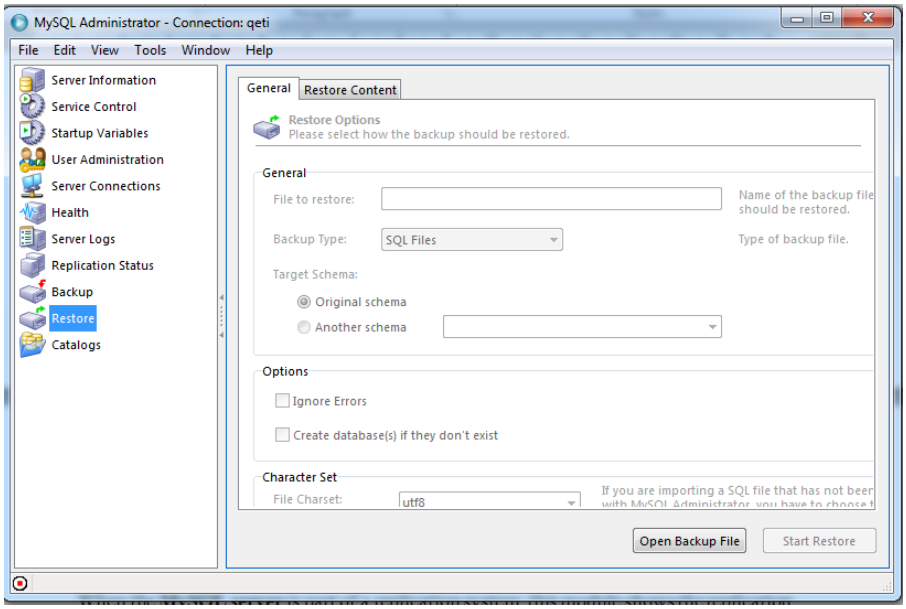
5) შესრულებაზე გაშვებამდე *SAVE PROJECT* -ით ვინახავთ დაყენებულ პარამეტრებს და ვაწკაპუნებთ "**Execute Backup Now**" -ზე.

მონაცემთა ბაზის აღდგენისათვის MySql Administrator უტილიტის გამოყენების შემთხვევაში სრულდება შემდეგი ბიჯები:

1) გავხსნათ MySql Administrator.

2) ვაწკაპუნებთ ღილაკზე "**Restore**".

ამ მოდულთან ერთად ჩვენ შეგვიძლია აღსადგენი მონაცემთა ბაზის წინასწარი ნახვა. ასლის ფაილის შერჩევის შემდეგ ჩანართში *RESTORE CONTENT* შეგვიძლია განვსზღვროთ მიზნობრივი ცხრილი, რომელშიც უნდა მოხდეს აღდგენა.



სურ. 10.2

შენიშვნა: ეს მოდული შეიძლება გამოყენებულ იქნას მხოლოდ MySQL Administrator-ში შექმნილი ასლის ფაილების შემთხვევაში და არა სხვა

ინსტრუმენტების მიერ შექმნილი ასლის ფაილების დროს (მაგალითად, phpMyAdmin, mysqldump და ა.შ.).

3) ვაწკაპუნებთ "**Open Backup File**"-ზე.

4) ვათვალიერებთ და განვსაზღვრავთ **Target Schema** დაწკაპუნებით ახალ სქემაზე ან არსებულ მონაცემთა ბაზაზე.

5) ვაწკაპუნებთ "**Start Restore**" - ზე.

არსებობს აღდგენის სხვა გზაც შემდეგი ბიჯების შესრულებით:

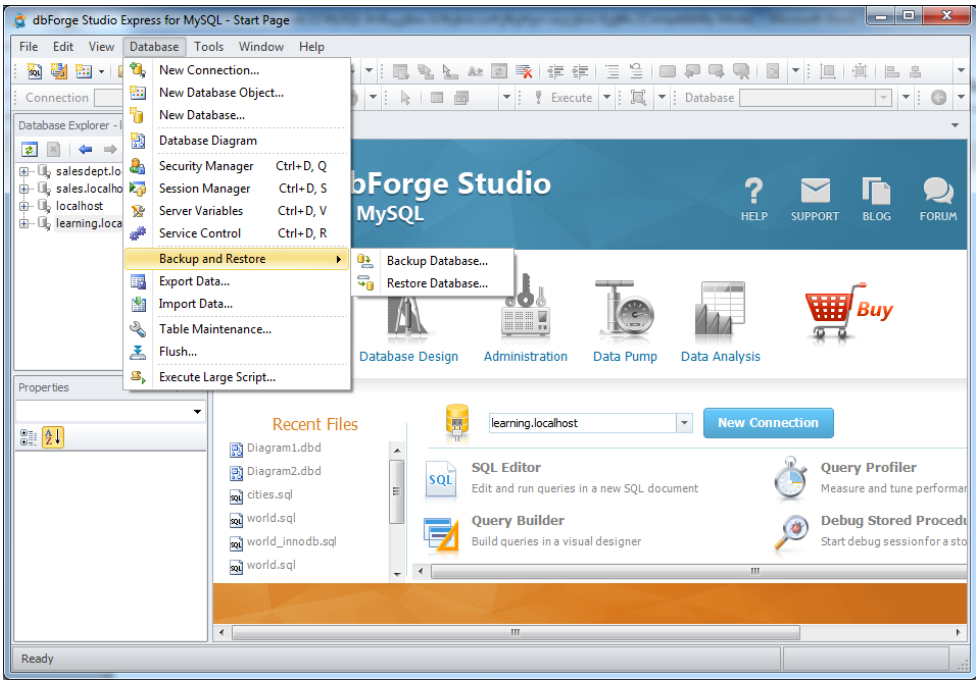
1) გავხსნათ **MySql Query Browser**.

2) მენიუში ვაწკაპუნებთ File -> ვხსნით script – > გადავხედავთ ასლის ფაილს.

3) ვაწკაპუნებთ Execute -ზე.

სარეზერვო ასლების შექმნა და აღდგენა dbForge Studio for MySQL - ის გამოყენებით

Database Backup Wizard გამოძახებისათვის **Database** მენიუდან ვირჩევთ **Backup Database.**



სურ. 10.3

მთლიანი მონაცემთა ბაზის სარეზერვო ასლის შექმნა

1. ვირჩევთ მონაცემთა ბაზას, ვწერთ ფაილის სახელს და ადგილსამყოფელს (location). ვაჭერთ ღილაკს **Next**.

2. ვირჩევთ **Structure** და **Data**, ვამოწმებთ **Include all objects**, ვაჭერთ **Next** ღილაკს.

3. ვაყენებთ ოპციებს და ვაჭერთ ღილაკს **Backup**.

4. ვირჩევთ ნებისმიერ script ფაილს SQL Editor - ში გასახსნელად და ვინახავთ ასლირების პროექტს სამომავლოდ. ვაჭერთ ღილაკს **Finish**.

გარკვეული მონაცემთა ბაზის სარეზერვო ასლის შექმნა

1. ვირჩევთ მონაცემთა ბაზას, ვწერთ ფაილის სახელს და ადგილსამყოფელს (location). ვაჭერთ ღილაკს **Next**.

2. ვირჩევთ საჭირო ინფორმაციას.

3.ვირჩევთ მონაცემთა ბაზის ობიექტებს მათი სტრუქტურის ასლისს შესაქმნელად, თუ არჩეული გვაქვს **Structure** ან **Data**. ვაჭერთ Next ღილაკს.

4. ვირჩევთ ცხრილებს. ვაჭერთ ღილაკს **Next**.

5.ვირჩევთ ოპციებს. ვაჭერთ ღილაკს **Backup**.

6. ვირჩევთ ნებისმიერ script ფაილს SQL Editor -ში გასახსნელად და ვინახავთ ასლირების პროექტს სამომავლოდ. ვაჭერთ ღილაკს **Finish**.

მონაცემთა ბაზის აღდგენა Backup Script -დან.

1. **Database** მენიუდან ვირჩევთ **Restore Database.**

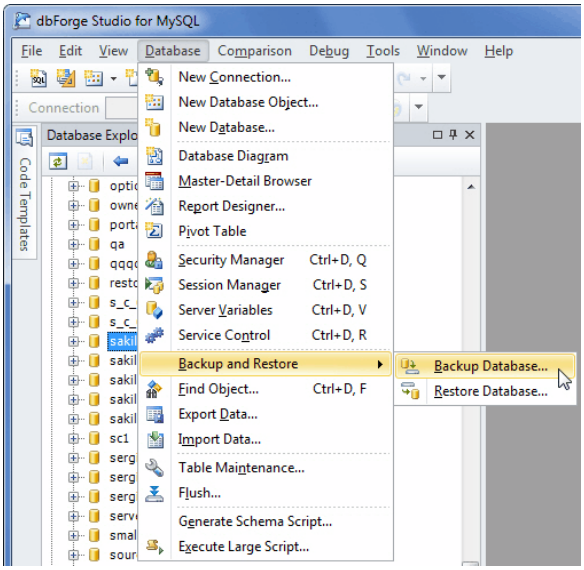
2. ვირჩევთ Backup Script -ს და მონაცემთა ბაზას აღდგენისათვის. თუ ფაილი შეიცავს ბრძანებას CREATE DATABASE, მაშინ მონაცემები აღდგება შექმნილ მონაცემთა ბაზაში ნაცვლად შერჩეული მონაცემთა ბაზისა.

3. შევამოწმოთ ფაილის შერჩევის სისწორე. ამიტომ შერჩევა ყოველთვის უნდა მოხდეს ჩამოშლადი სიიდან.

4. ვაჭერთ ღილაკს **Restore**.

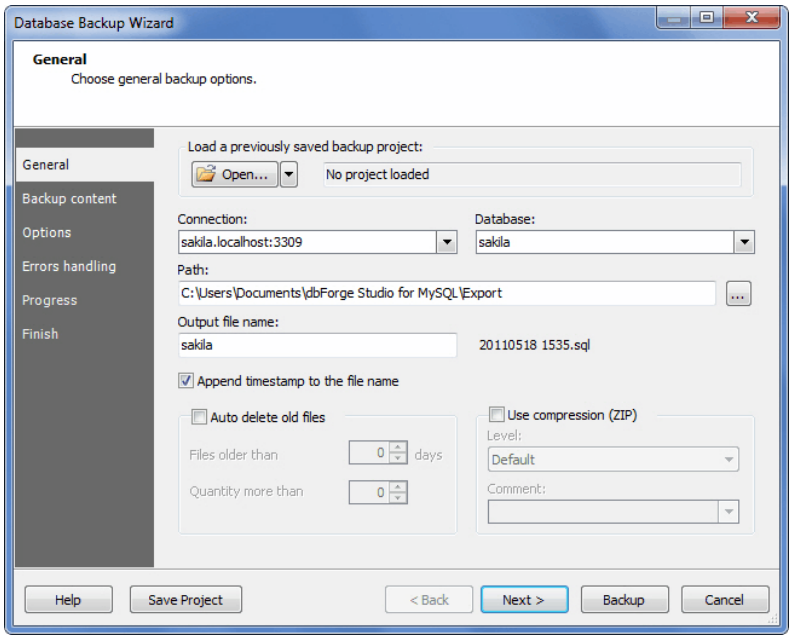
პროექტის ფაილის შექმნა

*1. **Database** მენიუში ვირჩევთ **Backup and Restore** და შემდეგ ვაწკაპუნებთ **Backup Database**.*



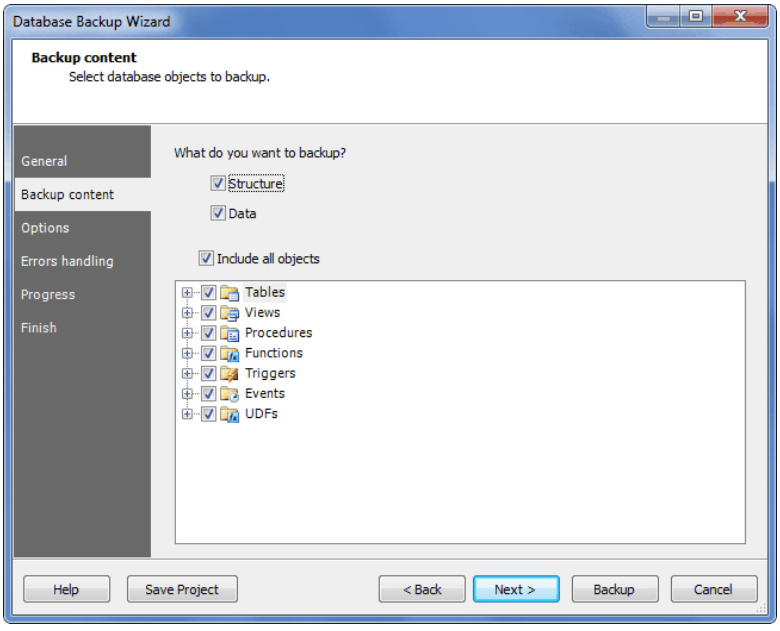
სურ. 10.4

2. განვსაზღვრავთ მიერთებას, მონაცემთა ბაზას, გზას, მიღებული ფაილის სახელს. ვაწკაპუნებთ **Next**.



სურ. 10.5

3. ვირჩევთ ობიექტების ჩამონათვალს. ვაწკაპუნებთ **Next**.



სურ. 10.6

4. ვირჩევთ ოპციებს. ვაწკაპუნებთ **Next**.

5. ვაყენებთ Errors მართვისა და Log ოპციებს. ვაწკაპუნებთ **Save Project**.

ლაბორატორიული სამუშაო 11

MySQL მონაცემთა ბაზების იმპორტი და ექსპორტი

სამუშაოს მიზანი:

1. ექსპორტ/იმპორტი ბრძანებითი სტრიქონის გამოყენებით
2. ექსპორტ/იმპორტი Workbench-ის გამოყენებით
3. ექსპორტ/იმპორტი phpMyAdmin - ის გამოყენებით
4. ექსპორტ/იმპორტი dbForge Studio - ს გამოყენებით
5. მონაცემთა ექსპორტი ბრძანებითი სტრიქონის მეშვეობით
6. მონაცემთა იმპორტი
7. MS Excel Import
8. Text Import
9. XML Import
10. Importing Data to a New Table
11. Migrating Data from Other Servers
12. Connector/ODBC -ის Microsoft Access -თან გამოყენება
13. MySQL-ის ინტერფეისი Microsoft Access გამოყენებით
14. MYSQL მონაცემთა ბაზის ფაილის ადგილმდებარეობის განსაზღვრა
15. მონაცემთა ბაზის გადატანა ერთი კომპიუტერიდან მეორეზე

1.ექსპორტ/იმპორტი ბრძანებითი სტრიქონის გამოყენებით

ექსპორტი. ამ მაგალითში ნაჩვენებია 'mytestdb' მონაცემთა ბაზის ექსპორტი ფაილში სახელწოდებით 'mytestdb.sql':

```
mysqldump -u root -p mytestdb > mytestdb.sql
```

იმპორტი. მონაცემთა იმპორტი მოიცავს ორ ბიჯს.

პირველი ბიჯი მდგომარეობს ცარიელი მონაცემთა ბაზის შექმნაში, რომელიც მზად იქნება მონაცემების მისაღებად:

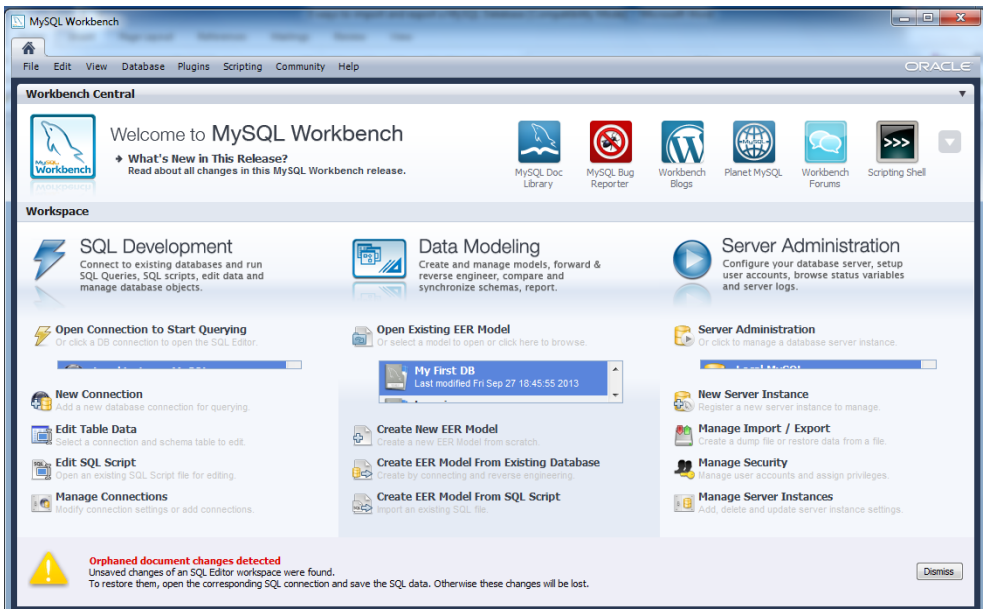
```
mysqladmin -u root -p create mytestdb2
```

მეორე ბიჯი იქნება მონაცემთა იმპორტის განხორციელება:

```
mysql -u root -p mytestdb2 < mytestdb.sql
```

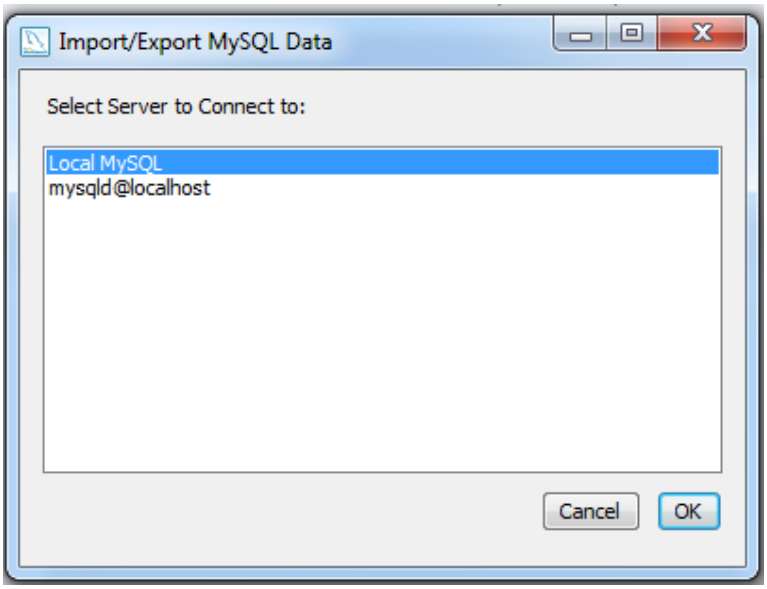
2. ექსპორტ/იმპორტი Workbench-ის გამოყენებით

Workbench-ის მთავარ ფანჯრის Server Administration-ში ვირჩევთ Manage Import/Export -ს.



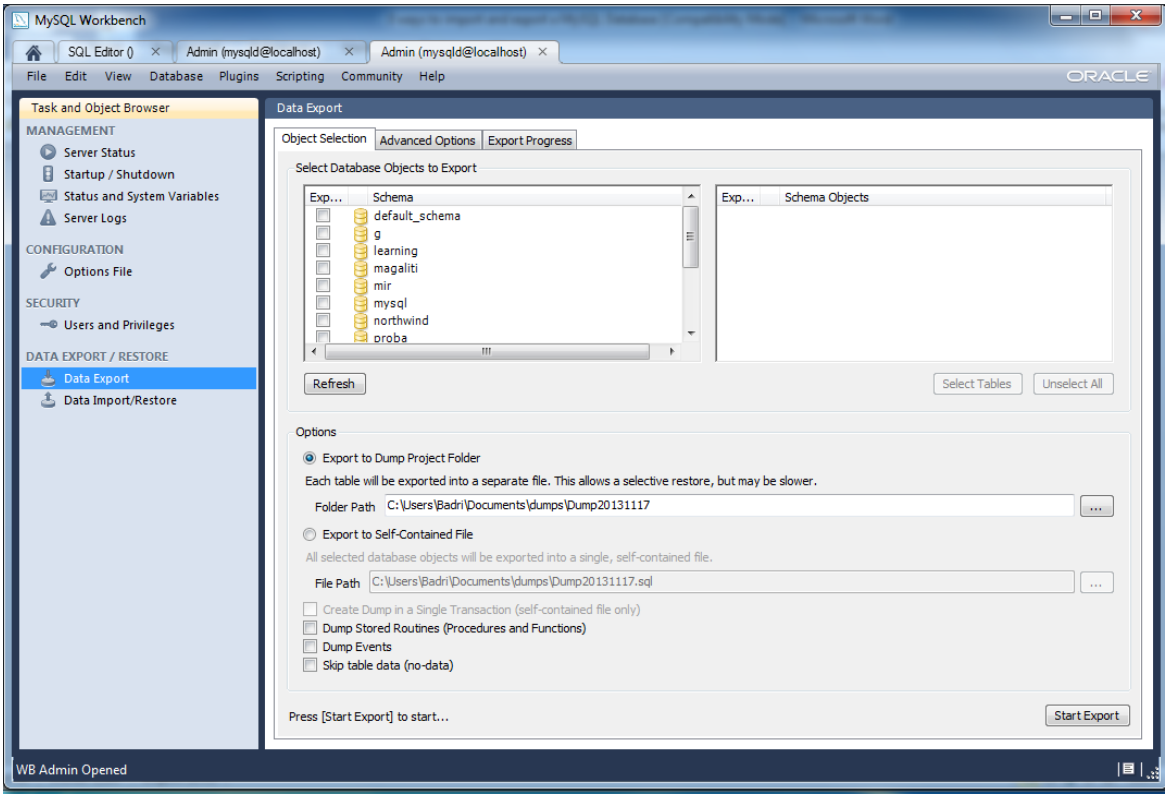
სურ. 11.1

გამოდის Import/Export MySQL Data ფანჯარა, სადაც ვირჩევთ Server Admin-ის მიერთების სერვერს.



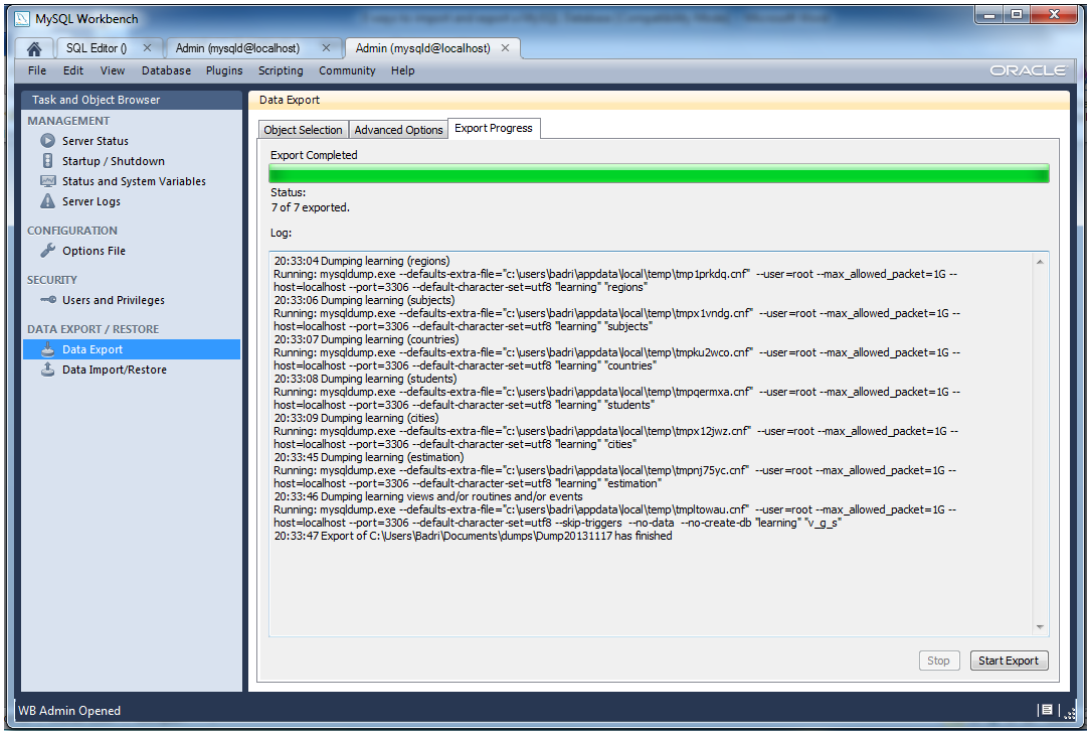
სურ. 11.2

Task and Object Browser არეში Data Export/Restore განყოფილებაში ვირჩევთ Data Export-ს. ვირჩევთ მონაცემთა ბაზას (ბაზებს) Select Dabase Objects to Export არეში და ვაჭერთ ღილაკს Start Export.



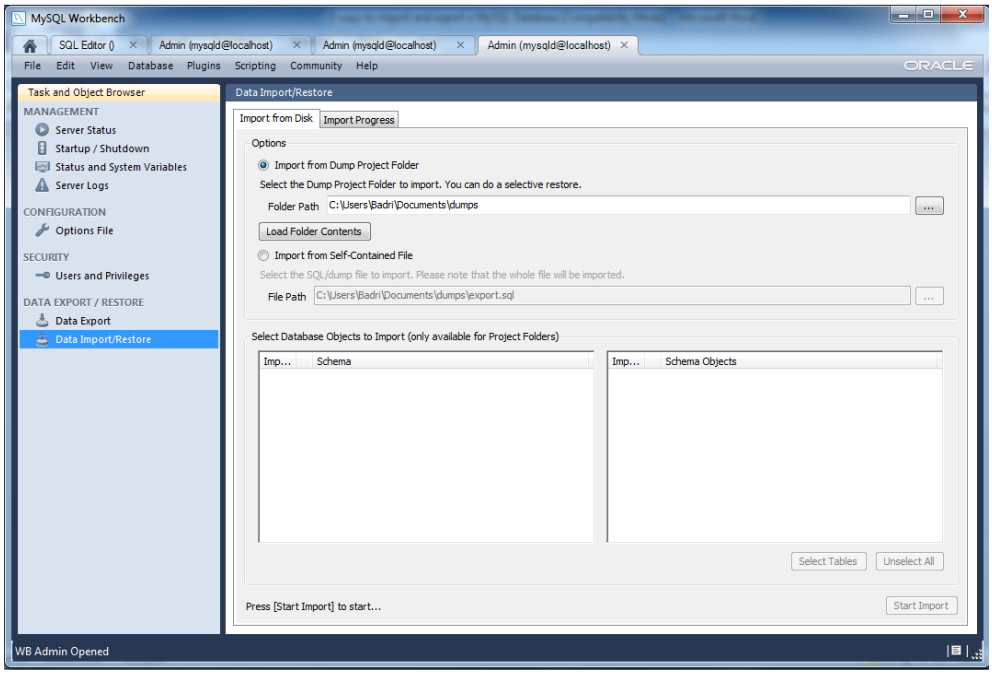
სურ. 11.3

შედეგად მონაცემთა ექსპორტის Dumping პროცესი სრულდება.



სურ. 11.4

იმპორტის შემთხვევაში Task and Object Browser არეში Data Export/Restore განყოფილებაში ვირჩევთ Data Import/Restore –ს.



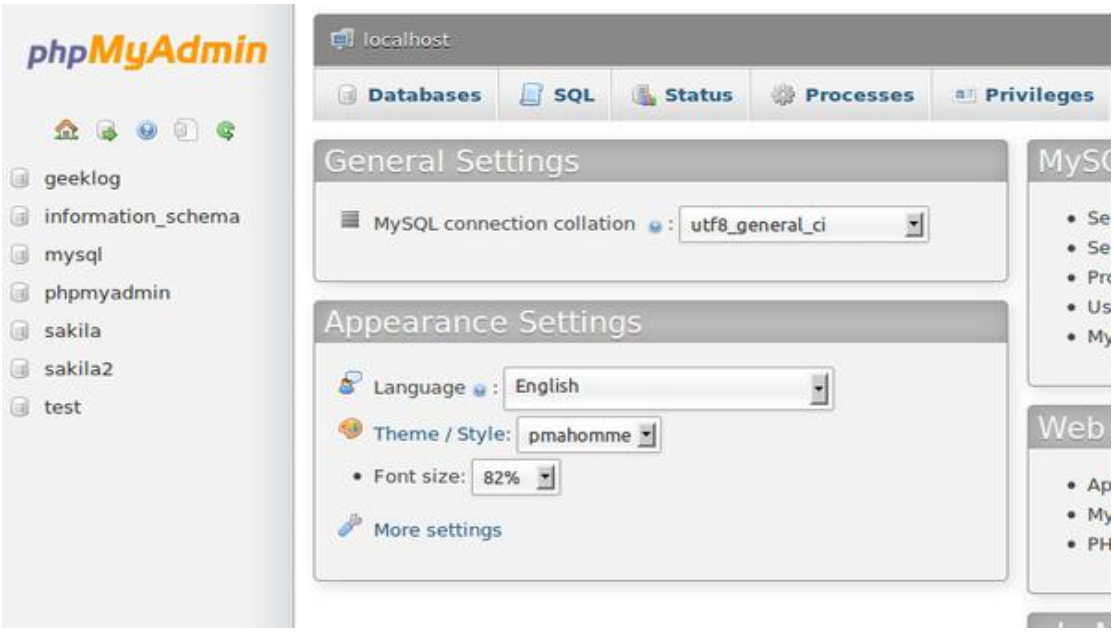
სურ. 11.5

ჩანართში Import From Disk ოპციებში (Options) ვირჩევთ იმპორტის წყაროს ანუ საიდან მოხდება იმპორტირება. შემდეგ ვადგენთ მონაცემთა ბაზას და Import Progress ჩანართში ვაჭერთ Start Import ღილაკს.

ეს მეთოდი ბევრად უფრო კარგად მუშაობს ვიდრე phpMyAdmin დიდი მონაცემთა ბაზების შემთხვევაში.

3. ექსპორტ/იმპორტი phpMyAdmin - ის გამოყენებით

მარცხენა სვეტში ვირჩევთ მონაცემთა ბაზას.



სურ. 11.6

ვაწკაპუნებთ Export ლინკზე და მონაცემთა ბაზას ვინახავთ ფაილში.

შემდეგ ახალ სერვერში მარცხენა სვეტში ვირჩევთ მონაცემთა ბაზას, ვაწკაპუნებთ Import ლინკზე და ვირჩევთ ფაილს, რომელშიც მოხდა ექსპორტირება.

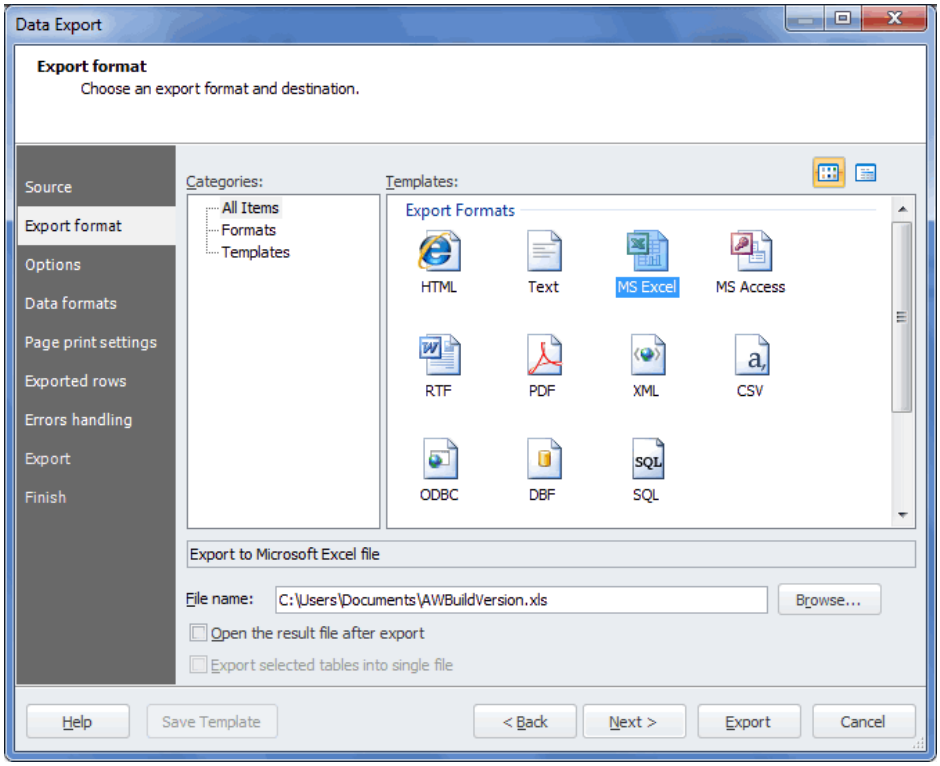
4. ექსპორტ/იმპორტი dbForge Studio - ს გამოყენებით

მონაცემთა ექსპორტი სხვადასხვა ფორმატში.

dbForge Studio -ს გამოყენებით შესაძლებელია მონაცემთა ექსპორტი სხვადასხვა ფორმატში.

1. გავხსნათ **Data Export** ოსტატი ქვემოთ მოყვანილი ქმედებიდან ერთ-ერთის შესრულებით:

- გვერდზე **Start** ვაწკაპუნებთ **Data Analysis** და შემდეგ ვაწკაპუნებთ **Export Data**.
- **Database Explorer**-ში საჭირო ცხრილზე ან წარმოდგენაზე მარჯვენა დაწკაპუნებით ვხსნით მენიუს, სადაც ვირჩევთ **Export Data**.
- მონაცემებზე მარჯვენა დაწკაპუნების შემდეგ მენიუდან ვირჩევთ **Export Data** ან **Data** ინსტრუმენტების პანელზე ვაჭერთ ღილაკზე **Export Data**.
- **Database** მენიუში ვაწკაპუნებთ **Export Data**.



სურ. 11.7

2. **Export** გვერდზე ვირჩევთ ექსპორტის ფორმატს. ვაწკაპუნებთ ღილაკზე **Next**.
3. ვირჩევთ ან შაბლონური ფაილიდან ჩავტვირთავთ ოპციებს.
4. **Options** ოსტატის გვერდზე ვაყენებთ დამატებით პარამეტრებს. ვაჭერთ **Next**.

5. ვირჩევთ საექსპორტო სვეტებს და ვაჭერთ **Next**.

6. ვირჩევთ საექსპორტო სტრიქონებს.

7. ვინახავთ ოპციებს შაბლონურ ფაილში, რითვისაც ვაწკაპუნებთ **Save Template** ღილაკზე და განვსაზღვრავთ შაბლონური ფაილის გზას და სახელს.

8. ვაჭერთ **Export** ღილაკზე და ვაკვირდებით ექსპორტის პროცესს.

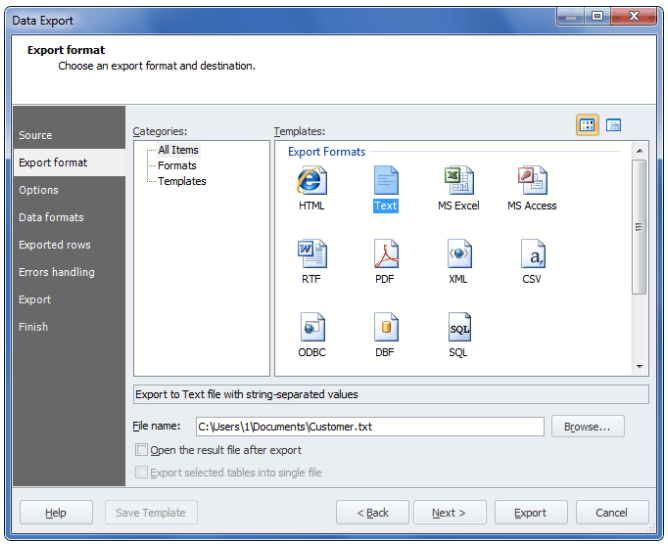
9. გახსნისათვის ვირჩევთ ექსპორტირებულ ფაილს და ვაჭერთ ღილაკზე **Finish**.

შაბლონების შენახვა და გამოყენება ექსპორტის განმავლობაში

შაბლონის შექმნა

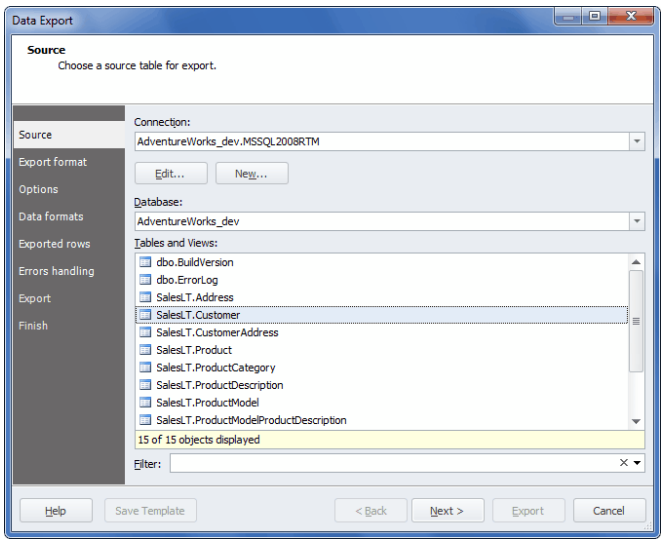
1. **Data Export** ოსტატში ვირჩევთ **Export** ფორმატის ჩანართს და შემდეგ ვირჩევთ **Text**.

2. განვსაზღვროთ დანიშნულების ფაილის გზას და სახელს. ვაწკაპუნებთ **Next**.



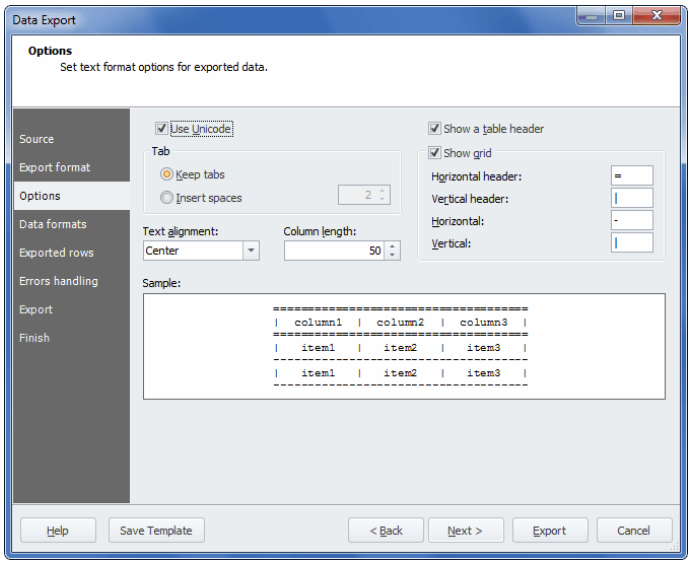
სურ. 11.8

3.ავირჩიოთ საჭირო MySQL სერვერის მიერთებები, მონაცემთა ბაზა და ცხრილი, რომლის ექსპორტი გვინდა.



სურ. 11.9

4. ჩანართში **Options** ვაყენებთ ტექსტის საზღვრებს **Left** და აგრეთვე სვეტის სიგრძეს 60. ვაწკაპუნებთ **Next**.



სურ. 11.10

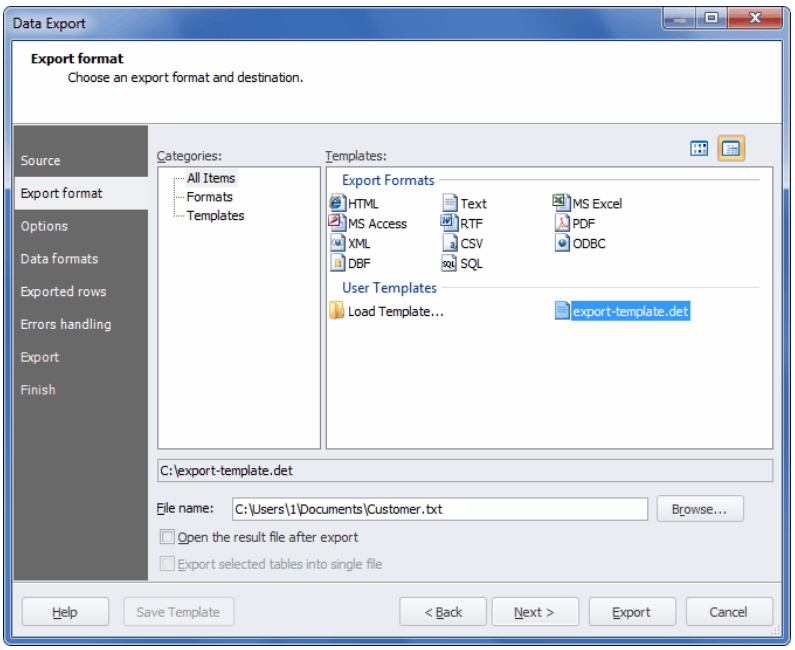
5. ვაწკაპუნებთ **Export** მონაცემთა ექსპორტის დაწყებისათვის.

6. ვინახავთ ყველა შერჩეულ პარამეტრებს, რისთვისაც ვაჭერთ **Save Template** ღილაკს**.** ფანჯარაში **Save As** განვსაზღვრავთ შაბლონური ფაილის სახელს და ადგილს.

შაბლონის ჩატვირთვა

1. გავხსნათ **Data Export** ოსტატი.

2. სექციაში **Templates** ვაწკაპუნებთ export-template.det-ზე, რომელიც შეიქმნა ხსენებული პროცედურის შედეგად.



სურ. 11.11

1. განვსაზღვრავთ დანიშნულების ფაილის გზას და სახელს.

2. ჩანართში **Source** ვირჩევთ **Source** მიერთებას, მონაცემთა ბაზას და ცხრილს.

3. გადავდივართ ჩანართში **Options,** ვნახულობთ ტექსტის საზღვრებს **Left** და აგრეთვე სვეტის სიგრძეს 60.

5. მონაცემთა ექსპორტი ბრძანებითი სტრიქონის მეშვეობით

ჯერ შევქმნათ შაბლონური ფაილი:

1. **Data Export Wizard** გახსნისათვის **Database** მენიუში ვაწკაპუნებთ **Export Data**.

2. გავიაროთ ოსტატის ყველა გვერდი, განვსაზღვროთ საჭირო ინფორმაცია და პარამეტრი.

3. შემდეგ შენახვისათვის ოსტატის ყველა გვერდზე ვაჭერთ ღილაკს Save Template.

ბრძანებითი სტრიქონის მეშვეობით მონაცემთა ექსპორტისათვის:

1.ვაწკაპუნებთ **Start**.

2. **Search programs and files** ველში ვკრეფთ **cmd** და ვაჭერთ ღილაკს **Enter**.

3. ვკრეფთ dbforgemysql.com ფაილის გზას, ადგილს dbForge Studio for MySQL ინსტალაციაში და ვაჭერთ ინტერვალის კლავიშს:

C:\Users\user>"C:\Program Files\Devart\dbForge Studio for MySQL\dbforgemysql.com"

4. განვსაზღვროთ ოპერაცია (data export) და შაბლონური ფაილი:

/dataexport /templatefile:<filepath>

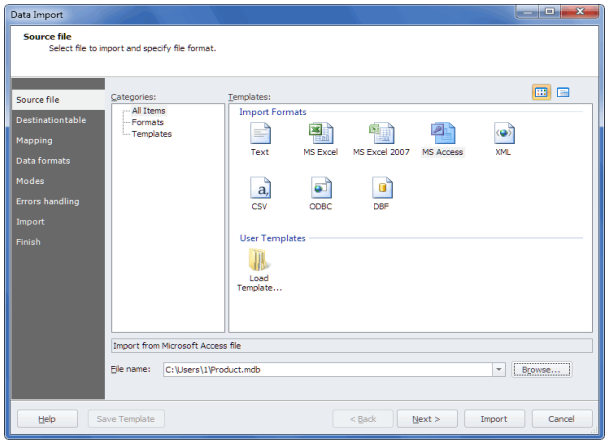
5. ავკრიფოთ ყველა სხვა არგუმენტი, რაც გვჭირდება.

/dataexport /?

6. ვაჭერთ **Enter** პროცესის შესრულებისათვის.

6. მონაცემთა იმპორტი

Data Import ოსტატის გამოყენებით მონაცემთა იმპორტისათვის საჭიროა ჯერ ავირჩიოთ ფორმატი.



სურ. 11.12

MS Access Import

1. ახალი ცხრილისათვის:

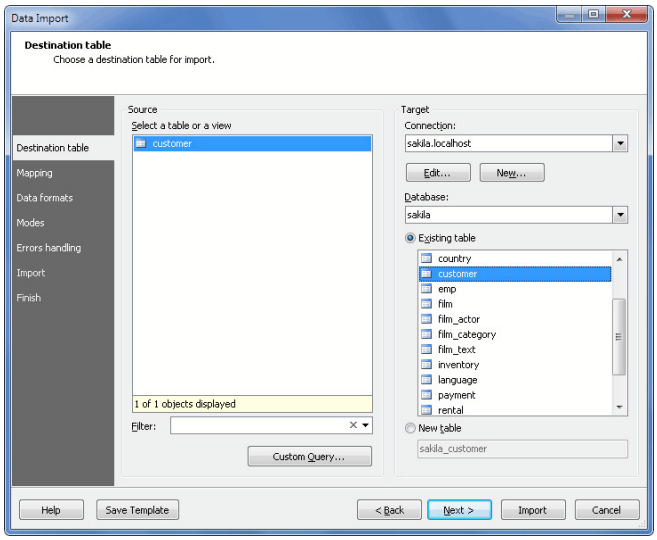
• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ **Access** იმპორტის ფორმატს და განვსაზღვრავთ მონაცემთა წყაროს ადგილს. ვაწკაპუნებთ **Next**.

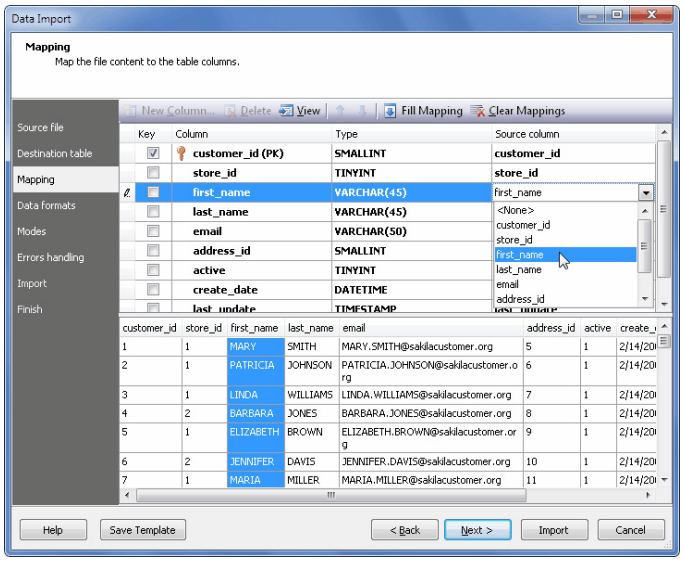
3. ვირჩევთ **Source** ცხრილს ან წარმოდგენას. ჩვენ შეგვიძლია გამოვიყენოთ მოთხოვნა ნაწილობრივი იმპორტისათვის. ვაწკაპუნებთ ღილაკზე **Custom Query** და შეგვაქვს ცვლილებები მოთხოვნაში. ცვლილებების შენახვისათვის ვაწკაპუნებთ **OK**.



სურ. 11.13

4. განვსაზღვრავთ MySQL მიერთებას, სქემას ანუ მონაცემთა ბაზას და ცხრილს იმპორტისათვის. ვაწკაპუნებთ **Next**.

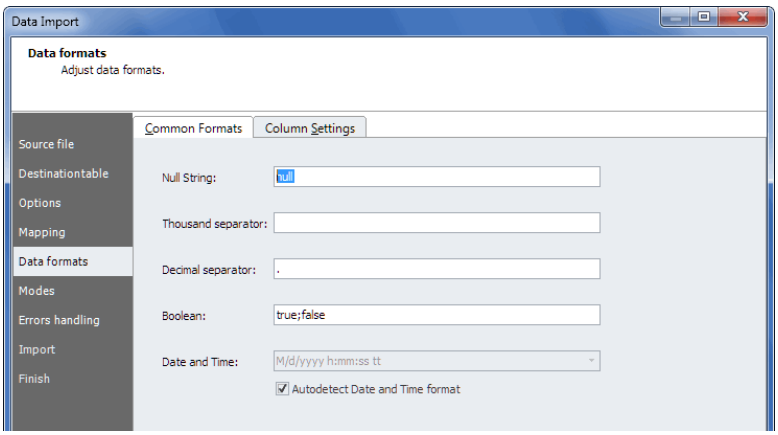
5. ჩამოშლად სიებში ვათვალიერებთ **Source** სვეტებს.



სურ. 11.14

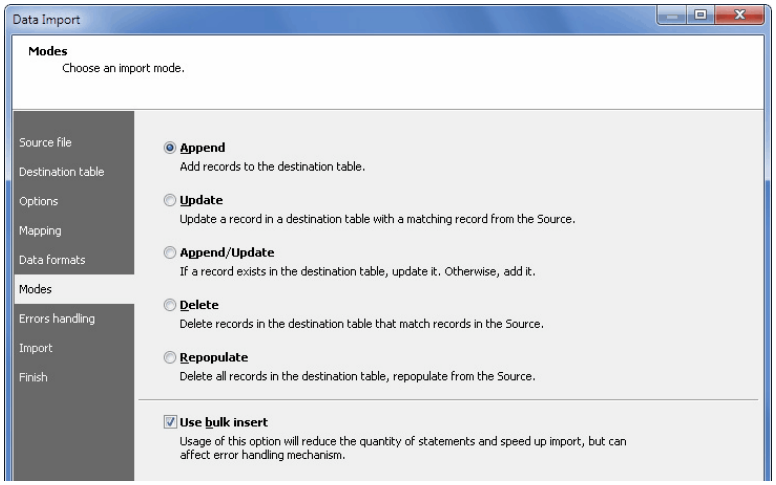
6. თუ ახალი ცხრილის იმპორტირება ხდება, მაშინ **Target** სვეტის თვისებების რედაქტირებისათვის ორჯერ ვაწკაპუნებთ. primary **Key** -სთან ერთად სვეტისათვის ვირჩევთ **Key** ალამს და ვაწკაპუნებთ **Next**.

7. განვსაზღვროთ მონაცემთა ფორმატი **Source** -სათვის და ვაწკაპუნებთ **Next**.



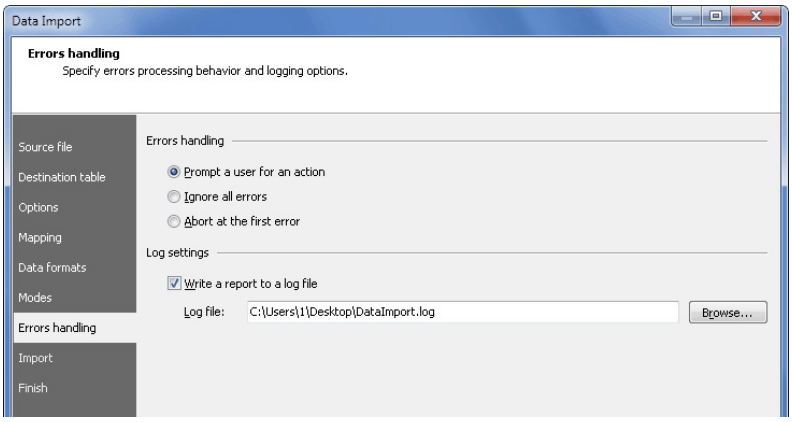
სურ. 11.15

8. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.16

9. ვირჩევთ შეცდომებზე რეაგირებას.



სურ. 11.17

10. ვაწკაპუნებთ **Import** და ვაკვირდებით იმპორტის მსვლელობას.
ვაწკაპუნებთ ღილაკზე **Show**log.
11. და ბოლოს, ვაწკაპუნებთ **Finish.**

CSV Import

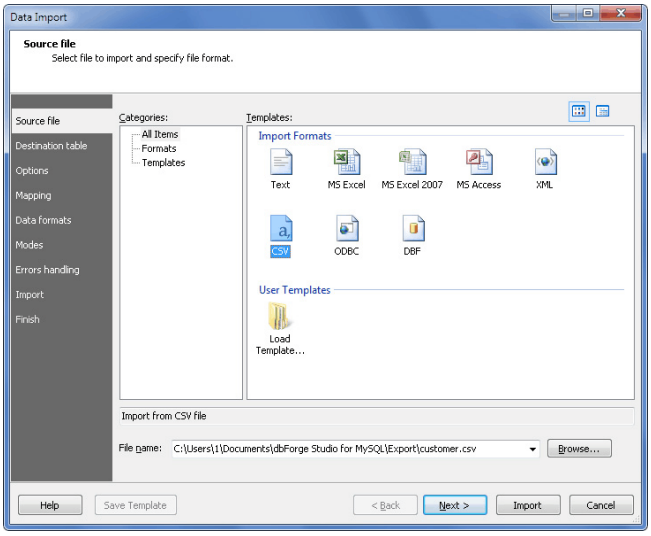
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

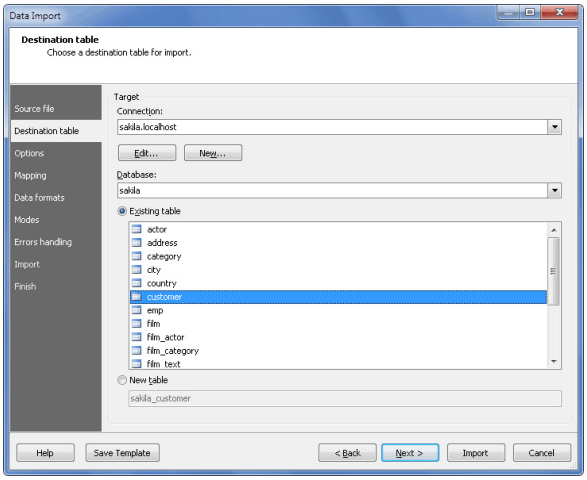
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ **CSV** იმპორტის ფორმატს და განვსაზღვრავთ **Source** data ადგილს. ვაწკაპუნებთ **Next**.



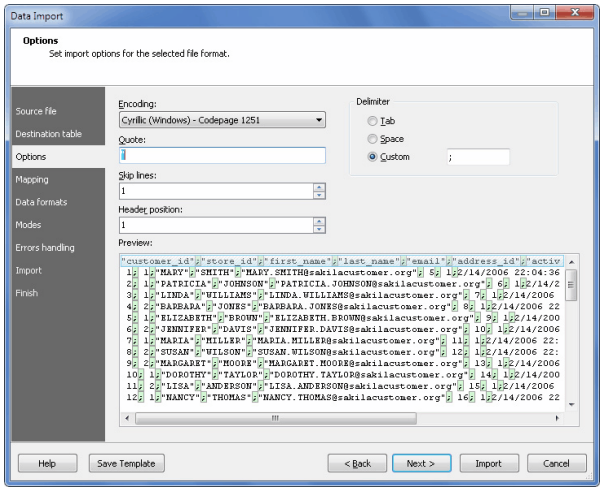
სურ. 11.18

3. განვსაზღვრავთ MySQL მიერთებას, სქემას ანუ მონაცემთა ბაზას და ცხრილს იმპორტისათვის. ვაწკაპუნებთ **Next**.



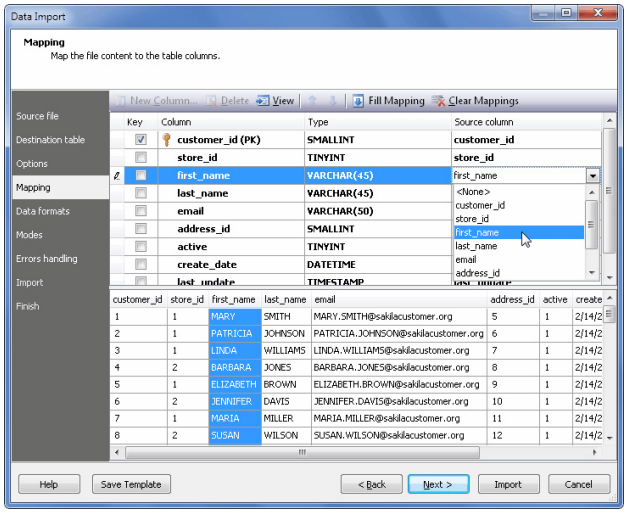
სურ. 11.19

4. ვაწკაპუნებთ **Next**.



