ივ. ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი
ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

ლაშა მარანელი

ვებ-ზე დაფუძნებული მონაცემთა ბაზის ადმინისტრირების სისტემა

სამაგისტრო პროგრამა: ინფორმაციული ტექნოლოგიები

სამაგისტრო ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მანანა ხაჩიძე

თბილისი, 2013

ანოტაცია

ნაშრომში განხილულია ვებ ტექნოლოგიების და მონაცემთა ბაზების განვითარების ეტაპები და ამ ეტაპებთან დაკავშირებული ძირითადი პრობლემები. განხილულია თანამედროვე ეტაპზე მონაცემთა ბაზების ადმინისტრირების ძირითადი საკითხები, რომელთაგან უმრავლესი სცილდება მხოლოდ მონაცემთა ბაზების კვლევის სივრცეს. ნაშრომში წარმოდგენილი კვლევის ერთერთი ძირითადი მიმართულებაა ვებ ორიენტირებული მონაცემთა ბაზები და მის ადმინისტრირებასთან დაკავშირებული საკითხები. აღწერილია ამ ტიპის მონაცემთა ბაზის ადმინისტრირების ძირითადი ეტაპები, მათი გადაჭრის მეთოდები და ხერხები. ამ მეთოდებისა და ხერხების განზოგადების საფუძველზე შექმნილია მომხმარებლის მხარდამჭერი სისტემა, რომელიც დახმარებას გაუწევს არაპროფესიონალ მომხმარებელს მოახდინოს საკუთარი მონაცემთა ბაზის ადმინისტრირება ვებ წვდომის ყველა ინსტრუმენტის გამოყენებით.

annotation

The paper deals with web technologies and database development stages phases and main problems of these stages. The main issues discussed in the current stage of database administration, most of which are beyond the scope of the database research area. One of the main directions of the research presented in this paper Web-oriented databases and issues related to its administration. Describes the main stages of this type of database administration, methods and ways of solving them. User support system based on the generalization of these methods and techniques, that will help the non-professional users to access and manage all their web and database administration tools.

სარჩევი

შესავალი

მეოცე საუკუნის მეორე ნახევარში ტექნოლოგიების განვითარებასთან ერთად, ადამიანს გაუჩნდა ამბიცია რთული ამოცანების გადაჭრის ავტომატიზაციისა. რადგან ეს შეუძლებელი იყო ჩვეულებრივი პერსონალური კომპიუტერისთვის, მოხდა ინტერნეტის განვითარება. ინტერნეტის განვითარებამ კი საფუძველი ჩაუყარა ვებ ტექნოლოგიებს, რომლის მეშვეობითაც მომხმარებელს საშუალება ქონდა გადაეცა თავისი ამოცანა მოშორებული, უფრო ძლიერი კომპიუტერისთვის (სერვერი), რომელიც ამოცანას ამუშავებდა და პასუხი კლიენტს ეგზავნებოდა.

მეოცე საუკუნის მიწურულს, საინფორმაციო ტექნოლოგიების ხელმისაწვდომობამ კიდევ უფრო გაამძაფრა საინფორმაციო სისტემების როლი ადამიანის ყოველდღიურ ცხოვრებაში. საინფორმაციო სისტემების პოტენციალმა ჩაენაცვლებინა ადამიანის რუტინული შრომა, შეემცირებინა შეცდომის დაშვების ალბათობა და გაეზარდა შესრულებული სამუშაოს ხარისხი და წარმადობა, გამოიწვია მათი გაჩენა ყველგან, ადამიანის მოღვაწეობის ყველა სფეროში, ბიზნესში, განათლებასა თუ მედიცინაში.

პირველ თავში განხილულია ვებ ტექნოლოგიების განვითარების ეტაპები და მისი როლი ადამიანის საქმიანობასა და ყოფა-ცხოვრებაში.

მეორე თავში განხილულია ~~ცნება~~ მონაცემთა რელაციური ბაზები, მათი განვითარების ეტაპები, თანამედროვე მდგომარეობა - განსსაკუტრებული ყურადღება ექცევა ვებ ორიენტირებულ მონაცემტა ბაზებს მათ შექმნას, შექმნასთან დაკავშირებული საკითხები, მათი ადმინისტრირება და ადმინისტრირების საჭიროება.

მესამე თავში აღწერილია მონაცემთა ბაზის ადმინისტრირების ინსტრუმენტის შექმნის ძირითადი პრინციპები და ის ძირითადი ეტაპები და განხორციელებული სამუშაოები რომლის შედეგადაც შემუშავდა ვებ-ზე დაფუძნებული მონაცემთა ბაზის ადმინისტრირების სისტემა.

1. ვებ ტექნოლოგიები

1970 წლის მაისში ჟურნალში „პოპულარული მეცნიერება“, Artur C. Clarke-მა იწინასწარმეტყველა რომ როდესმე სატელიტები მოიტანენ დაგროვილ ცოდნას თქვენს თითის წვერებთან („bring the accumulated knowledge of the world to your fingertips“) კონსოლის გამოყენებით, რომელიც გააერთიანებს ასლ გადამღებს, ტელეფონს, ტელევიზიას და პატარა კომპიუტერს, რომელიც საშუალებას მოგვცემს გადავგზავნოთ მონაცემები და ვიდეო კონფერენცია მოვახდინოთ დედამიწის ნებისმიერ წერტილში.

1989 წლის მარტში, ტიმ ბერნერს-ლიმ დაწერა ENQUIRE, მონაცემთა ბაზის და პროგრამული უზრუნველყოფის პროექტი, რომელიც მან შექმნა 1980 წელს, და აღწერა გულმოდგინედ დამუშავებული ინფორმაცია მენეჯმენტის სისტემისა.

Robert Cailliau-ს დახმარებით, მან გამოაქვეყნა უფრო კარგად ჩამოყალიბებული წინადადება (1990 წლის 12 ნოემბერი) რათა შექმნილიყო “Hypertext Project” სახელად “WorldWideWeb” (W3) როგორც ქსელი ჰიპერტექსტული დოკუმენტებისა, რომელიც ხელმისაწვდომი იქნებოდა ბრაუზერის მეშვეობით, კლიენტ-სერვერული არქიტექტურის წყალობით. ეს დოკუმენტი იუწყებოდა, რომ მხოლოდ ნახვისთვის განკუთვნილი ვებ-ი შეიძლებოდა შექმნილიყო სამი თვის განმავლობაში და საჭირო იქნებოდა 6 თვე რომ მიღწეულიყო ახალი ბმულების და ახალი ინფორმაციისთვის, რომელიც განკუთვნილი იქნებოდა მკითხველებისთვის, დაცული იქნება საავტორო უფლებები. მისი თქმით, შესაძლებელი ხდება ასევე მკითხველის გაფრთხილება, როდესაც მისთვის საინტერესო ინფორმაცია ხელმისაწვდომი ხდება. რადგან ჯერ მხოლოდ მიზანი იყო მხოლოდ და მხოლოდ კითხვადი ინფორმაციის განთავსება, ვებ-ის შემდგომი თაობის იდეა, დინამიური ვებ გვერდები, უფრო გვიან მომწიფდა.

ბერნერს-ლიმ NeXT-ის კომპიუტერი გამოიყენა პირველ ვებ სერვერად და პირველი ვებ ბრაუზერის, WorldWideWeb-ის დასაწერად, 1990 წელს. ამავე წლის შობისთვის მას უკვე ქონდა ყველა საჭირო ხელსაწყო მუშა ვებ-ისთვის. პირველი ვებ ბრაუზერი (რომელიც ასევე იყო ედიტორი), პირველი ვებ სერვერი, პირველი ვებ გვერდები, რომელიც თვითონ აღწერდა პროექტს. ასევე მანვე ატვირთა პირველი სურათი 1992 წელს, რომელზეც აღბეჭდილი იყოს CERN-ის სახლის ჯგუფი.

პირველი სერვერი ევროპის გარეთ გაჩნდა სტენფორდის წრფივი აჩქარების ცენტრში (SLAC) პალო ალტო, კალიფორნია, რომელიც ემსახურებოდა SPIRES-HEP მონაცემთა ბაზას.

ბერნერს-ლი ითხოვდა რომ შეეერთებინათ ჰიპერტექსტი ინტერნეტისთვის, მაგრამ რადგან არავინ გამოეხმაურა მის მოთხოვნას მან თავის თავზე აიღო და შექმნა კიდევ სამი დამატებითი ტექნოლოგია:

1. გლობალურად უნიკალური იდენტიფიკატორების სისტემა, რომელიც საშუალებას იძლევა ვებ-ში ან სადმე მოიძებნოს დოკუმენტი (UDI (universal document identifier)), რომელსაც შემდეგ გადაერქვა სახელი Uniform Resource Locator (URL) და Uniform Resource Identifier (URI).
2. გამოქვეყნების ენა HyperText Markup Language (HTML).
3. HyperText Transfer Protocol (HTTP).

The World Wide Web-ს ქონდა უამრავი განსხვავება სხვა მაშინ არსებულ ჰიპერტექსტის სისტემებთან შედარებით. ყველაზე მნიშვნელოვანი იყო ის, რომ ის იძლეოდა საშუალებას შექმნილიყო სერვერები და კლიენტები დამოუკიდებლად, დამატებოდა მათ შესაძლებლობები, რაც არანაირ ლიცენზიას არ საჭიროებდა. ის იყო სრულიად უფასო, რამაც განაპირობა მისი სწრაფი პოპულარიზაცია.

ძირითადად თანხმდებიან იმაზე რომ ამ ტექნოლოგიის გარდატეხის წერტილი იყო Mosaic ვებ ბრაუზერის გამოშვება 1993 წელს. ის იყო გრაფიკული ბრაუზერი რომელიც შექმნა სუპერკომპიუტინგის აპლიკაციების ნაციონალურმა ცენტრმა ილინოისის უნივერსიტეტში. აქამდე გრაფიკა არასდროს ყოფილა ტექსტთან ერთად, გამოიყენებოდა ისევ ძველი პროტოკოლები. ამ ბრაუზერის გამოშვებამ Web გახადა ყველაზე პოპულარული ინტერნეტ პროტოკოლი.

