ივანე ჯავახიშვილის სახელობის თბილისის
სახელმწიფო უნივერსიტეტი

სოფიო ჩოქური

მონაცემთა ბაზის უსაფრთხოების უზრუნველყოფის მეთოდები

სამაგისტრო ნაშრომი შესრულებულია საინფორმაციო ტექნოლოგიების
მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მანანა ხაჩიძე
ტექნიკის მეცნიერებათა კანდიდატი
აკადემიური დოქტორი
თანახელმძღვანელი: გელა ბესიაშვილი

თბილისი
2013

ანოტაცია

სამაგისტრო ნაშრომი შეეხება დღესდღეობით ისეთ აქტუალურ პრობლემას, როგორიცაა მონაცემთა ბაზის უსაფრთხოება. მასში განხილულია სხვადასხვა მეთოდები, მიდგომები და რამდენიმე სხვა ასპექტი , რომლებიც ბაზის უსაფრთხოებას უზრუნველყოფენ.

ნაშრომში დეტალურადაა აღწერილი თუ როგორ უნდა მოვახდინოთ უსაფრთხოების სისტემის სწორად დაგეგმვა და იმ შემთხვევაში თუ მონაცემთა ბაზასთან რამდენიმე მომხმარებელი მუშაობს გავმიჯნოთ მათი უფლებები. სწორად განვსაზღვროთ როლები, რომლებშიც იუზერები იქნებიან გაერთიანებული.

სამაგისტრო ნაშრომში მოყვანილია ბიბლიოთეკის მონაცემთა ბაზის მაგალითი და სხვადასხვა SQL-ის რეკომენდაციები მონაცემთა უსაფრთხოებასთან დაკავშირებით.

Annotation

Master's thesis about the urgent problems of today, such as database security. It discusses a variety of methods, approaches, and several other aspects, which ensure the security of the base.

The paper describes in detail how to properly plan and manage the security of the system if several users are working with a database separate from their rights. Determine what roles, in which participants will be united.

Master's thesis in the library database are examples of the various SQL's recommendations regarding the security of data.

შესავალი

კომპიუტერული ტექნიკის სწრაფი განვითარების, ელექტრონული საკომუნიკაციო არხების, ინტერნეტის შექმნისა და განვითარების შედეგად დღეს ფაქტობრივად ინფორმაციის ძირითად მატარებლად იქცა ელექტრონული სიგნალი. ჩვენს დროში, როცა სწრაფი კომუნიკაციის და ინფორმაციის ეფექტური მიმოცვლის საკითხი საკმაოდ აქტუალურია, აუცილებელი გახდა ელექტრონული და ციფრული ტექნოლოგიების გამოყენება თითქმის ყველა სფეროში. ისინი გვეხმარებიან მარტივად, დროის ძალიან მოკლე პერიოდში გადავაგზავნოთ ინფორმაცია მსოფლიოს ნებისმიერი წერტილიდან სხვა ნებისმიერ წერტილში. თანამედროვე კომპიუტერული სისტემები და ინფორმაციული ტექნოლოგიები საშუალებას გვაძლევენ მარტივად მივიღოთ, დავამუშავოთ, შევინახოთ და, საჭიროების შემთხვევაში, გამოვიყენოთ უზარმაზარი მოცულობის ინფორმაცია საკმაოდ სწრაფად და ეფექტურად. მიუხედავად იმისა , რომ დღეს დღეობით ინფორმაციული ტექნოლოგიების სფერო ძალიან განვითარებულია, რჩება პრობლემები რომელთა გადაჭრაც უმნიშვნელოვან საკითხს წარმოადგენს.

მონაცემთა ბაზის უსაფრთხოების საკითხი ამ პრობლემებს შორის ერთ-ერთი ყველაზე აქტიურია , რომლის გადაწყვეტასაც მსხვილი ტექნოლოგიური კომპანიები და სახელმწიფოთა მთავრობებიც კი ცდილობენ. ნაშრომი ეძღვნება სწორედ ინფორმაციის დაცვის აქტუალურ საკითხებს. მონაცემთა ბაზა ინფორმაციული სისტემების ძირითადად ფორმას წარმოადგენს. მონაცემთა ბაზის განსაზღვრა შეიძლება , როგორც ურთიერთდამოკიდებული ერთად შენახული მონაცემების შეთანაწყობა. მონაცემთა ბაზა შეიძლება არსებობდეს კონკრეტული პროგრამის დამოუკიდებლად და იგი განსაზღვრულია ბევრი მომხმარებლის ერთობლივი გამოყენებისათვის. ამგვარი ცენტრალიზაცია და მონაცემთა ბაზის ტექნოლოგიაში მონაცემთა დამოუკიდებლობამ განსაზღვრა შესაბამისი მონაცემთა ბაზის მართვის სისტემის შექმნის აუცილებლობა , რაც უზრუნველყოფს

მონაცემთა კორექტული განთავსების ოპერაციების შესრულებას, მათ საიმედო შენახვას, ძიებას, მოდიფიკაციასა და წაშლას.