სურ. 11.20

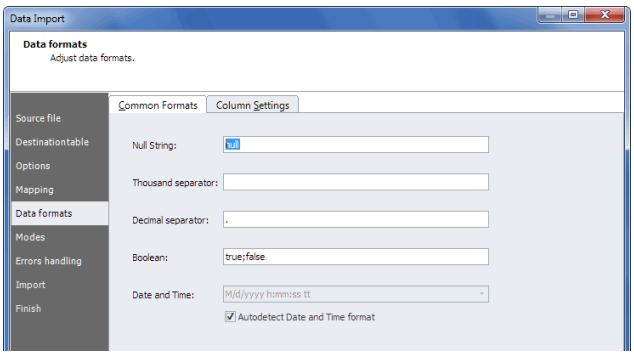
5. ჩამოშლადი სიიდან ვათვალიერებთ **Source** სვეტებს.



სურ. 11.21

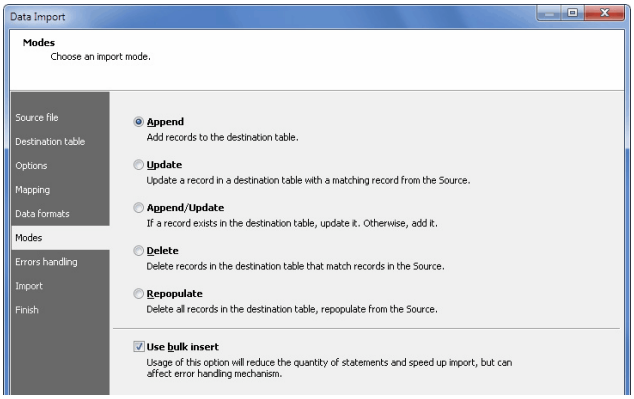
6. თუ იმპორტი ხდება ახალ ცხრილში, მაშინ შეგვიძლია სვეტის თვისებების რედაქტირება, რისთვისაც ორჯერ ვაწკაპუნებთ მათზე. ვირჩევთ **Key** ალამს სვეტისათვის **primarykKey** -სთან ერთად და ვაწკაპუნებთ **Next**.

7. განვსაზღვროთ მონაცემთა ფორმატი მონაცემთა წყაროსათვის და ვაწკაპუნებთ **Next**.



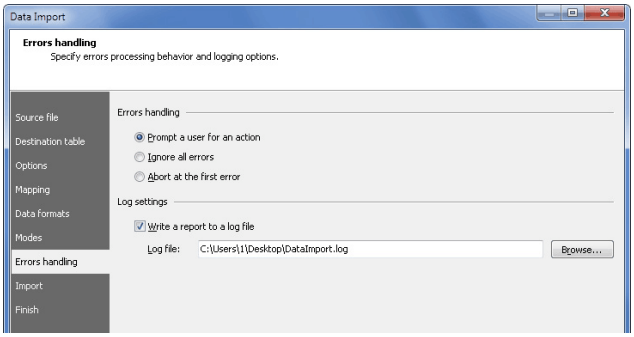
სურ. 11.22

8. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.23

9. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის განმავლობაში.



სურ. 11.24

10. ვაწკაპუნებთ **Import** და ვაკვირდებით იმპორტის მსვლელობას.

11. ვაწკაპუნებთ **Finish**.

DBF Import

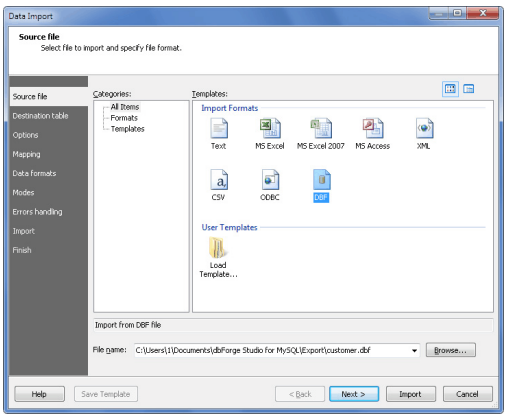
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

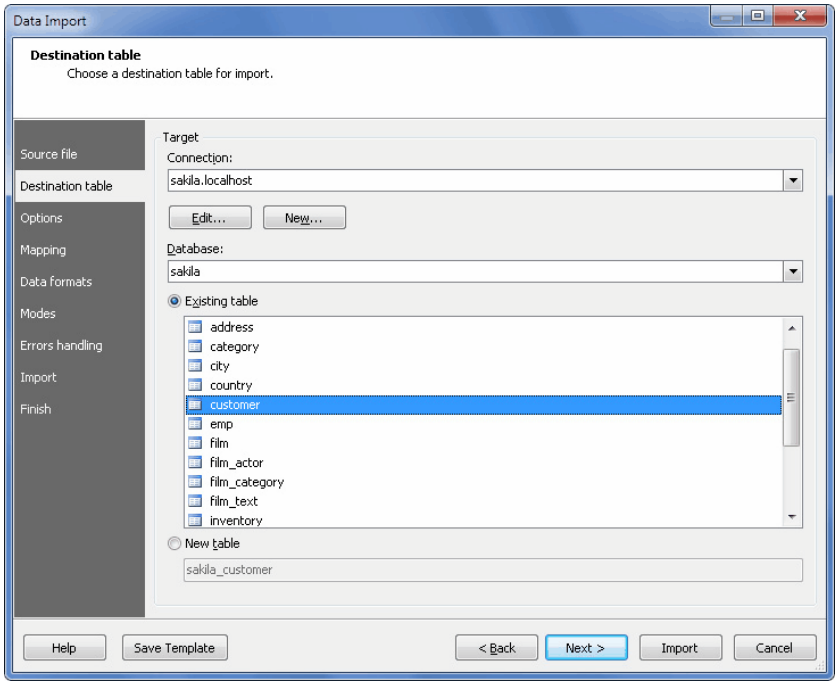
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ იმპორტის **DBF** ფორმატს და განვსაზღვროთ **Source** data ადგილი. ვაწკაპუნებთ **Next**.



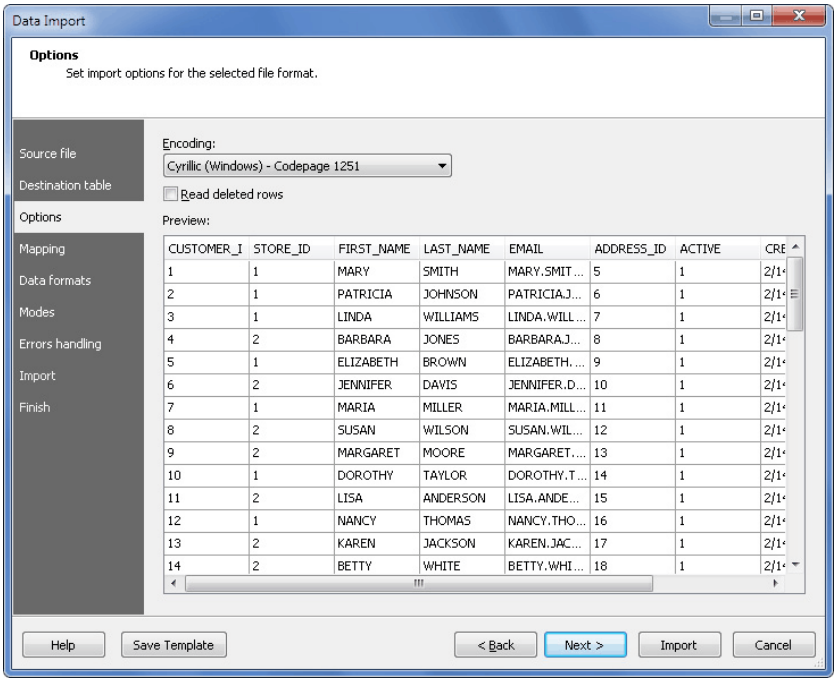
სურ. 11.25

3. განვსაზღვრავთ MySQL მიერთებას, სქემას ანუ მონაცემთა ბაზას და ცხრილს იმპორტისათვის. ვაწკაპუნებთ **Next**.



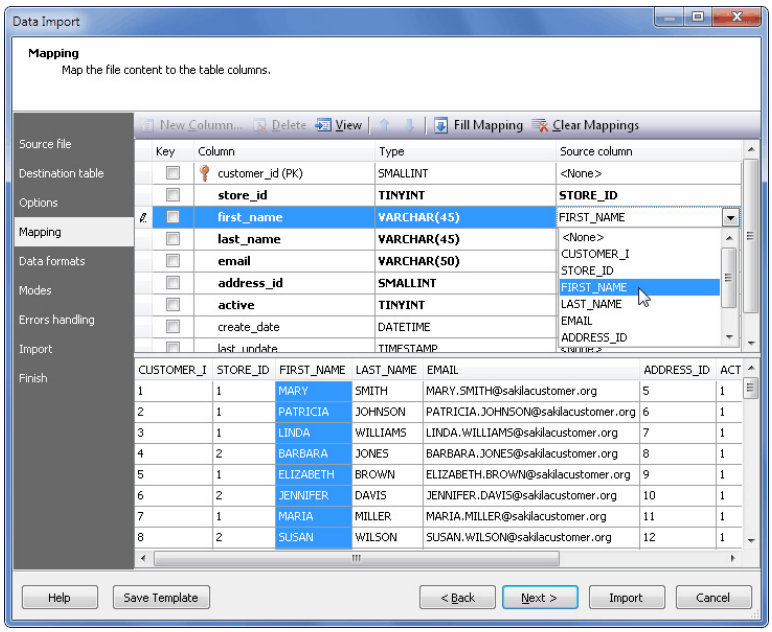
სურ. 11.26

4. ვაწკაპუნებთ **Next**.



სურ. 11.27

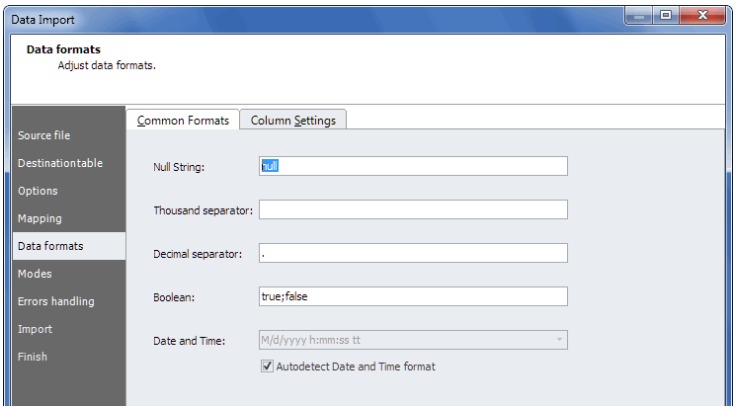
5. ვათვალიერებთ **Source** სვეტებს.



სურ. 11.28

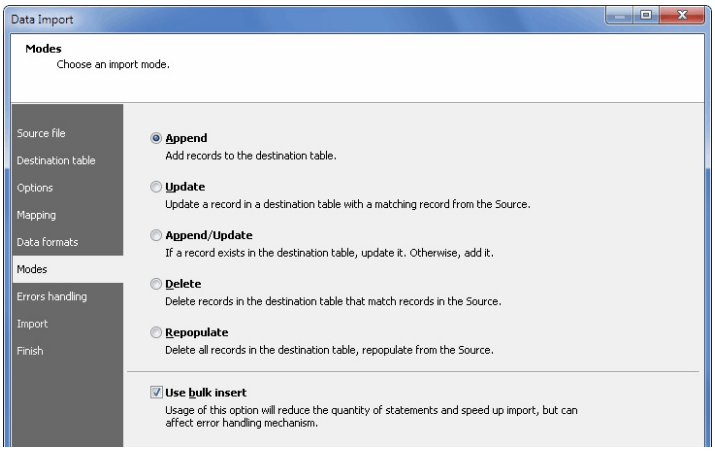
6. თუ იმპორტი ხდება ახალ ცხრილში, მაშინ შეგვიძლია სვეტის თვისებების რედაქტირება, რისთვისაც ორჯერ ვაწკაპუნებთ მათზე. ვირჩევთ **Key** ალამს სვეტისათვის **primarykKey** -სთან ერთად და ვაწკაპუნებთ **Next**.

7. **Source** data-სათვის განვსაზღვროთ ფორმატი და ვაწკაპუნებთ **Next**.



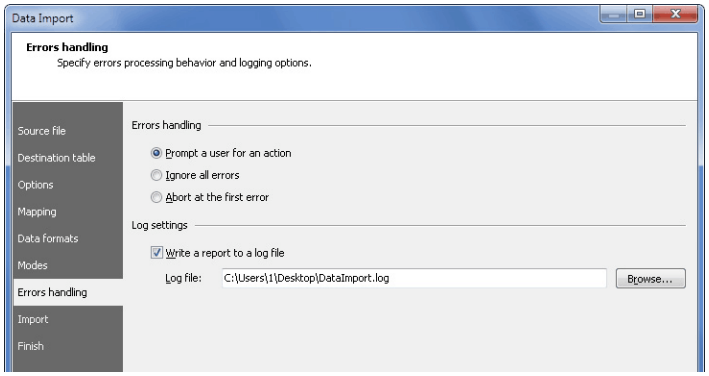
სურ. 11.29

8. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.30

9. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის განმავლობაში.



სურ. 11.31

10. ვაწკაპუნებთ **Import** და ვაკვირდებით იმპორტის მიმდინარეობას.

11. ვაწკაპუნებთ **Finish.**

7. MS Excel იმპორტი

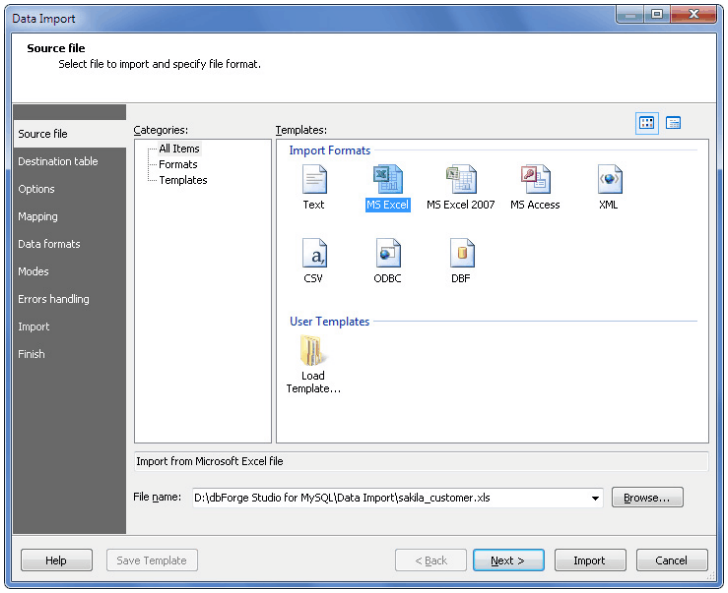
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

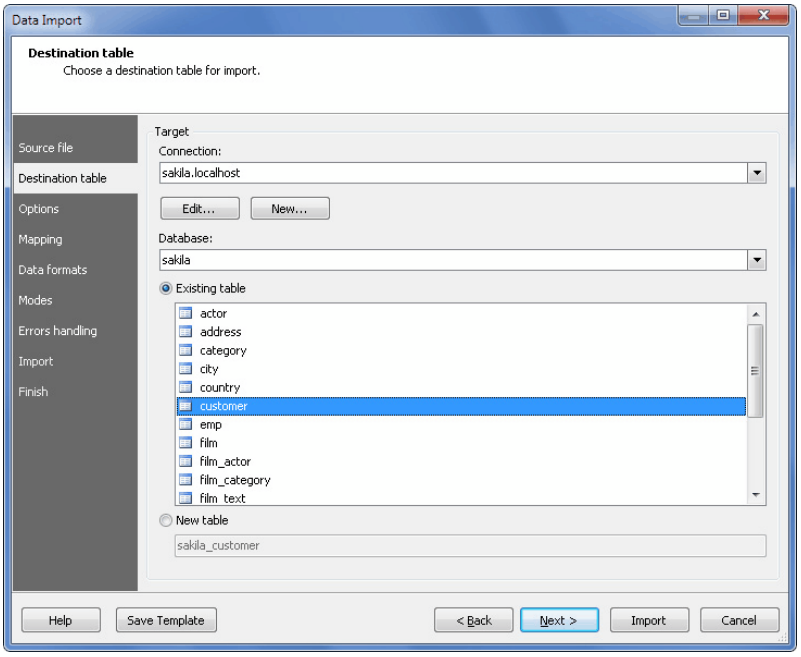
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ **Excel (for Excel 97-2003)** ან **Excel 2007** იმპორტის ფორმატს და განვსაზღვრავთ **Source** data -ს ადგილს. ვაწკაპუნებთ **Next**.



სურ. 11.32

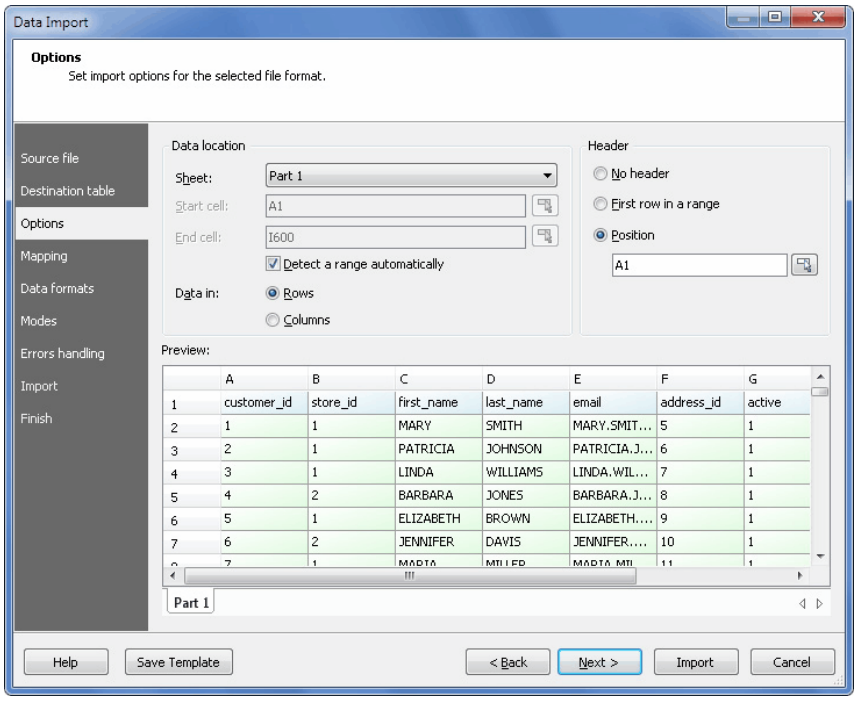
3. განვსაზღვრავთ MySQL მიერთებას, სქემას ანუ მონაცემთა ბაზას და ცხრილს იმპორტისათვის. ვაწკაპუნებთ **Next**.



სურ. 11.33

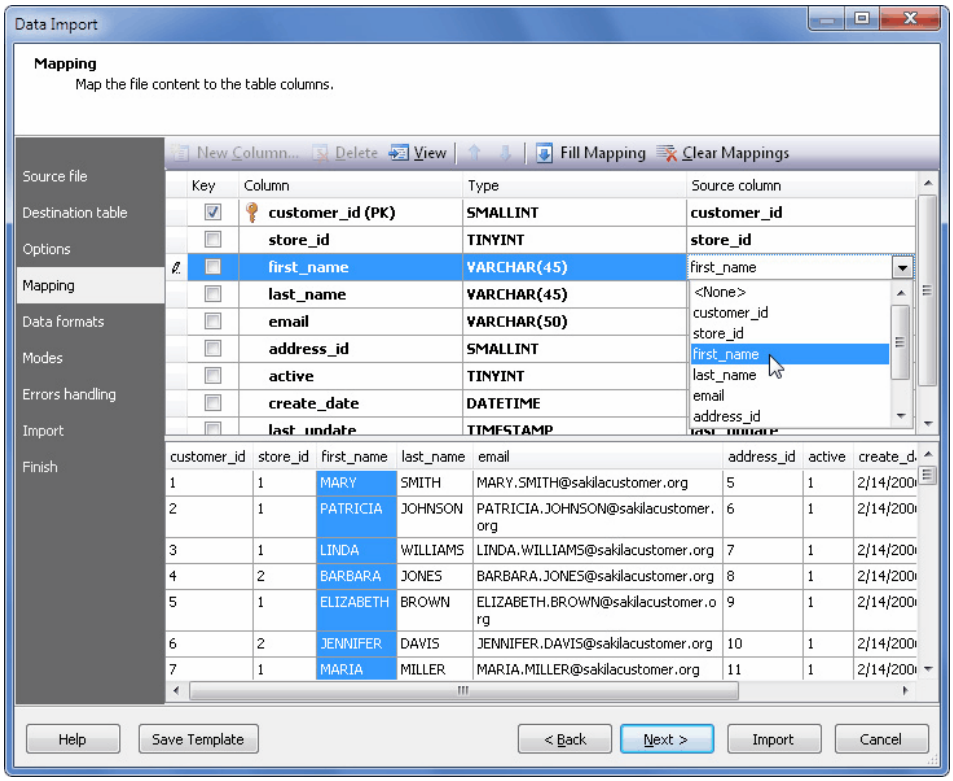
4. წინაწარ ვნახულობთ Source data-ს და განვსაზღვრავთ დამატებით ოპციებს იმპორტისათვის. ვაწკაპუნებთ Next.

.



სურ. 11.34

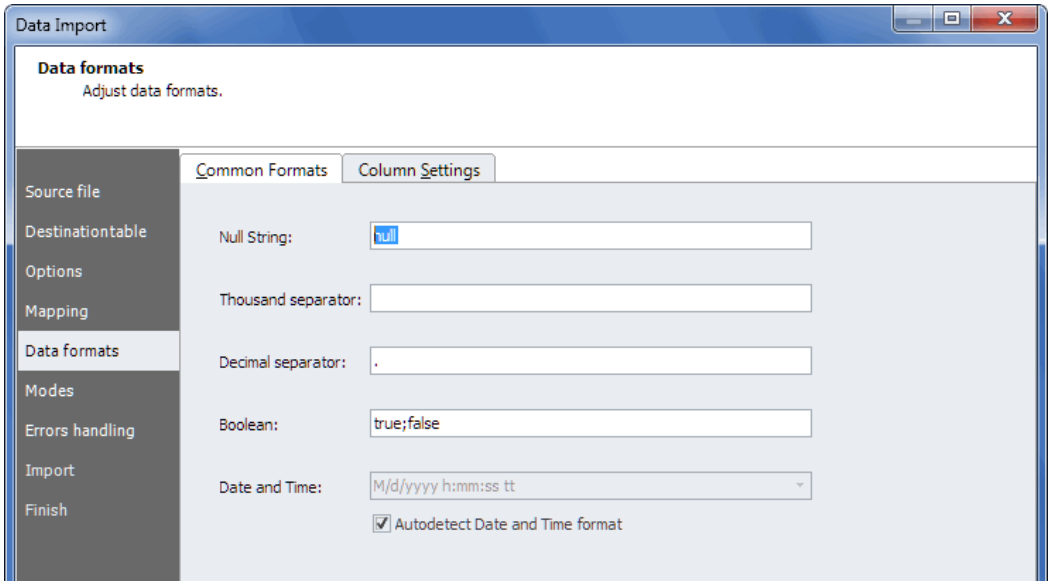
5. ჩამოშლადი სიიდან ვათვალიერებთ **Source** სვეტებს.



სურ. 11.35

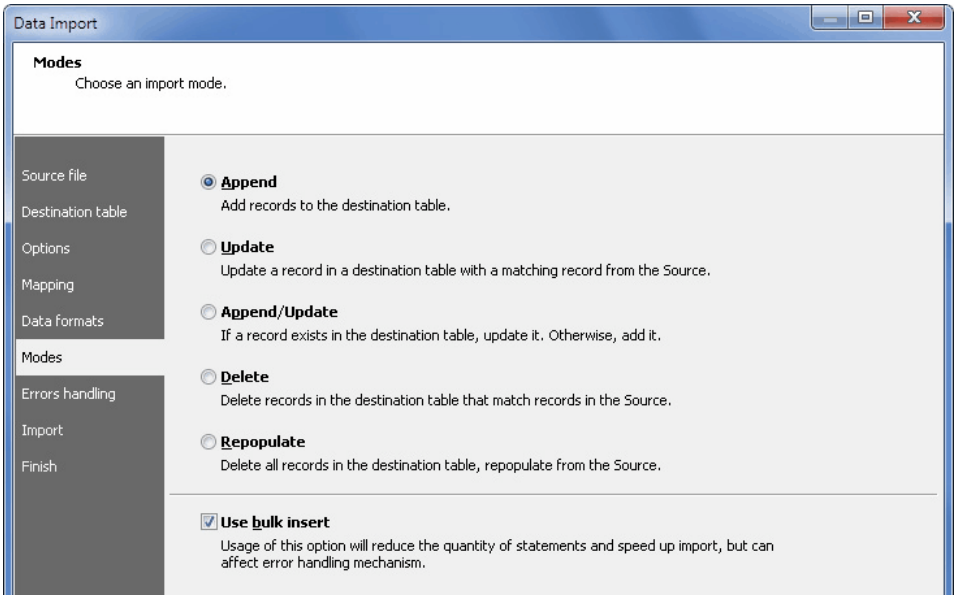
6. თუ იმპორტი ხდება ახალ ცხრილში, მაშინ შეგვიძლია სვეტის თვისებების რედაქტირება, რისთვისაც ორჯერ ვაწკაპუნებთ მათზე. ვირჩევთ **Key** ალამს სვეტისათვის **primarykKey** -სთან ერთად და ვაწკაპუნებთ **Next**.

7. განვსაზღვრავთ მონაცემთა ფორმატსSource data -სათვის და ვაწკაპუნებთ **Next**.



სურ. 11.36

8. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.37

9. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის განმავლობაში.

8. ტექსტის იმპორტი

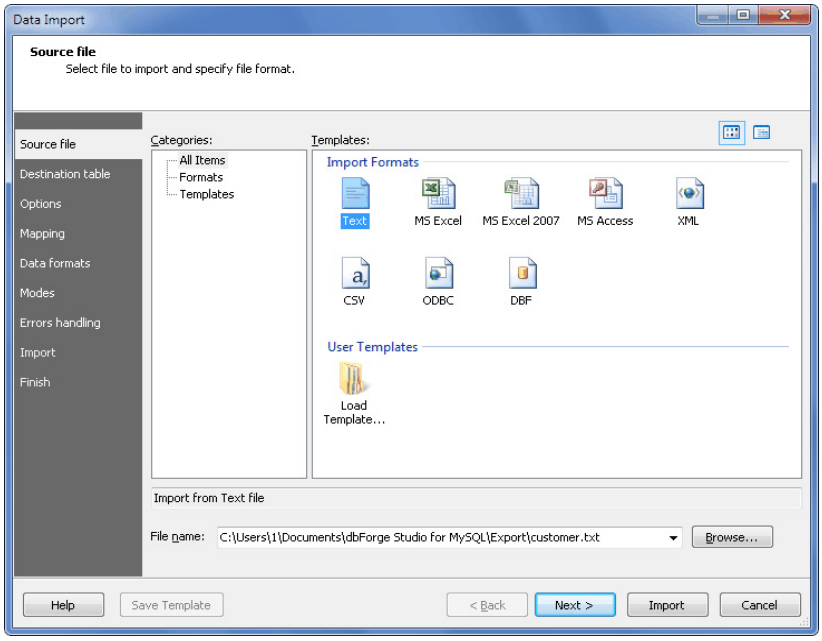
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

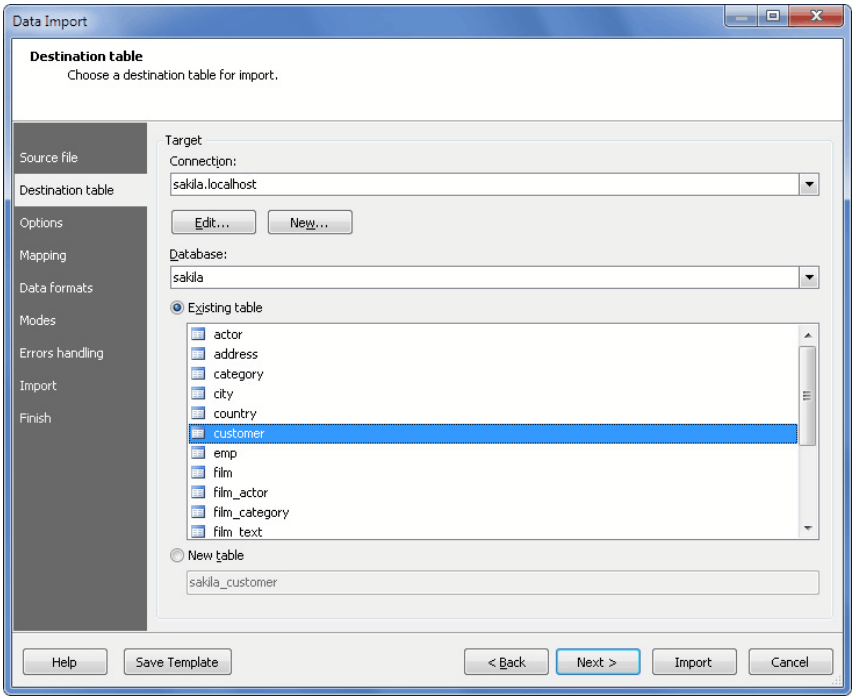
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ **Text** იმპორტის ფორმატს და განვსაზღვრავთ **Source** data ადგილს. ვაწკაპუნებთ **Next**.



სურ. 11.38

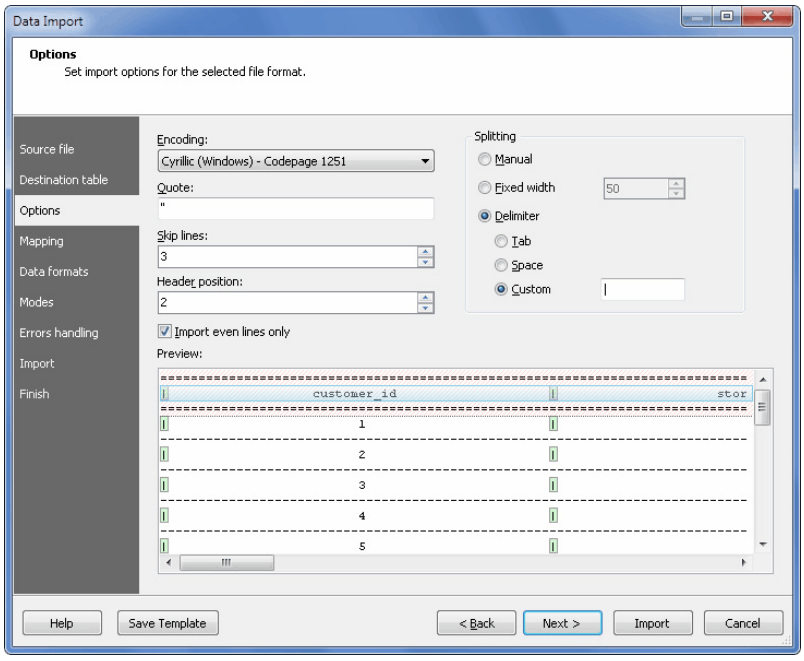
3. განვსაზღვროთ MySQL მიერთება, მონაცემთა ბაზა და ცხრილი, რომელშიც გვინდა იმპორტის განხორციელება. თუ ცხრილი შერჩეულია **Database Explorer** -ში **Data Import** ოსტატის გახსნის წინ, მაშინ ოსტატი გაიხსნება შერჩეული ცხრილის მიერთების პარამეტრებით. ვაწკაპუნებთ **Next**.



სურ. 11.39

4. ავირჩიოთ Source data -ს სვეტებში გაყოფის ხერხი:

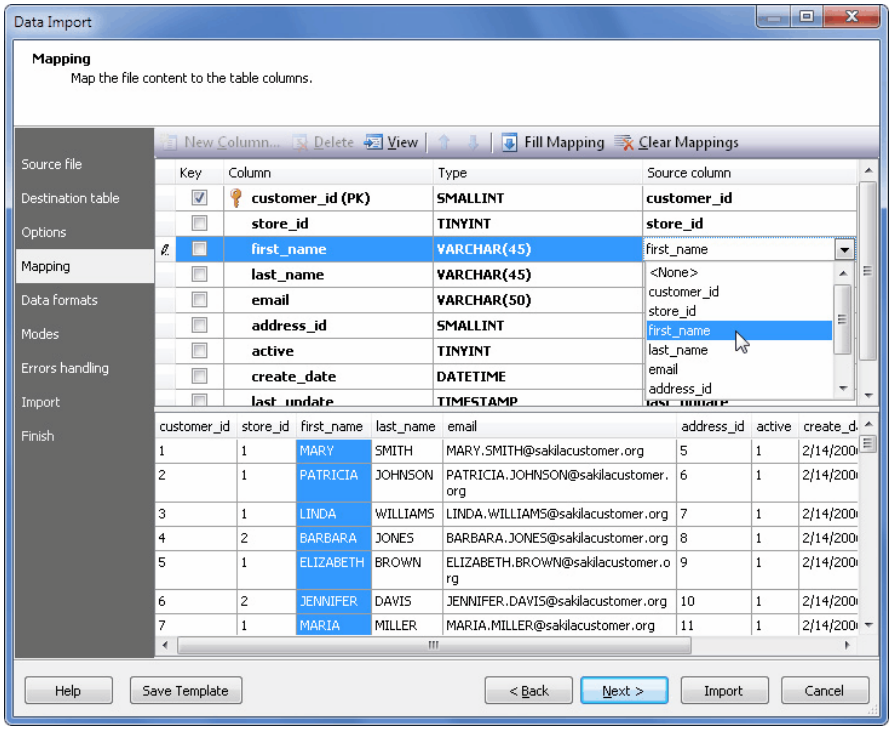
- Manual.
- Fixed width.
- Delimiter.



სურ. 11.40

5. განვსაზღვროთ იმპორტისათვის დამატებითი ოპციები. ვაწკაპუნებთ Next.

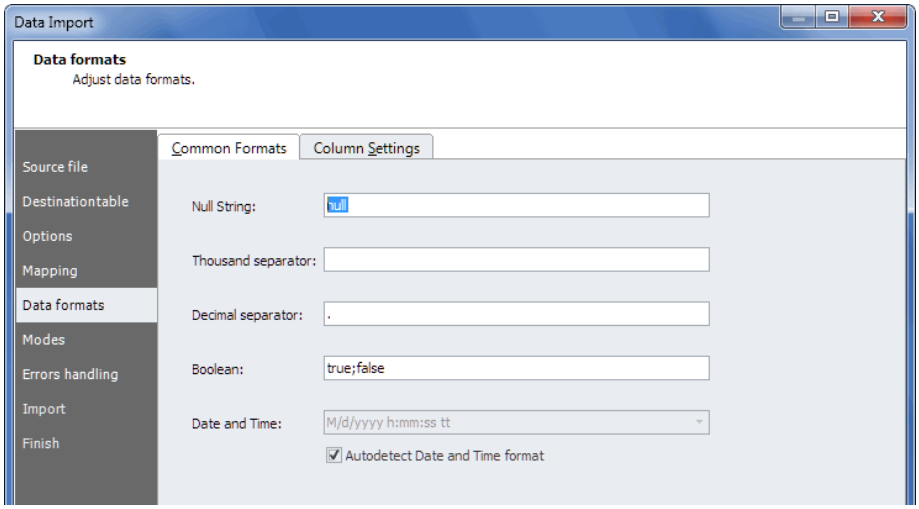
6. შევადგინოთ **Source** სვეტების განაწილება **Target** -სათვის. შევირჩიოთ **Target** სვეტი ცხრილის სათავეში და ვაჭერთ **Unmap** ღილაკს. ვადგენთ მას **Source** სვეტში, ვაწკაპუნებთ საჭიროებისამებრ ცხრილის ქვედა ნაწილში და ვაჭერთ **Map** ღილაკს.



სურ. 11.41

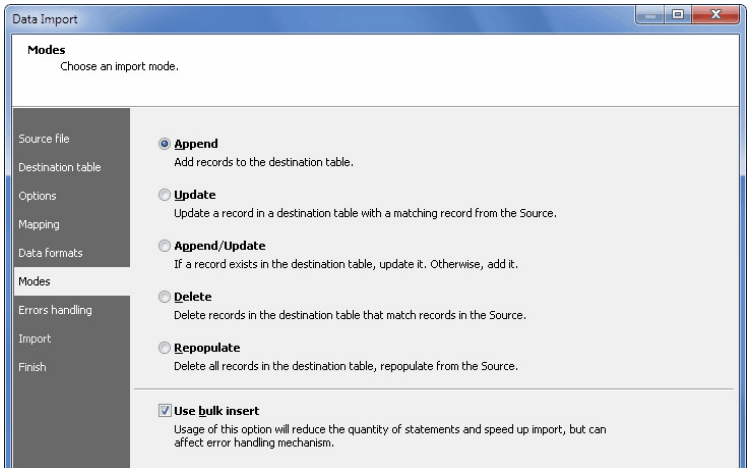
7. თუ ახალ ცხრილში ხდება იმპორტირება, ჩვენ შეგვიძლია Target სვეტის თვისებების რედაქტირება ორჯერ-დაწკაპუნებით ცხრილის სათავეში.
ვირჩევთ Key ალამს სვეტისათვის, რომელიც იქნება პირველადი გასაღები და ვაწკაპუნებთ Next.

8. განვსაზღვროთ მონაცემთა ფორმატი Source data -სათვის და ვაწკაპუნებთ Next.



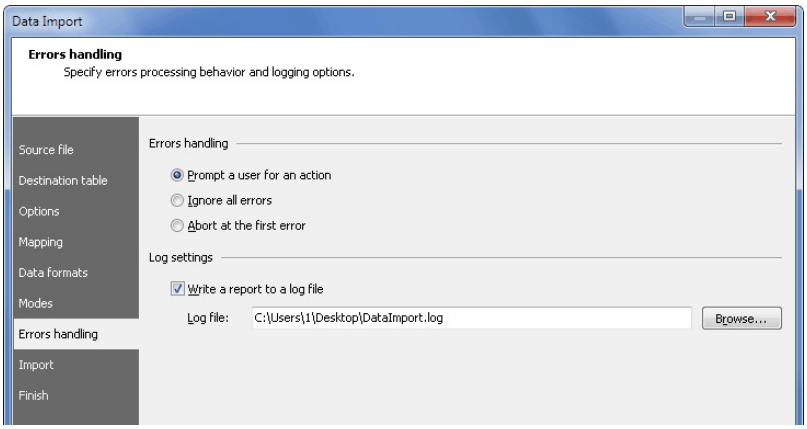
სურ. 11.42

9. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.43

10. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის სესიის განმავლობაში.



სურ. 11.44

11. ვაწკაპუნებთ **Import** და მივადევნებთ თვალს იმპორტის პროცესის მსვლელობას.

12. და ბოლოს, ვაწკაპუნებთ **Finish.**

9. XML იმპორტი

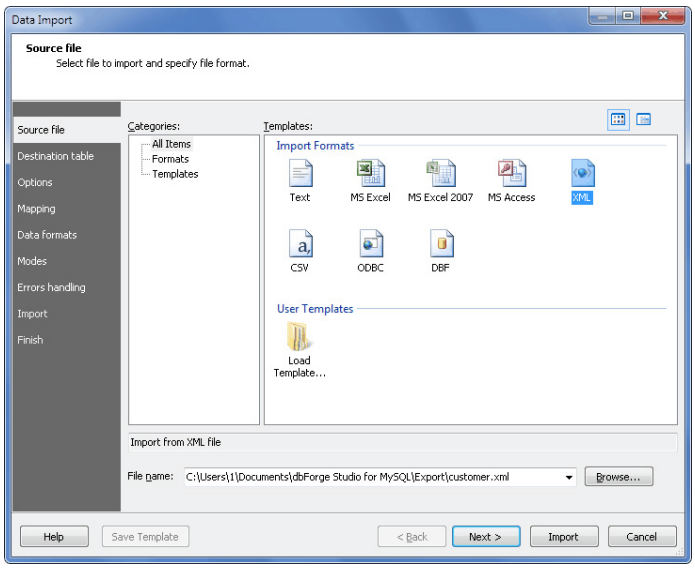
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

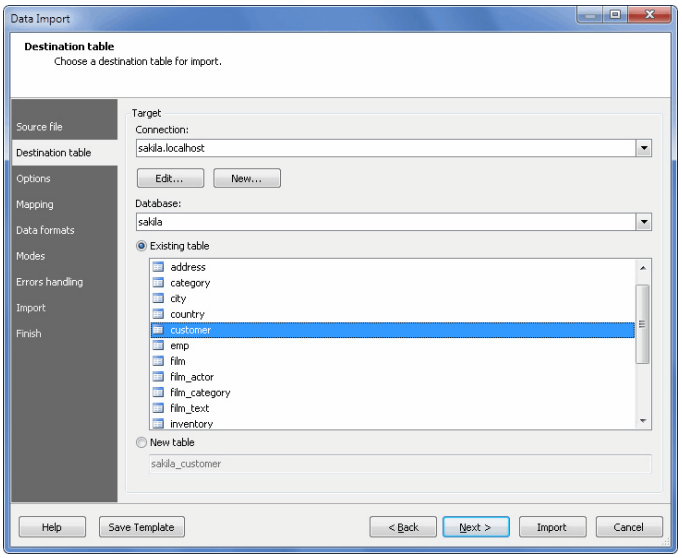
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ **XML** იმპორტის ფორმატს და განვსაზღვრავთ **Source** data -ის ადგილს. ვაწკაპუნებთ **Next**.



სურ. 11.45

3. განვსაზღვროთ MySQL მიერთება, მონაცემთა ბაზა და ცხრილი, რომელშიც გვინდა იმპორტის განხორციელება. თუ ცხრილი შერჩეულია **Database Explorer**-ში **Data Import** ოსტატის გახსნის წინ, მაშინ ოსტატი გაიხსნება შერჩეული ცხრილის მიერთების პარამეტრებით. MySQL სერვერის მიერთების შექმნის ან რედაქტირებისათვის ვაწკაპუნებთ შესაბამის ღილაკებზე. ვაწკაპუნებთ **Next**.



სურ. 11.46

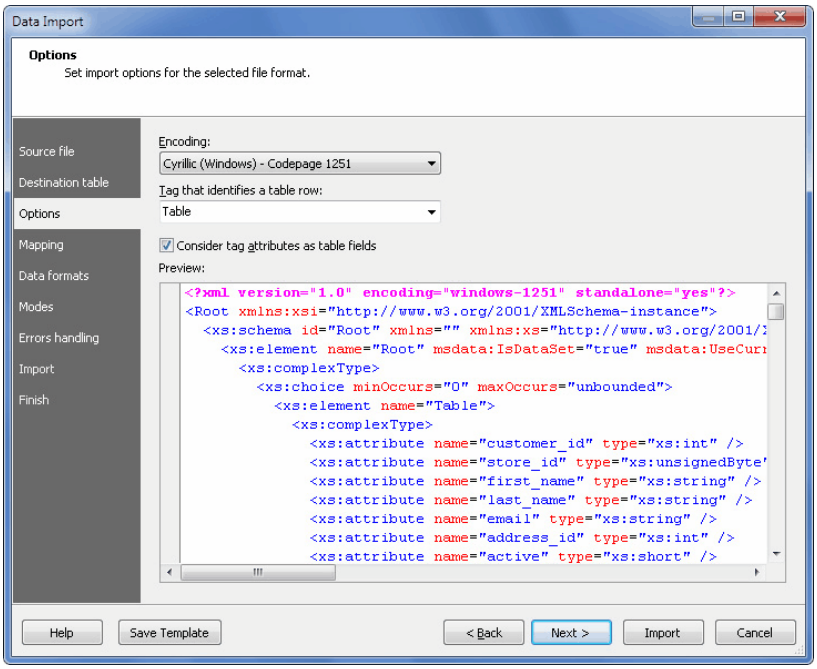
4. **Source** data-ს წინასწარი ნახვისა და იმპორტის დამატებითი ოპციების განსაზღვრისათვის:

• ვამოწმებთ **Source** კოდირების სისწორეს.

• ჩამოშლადი სიიდან ვირჩევთ tag-ს, რომელიც **Source** data -ში განსაზღვრავს ცხრილის სტრიქონს.

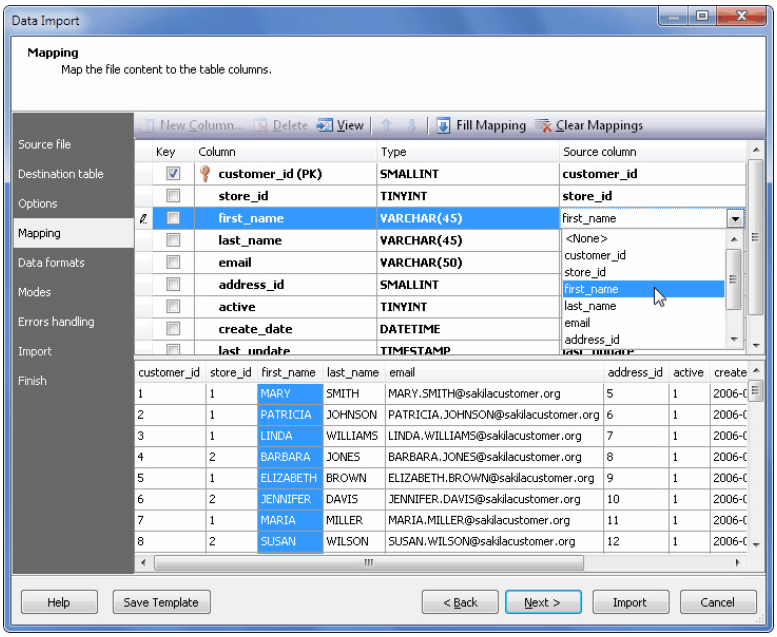
• ცხრილის ველების სახით ვირჩევთ **Consider** tag-ის ატრიბუტებს.

ვაწკაპუნებთ **Next**.



სურ. 11.47

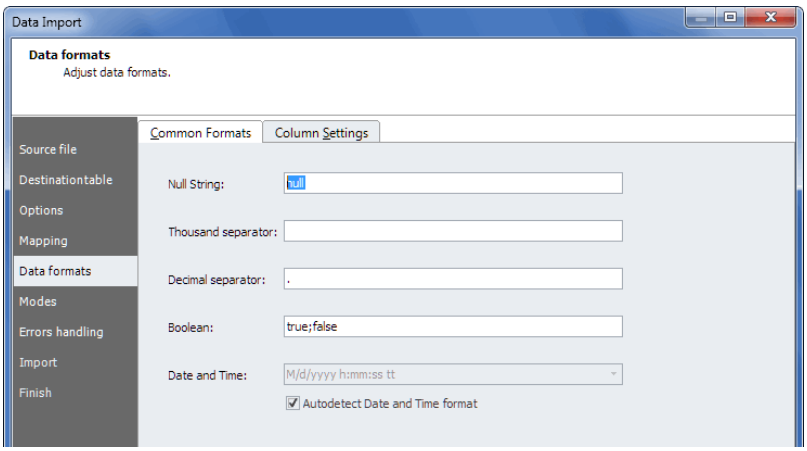
5. შევადგინოთ **Source** სვეტები **Target** -სათვის. ვაწკაპუნებთ **Source** სვეტის ველებზე და ჩამოშლადი სიიდან ვირჩევთ საჭირო სვეტებს.



სურ. 11.48

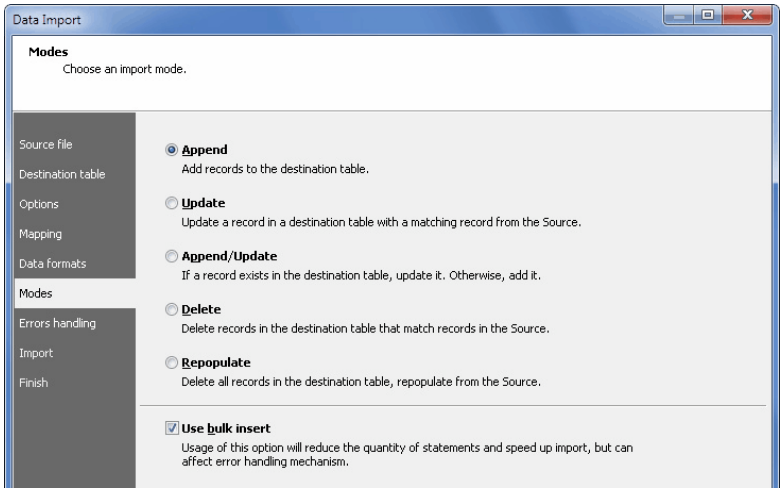
6. თუ ახალ ცხრილში ხდება იმპორტირება, ჩვენ შეგვიძლია **Target** სვეტის თვისებების რედაქტირება ორჯერ-დაწკაპუნებით ცხრილის სათავეში. ვირჩევთ Key ალამს სვეტისათვის, რომელიც იქნება პირველადი გასაღები და ვაწკაპუნებთ **Next**.

7. განვსაზღვროთ მონაცემთა ფორმატი Source data -სათვის და ვაწკაპუნებთ **Next**.



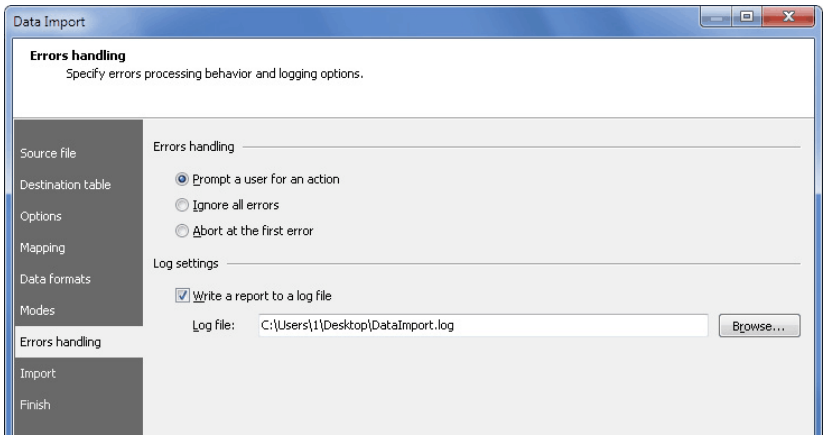
სურ. 11.49

8. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.50

9. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის სესიის განმავლობაში.



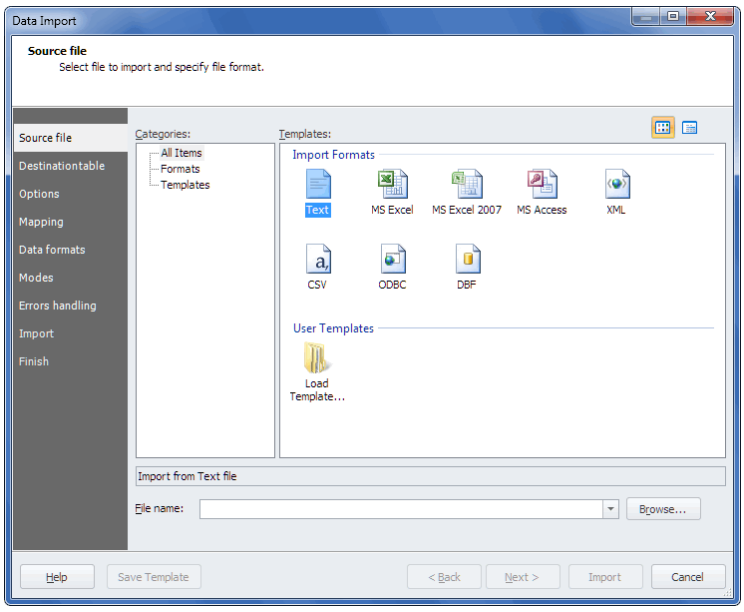
სურ. 11.51

10. ვაწკაპუნებთ **Import** და მივადევნებთ თვალს იმპორტის პროცესის მსვლელობას.

11. დასასრულს, ვაწკაპუნებთ **Finish**.

10. მონაცემთა იმპორტი ახალ ცხრილში

1.**Database** მენიუში ვაწკაპუნებთ **Import Data**. გაიხსნება Data Import ოსტატი.



სურ. 11.52

2. ვირჩევთ იმპორტის ფორმატს და განვსაზღვრავთ **Source** data -ის ადგილს. ვაწკაპუნებთ **Next**.

3. განვსაზღვროთ **Target** MySQL მიერთება და მონაცემთა ბაზა, ვირჩევთ ახალ ცხრილს და შეგვყავს ცხრილის სახელი, რომელშიც უნდა მოხდეს მონაცემთა იმპორტი. MySQL სერვერთან მიერთების შექმნისა და რედაქტირებისათვის ვაწკაპუნებთ შესაბამის ღილაკებზე. ვაწკაპუნებთ **Next**.

4. განვსაზღვროთ იმპორტის დამატებითი ოპციები. ვაწკაპუნებთ **Next**.

5. ვნახულობთ **Target** სვეტებს სათავეში და **Source** სვეტებს ოსტატის გვერდის ქვედა ნაწილში. ვაწკაპუნებთ **Source** სვეტის ველებზე და ჩამოშლადი სიიდან ვირჩევთ საჭირო სვეტებს.

6. **Target** სვეტის რედაქტირებისათვის ორჯერ ვაწკაპუნებთ მათზე ბადის სათავეში. სვეტისათვის ვირჩევთ Key ალამ-ს, რომელიც პირველად გასაღებად მიგვჩნია. ვაწკაპუნებთ **Next**.

7. Source data -სათვის განვსაზღვროთ მონაცემთა ფორმატი და ვაწკაპუნებთ **Next**.

8. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის სესიის განმავლობაში.

9. ვაწკაპუნებთ **Import** და ვაკვირდებით იმპორტის პროცესის მსვლელობას.

10. დასასრულს, ვაწკაპუნებთ **Finish**.

11. მონაცემთა მიგრაცია სხვა სერვერებიდან

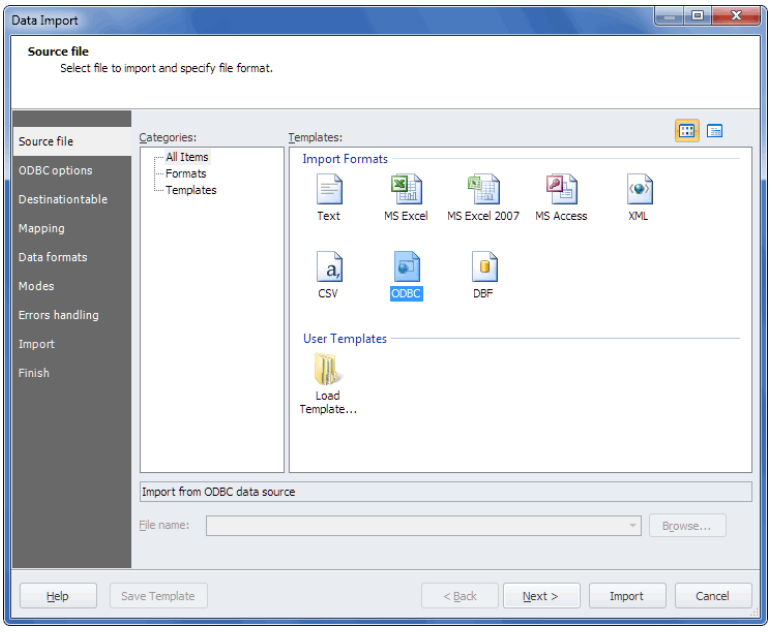
1. ახალი ცხრილისათვის:

• **Database** მენიუში ვაწკაპუნებთ **Import Data**.