არსებულ ინტერნეტთან დაკავშირებამ გამოიწვია სხვა ვებ საიტების შექმნა მსოფლიოს სხვადასხვა წერტილებში. თავის მხრივ მოხდა დომენური სახელების და HTML-ის საერთაშორისო სტანდარტების შემოღება. ყოველივე ამის შემდეგ დაარსდა W3C კონსორციუმი, რომელიც დღემდე კარნახობს ვებ-ის სტანდარტებს მსოფლიოს.

ინფორმაციის რაოდენობის ზრდასთან ერთად, საჭირო გახდა შექმნილიყო ისეთი ტექნოლოგია, რომელიც შეძლებდა დინამიური ინფორმაციის გენერირებას, სტატიკურ გვერდში, და შემდეგ ეს გვერდი გადაეცემოდა მომხმარებელს. შედეგად განვითარდა ახალი სტანდარტი Web 2.0 სახელწოდებით.

Web 2.0 აღწერს საიტებს, რომელიც იყენებს ტექნოლოგიას სტატიკურ გვერდებთან ერთად. ტერმინი ჩამოყალიბდა 1999 წელს, ხოლო შემდეგ პოპულარული გახდა 2004 წელს, ტიმ ო'რეილის მედიის საშუალებით. ტექნოლოგია გვთავაზობს World Wide Web-ის განახლებულ ვერსიას, ის არ არის ტექნიკური განახლება, მაგრამ ცვლის უამრავ ძირითად ასპექტს თუ როგორ არის დამზადებული და გამოყენებული ვებ გვერდი.

Web 2.0 ტექნოლოგიის საიტს საშუალება აქვს მოახდინოს მომხმარებელთან დიალოგი ვირტუალურად, იმ საიტებისგან განსხვავებით, სადაც მომხმარებელი მხოლოდ ათვალიერებს შიგთავსს. ამ ტექნოლოგიის მაგალითებია სოციალური ქსელის საიტები, ბლოგები, ვიდეო გაზიარება, სერვისები და ა.შ.

ეს ყველაფერი რომ შესაძლებელი იყოს, გამოიყენება რამდენიმე ტექნოლოგია, რომელთა შორისაა კლიენტის მხარის სკრიპტი, ანუ JavaScript, ასევე მისი ბიბლიოთეკები, მაგალითად, jQuery, რომელიც აღმოფხვრის ბრაუზერის მიერ წარმოქმნილ სკრიპტთან დაკავშირებულ შეუთავსებლობებს.

კლიენტის მხარის ტექნოლოგია (ბრაუზერი), ajax გამოიყენება web 2.0 პროექტებისთვის. Ajax ტექნოლოგია საშუალებას იძლევა ჯავასკრიპტის მეშვეობით აიტვირთოს და ჩამოიტვირთოს მონაცემები ვებ სერვერიდან მთლიანი გვერდის გადატვირთვის გარეშე.

მომხმარებელს რომ ქონდეს საშუალება გვერდთან ინტერაქტივისთვის, კომუნიკაცია, როგორიცაა მოთხოვნების დამუშავება ხდება სერვერზე განცალკევებულად იმ მონაცემებისგან, რომელიც უნდა დაბრუნდეს (ასინქრონულად). სხვა შემთხვევაში მომხმარებელი იძულებული იქნებოდა დალოდებოდა მონაცემებს, ისევე როგორც გვერდის გადატვირთვის შემთხვევაში. ეს ასევე ზრდის საიტების სისწრაფეს, რადგან მოთხოვნის გაგზავნა და პასუხის მიღება შეიძლება უფრო სწრაფად დასრულდეს.

მონაცემები რომელიც ბრუნდება აჯაქს მოთხოვნის საფუძველზე დაფორმატებულია XML ან JSON(JavaScript Object Notation) ფორმატში, ორი ყველაზე გავრცელებული

სტრუქტურა. ორივე ფორმატი გასაგებია ჯავასკრიპტისთვის, ამიტომ პროგრამისტს მარტივად შეუძლია გამოიყენოს და გადასცეს სტრუქტურირებული მონაცემების მის აპლიკაციას.

სერვერის მხარეს web 2.0 იყენებს მსგავს ტექნოლოგიებს, რასაც web 1.0. ენები როგორიცაა PHP, Ruby, Perl, Python, JSP, ASP.NET გამოიყენება პროგრამისტების მიერ რომ გამოიტანონ მონაცემები დინამიურად ფაილებიდან და მონაცემთა ბაზებიდან. ვებ 2.0 ტექნოლოგიაში შეიცვალა მონაცემების გამოტანის ფორმატი. ინტერნეტის განვითარების საწყისში არ იყო საჭირო სხვადასხვა ვებ საიტები დაკავშირებოდნენ ერთმანეთს და გაეზიარებინათ მონაცემები. ახლა კი ეს გახდა მთავარი. მონაცემების სხვადასხვა საიტებთან გასაზიარებლად ვებ საიტს საშუალება უნდა ქონდეს გამოიტანოს მონაცემები მანქანის წაკითხვად ფორმატში როგორიცაა XML და JSON. როდესაც საიტს აქვს ამის საშუალება, მეორეს შეუძლია გამოიყენოს მისი ნაწილი და ჩაირთოს ფუნქციონალი.

Web 2.0 აღიწერება სამ ნაწილად

- RIA - მდიდარი ინტერნეტ აპლიკაციები, განსაზღვრავს შესაძლებლობებს რომელიც გადავიდა ბრაუზერში დესკტოპ აპლიკაციებიდან, იქნება ეს გრაფიკული თუ გამოყენებითი თვალსაზრისით. ზოგიერთი ტერმინია Ajax და Flash.
- WOA - ვებ ორიენტირებული არქიტექტურა, ძირითადი ნაწილია Web 2.0-სა, რომელიც განსაზღვრავს რომ იყოს ამ სტანდარტის აპლიკაციების ფუნქციები გამოხატული ისე, რომ სხვა აპლიკაციებს შეეძლოთ მათი გამოყენება.
- სოციალური ვები - განსაზღვრავს თუ როგორ უკავშირდებიან მომხმარებლები ერთმანეთს web 2.0 -ს გამოყენებით.

აქედან გამომდინარე ეს ტექნოლოგია თავს უყრის კლიენტს და სერვერის პროგრამულ უზრუნველყოფას, შინაარსებს და ქსელის პროტოკოლებს. სტანდარტებზე ორიენტირებული ვებ ბრაუზერები იყენებენ სხვადასხვა ჩანართებს, რათა სწორად გამოიტანონ მონაცემები და მომხმარებელთან დიალოგი.

ეს ყველაფერი ზრდის ინფორმაციას, რომელიც საჭიროა ჩამოიტვირთოს, რათა მომხმარებელმა შეძლოს მისი ნახვა. საჭიროა ინფორმაცია იყოს მოწესრიგებული და ძებნადი, ერთმანეთთან სწორად იყოს დაკავშირებული. ამის საშუალებას კი გვაძლევს მონაცემთა ბაზა, რომელსაც საშუალება აქვს შეინახოს უამრავი ინფორმაცია და გახადოს ის წვდომადი ძალიან სწრაფად, მოთხოვნისთანავე.

მოწესრიგებული ინფორმაცია ბევრად უფრო ადვილი დასამუშავებელია, ვიდრე მოუწესრიგებელი, ეს ყველაფერი კი საბოლოოდ ზრდის დინამიური ვებ გვერდის გენერირების სისწრაფეს, მაგრამ ამ ინფორმაციის ნახვისთვის საჭიროა ასევე ჩამოიტვირთოს მომხმარებელთან, რაც ინტერნეტის სისწრაფესთან არის დაკავშირებული, რომელიც არ გამოირჩევა მაღალი ხარისხით და მდგრადობით. ეს ყველაფერი უფრო კომფორტული რომ იყოს და მომხმარებელს არ მოუწიოს დიდხანს ლოდინი, გამოიყენება ინფორმაციის ნაკადები. მაგ. ავიღოთ ვიდეოს ნახვა ბრაუზერში, რომელიც პირდაპირ სერვერიდან უნდა ჩამოიტვირთოს და არის დიდი ზომის ინფორმაცია, რომელიც სტატიკური ვებ გვერდივით სწრაფად ვერ ჩამოიტვირთება, არც მისი შეკუმშვა ხდება, ხარისხის დაკარგვის გარეშე. ამ შემთხვევაში ტექნოლოგია გულისხმობს Adobe Flex -ის გამოყენებას, ცნობილია როგორც ფლეში. იგი საშუალებას იძლევა მოხდეს ინფორმაციის ნაკადად ჩამოტვირთვა და პარალელურად ყურება. ასეთი საიტის მაგალითია www.youtube.com, რომელიც გამოირჩევა ვიდეო ინფორმაციის სიმრავლით.

2. მონაცემთა ბაზები - თანამედროვე პრაქტიკული მოდელები

2.1 მონაცემთა ბაზების განვითარება და რეალობა

პროცესორების, მეხსიერების, მახსოვრობისა და ქსელების განვითარებასთან ერთად გაიზარდა მონაცემთა ბაზების სისწრაფეც და მათი შესაძლებლობებიც.

მონაცემთა ბაზების განვითარების ტექნოლოგია შეიძლება დაიყოს სამ ერად დაფუძნებული მონაცემთა მოდელზე ან სტრუქტურაზე: navigational, sql/relational და post-relational. ორი ძირითადი ადრეული ნავიგაციური მოდელი იყო იერარქიული მოდელი, რომელიც შექმნა IBM-მა, მეორე კი Codasyl მოდელი (ქსელური მოდელი).

რელაციური მოდელი პირველად გამოქვეყნდა 1970 წელს კოდის მიერ, რომელიც იუწყებოდა რომ აპლიკაციები უნდა ეძებდნენ მონაცემებს შინაარსის მიხედვით და არა ბმულებზე მიყოლით. რელაციური მოდელი იქმნება ორ განზომილებიანი ცხრილის მეშვეობით, თითოეული ცხრილი ასახავს ერთ არსს. ის არ შექმნილა 1980 წლამდე, ვიდრე არ განვითარდა კომპიუტერის სიმძლავრე იმდენად, რომ შესაძლებელი ყოფილიყო ასეთი ბაზების მუშაობა. 1990 წლის პირველივე რიცხვებიდან რელაციური სისტემა გახდა დომინანტი ყველა დიდ მონაცემთა დამუშავების აპლიკაციაში და დღემდე ასეა, ზოგიერთი სფეროს გარდა. ამ მონაცემთა ბაზის ენაა SQL სტანდარტი, რომელიც დიდ გავლენას ახდენს სხვა მონაცემთა მოდელებზე.