კლიენტ-სერვერული არქიტექტურის მქონე განაწილებული მონაცემთა ბაზები ძირითადად ფუნქციონირებს გლობალური ან ლოკალური ქსელის მასშტაბებით. ეჭვს არ იწვევს, რომ ნებისმიერი კორპორაციული მართვის სისტემას უნდა გააჩნდეს ინფორმაციის დაცვის საშუალებანი. აქ იგულისხმება როგორც არაკორექტული მონაცემების დამახსოვრების თავიდან აცილების შესაძლებლობა, ასევე მონაცემთა ბაზასთან მომხმარებელთა უნებართვო (არასანქციური) მიმართვების გამორიცხვა. ინფორმაციის არაკორექტულობა შეიძლება გამოწვეული იყოს შემთხვევით, მონაცემების ან პროგრამის შეცდომით ჩაწერისას მანქანაში ან წინასწარი განზრახვით. ამგვარად, ინფორმაციის დაცვა ორი ამოცანის გადაწყვატას ითხოვს:

- მონაცემთა მთლიანობის უზრუნველყოფას
- საიდუმლოების გარანტიას (მონაცემების მისაღებად შეზღუდ-ვების დაყენებას).

მონაცემთა მთლიანობა.

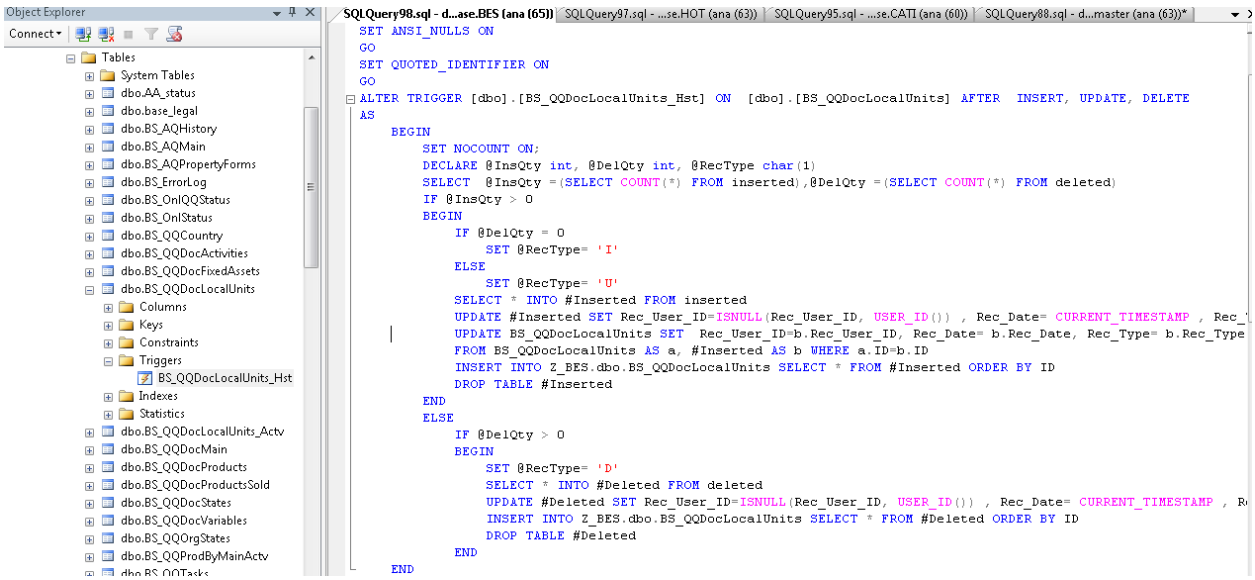
მონაცემთა ბაზის მთლიანობის უზრუნველსაყოფად იყენებენ შეზღუდვების შერჩევის მექანიზმს. ერთი მათგანია სტრუქტურული შეზღუდვები, მეორე კი უშუალოდ მონაცემთა მნიშვნელობების შეზღუდვები.

შეზღუდვები სტრუქტურულ დონეზე ეფუძნება მონაცემთა ბაზებში ფუნქციონალურ დამოკიდებულებათა (რელაციების, ატრიბუტების და ა.შ.) აღწერას. შემოიტანება სპეციალური _ გასაღებური ატრიბუტების, ინდექსების ცნებები (მარტივი ან შედგენილი). მათი საშუალებით ხორციელდება რელაციურ ფაილებში ინფორმაციის მოწესრიგება და მონაცემთა ძებნა, ამორჩევა.

მთლიანობის შეზღუდვების მეორე დონეა მონაცემთა მნიშვნელობების ცვლილებები. მონაცემთა მნიშვნელობები შეიძლება განსაზღვრული იყოს გარკვეულ დიაპაზონებში ან

მნიშვნელობები გაანგარიშებული უნდა იქნას რომელიმე მათემატიკური დამოკიდებულებით (ფორმულით). თუ მონაცემის მნიშვნელობა გამოვა განსაზღვრული დიაპაზონიდან ან არ შეესაბამება არსებულ მათემატიკურ დამოკიდებულებას, მაშინ ხორციელდება სპეციალური დამცველი ფუნქციების ამუშავება, რათა არ დაირღვეს ბაზის მთლიანობა.