არსებული ცხრილისათვის:

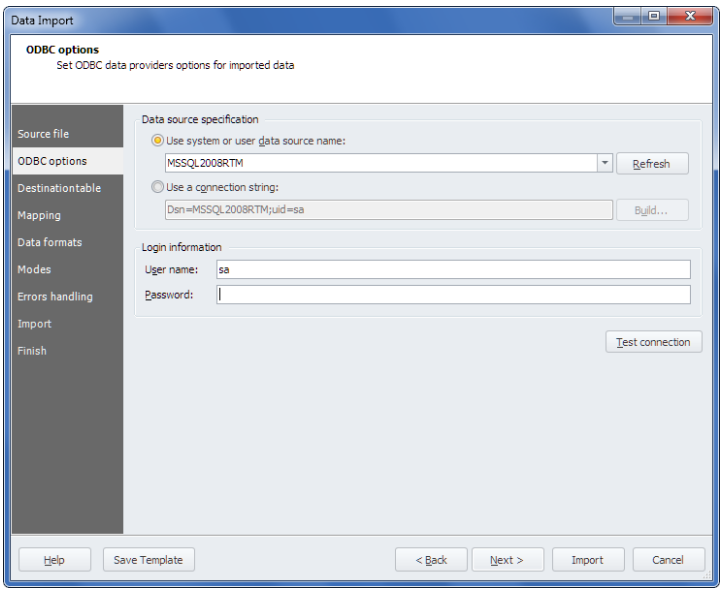
• **Database Explorer** -ში ცხრილზე მარჯვენა-დაწკაპუნებით გაიხსნება მენიუ, სადაც ვირჩევთ **Import Data**.

2. ვირჩევთ იმპორტის **ODBC** ფორმატს და განვსაზღვრავთ **Source** data-ს ადგილს. ვაწკაპუნებთ **Next**.



სურ. 11.53

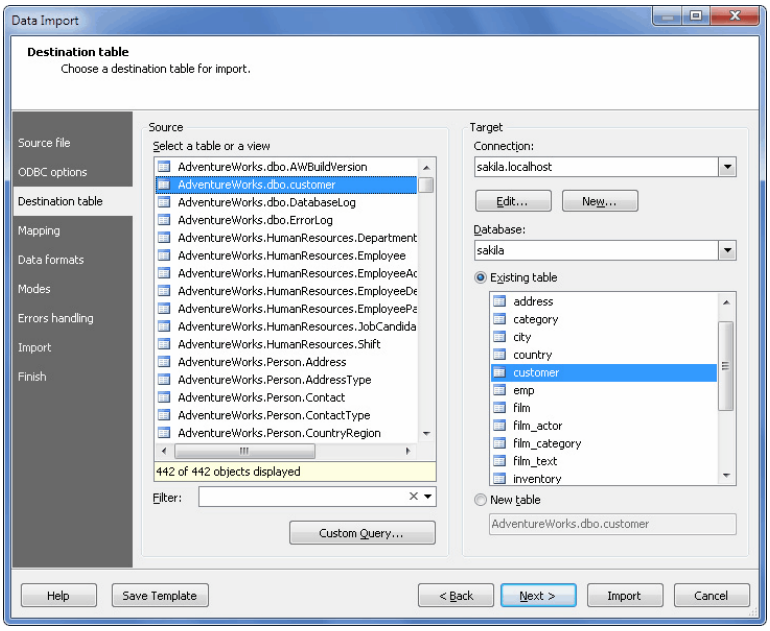
3. იმპოტისათვის განვსაზღვრავთ **ODBC** data პროვაიდერის ოპციებს. Build and selecting a required data source-ზე დაწკაპუნებით ჩამოშლადი სიიდან ვირჩევთ სისტემას ან მომხმარებლის მონაცემთა წყაროს სახელს ან განვსაზღვრავთ მიერთების სტრიქონს. შეგვაქვს მომხმარებლის სახელი და პაროლი. ვაწკაპუნებთ **Next**.



სურ. 11.54

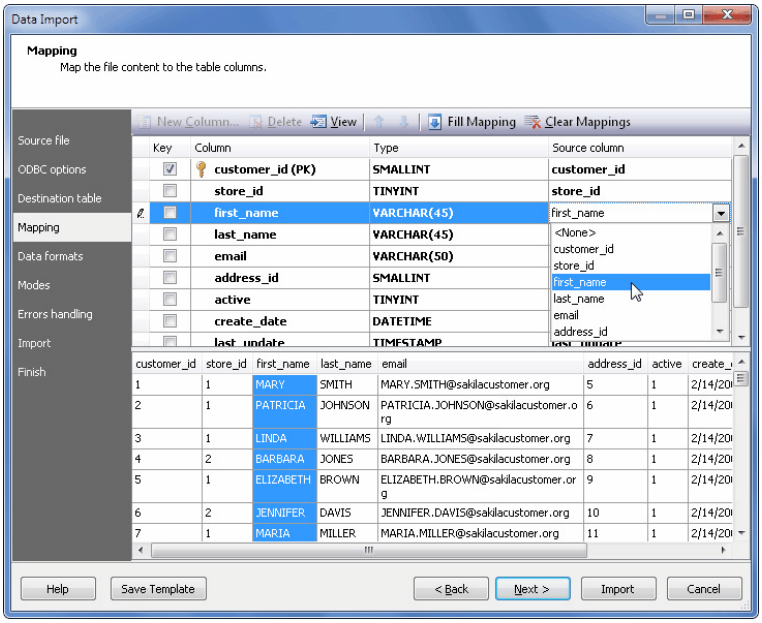
4. შევარჩიოთ **Source** ცხრილი ან წარმოდგენა. ჩვენ შეგვიძლია გამოვიყენოთ მოთხოვნა ნაწილობრივი იმპორტისათვის. ვაჭერთ ღილაკს **Custom Query** და

ვცვლით მოთხოვნას.
ცვლილებების შენახვისათვის ვაწკაპუნებთ **OK**.



სურ. 11.55

5. განვსაზღვროთ MySQL მიერთება და მონაცემთა ბაზა, ცხრილი. MySQL სერვერთან მიერთების შექმნისა და რედაქტირებისათვის ვაწკაპუნებთ შესაბამის ღილაკებზე. ვაწკაპუნებთ **Next**.
6. ვადგენთ Source სვეტებს Target-სათვის.

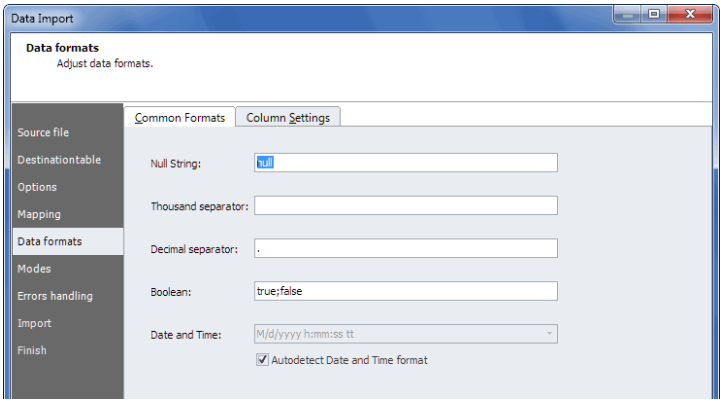


სურ. 11.56

7. თუ იმპორტი ხდება ახალ ცხრილში, ჩვენ შეგვიძლია **Target** სვეტის თვისებათა რედაქტირება, რისთვისაც ორჯერ ვაწკაპუნებთ მათზე ბადის სათავეში. ვირჩევთ **Key** ალამს სვეტისათვის, რომელიც პირველად გასაღებს წარმოადგენს და

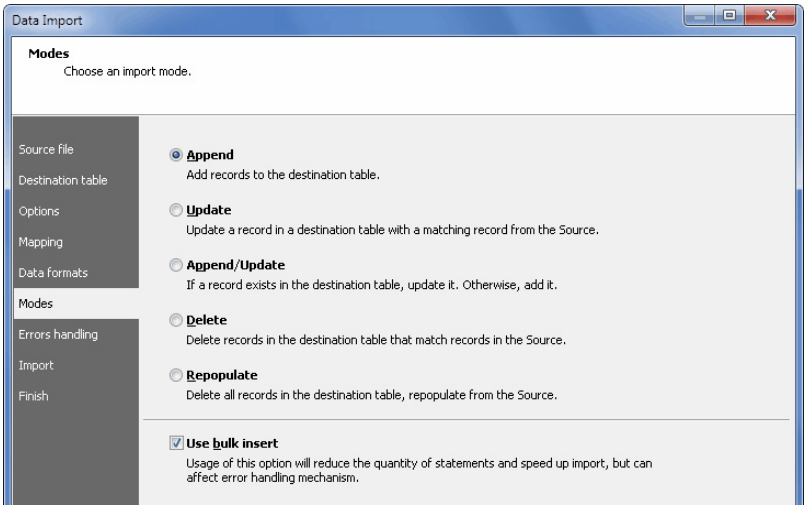
ვაწკაპუნებთ **Next**.

8. განვსაზღვრავთ მონაცემთა ფორმატს Source data -სათვის და ვაწკაპუნებთ **Next**.



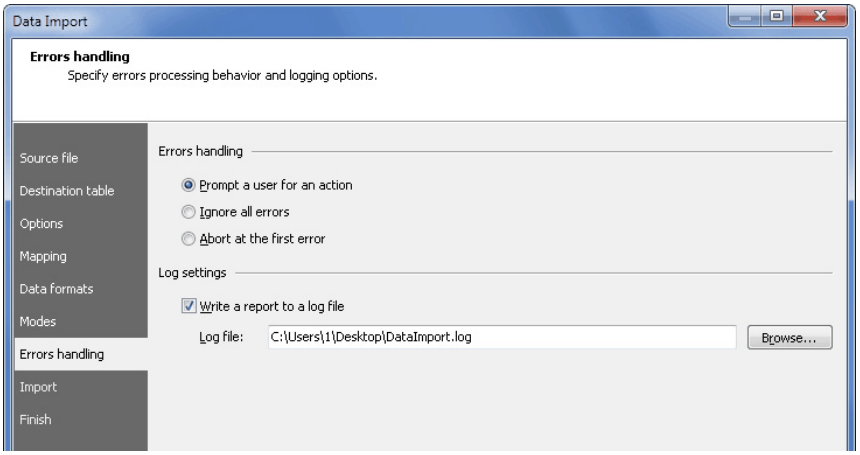
სურ. 11.57

9. ვირჩევთ იმპორტის სახეობას. ვაწკაპუნებთ **Next**.



სურ. 11.58

10. ვირჩევთ შეცდომებზე რეაგირებას იმპორტის სესიის განმავლობაში.



სურ. 11.59

11. ვაწკაპუნებთ **Import** და ვაკვირდებთ იმპორტის მიმდინარეობას.

12. და ბოლოს, ვაწკაპუნებთ **Finish**.

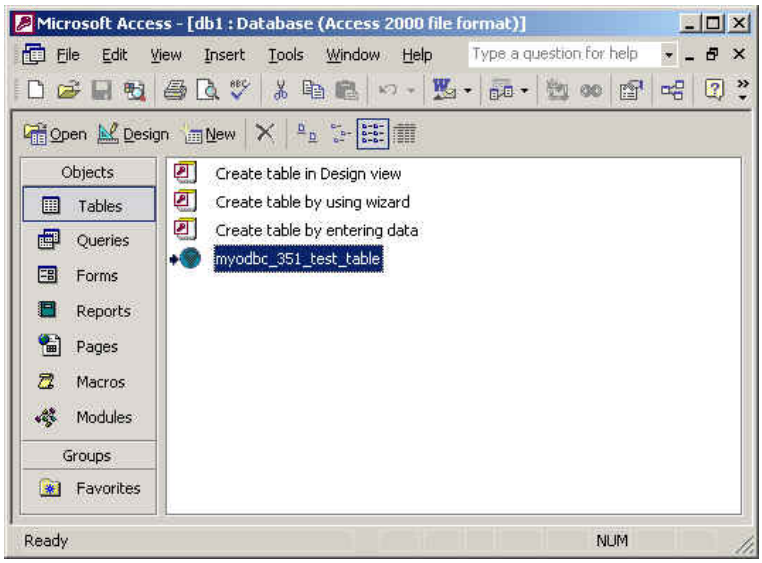
13. Connector/ODBC -ის Microsoft Access -თან გამოყენება

MySQL მონაცემთა ბაზის Microsoft Access-თან დაკავშირება შესაძლებელია Connector/ODBC მეშვეობით. MySQL მონაცემთა ბაზა შეიძლება გამოყენებულ იქნას როგორც იმპორტის, ისე ექსპორტის წყარო ან კიდევ როგორც ლინკირებული ცხრილი პირდაპირი გამოყენებისათვის Access აპლიკაციაში, რომელიც MySQL მონაცემთა ბაზის გარე ინტერფეისს წარმოადგენს.

Access მონაცემების ექსპორტი MySQL-ში

ცხრილის ექსპორტი Access მონაცემთა ბაზიდან MySQL -ში შესაძლებელია შემდეგი ინსტრუქციების შესრულების შედეგად:

1. გავხსნათ Access მონაცემთა ბაზა ან Access პროექტი. ჩნდება მონაცემთა ბაზის ფანჯარა თავისი ობიექტებით.



სურ. 11.60

2. ვაწკაპუნებთ **table** ან **query** სახელზე, რომლის ექსპორტი გვინდა და **File** მენიუში ვირჩევთ **Export**.

3. **Export Object Type Object name To** დიალოგურ ფანჯარაში, **Save As Type** ველში ვირჩევთ **ODBC Databases ()**:

სურ. 11.61

4. **Export** ფანჯარაში შეგვაქვს ფაილის სახელი და ვირჩევთ **OK**.

5. ჩნდება Select Data Source ფანჯარა, სადაც წარმოდგენილია

განსაზღვრული მონაცემთა წყაროები ODBC დრაივერისათვის ჩვენს კომპიუტერზე. ვაწკაპუნებთ ან File Data Source ან Machine Data Source ჩანართზე და შემდეგ ორჯერ ვაწკაპუნებთ Connector/ODBC ან Connector/ODBC 3.51.

Microsoft Access მიუერთდება MySQL Server ამ მონაცემთა წყაროს მეშვეობით და ახდენს ახალი ცხრილების ან მონაცემების ექსპორტს.

MySQL მონაცემების იმპორტი Access -ში

ცხრილის ან ცხრილების MySQL-დან Access ში იმპორტისათვის, საჭიროა შემდეგი ინსტრუქციების შესრულება:

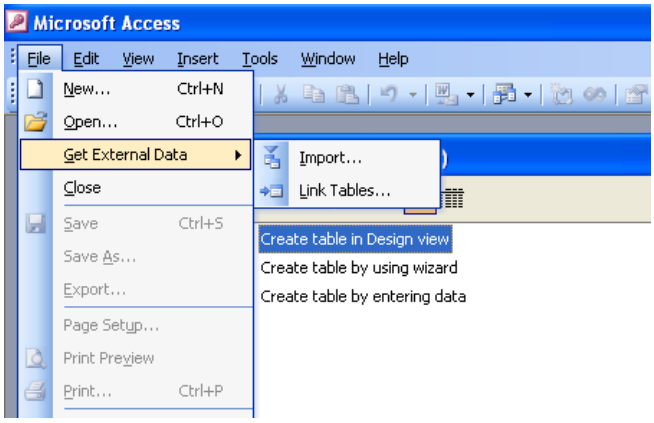
1. გავხსნათ მონაცემთა ბაზა ან ჩავრთოთ გახსნილი ბაზის ფანჯარა.
2. ცხრილების იმპორტისათვის **File** მენიუდან ვირჩევთ **Get External Data** და შემდეგ ვაწკაპუნებთ **Import**.
3. Import ფანჯარაში Files Of Type ველში ვირჩევთ **ODBC Databases ()**.
4. Select Data Source დიალოგურ ფანჯარაში **The Select Data Source** ფანჯრის სიაში ვათვალიერებთ განსაზღვრული მონაცემთა წყაროების სახელებს.
5. თუ ODBC მონაცემთა წყარო, რომელსაც ჩვენ შევირჩევთ, მოითხოვს სახელისა და პაროლის მითითებას, შეგვაქვს და შემდეგ ვაწკაპუნებთ **OK**.
6. Microsoft Access უერთდება MySQL სერვერს **ODBC data source** -ს მეშვეობით და ეკრანზე ჩნდება ცხრილების სია, რომლის იმპორტი შეგვიძლია.
7. იმპორტისათვის ვაწკაპუნებთ ყოველ ცხრილზე და შემდეგ ვაწკაპუნებთ **OK**.

14.MySQL-ის ინტერფეისი Microsoft Access გამოყენებით

Microsoft Access შეიძლება გამოყენებულ იქნას როგორც MySQL მონაცემთა ბაზის გარე ინტერფეისი, რისთვისაც Microsoft Access მონაცემთა ბაზის ცხრილები უნდა დავაკავშიროთ MySQL მონაცემთა ბაზის ცხრილებთან.

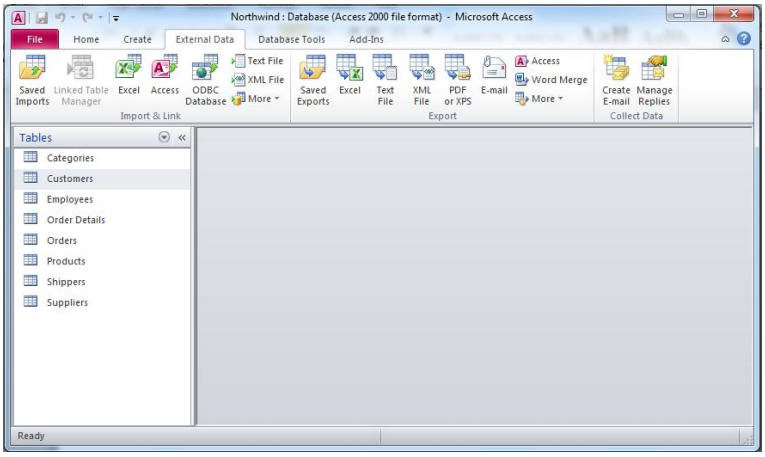
დაკავშირებული ცხრილის შესაქმნელად:

1. გავხსნათ Access მონაცემთა ბაზა, რომელთანაც გვინდა MySQL -ის დაკავშირება.
2. **File** მენიუდან ვირჩევთ **Get External Data->Link Tables**.



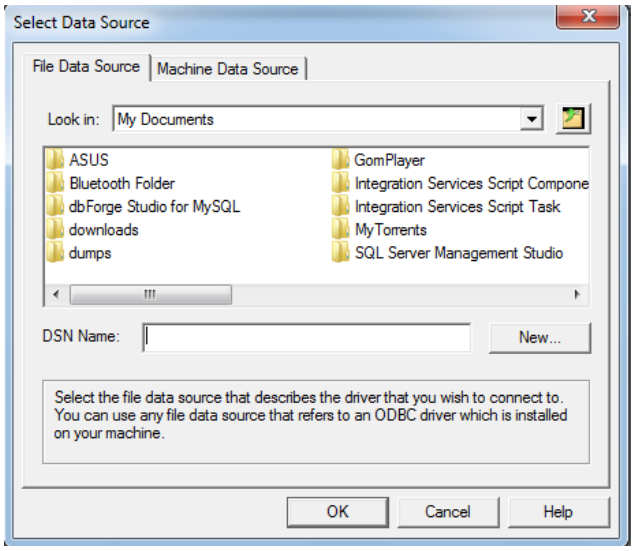
სურ. 11.62

3. ვირჩევთ **ODBC Databases ()**.



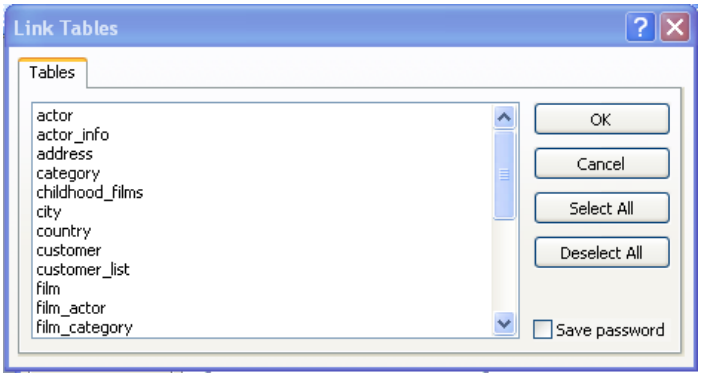
სურ. 11.63

4. **Select Data Source** ფანჯარაში ან **File Data Source** - დან ან **Machine Data Source** - დან ვირჩევთ არსებულ DSN. ჩვენ შეგვიძლია შევქმნათ ახალი DSN ღილაკის New გამოყენებით.



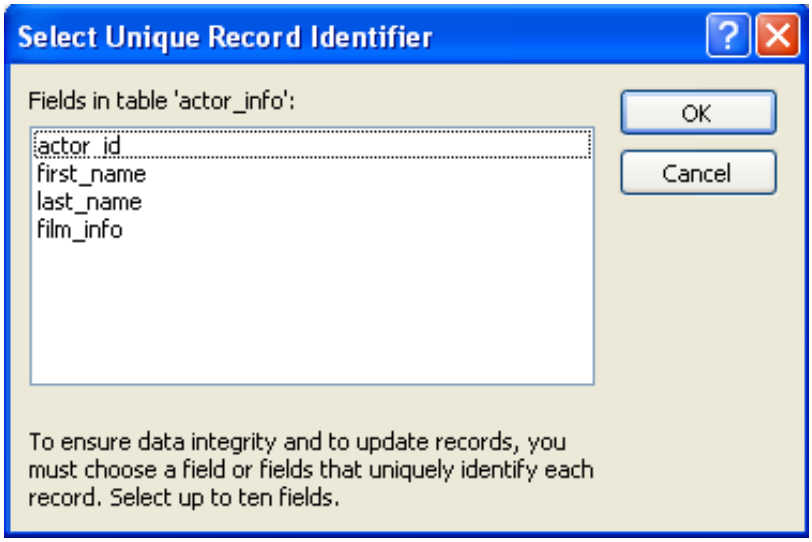
სურ. 11.64

5. **Link Tables** ფანჯარაში MySQL მონაცემთა ბაზიდან ვირჩევთ ერთ ან რამდენიმე ცხრილს.



სურ. 11.65

6. ვირჩევთ სვეტებს. ვაწკაპუნებთ **OK.**



სურ. 11.66

როცა პროცესი დასრულდება, ჩვენ შეგვიძლია ავაგოთ ინტერფეისი და მოთხოვნები დაკავშირებული ცხრილების მიხედვით, როგორც ეს ხდება Access -ში.

დაკავშირებული ცხრილის ნახვისათვის:

1. გავხსნათ მონაცემთა ბაზა, რომელიც შეიცავს კავშირებს MySQL ცხრილებთან.
2. **Tools** მენიუში ვირჩევთ **Add-ins** (**Database Utilities**) და შემდეგ ვაწკაპუნებთ **Linked Table Manager**.
3. ვაყენებთ ალამს ცხრილებისათვის, რომელთა კავშირების გასურლება გვინდა.
4. კავშირების გასურლებისათვის ვაწკაპუნებთ **OK**.

დაკავშირებული ცხრილებისათვის გზის შეცვლა:

1. გავხსნათ მონაცემთა ბაზა, რომელიც შეიცავს კავშირებს ცხრილებთან.
2. **Tools** მენიუში ვირჩევთ **Add-ins** (**Database Utilities**) და შემდეგ ვაწკაპუნებთ **Linked Table Manager**.
3. ვირჩევთ **Always Prompt For A New Location** ალამს.
4. ვირჩევთ ალამსცხრილებისათვის, რომელთა კავშირების შეცვლა გვინდა და შემდეგ ვაწკაპუნებთ **OK**.
5. დიალოგურ ფანჯარაში **Select New Location of** <table name> განვსაზღვროთ ახალი ადგილსამყოფელი. ვაწკაპუნებთ **Open** და შემდეგ ვაწკაპუნებთ **OK**.

15. MYSQL მონაცემთა ბაზის ფაილის ადგილმდებარეობის განსაზღვრა

ყველაზე მარტივი და პირდაპირი მეთოდი: ვიპოვოთ my.ini ფაილი იმ ფოლდერში, სადაც MYSQL არის დაინსტალირებული. თუ ჩვენ დავტოვებთ იმ გზას, როგორც უსიტყვოდ იყო შემოთავაზებული:

C:\Program Files\MySQL\MySQL Server XX.XX

(სადაც XX.XX არის MYSQL ვერსიის ნომერი)

გავხსნათ my.ini ტექსტურ რედაქტორში და მოვძებნოთ ტექსტი **datadir**.

```
datadir=”C:/ProgramData/MySQL/MySQL Server 5.5/Data/”
```

ეს მიუთითებს იმას, რომ ჩვენი MYSQL მონაცემთა ბაზა ისურება ჩვენს კომპიუტერზე.

უნდა აღინიშნოს, რომ InnoDB ცხრილების ფაილები ან **ibdata** ფაილი უნდა ისურებოდეს ერთად მონაცემთა ფოლდერში. თუმცა შესაძლებელია ისურებოდეს სხვა ადგილას, რომელიც განსაზღვრული იყო ინსტალაციის დროს.

ანალოგიურად მოვძებნოთ ibdata ფაილი my.ini ფაილში.

```
innodb_data_home_dir=”C:/MySQL Datafiles/”
```

16. მონაცემთა ბაზის გადატანა ერთი კომპიუტერიდან მეორეზე

როგორც წესი, მონაცემთა ბაზის გადატანა ერთი კომპიუტერიდან მეორეზე ხდება შემდეგნაირად:

1. გამოვიყენოთ Backup ფუნქცია (Advanced/Maintenance) პირველ კომპიუტერზე და შევქმნათ სარეზერვო (backup) ფაილი ვინჩესტერზე (რადგან შეუძლებელია მისი შექმნა პირდაპირ CD-ზე).
2. Windows Explorer -ის გამოყენებით ვაკოპირებთ backup ფაილს CD ROM -ზე (ან პირდაპირ სხვა კომპიუტერზე).
3. ვაკოპირებთ backup ფაილს CD-დან სხვა კომპიუტერის ვინჩესტერზე.
4. გამოვიყენოთ Restore ფუნქცია (Advanced/Maintenance) ჩვენი მონაცემთა ბაზის აღსადგენად backup ფაილიდან.

ლაბორატორიული სამუშაო 12
MySQL: ადმინისტრირება და უსაფრთხოების სისტემა

სამუშაოს მიზანი:

1. მონაცემთა ბაზის ადმინისტრირება
2. მონაცემთა ბაზის ადმინისტრირება MySQL Administrator-ში
3. MySQL ადმინისტრირება და მონიტორინგი MySQL Workbench -ის გამოყენებით
4. ადმინისტრირების ამოცანები dbForge Studio for MySQL -ში
5. უსაფრთხოების სისტემა
6. მომხმარებელთა მართვა MySQL -ში
7. მომხმარებლებისა და პრივილეგიების მართვა dbForge Studio გამოყენებით
8. ახალი კლიენტური ანგარიშის დაყენება
9. კლიენტური ანგარიშების მანიპულირება Security Manage-ში
10. მონაცემთა ბაზის ობიექტებისათვის პრივილეგიების მინიჭება
11. როლების გამოყენება Security Manager-ში

თეორიული მასალა: MySQL – ის ადმინისტრირების ინსტრუმენტების მიმოხილვა და გამოყენების ასპექტები.

1.მონაცემთა ბაზის ადმინისტრირება

მონაცემთა ბაზის ადმინისტრირებისათვის აუცილებელია ინფორმაციის ნახვა სერვერისა და ამ სერვერზე განთავსებული მონაცემთა ბაზების შესახებ. ამ მიზნით გამოიყენება სხვადასხვა ინსტრუმენტი.

MySQL show

ამ ინსტრუმენტთან ერთად მონაცემთა ბაზების, ცხრილებისა და სვეტების ნახვა სწრაფად არის შესაძლებელი. პარამეტრების გარეშე ბრძანებას გამოაქვს MySQL -ის მიერ მართული ყველა მონაცემთა ბაზის სია, ხოლო პარამეტრებით ბრძანებას გამოაქვს ინფორმაცია მონაცემთა ბაზების, ცხრილებისა და სვეტების შესახებ.

mysqlshow - ის სინტაქსისი შემდეგია:

mysqlshow [options] [databasename [tablename [columnname]]]

mysqlshow - Options

მაგალითი:

```
mysqlshow -u root -p
mysqlshow -u root -p mysql
mysqlshow -u sakilaadmin -psakila sakila customer
```

შემდეგი მაგალითი მიუთითებს ცვლადებს mysql სესიაში. მისი კოდი შემდეგია:

```
SHOW GLOBAL VARIABLES;
```

SHOW სესია VARIABLES;

MySQLadmin Tools

სხვადასხვა ადმინისტრაციული ამოცანა, ისეთი როგორიცაა ახალი მონაცემთა ბაზის შექმნა-წაშლა, პაროლის შეცვლა და ა.შ., დაკავშირებულია mysqladmin tool -თან, რომლის სინტაქსისი შემდეგია:

mysqladmin [options] ბრძანება1 ბრძანება2 ...

კოდის მაგალითი:

mysqladmin -uroot -p291069kg --sleep=1800 variables

კოდის მაგალითი:

mysqladmin -uroot -p291069kg --sleep=30 --relative extended-status

mysqladmin ბრძანებების გამოყენებით ჩვენ შეგვიძლია ვსუროთ იმ პროცესების სია, რომელიც მიმდინარეობენ MySQL -ში. შესაძლებელია აგრეთვე „დაკიდებული“ ან დიდხანს მიმდინარე მოთხოვნებთან არსებული კავშირების მოსპობა.

mysqladmin ბრძანებები	**Description**	**ბრძანება MySQL სესია**
processlist	Displays all MySQL threads	SHOW THREADS
kill id1, id2	Terminates the specified thread(s) via thread-ID	KILL
flush-threads	Empties the thread cache.	FLUSH THREADS

კოდის მაგალითი:

mysqladmin -uroot -p291069kg processlist

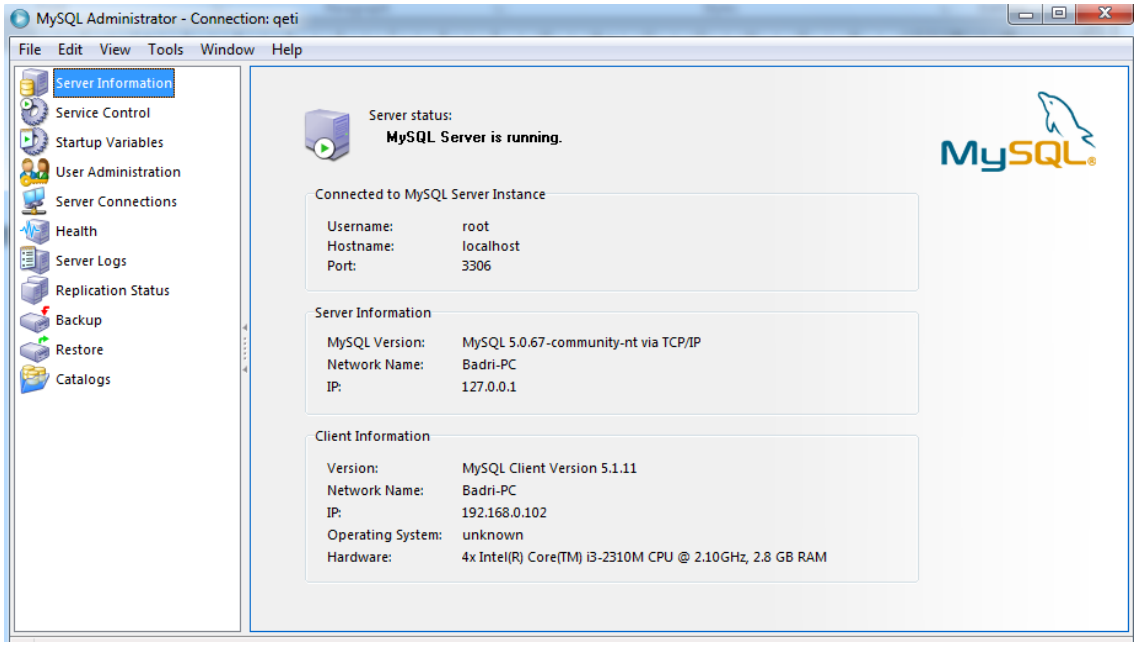
2. მონაცემთა ბაზის ადმინისტრირება MySQL Administrator-ში

MySQL Administrator tools მნიშვნელოვნად ამარტივებს MySQL სერვერის ადმინისტრირებისა და მართვის ამოცანებს (ჩატვირთვა, ღია შეერთებების რაოდენობის განსაზღვრა, სტატუსის განსაზღვრა და სხვ.). პროგრამა შეიძლება გამოყენებულ იქნას როგორც ლოკალური სერვერის, ისე ქსელის ადმინისტრირებისათვის.

MySQL Administrator-ის მრავალრიცხოვანი ფუნქციები განაწილებულია სხვადასხვა მოდულებში. მოკლედ მიმოვიხილოთ ზოგიერთი მათგანი.

Server Information

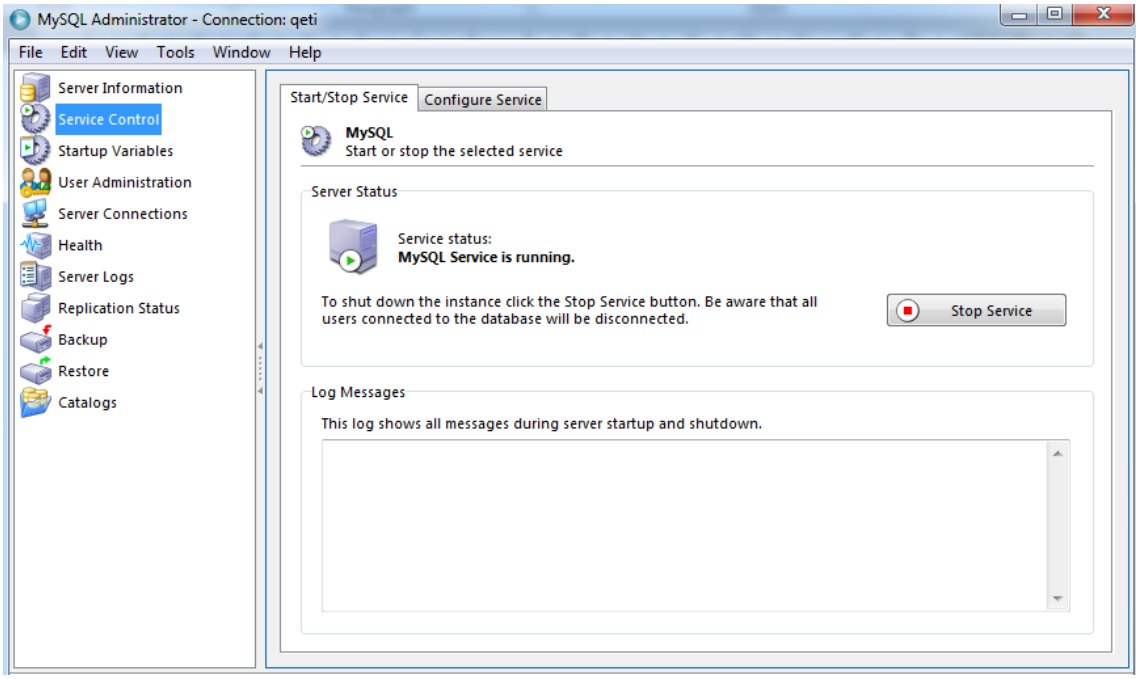
ეს მოდული აგროვებს ინფორმაციას სერვერთან შეერთებების სტატუსის შესახებ, მათი ვერსიების და კლიენტური ბიბლიოთეკების შესახებ.



სურ. 12.1

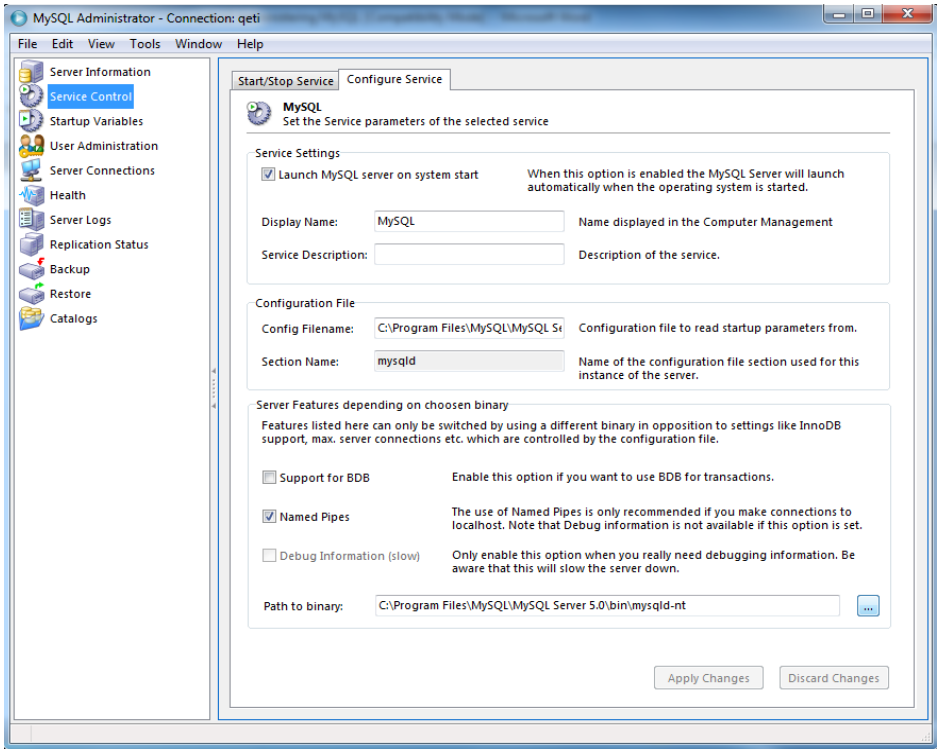
მოდული Service Control

ამ მოდულის საშუალებით შესაძლებელია სერვერის გაშვება და გაჩერება. იგი მუშაობს მხოლოდ ლოკალურ სერვერთან.



სურ. 12.2

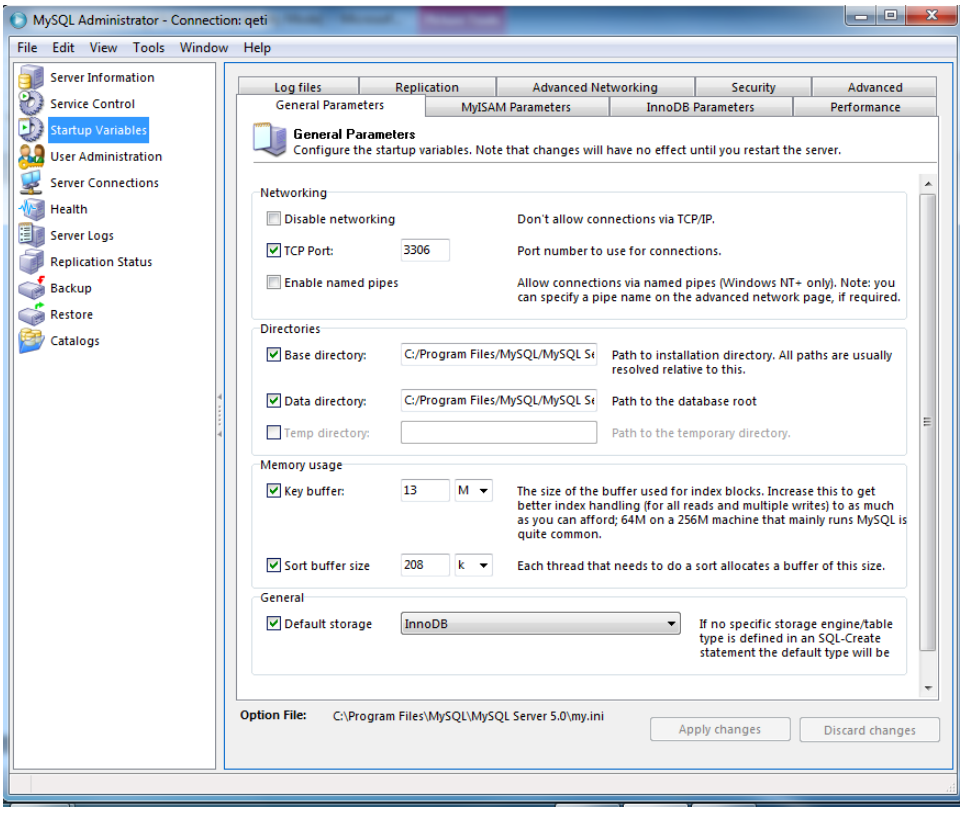
გარკვეული ოპციების დაყენება შესაძლებელია ჩანართში *CONFIGURE SERVICE*.



სურ. 12.3

მოდული **Startup Variables**

ეს მრავალჩანართიანი მოდული MySQL კონფიგურაციის ფაილის ვიზუალური რედაქტირების საშუალებას იძლევა.



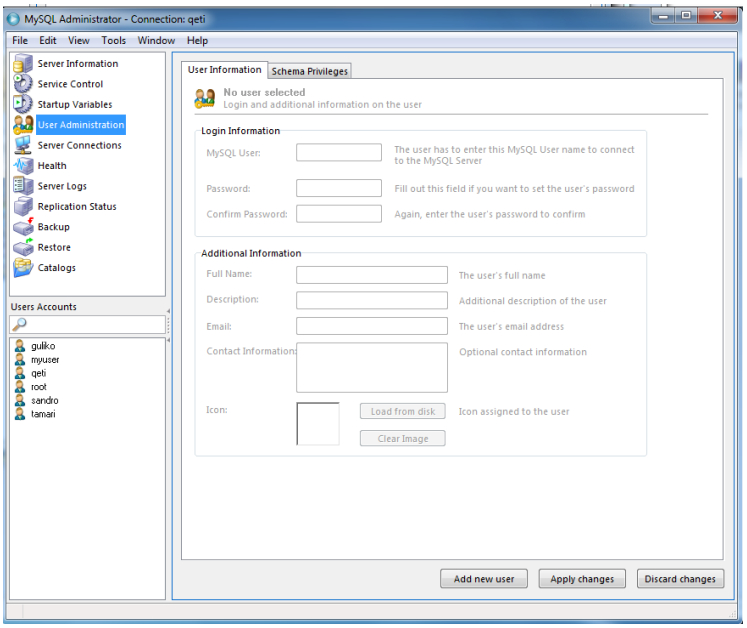
სურ. 12.4

თუ ფაილი არ არსებობს, ის ავტომატურად შეიქმნება რედაქტირების შედეგად ეკრანზე. MySQL Administrator - სათვის საჭიროა ფაილისათვის პრივილეგიების მინიჭება.

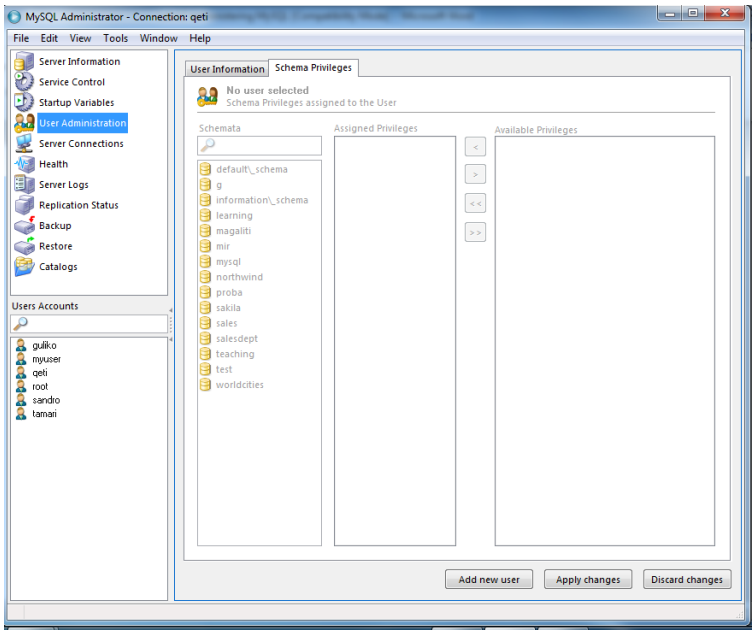
MySQL Administrator - ს ყოველი ოპციის გასწვრივ გააჩნია მომცრო ღილაკი (წითელი X სიმბოლო), რაც ნიშნავს, რომ ოპცია გააქტიურებულია.

მოდული User Administration

ეს მოდული ახალი მომხმარებლების შექმნას (*NEW USER* ბრძანებით) და არსებული მომხმარებლებისათვის წვდომის პრივილეგიების შეცვლის საშუალებას იძლევა (ჩანართში *SCHEMA PRIVILEGES).*



სურ. 12.5



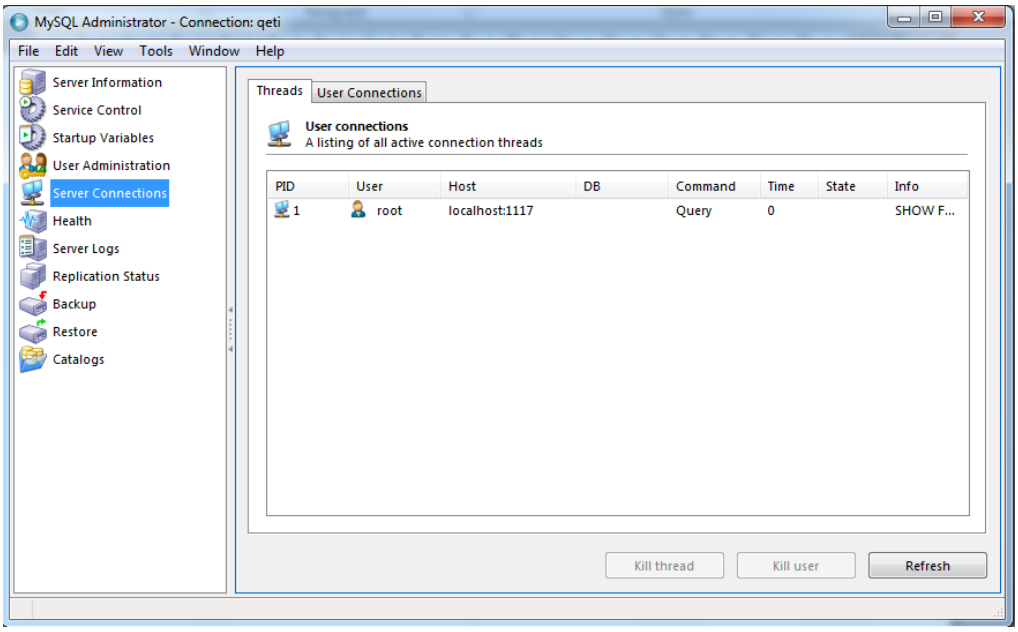
სურ. 12.6

მოდული Server Connections

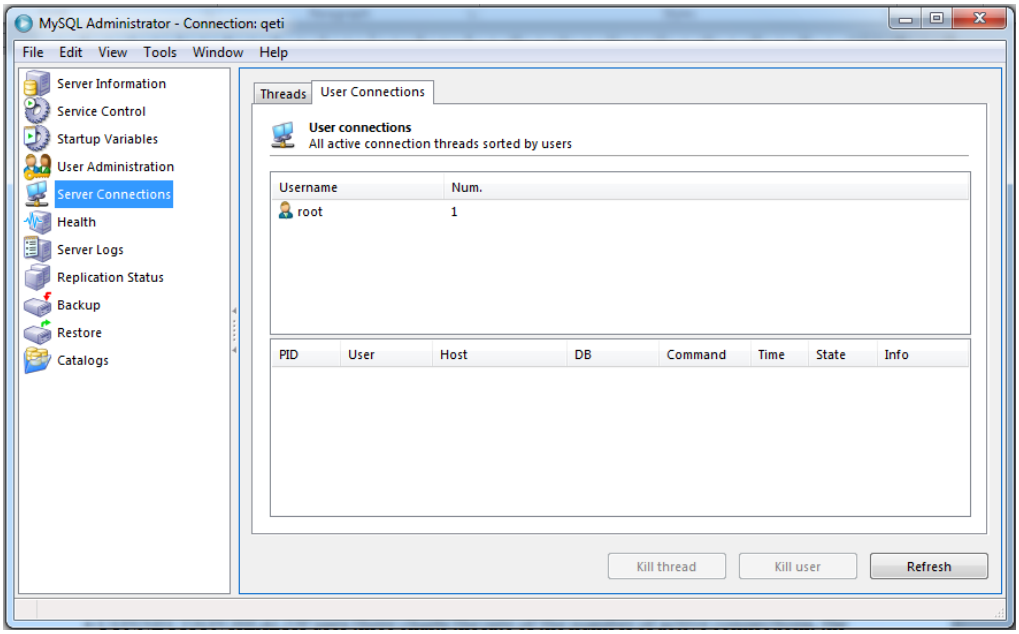
ეს მოდული შეიცავს ორ ჩანართს:

• THREADS გვიჩვენებს ყველა აქტიური კავშირის სიას (შეესაბამება SQL ბრძანებას SHOW PROCESSLIST).

• USER CONNECTION შეიცავს ყველა მიმდინარე აქტიური მომხმარებლის სიას მათ შეერთებებთან და კავშირებთან ერთად, რომელიც დაჯგუფებულია მომხმარებლის სახელის მიხედვით.



სურ. 12.7

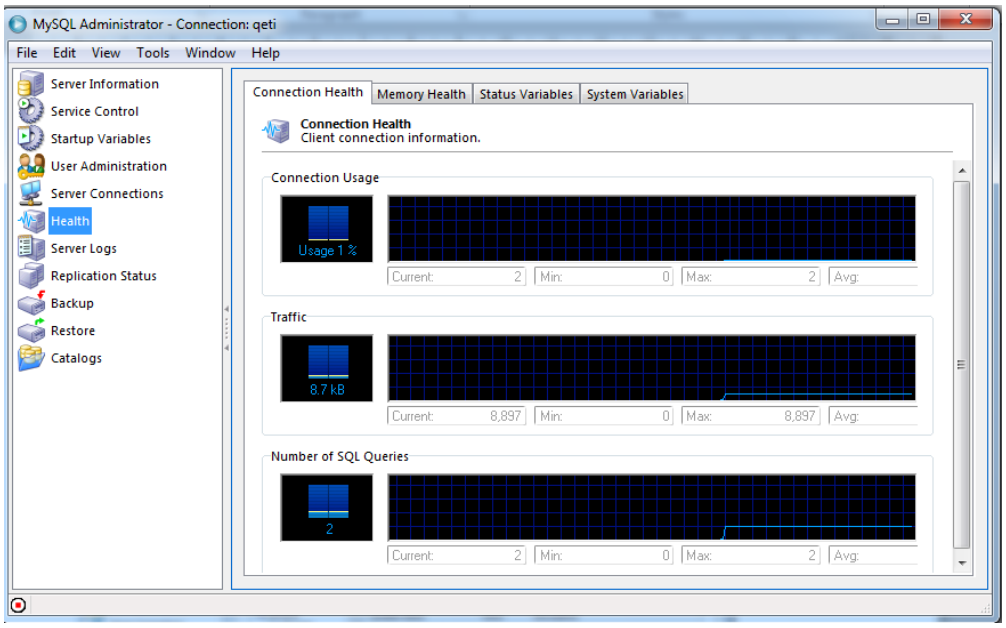


სურ. 12.8

მოდული Health

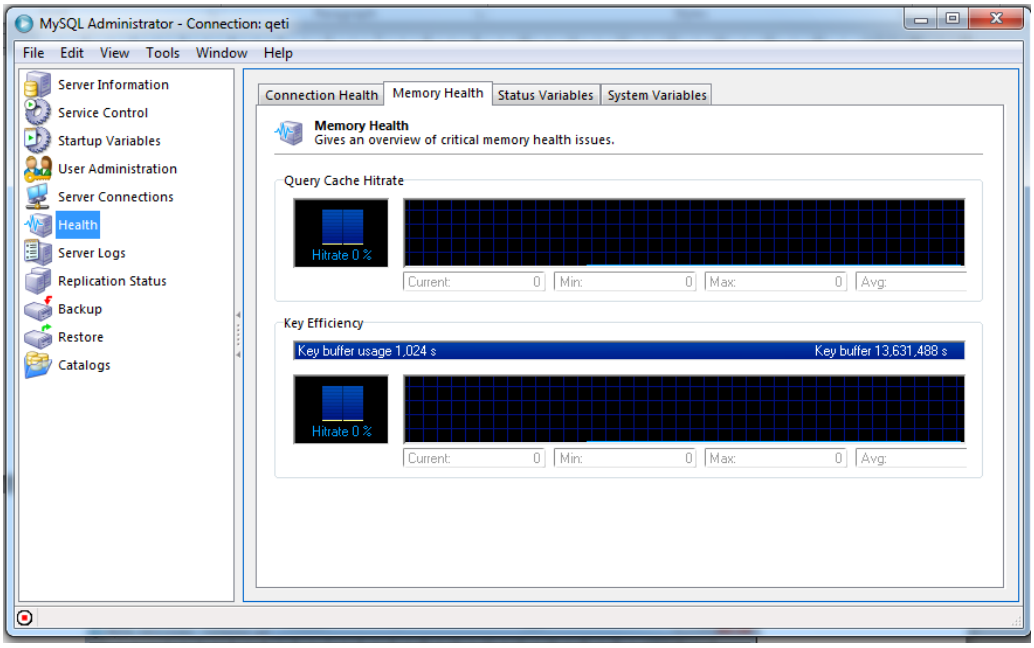
ეს ოთხ ჩანართიანი მოდული (ჩანართების სკრინშოტები მოყვანილია ქვემოთ) იძლევა ინფორმაციას MySQL სერვერის ჩატვირთვისა და გამოყენების შესახებ.

CONNECTION HEALTH იყენებს აქტიური შეერთებების, ტრაფიკისა და ერთ წამში შესრულებული მოთხოვნების რაოდენობის შეფასების სამ მახასიათებელს.



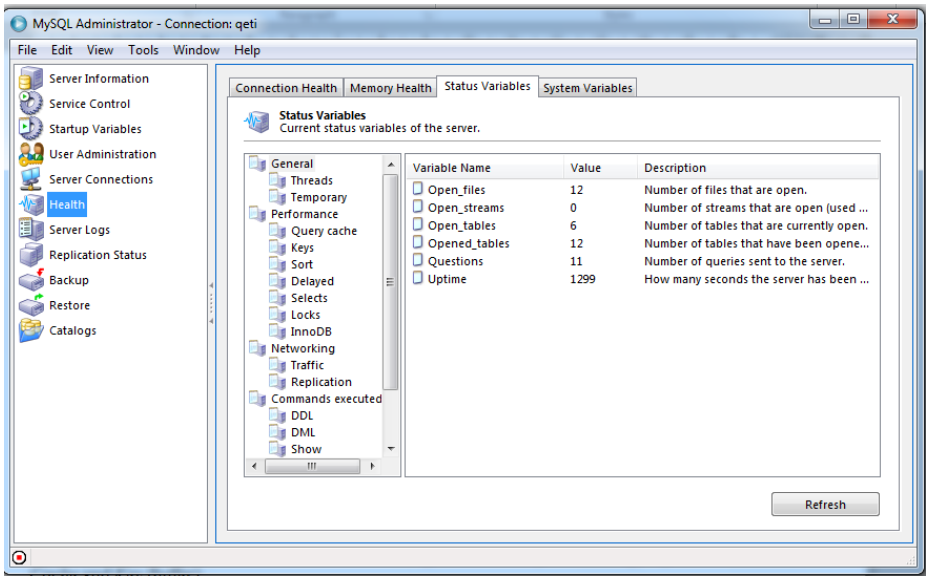
სურ. 12.9

MEMORY HEALTH გვიჩვენებს მეხსიერების ორი მნიშვნელოვანი ზონის (*Query Cache* და *Key Buffer*) მდგომარეობის შესახებ ინფორმაციას.