ობიექტური მონაცემთა ბაზის გამოგონება მოხდა 1980 წელს, რათა გვერდი აევლოთ იბიექტურ-რელაციური წინაღობების აცდენას, რომელიც შევიდა “post-relational” ტერმინში, ამავე დროს იქმნებოდა ობიექტურ-რელაციური მონაცემთა ბაზები.

პოსტ-რელაციური მონაცემთა ბაზების შემდეგი თაობა წამოვიდა 2000-იან წლებში რომელიც გახდა ცნობილი NoSQL სახელით, რომელიც საშუალებას იძლეოდა სწრაფი გასაღები-მნიშვნელობა რეალიზაციით და დოკუმენტზე-ორიენტირებული მონაცემთა ბაზებით. „შემდეგი თაობა“ ცნობილია NewSQL მონაცემთა ბაზების სახელწოდებით, რომელთაც სცადეს შემოეტანათ ახალი საშუალებები რელაციური/SQL მოდელის, თანაც უმიზნებდნენ NoSQL ბაზის მაღალ სისწრაფეს რელაციურ მოდელთან შედარებით.

თანამედროვე მონაცემთა ბაზებს აქვთ ჩაშენებული რელაციურობის მხარდაჭერა.

2.2 რელაციური მონაცემთა ბაზები

რელაცია არის ცხრილი, რომელიც არის ორ განზომილებიანი. იგი შეიცავს სტრიქონებს და სვეტებს. რელაციაში ყოველი სტრიქონი უნდა იყოს უნიკალური. სტრიქონი შეესაბამება არსის ობიექტს, ხოლო სვეტი მის ატრიბუტს. არ აქვს მნიშვნელობა სტრიქონების თანმიმდევრობას. ყოველი სვეტი არის ერთი ტიპის, მაგრამ მნიშვნელობები შესაძლებელია იყოს განსხვავებული. არ აქვს მნიშვნელობა სვეტების თანმიმდევრობას. რელაციაში ერთ-ერთი მოთხოვნაა, რომ სტრიქონს შეესაბამებოდეს უნიკალური გასაღები. გასაღები შეიძლება იყოს ერთი ან რამდენიმე ატრიბუტისგან შემდგარი, მაგრამ ისე, რომ გარანტირებული იყოს მისი უნიკალურობა. ასეთი ცხრილის ერთ-ერთი მოთხოვნაა რომ არ უნდა იყო სვეტში შეერთებული ანუ მრავალმნიშვნელობიანი მნიშვნელობები.

რელაციური ცხრილის მაგალითია:

Teachers
teacherID
name
office
phone
email
.....

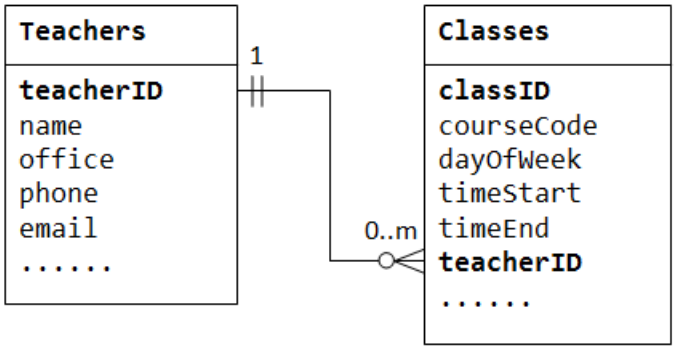
სადაც TeacherID არის ტექნიკური გასაღები. ტექნიკური გასაღები არის რიცხვების მიმდევრობა, დაწყებული 1-დან, ბიჯით ერთი. ასეთი გასაღების შემოღებით გარანტირებულია მისი უნიკალურობა. ტექნიკური გასაღები არის დამხმარე საშუალება, რომელიც გვეხმარება იმ შემთხვევაში, თუ არ არის გამოკვეთილი კანდიდატი გასაღები.

კანდიდატი გასაღები - არის ატრიბუტი, ან ატრიბუტთა ნაკრები, რომელსაც შეუძლია შეასრულოს ცხრილში გასაღების როლი. ასეთის არ არსებობის შემთხვევაში ვიყენებთ ტექნიკურ გასაღებს. თუ არის პირველადი გასაღებისთვის შესაბამისი ატრიბუტი, არ არის რეკომენდირებული ტექნიკური გასაღების გამოყენება.

რელაციური მონაცემთა ბაზის შემთხვევაში გვაქვს ერთმანეთთან დაკავშირებული ცხრილები, რომლებიც კავშირდება წარმომქმნელი ცხრილის პირველადი გასაღების

ჩამატებით შთამომავალ ცხრილში. შთამომავალ ცხრილში არსებულ მიმართებას მშობელ ცხრილზე გარე გასაღები ეწოდება. სწორედ ამ გასაღებით არის შესაძლებელი დავუკავშიროთ რამდენიმე ერთმანეთთან დაკავშირებული არსი ერთმანეთს.

მაგალითად:



მოცემულია ზემოთ ნახსენები მასწავლებლების ცხრილი და კლასების ცხრილი, რომელიც ერთმანეთთან არიან დაკავშირებული. ამ ტიპის კავშირს ერთი ბევრთან ეწოდება. შესაძლებელია არსებობდეს მასწავლებელი, მაგრამ არ იყოს ის კლასი, რომლის სპეციალისტიც არის ის და ამიტომ არ ასწავლიდეს. მასწავლებელი ყოველთვის არსებობს, ანუ მის მხარეს წერია ერთიანი, ხოლო კლასი შეიძლება საერთოდ არ იყოს, ან იყოს ერთზე მეტიც, ამიტომ მის მხარეს წერია 0...m. როგორც ზემოთ აღვნიშნეთ, კლასის ცხრილში არის მიმთითებელი მასწავლებელზე, teacherID, რაც წარმოადგენს ამ ცხრილის გარე გასაღებს.

რელაციურ მონაცემთა ბაზაში კავშირები სამი ტიპისაა, 1:1, 1:N, N:M

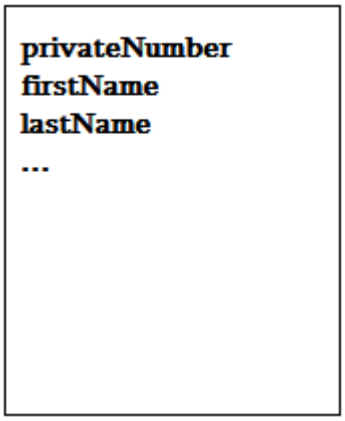
განვიხილოთ და დავახასიათოთ თითოეული მათგანი.

ერთი ერთთან კავშირი გულისხმობს რომ ერთი არსი დაკავშირებულია მხოლოდ და მხოლოდ ერთ არსთან მეორე ცხრილში. ზოგჯერ შესაძლებელია ამ ორი ცხრილის გაერთიანებაც, თუ ამით არ დავარღვევთ ნორმალიზაციის და რელაციის წესებს, ან თუ ამოცანიდან გამომდინარე მისაღები იქნება ეს დარღვევა.

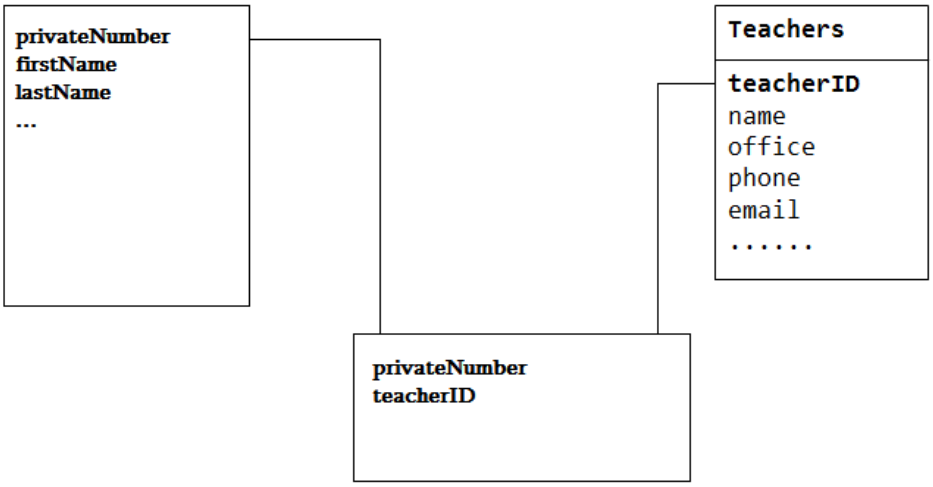
ერთი ბევრთან კავშირი, რომელიც ზემოთ ნაჩვენებ სურათზეა ასახული, გულისხმობს რომ ერთი არსი შეიძლება დაუკავშირდეს რამდენიმე არსს მეორე ცხრილში. რაც ხორციელდება მშობელი ცხრილის პირველადი გასაღების ჩასმით შთამომავალ ცხრილში.

კავშირი მრავალი მრავალთან ყველაზე რთულია ამ სამიდან. მისი განხორციელებისთვის საჭიროა გადაკვეთის ცხრილის შემოტანა. ეს ცხრილი ხდება შთამომავალი ორი წინაპრისა, რომელთა პირველადი გასაღებები იწერება გადაკვეთის

ცხრილში. ამ ცხრილის პირველად გასაღებად ხშირად ირჩევენ გარე გასაღებების კომბინაციას. შეიძლება ითქვას რომ ეს არის ორი ერთი ბევრთან კავშირი.



ნახატზე წარმოდგენილია სტუდენტი ცხრილი, რომლის პირველად გასაღებს წარმოადგენს სტუდენტის პირადი ნომერი. განვიხილოთ მისი კავშირი მასწავლებელ ცხრილთან. როგორც ცნობილია, შესაძლებელია ერთ სტუდენტს ყავდეს რამდენიმე მასწავლებელი და ერთ მასწავლებელს კი ბევრი სტუდენტი, ანუ მათ შორის არის მრავალი მრავალთან კავშირი. შევხედოთ მას დიაგრამაზე, სადაც გამოყენებულია გადაკვეთის ცხრილი.