მაგალითისათვის შეიძლება განვიხილოთ თანამედროვე მონაცემთა ბაზების სისტემებში (მაგალითად, MsSQL_Server, Oracle და სხვ.) ინსტრუმენტული საშუალება Triggers (ჩართვა-გამორთვის ფუნქცია). ტრიგერები უზრუნველყოფს მონაცემთა მთლიანობას ლოგიკურად დაკავშირებულ ცხრილებში (რელაციებში). სისტემაში სტანდარტული ან კერძო ფუნქციების ამუშავება განისაზღვრება მომხმარებლის მიერ, ტრიგერები კი არაა დამოკიდებული პროგრამაზე, იგი ჩაირთვება ყოველთვის, როდესაც ადგილი აქვს მონაცემთა ბაზაში ინფორმაციის განახლებას: ახლის ჩამატებას, ძველის წაშლას ან შეცვლას. ამგვარად, ტრიგერების ერთ_ერთი მთავარი ფუნქციაა სისტემაში მონაცემთა ცვლილებების სტატისტიკის წარმოება.



სურათი 1.1

სურათი 1.1 ზე მოყვანილია ტრიგერის მაგალითი, რომელიც DocLocalUnit- ცხრილის კოპიას გააკეთებს მაშინ როცა ამ ცხრილში მონაცემის რაიმე განახლებას ექნება ადგილი.

მართვის ამოცანებში ამას დიდი მნიშვნელობა აქვს. მაგალითად, შესაძლებელია მონაცემთა ცვლილებები დამუშავდეს მათი მნიშვნელობების საკონტროლო მაჩვენებლებთან,

რომელთა მიღწევის შემთხვევაში სისტემა სასწრაფოდ გააფრთხილებს მომხმარებელს მოცემულ სიტუაციაში შესაძლო გადაწყვეტილებების შესახებ.

მონაცემთა ბაზების მართვის სისტემაში მთლიანობის დაცვის მექანიზმის სახით, ზოგადად გამოიყენება:

Data -> Stored Procedure.

მონაცემთა ბაზის მთლიანობა გულისხობს მასში არსებული მონაცემების დაცვას შეცდომითი (არასწორი) ცვლილებებისაგან. მონაცემთა ბაზის მთლიანობის მხარდაჭერა მდგომარეობს იმაში,რომ უზრუნველყოს როგორც თვით მონაცემთა ყველა ელემენტის მნიშვნელობა, ასევე მათი ურთიერთქმედაბის კორექტულობა(ჭეშმარიტება) დროის ნებისმიერ მომენტში.

აღდგენა

მონაცემთა ბაზაში არსებული მონაცემები უნდა იყოს მდგრადი არასასურველი ფიზიკური ზემოქმედებისა და პროგრამულ უზრუნველყოფაში შესაძლო შეცდომებისადმი. ამიტომ აუცილებლად უნდა იყოს გათვალისწინებული მონაცემთა ბაზის იმ დგომარეობის აღდგენის მექანიზმები, რომელიც მას ჰქონდა დაზიანებამდე.

მართვის საკითხები წვდომის შესახებ და მონაცემთა ბაზის მთლიანობის შენარჩუნება უკავშირდებიან ერთმანეთს, და ხშირ შემთხვევაში მათი ამოხსინასათვის გამოიყენებენ ერთსადა იმავე მექანიზმებს.

მონაცემთა მთლიანობის მართვა

მონაცემთა მთლიანობის დარღვევა შეაძლება გამოწვეული იყოს შემდეგი მიზეზებით:

- ხელსაწყოების წარუმატებლობა, ფიზიკური ზემოქმედებები ;
- მონაცემთა ბაზის მართვის სისტემისა და ოპერატიული სისტეის პროგრამული შეცდომები;
- შეცდომები გამოყენებით პროგრამებში;

- მომხარებლების მიერ კონფლიქტური საკითხების ერთობლივი შესრულება და ა.შ.
- მონაცემთა მთლიანობის დარღვევა ასევე შესაძლოა კარგად დამუშავებული სისტემებშიც კი. ამიტომ მნიშვნელოვანია არამარტო მთლიანობის დარღვევის თავიდან აცილება,არამედ დარღვევის ფაქტის დროული აღმოჩენა და ასევე მთლიანობის ოპერატიული აღდგენა მისი დარღვევის შემდეგ.

პარალელიზმით მართვა

მთლიანობის შენარჩუნება ზემოდ აღნიშნული პარამეტრების მიხედვით წარმოადგენს საკმაოდ რთულ პრობლემას მონაცემთა ბაზის სისტემაში. სისტემებში, რომლებიც ორიენტირებულია მრავალი მომხმარებლის მუშაობის რეჯიმზე, ჩნდება რიგი ახალი პრობლემებისა, რომლებიც დაკავშირებულია მომხმარებელთა კონფლიქტური მოთხოვნების პარალელურ შესრულებასთან .

მონაცემთა ბაზის მთლიანობის დარღვევა შესაძლებელია მათზე ერთდროული ოპერაციების შესრულებისას, ამიტომ უნდა იყოს გათვალისწინებული მონაცემთა მართვის მექანიზმები,რომლების უზრუნველყოფენ მონაცემთა ბაზის მთლიანობას რამდენიმე ოპერაციის შერსულებისას.