სურ. 12.10

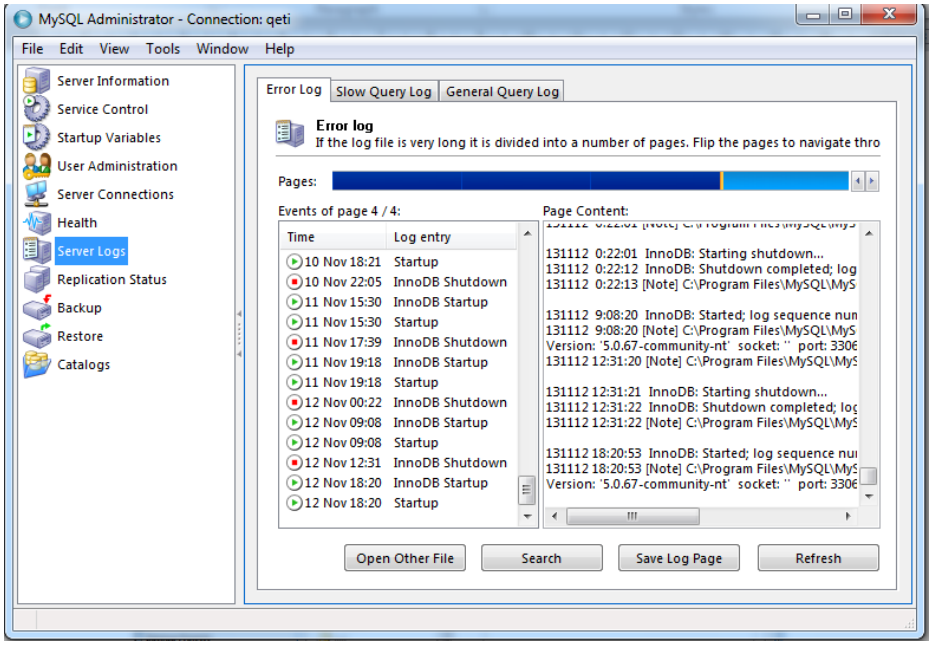
STATUS VARIABLES და *SERVER VARIABLES* იყენებს ცხრილებს უთვალავი დამატებითი სტატუსის მართვისათვის.



სურ. 12.11

მოდული Server Logs

ამ მოდულთან ერთად დაუბრკოლებლად შეგვიძლია MySQL server-ის ჩანაწერების ნახვა.



სურ. 12.12

მოდული Backup

ამ მოდულთან ერთად შეგვიძლია მონაცემთა ბაზების სარეზერვო ასლების შექმნა.

Replication Status

როცა MySQL server წარმოადგენს რეპლიკაციის სისტემის ნაწილს, მაშინ ეს მოდული გვიჩვენებს რეპლიკაციის სტატუსს.

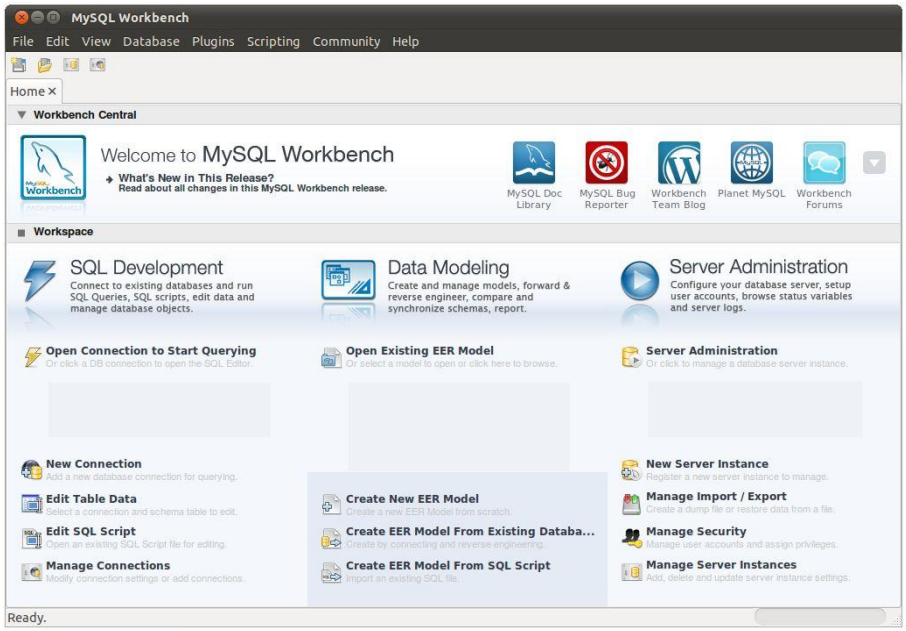
მოდული Catalogs

Catalogs არის მოდული, რომელიც განკუთვნილია მონაცემთა ბაზებისა და ცხრილების მართვისათვის, კერძოდ, შექმნისა და მოდიფიცირებისათვის.

3.MySQL ადმინისტრირება და მონიტორინგი MySQL Workbench - ის გამოყენებით

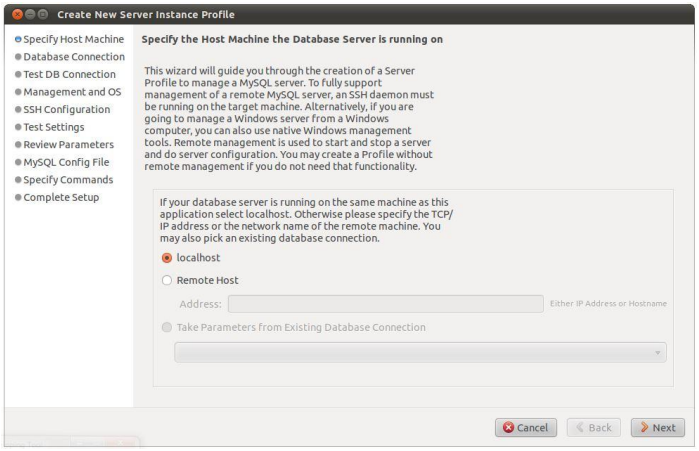
Server Instance Profile

MySQL Workbench - ის მთავარ ეკრანზე ვირჩევთ ოპციას Server Administration:



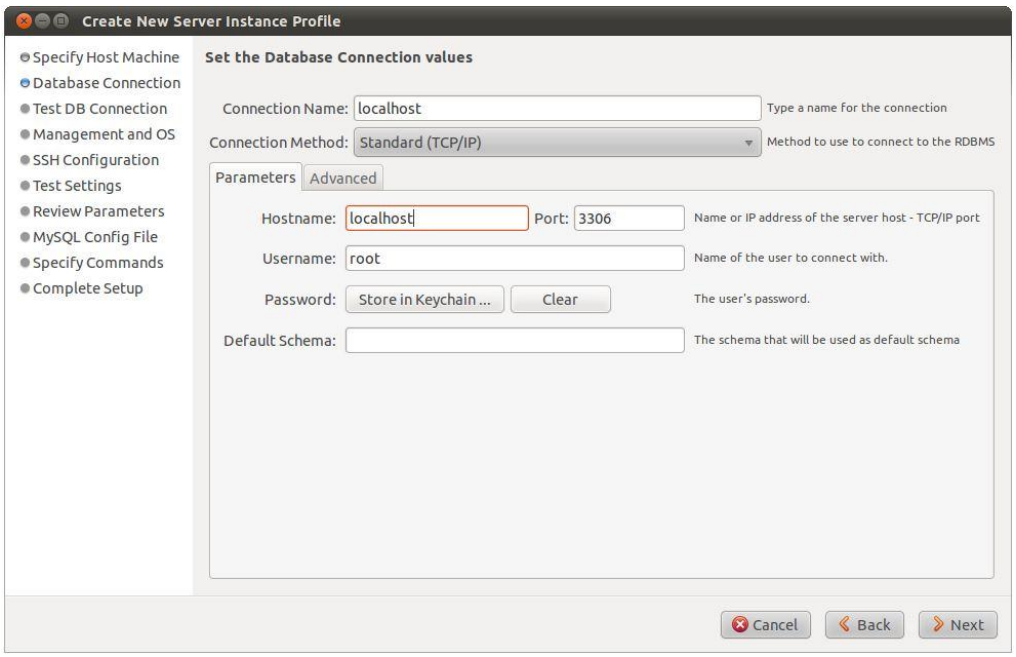
სურ. 12.13

ვაწკაპუნებთ ოპციაზე New Server Instance. შედეგად ჩნდება ოსტატი Create New Server Instance Profile. ეს ოპცია უზრუნველყოფს წვდომას როგორც ლოკალურ, ისე დაშორებულ სერვერთან. ამ უკანასკნელ შემთხვევაში შესაბამისი გადამრთველი უნდა ავირჩიოთ და შევიტანოთ სისტემის hostname ან IP მისამართი.



სურ. 12.14

შემდეგ ეკრანზე შეგვაქვს შეერთების დაწვრილებითი სახელი შეერთების ტიპის (TCP/IP, ლოკალური პანელი ან უფრო უსაფრთხო TCP/IP SSH-სთან ერთად) მითითებით და აგრეთვე მომხმარებლის სახელი, პორტი, რომლის მეშვეობითაც მონაცემთა ბაზისადმი წვდომა უნდა მოხდეს.



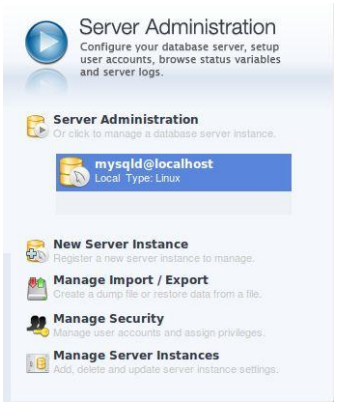
სურ. 12.15

ვაწკაპუნებთ ღილაკზე Next. ოსტატი მოითხოვს პაროლს, მომხმარებლის სახელის მიხედვით და შეერთებას. თუ შეერთება წარმატებით დამყარდა, ჩნდება შეტყობინება ამის თაობაზე. წინააღმდეგ შემთხვევაში ვბრუნდებით წინა ეკრანზე შეერთების დაყენების კორექტირებისათვის.

მომდევნო ეკრანზე ვირჩევთ ოპერაციულ სისტემას და მონაცემთა ბაზის ტიპს, რომელიც არის განთავსებული მიზნობრივ სისტემაში. თუ შეერთების ტესტირების განმავლობაში, ოსტატი მათ წარმატებით აღმოაჩენს, მაშინ შესაბამისად მოხდება ამ მნიშვნელობათა ზუსტი კონფიგურაცია.

შემდეგ ეკრანზე ოსტატი მოძებნის შეერთებას host კომპიუტერთან, რომელზეც workbench არის გაშვებული. კვლავ ვაწკაპუნებთ ღილაკზე Next, თუ ტესტი წარმატებით იქნა შესრულებული.

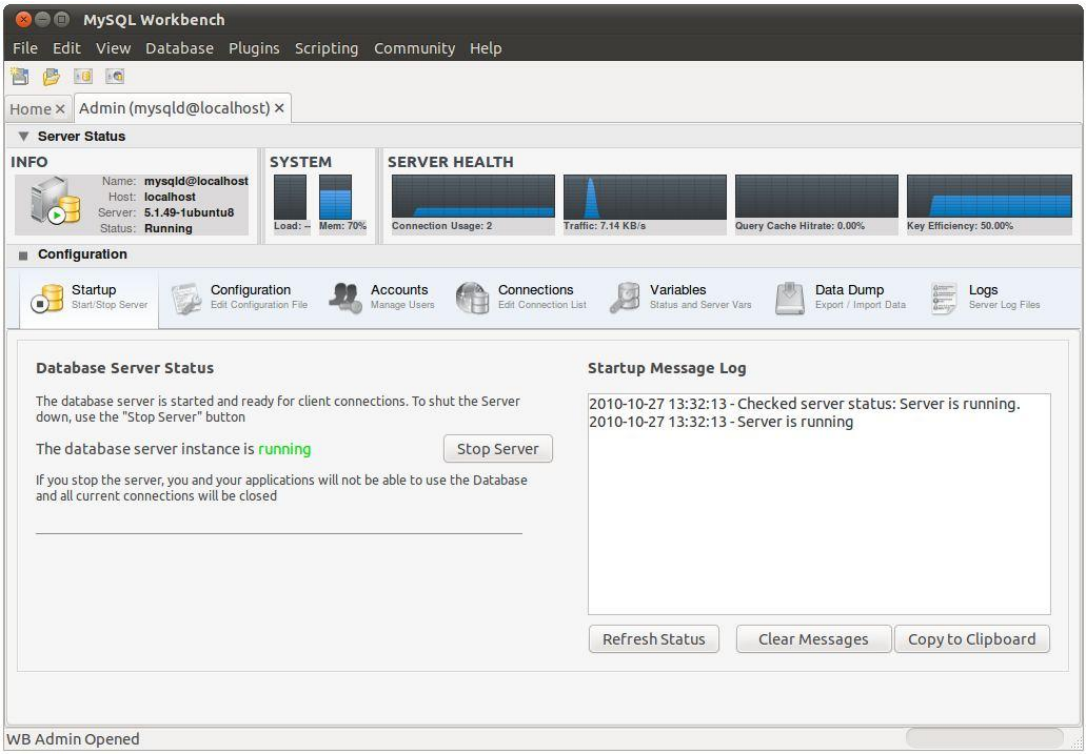
ბოლოს, ვნახულობთ სერვერის ამ ეგზემპლარის სახელს და მისი შექმნის პროცესის დასრულებისათვის ვაწკაპუნებთ ღილაკზე Finish. ახლადშექმნილი სერვერის ეგზემპლარი უკვე აისახება ჩამონათვალში Server Administration სვეტში. შემდეგი ბიჯი იქნება სერვერთან მიერთება.



სურ. 12.16

MySQL Database Server -თან მიერთება

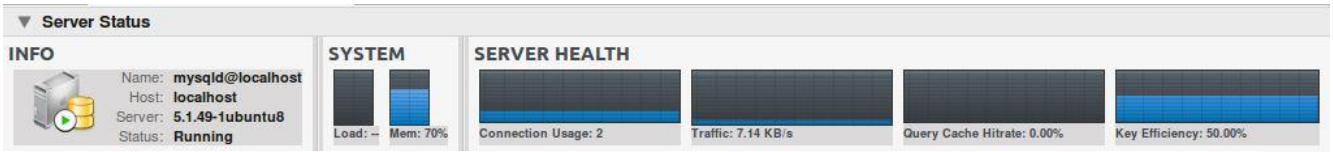
ადმინისტრირების ამოცანების შესრულებისათვის Admin Panel - ის Home ჩანართზე ორჯერ ვაწკაპუნებთ საჭირო სერვერის ეგზემპლარზე.



სურ. 12.17

MySQL Workbench Admin Panel შეიცავს მონიტორინგისა და კონფიგურაციის შემდეგ ოპციებს:

Server Status ასახავს ინფორმაციას სერვერის თაობაზე, რომელთანაც workbench მოცემულ მომენტში არის მიერთებული:

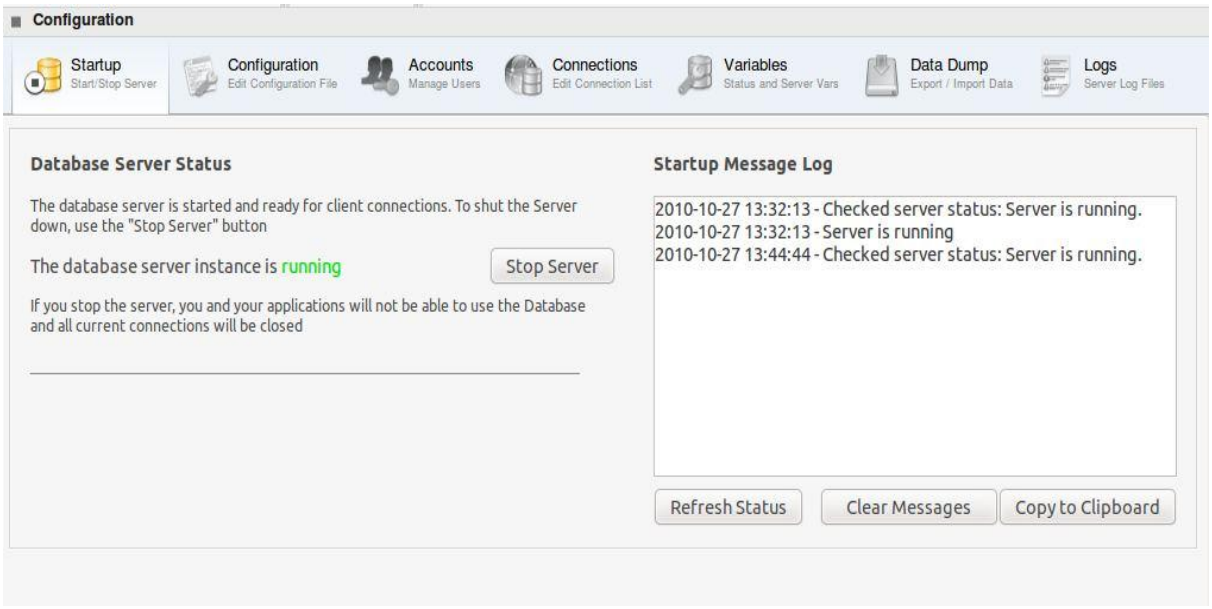


სურ. 12.18

Server Status პანელი იყოფა შიდა სამ ქვესექციად:

- ***INFO*** .
- ***SYSTEM***.
- ***SERVER HEALTH***.

Configuration ასახავს სხვადასხვა ინფორმაციას მიერთებების, წვდომის თაობაზე და აგრეთვე კონფიგურაციის ცვლილებების შესახებ.



სურ. 12.19

Configuration -ის კატეგორიებს მიეკუთვნება:

▪ **Startup** –უზრუნველყოფს მონაცემთა ბაზის სერვერის გაჩერება/გაშვებას.

▪ **Configuration** – უზრუნველყოფს აწყობის პროცესს, რომელიც აისახება my.cnf კონფიგურაციის ფაილში.

▪ **Accounts** – მომხმარებლის ე.წ. ანგარიში, სადაც ხდება მისი სააღრიცხვო ჩანაწერების დამატება ან ამოგდება, აგრეთვე პრივილეგიების ცვლილებები.

▪ **Connections** – მონაცემთა ბაზის სერვერთან მიმდინარე მიერთებების ჩამონათვალის წარმოება.

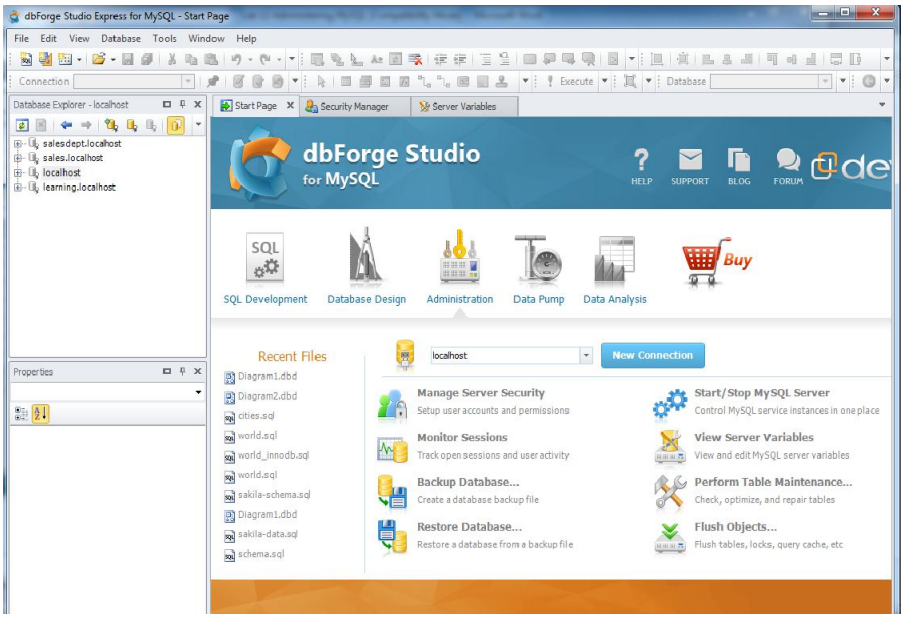
▪ **Variables** – უზრუნველყოფს MySQL Server - ში და ცვლადებისადმი წვდომას.

▪ **Data Dump** – უზრუნველყოფს ფაილში მონაცემთა ბაზისა და ცხრილების მონაცემების გადატანას ან ფაილიდან მათ აღდგენას.

▪ **Logs** – უზრუნველყოფს წვდომას MySQL სერვერის რეგისტრაციის ჟურნალის ფაილთან, რომელთანაც workbench არის მიერთებული.

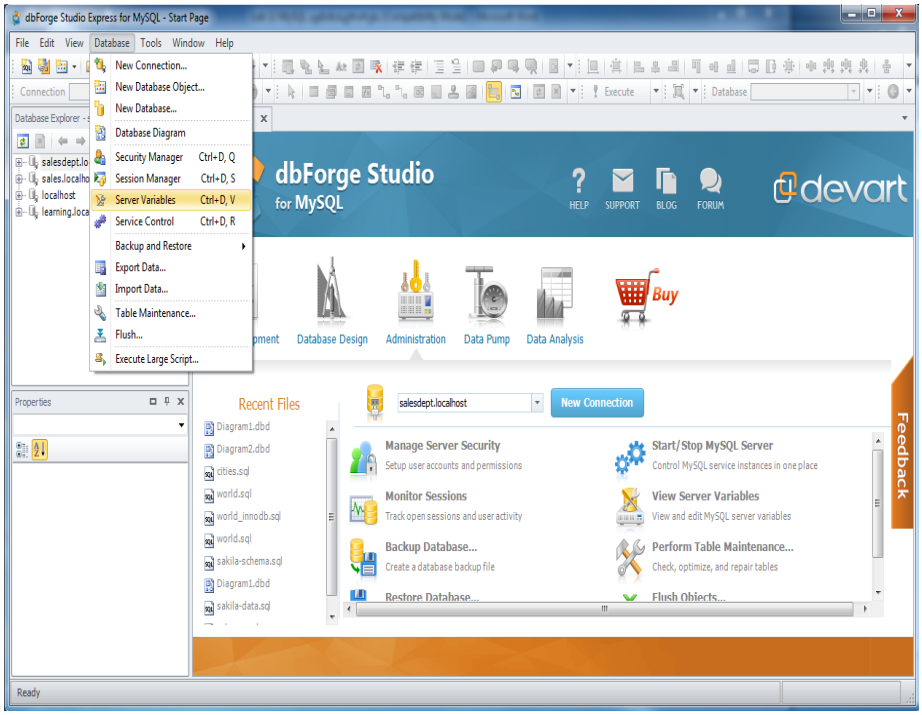
4.ადმინისტრირების ამოცანები dbForge Studio for MySQL-ში

ამ განყოფილებაში ვისაუბრებთ მონაცემთა ბაზების ადმინისტრირების ზოგიერთ ამოცანაზე, როგორიც არის სერვერის ცვლადების კონტროლი, სერვერის მენეჯმენტი და სხვ.



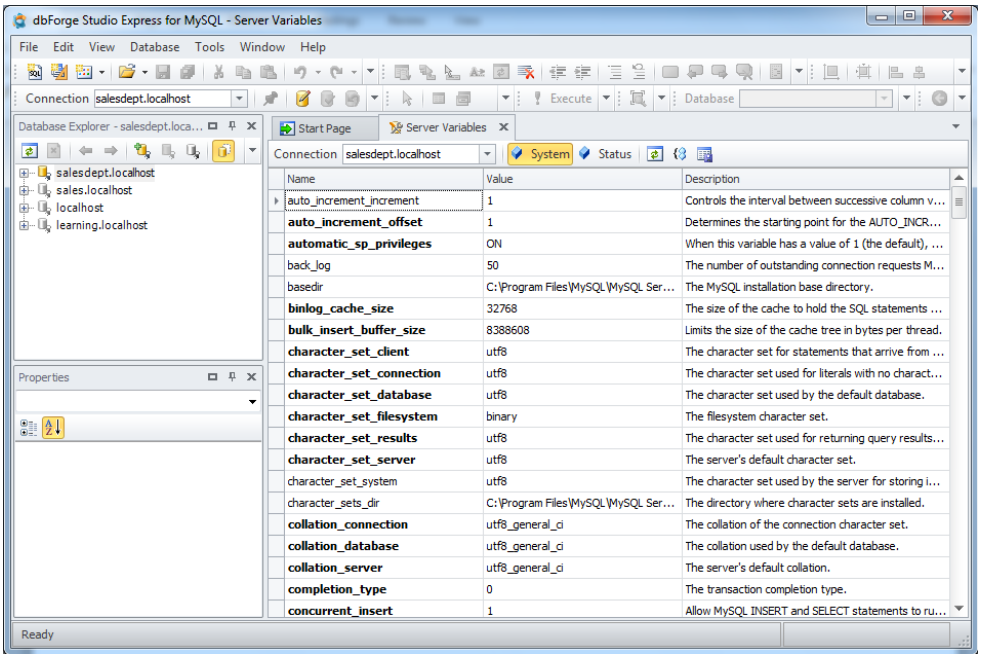
სურ. 12.20

Server Variables. Server Variables ფანჯრის გახსნისათვის **Database** მენიუში ვირჩევთ **Server Variables**.

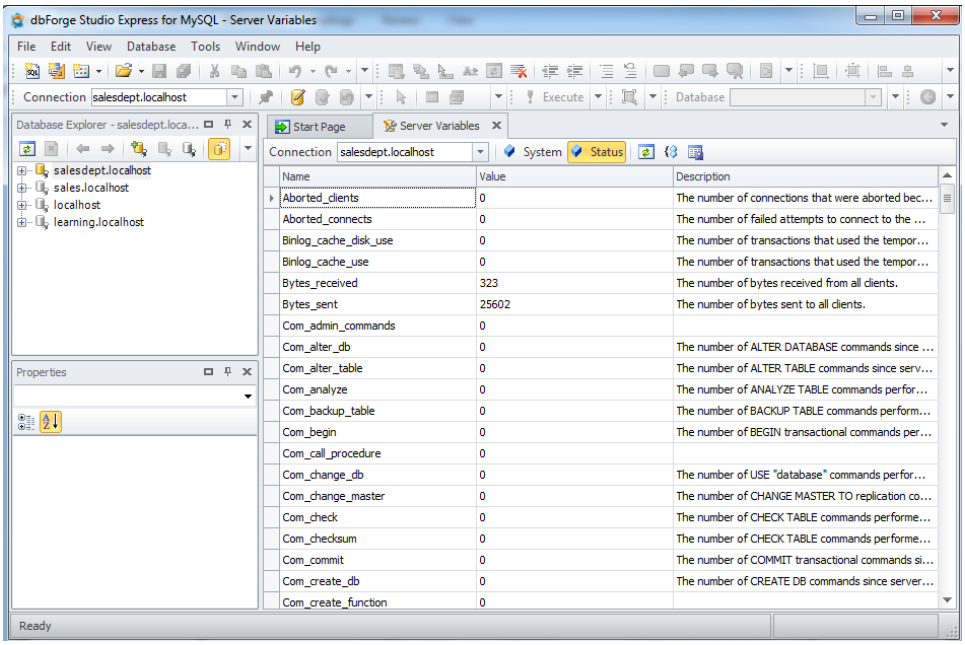


სურ. 12.21

System variables მიუთითებს თუ როგორ არის MySQL სერვერი კონფიგურირებული. სერვერის მუშაობის პროცესში ზოგიერთი მათგანი შეიძლება დინამიურად შეიცვალოს. სისტემასა და ცვლადების სტატუსს შორის ჩართვისათვის ვაწკაპუნებთ ღილაკებზე **System** ან **Status**.



სურ. 12.22

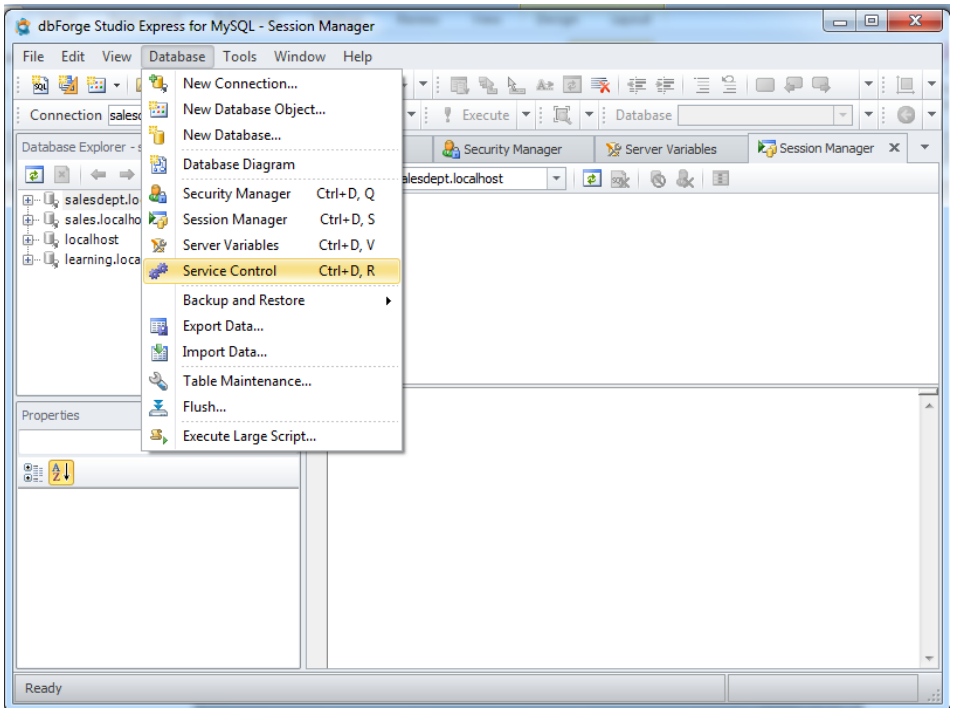


სურ. 12.23

ცვლადების მნიშვნელობათა შეცვლისთვის ვაწკაპუნებთ Value სვეტში საჭირო სტრიქონზე ცვლადის რედაქტირებისათვის. ცვლილებათა გამოყენებისათვის ვაჭერთ ENTER, ან გაუქმებისათვის ვაჭერთ ESC.

Managing Services - სერვისების მართვა

dbForge Studio for MySQL -ში სერვისების მართვისათვის **Database** მენიუში ვირჩევთ **Service Control**.

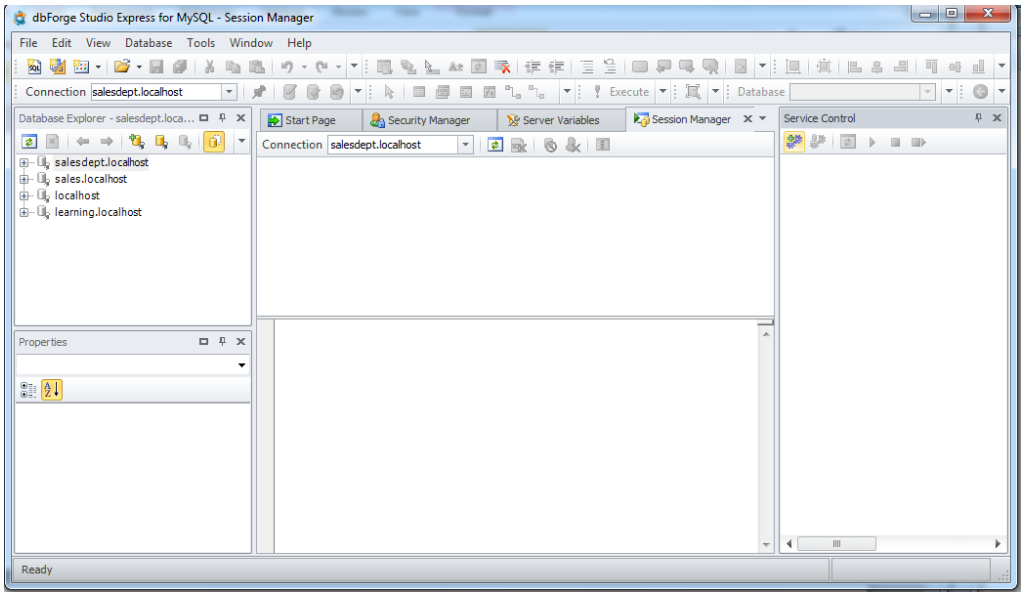


სურ. 12.24

სერვისების მართვისათვის საჭიროა მათი დამატება Service Control სიაში.

სერვისების დამატება Service Control სიაში

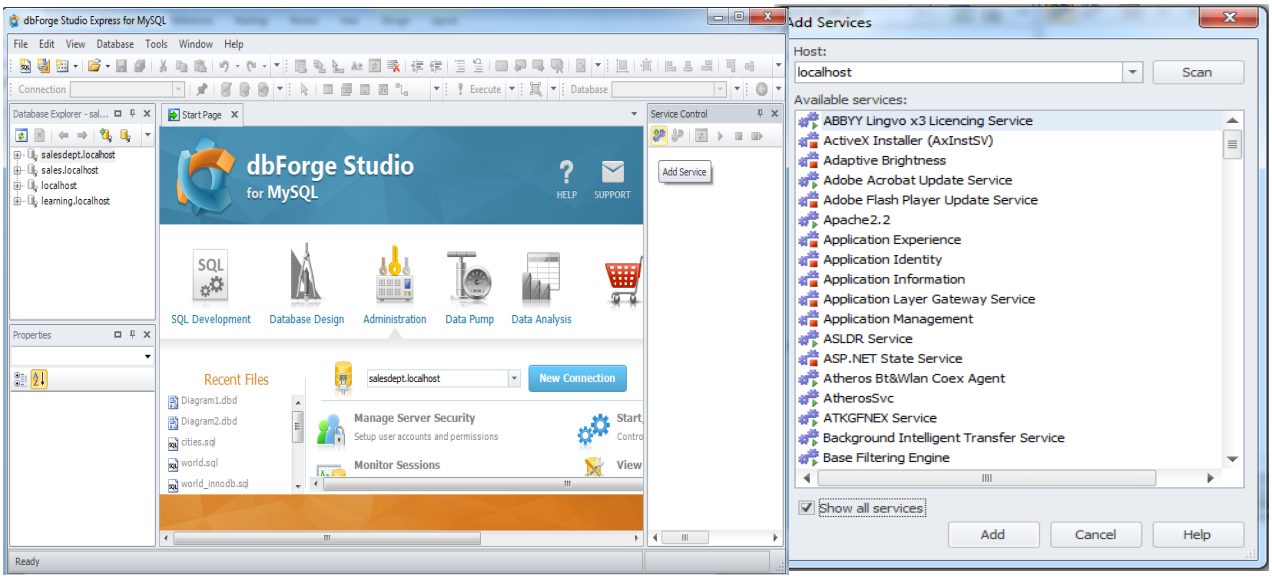
1.ვაწკაპუნებთ ღიალკზე **Add Service**.



სურ. 12.25

2. ჯერ უნდა განვსაზღვროთ საჭირო Host.

3. ვირჩევთ **Show all services**, თუ გვჭირდება სერვისების დაყენება, რომელთა სახელი არ შედის ჩამონათვალში.



სურ. 12.26

4. ვაწკაპუნებთ ღილაკზე **Scan**. თუ ჩვენ არასაკმარისი პრივილეგიები გაგვაჩნია, მივიღებთ შესაბამის მესიჯს.

5. ვირჩევთ საჭირო სერვისებს და ვაწკაპუნებთ ღილაკზე **Add**.

Associating Connections with a Service

მიერთებები შეიძლება დაკავშირებული იყოს სერვისთან. დაკავშირების შემდეგ თუ მიერთება დაიხურება სერვისი შეჩერდება. მიერთების დაკავშირებისათვის საჭიროა შემდეგი ბიჯების შესრულება:

1. ვაწკაპუნებთ ღილაკზე **Associated Connections**.
2. ვირჩევთ საჭირო მიერთებებს.
3. ვაწკაპუნებთ **OK**.

სერვისების მანიპულირება Server Control List -ში

Service Control list-ში საჭირო სერვისების დამატების შემდეგ, ჩვენ შეგვიძლია ისინი გავუშვათ, შევაჩეროთ და გადავტვირთოთ შესაბამის ღილაკზე დაწკაპუნებით.

სერვისების მდგომარეობათა გასურლებისათვის გამოიყენება ღილაკი **Refresh**.

Service Control list-დან სერვისის ამოგდებისათვის ვირჩევთ მას და ვაჭერთ DELETE ან მარჯვენა დაწკაპუნების შემდეგ popup მენიუდან ვირჩევთ **Remove service.**

Controlling სესიაs - სესიების მართვა

მომხმარებლის სესიების მართვისათვის უნდა ვსუროთ სესიის ინფორმაცია, გაჩერების მოთხოვნის შესრულება და სესიების გათიშვა.

წინაპირობები

- MySQL server 5.0.13 ან უფრო მაღალი ვერსია.
- უნდა შეიქმნას მონაცემთა ბაზა მინიმუმ ორი ცხრილით.

სესიის დაწყება

შევასრულოთ მოთხოვნა

```
SELECT * FROM
demobase.emp e1,
demobase.emp e2,
demobase.emp e3,
demobase.emp e4,
demobase.emp e5
```

შემდეგი ბიჯების თანამიმდევრობით:

1. **Standard** ინსტრუმენტების სტრიქონში **New SQL** ღილაკზე დაწკაპუნებით გავხსნათ SQL რედაქტორი.

2. ავკრიფოთ ან Copy/Paste -ით შევიტანოთ კოდი დოკუმენტზე.

3. **SQL** ინსტრუმენტების სტრიქონში **New SQL** ღილაკზე ❗ დაწკაპუნებით შევასრულოთ დოკუმენტი.

სესიების მართვა

Database მენიუში ვირჩევთ **სესია Manager**.

ვაწკაპუნებთ Host სვეტის დასახელებაზე.

ვპოულობთ ჩვენს მიერთებებს (ჩვენი host-დან).

ბრძანება სვეტში ვაწკაპუნებთ მიერთებაზე, რომელსაც აქვს 'Query'.

სესია Manager -ის ქვედა ნაწილში ჩვენ დავინახავთ შესრულებაში მყოფ მოთხოვნის ტექსტს. ვპოულობთ ამ მოთხოვნასთან ჩვენს მიერთებებს.

სესიისა და მოთხოვნის ლიკვიდაცია (Killing)

ჩვენი მოთხოვნის სესიაზე მარჯვენა დაწკაპუნებით გამოდის popup მენიუ, სადაც ვირჩევთ **Kill Query**. შემდეგ ვაწკაპუნებთ **Yes** დიალოგურ ფანჯარაში.

დავუკავშირდეთ მონაცემთა ბაზის წინა ეგზემპლიარს dbForge Studio - ში, სადაც ქვემოთ გადავახვიოთ მონაცემთა ბადე. მივიღებთ მესიჯს "Query execution was interrupted".

შევაერთოთ dbForge Studio for MySQL და სესია Manager. კვლავ სესიაზე მარჯვენა დაწკაპუნებით იხსნება popup მენიუ, საიდანაც ვირჩევთ **Kill სესიას.**

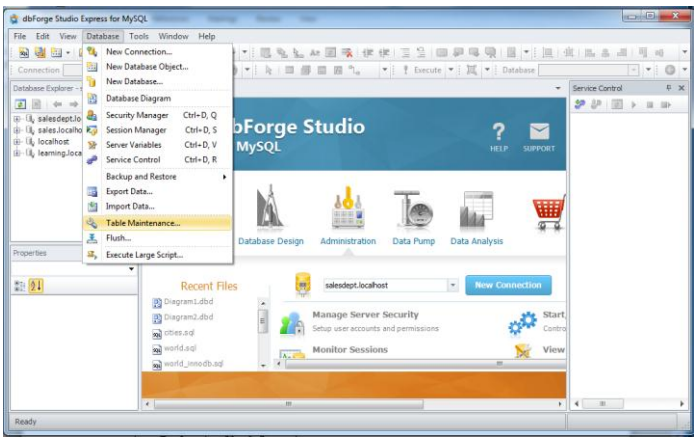
გადავერთოთ dbForge Studio for MySQL -ს წინა ეგზემპლიარზე და Database Explorer ინსტრუმენტალურ სტრიქონში ვაწკაპუნებთ ღილაკზე **Refresh**. მივიღებთ მესიჯს "Lost connection during query".

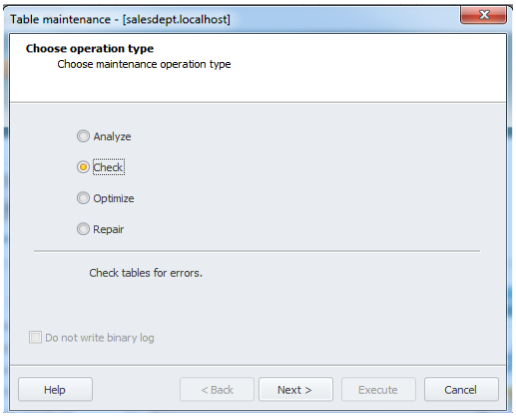
თუ ჩვენ გვაქვს გლობალური SUPER პრივილეგია, მაშინ შეგვიძლია ყველა

მომხმარებლის მოთხოვნებისა და სესიების ლიკვიდაცია, გარდა საკუთარი მოთხოვნებისა და სესიებისა.

Performing Maintenance and Flush Operations

dbForge Studio for MySQL -ში Table Maintenance Wizard -ის გახსნისათვის **Database** მენიუში ვირჩევთ **Table Maintenance**.





სურ. 12.27

ანალიზის ოპერაციის შესრულება

ცხრილების ან წარმოდგენების ანალიზისათვის სრულდება შემდეგი ქმედებები:

1. გავხსნათ Table Maintenance Wizard.
2. შეგვიძლია ავირჩიოთ **Do not write binary log**, თუ არ გვინდა ანალიზის ოპერაციის ჩატარება აღნიშნული ხერხით.
3. ვაწკაპუნებთ **Analyze** და შემდეგ ვაწკაპუნებთ ღილაკზე **Next**.
4. ვირჩევთ სანახველ ცხრილებს და ვაწკაპუნებთ ღილაკზე **Execute**.

ძებნის ოპერაციის შესრულება

ცხრილების ან წარმოდგენების ძებნისათვის სრულდება შემდეგი ქმედებები:

1. გავხსნათ Table Maintenance Wizard.
2. ვაწკაპუნებთ **Check** და შემდეგ ვაწკაპუნებთ **Next** ღილაკზე.
3. ვირჩევთ სანახველ ცხრილებსა და წარმოდგენებს ძებნისათვის და ვაწკაპუნებთ ღილაკზე **Next**.
4. ვირჩევთ ძებნის ხერხს და ცხრილებს ძებნისათვის და ვაწკაპუნებთ ღილაკზე **Execute**.

ოპტიმიზაციის ოპერაციის შესრულება

ცხრილების ან წარმოდგენების ოპტიმიზაციისათვის სრულდება შემდეგი ქმედებები:

1. გავხსნათ Table Maintenance Wizard.
2. შეგვიძლია ავირჩიოთ **Do not write binary log**, თუ არ გვინდა ანალიზის ოპერაციის ჩატარება აღნიშნული ხერხით.

3. ვაწკაპუნებთ **Optimize** და შემდეგ ვაწკაპუნებთ ღილაკზე **Next**.

4.Select tables you want to optimize და ვაწკაპუნებთ ღილაკზე **Execute**.

შეკეთების ოპერაციის შესრულება

ცხრილების ან წარმოდგენების შეკეთებისათვის სრულდება შემდეგი ქმედებები:

1. გავხსნათ Table Maintenance Wizard.

2. შეგვიძლია ავირჩიოთ **Do not write binary log**, თუ არ გვინდა ანალიზის ოპერაციის ჩატარება აღნიშნული ხერხით.

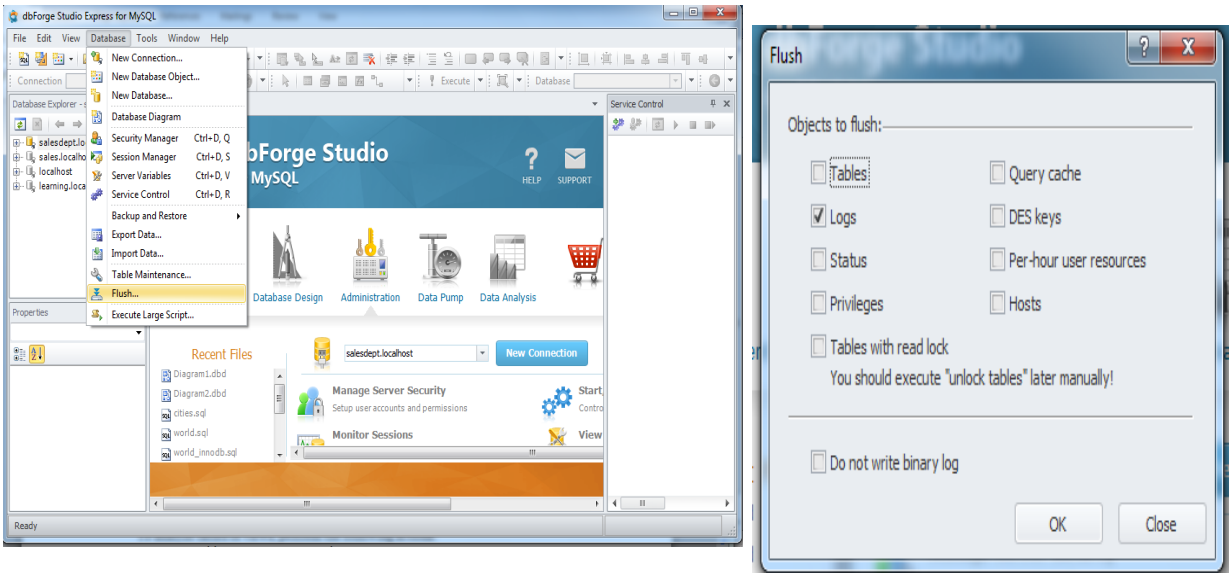
3. ვაწკაპუნებთ **Repair** და შემდეგ ვაწკაპუნებთ ღილაკზე **Next**.

4. ვირჩევთ სანახველ ცხრილებსა და წარმოდგენებს შეკეთებისათვის და ვაწკაპუნებთ ღილაკზე **Next**.

5. ვირჩევთ შეკეთების ოპციებს და ვაწკაპუნებთ ღილაკზე **Execute**.

გასუფთავების ოპერაციის შესრულება

Flush დიალოგური ფანჯრის გახსნისათვის **Database** მენიუში ვირჩევთ **Flush**.



სურ. 12.28

ვირჩევთ სანახველ ობიექტს და ვაწკაპუნებთ ღილაკზე **OK**.

flush ოპერაციის შესრულებისათვის Database Explorer-ში ერთერთ ცხრილზე მარჯვენა დაწკაპუნებით იხსნება popup მენიუ, სადაც ვირჩევთ **Flush**.

5. უსაფრთხოების სისტემა

MySQL მონაცემთა ბაზების უსაფრთხოების სისტემა ემყარება შემდეგ ძირითად ცნებებს:

- აუტენტიფიკაციის (authentication) გამოყენების ერთ-ერთ მთავარ არსს წარმოადგენს არასანქცირებული მომხმარებლის სისტემაში შეღწევის დროული გამოვლინება და თავიდან აცილება. ეს პროცესი შეიძლება მნიშვნელოვნად გაძლიერდეს დაშიფვრის გამოყენებით.
- ავტორიზაცია (authorization) გამოიყენება მომხმარებლის (აუტენტიფიციკაციის შემოწმების შემდეგ) იდენტიფიკაციის მიზნით, როდესაც სისტემა განსაზღვრავს რომელ რესურსებთან წვდომის ნებართვები ენიჭება კონკრეტულ მომხმარებელს.

პრივილეგიების მინიჭება

ბრძანება GRANT გამოიყენება, როცა საჭიროა ახალი პრივილეგიების მინიჭება:

GRANT *privilege_type*
[(*column_list*)] [, *privilege_type* [(*column_list*)] ...]
ON {*table_name* | * | *.* | *database_name*.*}
TO *user_name* [IDENTIFIED BY '*password*']
[, *user_name* [IDENTIFIED BY '*password*'] ...]
[REQUIRE {SSL|X509} [ISSUER *issuer*] [SUBJECT *subject*]]
[WITH GRANT OPTION]

პრივილეგიების გაუქმება

ბრძანება REVOKE გამოიყენება მომხმარებლისათვის ან მომხმარებელთა ჯგუფისათვის ადრე მინიჭებული პრივილეგიების გასაუქმებლად:

REVOKE *privilege_type [(column_list)]*
[, privilege_type [(column_list)] ...]
ON {*table_name* | * | *.* | *database_name*.*}
FROM *user_name* [, *user_name* ...]

6. მომხმარებელთა მართვა MySQL -ში

მომხმარებელთა დამატება

პრივილეგიის გარეშე მომხმარებელთა დამატება, შესაბამისად ახალი კლიენტური ანგარიშის (user account) გახსნა შესაძლებელია CREATE USER ბრძანების გამოყენებით, რომლის სინტაქსისია:

CREATE USER *user* [IDENTIFIED BY [PASSWORD] '*password*']
[, *user* [IDENTIFIED BY [PASSWORD] '*password*']] ...

ხოლო მისთვის გარკვეული პრივილეგიების მინიჭება ხდება ბრძანებით:

GRANT <permissions> ON <objects>

TO 'user'@'localhost' IDENTIFIED BY 'password';

მომხმარებელთა წაშლა

ბრძანება DROP USER გამოიყენება მომხმარებლის წაშლისათვის ყველა თავის პრივილეგიებთან ერთად:

DROP USER *user* [, *user*]...

მომხმარებლების გადარქმევა

არსებული მომხმარებლის სახელის გადარქმევისათვის შეიძლება ბრძანების RENAME USER გამოყენება:

RENAME USER *old_user* TO *new_user*

[*old_user* TO *new_user*]...

მომხმარებელთა პრივილეგიების ნახვა

MySQL მომხმარებელთა პრივილეგიების შესახებ ინფორმაცია მიიღება SHOW GRANTS ბრძანების მეშვეობით, რომლის სინტაქსისი შემდეგია:

SHOW GRANTS FOR *username*;

7. მომხმარებლებისა და პრივილეგიების მართვა dbForge Studio გამოყენებით

MySQL - ში განირჩევა კლიენტური ანგარიშის (user account) რამდენიმე ბაზისური კონცეპტი.

მომხმარებლის სახელი (User Names).

მომხმარებლის სახელი შეიძლება იყოს 16 სიმბოლოზე მეტი სიგრძის.

ჰოსტი (Host).

MySQL-ში მომხმარებელი იდენტიფიცირდება ორი მონაცემით: მომხმარებლის სახელისა და ჰოსტის სახელის მიხედვით, რომელთანაც არის მიერთებული. როცა ვქმნით ახალ კლიენტურ ანგარიშს, საჭიროა ჰოსტის განსაზღვრაც. ჩვენ შეგვიძლია დავაყენოთ ჰოსტი სახელთან (მაგ., myserver.com) ან IP მისამართის მიხედვით. შეგვიძლია გამოვიყენოთ სპეციალური სიმბოლო "%" ჰოსტის სახელში ან IP -ში. მაგალითად, ჰოსტის სახელი '%.mydomain.com' დომეინიდან 'mydomain.com'. '144.155.166.%' ნიშნავს 144.155.166 კლასის C ქვექსელის ნებისმიერ ჰოსტს.

პრივილეგიები (Privileges)

მომხმარებელს შეიძლება გააჩნდეს მონაცემთა ბაზების ობიექტების სხვადასხვა დონის (მონაცემთა ბაზის, ცხრილის, სვეტის, პროგრამის) პრივილეგია.

კლიენტური ანგარიშის რესურსები (account resources)

MySQL - ში შეიძლება გვქონდეს გარკვეული შეზღუდვები კლიენტური

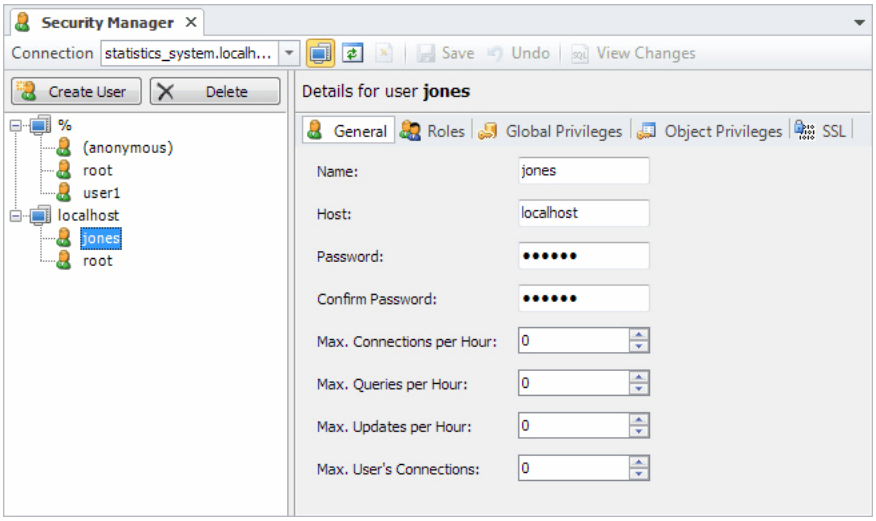
ანგარიშის რესურსებზე. მაგალითად, მიერთებათა რაოდენობის მაქსიმუმი, მოთხოვნათა რაოდენობა საათში და ა.შ.

8. ახალი კლიენტური ანგარიშის დაყენება

ახალი კლიენტური ანგარიშის შექმნა

განვიხილოთ კლიენტური ანგარიშის შექმნის პროცედურა კონკრეტულ მაგალითზე.

1. Database Explorer -ში უნდა შევამოწმოთ ჩვენი მიერთების ჩართულობა და აქვს თუ არა ჩვენს მონაცემთა ბაზას ("Demobase") საკმარისი პრივილეგიები.
2. **Database** მენიუდან ვირჩევთ **Security Manager**.
3. მომხმარებელთა სიის თავზე ვაწკაპუნებთ ღილაკზე **Create User**.
4. შეგვაქვს მომხმარებლის სახელი (მაგალითად, "Jones") და ჰოსტი (მაგალითად, 192.168.0.1) შესაბამის ველებში.
5. შეგვაქვს მომხმარებლის პაროლი ველებში **Password** და **Confirm Password** ან ვტოვებთ კლიენტური ანგარიშის შექმნის ბლანკს პაროლის გარეშე.
6. ვხსნით ჩანართს **Global Privileges**.
7. ვამოწმებთ **Select** პრივილეგიას.
8. Security Manager ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**.



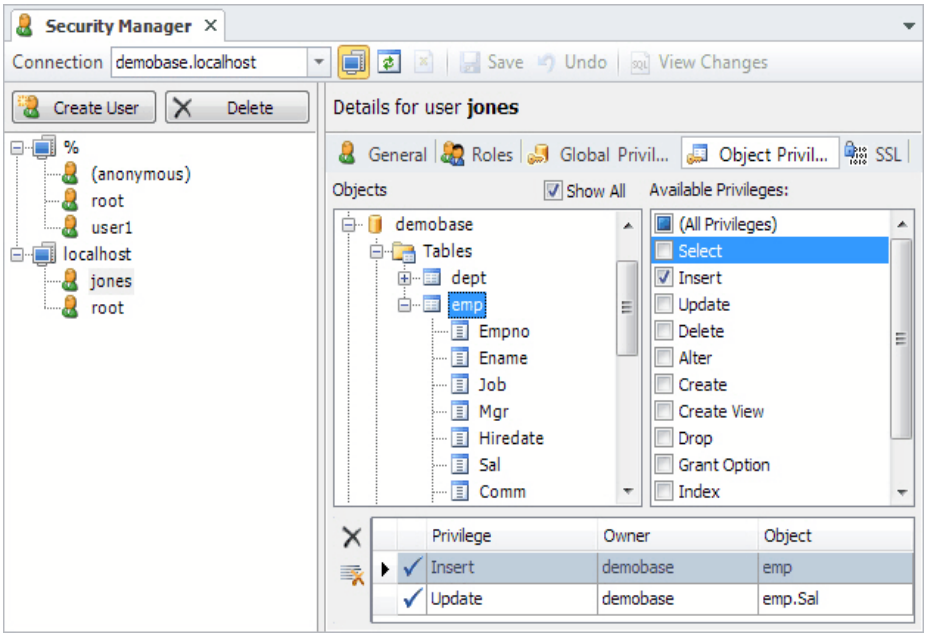
სურ. 12.29

მონაცემთა ბაზის სპეციფიურ ობიექტებისათვის (Specific Database Objects) პრივილეგიების მინიჭება

1. Database Explorer -ში უნდა შევამოწმოთ ჩვენი მიერთების ჩართულობა და აქვს თუ არა ჩვენს მონაცემთა ბაზას ("Demobase") საკმარისი პრივილეგიები.
2. Database Explorer-ის ხეზე ცხრილისათვის (მაგალითად, 'emp') popup მენიუდან ვირჩევთ **Edit Privileges** ან ვხსნით Security Manager -ის **Object**

Privileges ჩანართს და **Objects** ხეზე ვირჩევთ ცხრილს 'emp' table node in the **Objects** tree.