სურათი ასახავს სტუდენტის და მასწავლებლის, მრავალი მრავალთან კავშირს. ამ შემთხვევაში დამხმარე ცხრილის, ანუ გადაკვეთის ცხრილის პირველადი გასაღებია კომპოზიტური(ორი ან მეტი ატრიბუტისგან შემდგარი) გასაღები, privateNumber, teacherID. არსი კავშირის დიაგრამაზე ეს ცხრილი არის სუსტი არსი, რადგან მასში ასახულ ობიექტებს

არ შეუძლიათ დამოუკიდებლად არსებობა. ამის საპირისპიროდ, ძლიერი არსი არის ცხრილი, რომელიც შეიძლება არსებობდეს დამოუკიდებლად.

2.3 მოდიფიკაციის ანომალიები

რელაციური მონაცემთა ბაზა გულისხმობს მოდიფიკაციის ანომალიების არ არსებობას. ამაში გვეხმარება ნორმალიზაციის წესები და ფორმები. მოდიფიკაციის ანომალია ჩნდება ცხრილში მონაცემის დამატებით, ცვლილებით ან წაშლით.

განვიხილოთ თითოეული მათგანი.

მონაცემების დამატებით გაჩენილი ანომალია ხდება მაშინ, როდესაც ერთი და იმავე მონაცემის ჩაწერა ხდება ბაზაში ერთზე მეტ ადგილას. სწორად ნორმალიზებულ ბაზაში ინფორმაციის დამატება ხდება მხოლოდ ერთ ადგილას. მრავალ ადგილზე დამატების საჭიროება გულისხმობს ამ ბაზის დიზაინის მოუქნელობაზე. გასათვალისწინებელია ასევე მომხმარებლის შეცდომებიც.

მონაცემის ცვლილების ანომალია მსგავსად დამატების ანომალიისა მაშინ ხდება, როდესაც საჭიროა მონაცემის შეცვლა რამდენიმე ადგილას. აქვე უნდა გავითვალისწინოთ მომხმარებლის შეცდომები.

წაშლის ანომალია ხდება, როდესაც საჭიროა ერთი მონაცემის წაშლა, რამდენიმე ადგილიდან, რათა არ მოხდეს ზედმეტი ინფორმაციის დარჩენა ბაზაში. თუკი მომხმარებელს სურს რაიმეს წაშლა, საკმარისი უნდა იყოს მხოლოდ ერთი ადგილიდან წაშლა.

რეკომენდირებულია ბაზის დიზაინის ნორმალიზება ამ წესების გათვალისწინებით, რათა ბაზაში არ აღმოჩნდეს შემდეგ, ისეთი მონაცემები, რომელიც გამოიწვევს ბაზისთვის გამოყოფილი ადგილის ზედმეტად გაზრდას.

3. მონაცემთა ბაზების ადმინისტრირება - ძირითადი საკითხები

მონაცემთა ბაზის ადმინისტრირება არის DBMS პროგრამული უზრუნველყოფის მოვლა-პატრონობა. ზოგიერთ DBMS სისტემას, როგორიცაა Oracle, MS Sql Server, ესაჭიროება მიმდინარე ადმინისტრირება. ამის გამო, კორპორაციები ქირაობენ სპეციალიზებულ პერსონალს DBMS სისტემების მართვისთვის, მათ ბაზის ადმინისტრატორები ეწოდებათ (DBA).

მონაცემთა ბაზის ადმინისტრატორს გააჩნია შემდეგი მოვალეობები.

- ინსტალაცია, კონფიგურაცია და განახლებების გაკეთება მონაცემთა ბაზის სერვერისთვის და შესაბამისი პროდუქტებისთვის
- მონაცემთა ბაზის შესაძლებლობების შეფასება
- სარეზერვო ასლების და ბაზის აღდგენის პოლისების და პროცედურების შემოღება და ადმინისტრაცია
- მონაცემთა ბაზის დიზაინი და იმპლემენტაცია
- მონაცემთა ბაზის დაცვა
- მონაცემთა ბაზის ტუნინგი და სისწრაფის მონიტორინგი
- აპლიკაციების ტუნინგი და სისწრაფის მონიტორინგი
- დოკუმენტაციების და სტანდარტების შემოღება და შესრულება
- მონაცემთა ბაზის ზრდის და ცვლილებების დაგეგმვა
- გუნდში მუშაობა და 24x7-ზე მხარდაჭერა.
- ძირითადი ტექნიკური პრობლემების გამოკვლევა
- მონაცემთა ბაზის აღდგენა

მონაცემთა ბაზის ადმინისტრირების სამი ტიპი არსებობს:

1. სისტემური ადმინისტრატორი: ფოკუსირდება ფიზიკურ ასპექტებზე, როგორიცაა ინსტალაცია, კონფიგურაცია, პატჩინგი, განახლება, სარეზერვო ასლების შენახვა, აღდგენა, სისწრაფის ოპტიმიზაცია, მოვლა და აღდგენა
2. განვითარების ადმინისტრატორები: ფოკუსირდება ლოგიკურ და დეველოპმენტის ასპექტებზე მონაცემთა ბაზისა, როგორიცაა მონაცემთა

მოდელის დიზაინი და მოვლა, DDL გენერაცია, SQL-ს წერა და ტუნინგი, ჩაშენებული პროცედურების წერა, მუშაობენ დეველოპერებთან ერთად, რათა დაეხმარონ აირჩიონ სწორი DBMS ფუნქციები და სხვა წინასწარი მოქმედებების განსაზღვრა.

3. აპლიკაციის ადმინისტრატორი: ძირითადად გვხვდება ორგანიზაციებში, რომელთაც შეიძინეს პროგრამული უზრუნველყოფა მესამე პირისგან, როგორიცაა ERP და CRM სისტემები. ეს ადმინისტრატორები არიან „ღობესავით“ მონაცემთა ბაზას და პროგრამას შორის და პასუხს აგებენ რომ აპლიკაცია სრულად ოპტიმიზირებულია მონაცემთა ბაზისთვის და პირიქით. ისინი მართავენ აპლიკაციის კომპონენტებს, რომლიც მუშაობს ბაზასთან და აპლიკაციის ინსტალაცია და პატჩინგიც მათ კისერზეა, ასევე განახლება, მონაცემთა ბაზის კლონირება, მონაცემთა გასუფთავება, დატვირთვის მენეჯმენტი და ა.შ.

მიღებულია რომ ინდივიდი სპეციალიზირდება ერთი ტიპის ბაზის ადმინისტრაციაზე, მაგრამ ხშირად შეხვდებით ისეთ პატარა ფირმებს, სადაც პიროვნება ან ჯგუფი ახდენს რამდენიმე ტიპის მონაცემთა ბაზის ადმინისტრირებას.

დონე, რომელზეც არის ადმინისტრირება ავტომატიზებული, გვკარნახობს შესაძლებლობებს, რომელიც უნდა ქონდეს პერსონალს, რომელიც მოახდენს ადმინისტრირებას. ერთის მხრივ, სისტემა რომელიც არ არის ავტომატიზებული, მოითხოვს უფრო გამოცდილ თანამშრომლებს ადმინისტრირებისთვის, სავარაუდოდ 5-10 ბაზა/ადმინისტრატორზე. ალტერნატივაა, რომ ორგანიზაციამ აირჩიოს მონაცემთა კარგად ავტომატიზებული ბაზა, რა შემთხვევაშიც აღარ იქნება საჭირო ძალიან გამოცდილი ადმინისტრატორი. ამ შემთხვევაში პერსონალი შეიძლება დაიყოს: კარგი გამოცდილი ადმინისტრატორები რომელიც ქმნიან ავტომატიზაციას და უბრალოდ ადმინისტრატორები, რომლებიც უშვებენ მას.

ბაზის ადმინისტრირება არის კომპლექსური, განმეორებადი, დროში გაწელვადი და მოითხოვს განსხვავებულ ვარჯიშს. რადგან მონაცემთა ბაზა შეიცავს მნიშვნელოვან მონაცემებს, ფირმები ძირითადად ეძებენ კანდიდატებს, რომელთაც აქვთ რამდენიმე წლიანი გამოცდილება. ადმინისტრირება ხშირად ითხოვს სამუშაო საათების გარდა მუშაობას, მაგ. როდესაც დაგეგმილია გამორთვა. ადმინისტრატორებს კარგად უნაზღაურდებათ ეს სამუშაო.

ერთი კვალიფიკაცია არის აუცილებელი მოთხოვნა ბაზის ადმინისტრატორის შერჩევისას. ეს არის სარეზერვო ასლის შექმნა და აღდგენა უბედური შემთხვევების დროს. საკითხავია როდის მოხდება უბედური შემთხვევა, ითვალისწინებს შემდეგს: ბაზის მარცხი, უბრალოდან კატასტროფულამდე. ეს შეიძლება იყოს მონაცემების წაშლა, მედიის (მყარი უზრუნველყოფის) ხარვეზი, ან მომხმარებლის შემოტანილი შეცდომა. ნებისმიერი სიტუაციიდან ადმინისტრატორს უნდა შეეძლოს გამოძრომა და დაბრუნება მონაცემთა ბაზისა საჭირო წერტილში და მონაცემთა დაკარგვის თავიდან აცილება. მაღალკვალიფიციურ ადმინისტრატორს შეუძლია გაატაროს რამდენიმე წუთი ან საათი რომ ბაზა დაბრუნდეს ისევ მუშა მდგომარეობაში.

DBMS პროგრამებს ხშირად მოყვება თან ხელსაწყოები, რათა გაუადვილოს ადმინისტრატორს საქმე. ასეთ ხელსაწყოებს ეწოდებათ თანდაყოლილი. მაგალითად MS Sql სერვერს მოყვება enterprize manager და Oracle-ს SQL plus, enterprize manager/grid control. ასევე მესამე პირებიც ქმნიან ამ ტიპის პროექტებს, რომელიც მომხმარებელს სთავაზობს გრაფიკულ ინტერფეისს და ეხმარება ადმინისტრატორს შეასრულოს სხვადასხვა ფუნქციები უფრო მარტივად.