მონაცემთა მთლიანობის დაცვის მექანიზმის მნიშვნელოვან საშუალებას წარმოადგენს ოპერაციათა გაერთიანება, რის შედეგადაც მონაცემთა ბაზა მთლიანობის ერთი მდგომარეობიდან მეორე მდგომარეობაში გადადის, მუშაობის ერთ ლოგიკურ ელემენტში, რომელსაც ტრანზაქცია ეწოდება. ტრანზაქციის მექანიზმის არსი იმაში მდგომარეობს , რომ ტრანზაქციის დასრულებამდე მონაცემთა ყველა მაიპულაცია მიმდინარეობს მონაცემთა ბაზის გარე, ხოლო რეალური ცვლლილებების შეტანა მონაცემთა ბაზაში ხდება მხოლოდ ტრანზაქციის ნორმალური დასრულების შემდგომ.

მონაცემთა უსაფრთხოების თვალსაზრისით მონაცემთა ბაზაში ცვლილებების ასახვის ასეთი მექანიზმი ფრიად მნიშვნელოვანია. თუ ტრანზაქცია იქნა შეწყვეტილი,მაშინ

მონაცემთა ბაზის მართვის სისტემის სპეციალური ჩანართი საშუალებები ასრულებენ ეგრეთწოდებულ მონაცემთა ბაზის დაბრუნებას იმ მდგომარეობამდე, რომელიც ტრანზაქციის წინამორბედია. თუ ტრანზაქციის ერთი შესრულება არ არღვევს მონაცემთა ბაზის მთლიანობას, მაშინ ტრანზაქციის რამდენიმე ერთდროული შესრულებისას მონაცემთა ბაზის მთლიანობა შეიძლება დაირღვეს. ამგვარი შეცდომების თავიდან ასაცილებლად, მონაცემთა ბაზის მართვის სისტემამ უნდა შეინარჩუნოს ის მექანიზმები,რომლებიც უზრუნველყოფენ ეგრედწოდებულ ბლოკირებას. ამასთან გარანტირებულია, რომ ვერავინ ვერ მიიღებს წვდომას მონაცემთა მოდიფიცირებულ ელემენტებზე, სანამ ტრანზაქცია არ გაანთავისუფლებს მას. ბლოკირების მექანიზმის გამოყენებას მოჰყვება პარალელიზმის მართვის ახალი პრობლემები, იქმნება ორი ტრანზაქციის სიტუაცია. ამასთან, თუ ერთი ტრანზაქცია ცდილობს დაბლოკოს ობიექტი რომელიც უკვე დაბლოკილია მეორე ტრანზაქციით, მაშინ მასოუწევს ცდა სანამ არ იქნება მოხსნილი ამ ობიექტის ბლოკირება მეორე ტრანზაქციის მიერ. ეს ნიშნავს იმას,რომ ობიექტის ბლოკირება შეუძლია მხოლოდ ერთ ტრანზაქციას.

წვდომის მართვა მონაცემთა ბაზაში.

მონაცემთა ბაზის სისტემის უმეტესობა წარმოადგენს მონაცემთა საერთო ცენტრალიზებული შენახვის საშუალებას. ეს საკმაოდ ამცირებს მონაცემთა მოცულობას, ადვილს ხდის მონაცემთა წვდომას და მონაცემთა ეფექტური დაცვის შესაძლებლობას იძლევა. მაგრამ მონაცემტა ბაზის ტექნოლოგიაში იქმნება რიგი პრობლემებისა, რომლებიც დაკავშირებულია, მაგალითად, იმასთან, რომ სხვადასხვა მომხმარებლებს ერთი მონაცემების წვდომა უნდა ჰქონდეთ და არ ჰქონდეთ სხვა მონაცემების წვდომა. ამიტომ მონაცემთა ბაზის წვდომის საიმედო განსხვავება ფაქტიურად შეუძლებელია სპეციალური საშუალებებისა და მეთოდების გარეშე.

თანამედროვე მონაცემთა ბაზის მართვის სისტემას ძირითადად გააჩნია ჩანართი საშუალებები,რომლებიც სისტემის ადმინისტრატორს აძლევს საშუალებას განსაზღვროს მომხმარებლის უფლებები მანეცემთა ბაზის ნებისმიერი ნაწილების წვდომისას. ამასთან შესაძლებელია არამარტო წვდომის მინიჭება ამა თუ იმ მომხმარებლისათვის,არამედ

შესაძლებელია წვდომის ტიპების განსაზღვრა: რისი გაკეთება შეუძლია კონკრეტულ მომხმარებელს კონკრეტული მონაცემებთან (წაკითხვა, მოდიფიცირება, წაშლა და ა.შ).

საიმედოობის უზრუნველყოფა

მონაცემთა ყოველ ელემენტში ინფორმაცია შეიტანება ამ ელემენტის აღწერის შესაბამისად.

მონაცემთა ბაზები კოლექტიური მოხმარების რესურსია, ამიტომაც მათ მრავალი ელემენტი იყენებს. როგორც წესი, ეს ელემენტები განსხვავდება თავიანთი სტატუსებით. ყველას არა აქვს ყველაფრის მიღების უფლება. ამიტომაც მონაცემთა ბაზის ადმინისტრატორი ქმნის სპეციალურ საპაროლო სისტემას (ჟურნალს), რომელშიც ჩაწერილია თითოეული მომხმარებლის ვინაობა, სტატუსი და მონაცემთა გამოყენების უფლებები.