3. ვირჩევთ კლიენტური ანგარიშს (Jones@<hostname>).
4. ვირჩევთ ცხრილს 'emp' მონაცემთა ბაზის ობიექტების სიიდან.
5. ვამოწმებთ **Insert** ალამს მარჯვენა სიაში.
6 . გავხსნათ ცხრილის 'emp' კვანძი.
7. ვირჩევთ სვეტს 'Sal'.
8. ვამოწმებთ **Update** ალამს მარჯვენა სიაში.
9. Security Manager ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**.

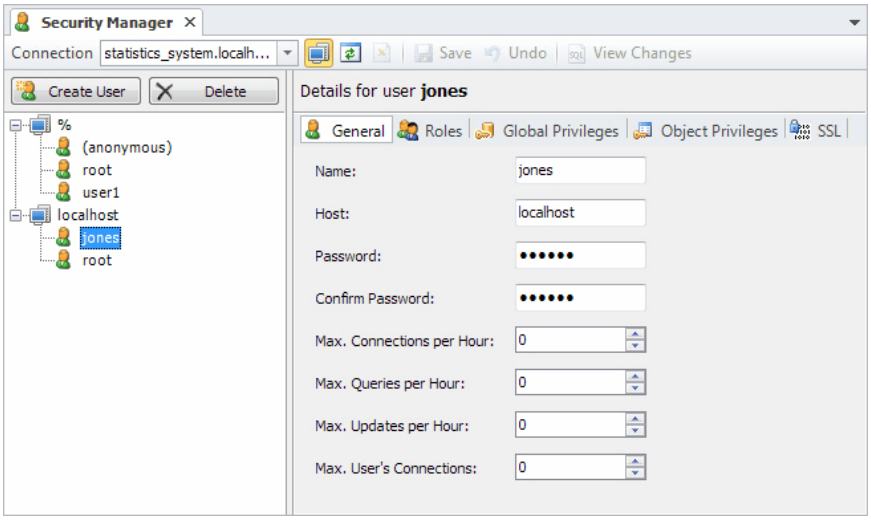


სურ. 12.30

9. კლიენტური ანგარიშების მანიპულირება Security Manage-ში

Security Manager უზრუნველყოფს ადვილად დავამატოთ, შევცვალოთ ან ამოვაგდოთ მომხმარებელი ვიზუალური ხერხით ყოველგვარი SQL კოდის აკრეფვის გარეშე.

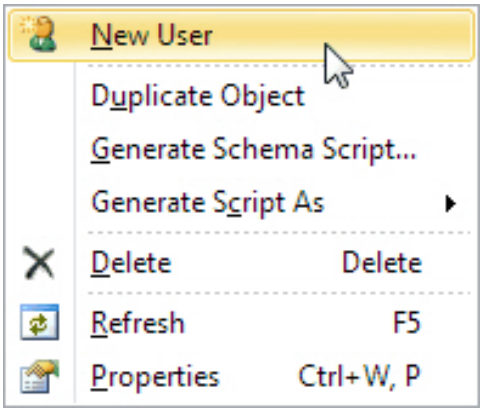
Security Manager -ის გახსნისათვის, **Database** მენიუდან ვირჩევთ **Security Manager**.



სურ. 12.31

კლიენტური ანგარიშის შექმნა

1. გავხსნათ Security Manager.
2. User List -ის ზევით ვაჭერთ ღილაკს **Create User** ან მარჯვენათი ვაწკაპუნებთ მომხმარებლის კვანძზე და კონტექსტურ მენიუში ვირჩევთ **New User**.



სურ. 12.32

3. ვავსებთ ველებს **Name**, **Host**, **Password**, და **Confirm Password**.
4. ვანიჭებთ აუცილებელ პრივილეგიებს მომხმარებელს.
5. ოპციების მეშვეობით მომხმარებლისათვის ვაყენებთ შეზღუდვებს კლიენტური ანგარიშის რესურსებზე..
6. **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**.

მომხმარებელთა დუბლირება

მომხმარებელთა დუბლირებისათვის **Security Manager**-ში ორჯერ ვაწკაპუნებთ მომხმარებელზე და კონტექსტურ მენიუში ვირჩევთ **Duplicate Object.** რედაქტორის მარჯვენა ნაწილი უნდა შეიცავდეს ახალ მომხმარებელს, რომელიც იქნება არსებულის ზუსტი ასლი გარდა სახელისა. შევცვალოთ მომხმარებელი, თუ

გვჭირდება და შევისუროთ. აღსანიშნავია, რომ მომხმარებელი არ შეიქმნება ცვლილებების შენახვამდე ანუ **Save** ბრძანებამდე.

კლიენტური ანგარიშის რედაქტირება

გავხსნათ **Security Manager** და ხეზე შევირჩიოთ საჭირო კლიენტური ანგარიში. რედაქტირების შემდეგ **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**. თუ გვინდა ცვლილებების გაუქმება **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Undo**. ჩვენ შეგვიძლია შეცვლილი SQL ბრძანებების ნახვა, რისთვისაც **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **View Changes.**.

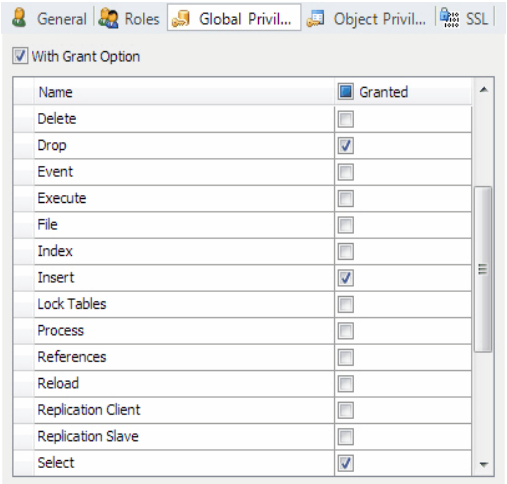
კლიენტური ანგარიშის წაშლა

გავხსნათ **Security Manager**. კლიენტურ ანგარიშზე მარჯვენა დაწკაპუნებით იხსნება კონტექსტური მენიუ, სადაც ვირჩევთ **Delete** ან სიაში ვირჩევთ საჭირო მომხმარებელს და ვაჭერთ ღილაკს **Delete** მომხმარებელთა სიის (User List) ზემოთ ან ვაჭერთ **DELETE** კლავიატურაზე.

გლობალური პრივილეგიების მინიჭება და გაუქმება

Security Manager -ის მეშვეობით ადვილად არის შესაძლებელი გლობალური პრივილეგიების მინიჭება და გაუქმება.

1. გავხსნათ **Security Manager**.
2. **Security Manager** -ის მარცხენა ნაწილში კლიენტურ ანგარიშის სიიდან ვირჩევთ მომხმარებელს.
3. ვხსნით **Global Privileges** ჩანართს.
4. ვირჩევთ ალამებს საჭირო პრივილეგიების გასწვრივ, ხოლო გაუქმებისათვის შესაბამის ალმებს ვასუფთავებთ.



სურ. 12.33

5.ამორჩევით ვაყენებთ **With Grant Option** ალამს მომხმარებელზე პრივილეგიების მინიჭებისათვის.

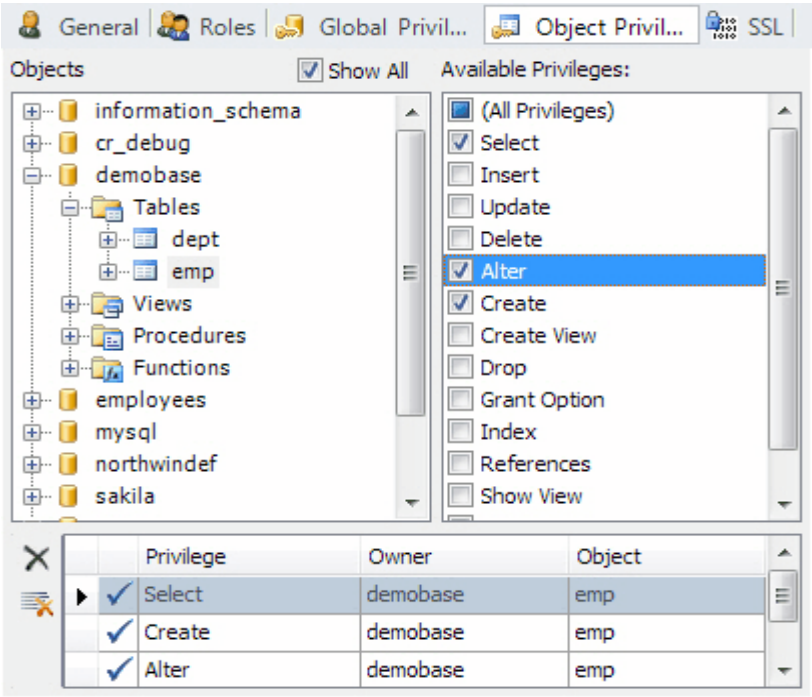
6. ცვლილებების დაფიქსირებისათვის **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**.

10. მონაცემთა ბაზის ობიექტებისათვის პრივილეგიების მინიჭება

Security Manager -ის **Object Privileges** ჩანართში შეგვიძლია მონაცემთა ბაზის ობიექტებს მივანიჭოთ პრივილეგიები. ამისათვის **Database** მენიუში ვირჩევთ **Security Manager** ან **Database Explorer**-ში მონაცემთა ბაზის ობიექტების კონტექსტური მენიუდან ვირჩევთ **Edit Privileges**.

1. ვხსნით **Security Manager**.

2. Security Manager-ის მარცხენა ნაწილში სიიდან ვირჩევთ კლიენტურ ანგარიშს და ვხსნით **Object Privileges** ჩანართს.



სურ. 12.34

3. ვირჩევთ მონაცემთა ბაზის ობიექტს **Objects** ხეზე. ამის შემდეგ ჩვენ დავინახავთ დასაშვები პრივილეგიების სიას.

4. მოვნიშნოთ პრივილეგიები.

5. **Security Manager** ინსტრუმენტების პანელზე ვაჭერთ ღილაკს **Save**.

Security Manager-ის **Object Privileges** ჩანართში ჩვენ შეგვიძლია აგრეთვე შერჩეული პრივილეგიების გაუქმება ორჯერადი დაწკაპუნებით ან Ctrl+Delete

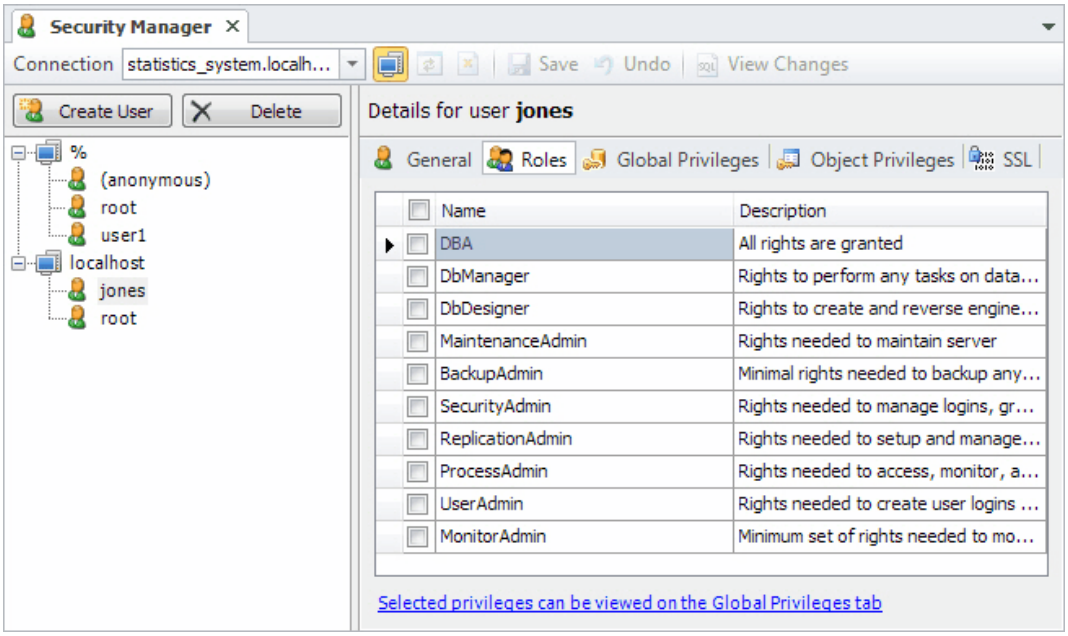
კლავიშების დაჭერით ვხსნით კონტექსტურ მენიუს, სადაც ვაწკაპუნებთ **Revoke Selected Privilege** ან ვაწკაპუნებთ ღილაკზე შემდეგ სიაში.

ყველა მონიშნული პრივილეგიის გასაუქმებლად კონტექსტურ მენიუში ვირჩევთ **Revoke All Privileges** ან ვაწკაპუნებთ ღილაკზე სიის ახლოს.

11. როლების გამოყენება Security Manager-ში

როლების გამოყენება საშუალებას გვაძლევს მომხმარებლები გავაერთიანოთ ერთ ადმინისტრაციულ ერთეულში. განვსაზღვრავთ რა პრივილეგიებს კონკრეტული როლისთვის, ავტომატურად ვაძლევთ იგივე პრივილეგიებს ამ როლის ყველა წევრს. თუ მომხმარებელს უნდა მიენიჭოს იგივე უფლებები, როგორიც მინიჭებული აქვს როლს, მაშინ ეს მომხმარებელი ამ როლში უნდა ჩავრთოთ. ასეთი მდგომა ამარტივებს მონაცემთა ბაზის ადმინისტრირებას.

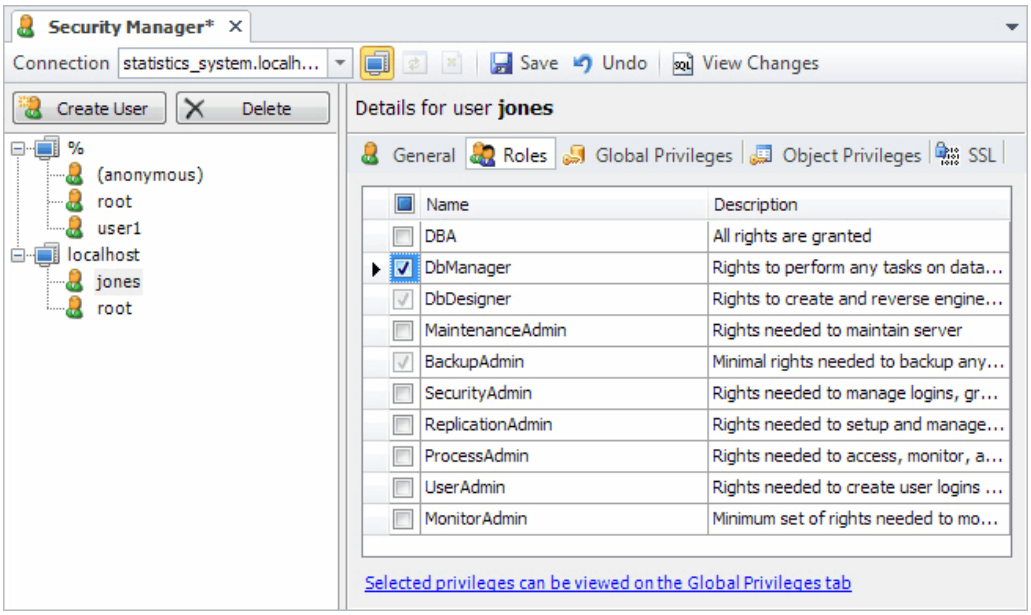
1. მთავარი მენიუდან ვირჩევთ **Database** -> **Security Manager**.
2. **Security Manager** ფანჯრის მარცხენა ნაწილში ვირჩევთ კლიენტურ ანგარიშს.
3. **Security Manager** ფანჯრის მარჯვენა ნაწილში გადავდივართ **Roles** ჩანართზე.



სურ. 12.35

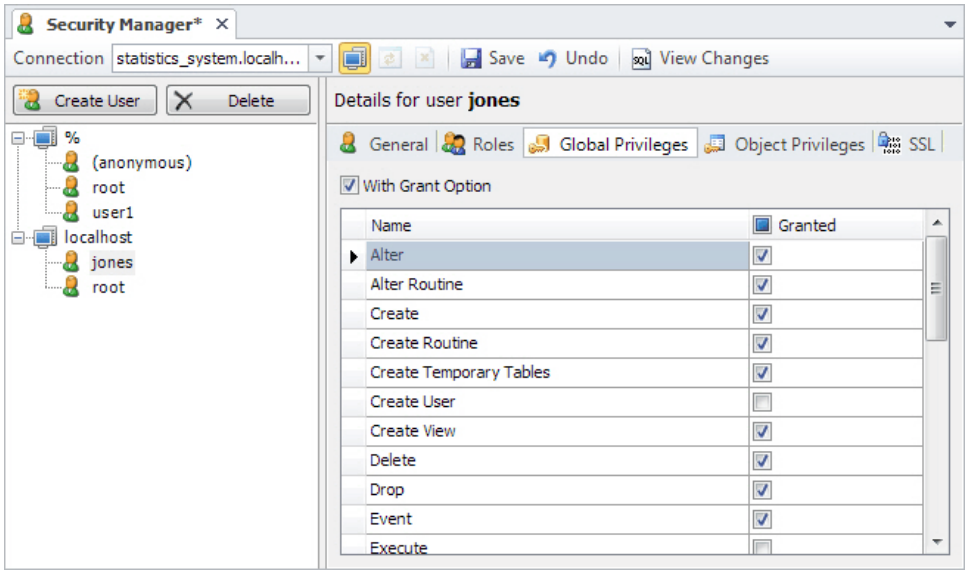
4. ვამოწმებთ საჭირო როლს ან როლებს. ამისათვის ვაყენებთ ალამებს როლების ახლოს:

- შემოწმებულია ანუ ამ როლის პრივილეგიები ჩართულია მომხმარებლის პრივილეგიებში.
- შემოწმებულია, მაგრამ არ არის გააქტიურებული.
- შეუმოწმებელია ანუ ამ როლის პრივილეგიები არ არის მინიჭებული.



სურ. 12.36

5. ყოველი შერჩეული როლისათვის (ან როლებისათვის) მინიჭებული პრივილეგიების ნახვა Global Privileges ჩანართში.



სურ. 12.37

მრავალ მომხმარებელთან მუშაობა

ასეთი მიდგომა მოსახერხებელია, როცა საჭიროა ერთნაირი პრივილეგიების ან როლების დამატება რამდენიმე მომხმარებლისათვის. მრავალი მომხმარებლის შერჩევისათვის ვაჭერთ Ctrl-ს და ამორჩევით დავაწკაპუნებთ საჭირო მომხმარებლებზე. ხოლო თუ მომხმარებლები განლაგებული არიან თანამიმდევრულად, პირველი მომხმარებლის შერჩევისათვის ვაჭერთ Shift და ვაწკაპუნებთ ბოლო მომხმარებელზე.

ზოგიერთი მომხმარებლისათვის შესაძლებელია როლების, გლობალური და ობიექტური პრივილეგიების რედაქტირება შესაბამისად Security Manager ჩანართში.

თუ შევირჩევთ ჰოსტს, მაშინ ეკრანზე მივიღებთ "Cannot show editor because no users are selected or objects of different types are selected" შეტყობინებას.

პრივილეგიების მდგომარეობა განისაზღვრება შემდეგი აღნიშვნებით.

- გლობალური პრივილეგიებისათვის:
  -  - პრივილეგია დადგენილია ყველა შერჩეული მომხმარებლისათვის;
  -  - პრივილეგია დადგენილია ზოგიერთი შერჩეული მომხმარებლისათვის;
  -  - პრივილეგია არ არის დადგენილი არცერთი მომხმარებლისათვის;
- ობიექტური პრივილეგიებისათვის :
  -  - პრივილეგია დადგენილია ყველა შერჩეული მომხმარებლისათვის;
  -  - პრივილეგია არ არის დადგენილი არცერთი მომხმარებლისათვის.

ლაბორატორიული სამუშაო 13
მონაცემთა ბაზის პროექტის შემუშავება

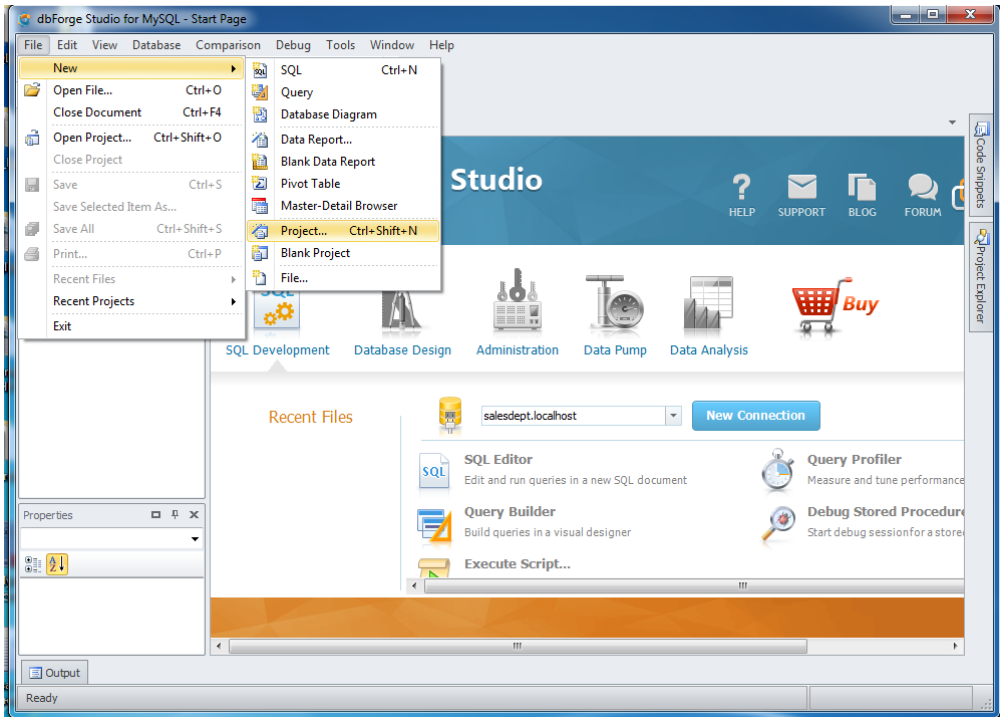
სამუშაოს მიზანი:

1. მონაცემთა ბაზის პროექტთან მუშაობა
2. ფაილების მართვა პროექტში
3. მუშაობა მონაცემთა ბაზის ობიექტებთან პროექტში
4. პროექტის აგება და განთავსება
5. მუშაობა პროექტთან
6. მონაცემთა ანალიზი
7. Pivot Table კომპონენტები
8. მონაცემთა ანგარიშები
9. სტატიკური ანგარიშის შექმნა
10. მარტივი ანგარიშის შექმნა
11. Master-Detail Report -ის შექმნა
12. ანგარიშების შენახვა და ჩატვირთვა

1.მონაცემთა ბაზის პროექტთან მუშაობა

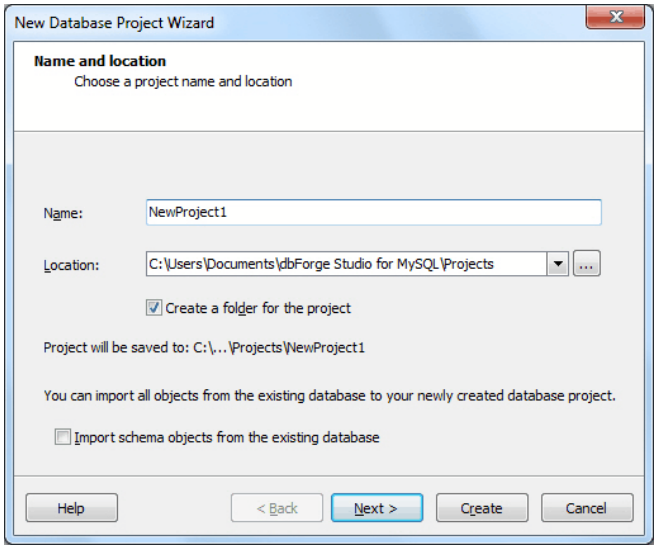
პროექტის შექმნა:

1. მენიუდან ვირჩევთ **File->New->Project**.



სურ. 13.1

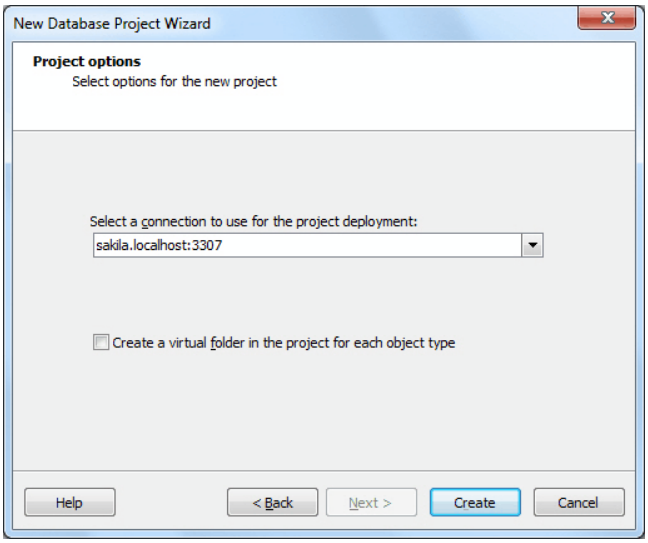
გაიხსნება**New Database Project Wizard**.



სურ. 13.2

2. შეგვაქვს პროექტის სახელი და განთავსების ადგილი.

3. ჩვენ შეგვიძლია აგრეთვე ავირჩიოთ და გამოვიყენოთ მიერთება პროექტის განთავსების მიზნით.



სურ. 13.3

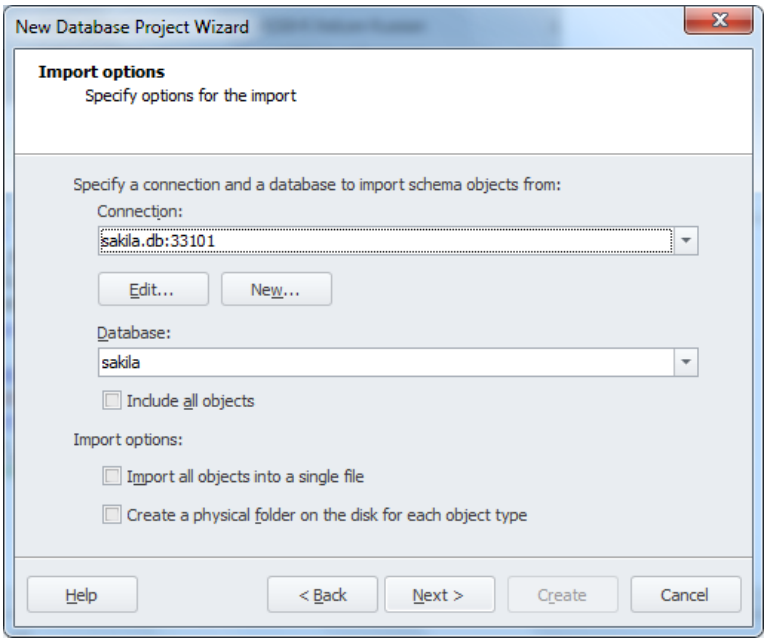
4. ვაწკაპუნებთ **Create** ღილაკზე. პროექტი გამოჩნდება **Project Explorer** ფანჯარაში. ვაწკაპუნებთ ღილაკზე **Save** ინსტრუმენტების სტრიქონში.

ცარიელი პროექტის შექმნა

File მენიუში ვირჩევთ **New -> Blank Project.**

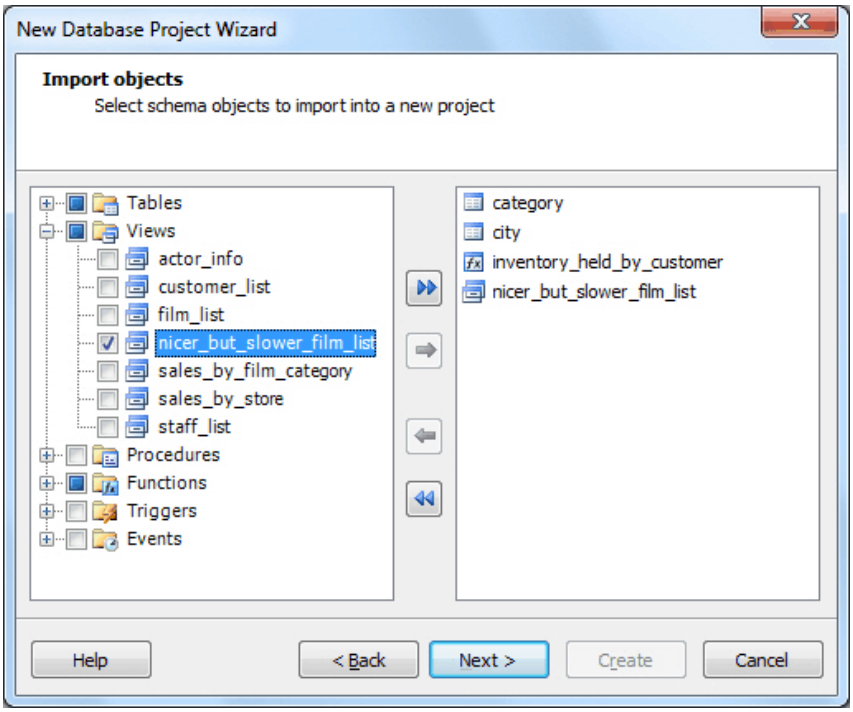
მონაცემთა ბაზის ობიექტების იმპორტი პროექტში:

1. ვირჩევთ **Import schema objects** არსებულ მონაცემთა ბაზაზე ალამის დაყენებით და ვაწკაპუნებთ ღილაკზე **Next.**



სურ. 13.4

2.ჩვენ შეგვიძლია ავირჩიოთ მხოლოდ საჭირო მონაცემთა ბაზის ობიექტები ჩამონათვალიდან ოსტატის შემდეგ გვერდზე.



სურ. 13.5

3. ვაწკაპუნებთ ღილაკზე **Create.** ჩვენ დავინახავთ შექმნილ პროექტს იმპორტირებული ობიექტებით **Project Explorer** ფანჯარაში.

ვაწკაპუნებთ ღილაკზე **Save** ინსტრუმენტების სტრიქონზე.

მონაცემთა ბაზის პროექტის შენახვა

პროექტის შენახვისათვის Project Explorer-ში პროექტზე მარჯვენა დაწკაპუნებით იხსნება popup მენიუ, სადაც ვირჩევთ **Save Project** ან ვააქტიურებთ **Project Explorer** და ვაჭერთ ღილაკს **Save** ან **File** მენიუში ვირჩევთ **Save <project name>**.

მონაცემთა ბაზის პროექტის გახსნა

არსებული პროექტის გახსნისათვის **File** მენიუდან ვირჩევთ **Open -> Project**.

2. ფაილების მართვა პროექტში

მუშაობა ფოლდერებთან

ფოლდერის შექმნისათვის პროექტზე ან ფოლდერზე მარჯვენა დაწკაპუნებით Project Explorer-ში იხსნება popup მენიუ, სადაც ვირჩევთ **Add Folder**.

ფოლდერის წაშლისათვის მარჯვენა დაწკაპუნებით მასზე Project Explorer-ში იხსნება popup მენიუ, სადაც ვირჩევთ **Delete.**

ჩვენ შეგვიძლია ფოლდერში ან ფოლდერიდან SQL script-ებისა და მოთხოვნების გადატანა dragging-dropping მეშვეობით.

ფაილის დამატება პროექტში

ამისათვის მარჯვენა დაწკაპუნებით პროექტზე ან Project Explorer-ში მის ფოლდერზე გაიხსნება popup მენიუ, სადაც ვირჩევთ **Add New File**

-ან-

Project მენიუში ვირჩევთ **Add New File**.

გამოჩნდება **Add New File** ფანჯარა. ვირჩევთ ფაილის სანახველ ნიმუშს. შეგვაქვს ფაილის სახელი. ვაწკაპუნებთ **Add** ფაილის შესაქმნელად კოდთან ერთად.

პროექტში არსებული ფაილის დამატებისათვის მარჯვენა დაწკაპუნებით პროექტზე ან Project Explorer-ში მის ფოლდერზე გაიხსნება popup მენიუ, სადაც ვირჩევთ **Add Existing File**.

პროექტიდან ფაილის ამოგდება

ფაილის ამოგდება არ ნიშნავს მის ფიზიკურ წაშლას, არამედ მხოლოდ პროექტთან მისი ლინკის წაშლას. ფაილის ამოგდებისათვის მარჯვენა დაწკაპუნებით მასზე Project Explorer-ში ჩნდება popup მენიუ, სადაც ვირჩევთ Delete და ვაწკაპუნებთ Remove-ს დიალოგურ ფანჯარაში, ხოლო ფიზიკური წაშლისათვის

დიალოგურ ფანჯარაში ვირჩევთ Delete-ს.

3.მუშაობა მონაცემთა ბაზის ობიექტებთან პროექტში

მონაცემთა ბაზის ობიექტების გადარქმევა:

1. Schema View ფანჯარაში ობიექტზე მარჯვენა დაწკაპუნებით იხსნება popup მენიუ.
2. ვირჩევთ **Rename**.
3. შეგვაქვს ახალი სახელი.
4.ვათვალიერებთ ცვლილებებს და ვაწკაპუნებთ **Apply.**

მონაცემთა ბაზის ობიექტების წაშლა:

1. Schema View ფანჯარაში ობიექტზე მარჯვენა დაწკაპუნებით იხსნება popup მენიუ.
2. ვირჩევთ **Delete**.
3. ვაწკაპუნებთ **Yes**.

Project Build Configurations გამოყენება

ყოველ პროექტს აქვს ერთი კონფიგურაცია სახეწოდებით **Default**.

კონფიგურაციის დამატება

1. **Project** ინსტრუმენტების სტრიქონში **Configuration** ჩამოშლად სიაში **Configuration Manager** არჩევით ვხსნით Configuration Manager.
2. ვაწკაპუნებთ ღილაკზე **Add**. ჩნდება New Project Configuration ფანჯარა.
3. შეგვაქვს სახელი და ვირჩევთ კონფიგურაციას.
4. ვაწკაპუნებთ **OK**.

კონფიგურაციის ამოგდება

1. **Project** ინსტრუმენტების სტრიქონში **Configuration** ჩამოშლად სიაში **Configuration Manager** არჩევით ვხსნით Configuration Manager.
2. ვირჩევთ კონფიგურაციას, რომლის ამოგდება გვინდა და ვაწკაპუნებთ **Remove**.

კონფიგურაციის გადარქმევა

1. **Project** ინსტრუმენტების სტრიქონში **Configuration** ჩამოშლად სიაში **Configuration Manager** არჩევით ვხსნით Configuration Manager.
2. ვირჩევთ საჭირო კონფიგურაციას Configuration Manager -ში და ვაწკაპუნებთ ღილაკზე**Rename** ან ვაჭერთ **F2** კლავიშს.
3. ვბეჭდავთ ახალ სახელს.
4. ვაჭერთ **ENTER**.

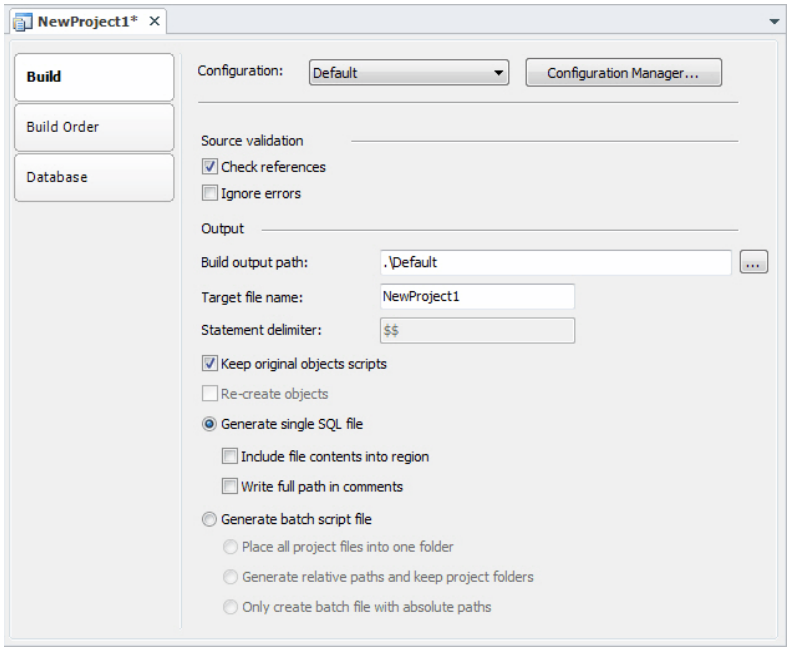
4. პროექტის აგება და განთავსება

პროექტის აგება

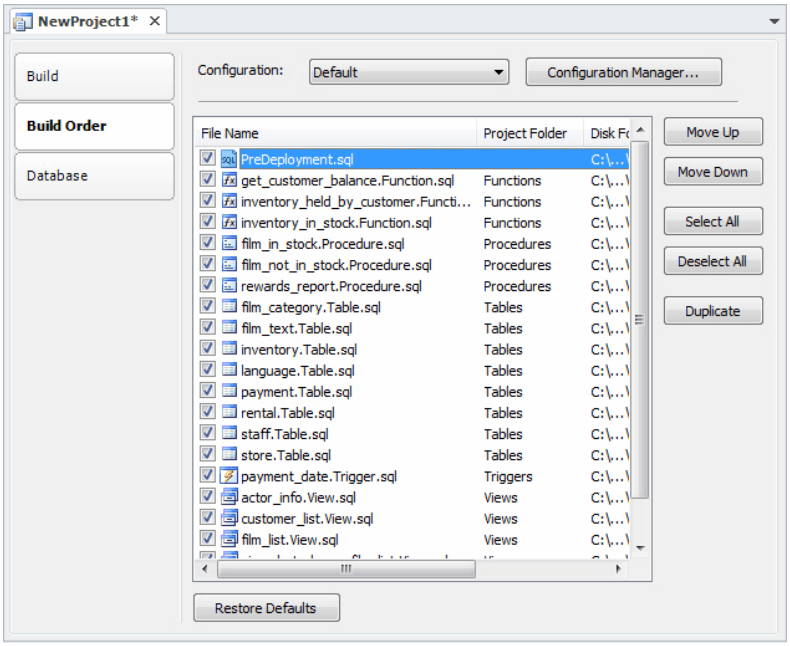
პროექტის აგებამდე ჩვენ შეგვიძლია დავაყენოთ პროექტის ოპციები.

1. მარჯვენა დაწკაპუნება **Project Explorer** ფანჯარაში. ვირჩევთ **Properties** ოპციას ან **Project** მენიუში ვაწკაპუნებთ **Properties**.

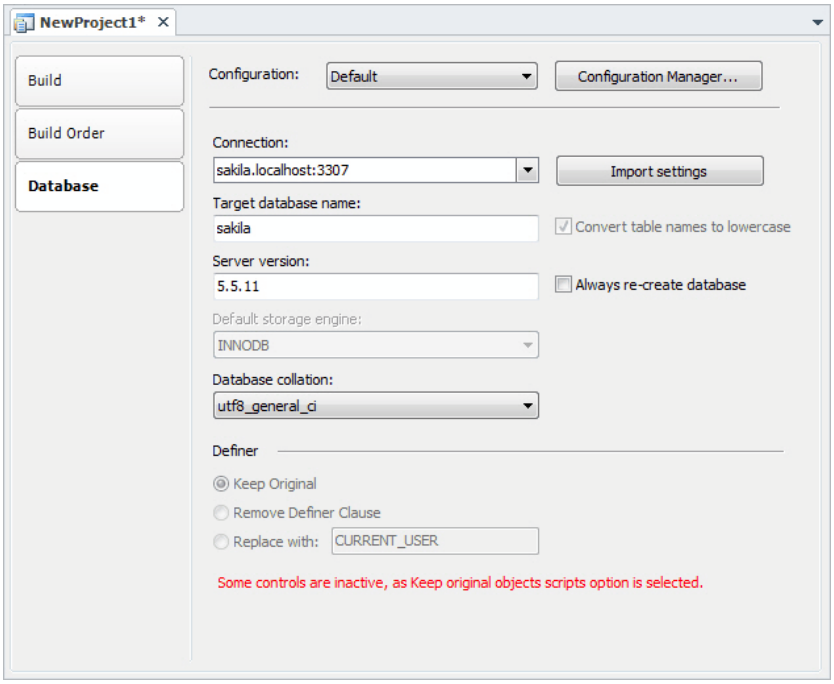
იხსნება **Project Properties** ფანჯარა სამი ჩანართით: **Build**, **Build Order** და **Database**.



სურ. 13.6



სურ. 13.7



სურ. 13.8

პროექტის ასაგებად ვასრულებთ შემდეგ ქმედებებს:

1. **Project** მენიუში ვაწკაპუნებთ **Properties.**
2. **Project Properties** ფანჯრის **Build** ჩანართში ვირჩევთ პროექტის კონფიგურაციას.
3. მარჯვენათი ვაწკაპუნებთ **Project Explorer** ფანჯარაში. მენიუდან ვირჩევთ **Build the Project**.

პროექტის განთავსება სერვერზე

პროექტის სრულად განთავსებისათვის **Build** მენიუში ვაწკაპუნებთ **Deploy the Project**.

მონაცემთა ბაზის მთლიანობის უზრუნველსაყოფად ვიყენებთ ოპციას **Always re-create a database**.

5.მუშაობა პროექტთან

წინა პირობები

პროექტის შექმნისათვის უნდა გაგვაჩნდეს შემდეგი პრივილეგიები:

- CREATE
- DROP
- INSERT
- DELETE
- UPDATE
- SELECT

•CREATE USER
•GRANT OPTION

პროექტის შექმნა

შესაქმნელი ფოლდერების სტრუქტურა

მთავარი ფოლდერი უნდა შეიცავდეს მინიმუმ სამ ფოლდერს: 'Security', 'Database objects', 'Data'. მონაცემთა ბაზის ყოველი ობიექტი იქმნება ფოლდერში 'Database objects' შემდეგი ბიჯების შესრულებით:

1. მარჯვენა დაწკაპუნებით Project Explorer -ის ფესვზე იხსნება popup მენიუ, სადაც ვირჩევთ **Add Folder**

- ან -

ვირჩევთ პროექტს Project Explorer -ის ფესვზე და **Project** მენიუში ვირჩევთ **Add Folder**.

2. შეგვაქვს ახალი ფოლდერის სახელი ('Security').
3. ვიმეორებთ 1 - 3 ბიჯებს ფოლდერებისათვის 'Database objects' და 'Data'.
4. მარჯვენა დაწკაპუნებით 'Database objects' ფოლდერზე Project Explorer -ის ფესვზე იხსნება popup მენიუ.
5. ვირჩევთ **Add Folder**.
6. შეგვაქვს ახალი ფოლდერის სახელი ('Tables').
7. ვიმეორებთ 4 - 6 ბიჯებს ფოლდერებისათვის 'Views' და 'Triggers'.

პროექტში SQL Script ფაილების დამატება

1. **Standard** ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ ღილაკზე **New SQL**.
2. ვკრეფთ ან ვაკოპირებთ კოდს SQL დოკუმენტში.
3. **Standard** ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ ღილაკზე **Save**.
4. **Save As** ფანჯარაში ვირჩევთ ფაილის განთავსების ადგილს, შეგვყავს სახელი.
5. ვინახავთ ფაილს.
6. Project Explorer -ში მარჯვენა დაწკაპუნებით ფოლდერზე 'Tables' ვხსნით popup მენიუს, სადაც ვირჩევთ **Add Existing File**

- ან -

Project Explorer -ში ვაწკაპუნებთ ფოლდერზე 'Tables' და **Project** მენიუში ვირჩევთ **Add Existing File**.

7. ვირჩევთ შექმნილ ფაილს **Open** ფანჯარაში.

სხვაგვარად ეს ქმედება შეიძლება შესრულდეს შემდეგნაირად:

1. **Standard** ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ ღილაკზე **New SQL**.
2. ვკრეფთ ან ვაკოპირებთ კოდს SQL დოკუმენტში:
3. **Standard** ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ ღილაკზე **Save**.
4. **Save As** ფანჯარაში ვირჩევთ ფაილის განთავსების ადგილს, შეგვყავს სახელი.
5. ვინახავთ ფაილს.

6. **File** მენიუდან ვირჩევთ **Move Table emp.sql into Project** და ვაწკაპუნებთ **New Project**.

7. ფაილის იკონა გადაგვაქვს (Drag) **Project Explorer** -ის 'Tables' ფოლდერში.

ჩვენ შეგვიძლია ფაილის ნიმუშების გამოყენება როცა ახალ ფაილს ვამატებთ პროექტში.

1. **Project Explorer**-ში მარჯვენა დაწკაპუნებით ფოლდერზე 'Views' ჩნდება კონტესტური მენიუ, სადაც ვირჩევთ **Add New File**

- ან -

Project Explorer-ში ვაწკაპუნებთ ფოლდერზე 'Views' და **Project** მენიუდან ვირჩევთ **Add New File**.

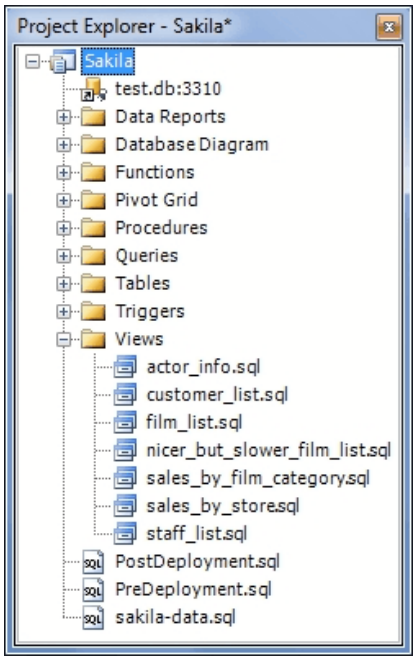
2. **Templates** -ში ვირჩევთ **View**-ს და ვაჭერთ **Add**.

3. წარმოდგენის ფაილს 'Test.VIEW1' ვუცვლით სახელს SQL დოკუმენტში.

4. ვკრეფთ ან ვაკოპირებთ მოთხოვნის ტექსტს **AS** გასაღებური სიტყვის შემდეგ.

5. ვინახავთ დოკუმენტს ახალი სახელით.

ანალოგიურად ვამატებთ script ფაილებს ფოლდერებში 'Data', 'Triggers', 'Security'და სხვ. ამის შემდგომ ვინახავთ პროექტს.



სურ. 13.9

სქემისა და პროექტის შედარება

შედარების დოკუმენტის ახალი სქემის შექმნისათვის საჭიროა შემდეგი ქმედებების შესრულება:

1. **Comparison** მენიუში ვირჩევთ **New Schema Comparison**.

ვაწკაპუნებთ **Project** წყარო (source) ან მიზნობრივ (target) ობიექტზე. შემდეგ

ვირჩევთ პროექტს ველში ქვემოთ.

2. ვირჩევთ მიერთებას შედარების სქემისათვის. ვაწკაპუნებთ ღილაკებზე **Edit Connection** ან **New Connection**.

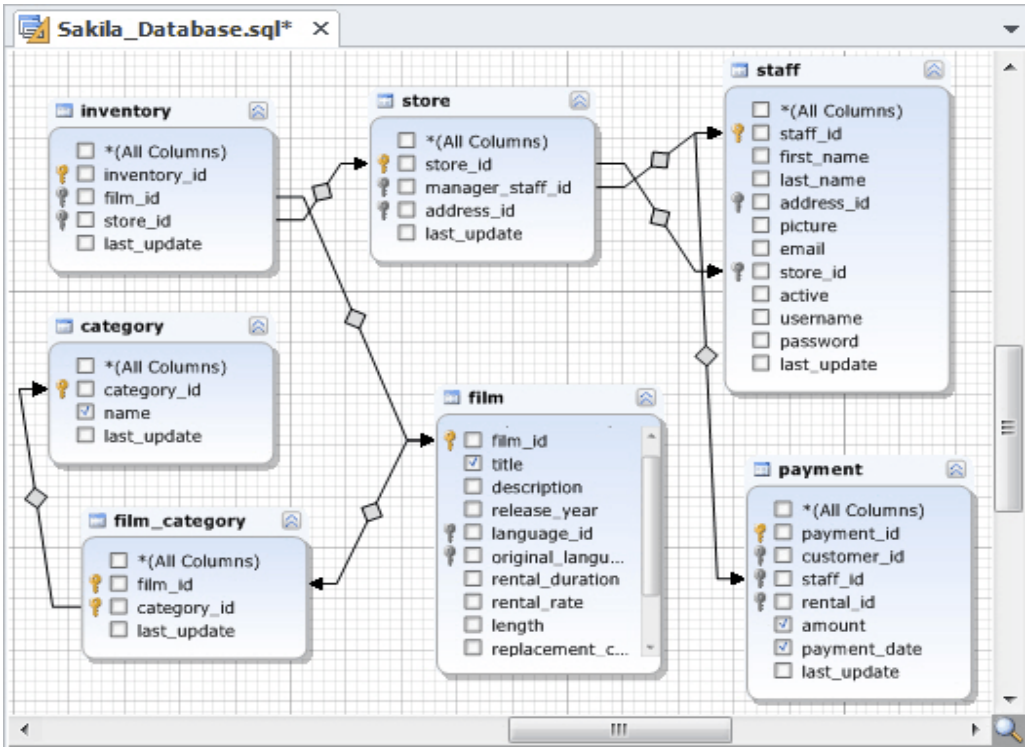
3. ვირჩევთ შესადარებელ მონაცემთა ბაზას.

4. ვაწკაპუნებთ ღილაკზე **OK**.

6. მონაცემთა ანალიზი

მონაცემებთან ურთიერთობის დროს მეტად მოსახერხებელია ე.წ. მთავარი ანუ ძირითადი (**Pivot**) ცხრილის როგორც მონაცემთა მართვის ინსტრუმენტის გამოყენება.

მაგალითისათვის განვიხილოთ სადემონსტრაციო მონაცემთა ბაზა **Sakila** (იხილე დანართი)



სურ. 13.10

სადაც Query Builder-ში ვქმნით მოთხოვნას:

•**title** ცხრილიდან **Film.**

•**name** ცხრილიდან **Category.**

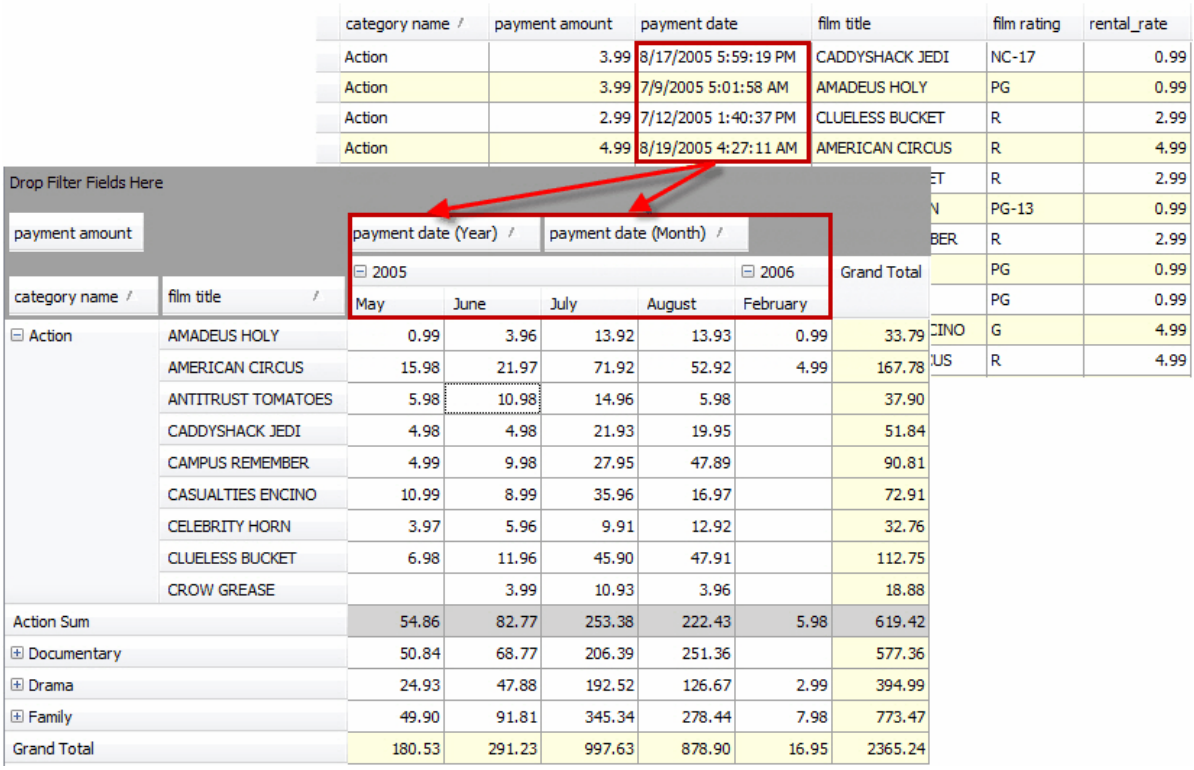
•**amount** და **payment_date** ცხრილიდან **Payment**.

შესრულებული მოთხოვნის შედეგები წარმოდგენილია ცხრილის სახით.

category name	payment amount	payment date	film title	film rating	rental_rate
Action	3.99	8/17/2005 5:59:19 PM	CADDYSHACK JEDI	NC-17	0.99
Action	3.99	7/9/2005 5:01:58 AM	AMADEUS HOLY	PG	0.99
Action	2.99	7/12/2005 1:40:37 PM	CLUELESS BUCKET	R	2.99
Action	4.99	8/19/2005 4:27:11 AM	AMERICAN CIRCUS	R	4.99
Action	7.99	8/23/2005 12:44:15 AM	CLUELESS BUCKET	R	2.99
Action	0.99	6/18/2005 7:39:05 PM	CELEBRITY HORN	PG-13	0.99
Action	2.99	8/22/2005 7:46:36 PM	CAMPUS REMEMBER	R	2.99
Action	0.99	6/19/2005 10:20:09 AM	AMADEUS HOLY	PG	0.99
Action	0.99	8/19/2005 3:37:25 AM	CROW GREASE	PG	0.99
Action	9.99	7/30/2005 7:27:05 PM	CASUALTIES ENCINO	G	4.99
Action	9.99	7/7/2005 3:35:33 AM	AMERICAN CIRCUS	R	4.99

სურ. 13.11

ახლა ვსუროთ როგორ აისახება მონაცემები Pivot Table -ში.