სხვა ტიპის ხელსაწყოები არსებობს ახალი მონაცემთა ბაზის შექმნისთვის ან უკვე არსებულის მენეჯმენტისთვის. ახალი ბაზის შექმნა შეიძლება შედგებოდეს ათასობით წინასწარი ნაბიჯისგან, რათა დავაკმაყოფილოთ წინასწარი მოთხოვნები. ამასთან ყველა უნდა დასრულდეს წარმატებით, რათა გადავიდეთ შემდეგზე. ადამიანს არ შეუძლია ეს ყველაფერი ზუსტად ერთი და იმავე თანმიმდევრობით გაიაროს. რაც უფრო იზრდება ადმინისტრატორების რიცხვი, ავტომატიზაციის გარეშე უნიკალური კონფიგურაციებიც იზრდება და უფრო რთული ხდება. ყველა ეს რთული პროცედურა შეიძლება მოდელირდეს საუკეთესო ადმინისტრატორის მიერ ბაზის ავტომატიზაციის სისტემაში და შესრულდეს სტანდარტული ადმინისტრატორის მიერ. პროგრამები იქმნება სპეციალურად, რათა გაუმჯობესდეს მდგრადობა და განმეორებადობა ამ პროცედურებისა.

4. ვებ-ზე დაფუძნებული მონაცემთა ბაზის ადმინისტრირების სამომხმარებლო სისტემა

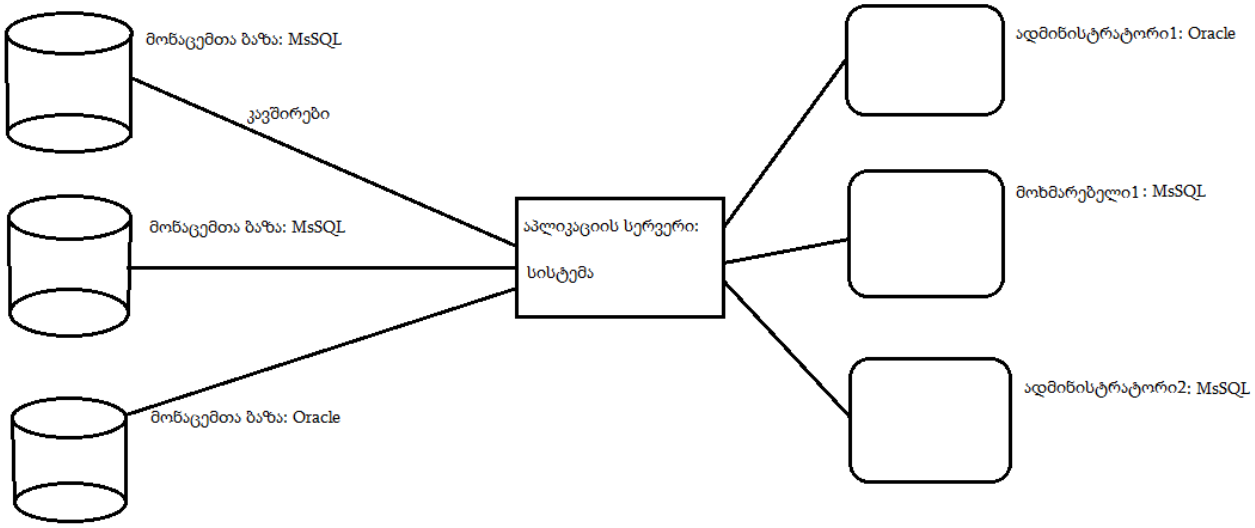
მონაცემთა ბაზის გამოყენების საყოველთაობის ზრდა და ვებ ორიენტირებული სხვადასხვა დანიშნულების საინფორმაციო სისტემების რაოდენობრივი და თვისობრივი ზრდა მწვავედ აყენებს საკითხს მონაცემთა ბაზების ადმინისტრირების ისეთი ინსტრუმენტის შექმნის აუცილებლობის შესახებ, რომელიც არაპროფესიონალ მომხმარებელს დაეხმარება დამოუკიდებლად მოახდინოს მისი საკუთარი ვებ-ორიენტირებული მონაცემთა ბაზის ადმინისტრირება ბაზის მთლიანობის დარღვევის გარეშე.

ამ პრობლემის გადასაჭრელად შემუშავდა პროგრამული სისტემა, რომელშიც რეალიზებულია ბაზის ადმინისტრირების ზოგიერთი ძირითადი საშუალება ვიზუალურ დონეზე, ხოლო მოთხოვნების დონეზე შესაძლებელია ადმინისტრირების ყველა ძირითადი ოპერაციის შესრულება.

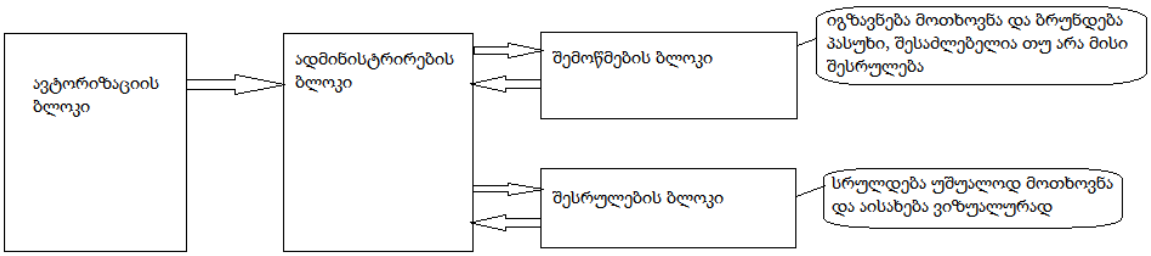
ფოკუსირდება აპლიკაციის ადმინისტრატორის მოვალეობაზე, რომელიც ქმნის ბაზის დიზაინს. ამ ყველაფრისთვის შექმნილია ვიზუალური ინტერფეისი, და შესაძლებელია მუშაობა MS Sql და Oracle ბაზასთან.

ვებზე დაფუძნებული სისტემის სიმარტივე იმაში მდგომარეობს, რომ იგი უზრუნველყოფს მოშორებული სერვერიდან წვდომას ბაზის სერვერზე, რა თქმა უნდა ამისთვის საჭიროა გვქონდეს დაშვება და ვიცოდეთ ადმინისტრატორის მომხმარებლის სახელი/პაროლი. ასევე სისტემა უზრუნველყოფს მაქსიმალურ დაცვას, რაც შეიძლება შეეზღუდოს ადმინისტრატორს თვითონ ვებ-ის დონეზე, ხოლო შემდეგ ბაზის დონეზეც.

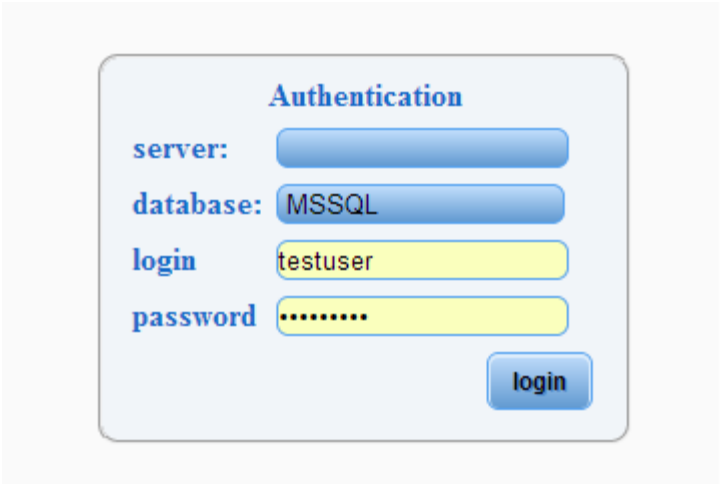
პროექტის ფარგლებში წარმოდგენილია ბაზის ადმინისტრირების უნივერსალური მოდელი. მოდელი საშუალებას იძლევა ჩაშენდეს რიგითი მომხმარებლის პროექტში, რომელსაც ესაჭიროება ბაზის ადმინისტრირება. რიგით მომხმარებლისთვის ძალიან მარტივია გამოიყენოს ეს პროგრამა, რადგან დაფუძნებულია ვებ ტექნოლოგიებზე და ადმინისტრირების საშუალება გვაქვს ვიზუალური ინტერფეისის გამოყენებით, რაც ამარტივებს ბაზის მართვას.



სისტემაში ჩაშენებულია რამდენიმე ბლოკი: 1. იდენტიფიკაციის ბლოკი. 2.ადმინისტრირების ბლოკი. 3.შემოწმების ბლოკი. 4.ბაზაში მონაცემების შეტანა.



1. იდენტიფიკაციის ბლოკი. ამ მოდულში ხდება მომხმარებლისა და ბაზის იდენტიფიცირება, რომელ ბაზაზე სურს მომხმარებელს დაკავშირება, გააჩნია თუ არა მას ამის უფლება და თუ გააჩნია, მაშინ რა უფლებებით სარგებლობს იგი ამ ბაზაზე.



2. ადმინისტრირების ბლოკში თავმოყრილია მთელი ადმინისტრირების სისტემა. აქ წარმოდგენილია თუ რა მონაცემთა ბაზები არსებობს სერვერზე, თითოეულ მონაცემთა ბაზაში რა ცხრილები ინახება, შესაძლებელია კონკრეტული ცხრილის კონკრეტული ჩანაწერის ცვლილება/წაშლა ვიზუალური ინტერფეისის გამოყენებით, ან მოთხოვნის საფუძველზე. არსებობს მოთხოვნის არე, სადაც მომხმარებელს საშუალება აქვს დაწეროს მისთვის საჭირო მოთხოვნა, თუ ის არ არის გათვალისწინებული სისტემაში, ეს მოთხოვნა გაივლის როგორც სინტაქსურ, ასევე უფლებების შემოწმებას და მომხმარებელს საშუალება ექნება დადებითი პასუხის შემთხვევაში შეასრულოს იგი. აქვეა მომხმარებლების ჩამონათვალი, რომელი მომხმარებელი რა ფუნქციებით სარგებლობს, არის ის სისტემის ადმინისტრატორი თუ ბაზის მფლობელი. კონსტრეინტების არე, სადაც შესაძლებელია რამდენიმე სახის კონსტრეინტის დამატება: Check, FK, Unique. ვიზუალური ინტერფეისი საშუალებას გვაძლევს შევიტანოთ ინფორმაცია საჭირო ადგილას. ამ ინფორმაციის საფუძველზე გენერირდება მოთხოვნა.