პაროლის გარეშე მომხმარებელი ვერ შეუერთდება მონაცემთა ბაზას. სტატუსში იგულისხმება მომხმარებელთა კლასიფიკაცია. მაგალითად, კლიენტები, რომელთაც აქვთ მონაცემების მხოლოდ წაკითხვის (Read Only) უფლება. მათ არ შეუძლიათ შეიტანონ ახალი ან ჩაასწორონ მონაცემები ბაზაში.

უფლებამოსილების გამიჯვნის წესები

ხშირად, მონაცემთა ბაზასთან რამდენიმე მომხმარებელი მუშაობს. ამ დროს აუცილებელია მასთან მიმართვის უფლებების გამიჯვნა. ეს მოითხოვს უსაფრთხოების სისტემის სწორად დაგეგმვას. დაგეგმვის პროცესში უნდა განისაზღვროს ის მომხმარებლები, რომლებსაც ექნებათ მონაცემთა ბაზასთან მიმართვის უფლება და ის მოქმედებები, რომელთა შესრულების უფლება ექნებათ მათ.

უსაფრთხოების სისტემის პირველი ეტაპი მდგომარებს იმ მომხმარებლების განსაზღვრაში, რომლებსაც ექნებათ მონაცემთა ბაზასთან მიმართვის უფლება. მაგრამ, ჯერ მომხმარებლებს უნდა ჰქონდეთ სერვერთან მიმართვის უფლება. ამისათვის, შეგვიძლია გამოვიყენოთ სერვერის ან Windows NT-ის უსაფრთხოების სისტემა. პირველ შემთხვევაში

მომხმარებელი უნდა დავარეგისტრიროთ სერვერზე, ანუ მისთვის უნდა შევქმნათ სააღრიცხვო ჩანაწერი. მეორე შემთხვევაში, მომხმარებელი უნდა დავარეგისტრიროთ Windows NT სისტემაში, ანუ მისთვის შევქმნათ სააღრიცხვო ჩანაწერი დომენში. უნდა გვახსოვდეს, რომ თუ მომხმარებელს აქვს სერვერთან მიმართვის უფლება, ეს არ ნიშნავს იმას, რომ მას აქვს რომელიმე მონაცემთა ბაზასთან ან მის ობიექტთან მიმართვის უფლებაც.

უსაფრთხოების სისტემის დაგეგმვის მეორე ეტაპი მდგომარეობს იმ მოქმედებების განსაზღვრაში, რომელთა შესრულების უფლებაც აქვს კონკრეტულ მომხმარებელს მონაცემთა ბაზაში. მონაცემთა ბაზასთან და მის ობიექტებთან სრული მიმართვის უფლება აქვს ადმინისტრატორს. მეორე ადამიანი ადმინისტრატორის შემდეგ არის ობიექტის მფლობელი. მონაცემთა ბაზაში ობიექტს შექმნის დროს ენიშნება მფლობელი, რომელსაც შეუძლია განსაზღვროს ამ ობიექტთან მიმართვის უფლებები. მომხმარებლების მესამე კატეგორიას აქვს მიმართვის ის უფლებები, რომლებიც მათ მისცა ადმინისტრატორმა ან ობიექტის მფლობელმა.

მომხმარებლებისათვის მისაცემი უფლებები ჭკვიანურად უნდა დავგეგმოთ. მაგალითად, არ არის აუცილებებლი, რომ ფირმის დირექტორს ან რიგით თანამშრომელს მივცეთ ხელფასების შესახებ მონაცემების შემცველ ცხრილში მონაცემების შეცვლის უფლება. მათ შეგვიძლია მივცეთ ახალი ინფორმაციის ჩამატების ნებართვა. არასწორი მონაცემების შეტანის შემთხვევაში ფირმა სერიოზულად არ დაზარალდება. მაგრამ, თუ მათ მივცემთ მონაცემების შეცვლის ან წაშლის უფლებას, მაშინ ამან ფირმას შეიძლება დიდი ფინანსური ზიანი მიაყენოს.

სერვერის უსაფრთხოების სისტემის არქიტექტურა

სერვერის უსაფრთხოების სისტემა ემყარება მომხმარებლებსა და სააღრიცხვო ჩანაწერებს. მომხმარებლები გადიან შემოწმების ორ ეტაპს. პირველ ეტაპზე ხდება მომხმარებლის იდენტიფიცირება სააღრიცხვო ჩანაწერის სახელისა და პაროლის მიხედვით, ე.ი. აუტენტიფიცირება. თუ სახელი და პაროლი სწორადაა შეტანილი, მაშინ მომხმარებელი მიეერთდება სერვერს. სერვერთან მიერთება ანუ დარეგისტრირება მომხმარებელს არ

აძლევს მონაცემთა ბაზასთან ავტომატურად მიმართვის უფლებას. თითოეული მონაცემთა ბაზისთვის სარეგისტრაციო სახელი (სააღრიცხვო ჩანაწერი, login) უნდა აისახოს მონაცემთა ბაზის მომხმარებლის სახელში (user). მეორე ეტაპზე, მომხმარებლისთვის (როგორც მონაცემთა ბაზის მომხმარებლისათვის) გაცემული უფლებების საფუძველზე, მისი სარეგისტრაციო სახელი (login) შესაბამის მონაცემთა ბაზასთან მიმართვის უფლებას იღებს. სხვადასხვა მონაცემთა ბაზაში ერთი და იგივე მომხმარებლის სარეგისტრაციო სახელს შეიძლება ჰქონდეს ერთნაირი ან სხვადასხვა სახელები (მომხმარებლების სახელები) მიმართვის სხვადასხვა უფლებებით.