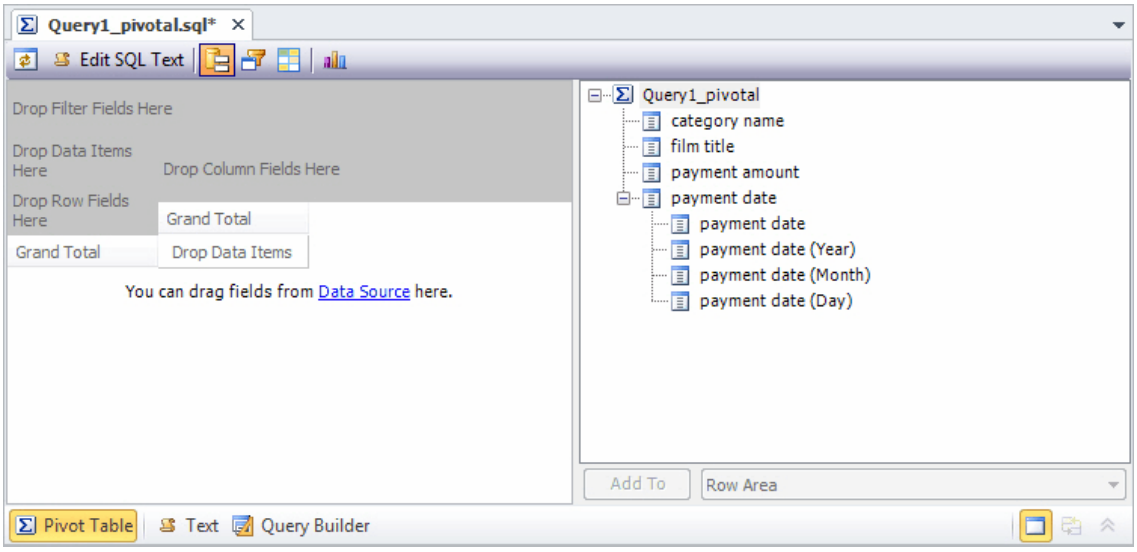


სურ. 13.12

Pivot Table -ის შექმნისათვის საჭიროა შემდეგი ქმედებების შესრულება:

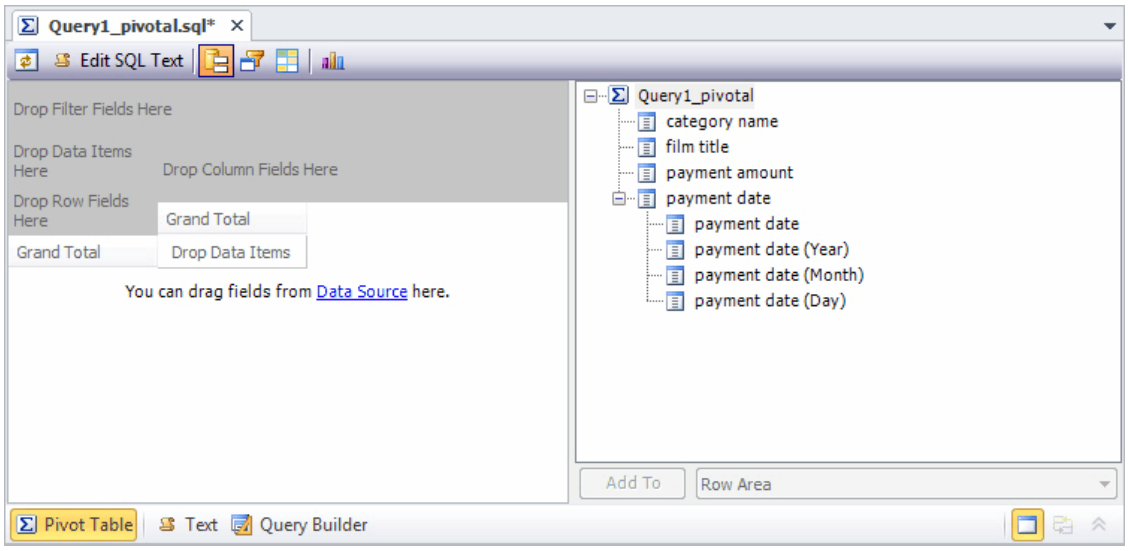
1. მოთხოვნის დოკუმენტის გახსნისათვის მენიუში ვირჩევთ **File >New>Query** ან ინსტრუმენტების პანელზე ვაწკაპუნებთ **New Query** ღილაკზე. **Database Explorer** -ში ვირჩევთ პროექტის ფარგლებში შექმნილი მონაცემთა ბაზის საჭირო ცხრილებს და (drag-and-drop) -ით გადაგვაქვს ისინი მოთხოვნის დოკუმენტში.

2. ჩავრთოთ **Pivot Table.**



სურ. 13.13

Data Source ავტომატურად იხსნება იმ ველებთან ერთად, რომლებიც ასახული და განსაზღვრულია ჩვენი მოთხოვნის დოკუმენტში. განხილული მაგალითის შემთხვევაში გადახდის თარიღის (payment date) ველი დეკომპოზირებულია ოთხი ქვე-ველის სახით: year, month, day და master.



სურ. 13.14

3. შემდეგ ვიზრუნოთ იმაზე, თუ რამდენად ეფექტურად შეიძლება განთავსდეს შერჩეული ველები pivot ცხრილში:

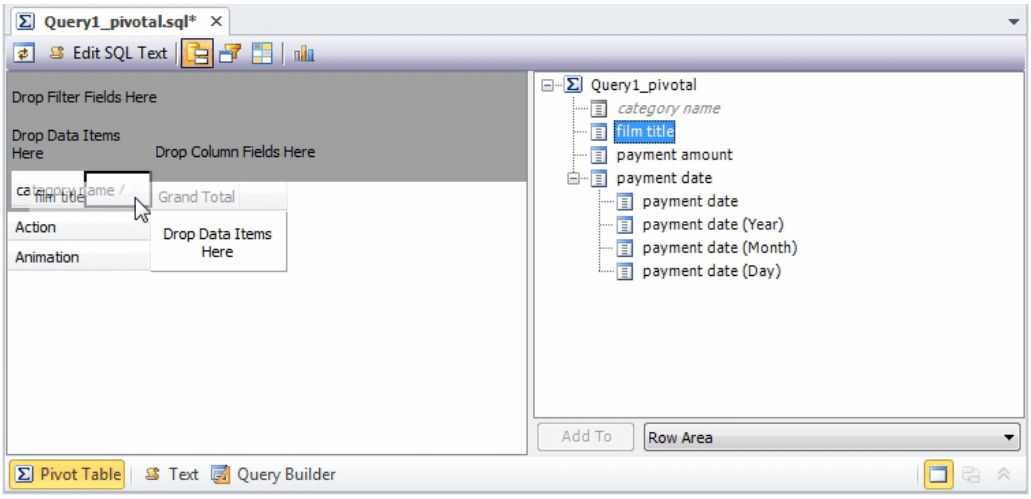
• ცხრილის Category ველებს name და film title გააჩნიათ სტრიქონული ტიპის მნიშვნელობები. ამასთან, ზოგიერთი მათგანი საკმაოდ გრძელია. ამიტომ, უმჯობესია მათი განთავსება როგორც ჰორიზონტალური ველი, ხოლო მათი მნიშვნელობები ვერტიკალურად აისახება pivot ცხრილის მარცხენა ნაწილში table.

• ცხრილის Payment ველი amount შეიცავს რიცხვითი ტიპის მნიშვნელობებს. ამდენად, უმჯობესია მათი განთავსება data არეს შიგნით, სადაც საერთო ჯამები და

კერძო ჯამები იქნება გამოანგარიშებული.

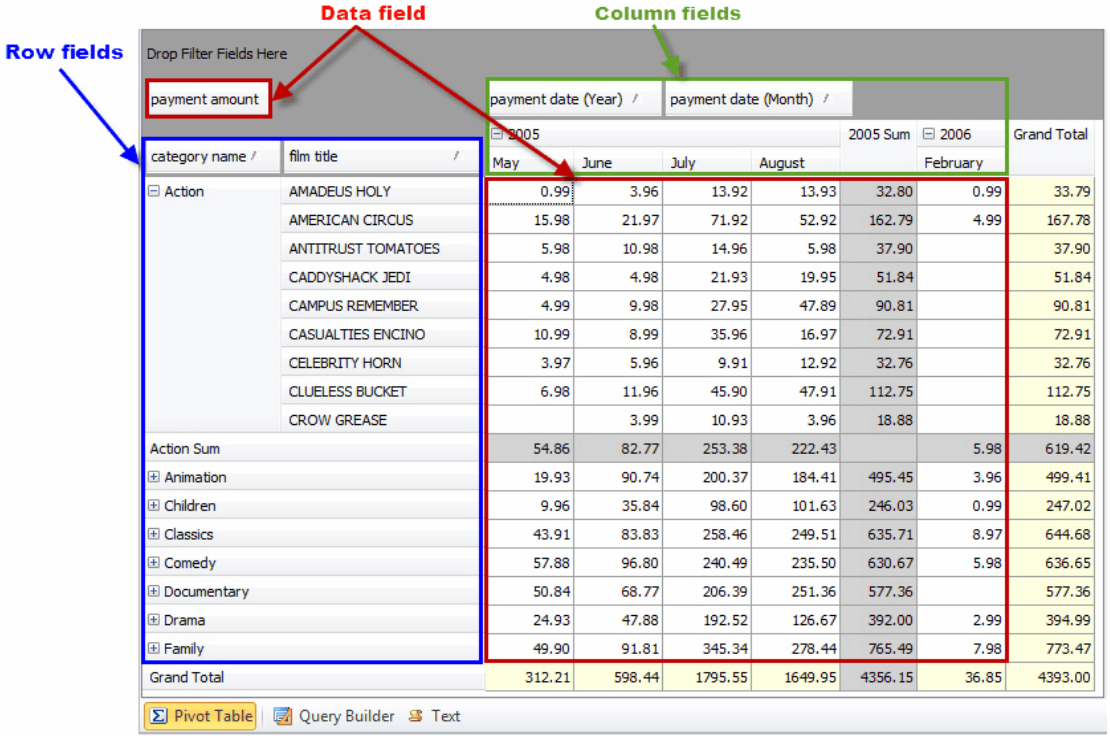
• თუ ცხრილის Payment ველი date აისახება pivot ცხრილში ჰორიზონტალურად, მაშინ ქვეველები day, month და year დაჯგუფდებიან.

Data Source-დან ველების, drag-and-drop მეთოდის საშუალებით, pivot ცხრილის შესაბამის არეში დამატებისათვის Destination-ს ჩამოშლადი სიიდან ვირჩევთ საჭირო არეს და ვაჭერთ ღილაკს Add To ან ვაჭერთ Enter. ორი ან მეტი ველის pivot ცხრილის იმავე არეში დამატებისათვის ჯერ გადაგვაქვს პირველი ველი, ხოლო მეორე ველი პირველი ველის წინ ან შემდეგ.



სურ. 13.15

4. ჩვენს pivot ცხრილს ექნება შემდეგი სახე.

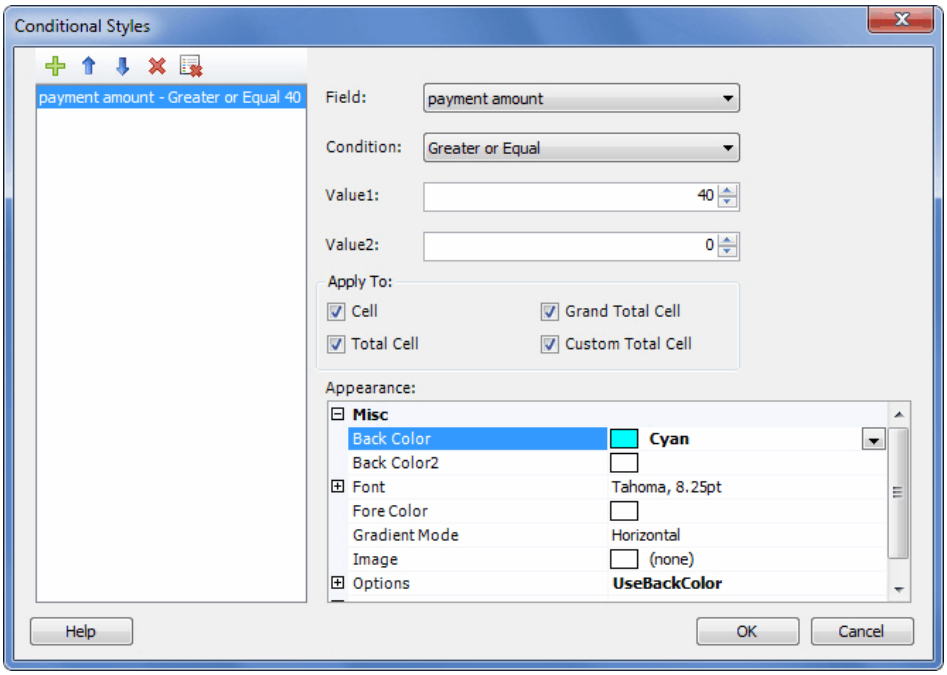


სურ. 13.16

ველებში მნიშვნელობების გამოსატანად ვაჭერთ Sort ღილაკს.

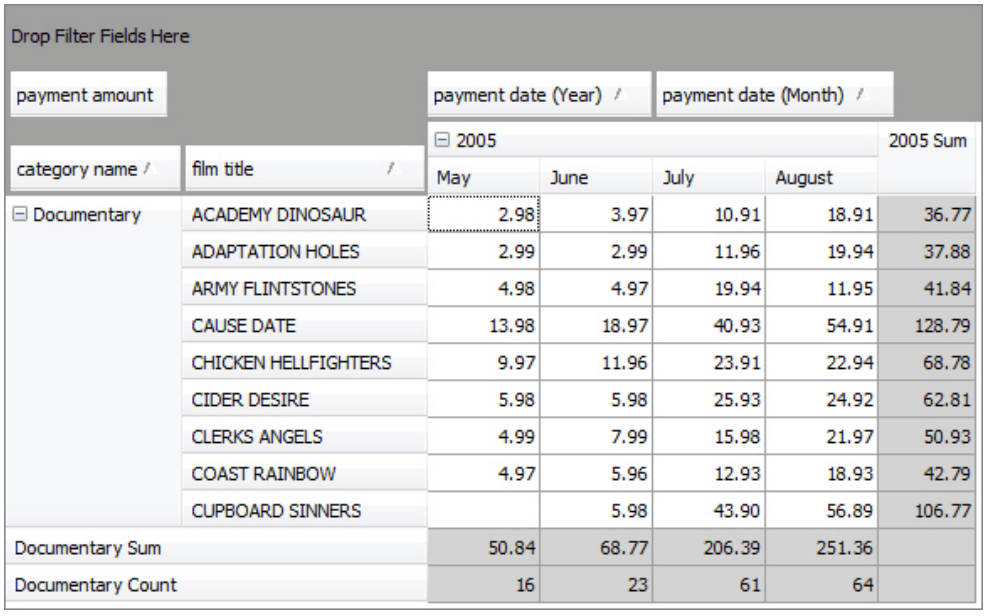
5. თუ საჭიროება მოითხოვს, შეგვიძლია გამოვიყენოთ პირობითი სტილი (conditional styles), რისთვისაც:

- **Pivot Table** ინსტრუმენტების სტრიქონში ვაწკაპუნებთ ღილაკზე **Conditional Styles** ან pivot ცხრილის ზედა კოლონტიტულში ორჯერადი დაწკაპუნებით ვხსნით მენიუს, სადაც ვირჩევთ **Conditional Styles.** შემდეგ ფანჯარაში ვაწკაპუნებთ ღილაკზე ახალი პირობის დასამატებლად.



სურ. 13.17

შედეგები აისახება pivot ცხრილში.



სურ. 13.18

7.Pivot Table კომპონენტები

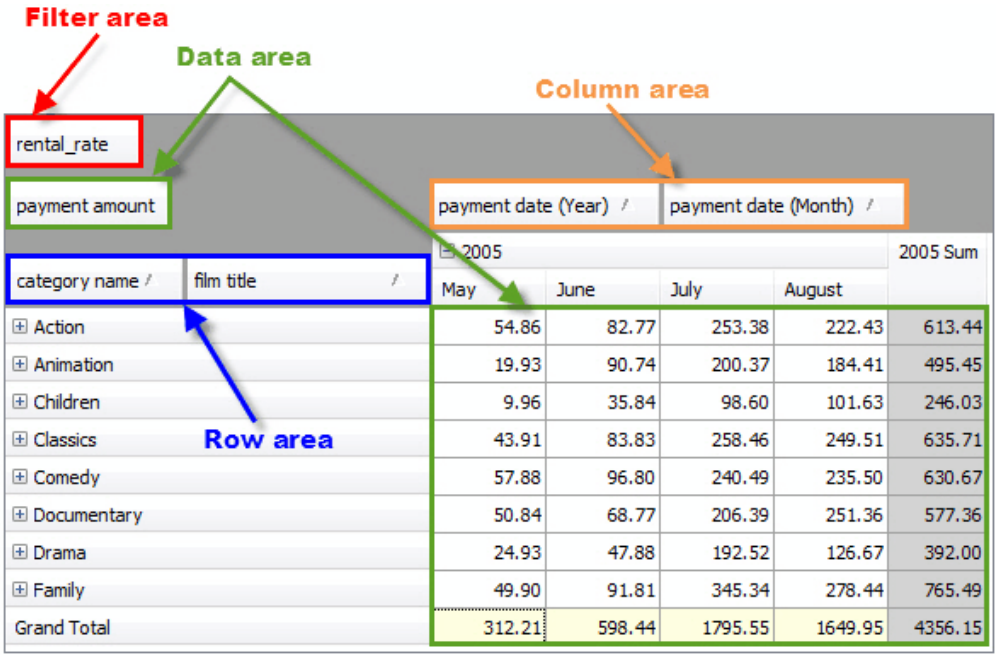
არეები:

•Filter area.

•Row area.

•Column area.

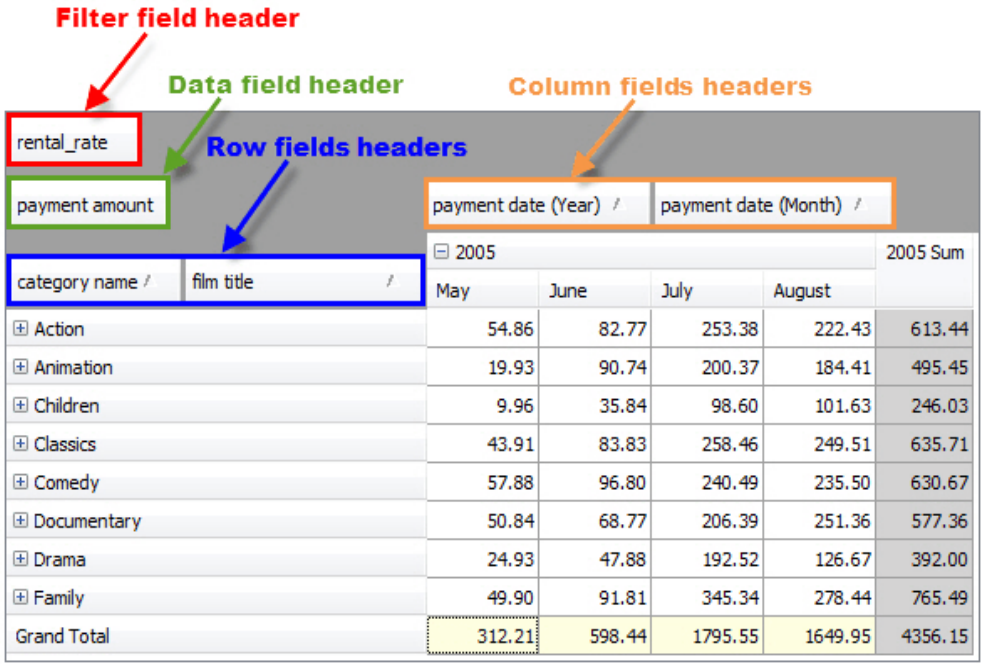
•Data area.



სურ. 13.19

Pivot Table კომპონენტებია:

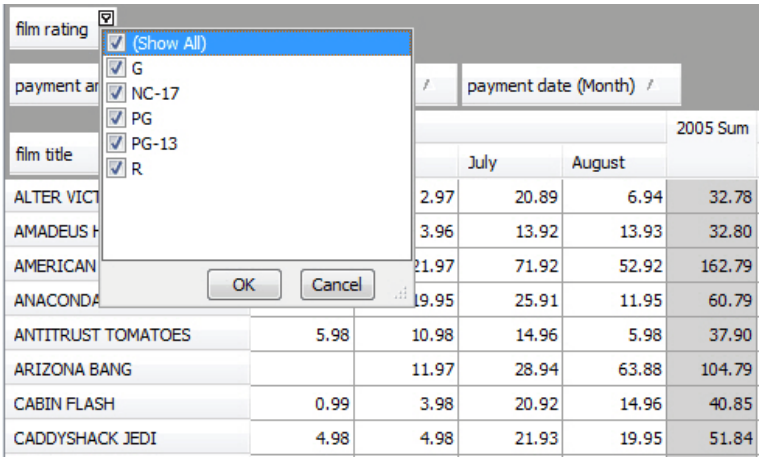
- Headers ანუ ზედა კოლონტიტული.



სურ. 13.20

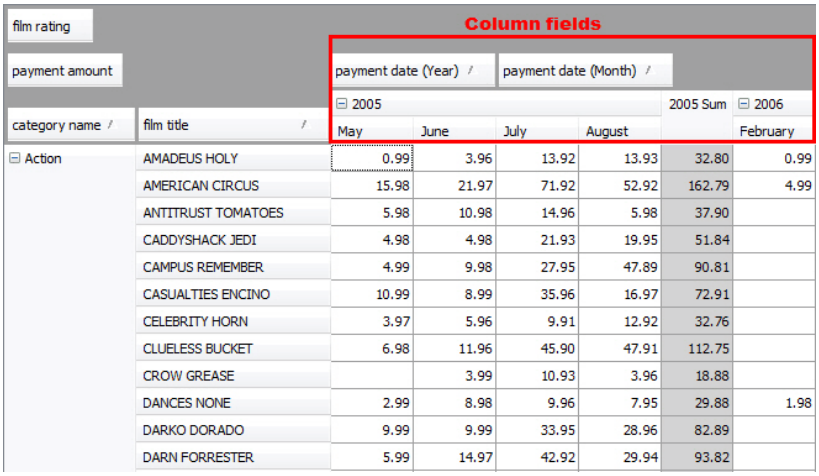
- Filter field ფილტრის ველი. მისი კრიტერიუმის დაყენებისათვის **filter** ველში ვაწკაპუნებთ ღილაკზე **Filter** და ვირჩევთ კრიტერიუმს ჩამოშლადი სიიდან.

-



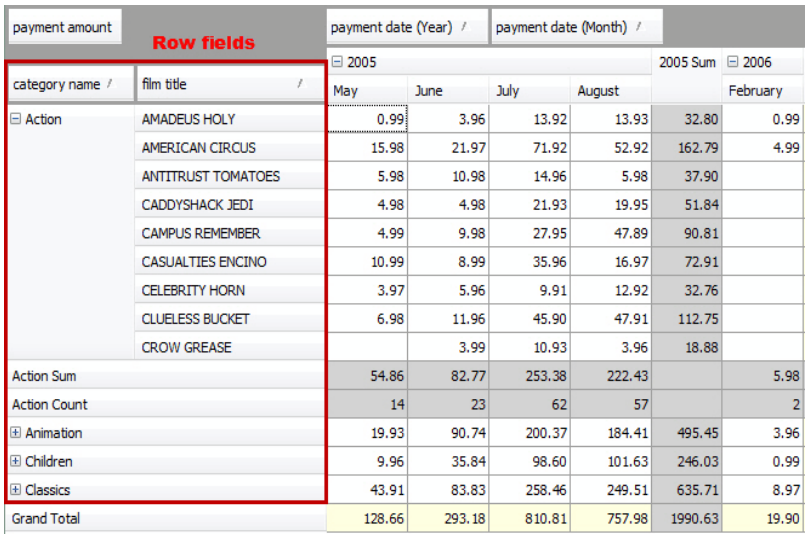
სურ. 13.21

- Column field ანუ ვერტიკალური ველი.



სურ. 13.22

- Row field ანუ ჰორიზონტალური ველი.



სურ. 13.23

- Data field ანუ ველი, სადაც განთავსებულია მონაცემები და მათემატიკური ფუნქციების გამოთვლების შედეგები:

• Sum.

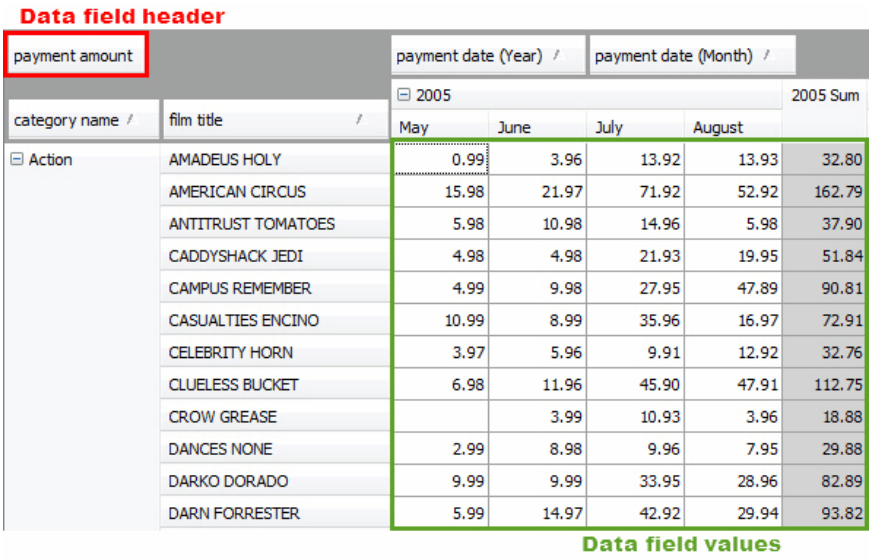
• Average.

• Count.

• Min, Max.

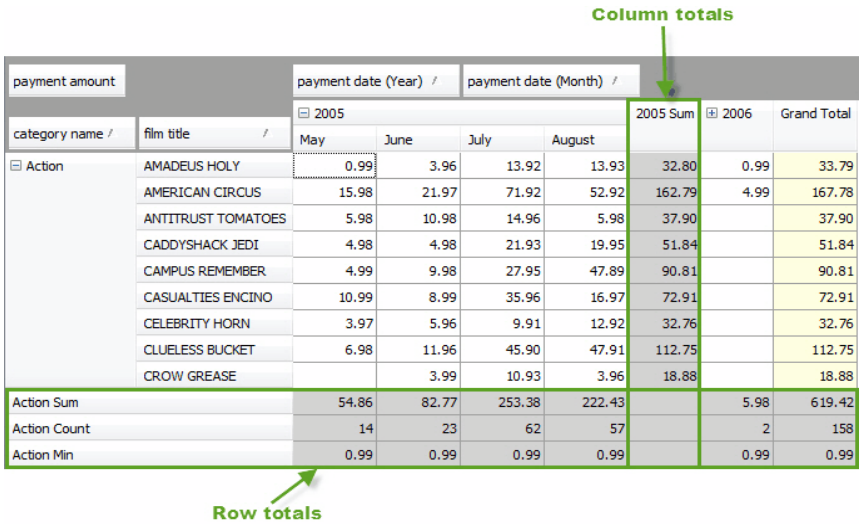
• მთლიან მონაცემებზე ან ქვესიმრავლეზე დაფუძნებული სტანდარტული გადახრა.

• მთლიან მონაცემებზე ან ქვესიმრავლეზე დაფუძნებული დისპერსიების პოპულაცია.



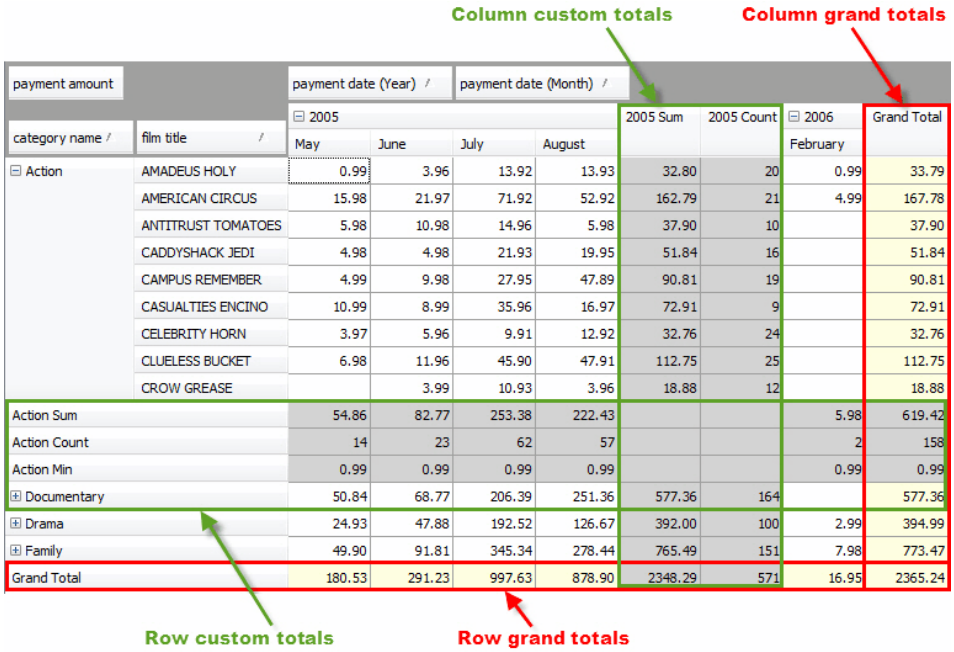
სურ. 13.24

Row Totals წარმოადგენს სტრიქონების რიცხვით მნიშვნელობათა ჯამების გამომთვლელს. **Column Totals** კი სვეტებში რიცხვითი მნიშვნელობების ჯამის გამოთვლისათვის არის განკუთვნილი.



სურ. 13.25

Row Grand Totals სტრიქონული ჯამების მნიშვნელობებს ავტომატურად აჯამებს. შესაბამისად, ავტომატურად ხდება სვეტების ჯამების შეჯამება **Column Grand Totals** -ის მეშვეობით.



payment amount		payment date (Year)		payment date (Month)					
		2005				2005 Sum	2005 Count	2006	Grand Total
category name	film title	May	June	July	August			February	
Action	AMADEUS HOLY	0.99	3.96	13.92	13.93	32.80	20	0.99	33.79
	AMERICAN CIRCUS	15.98	21.97	71.92	52.92	162.79	21	4.99	167.78
	ANTITRUST TOMATOES	5.98	10.98	14.96	5.98	37.90	10		37.90
	CADDYSHACK JEDI	4.98	4.98	21.93	19.95	51.84	16		51.84
	CAMPUS REMEMBER	4.99	9.98	27.95	47.89	90.81	19		90.81
	CASUALTIES ENCINO	10.99	8.99	35.96	16.97	72.91	9		72.91
	CELEBRITY HORN	3.97	5.96	9.91	12.92	32.76	24		32.76
	CLUELESS BUCKET	6.98	11.96	45.90	47.91	112.75	25		112.75
	CROW GREASE		3.99	10.93	3.96	18.88	12		18.88
Action Sum		54.86	82.77	253.38	222.43			5.98	619.42
Action Count		14	23	62	57			2	158
Action Min		0.99	0.99	0.99	0.99			0.99	0.99
Documentary		50.84	68.77	206.39	251.36	577.36	164		577.36
Drama		24.93	47.88	192.52	126.67	392.00	100	2.99	394.99
Family		49.90	91.81	345.34	278.44	765.49	151	7.98	773.47
Grand Total		180.53	291.23	997.63	878.90	2348.29	571	16.95	2365.24

სურ. 13.26

მონაცემთა ნახვა Master-Detail Tables -ში

dbForge Studio იძლევა საშუალებას ვსუროთ მონაცემები მშობელ ცხრილებსა თუ წარმოდგენებში და აგრეთვე მათთან დაკავშირებულ შვილობილ ცხრილებსა თუ წარმოდგენებში, რაც ხდება Master-Detail Browser-ის დახმარებით.

არსებობს **Master-Detail Browser** -ის რამდენიმე გზა:

• **Database** მენიუში ვაწკაპუნებთ **Master-Detail Browser**

• **File** მენიუში ვაწკაპუნებთ **New** და შემდეგ ვაწკაპუნებთ **Master-Detail Browser**

• **Database Explorer** ფანჯარაში მარჯვენათი ვაწკაპუნებთ ცხრილზე (ან ცხრილებზე) ან წარმოდგენაზე (წარმოდგენებზე) და შემდეგ ვაწკაპუნებთ **Send to** და ბოლოს ვაწკაპუნებთ **New Master-Detail** ოპციაზე.

When you open the browser, a master-detail document with the ***.mdet** extension is created.

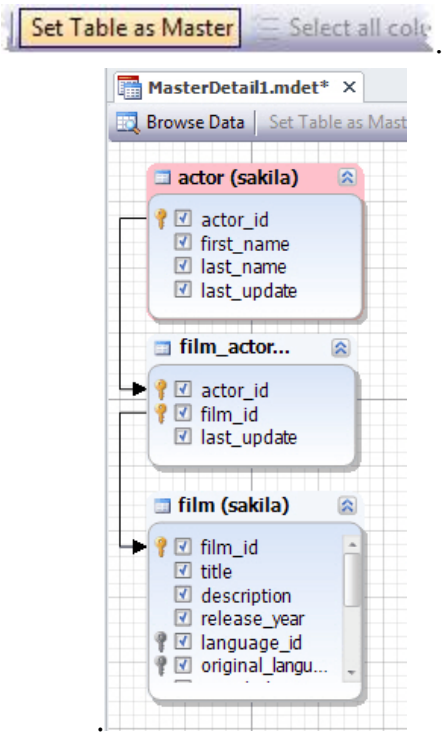
მონაცემთა ნახვა Related Tables -ში

დაკავშირებულ ცხრილებში მონაცემთა ნახვა განვიხილოთ სადემონსტრაციო მონაცემთა ბაზის მაგალითზე:

1. **Database** მენიუში ვაწკაპუნებთ **Master-Detail Browser**.

2. გადაგვაქვს actor, film_actor_id და film ცხრილები **Database Explorer** -ის designer area-ში.

3. ვირჩევთ ცხრილს actor და ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ **Set Table** როგორც Master:



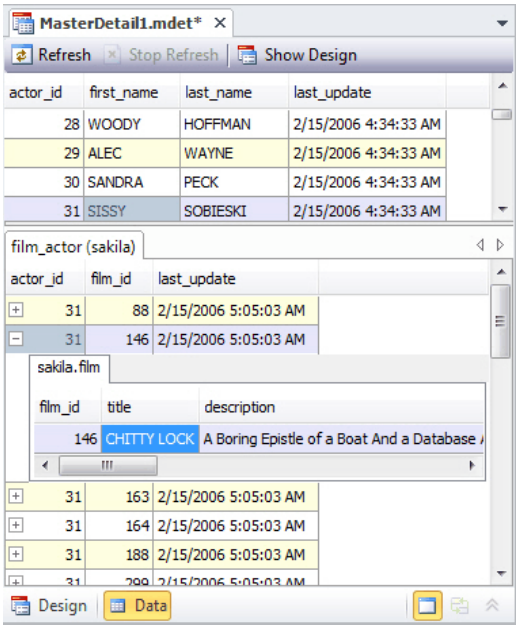
სურ. 13.27

4. ვაკავშირებთ სვეტებს, თუ ამის საჭიროება არსებობს.

5. ინტერესებიდან გამომდინარე ვირჩევთ (ან ვაუქმებთ) ალამებს სვეტების მახლობლად.

6. ინსტრუმენტების სტრიქონზე ვაწკაპუნებთ  **Browse Data** (F5).

7. გადავაადგილდეთ master ცხრილის ჩანაწერებს შორის და დეტალური მონაცემების ნახვისათვის ვაწკაპუნებთ + .



სურ. 13.28

Pivot Table-ის გრაფიკის აგება

1. ინსტრუმენტების პანელზე ვაწკაპუნებთ ღილაკზე **Show Chart.** გაიხსნება **Charts** გრაფიკის ტიპები.

2. ვირჩევთ მონაცემებს, რომლებიც უნდა ასახულ იქნას **Pivot Table** გრაფიკში, რისთვისაც ვაჭერთ **Ctrl** და ვაწკაპუნებთ საჭირო უჯრედზე ან ცხრილზე.

3. ვაზუსტებთ Chart ოპციებს, რისთვისაც ვიყენებთ **Chart** ინსტრუმენტების პანელს, popup მენიუს ან **Chart Designer**.

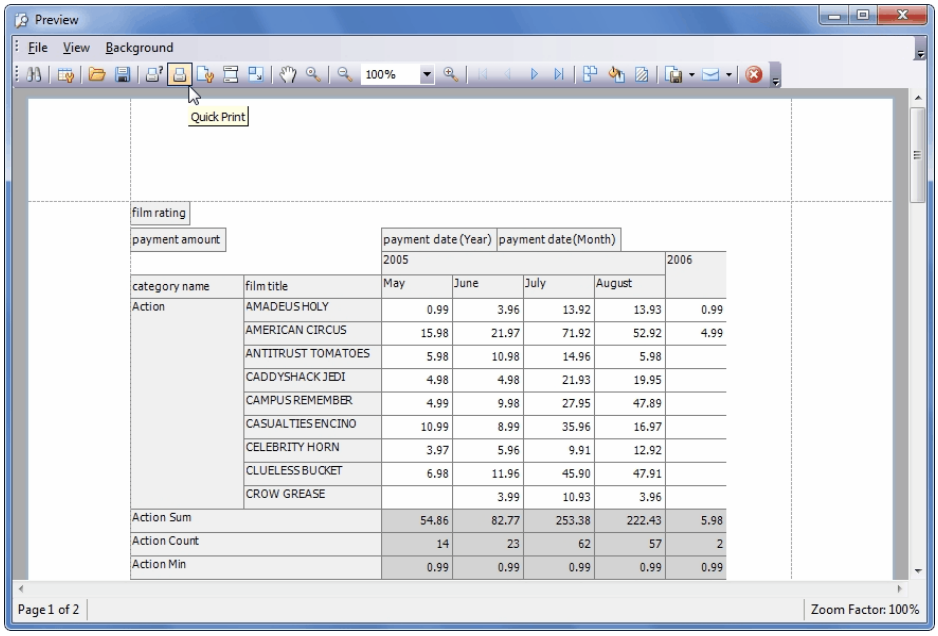
სურ. 13.29

ბეჭდვა

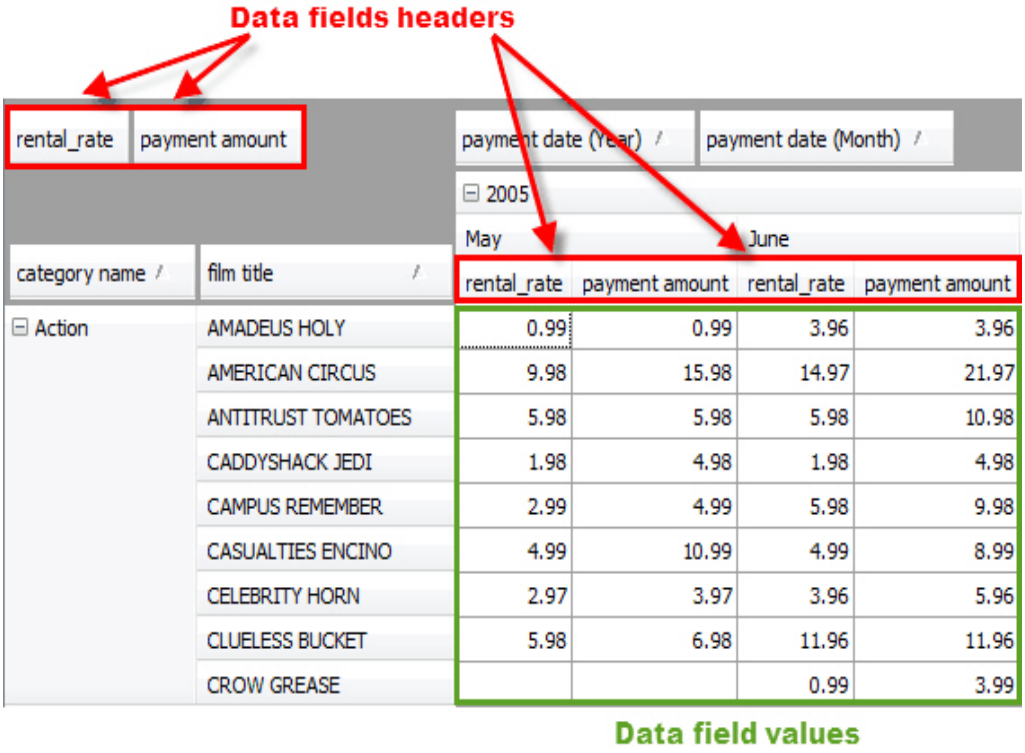
pivot ცხრილის ბეჭვდისათვის:

1. **File** მენიუდან ვირჩევთ **Print**. იხსნება ფანჯარა **Preview**, სადაც ვაზუსტებთ pivot ცხრილის მოდელს.

2. ვაწკაპუნებთ ღილაკზე **Quick Print**.



სურ. 13.30

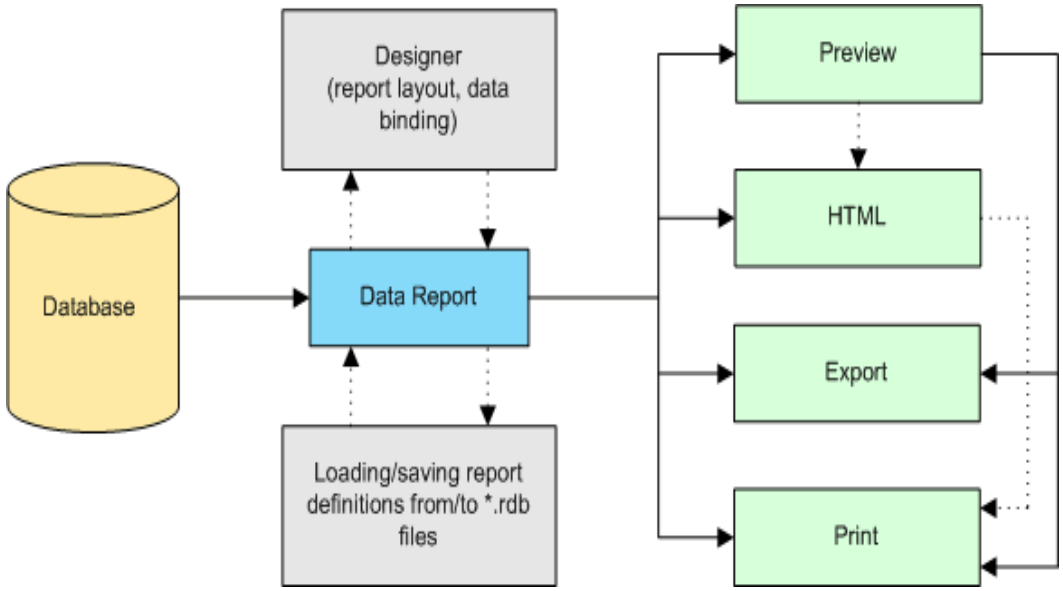


სურ. 13.31

8. მონაცემთა ანგარიშები

ანგარიშის აგების საფუძვლები

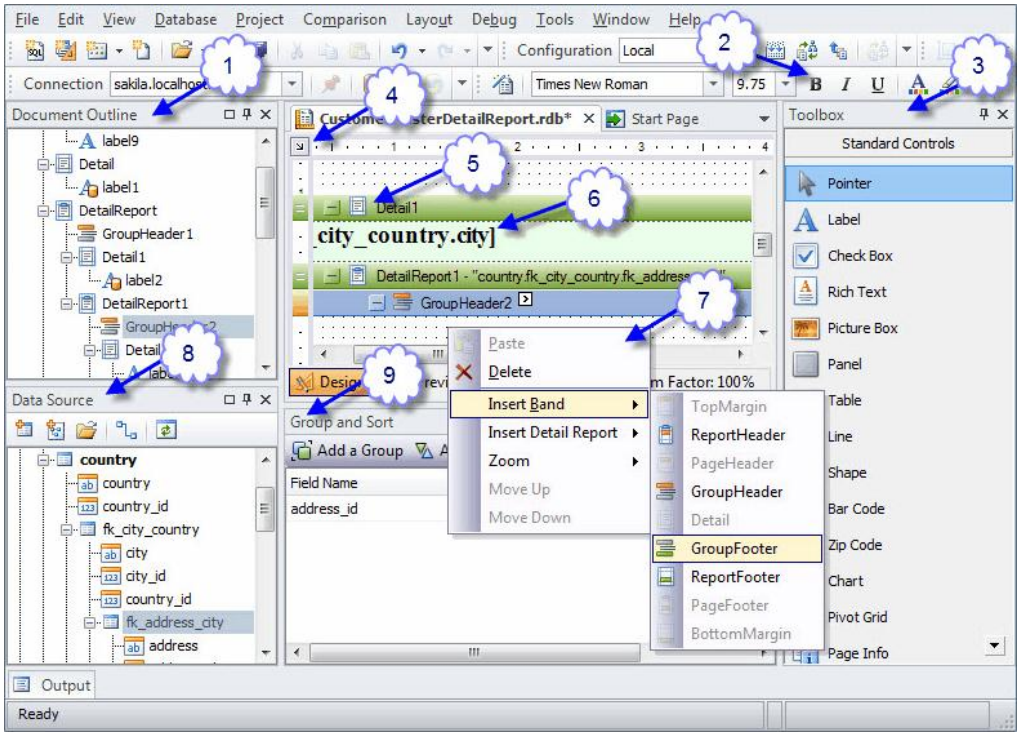
ანგარიშის სასიცოცხლო ციკლი dbForge Studio - ში ილუსტრირებულია შემდეგ დიაგრამაზე:



სურ. 13.32

ანგარიშის კონსტრუქტორის (Report Designer) ელემენტები

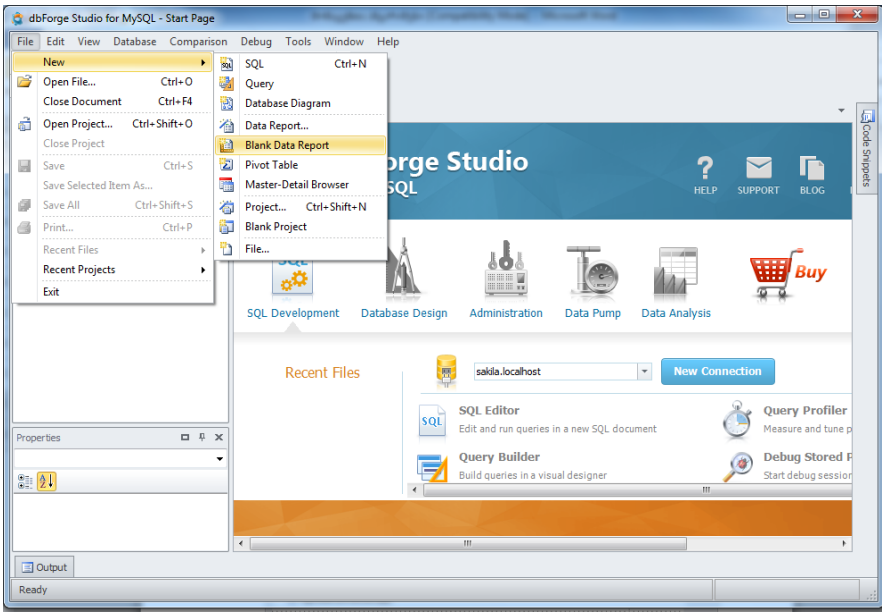
ანგარიშის კონსტრუქტორი შედგება შემდეგი ელემენტებისაგან:



სურ. 13.33

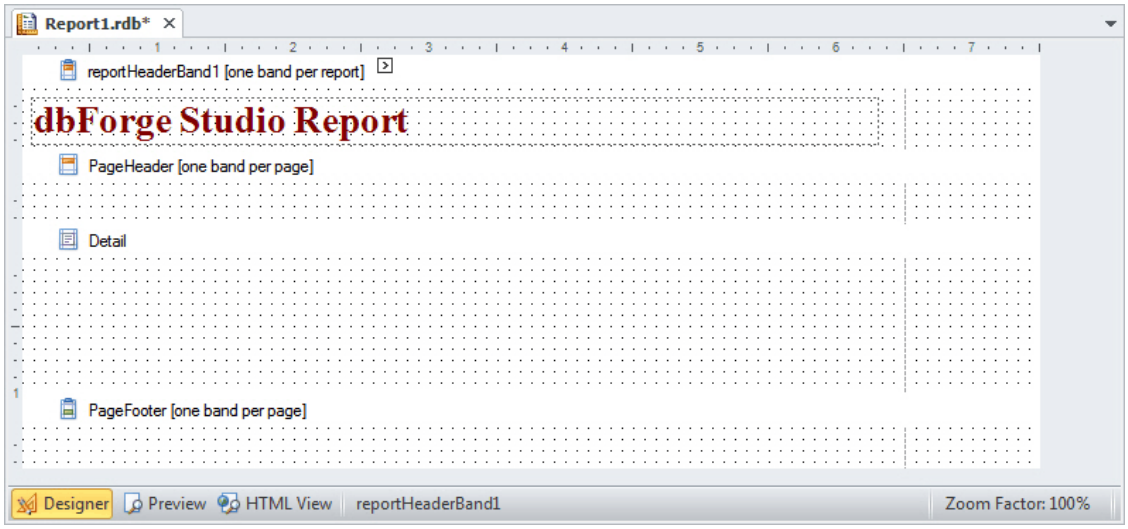
9.სტატიკური ანგარიშის შექმნა

1. გავხსნათ dbForge Studio.
2. **File** მენიუში ვირჩევთ **New** > **Blank Data Report**.



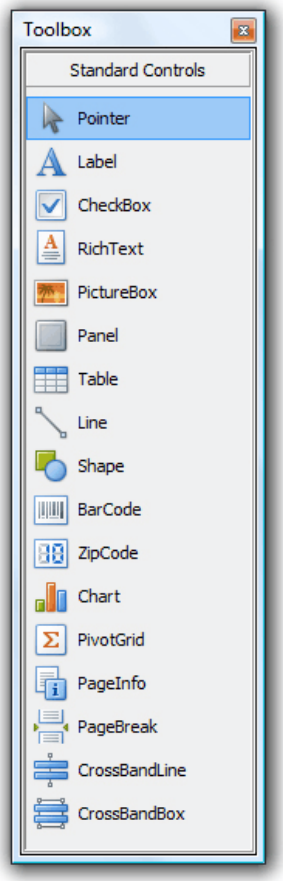
სურ. 13.34

ახლადშექმნილ ანგარიშში დოკუმენტის არეს კონსტრუქტორს (designer) ექნება შემდეგი სახე:



სურ. 13.35

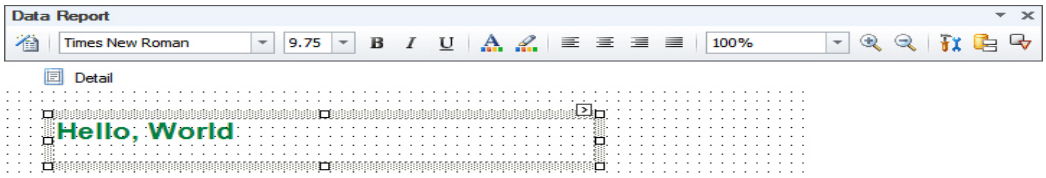
3. **Data Report** ინსტრუმენტების სტრიქონზე ვაჭერთ იკონაზე . გაიხსნება **Toolbox** ფანჯარა.



სურ. 13.36

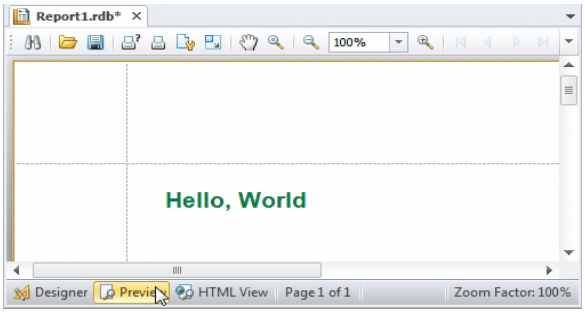
4. **Toolbox** ფანჯარაში ვირჩევთ **Label** მართვის ელემენტს და გადაგვაქვს ანგარიშის **Detail Band** -ში.

5. ორჯერ ვაწკაპუნებთ შექმნილ label-ში რედაქტორის გააქტიურებისათვის, სადაც შეგვყავს ტექსტი (მაგალითად, **"Hello, World!").** ფერისა და ფონტის დაყენების შემდეგ მივიღებთ:



სურ. 13.37

6. ახლა გავხსნათ **Preview** ჩანართი:



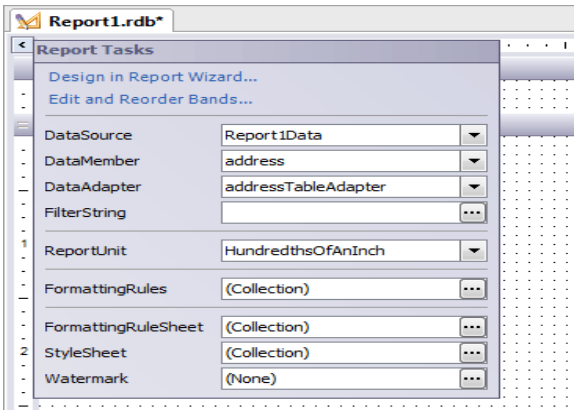
სურ. 13.38

10.მარტივი ანგარიშის შექმნა

dbForge Studio-ს მეშვეობით მარტივი ანგარიშის შექმნისათვის საჭიროა შემდეგი ქმედებების შესრულება:

- ანგარიში შეზღუდულია მონაცემებით, რომელიც არსებობენ Sakila მონაცემთა ბაზის ცხრილში **address**. ცხრილის ანგარიშში დასამატებლად **Database Explorer** -დან უნდა გადავიტანოთ **Data Source** ფანჯარაში.
- ვაწკაპუნებთ ანგარიშის კონსტრუქტორის მარცხენა-ზედა კუთხეში ანგარიშის smart ჩანართის გახსნის მიზნით.

ვირჩევთ **Edit and Reorder Bands...** ოპციას. მაგალითად, ვირჩევთ PageHeader კოლონტიტულს და ამოვაგდებთ მას, თუ მისი საჭიროება არ არსებობს. სამაგიეროდ, ვამატებთ **ReportHeader** კოლონტიტულს.



სურ. 13.39

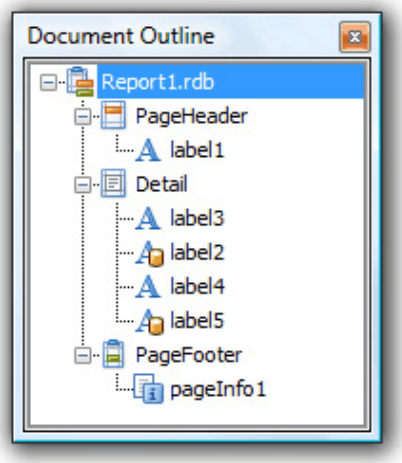
3. გადაგვაქვს **Label** მართვის კომპონენტი Toolbox-დან Report Header-ში (მსგავსად სტატიკური ანგარიშისა) და **Customers** ცხრილის **Customers** to **Customers' Address List** ველში ვადგენთ მისთვის **Text** თვისებას.

4. ვქმნით ორ ცხრილს და მათთვის შესაბამისად ვადგენთ **Text** თვისებებს: "Phone:" და "Address:".

5. შევქმნათ ორი label კომპონენტი **address** და **phone** ველებისათვის. ამისათვის, შესაბამისი ველების იტემები გადაგვაქვს **Data Source** ხიდან ანგარიშის **Detail** არეში.

6. გვერდების ნუმერაციისათვის ანგარიშის ყოველ გვერდზე Toolbox-დან გადავიტანოთ **PageInfo** იტემი და განვათავსოთ **PageFooter** არეში. გვერდების ნომრების ფორმატის შეცვლისათვის მარჯვენა ზედა კუთხეში დაწკაპუნებით გავხსნათ smart ჩანართი. მაგალითად, ვაყენებთ თვისებისათვის მნიშვნელობას **"Current of Total" Page Numbers**.

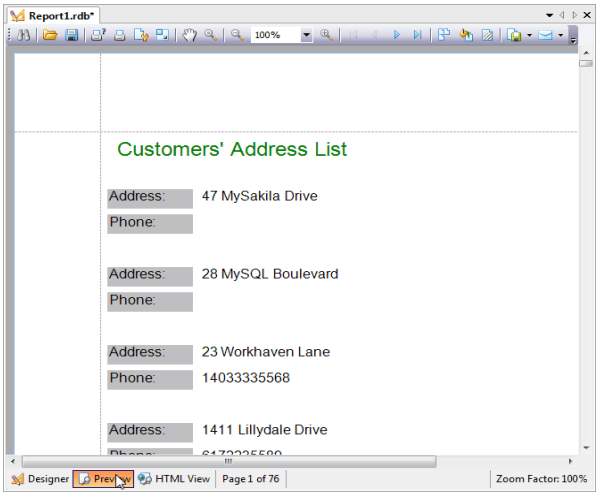
7. შექმნილი ანგარიშის სტრუქტურა აისახება **Document Outline** ფანჯარაში.



სურ. 13.40

8. **File** მენიუში დაწკაპუნებით **Save Report1.rdb** -ზე ამ ანგარიშს ვინახავთ master-detail ანგარიშში გამოყენებისათვის.