სურათზე წარმოდგენილია book ბაზის Table_1 ცხრილის სტრუქტურა, სტრუქტურის ცვლილების საფუძველზე შეიცვლება ცხრილის სტრუქტურაც. შესაძლებელია დაემატოს რამდენიმე სტრიქონიც, რის შედეგადაც ანალოგიურ ფორმას მივიღებთ და შესაძლებელი იქნება შეტანილი ინფორმაციის საფუძველზე დაგენერირდეს მოთხოვნა და შეიცვალოს ცხრილი.

3. შემოწმების ბლოკში ზემოთ ხსენებული მოთხოვნა მოწმდება სინტაქსურ შეცდომებზე. ხდება მომხმარებლის უფლების შემოწმება, გააჩნია თუ არა მას ამ მოთხოვნის შესრულების საშუალება, მაგ. ცხრილის შექმნა ან წაშლა.
4. ბაზაში მონაცემები შეიტანება მოთხოვნის საფუძველზე. თუ ყველა ზემოთ ხსენებული მოდული წარმატებით დასრულდა, მაშინ ბაზაში მოხდება მოთხოვნის განხორციელება, თუ არა, მაშინ ამ მოთხოვნას უარვყოფთ.

სისტემა რიგით ბაზის მომხმარებელს გაუადვილებს მონაცემთა ბაზის შექმნა/ადმინისტრირებას იმ დონეზე, რომ შესაძლებელი იყოს ამ ბაზის რეალურ სისტემაში გამოყენება. პროგრამის შესაძლებლობები ბაზის ადმინისტრატორის ხელში თითქმის შემოუსაზღვრელია, მაგრამ რიგით მომხმარებელს დაეხმარება შექმნას ცხრილები, დაამატოს მათზე ინდექსები, შექმნას ცხრილებს შორის კავშირები და ა.შ.

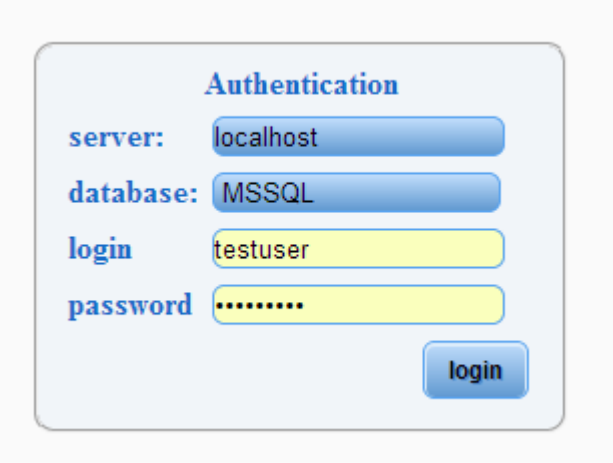
პროგრამა დაფუძნებულია ვებ ტექნოლოგიებზე, იგი მომხმარებელთან ამყარებს ვიზუალურ კონტაქტს, ხოლო მომხმარებლის მიერ შეყვანილი ინფორმაციის საფუძველზე აგენერირებს მოთხოვნას რომელიც შემდეგ სრულდება ბაზის მხარეს. შესრულებული მოთხოვნა, მყისიერად ისახება მომხმარებელთან.

სისტემას აქვს საშუალება არ გამოიყენოს ვიზუალური ელემენტები, ამისთვის არის ჩანართი SQL-ტაბი, რომელთან მუშაობაც შესაძლებელია გამოცდილი ადმინისტრატორებისთვის, აქ შესაძლებელია ჩაიწეროს ნებისმიერი მოთხოვნა(CRUD(Create, Read, Update, Delete) ჯგუფიდან და სხვა) და იგი შესრულდება მონაცემთა ბაზაზე.

ეს სისტემა ასევე გვაძლევს საშუალებას რომ მივუერთდეთ სასურველ ბაზას, სადაც მხოლოდ მომხმარებლის ფუნქციებით ვსარგებლობთ და შევასრულოთ ჩვენთვის ნებადართული ცვლილებები, მაგ. ცხრილების შექმნა/ცვლილება/წაშლა, მონაცემების დამატება/ცვლილება/წაშლა, კონსტრეინტების დამატება და ა.შ.

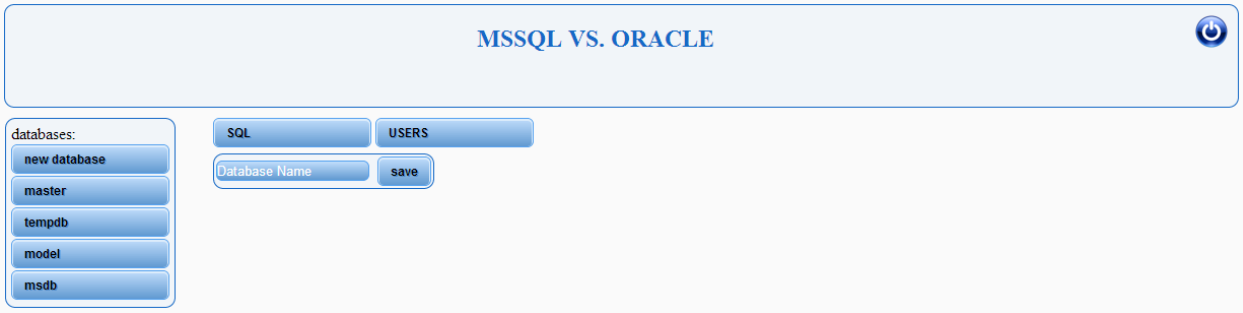
განვიხილოთ მუშაობის ციკლი თანმიმდევრობით:

დასაწყისი, რაც პირველად გამოჩნდება არის ავტორიზაცია პროგრამაში და მონაცემთა ბაზაზე, რომელსაც ემსახურება ფორმა:

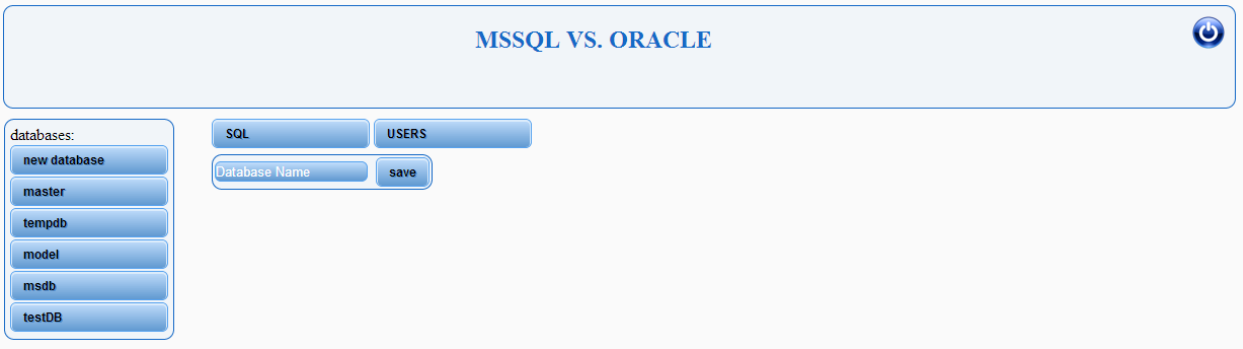


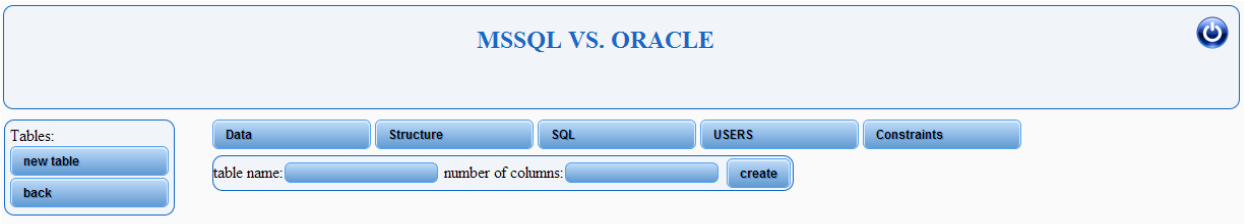
შეტანილი მონაცემები გადამოწმდება სერვერზე, რომელთან დაკავშირებასაც ვცდილობთ, თუ წარმატებით არ დასრულდება, გამოვა შეცდომის შეტყობინება.

ავტორიზაციის წარმატებით გავლის შემდეგ, სისტემა გვიშვებს მთავარ გვერდზე, სადაც წარმოდგენილია შესაბამის სერვერზე არსებული მონაცემთა ბაზების ჩამონათვალი და მათზე შესასრულებელი ფუნქციონალი



აქვეა შესაძლებელია მონაცემთა ბაზის შექმნის საშუალება, ამისთვის მხოლოდ Database Name-ში ვუთითებთ ბაზის სასურველ სახელს და ვაწვებით ღილაკს save. მაგ. შევქმნათ testDB, როგორც ვხედავთ იგი აისახება ბაზების ჩამონათვალში.





შემდეგი მოქმედება იქნება ცხრილის შექმნა ბაზაში, ამისთვის ვირჩევთ ბაზას და ჩნდება შესაბამისი ფორმა. ყურადღება მიაქციეთ, რომ მარცხენა მხარეს უკვე გაჩნდა ბაზაში არსებული ცხრილების სახელები, მონაცემთა ბაზების მაგიერ. მიმდინარე ბაზაში ჯერ არ გვაქვს ცხრილები, შესაბამისად ის ცარიელია.