მონაცემთა ბაზებთან მიმართვისათვის პროგრამა-დანართებსაც უნდა ჰქონდეთ შესაბამისი უფლებები. ჩვეულებრივ, პროგრამა-დანართებს ისეთივე უფლებები ეძლევათ, როგორიც აქვთ მათ გამშვებ მომხმარებლებს. მაგრამ, ზოგიერთი პროგრამა-დანართის მუშაობისთვის საჭიროა მიმართვის უფლებების ისეთი ნაკრები, რომელიც არ არის დამოკიდებული მომხმარებლის მიმართვის უფლებებზე. ასეთი უფლებების მინიჭება შესაძლებელია როლების (ჯგუფების) გამოყენებით.

სერვერის დონეზე უსაფრთხოების სისტემის განხილვისას შემდეგი ცნებები გამოიყენება:

- აუტენტიფიცირება (authentication);
- სააღრიცხვო ჩანაწერი (login);
- სერვერის ფიქსირებული როლები (fixed server roles).

მონაცემთა ბაზის დონეზე უსაფრთხოების სისტემის განხილვისას შემდეგი ცნებები გამოიყენება:

- მონაცემთა ბაზის მომხმარებელი (database user);
- მონაცემთა ბაზის ფიქსირებული როლი (fixed database role);
- მონაცემთა ბაზის მომხმარებლის როლი (users database role);
- პროგრამა-დანართის როლი (application role).

უსაფრთხოების სტრუქტურის ელემენტები

სერვერის უსაფრთხოების სისტემის ელემენტებია: სააღრიცხვო ჩანაწერები (logn), მომხმარებლები (user), როლები (role) და ჯგუფები (group).

მომხმარებელმა, რომელიც სერვერს უერთდება, უნდა მოახდინოს თავისი თავის აუტენტიფიცირება სააღრიცხვო ჩანაწერის გამოყენებით. აუტენტიფიცირების წარმატებით გავლის შემდეგ მას შეეძლება სერვერთან მიმართვა. მონაცემთა ბაზასთან მიმართვისათვის მომხმარებლის სააღრიცხვო ჩანაწერი (login) აისახება მონაცემთა ბაზის მომხმარებელში (user). მონაცემთა ბაზის მომხმარებელს შეუძლია მიმართოს ამ მონაცემთა ბაზის ნებისმიერ ობიექტს: ცხრილს, წარმოდგენას, შენახულ პროცედურას და ა.შ.

სააღრიცხვო ჩანაწერის ასეთი ასახვა მონაცემთა ბაზის მომხმარებელში აუცილებელია თითოეული მონაცემთა ბაზისთვის, რომელთანაც მიმართვა სურს მომხმარებელს. ასახვები ინახება მიმდინარე მონაცემთა ბაზის sysusers სისტემურ წარმოდგენაში. ასეთი მიდგომა უზრუნველყოფს უსაფრთხოების მაღალ დონეს, რადგან თავიდან ვიცილებთ მომხმარებლის ავტომატურ მიმართვას სხვა მონაცემთა ბაზებთან. მონაცემთა ბაზების მომხმარებლები, თავის მხრივ, შეგვიძლია გავაერთიანოთ ჯგუფებში ან როლებში უსაფრთხოების სისტემის მართვის გასამარტივებლად.

იმ შემთხვევაში, თუ სააღრიცხვო ჩანაწერი არ აისახება მონაცემთა ბაზის მომხმარებელში, მაშინ მომხმარებელი მაინც შეძლებს მონაცემთა ბაზასთან მიმართვას guest სტუმრის სახელით, თუ ეს მომხმარებელი, ბუნებრივია, არსებობს მონაცემთა ბაზაში. ჩვეულებრივ, guest მომხმარებელს ეძლევა მინიმალური უფლებები მხოლოდ წაკითხვის რეჟიმში.

სქემები

სქემა არის მასში შემავალი ობიექტების კოლექცია, რომელიც ერთ სახელების სივრცეს ქმნის. სახელების სივრცე არის სიმრავლე, რომლის თითოეულ ელემენტს უნიკალური სახელი აქვს. სქემა შეიძლება შეიცავდეს ცხრილებს, წარმოდგენებს, ფუნქციებს, პროცედურებს, ტრიგერებს, ინდექსებსა და მომხმარებლებს. ერთი სქემის შიგნით ობიექტებს სხვადასხვა სახელები უნდა ჰქონდეს.