9. ახლა გავხსნათ **Preview** ჩანართი დოკუმენტის ქვედა ნაწილში:



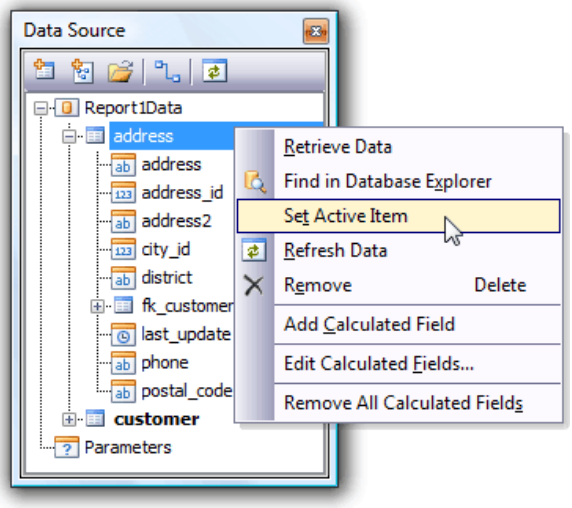
სურ. 13.41

11.Master-Detail Report-ის შექმნა

1. გავხსნათ ანგარიში: File მენიუში ვაწკაპუნებთ **Open file**.

2. master-detail ანგარიშის შექმნისათვის საჭიროა **customer** ცხრილის დამატება ანგარიშში, რისთვისაც ცხრილი გადაგვაქვს **Database Explorer** -დან **Data Source** ფანჯარაში.

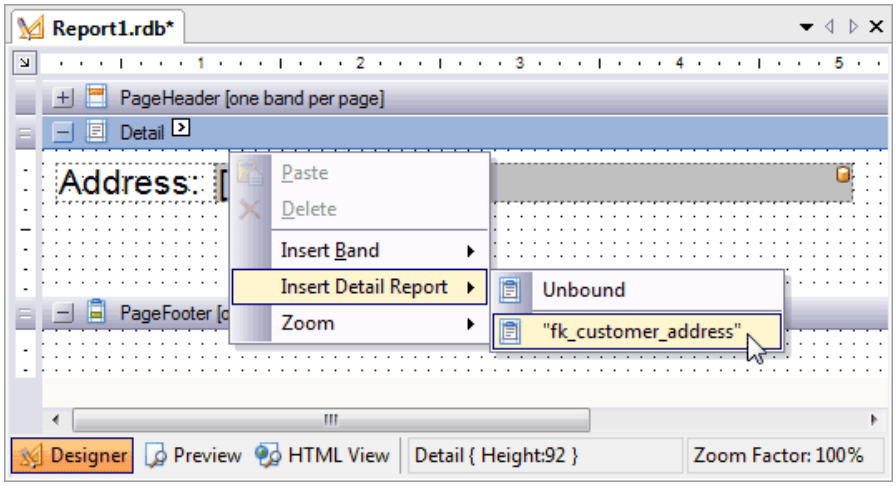
3. **Data Source** ფანჯარაში მარჯვენა დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ **Set Active Item**:



სურ. 13.42

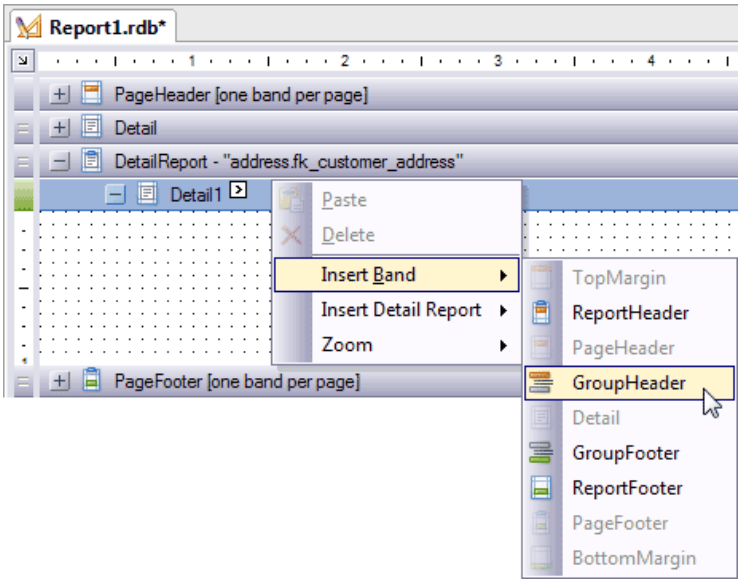
აღსანიშნავია, რომ **Data Source** ხეში fk_customer_address უნდა იყოს აქტიურ მდგომარეობაში. თუ ეს ასე არ არის, მაშინ **Data Source** ხეში **Data Source** ვაწკაპუნებთ ინსტრუმენტების სტრიქონის **Refresh Data** ღილაკზე.

4. ახალი detail ანგარიშის დამატებისათვის, მარჯვენა დაწკაპუნებით ანგარიშზე ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ პუნქტს **Insert Detail Report** და ვაწკაპუნებთ **fk_customer_address** ოპციაზე. შედეგად გამოჩნდება **Detail Report**:



სურ. 13.43

5. შემდეგ detail ანგარიშზე მარჯვენა დაწკაპუნებით ვხსნით კონტექსტურ მენიუს, სადაც ვირჩევთ პუნქტს **Insert Band** და ვაწკაპუნებთ **Group Header**. ამის შედეგად detail ანგარიშს დაემატება **GroupHeaderBand**:



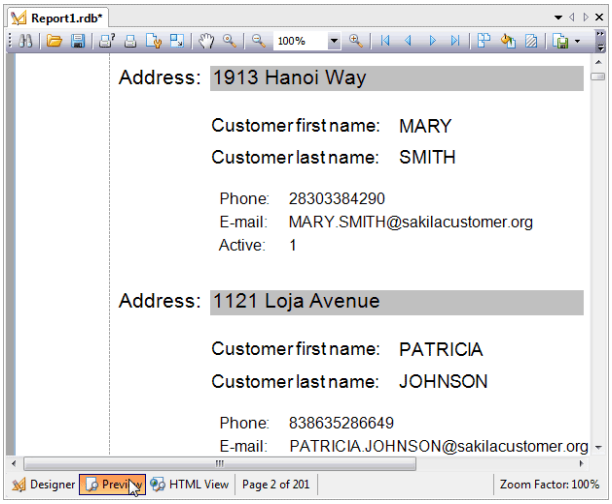
სურ. 13.44

ჩვენ შეგვიძლია დავამატოთ Label კომპონენტი GroupHeader-ში და მასში შევცვალოთ ტექსტი "Customer details:"-ით.

6. **Data Source** ხის **fk_customer_address**-დან ანგარიშის **Detail Report** ზოლში დავამატოთ **first_name, last_name** და **email** ველები. შესაბამისად ვამატებთ label -ებს "Customer first name:", "Customer last name:", "E-mail:".

ჩვენ შეგვიძლია აგრეთვე **active** ველის გადატანა მაუსის მარჯვენა ღილაკით. ჩნდება კონტექსტური მენიუ, სადაც ვირჩევთ **CheckBox** კომპონენტს. ვცვლით CheckBox ტექსტს "Active"-ით. და ა.შ.

8. გავხსნათ **Preview** ჩანართი დოკუმენტის ქვედა ნაწილში:



სურ. 13.45

12.ანგარიშების შენახვა და ჩატვირთვა

dbForge Studio for MySQL-ის გამოყენებით შექმნილი ანგარიში შეგვიძლია შევისუროთ .rdb გაფართოების ფაილში.

ანგარიშის შენახვისათვის **File** მენიუში ვაწკაპუნებთ **Save [Document Name].rdb** ან ვაჭერთ **Ctrl+S**. ეს ქმედებები აისახება **Save File As** დიალოგურ ფანჯარაში.

არსებული ანგარიშის .rdb ფაილიდან ჩატვირთვისათვის **File** main მენიუში ვაწკაპუნებთ **Open File** ან ვაჭერთ **Ctrl+O**. ეს ქმედებები აისახება **Open File** დიალოგურ ფანჯარაში.

ანგარიშების ექსპორტი სხვადასხვა ფორმატით

შექმნილი ანგარიშის PDF, HTML, MHT, RTF, XLS, XLSX, CSV, ტექსტური ფაილის ან გამოსახულების ფაილის ფორმატში ექსპორტირებისათვის საჭიროა შემდეგი მოქმედებების შესრულება:

1. ვაწკაპუნებთ ღილაკზე **Preview** ანგარიშის ქვემოთ.
2. დოკუმენტის ინსტრუმენტების სტრიქონში ვაწკაპუნებთ ღილაკზე **Export Document** და ვირჩევთ სანახველ ფორმატს.
3. შემდეგ განვსაზღვრავთ საჭირო პარამეტრებს (რაც კონკრეტული ფორმატის მიხედვით განისაზღვრება).
4. ვინახავთ ფაილს.

ლაბორატორიული სამუშაო14
MySQL მონაცემთა ბაზებთან მუშაობა PHP საშუალებით

სამუშაოს მიზანი:

1.შესავალი MySQL და PHP

2.ინტერნეტ-მაღაზიის შექმნა PHP და MySQL მეშვეობით

MySQL-თან ურთიერთქმედება PHP-ს მძლავრ ინსტრუმენტად აქცევს. ამჯერად განვიხილოთ MySQL-სა და PHP-ის ერთობლივი მუშაობის ელემენტები, რომელიც შემდგომ იქნება გამოყენებული ინტერნეტ-მაღაზიის შექმნის პროცესში.

1.შესავალი MySQL და PHP

მიერთება სერვერთან. ფუნქცია mysql_connect. მონაცემთა ბაზასთან მუშაობის დაწყებამდე აუცილებელია მასთან ქსელური შეერთების დამყარება და აგრეთვე მომხმარებლის ავტორიზაციის ჩატარება. ამისათვის გამოიყენება ფუნქცია mysql_connect():

resource mysql_connect([string $server[,string $username[,string $password]]])

ეს ფუნქცია აყენებს ქსელურ შეერთებას MySQL მონაცემთა ბაზასთან, რომელიც განთავსებულია $server ჰოსტზე (უსიტყვოდ localhost, ე.ი. მიმდინარე კომპიუტერი) და აბრუნებს ღია შეერთების იდენტიფიკატორს. რეგისტრაციის დროს მიეთითება მომხმარებლის სახელი $username და პაროლი $password (უსიტყვოდ მომხმარებლის სახელი, ვისგანაც გაშვებულია მიმდინარე პროცესი – სკრიპტების გაწყობის დროს: root და ცარიელი პაროლი):

```
<?
$dblocation = “localhost”; //სერვერის სახელი
$dbuser = “root”; //მომხმარებლის სახელი
$dbpasswd = “”; //პაროლი
//ხორციელდება მონაცემთა ბაზის სერვერთან შეერთება
//ფუნქციის გამოძახების წინ @ სიმბოლოთი ვაჩერებთ შეცდომების გამოტანას
$dbcnx = @ mysql_connect($dblocation, $dbuser, $dbpasswd);
if (!$dbcnx) //თუ დესკრიპტორი 0 -ის ტოლია, მაშინ შეერთება არ არის დამყარებული
{
//გამოგვაქვს გაფრთხილება
echo(“<P>მოცემულ მომენტში მონაცემთა ბაზის სერვერი მიუწვდომელია, ამიტომ გვერდის კორექტული ასახვა შეუძლებელია.</P>”);
exit ();
}
?>
```

ცვლადებში $dblocation, $dbuser და $dbpasswd ისურება სერვერის სახელი, მომხმარებლის სახელი и პაროლი.

სერვერთან შეერთების გაწყვეტა. ფუნქცია mysql_close. MySQL – სერვერის შეერთება ავტომატურად დაიხურება სცენარის მუშაობის დამთავრების შემდგომ ან mysql_close ფუნქციის გამოძახების შემთხვევაში.

Bool mysql_close ([resource $link_identifier])

ეს ფუნქცია წყვეტს MySQL – სერვერის შეერთებას და ოპერაციის წარმატებით შესრულების დროს აბრუნებს true-ს, ხოლო წინააღმდეგ შემთხვევაში false-ს. ფუნქცია არგუმენტის სახით იღებს მონაცემთა ბაზის სერვერთან შეერთების დესკრიპტორს, რომელსაც mysql_connect ფუნქცია აბრუნებს.

მაგალითად:

```
<?
$dblocation = „localhost“; //სერვერის სახელი
$dbuser = „root“; //მომხმარებლის სახელი
$dbpasswd = „“; //პაროლი
//ხორციელდება მონაცემთა ბაზის სერვერთან შეერთება
//ფუნქციის გამოძახების წინ @ სიმბოლოთი ვაჩერებთ შეცდომების გამოტანას
$dbcnx = @ mysql_connect($dblocation, $dbuser, $dbpasswd);
if (!$dbcnx) //თუ დესკრიპტორი 0 -ის ტოლია, მაშინ შეერთება არ არის დამყარებული
{
//გამოგვაქვს გაფრთხილება
echo(„<P>მოცემულ მომენტში მონაცემთა ბაზის სერვერი მიუწვდომელია, ამიტომ გვერდის კორექტული ასახვა შეუძლებელია.</P>“);
exit ();
}
if (mysql_close($dbcnx)) //ვწყვეტთ შეერთებას
{
echo(„მონაცემთა ბაზის შეერთება შეწყვეტილია“);
}
else
{
echo(„შეერთების შეწყვეტა არ მოხერხდა“);
?>
```

MySQL –ის PHP -თან ერთად მუშაობის მაგალითი

მონაცემთა ბაზასთან მიერთება

მაგალითის სახით განვიხილოთ მონაცემთა ბაზის ცხრილის შექმნა, რისთვისაც ვიყენებთ შემდეგ ბრძანებას:

CREATE TABLE friends (name VARCHAR(30), fav_color VARCHAR(30), fav_food VARCHAR(30), pet VARCHAR(30));

შემდეგ შეგვაქვს მონაცემები ცხრილში:

INSERT INTO friends VALUES (“Rose”, “Pink”, “Tacos”, “Cat”), (“Bradley”, “Blue”, “Potatoes”, “Frog”), (“Marie”, “Black”, “Popcorn”, “Dog”), (“Ann”, “Orange”, “Soup”, “Cat”)

Print “Your table has been created”;

PHP ფაილის მონაცემთა ბაზასთან მიერთება შესაძლებელია ქვემოთ მოყვანილი კოდის გამოყენებით.

```
<?php
// Connects to your Database
mysql_connect(„your.hostaddress.com“, „username“, „password“) or die(mysql_error());
mysql_select_db(“Database_Name”) or die(mysql_error());
Print “Your table has been populated”;
?>
```

სადაც ჩვენ შეგვიძლია საჭიროებისამებრ შევცვალოთ server, username, password, and Database_Name.

მონაცემთა დაბრუნება.

ვთქვათ ჩვენ გვინდა მონაცემთა გამოტანა ახლადშექმნილი ცხრილიდან “friends”

```
// Collects data from “friends” table
$data = mysql_query(“SELECT * FROM friends”)
or die(mysql_error());
```

შემდეგ ჩვენ განვათავსებთ ამ ინფორმაციას მასივში:

```
// puts the “friends” info into the $info array
$info = mysql_fetch_array( $data );
```

ახლა ამოვბეჭდოთ მონაცემები:

```
// Print out the contents of the entry
Print “<b>Name:</b> “.$info[‘name’] . “ “;
```

```
Print “<b>Pet:</b> “.$info[‘pet’] . “ <br>”;
```

თუმცა ამგვარად ჩვენ მხოლოდ ერთჯერად შეტანას თუ შევძლებთ მონაცემთა ბაზაში. თუ გვსურს ინფორმაციის სრულად შეტანა, მაშინ ჩვენ მოგვიწევს ციკლის გამოყენება. მაგალითად:

```
while($info = mysql_fetch_array( $data ))
{
Print “<b>Name:</b> “.$info[‘name’] . “ “;
Print “<b>Pet:</b> “.$info[‘pet’] . “ <br>”;
}
```

ახლა ყოველივე ზემოთ მოყვანილი ჩავსვათ php კოდში:

```
<?php
// Connects to your Database
mysql_connect(“your.hostaddress.com”, “username”, “password”) or die(mysql_error());
mysql_select_db(“Database_Name”) or die(mysql_error());
$data = mysql_query(“SELECT * FROM friends”)
or die(mysql_error());
Print “<table border cellpadding=3>”;
while($info = mysql_fetch_array( $data ))
{
Print “<tr>”;
Print “<th>Name:</th> <td>”.$info[‘name’] . “</td> “;
Print “<th>Pet:</th> <td>”.$info[‘pet’] . “ </td></tr>”;
}
Print “</table>”;
?>
```

SQL მოთხოვნები PHP -თან ერთად

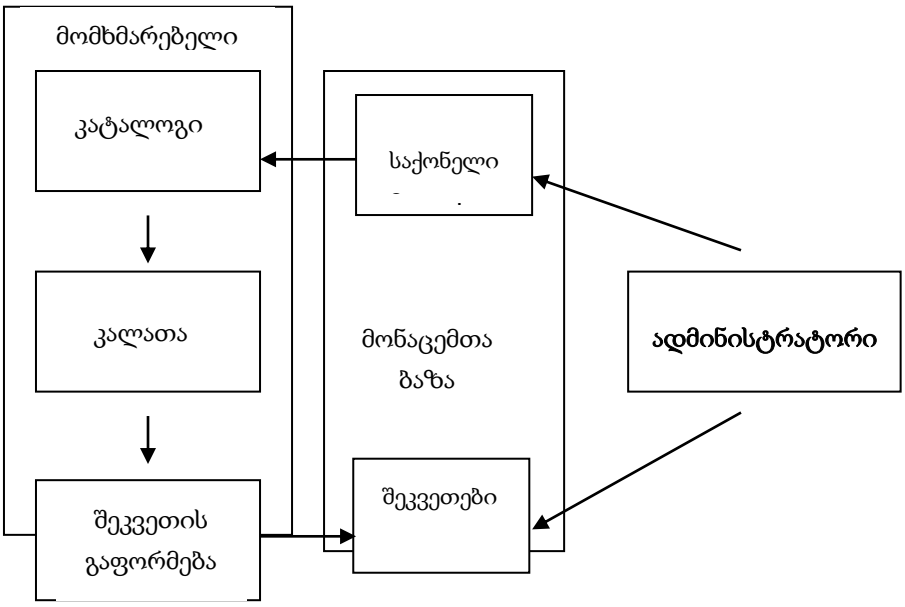
მაგალითის სახით შევქმნათ ჩვენი მონაცემთა ბაზის მოთხოვნა, სადაც ამორჩევის ლოგიკური პირობა წარმოდგენილია WHERE ბრძანების მეშვეობით:

```
<?php
// Connects to your Database
mysql_connect(“your.hostaddress.com”, “username”, “password”) or die(mysql_error());
mysql_select_db(“Database_Name”) or die(mysql_error());
$data = mysql_query(“SELECT * FROM friends WHERE pet=’Cat’”)
or die(mysql_error());
Print “<table border cellpadding=3>”;
while($info = mysql_fetch_array( $data ))
```

```
{
Print “<tr>”;
Print “<th>Name:</th> <td>”.$info[‘name’] . “</td> “;
Print “<th>Color:</th> <td>”.$info[‘fav_color’] . “</td> “;
Print “<th>Food:</th> <td>”.$info[‘fav_food’] . “</td> “;
Print “<th>Pet:</th> <td>”.$info[‘pet’] . “ </td></tr>”;
}
Print “</table>”;
?>
```

2.ინტერნეტ-მაღაზიის შექმნა PHP და MySQL მეშვეობით

განვიხილოთ გამარტივებული ინტერნეტ-მაღაზიის შექმნის პროცესი, სადაც გვაქვს მაღაზია ე.წ. ვიტრინის სახით (საქონლის აღწერილობით) და ასევე ვირტუალური ე.წ. „კალათა“ (შერჩეული საქონლის აღწერილობით), რომელიც იმავდროულად წარმოადგენს შეკვეთას. ამგვარად, გვაქვს მხოლოდ ორი ცხრილი – საქონელი და შეკვეთები. კალათა რეალიზდება Cookies მექანიზმების საშუალებით.



სურ. 14.1

დასაწყისისათვის, მაღაზიის შექმნის პროცესი დავყოთ რამდენიმე ეტაპად:

- მონაცემთა ბაზის აგება მაღაზიისათვის და „საქონელი“ ცხრილის შევსება,
- საქონლის კატალოგის დათვალიერება,
- კალათას ფორმირება,
- შეკვეთის გაფორმება,
- მიღებული შეკვეთების დათვალიერებე ადმინისტრატორის მიერ.

მონაცემთა ბაზის შექმნა. ინფორმაციის მომზადება

აუცილებელია შეიქმნას ფაილი სახხელწოდებით install.php, რომელიც გაშვებისას შექმნის ცხრილებს და მათში შეიტანს მონაცემებს.

ფუნქცია № 0: აუცილებელი ცხრილების შექმნა მონაცემთა ბაზაში

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის  სახელი","პაროლი") or die("MySQL Connection Failed");
/* მონაცემთა ბაზის სერვერთან მიერთება */
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection Failed");
/* მონაცემთა ბაზის არჩევა*/

function install() {

$content="ცხრილების შექმნა<br/>";
mysql_query("create table products(id int auto_increment primary key, name tinytext, section tinytext, description text, price float)") or die("ცხრილი საქონლით არ შექმნილა<br/>");
$content=$content." ცხრილი საქონლით შეიქმნა<br/>";
/* ცხრილი საქონლით<*/
        mysql_query("insert into products(name, section, description, price) values('ელექტრული ჩაიდანი WV-232', 'ჩაიდნები', 'Tefal კომპანიის უკანასკნელი ნაწარმი ჩაიდნების დარგში', '2950')") or die("საქონელი არ დამატებულა<br/>");
$content=$content."საქონელი დაემატა<br/>";

        mysql_query("insert into products(name, section, description, price) values('ელექტრული ჩაიდანი WV-231', 'ჩაიდნები', 'კლასიკური ჩაიდანი ნეონის განათებით', '2630')") or die("("საქონელი არ დამატებულა<br/>");
$content=$content." საქონელი დაემატა<br/>";

        mysql_query("insert into products(name, section, description, price) values('ხორცის საკეპი MK-415', 'ხორცის საკეპები', 'ყველაზე მძლავრი ხორცის საკეპი დღეისათვის – ხორცს კეპავს ძვლებთან ერთად!', '6400')") or die("საქონელი არ დამატებულა<<br/>");
$content=$content." საქონელი დაემატა<br/>";
/* ახალი საქონლის დამატება */
```

```
mysql_query("create table purchases(id int auto_increment primary key, date tinytext, name tinytext, address tinytext, email tinytext, cart tinytext)") or die("ცხრილი საქონლით არ შექმნილა<br/>");
$content=$content." ცხრილი საქონლით შეიქმნა<br/>";
/* ცხრილი საქონლით */

return $content;
/* ფუნქცია აბრუნებს ცხრილების შექმნისა და მათში ჩანაწერების შედეგებს */
}

echo install();
/* ფუნქციის მუშაობის შედეგების ეკრანზე გამოტანა*/
?>
```

კატალოგი

შემდეგ ეტაპზე საჭიროა საქონლის კატალოგის შექმნა. ამისათვის ჯერ mysql_query ფუნქციის დახმარებით products ცხრილიდან ყველა მონაცემს ვიღებთ $result ცვლადში. for ციკლის მეშვეობით გადავარჩევთ მონაცემთა ყველა სტრიქონს, სადაც ისინი წარმოდგენილი არიან mysql_fetch_array ფუნქციის მეშვეობით.

შევქმნათ ფაილი catalogue.php კატალოგით.

ფუნქცია № 1: საქონლის კატალოგი

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or die("MySQL Connection Failed");
/* მონაცემთა ბაზის სერვერთან მიერთება */
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection Failed");
/* მონაცემთა ბაზის არჩევა */

function catalogue() {

$result=mysql_query("select * from products");
/* ყველა საქონლის შედეგებს ვიღებთ ცვლადში $result*/

$content="";
/* ვქმნით ცვლადს $content, რომელშიც ჩაიწერება ყველაფერი, რაც საჭიროა მომხმარებელმა რომ სუროს (HTML-ფაილის შექმნა) */
```

```
for ($i=0; $i<mysql_num_rows($result); $i++) {
/* for ოპერატორის მეშვეობით გადავარჩიოთ ყველა სტრიქონი ცხრილში საქონლით */

        $row=mysql_fetch_array($result);
/* ვიღებთ მონაცემთა მასივს - ერთი საქონლისათვის*/

  $content=$content."--------------------<br/><br/>
        <strong>".$row['name']."</strong><br/>".$row['description']."<br/><br/>
        <strong>ფასი:</strong> ".$row['price']." ლარი<br/><br/>
                <a href=\"catalogue.php?addtocart=".$row['id']."\">ჩავდოთ
          კალათაში</a><br/><br/>";
/* შევადგინოთ ინფორმაცია საქონლის შესახებ*/

}
return $content;
/* ვაბრუნებთ $content ცვლადში */

}

echo catalogue();
/* გამოგვაქვს შედეგები ეკრანზე*/

echo "<a href=\"cart.php\">კალათაში გადასვლა</a>";
/* ოპცია „კალათაში გადასვლა“*/
?>
```

შემოწმებისათვის ცხრილში products დავამატოთ რამდენიმე საქონელი და შევამოწმოთ მათი არსებობა კატალოგში.

საქონლის დამატება კალათაში

მთელი ინფორმაცია კალათის შესახებ დროებითია და ისურება კომპიუტერში ბრაუზერის ფანჯრის დახურვამდე. ამისათვის გამოიყენება cookies.

Cookies – წარმოადგენს ცვლადებს ფორმატით სახელი=მნიშვნელობა, რომლებიც ისურება მომხმარებლის კომპიუტერზე და მათი წვდომა შესაძლებელია საიტიდან, სადაც ჩაწერილი არიან. cookies-ში მონაცემების ჩასაწერად გამოიყენება PHP-ის ფუნქცია setCookie($name,$value) ორი პარამეტრით – ცვლადის სახელი და მისი მნიშვნელობა. ჩვენს ცვლადს დავარქვათ cart, ხოლო მისი მნიშვნელობა იქნება

კალათას შემადგენლობა საქონლის იდენტიფიკატორების სახით, გამოყოფილი რაიმე ნიშნით (ვთქვათ, |). კალათადან მონაცემების მისაღებად გამოიყენება სტანდარტული მასივი $_COOKIE.

ოპცია „კალათაში ჩადება“ წარმოადგენს მიმართვას php-ფაილზე საქონლის იდენტიფიკატორით, რომელიც გადაეცემა GET მეთოდით, მაგალითად, href=”cataloge.php?addtocart=1”. ამავე ფაილში, კატალოგის გამოტანის ფუნქციის წინ, საჭიროა ჩაისვას სამისამართო სტრიქონში addtocart ცვლადის არსებობის შემოწმების ფუნქცია. ამისათვის, მოვათავსოთ ეს ფუნქცია კატალოგიან ფაილში catalogue() ფუნქციის აღწერის წინ:

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection Failed");

function cartStatus() {
.....
}

function catalogue() {
.....
}
echo catalogue();

?>
```

ფუნქცია №2: საქონლის დამატება და კალათას მდგომარეობა

```
function cartStatus() {

/* ნაწილი პირველი – საქონლის დამატება კალათაში */

$cart="";
/* შევქმნათ ცვლადი $cart, რომელშიც ჩავწერთ კალათას*/

if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
/* თუ cookies-ში არსებობს ცვლადი „cart“,  მაშინ მისი მნიშვნელობა მივანიჭოთ ცვლადს $cart */

if (@$_GET[addtocart]) {
```

/* თუ სამისამართო სტრიქონში არსებობს ცვლადი „addtocart“ (მეთოდი GET), მაშინ*/

```
if (!@$_COOKIE['cart']) {
/* თუ cookies-ში არ არსებობს ცვლადი „cart“*/

        $cart=$_GET['addtocart'];
/* ცვლადს $cart მივანიჭოთ კალათაში დასამატებელი საქონლის
  იდენტიფიკატორი*/

}
else {
/* თუ cookies-ში არსებობს ცვლადი „cart“,  მაშინ საჭიროა იდენტიფიკატორებს
  შორის ჩავსვათ ნიშანი  | და მხოლოდ ამის შემდეგ შეიძლება ახალი
  იდენტიფიკატორის დამატება*/

        $cart=$cart."|". $_GET['addtocart'];

}

        setCookie("cart","$cart");
/* cookies-ში ცვლადი „cart“ შევცვალოთ $cart-ით კალათას ახალი, შეცვლილი
  შემადგენლობით*/

}

/* ნაწილი მეორე – კალათაში საქონლის რაოდენობის მიღება*/

if ($cart=="") { $cartCount="0"; }
```

/* თუ ცვლადი $cart, რომელიც ადრე cookies -დან იყო მიღებული, წარმოადგენს ცარიელ სტრიქონს, ეს ნიშნავს, რომ კალათა ცარიელია და ცვლადს, რომელიც შეიცავს საქონლის რაოდენობას, ვუწოდებთ $cartCount მიენიჭება მნიშვნელობა 0 */

```
else {
/* თუ ცვლადი $cart არ არის ცარიელი და კალათაში არის საქონელი... */

$cartArr=explode("|",$cart);
```

/* ...უნდა შეიქმნას მასივი საქონლის იდენტიფიკატორებით, რისთვისაც ვიყენებთ სტანდარტულ ფუნქციას explode(), რომელიც სტრიქონს ყოფს ქვესტრიქონებად და შედეგად კალათაში გვექნება იმდენი საქონელი, რამდენი

```
  ელემენტიც არის მასივში*/

$cartCount=count($cartArr);
        /* სტანდარტულ ფუნქცია count() აბრუნებს მასივის ბრჭყალებში
  მითითებული ელემენტების რაოდენობას; ვანიჭებთ ამ მნიშვნელობას
  $cartCount ცვლადს*/

}

return $cartCount;
/* ფუნქცია აბრუნებს კალათაში საქონლის რაოდენობას*/

}
```

ფუნქცია დაწერილია, ჩასმულია საჭირო ადგილას, მაგრამ არ არის გაშვებული. მისი გაშვებისათვის ვიძახებთ cartStatus(), მაგრამ მაშინ იგი დაამატებს ახალ საქონელს კალათაში. მისი გაშვებისათვის ვკრეფთ:

```
<?

@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or
die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection
Failed");

function cartStatus() {
.....
}

function catalogue() {
.....
}

$cartCount=cartStatus();
/* ვუშვებთ ფუნქციას და ვაბრუნებთ მის მნიშვნელობას ცვლადში $cartCount */

echo "საქონლის რაოდენობა კალათაში: ".$cartCount." шт.<br/><br/><br/>";
/* ახლა კატალოგიანი გვერდის სულ ზედა ნაწილში იქნება მითითებული
მოცემული მომენტისათვის კალათაში საქონლის რაოდენობა*/

echo catalogue();
```

```
/* გამოგვაქვს საქონლის კატალოგი*/

?>
```

კალათა

კალათა შედგება შემდეგი ღილაკებისაგან: შეკვეთის გაფორმება, კალათისა და საქონლის სიის გასუფთავება. შევქმნათ ფაილი cart.php.

ფუნქცია №3: კალათა

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection Failed");

function cart() {

$cart="";
/* შევქმნათ ცვლადი $cart, რომელშიც ჩავწერთ კალათას*/

if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
/*თუ cookies-ში არსებობს ცვლადი „cart“,  მაშინ მისი მნიშვნელობა მივანიჭოთ ცვლადს $cart*/

$cartArr=explode("|",$cart);
/* გამოვიყენოთ ჩვენთვის უკვე ნაცნობი ფუნქცია explode, რათა მივიღოთ კალათაში არსებული საქონლის იდენტიფიკატორების მასივი*/

if ($cart=="") {
return "კალათა ცარიელია";
        /* თუ ცვლადი $cart წარმოადგენს ცარიელ სტრიქონს, მაშინ ფუნქციის
  შედეგი იქნება წარწერა „კალათა ცარიელია“*/
}
else {
$cartCount=count($cartArr);
        /* ჩავწეროთ $cartCount ცვლადში კალათაში საქონლის რაოდენობა,
  რომელიც count() ფუნქციის შედეგად არის მიღებული*/

$cartTotalPrice=0;
        /* შევქმნათ ცვლადი $cartTotalPrice, რომელშიც გამოვთვლით საერთო
```

ღირებულებას*/

```
        foreach($cartArr as $k=>$v) {
                    /* foreach ოპერატორის მეშვეობით გადავარჩოით მასივის
                  ყველა ელემენტი, სადაც $k – ელემენტის ნომერი მასივში
                  (ნულიდან დაწყებული), $v – საქონლის იდენტიფიკატორი */

                $result=mysql_query("select * from products where id='$v'");
                    /* ვიღებთ $result ცვლადში შედეგს შემდეგი mysql-
                  მოთხოვნისათვის „ამოვარჩიოთproducts  ცხრილიდან ყველა,
                  სადაც იდენტიფიკატორი ($id) ტოლია $v ცვლადისა
                  (იდენტიფიკატორი საქონლის მასივიდან)“ */

                $row=mysql_fetch_array($result);
          /* მიღებული მონაცემები გარდავქმნათ $row ასოცირებულ მასივად*/

                    $number=$k+1;
                    /* თითოეულ საქონელს კალათაში უნდა ჰქონდეს ნომერი
                  ერთიდან დაწყებული, ხოლო php აწარმოებს ანგარიშს ნულიდან
                  დაწყებული, რისთვისაც php -ს მიერ მინიჭებულ რიგით ნომერს
                  დაემატება ერთიანი. მიღებული მნიშვნელობა ჩავწეროთ $number
                  ცვლადში*/

                    $content=$content.$number." - ".$row['name']." - ".$row['price']."
                  руб.<br/>";
                    /* ვაფორმებთ მიღებულ მონაცემებს*/

                $cartTotalPrice=$cartTotalPrice+$row['price'];
                    /* მორიგი საქონლის ფასი მივუმატოთ $cartTotalPrice ცვლადს
                  (ნავაჭრის საერთო ღირებულება)*/
        }

        $content=$content."<br/><br/> Всего: ".$cartTotalPrice." руб. ";
        /* დავამთავროთ საქონლის გადარჩევა კალათაში და გამოვიყვანოთ
ნავაჭრის საერთო ღირებულება*/

        $content=$content."<br/><br/><a href=\"orderform.php\">შეკვეთის
გაფორმება</a>\n";
        /* ოპცია „შეკვეთის გაფორმება“*/
```

```
}
return $content;
/* შედეგების გამოტანა */

}

echo cart();
/* ფუნქციის მუშაობის შედეგი გამოგვაქვს ეკრანზე*/

?>
```

კალათის გასუფთავებისათვის შეიძლება გამოვიყენოთ ფუნქცია setCookie(), რომელშიც ცვლადს cart მივანიჭებთ ცარიელ სტრიქონს: setCookie(cart, "")

შეკვეთის გაფორმება

შეკვეთის გაფორმება შედგება ორი ნაწილისაგან: მომხმარებლის შესახებ მონაცემების შეტანის ფორმა და ამ ინფორმაციის მონაცემთა ბაზაში ჩაწერა კალათას შედგენილობასთან ერთად. ფორმა შევქმნათ ფაილში orderform.php. ტეგში form მიუთითოთ method=”post” და action=”order.php”. ფორმაში უნდა იყოს შემდეგი ველები: name (გვარი და სახელი), address (საფოსტო ინდექსი), email (ელექტრონული ფოსტა).

ამის შემდეგ, შევქმნათ ფაილი order.php, რომელშიც მოვათავსებთ შეკვეთის გაფორმების ფუნქციას.

ფუნქცია №4: შეკვეთის გაფორმება

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or
die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection
Failed");

function order() {
$name=@$_POST['name'];
$address=@$_POST['address'];
$email=@$_POST['email'];
/* ცვლადებში $name, $address და $email მივიღებთ post მეთოდით გადაცემულ
ცვლადებს ფორმიდან */

$cart="";
```

```
/* შევქმნათ ცვლადი $cart, რომელშიც ჩავწერთ კალათას*/

if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
/*თუ cookies-ში არსებობს ცვლადი „cart“,  მაშინ მისი მნიშვნელობა მივანიჭოთ ცვლადს $cart*/
$date=time();
/* ცვლადში $date ჩავწეროთ მიმდინარე თარიღი*/

mysql_query("insert into purchases(date, name, address, email, cart) values('$date', '$name', '$address', '$email', '$cart')");
/* ნავაჭრის ცხრილში ვამატებთ ახალ ჩანაწერს*/

return "თქვენი შეკვეთა მიღებულია სერვერზე. გმადლობთ";

}

echo order();
/* ფუნქციის მუშაობის შედეგი გამოგვაქვს ეკრანზე*/

?>
```

მუშაობა შეკვეთებთან

ლაბორატორიული სამუშაოს მიზანს წარმოადგენს გვერდის მიღება ყველა ჩანაწერით. ამისათვის purchases ცხრილიდან უნდა ავიღოთ ყველა ჩანაწერი (for ციკლისა და mysql_fetch_array() ფუნქციის მეშვეობით) და გამოვიტანოთ ისინი გვერდზე. ამოცანა რთულდება იმით, რომ cart სვეტში არსებული საქონლის იდენტიფიკატორები უნდა გარდავქმნათ დასახელებებად.

შევქმნათ ფაილი allorders.php და განვათავსოთ მასში შემდეგი ფუნქცია.

ფუნქცია №5: შეკვეთის გაფორმება

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection Failed");

function allorders() {

$resultPurchases=mysql_query("select * from purchases");
```

```
/*$resultPurchases ცვლადში ვიღებთ ყველა შეკვეთას*/

$content="";
/* შევქმნათ ცვლადი $content, რომელშიც ჩავწერთ ყველაფერს იმას, რაც უნდა ვაჩვენოთ  მომხმარებელს (HTML-ფაილის შექმნა) */

for ($i=0; $i<mysql_num_rows($resultPurchases); $i++) {
/* for ოპერატორის გამოყენებით შეკვეთების ცხრილში გადავარჩიოთ ყველა სტრიქონი */
        $rowPurchases=mysql_fetch_array($resultPurchases);
        /* ცვლადში $rowPurchases ვიღებთ მონაცემთა ერთრიგიან (ერთი
  შეკვეთის) მასივს */

$content=$content."----------<br/><br/>
        <strong>".$rowPurchases['name']."</strong><br/>
        ".$rowPurchases['address']."<br/>
        ".$rowPurchases['email']."<br/>
        <strong>შეკვეთა:</strong><br/>";
/* ვადგენთ ინფორმაციას კლიენტის შესახებ */

  $cart=$rowPurchases['cart'];
/* ცვლადში cart ჩავწეროთ კალათას შედგენილობა*/

$cartArr=explode("|",$cart);
        /* გამოვიყენოთ ჩვენთვის უკვე ნაცნობი ფუნქცია explode, რათა მივიღოთ
  კალათაში არსებული საქონლის იდენტიფიკატორების მასივი*/

if ($cart!="") {
/* თუ კალათა არ არის ცარიელი */
        foreach($cartArr as $k=>$v) {
                $result=mysql_query("select * from products where id='$v'");
                $row=mysql_fetch_array($result);
                        $number=$k+1;
                        $content=$content.$number." - ".$row['name']." - ".$row['price']."
                  руб.<br/>";
        }
}
}
return $content;
/* ვაბრუნებთ ცვლადს $content */
}
```

```
echo allorders();
/* ფუნქციის მუშაობის შედეგი გამოგვაქვს ეკრანზე*/
?>
```

თუ ყველაფერი სწორედ იქნა გაკეთებული, მაშინ ფუნქციის მუშაობის შედეგი იქნება მომხმარებლის მონაცემები და მის მიერ გაკეთებული შეკვეთა.

დასკვნა:

ლაბორატორიული სამუშაოს დასრულების შემდეგ აუცილებელია მონაცემთა ბაზაში ყველა შექმნილი ცხრილის წაშლა. ამისათვის ვშლით ფაილს delete.php შემდეგი შედგენილობით:

```
<?
@mysql_connect("ჰოსტის სახელი","მომხმარებლის სახელი","პაროლი") or
die("MySQL Connection Failed");
@mysql_select_db("მონაცემთა ბაზის სახელი") or die("MySQL Database Selection
Failed");
mysql_query("delete products from products");
mysql_query("delete purchases from purchases");
?>
```

სცენარების კოდები

სათაურისა და მენიუს შაბლონი (header.phtml)

```
<?
header("Cache-control: no-cache");

$id_bask=$HTTP_COOKIE_VARS["id_bask"];
if (! isset($id_bask))
{
$uniq_ID=uniqid("ID");
setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
// შევქმნათ გასაღები
}
else
setcookie("id_bask", $id_bask, time()+60*60*24*14);
// ხელმეორედ შევქმნათ გასაღები იგივე მნიშვნელობით

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<html>
<head>
<title>წიგნების მაღაზია</title>
</head>

<body background="EULA.jpg" style="background-repeat:repeat-y"
leftmargin="130" rightmargin="5" bgProperties=fixed>
<table border="0" align="right" width="90%" cellpadding="0"
cellspacing="0">
<tr><td>

<table border="0" align="right" width="100%" >
<tr>
<td align="center" bgcolor="#cccccc">
<form action="auto.phtml" method="post">
<table>
<tr><td align="right"><font size=-2>ლოგინი:</font></td>
<td align="left"><input type=text style="width:60; height:20;"
name=login></td></tr>
<tr><td align="right"><font size=-2>პაროლი:</font></td>
<td align="left"><input type=password style="width:60;
height:20;" name=pass>
<input type=submit value=ok style="height:20;"></td></tr>
</table>
<b><small>
<?
if(isset($HTTP_სესია_VARS["log"]))
{
print $HTTP_სესია_VARS["log"];
print "<br><a href='cabinet.phtml'>პირადი კაბინეტი</a>";
}

?>
</small></b></td>
</form>
<td colspan="4" align="center" bgcolor="#ccccff">
<font face="Arial" size="+3"><i><b>წიგნების მაღაზია</b></i></font></td></tr>
<tr><td align="center" bgcolor="#aaddff" width="20%">
<a href="catalog.phtml"><b>კატალოგი</b></a></td>
<td align="center" bgcolor="#ddaaff" width="20%">
<a href="basket.phtml"><b>კალათა</b></a></td>
<td align="center" bgcolor="#aaaaff" width="20%">
```

```
<a href="reg.phtml"><b>რეგისტრაცია</b></a></td>
<td align="center" bgcolor="#ffaaff" width="20%">
<a href="order.phtml"><b>შეკვეთა</b></a></td>
<td align="center" bgcolor="#aaffee" width="20%">
<a href="exit.phtml"><b>გამოსვლა</b></a></td>
</tr>
</table>
</td></tr>
<tr><td align="center" bgcolor=<?print $color?>><font
face="Arial" size="+2">
<i><?print $title?></i></font><br>
</td></tr>
```

შაბლონი გვერდის ქვედა ნაწილისათვის (footer.phtml)

```
<tr><td><center><hr><br>
<a href="index.phtml">მთავარ გვერდზე</a></center></td></tr>
</table>
</body>
</html>
```

მთავარი გვერდი (index.phtml)

```
<?
$title="Welcome!";
$color="#ccccff";
include("header.phtml");
?>
<tr><td>
<center><h2><font color="#555599"><br>კეთილი იყოს თქვენი მობრძანება ჩვენს ელექტრონულ წიგნების მაღაზიაში! <br><br>აქ წარმოდგენილია ცნობილი გამომცემლების ლიტერატურა. ნებისმიერ მკითხველს შეუძლია იპოვოს წიგნი მისი გემოვნების მიხედვით! გთავაზობთ კომპიუტერულ ლიტერატურას, ენციკლოპედიებს, ცნობარებს, ცნობილ მხატვრულ ნაწარმოებებს და მრავალ სხვა საინტერესო რამეს!</font></h2></center>
</td></tr>
<? include("footer.phtml");

?>
```

მონაცემთა ბაზასთან მიერთების შაბლონი (conn*ect.phtml)*

```
<?
```

```
mysql_connect("localhost", "root", "") or
die ("შეუძლებელია სერვერთან შეერთება!");
mysql_select_db("books") or
die ("შეუძლებელია მონაცემთა ბაზასთან შეერთება!");
?>
```

გამომცემლობებისა და კატეგორიების სიის ნახვა (catalog.phtml)

```
<?
$title="კატალოგი";
$color="#aaddff";

include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება!");
$strSQL2="SELECT * FROM categories ORDER BY name_cat";
$result2=mysql_query($strSQL2)
or die("შეუძლებელია მოთხოვნის შესრულება!");
?>
<tr><td>
<table border=0 width=100%>
<tr><td width="50%"><center><h3>გამომცემლები</h3></center><ul>
<?
while($row=mysql_fetch_array($result1))
{?>
<li><a href="show.phtml?type=1&id_publ=
<?print $row["id_publ"];?>">
<?print $row["name_publ"];?></a>
<?}?>
</ul></td>

<td width="50%"><center><h3>კატეგორიები</h3></center><ul>
<?
while($row=mysql_fetch_array($result2))
{?>
<li><a href="show.phtml?type=2&id_cat=
<?print $row["id_cat"];?>">
<?print $row["name_cat"];?></a>
<?}?>
</ul></td>
```

```
</tr>
</table>
</td></tr>
<?
include("footer.phtml");
mysql_close();
?>
```

წიგნების სიის დათვალიერება გამომცემლის ან კატეგორიის მიხედვით (show.phtml)

```
<?
$id_publ=$HTTP_GET_VARS["id_publ"];
$id_cat=$HTTP_GET_VARS["id_cat"];
$type=$HTTP_GET_VARS["type"];

include("connect.phtml");

if ($type==1)
{
$strSQL1="SELECT name_publ FROM publishers WHERE
id_publ=".$id_publ;
$result=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება1!");
if($row=mysql_fetch_array($result))
$title=$row["name_publ"];
$strSQL1="SELECT id_book, image, author, name_book,
books.id_publ, name_publ, pages, price, books.id_cat, name_cat
FROM books, publishers, categories WHERE
books.id_cat=categories.id_cat AND
books.id_publ=publishers.id_publ AND books.id_publ=".$id_publ;
}
if ($type==2)
{
$strSQL1="SELECT name_cat FROM categories WHERE
id_cat=".$id_cat;
$result=mysql_query($strSQL1)
or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
if($row=mysql_fetch_array($result))
$title=$row["name_cat"];
$strSQL1="SELECT id_book, image, author, name_book,
books.id_publ, name_publ, pages, price, books.id_cat, name_cat
FROM books, publishers, categories WHERE
books.id_cat=categories.id_cat AND
```

```
books.id_publ=publishers.id_publ AND books.id_cat=".$id_cat;
}
$result1=mysql_query($strSQL1) or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");

include("header.phtml");
?>
<tr><td>
<table border="1" width="100%" align="right" >
<?
while($row=mysql_fetch_array($result1))
{?>
<tr>
<td align="center"><img src="images/<?print $row["image"];?>"
alt="<?print $row["name_book"];?>" border="0">
<center><a href="dobasket.phtml?type=1&id_book=
<?print $row["id_book"];?>">
<font size=-1>კალათაში ჩადება</font></a></center></td>
<td>
<table>
<tr><td align="right"><i>ავტორი: </i></td>
<td><?print $row["author"];?></td></tr>
<tr><td align="right"><i>დასახელება: </i></td>
<td><?print $row["name_book"];?></td></tr>
<tr><td align="right"><i>გამომცემლობა: </i></td>
<td><a href="show.phtml?type=1&id_publ=
<?print $row["id_publ"];?>"><?print $row["name_publ"];?></a>
</td></tr>
<tr><td align="right"><i>გვერდების რაოდენობა: </i></td>
<td><?print $row["pages"];?></td></tr>
<tr><td align="right"><i>ფასი: </i></td>
<td><?print $row["price"];?></td></tr>
<tr><td align="right"><i>კატეგორია: </i></td>
<td><a href="show.phtml?type=2&id_cat=
<?print $row["id_cat"];?>"><?print $row["name_cat"];?></a>
</td></tr>
</table>
</td>
</tr>
<?}?>
</table>
</td></tr>
<?
```

```
include("footer.phtml");
?>
```

კალათას დათვალიერება (basket.phtml)

```
<?
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

$title="Ваша კალათა";
$color="#ddaaff";
include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT COUNT(*) as count FROM basket_books
WHERE id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
$row=mysql_fetch_array($result1);
if($row["count"]==0)
{
?>
<tr><td bgcolor='#ff9999' align='center'>
<b>თქვენი კალათა ცარიელია!</b></td></tr>
<?
}
else
{
$strSQL1="SELECT image, author, name_book, pages, price, kolvo,
id_bask, books.id_book FROM books, basket_books WHERE
books.id_book=basket_books.id_book AND
id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1) or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
?>
<tr><td>
<table border="1" width="100%" align="right" >
<tr><td align="right"><i>ავტორი: </i></td>
<td align="right"><i>დასახელება: </i></td>
<td align="right"><i>ფასი: </i></td>
<td align="right"><i>რაოდენობა: </i></td>
<td></td></tr>
<?
$sum=0;
while($row=mysql_fetch_array($result1))
```

```
{
?>
<tr>
<td><?print $row["author"];?></td>
<td><b><?print $row["name_book"];?></b></td>
<td><?print $row["price"];?></td>
<td><?print $row["kolvo"];?>
<a href="dobasket.phtml?type=1&id_book=
<?print $row["id_book"];?>" title="დამატება">[ + ]</a>
<a href="dobasket.phtml?type=2&id_book=
<?print $row["id_book"];?>" title="შემცირება">[ - ]</a>
</td>
<td> <a href="dobasket.phtml?type=3&id_book=
<?print $row["id_book"];?>">ამოღება</a></td>
</tr>
<?
$sum=$sum+$row["price"]*$row["kolvo"];
}?>
<tr><td align="right"></td><td align="right"><i>სულ:
</i></td><td align="right"><?print $sum;?></td><td
align="right"></td></tr>
</table>
<tr><td><center><a href=dobasket.phtml?type=4>
<b>კალათას დაცლა</b></a></center></td></tr>
<tr><td><center><a href="order.phtml">
<b>შეკვეთის გაფორმება</b></a></center></td></tr>
<?
}
include("footer.phtml");
?>
```

მოქმედებები კალათასთან (dobasket.phtml)

```
<?
$type=$HTTP_GET_VARS["type"];
$id_book=$HTTP_GET_VARS["id_book"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

include("connect.phtml");

if($type==1) // კალათაში ჩადება
{
$strSQL="SELECT * FROM basket_books WHERE
```

```
id_book=".$id_book." AND id_bask='".$id_bask."'";
$result=mysql_query($strSQL)
or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
if ($row=mysql_fetch_array($result))
{
$strSQL="UPDATE basket_books SET kolvo=kolvo+1 WHERE
id_book=".$id_book." AND id_bask='".$id_bask."'";
}
else
{
$strSQL="INSERT INTO basket_books (id_bask, id_book,
kolvo) VALUES ('".$id_bask."',".$id_book.",1)";
}
mysql_query($strSQL);
}
else
if($type==2) // რაოდენობის შემცირება
{
$strSQL="SELECT * FROM basket_books WHERE
id_book=".$id_book." AND id_bask='".$id_bask."'";
$result=mysql_query($strSQL)
or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
if ($row=mysql_fetch_array($result))
{
if ($row["kolvo"]>1)
{
$strSQL="UPDATE basket_books SET kolvo=kolvo-1 WHERE
id_book=".$id_book." AND id_bask='".$id_bask."'";
}
else
{
$strSQL="DELETE FROM basket_books WHERE
id_book=".$id_book." AND id_bask='".$id_bask."'";
}
}
mysql_query($strSQL);

}
else
if($type==3) // კალათიდან ამოღება
{

$strSQL="DELETE FROM basket_books WHERE
```

```
id_book=".$id_book." AND id_bask='".$id_bask."'";
mysql_query($strSQL);
}
else
if($type==4) // კალათას დაცლა
{
$strSQL="DELETE FROM basket_books WHERE
id_bask='".$id_bask."'";
mysql_query($strSQL);
}
include("basket.phtml");
?>
```

რეგისტრაცია (reg.phtml)

```
<?
$title="რეგისტრაცია";
$color="#aaaaff";

$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$pass=$HTTP_POST_VARS["pass"];
$pass2=$HTTP_POST_VARS["pass2"];
$login=$HTTP_POST_VARS["login"];
$type=$HTTP_POST_VARS["type"];
$subscribe=$HTTP_POST_VARS["subscribe"];

include("connect.phtml");
if($type==1)
{

if($fam!="" && $im!="" && $addr!="" && $mail!="" && $login!=""
&& $pass!="" && $pass2!="")
{
if($pass!=$pass2)
{
$message="<tr><td bgcolor='#ff9999' align='center'><b>
პაროლისა და განმეორების ველები არ ემთხვევიან!!!</b></td></tr>";
}
else
{
$strSQL1="SELECT id_cust FROM customers
```

```
WHERE login='".$login."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება!");
if($row=mysql_fetch_array($result1))
{
$message="<tr><td bgcolor='#ff9999' align='center'>

<b>ასეთი ლოგინი უკვე არსებობს!!! აირჩიეთ სხვა ლოგინი</b></td></tr>";
}
else
{
$strSQL1="INSERT INTO customers
(fam, im, addr, mail, login, pass, subscribe)
VALUES('".$fam."','".$im."','".$addr."','".$mail.
"','".$login."','".$pass."','".$subscribe."')";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება!");
$message="<tr><td bgcolor='#66cc66' align='center'>
<b>რეგისტრაცია წარმატებით შესრულდა</b></td></tr>";
$success=true;
}
}
}
else
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>ყველა ველი არ არის შევსებული!!!</b></td></tr>";
}

include("header.phtml");
print $message;

if(!$success)
{
?>

<form action=reg.phtml method=post>
<tr><td align="center">
<small>ვარსკვლავებით აღნიშნულია აუცილებელი ველები</small>
<table border="0" width="100%" align="right" >
<tr><td align="right" width="50%"><i>გვარი: </i></td><td>
<input type=text name=fam value="<?print $fam?>">*</td></tr>
<tr><td align="right"><i>სახელი: </i></td><td>
```

```
<input type=text name=im value="<?print $im?>">*</td></tr>
<tr><td align="right"><i>მისამართი: </i></td><td>
<input type=text name=addr value="<?print $addr?>">*</td></tr>
<tr><td align="right"><i>E-mail: </i></td><td>
<input type=text name=mail value="<?print $mail?>">*</td></tr>
<tr><td align="right"><i>ლოგინი: </i></td><td>
<input type=text name=login value="<?print $login?>">*</td>
</tr>
<tr><td align="right"><i>პაროლი: </i></td><td>
<input type=password name=pass value="">*</td></tr>
<tr><td align="right"><i>Повтор пароля: </i></td><td>
<input type=password name=pass2 value="">*</td></tr>
<tr><td></td><td>
<input type="checkbox" value="1" name="subscribe">
<i>ხელმოწერა ახალი ამბების დაგზავნაზე</i></td></tr>
<input type=hidden value=1 name=type>
<tr><td align="right"></td><td>
<input type=submit value="გაგზავნა"></td></tr>
</table>
</form>
</td></tr>
<?
mysql_close();
}
include("footer.phtml");
?>
```

ავტორიზაცია auto.phtml)

```
<?
$title="ავტორიზაცია";
$color="#aaaaff";

$pass=$HTTP_POST_VARS["pass"];
$login=$HTTP_POST_VARS["login"];

include("connect.phtml");

$strSQL1="SELECT * FROM customers WHERE login='".$login.
"' AND pass='".$pass."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება!");
if($row=mysql_fetch_array($result1))
```

```
{
$start=სესია_start();
სესია_register("log");
$HTTP_სესია_VARS["log"]=$row["fam"]." ".$row["im"];
სესია_register("id");
$HTTP_სესია_VARS["id"]=$row["id_cust"];
$message="<tr><td bgcolor='#66cc66' align='center'>
<b>ავტორიზაცია წარმატებით შესრულდა</b></td></tr>";
$success=true;
}
else
{
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>ასეთი ლოგინი/ პაროლი არ არსებობს!!!</b></td></tr>";
}
mysql_close();

if($success)
{
include ("cabinet.phtml");
}
else
{
include("header.phtml");
print $message;
include("footer.phtml");
}
?>
```