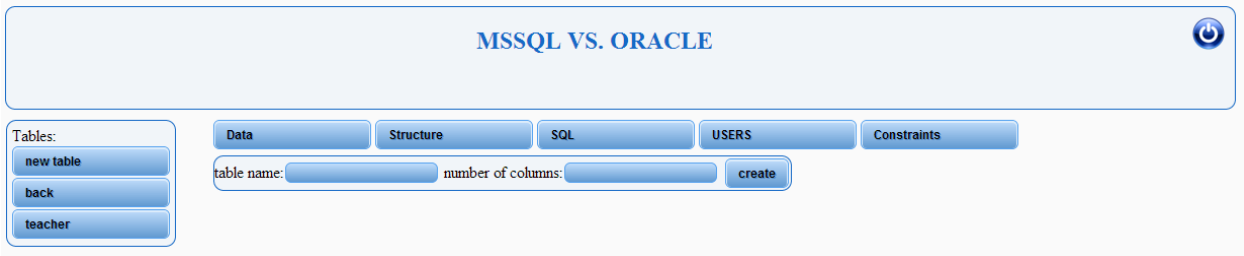
სადაც უნდა მივუთითოთ ცხრილის სახელი და რამდენი ველი ექნება ცხრილს, ორივე პარამეტრის შეცვლა შესაძლებელია მომავალში

მაგ. შევქმნათ teacher ცხრილი, სამი ველით, იდენტიფიკატორი ანუ teacherID, სახელი და გვარი. ამისთვის table name:-ში ვწერთ teacher, ხოლო Number of columns-ში 3 და ვაწვებით create

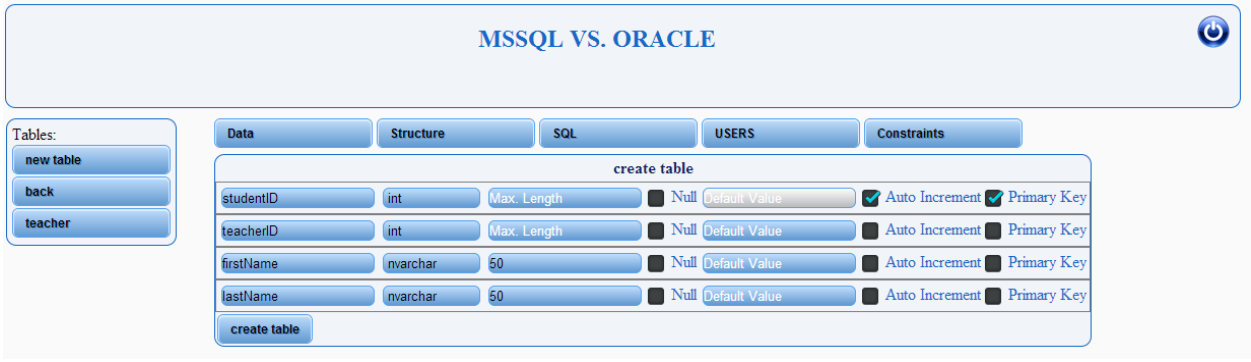
გამოვა ფორმა, რომელიც აზუსტებს ცხრილის სტრუქტურას. აქ იმდენი სტრიქონია ჩამოთვლილი, რამდენი ველიც მოვითხოვეთ წინა ფორმაში, თითოეული სტრიქონი შეესაბამება ველს.

შევიტანოთ შესაბამისი ინფორმაცია და დავაჭიროთ create table.

შესაბამისად შეიქმნება ცხრილი სამი ველით და ჩაჯდება ცხრილების ჩამონათვალში.

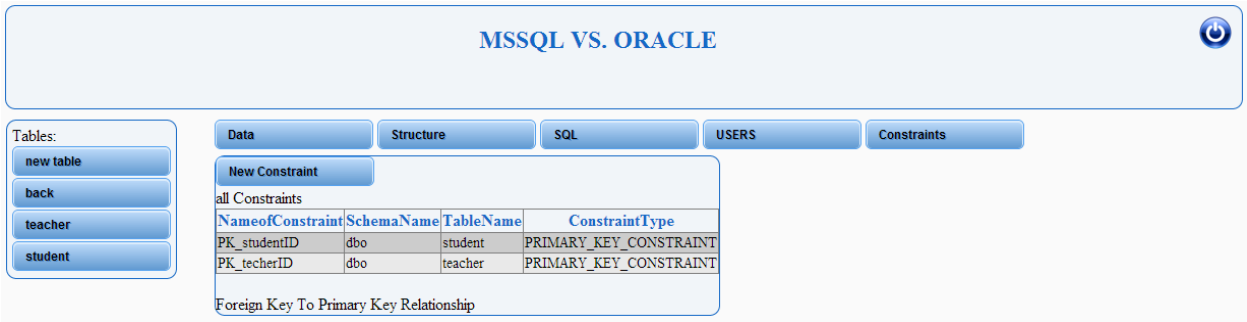


ახლა დავამატოთ მეორე ცხრილი, student, ოთხი ველით, ვიგულისხმოთ რომ ერთი მასწავლებელი ასწავლის რამდენიმე სტუდენტს, ანუ გვაქვს ერთი ბევრთან კავშირი, ამისთვის ცხრილში ვითვალისწინებთ სვეტს, teacherID, რომელიც შეესაბამება პირველად გასაღებს teacher ცხრილიდან.



შევქმნათ ეს ცხრილიც და გადავიდეთ მათ შორის კავშირის ბაზის დონეზე ასახვაზე, ანუ უნდა შევქმნათ გარე გასაღების მიმართება.

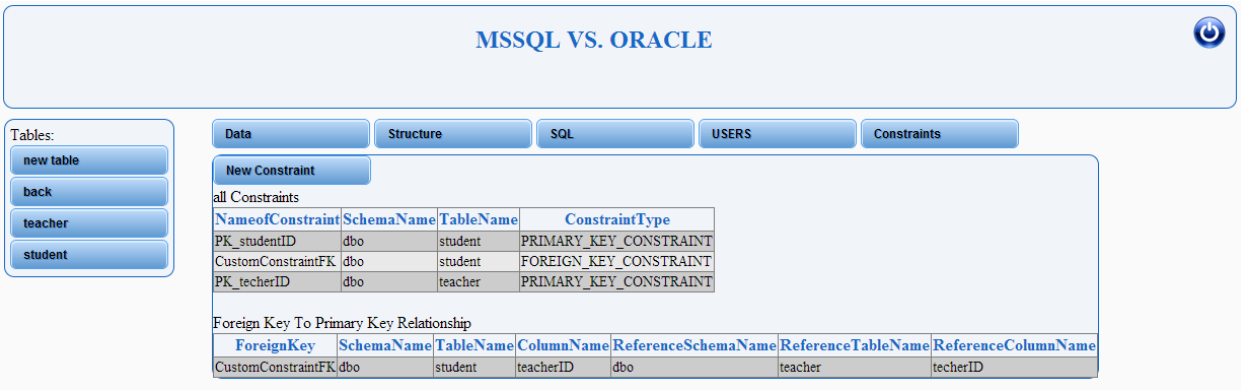
ამისთვის უნდა გადავიდეთ constraint ჩანართზე, რომელიც არის ზედა მენიუში



ვირჩევთ new constraint, რომელსაც გამოაქვს ფანჯარა კონსტრეინტების შესაქმნელად. გათვალისწინებულია სამი სახის კონსტრეინტი, Check, Unique, Foreign key, ჩვენ გვაინტერესებს foreign key

ვირჩევთ შესაბამის კონსტრეინტის ტიპს. კონსტრეინტის სახელი არ არის სავალდებულო, ბაზას შეუძლია თვითონ დაარქვას მას სახელი, მაგრამ ველები აუცილებელია. როგორც ზემოთ ვთქვით, teacherID იქნება გარე გასაღები student ცხრილში, ვავსებთ ინფორმაციას სურათის მიხედვით.

ვაწვებით save, და ვხედავთ რომ წეღანდელ კონსტრეინტებს დაემატა კიდევ ერთი



მოვიდა დრო, რომ შევიტანოთ მონაცემები თითოეულ ცხრილში. ეს ხორციელდება შემდეგნაირად, ვირჩევთ რომელ ცხრილში გვინდა მონაცემების დამატება. მაგ ვირჩევთ teacher ცხრილს. შემდეგ გადავდივართ SQL ჩანართზე, სადაც ვხედავთ ქვემოთ რამდენიმე ღილაკს, clear, select *, select, insert, update, delete, check, go.

ჩვენ შემთხვევაში ავირჩიოთ insert ღილაკი, იგი დააგენერირებს უკვე მზა მოთხოვნას, რომელსაც ესაჭიროება მცირე ცვლილებები. დაგენერირებული მოთხოვნა:

„INSERT INTO teacher(teacherID, firstName,lastName) VALUES('VALUE-0','Value-1',Value-2'". რა თქმა უნდა ამ მოთხოვნას ესაჭიროება გადაკეთება, გადავაკეთოთ იგი, სურათზე ნაჩვენების მიხედვით და დავაჭიროთ check, შემდეგ go ღილაკს.

დავინახავთ რომ მოხდა სტრიქონის დამატება ჩვენს ცხრილში

თუ გვინდა მონაცემის ცვლილება, დავაჭიროთ Select ღილაკს სტრიქონის გასწვრივ, გამოჩნდება სტრიქონის ცვლილების ფორმა

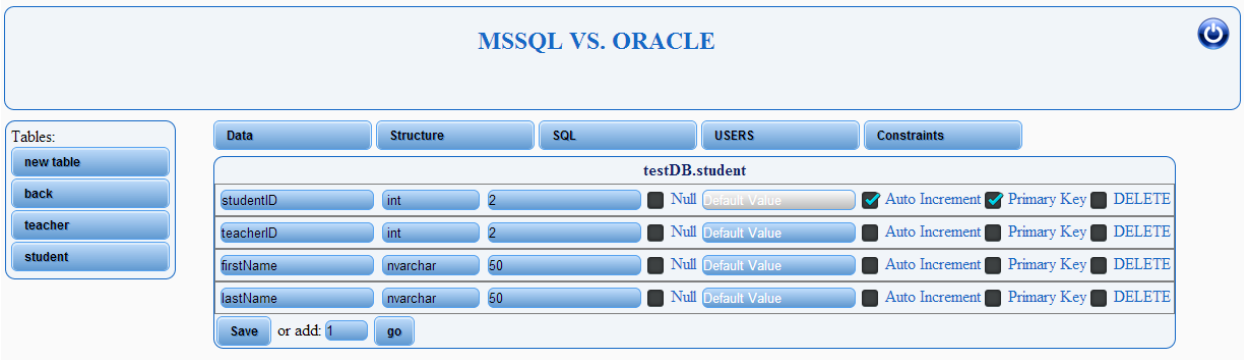
ყურადღება მიაქციეთ, რომ პირველადი გასაღების ველის ცვლილების საშუალებას არ იძლევა, იგი ავტომატურად გაფერმკრთალებულია. დანარჩენი შეგვიძლია სასურველად ვცვალოთ და save ღილაკს დავაჭიროთ.

Delete ღილაკის დაჭერის შემთხვევაში მიმდინარე სტრიქონი ამოიშლება ბაზიდან.