მონაცემთა ბაზის შექმნის დროს იქმნება სტანდარტული სქემები. ერთ-ერთი მათგანია dbo. თუ ობიექტის შექმნის დროს ან ობიექტთან მიმართვის დროს სქემის სახელი არ არის მითითებული, მაშინ ავტომატურად იგულისხმება dbo სქემა. ჩვენ შეგვიძლია შევქმნათ სქემები და მასში მოვათავსოთ სხვადასხვა ობიექტები.

მომხმარებლები

მას შემდეგ, რაც მომხმარებელმა წარმატებით გაიარა აუტენტიფიცირება და მიიღო სააღრიცხვო ჩანაწერის იდენტიფიკატორი (login ID), ის ითვლება დარეგისტრირებულად და მას ეძლევა სერვერთან მიმართვის უფლება. მომხმარებლის სააღრიცხვო ჩანაწერი (login) ასოცირდება კონკრეტული მონაცემთა ბაზის მომხმარებელთან (user). მომხმარებლები წარმოადგენენ სერვერის სპეციალურ ობიექტებს, რომელთა საშუალებით განისაზღვრება მიმართვის უფლებები და მონაცემთა ბაზაში ობიქტების მფლობელობა. მომხმარებლის სახელი შეიძლება გამოყენებული იყოს მიმართვის უფლების მისაცემად როგორც ერთი ადამიანისთვის, ისე ადამიანების ჯგუფისთვის.

მონაცემთა ბაზის შექმნის დროს განისაზღვრება ორი სტანდარტული მომხმარებელი: dbo და guest.

თუ სააღრიცხვო ჩანაწერი აშკარად არ არის დაკავშირებული მომხმარებელთან მაშინ მომხმარებელს ეძლევა მონაცემთა ბაზებთან არააშკარა მიმართვის უფლება guest სტუმრის სახელის გამოყენებით. ე.ი. ასეთი სააღრიცხვო ჩანაწერი, რომელმაც მიიღო სერვერთან მიმართვის უფლება, ავტომატურად აისახება guest მომხმარებელში ყველა მონაცემთა ბაზისთვის. თუ ჩვენ მონაცემთა ბაზიდან წავშლით guest მომხმარებელს, მაშინ სააღრიცხვო ჩანაწერები, რომლებსაც არ აქვთ აშკარა ასახვა მომხმარებლის სახელში, ვერ შეძლებენ მონაცემთა ბაზასთან მიმართვას. უნდა აღვნიშნოთ რომ guest მომხმარებელს არ აქვს ავტომატური მიმართვის უფლება მონაცემთა ბაზებთან. მონაცემთა ბაზის მფლობელმა თვითონ უნდა გადაწყვიტოს მისცეს თუ არა guest მომხმარებელს ეს უფლება.

მაქსიმალური უსაფრთხოების უზრუნველყოფისათვის შეგვიძლია წავშალოთ guest მომხმარებელი ყველა მონაცემთა ბაზიდან, გარდა master და tempdb სისტემური ბაზებიდან.
მონაცემთა ბაზის მფლობელი (DataBase Owner, dbo) - სპეციალური მომხმარებელია, რომელსაც მაქსიმალური უფლებები აქვს მონაცემთა ბაზაში. sysadmin როლის ნებისმიერი წევრი ავტომატურად აისახება dbo მომხმარებელში. თუ მომხმარებელი, რომელიც არის sysadmin როლის წევრი, ქმნის რაიმე ობიექტს, მაშინ ამ ობიექტის მფლობელად დაინიშნება არა ეს მომხმარებელი, არამედ dbo. მაგალითად, თუ Saba მომხმარებელი, რომელიც არის ადმინისტრაციული ჯგუფის წევრი, ქმნის Cxrili_1 ცხრილს, მაშინ ცხრილის სრული სახელი იქნება არა Saba.Cxrili_1, არამედ dbo.Cxrili_1. ამავე დროს, თუ Saba არის არა სერვერის

sysadmin როლის წევრი, არამედ მონაცემთა ბაზის მფლობელის db_owner როლის წევრი, მაშინ ცხრილის სახელი იქნება Saba.Cxrili_1.

dbo მომხმარებლის წაშლა არ შეიძლება.

სერვერის როლები

როლი საუალებას გვაძლევს გავაერთიანოთ მომხმარებლები, რომლებიც ერთნაირ ფუნქციებს ასრულებენ. ეს საჭიროა სერვერის უსაფრთხოების სისტემის ადმინისტრირების გასამარტივებლად.

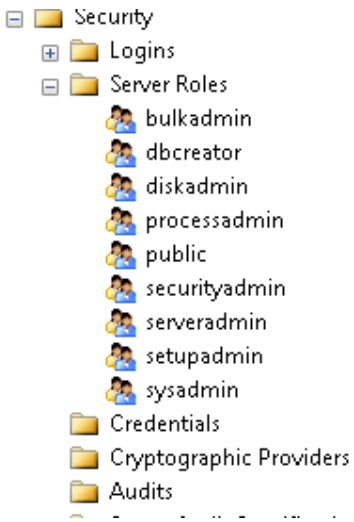
სერვერზე რეალიზებულია ორი სტანდარტული როლი: სერვერის დონეზე და მონაცემთა ბაზის დონეზე. სერვერის დაყენების დროს იქმნება სერვერის ცხრა ფიქსირებული როლი და მონაცემთა ბაზის ცხრა ფიქსირებული როლი. ამ როლებს ჩვენ ვერ წავშლით და არ შეიძლება მათი უფლებების მოდიფიცირება. იმისათვის, რომ მომხმარებელს მივანიჭოთ ის უფლებები, რომელიც აქვს სერვერის რომელიმე ფიქსირებულ როლს, საჭიროა მომხმარებლის ამ როლში ჩართვა.