პირადი კაბინეტი (cabinet.phtml)

```
<?
$title="პირადი კაბინეტი";
$color="#aaaaff";

$log=$HTTP_სესია_VARS["log"];
$id=$HTTP_სესია_VARS["id"];

if(!isset($log))
{
$success=false;
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>თქვენ არ ხართ ავტორიზებული!!!</b></td></tr>";
```

```
}
else
$success=true;
include("header.phtml");
print $message;

if($success)
{
include("connect.phtml");
$strSQL="SELECT * FROM customers WHERE id_cust='".$id."'";
$result=mysql_query($strSQL)
or die("შეუძლებელია მოთხოვნის შესრულება!");
if($row=mysql_fetch_array($result))
{
?>
<form action=change.phtml method=post>
<tr><td>
<h2>თქვენი პირადი მონაცემები</h2>
<table border="0" width="100%" align="right" >
<tr><td align="right"><i>გვარი: </i></td><td>
<input type=text name=fam value="<?print $row["fam"]?>"></td>
<td align="right"><i>სახელი: </i></td><td>
<input type=text name=im value="<?print $row["im"]?>"></td>
</tr>
<tr><td align="right"><i>მისამართი: </i></td><td>
<input type=text name=addr value="<?print $row["addr"]?>"></td>
<td align="right"><i>E-mail: </i></td><td>
<input type=text name=mail value="<?print $row["mail"]?>"></td>
</tr>
<tr><td align="right" colspan=3><i>
<input type="checkbox" value="1" name="subscribe"
<? if($row["subscribe"]==1) print "checked"; ?> >


<tr><td align="center" colspan="4">
<input type="submit" value="ცვლილებების შენახვა"></td></tr>
</table>
</form>
</td></tr>
<tr><td>
<h2>თქვენი შეკვეთა</h2>
<?
$strSQL1="SELECT id_order, date_ord FROM orders
```

```
WHERE id_cust='".$id."' ORDER BY date_ord DESC";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება1!");
while($row1=mysql_fetch_array($result1))
{
$order=$row1["id_order"];
$strSQL2="SELECT author, name_book, pages,
price, kolvo, id_order, books.id_book
FROM books,   order_books WHERE
books.id_book=order_books.id_book
and id_order='".$order."'";
$result2=mysql_query($strSQL2)
or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
?>
<tr><td>
<hr>
<b>შეკვეთა № <?=$order?> от <?=$row1["date_ord"]?><br></b>
<table border="1" width="100%" align="right" >
<tr><td align="right" width="20%"><i>ავტორი: </i></td>
<td align="right" width="50%"><i>დასახელება: </i></td>
<td align="right" width="15%"><i>ფასი: </i></td>
<td align="right" width="15%"><i>რაოდენობა: </i></td></tr>
<?
$sum=0;
while($row2=mysql_fetch_array($result2))
{
?>
<tr>
<td><?print $row2["author"];?></td>
<td><b><?print $row2["name_book"];?></b></td>
<td><?print $row2["price"];?></td>
<td><?print $row2["kolvo"];?></td>
</tr>
<?$sum=$sum+$row2["price"]*$row2["kolvo"];
}
$strSQL3="SELECT name_cat FROM categories, orders
WHERE categories.id_cat=orders.bonus AND
id_order='".$order."'";
$result3=mysql_query($strSQL3)
or die("შეუძლებელია მე-3 მოთხოვნის შესრულება!");
if($row3=mysql_fetch_array($result3))
{?>
```

```
<tr>
<td colspan=2>უფასო კატალოგი по теме <b>
<?print $row3["name_cat"];?></b></td>
<td>0</td>
<td>1</td>
</tr>
<?}
?>
<tr><td></td><td align="right"><i>ИТОГО: </i></td>
<td><?print $sum;?></td><td></td></tr>
</table>
</td></tr>

<?
}
}
mysql_close();
}

include("footer.phtml");
?>
```

პირადი მონაცემების ცვლილება (change.phtml)

```
<?
$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$id=$HTTP_სესია_VARS["id"];
$subscribe=$HTTP_POST_VARS["subscribe"];

$title="რეგისტრაცია";
$color="#aaaaff";

include("connect.phtml");

if($fam!="" && $im!="" && $addr!="" && $mail!="")
{
$strSQL1="UPDATE customers SET fam='".$fam."',
im='".$im."',addr='".$addr."', mail='".$mail."',
subscribe='".$subscribe."' WHERE id_cust=".$id;
$result1=mysql_query($strSQL1)
```

```
or die("შეუძლებელია მოთხოვნის შესრულება!");
$HTTP_სესია_VARS["log"]=$fam." ".$im;
// სეანსური ცვლადის მნიშვნელობები გასურლებულია
$message="<tr><td bgcolor='#66cc66' align='center'>
<b>მონაცემების ცვლილებები შეიცვალა</b></td></tr>";
}
else
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>ყველა ველი არ არის შევსებული!!!</b></td></tr>";
include("header.phtml");
print $message;
include("footer.phtml");
?>
```

გამოსვლა –ავტორიზაციის გაუქმება (exit.phtml)

```
<?
სესია_unregister("log");
სესია_unregister("id");
სესია_destroy();

include("index.phtml");
?>
```

შეკვეთის დათვალიერება (order.phtml)

```
<?
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

$title="თქვენი შეკვეთა";
$color="#ffaaff";
include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT COUNT(*) as count FROM basket_books
WHERE id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
$row=mysql_fetch_array($result1);
if($row["count"]==0)

{
?>
<tr><td bgcolor='#ff9999' align='center'>
```

```
<b>თქვენი კალათა ცარიელია!</b></td></tr>
<?
}
else
{
$strSQL1="SELECT image, author, name_book, pages, price, kolvo,
id_bask, books.id_book FROM books, basket_books
WHERE books.id_book=basket_books.id_book AND
id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");

?>
<tr><td>
<table border="1" width="100%" align="right" >
<tr><td align="right"><i>ავტორი: </i></td>
<td align="right"><i>დასახელება: </i></td>
<td align="right"><i>ფასი: </i></td>
<td align="right"><i>რაოდენობა: </i></td></tr>
<?
$sum=0;
while($row=mysql_fetch_array($result1))
{
?>
<tr>
<td><?print $row["author"];?></td>
<td><b><?print $row["name_book"];?></b></td>
<td><?print $row["price"];?></td>
<td><?print $row["kolvo"];?></td>
</tr>
<?
$sum=$sum+$row["price"]*$row["kolvo"];
}
?>
<tr><td></td><td align="right"><i>ИТОГО: </i></td>
<td><?print $sum;?></td><td></td></tr>
</table>
<tr><td><br><b>მიტანის ხერხი:</b>
<input type="radio" value=1 name="dostavka" checked>საქართველოს ფოსტა
<input type="radio" value=2 name="dostavka"> კურიერი
<input type="radio" value=3 name="dostavka"> მყიდველის მიერ გატანა
</td><tr>
```

```
<tr><td>უფასო კატალოგის გამოგზავნა თემის მიხედვით:
<select name="bonus">
<option value="0">
<? $strSQL1="SELECT * FROM categories";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მოთხოვნის შესრულება!");
while($row=mysql_fetch_array($result1))
{?>
<option value="<? print $row["id_cat"]?>">
<? print $row["name_cat"];
}
?>
</td><tr>
<tr><td><center><a href=doorder.phtml>
<b>შეკვეთის გაგზავნა</b></a></center></td></tr>
<?
}
include("footer.phtml");
?>
```

შეკვეთის გაგზავნა (doorder.phtml)

```
<?
$log=$HTTP_სესია_VARS["log"];
$id=$HTTP_სესია_VARS["id"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];
$dostavka=$HTTP_POST_VARS["dostavka"];
$bonus=$HTTP_POST_VARS["bonus"];

$title="თქვენი შეკვეთა";
$color="#ffaaff";

include("connect.phtml");

if(!isset($log))
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>თქვენ არ ხართ ავტორიზებული!!!</b></td></tr>";
else
{
$strSQL1="SELECT COUNT(*) as count FROM basket_books
WHERE id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
$row=mysql_fetch_array($result1);
```

```
if($row["count"]==0)
$message="<tr><td bgcolor='#ff9999' align='center'>
<b>თქვენი კალათა ცარიელია!</b></td></tr>";
else
{
// ახალი შეკვეთის შექმნა
$order=uniqid("OR");
$strSQL="INSERT INTO orders
(id_order, date_ord, id_cust, dostavka, bonus)
VALUES ('".$order."',CURDATE(),".$id.",
'".$dostavka."','".$bonus."')";
mysql_query($strSQL)
or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
// მყიდველის კალათიდან ყველაფრის წაკითხვა
$strSQL="SELECT * FROM basket_books
WHERE id_bask='".$id_bask."'";
$result=mysql_query($strSQL)
or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
while ($row=mysql_fetch_array($result))
{
//შეკვეთის შემადგენლობაში გადაწერა
$strSQL="INSERT INTO order_books (id_order, id_book,
kolvo) VALUES ('".$order."',".$row["id_book"].
",".$row["kolvo"].")";
mysql_query($strSQL)
or die("შეუძლებელია მე-3 მოთხოვნის შესრულება!");
}
// მყიდველის კალათის გასუფთავება
$strSQL="DELETE FROM basket_books
WHERE id_bask='".$id_bask."'";
mysql_query($strSQL)

or die("შეუძლებელია მე-4 მოთხოვნის შესრულება!");
$uniq_ID=uniqid("ID");
setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
$message="<tr><td bgcolor='#66cc66' align='center'>
<b>თქვენი შეკვეთა გაგზავნილია</b></td></tr>";
}
}
include("header.phtml");
print $message;
include("footer.phtml"); ?>
```

ფასების სია XML ფორმატში (price.phtml)

```
<?
header ("Content-type: text/xml");
print "<?xml version=\"1.0\" encoding=\"windows-1251\" ?>";
include("connect.phtml");
$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1) or die("შეუძლებელია 1-ლი მოთხოვნის შესრულება!");
print "<ფასების სია>";
while($row=mysql_fetch_array($result1))
{
print "<გამომცემელი
код='".$row["id_publ"]."'>".$row["name_publ"];
$strSQL2="SELECT id_book, author, name_book, pages, price,
name_cat FROM books, categories WHERE
books.id_cat=categories.id_cat AND
books.id_publ=".$row["id_publ"];
$result2=mysql_query($strSQL2) or die("შეუძლებელია მე-2 მოთხოვნის შესრულება!");
while($row2=mysql_fetch_array($result2))
{
print "<წიგნი ავტორი='".$row2["author"]."'
დასახელება='".$row2["name_book"]. "'>";
print "<გვერდი>".$row2["pages"]."</გვერდი>";
print "<ფასი>".$row2["price"]."</ფასი>";
print "<კატეგორია>".$row2["name_cat"]."</კატეგორია>";
print "</წიგნი>";
}
print "</გამომცემელი>";
}
print "</ფასების სია>";
mysql_close();
?>
```

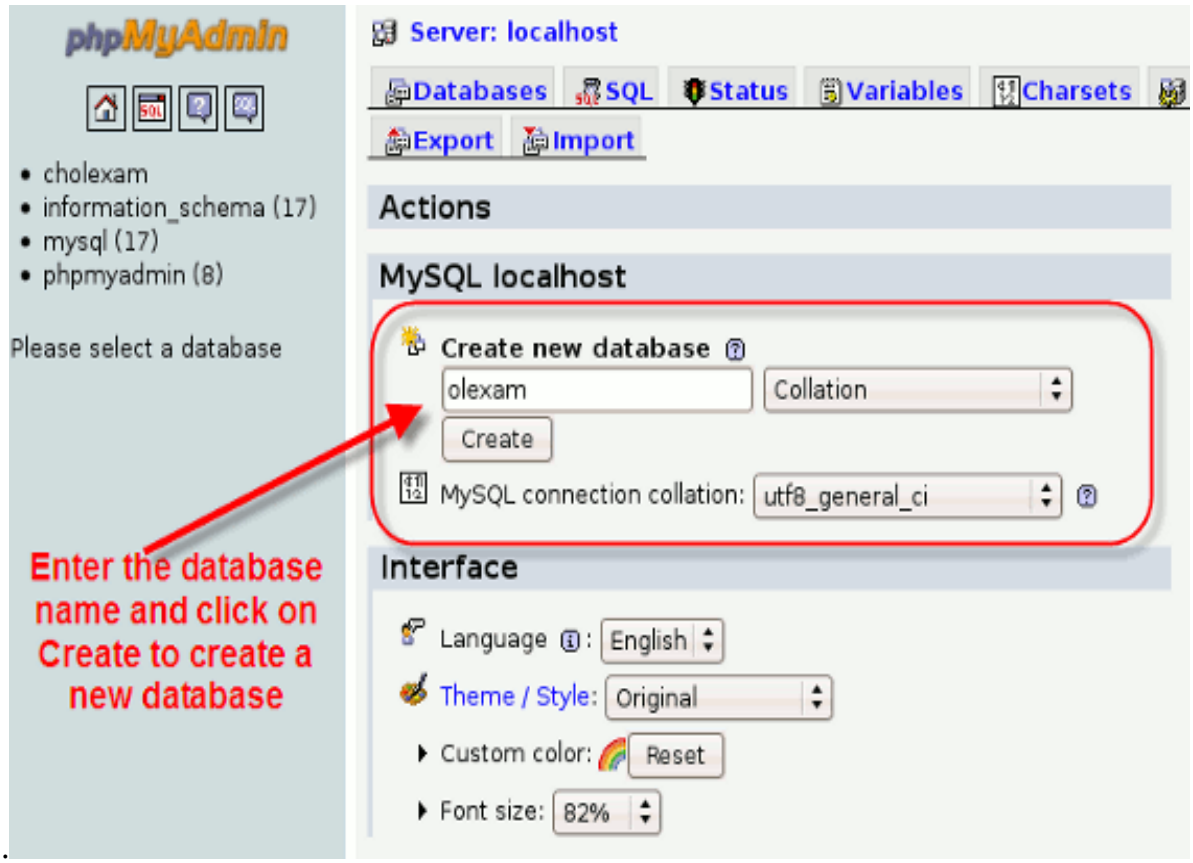
ლაბორატორიული სამუშაო № 15
მუშაობა PHPMYADMIN-თან

სამუშაოს მიზანი:

1. MySQL მონაცემთა ბაზის შექმნა PHP MyAdmin-ის გამოყენებით
2. PhpMyAdmin: მონაცემთა ბაზის მენეჯმენტი
3. PhpMyAdmin: MySQL მოთხოვნის შესრულება
4. PhpMyAdmin: ახალი მომხმარებლის დამატება

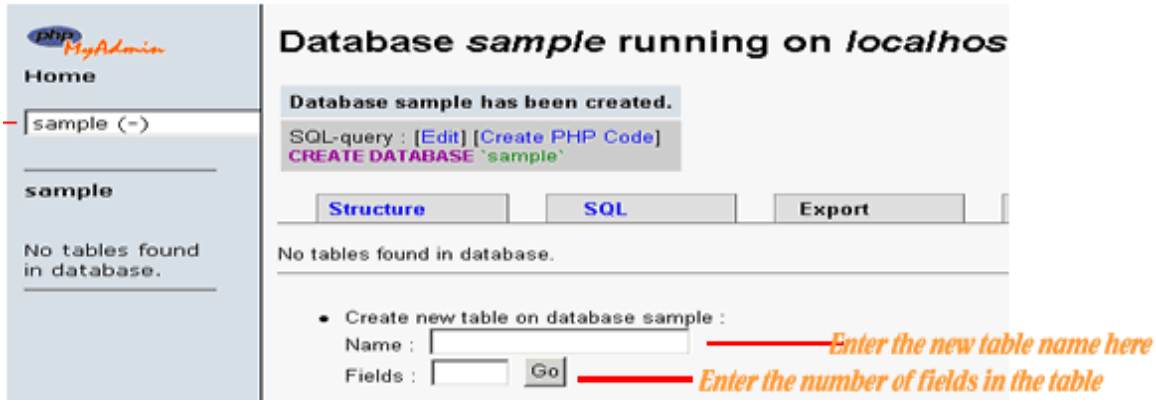
1.MySQL მონაცემთა ბაზის შექმნა PHP MyAdmin -ის გამოყენებით

phpMyAdmin -ში შესვლის შემდეგ ჩვენ შეგვიძლია ახალი მონაცემთა ბაზის შექმნა, რისთვისაც შეგვაქვს მონაცემთა ბაზის სახელი, შესაბამისად ვაყენებთ Character set -სა და Collation -ს და ვაწკაპუნებთ ღილაკზე Create.



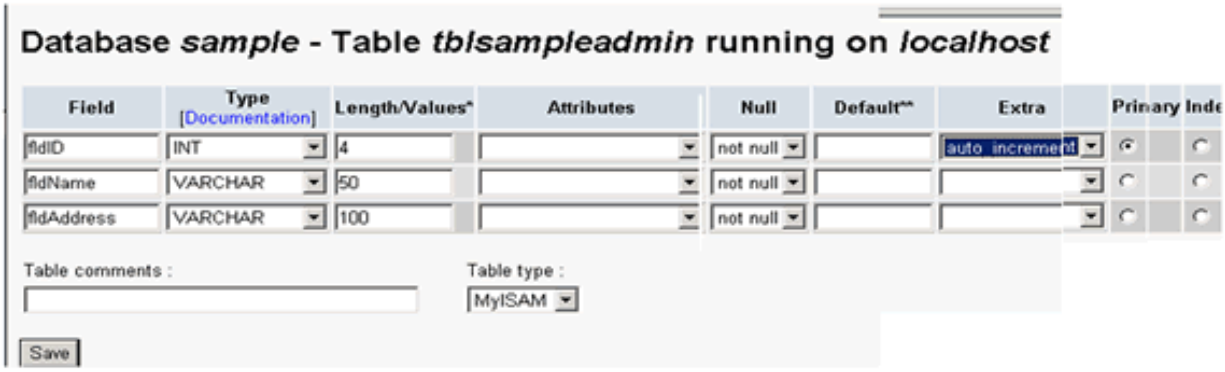
სურ. 15.1

შემდეგ ახალი ცხრილის შექმნისათვის შეგვაქვს სახელი და ველების რაოდენობა და ვაწკაპუნებთ ღილაკზე Go.



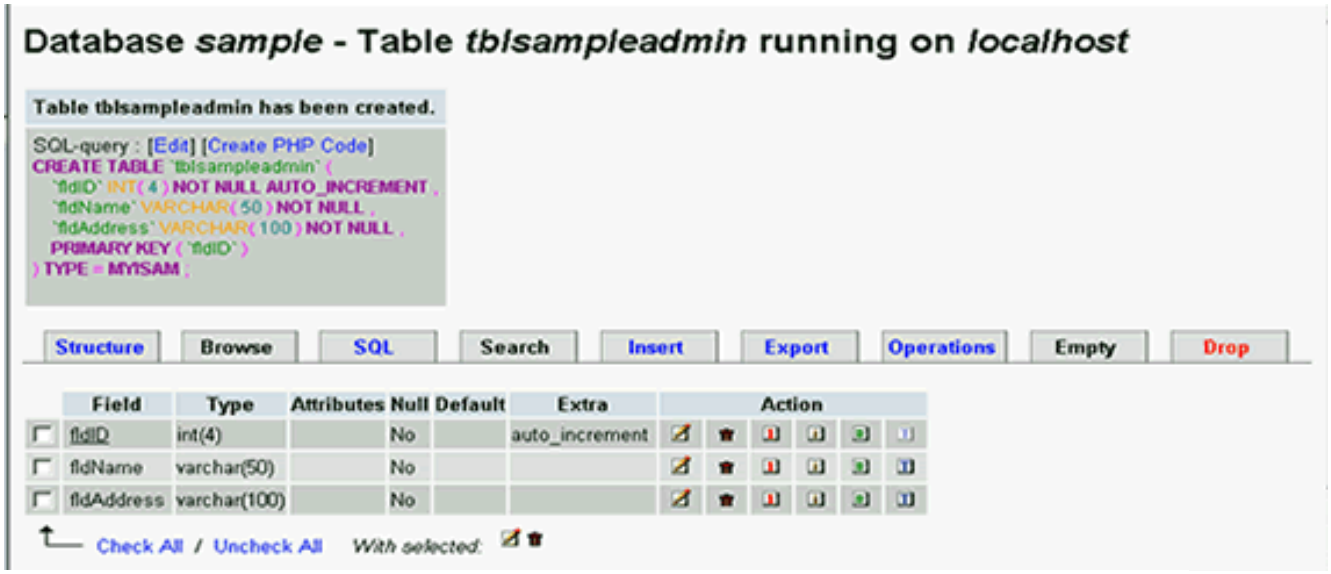
სურ. 15.2

შემდეგი ბიჯი უკვე ველების შექმნა იქნება. თითოეული ველისათვის შეგვაქვს ველის სახელი, ტიპი, სიგრძე და სხვა ოპციები. ცხრილის შექმნის დასრულებისათვის ვაწკაპუნებთ ღილაკზე Save.



სურ. 15.3

ცხრილის წარმატებით შექმნის შემთხვევაში ჩნდება შემდეგი გამოსახულება:

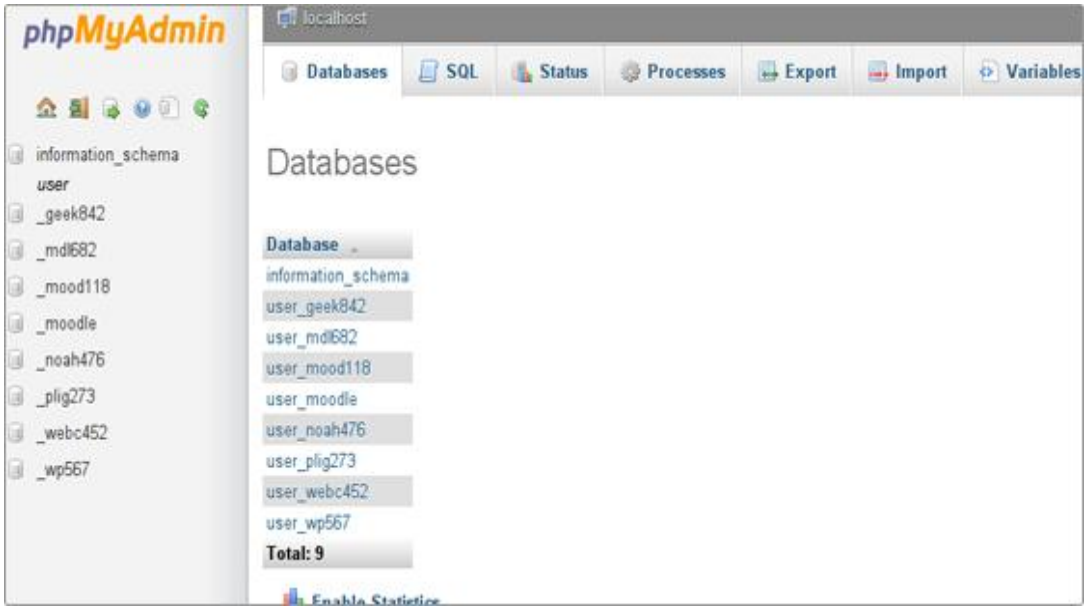


სურ. 15.4

ახლა უკვე Insert, Edit და Delete ოპერაციების შესასრულებლად ვაჭერთ შესაბამის ღილაკებს.

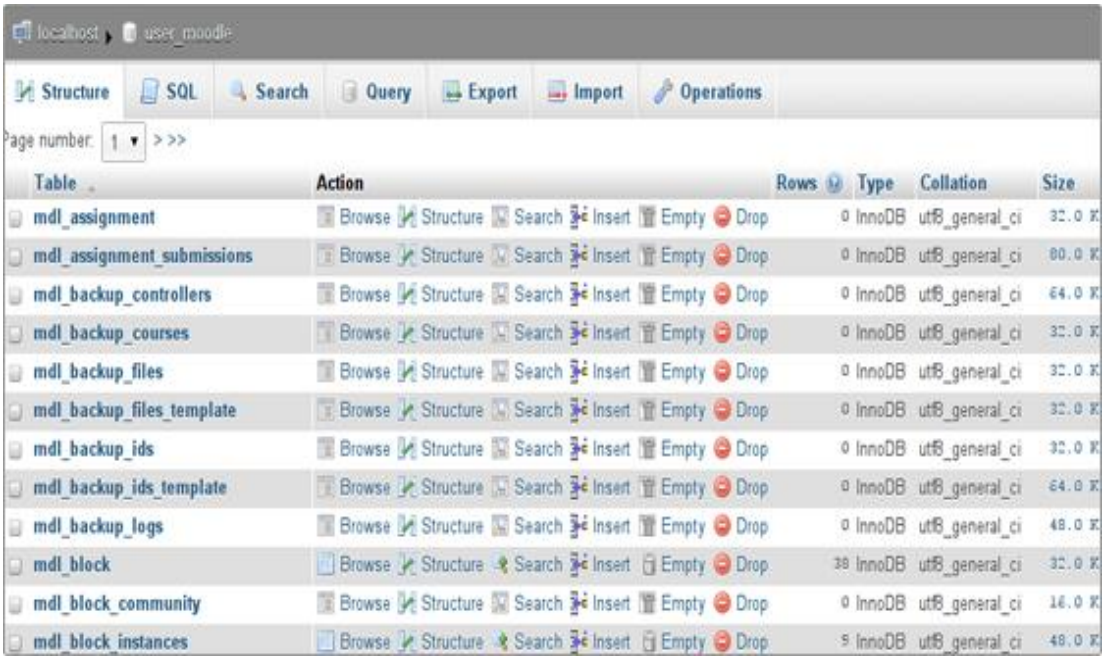
2. PhpMyAdmin: მონაცემთა ბაზის მენეჯმენტი

PhpMyAdmin ინსტრუმენტის მთავარი დანიშნულება არის ჩვენი მონაცემთა ბაზის მენეჯმენტი. ვაწკაპუნებთ Databases ჩანართზე. ვირჩევთ მონაცემთა ბაზას, რომლის მართვა გვინდა და ვაწკაპუნებთ მასზე.



სურ. 15.5

მომდევნო ეკრანზე დავინახავთ ამ მონაცემთა ბაზის ცხრილების სიას ნებართვების, ჩანაწერთა რაოდენობის, storage engine -ის, collation -ის, ცხრილების ზომებისა და ოპციათა ჩანართების მითითებით ეკრანის ზედა ნაწილში.

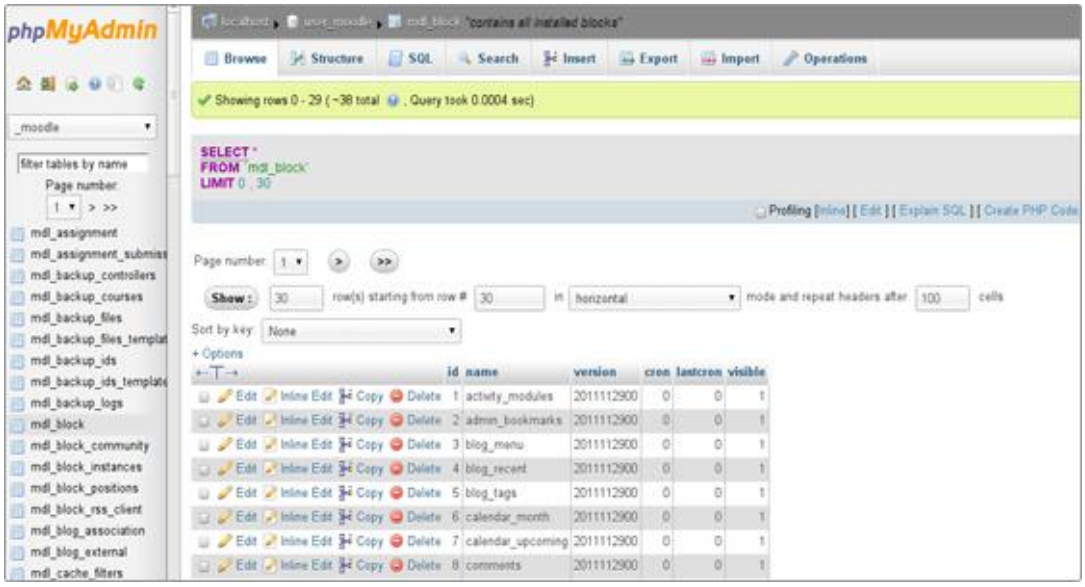


სურ. 15.6

განვიხილოთ შემდეგი ქმედებები:

Browse

ცხრილებში არსებული ჩანაწერების რედაქტირებისათვის ვაწკაპუნებთ იკონაზე Browse. იხსნება ახალი ფანჯარა ჩანაწერებით.

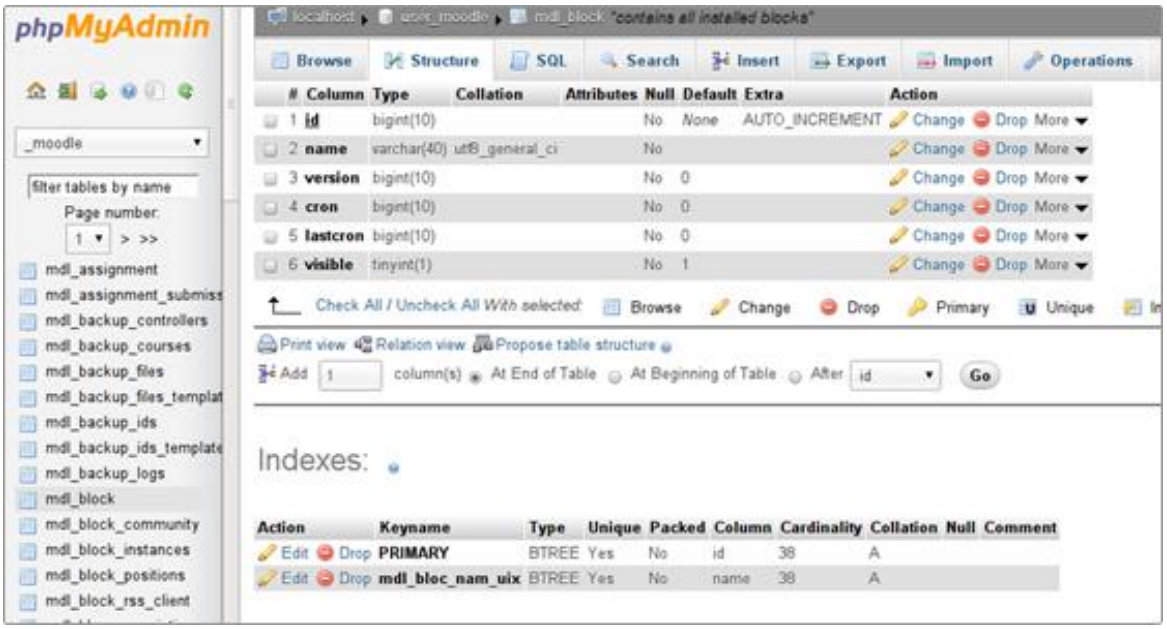


სურ. 15.7

იკონაზე Pen დაწკაპუნებით შეგვიძლია შერჩეული ჩანაწერის სტრუქტურისა და მნიშვნელობების რედაქტირება.

Structure

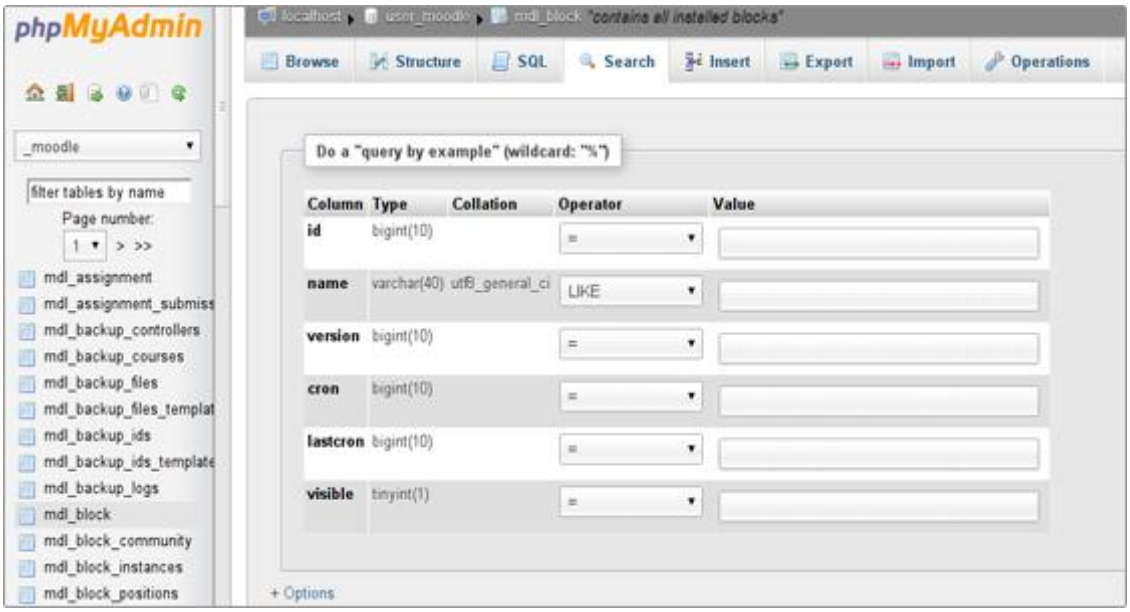
ცხრილის სტრუქტურის ნახვისათვის ვიყენებთ ოპციას Structure.



სურ. 15.8

Search

შერჩეულ ცხრილში ძებნის ქმედებათა განხორციელებისათვის ვიყენებთ Search ოპციას.

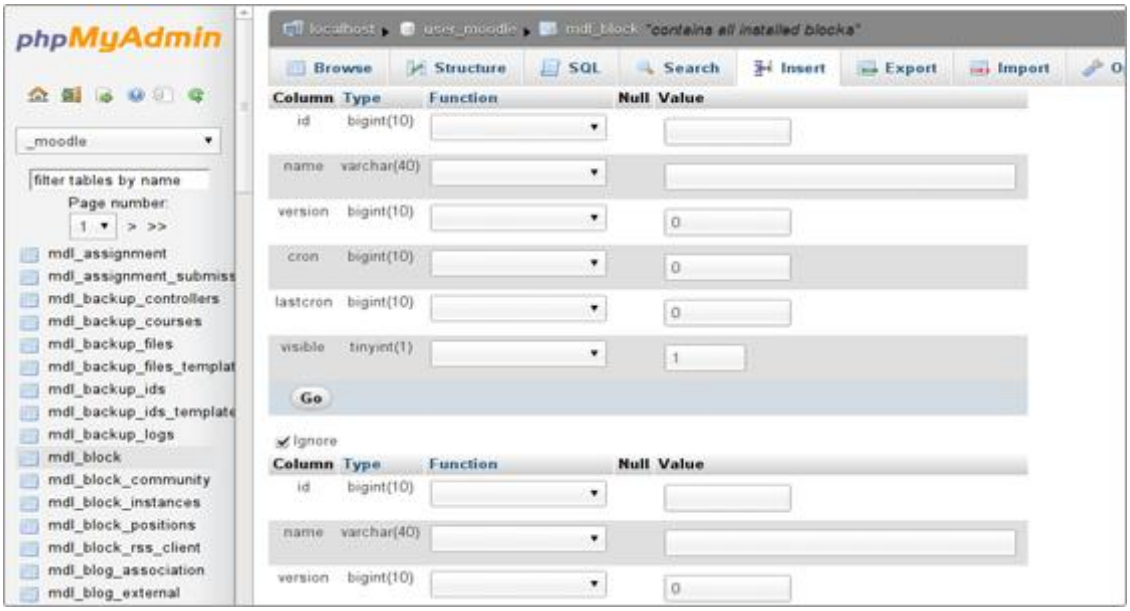


სურ. 15.9

ჩვენ შეგვიძლია აგრეთვე WHERE ბრძანების ან "query by example" გამოყენება. მოთხოვნის შესრულებისათვის ვაწკაპუნებთ ღილაკზე Go.

Insert

ჩანაწერების შეტანისათვის ვიყენებთ Insert -ს.

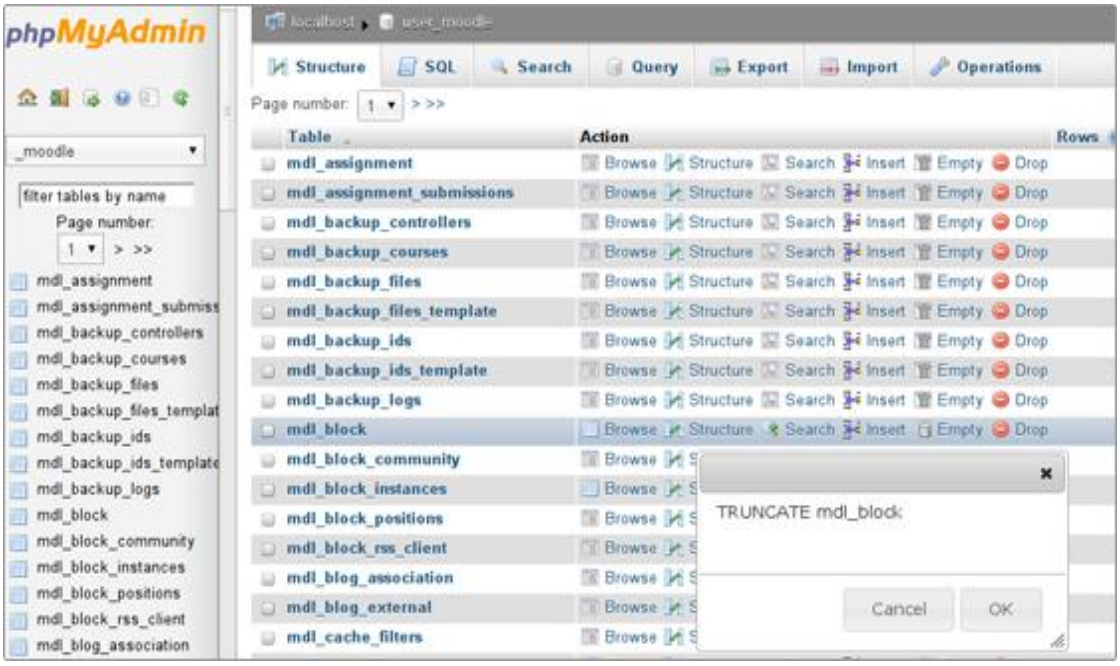


სურ. 15.10

შეგვაქვს მნიშვნელობები და ვაწკაპუნებთ ღილაკზე Go.

Empty

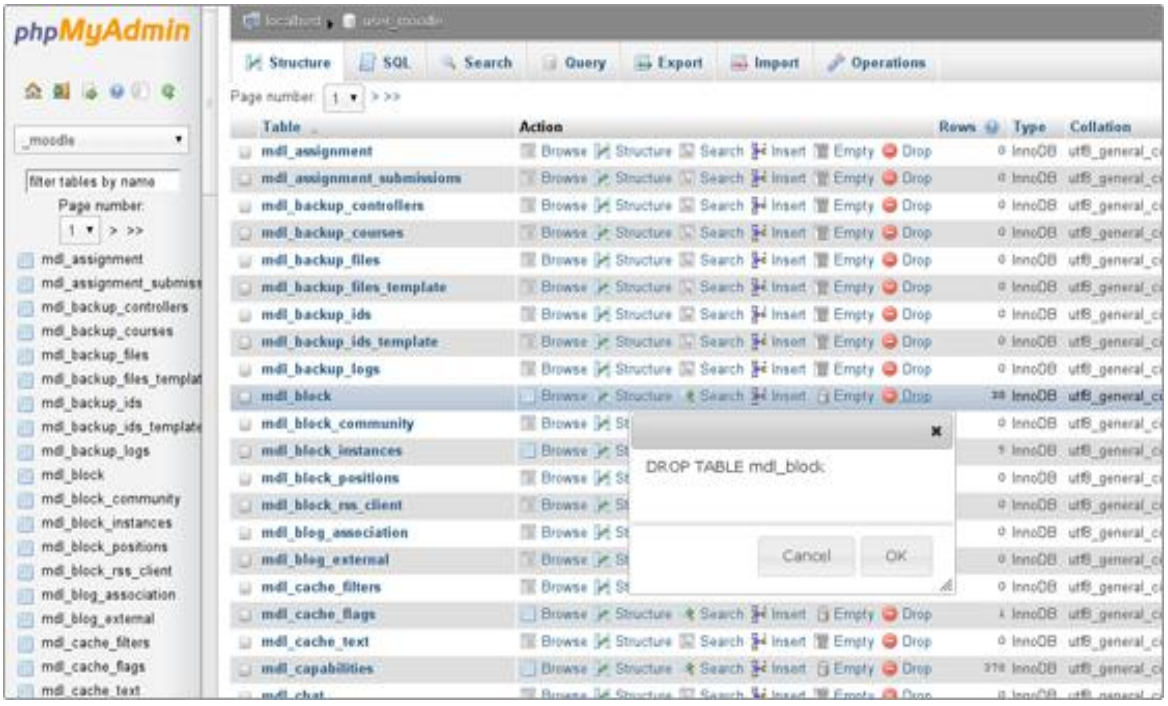
თუ გვინდა ცხრილის დაცარიელება მნიშვნელობებისაგან, მონაცემთა წაშლა და ცარიელი ცხრილის შენახვა ვიყენებთ Empty -ს.



სურ. 15.11

Drop

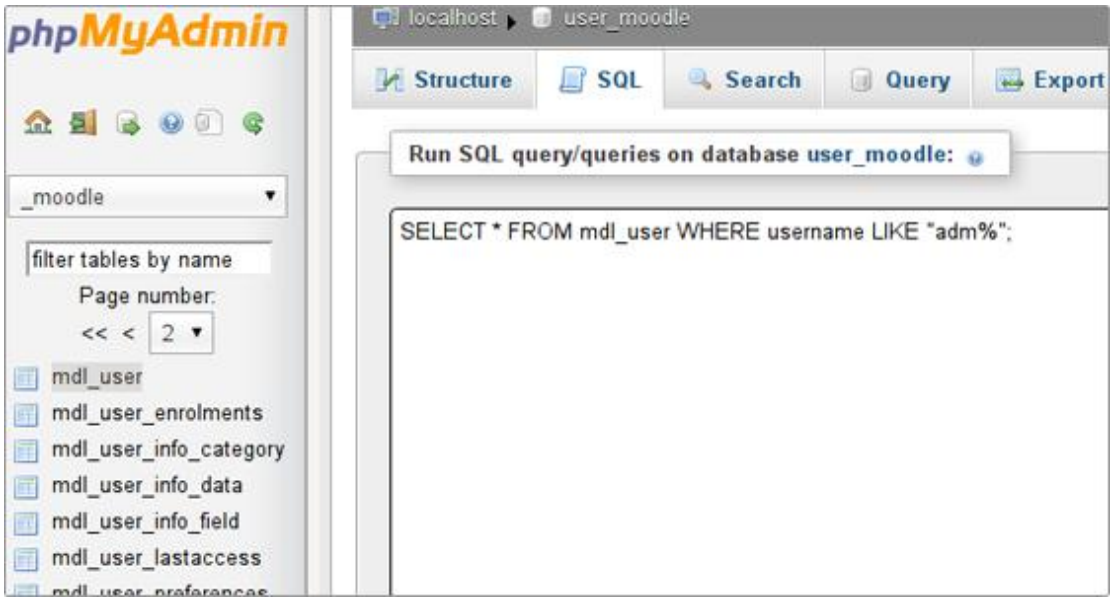
ცხრილის მთლიანად წაშლისათვის ყველა ჩანაწერთან ერთად ვიყენებთ Drop ბრძანებას. .



სურ. 15.12

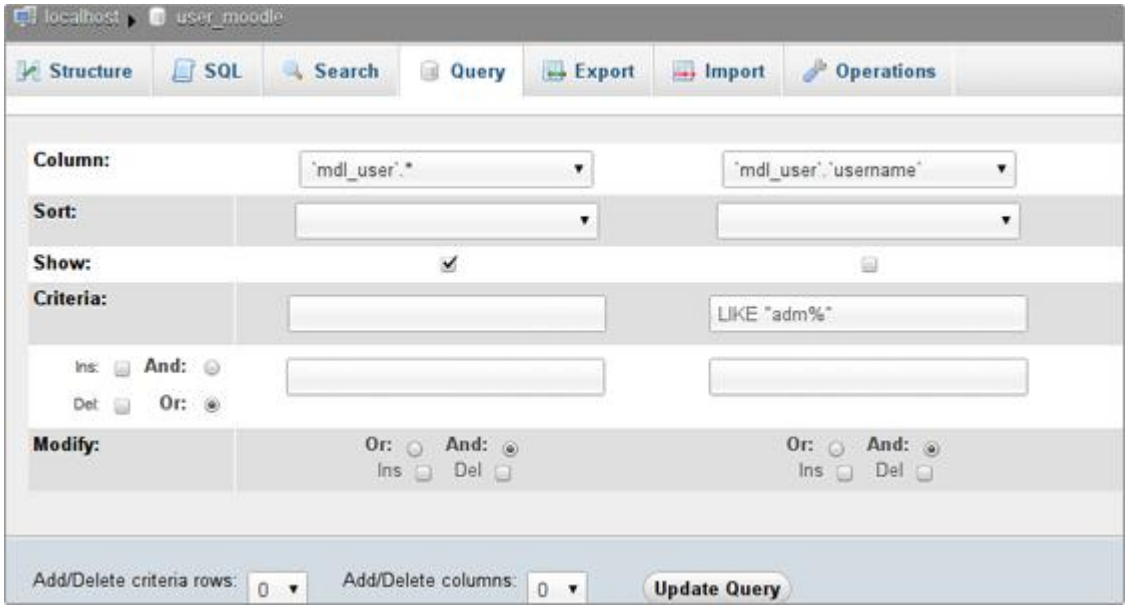
3.PhpMyAdmin: MySQL მოთხოვნის შესრულება

მოთხოვნის ფორმირება ხდება SQLჩანართში, სადაც შეგვაქვს მოთხოვნის კოდი და მისი შესრულებისათვის ვაწკაპუნებთ ღილაკზე Go.



სურ. 15.13

MySQL მოთხოვნის სხვა ხერხი არსებობს, როდესაც მისი აგება ხდება Query ჩანართში.

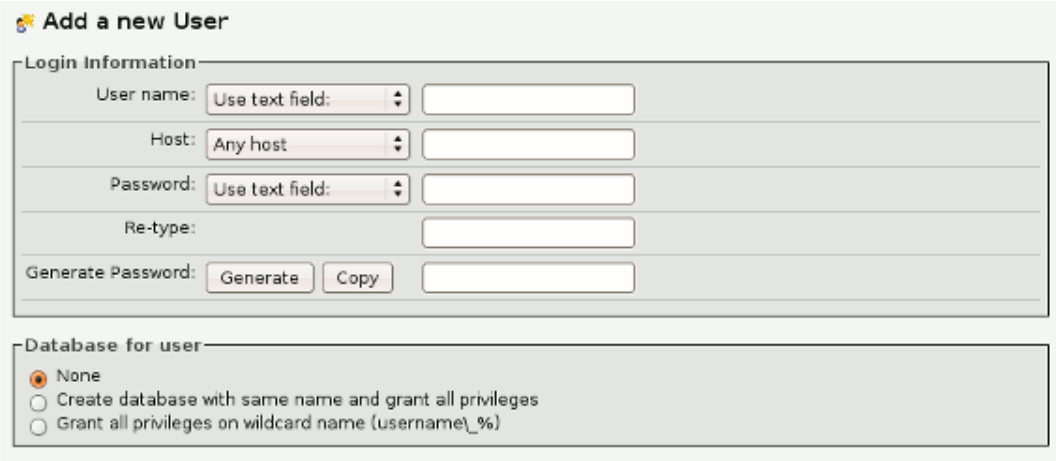


სურ. 15.14

Modify განყოფილებაში განვსაზღვრავთ ველებს შორის დამოკიდებულებას (სადაც ისინი შეერთებული არიან AND ან OR ლოგიკური ოპერატორების მეშვეობით). მოდიფიკაციის დასრულებისათვის ვაწკაპუნებთ ღილაკზე Update Query, ხოლო მოთხოვნის შესრულებისათვის ვაწკაპუნებთ ღილაკზე Submit Query.

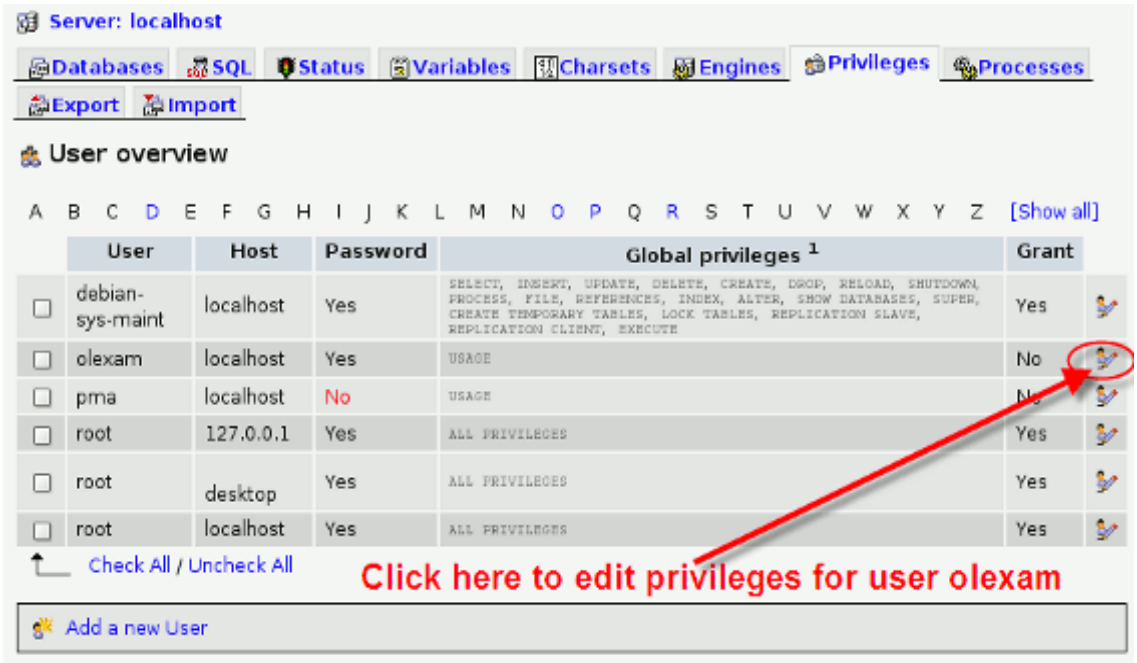
4. PhpMyAdmin: ახალი მომხმარებლის დამატება

ახალი მომხმარებლის დამატებისათვის ვაწკაპუნებთ Priviledges -ზე, რომლის შემდეგ ჩნდება Priviledges გვერდი. ვაწკაპუნებთ Add a new User -ზე. გამოდის ახალი მომხმარებლის დამატების ფორმა.



სურ. 15.15

კონკრეტული მომხმარებლისათვის მონაცემთა ბაზისადმი წვდომის განხორციელება შესაძლებელია ცალკეული მონაცემთა ბაზისათვის სხვადასხვა მომხმარებლის სახელისა და პაროლის შექმნით. ამასთან, ყოველი მომხმარებლისათვის გრანტი გაიცემა პრივილეგიების მინიჭების შესაბამისად.

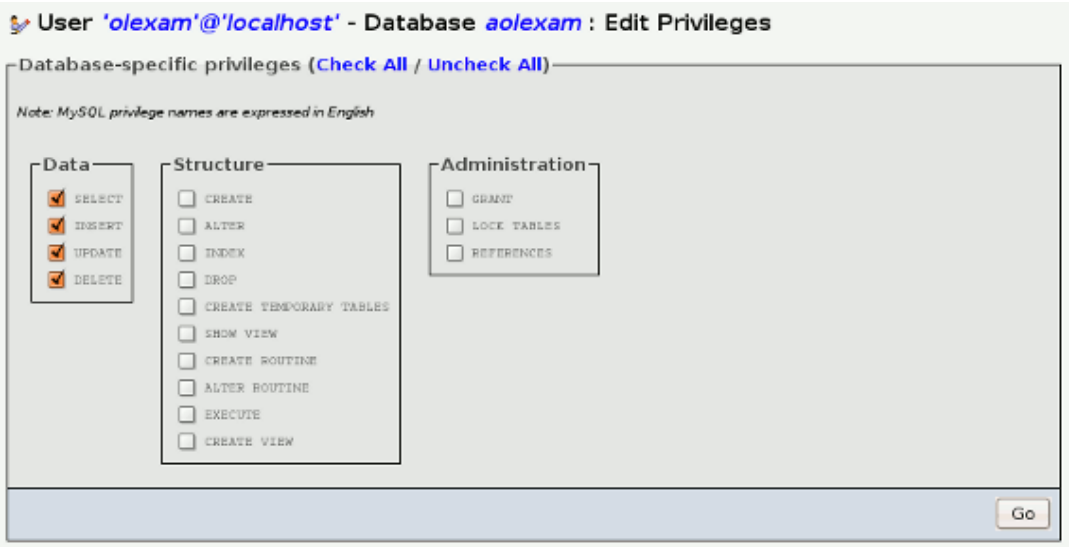


სურ. 15.16

პრივილეგიის დამატება ხდება Database-specific privileges სექციაში.

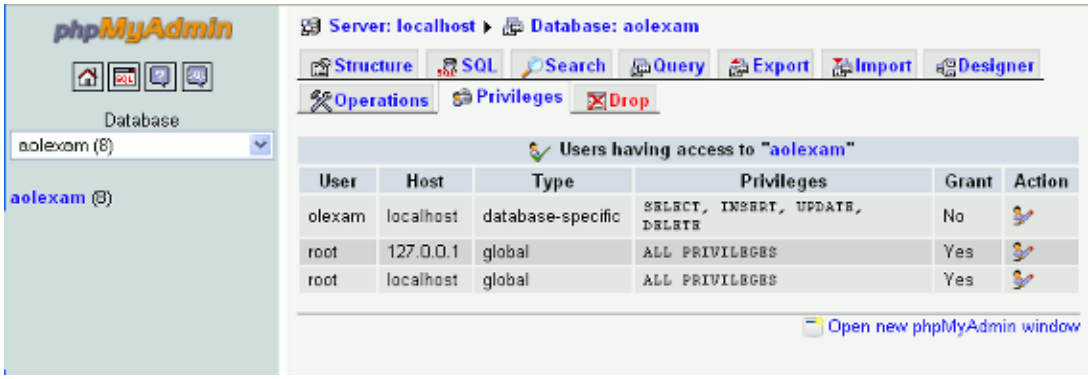
სურ. 15.17

შემდეგ გვერდზე გამოდის მომხმარებლის სახელი, მონაცემთა ბაზის დასახელება და Edit Privileges. გადავაადგილდებით Database-specific privileges სექციაში, სადაც ვაყენებთ ალამებს შესაბამის პრივილეგიებზე: SELECT, INSERT, UPDATE, DELETE და ვაწკაპუნებთ ღილაკზე GO.



სურ. 15.18

შემდეგ გვერდზე ჩნდება სერვერი, მონაცემთა ბაზა და მომხმარებლების სია თავიანთი პრივილეგიებით.



სურ. 15.19

ლიტერატურა

1. მეფარიშვილი ბ. `მონაცემთა ბაზების მართვის სისტემები'', სახელმძღვანელო, სტუ, 2009. 681.3.016(02)/13.
2. მეფარიშვილი ბ. `მონაცემთა ბაზების ადმინისტრირება'', სახელმძღვანელო, სტუ, 2009. 681.3.016(02)/10.
3. მეფარიშვილი ბ. `მონაცემთა ბაზების მართვის სისტემები'', მეთოდური მითითებანი ლაბორატორიული და პრაქტიკული მეცადინეობებისათვის, სტუ, 2008. 681.3.016(02)/17.
4. რომან სამხარაძე SQL სერვერი © საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი||,2008.
   681.3.016(02)/16. ISBN 978-9941-14-190-4. http://www.gtu.ge/publishinghouse/
5. სურგულაძე გ., შონია ო., ყვავაძე ლ. მონაცემთა ბაზების მართვის სისტემები (Ms SQL Server). დამხმ. სახელმძღ., სტუ, 2004.
   ბიბლ.: 681.3.06 (02).38; http://www.gtu.ge/katedrebi/kat94/pdf/sql.pdf