იგივე მიმდევრობით კეთდება სტუდენტის ჩამატება ცხრილში. განვიხილოთ, რათა ვაჩვენოთ Check ღილაკის ფუნქცია

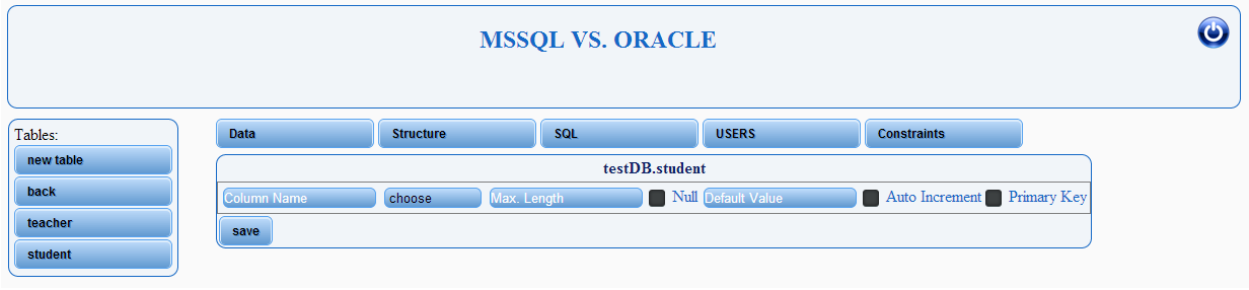
როგორც ხედავთ, პროგრამამ დააფიქსირა შეცდომა, „შეცდომითი ან შეუმოწმებელი მოთხოვნა“, რაც გამოიწვია go ღილაკზე დაჭერამ. მოთხოვნაში არ არის სინტაქსური შეცდომები, მაგრამ შეცდომის თავიდან ასაცილებლად, პროგრამა მაინც გვთხოვს მოთხოვნის შემოწმებას. დავაწყვეთ Check ღილაკს, go გამწვანდება და შესაბამისად შევძლებთ მონაცემის დამატებას.

პროგრამა გვაძლევს ცხრილის სტრუქტურის ცვლილების საშუალებასაც, განვიხილოთ student ცხრილის სტრუქტურა



უკვე ნაცნობი სტრუქტურაა, მაგრამ განსხვავებულია მცირე წვრილმანებით, დამატებულია delete საშუალება, სტრიქონის გასწვრივ, რომლის მონიშვნის შემთხვევაშიც შესაბამისი სტრიქონი წაიშლება.

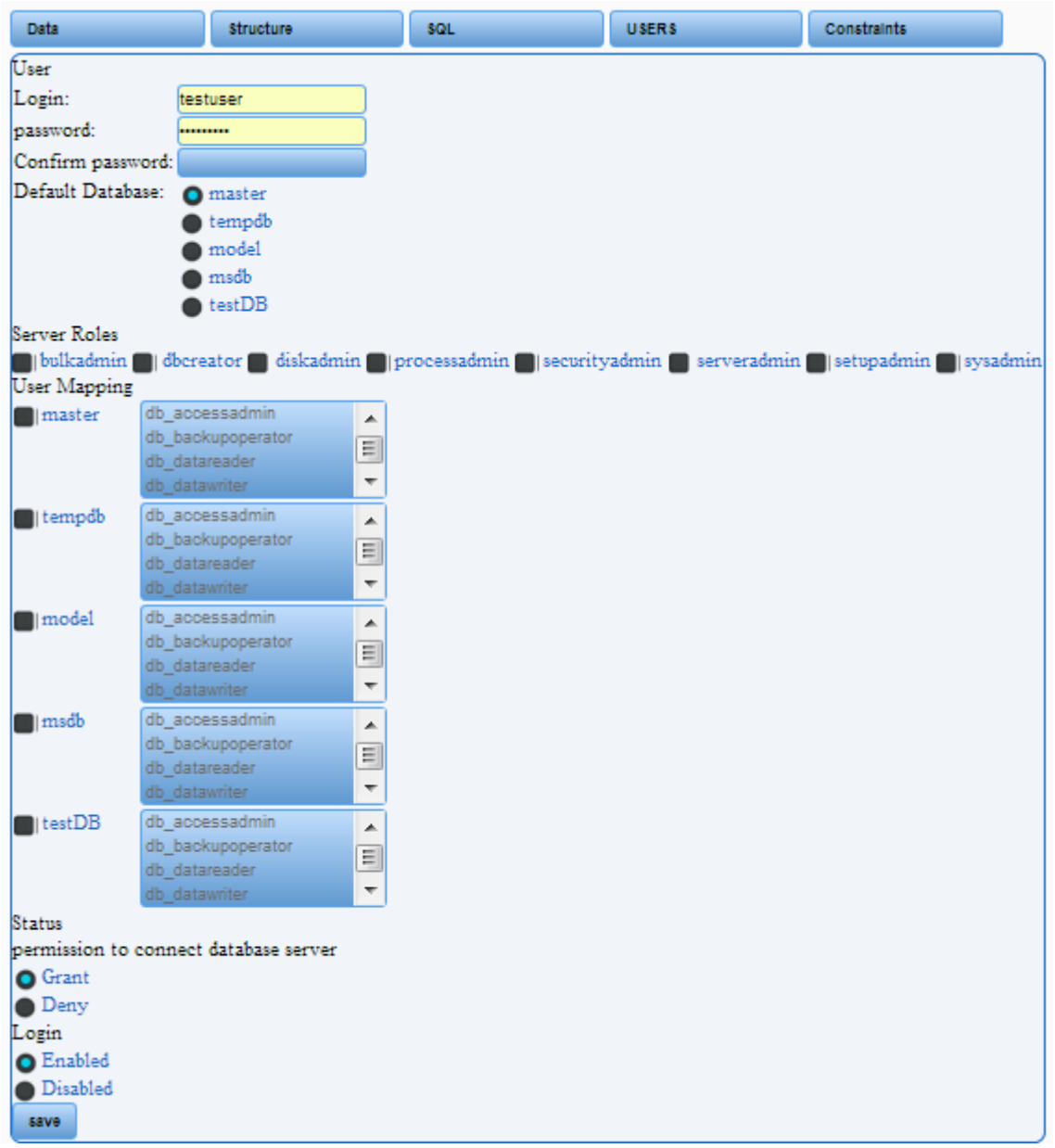
ასევე შეამჩნევდით „or add: 1 go“. აქედან შემიძლია დავამატო კიდევ ერთი ან მეტი სვეტი. თუ გამოგვრჩა სტუდენტის ცხრილში grade სვეტი, დავაჭიროთ go-ს და შევავსოთ შემდეგი ფორმა.



შევხედოთ ახალ სტუდენტი ცხრილის ახალ სტრუქტურას, სადაც უკვე ჩამატებულია შესაბამისი ატრიბუტი

იგი აუცილებლად ნებას უნდა რთავდეს “NULL” მნიშვნელობებს, რათა ბაზამ შეძლოს ცხრილის რედაქტირება.

და ბოლოს, შესაძლებელია ამავე ბაზიდან ახალი მომხმარებლის დამატება, რისთვისაც ივსება ფორმა



პირველ რიგში მოყვანილია მომხმარებლის სახელის და პაროლის შესაბამისი ველები, შემდეგ, რა როლებით ისარგებლოს ახალმა მომხმარებელმა სერვერზე.

რომელ ბაზასთან რა უფლებები ექნება და ბოლოს სტატუსი და მომხმარებლის user-ის აქტივაცია/დეაქტივაცია.

ამით ჩვენ დავასრულეთ სისტემის მცირე ნაწილის განხილვა.

დასკვნა

ჩვენი მიზანი იყო შეგვექმნა მომხმარებლის მხარდამჭერი სისტემა, რომელიც დახმარებას გაუწევს არაპროფესიონალ მომხმარებელს მოახდინოს საკუთარი მონაცემთა ბაზის ადმინისტრირება ვებ წვდომის ყველა ინსტრუმენტის გამოყენებით.

ამისთვის შეიქმნა პროექტი, რომლის ავტომატიზირების დონეც საშუალებას იძლევა მხოლოდ მონაცემები შევიყვანოთ ფორმებში და პროგრამამ თავის თავზე აიღოს მოთხოვნების გენერირება.

სისტემა რეალიზებულია ASP.Net Web forms ტექნოლოგიაზე, რაც საშუალებას იძლევა დაიწეროს დიდი სისტემა მცირე დროში, რადგან საჭირო ვიზუალური ელემენტები უკვე არსებობს, განსაზღვრულია კომპანია Microsoft-ის მიერ.

ასევე გამოყენებულია .Net, JavaScript(jQuery), ajax ტექნოლოგიები.

პროგრამას აქვს საშუალება შეუერთდეს და აწარმოოს როგორც Oracle, ასევე MsSql მონაცემთა ბაზის ადმინისტრირება, ასევე შესაძლებელია დაემატოს კიდევ სხვა მონაცემთა ბაზები, რაც განსაზღვრულ სამუშაოების ჩატარებას გულისხმობს.

პროგრამა შესაძლებელია განთავსდეს ინტერნეტ-ში და შემდეგ ბაზის ადმინისტრატორებმა გამოიყენონ იგი დანიშნულებისამებრ.

ლიტერატურა

1. World Wide Web

   http://en.wikipedia.org/wiki/World_Wide_Web

2. Web 2.0

   http://en.wikipedia.org/wiki/Web_2.0

3. Database

   http://en.wikipedia.org/wiki/Database

4. Database administration and automation

   http://en.wikipedia.org/wiki/Database_administration_and_automation

5. Database systems

   http://www.database-books.us/databasesystems.php

6. Beynon-Davies P. (2004). *Database Systems* 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

7. Codd, E.F. (1970)."A Relational Model of Data for Large Shared Data Banks"

8. Connolly, Thomas and Carolyn Begg. *Database Systems.* New York: Harlow, 2002.

9. "World Wide Web Consortium". "The World Wide Web Consortium (W3C)..."

10. "Inventing the Web: Tim Berners-Lee's 1990 Christmas Baby

11. Niels Brügger, ed. *Web History* (2010) 362 pages; Historical perspective on the World Wide Web, including issues of culture, content, and preservation.

12. Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. (June 1999). *Hypertext Transfer Protocol – HTTP/1.1*. Request For Comments 2616. Information Sciences Institute.