სერვერის როლების (server role) გამოყენებით სერვერის ოპერატორებს ენიჭებათ მხოლოდ ის უფლებები, რომლებსაც ადმინისტრატორი ჩათვლის საჭიროდ. სერვერის ნებისმიერ როლში შეგვიძლია ჩავრთოთ სერვერის ან Windows NT-ის სააღრიცხვო ჩანაწერი.
სერვერის სტანდარტული როლები (fixed server role) და მათი უფლებები მოყვანილია ცხრილში 2.1.

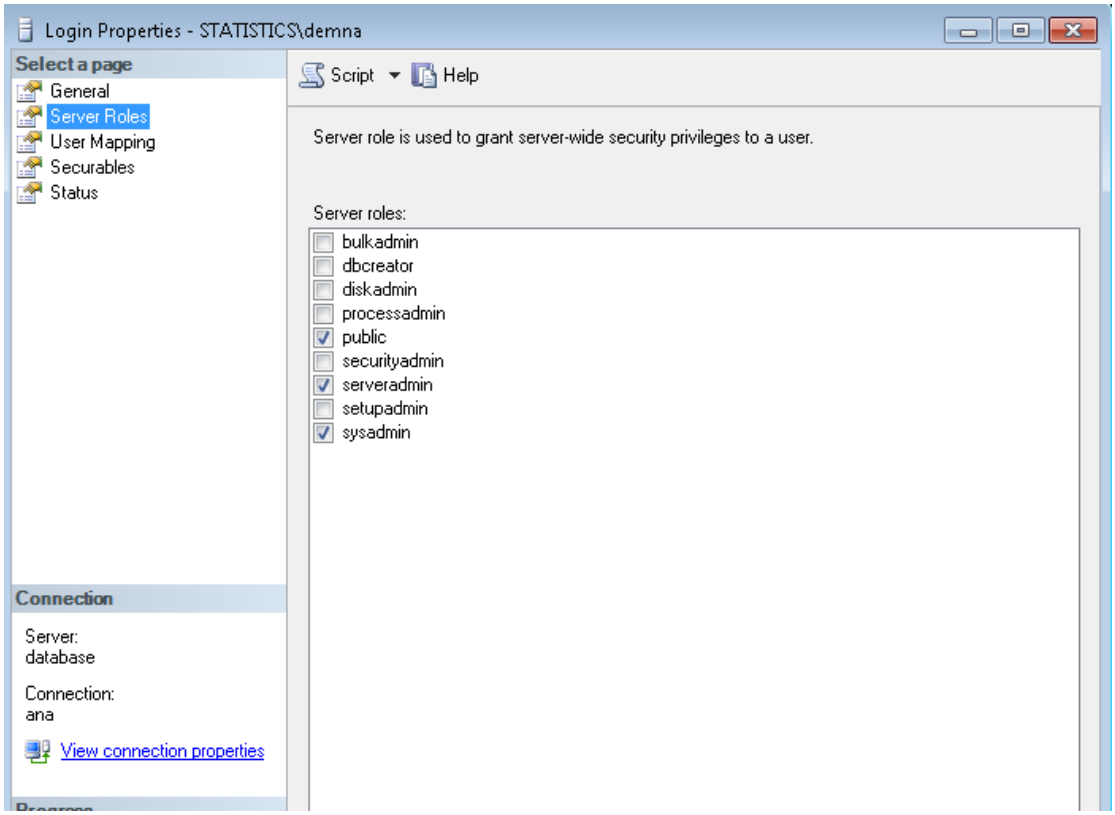
ცხრილი 2.1.

Sysadmin	შეუძლია ნებისმიერი მოქმედების შესრულება სერვერზე
Serveradmin	ასრულებს სერვერის კონფიგურირებასა და გამორთვას
Setupadmin	მართავს ბმულ სერვერებსა და პროცედურებს, რომლებიც ავტომატურად გაიშვება სერვერის გაშვებისას
Securityadmin	მართავს სააღრიცხვო ჩანაწერებსა და მონაცემთა ბაზის შექმნის წესებს, ასევე, შეუძლია შეცდომე-ბის ჟურნალის წაკითხვა
Processadmin	მართავს სერვერის მიერ გაშვებულ პროცესებს

Dbcreator	შეუძლია მონაცემთა ბაზების შექმნა და მოდიფი-ცირება
Diskadmin	მართავს სერვერის ფაილებს
Bulkadmin (Bulk Insert administrators)	შეუძლია ჩასვას მონაცემები მასობრივი გადაწე-რის საშუალებების გამოყენებით



სურათი 1.5



სურათი 1.6

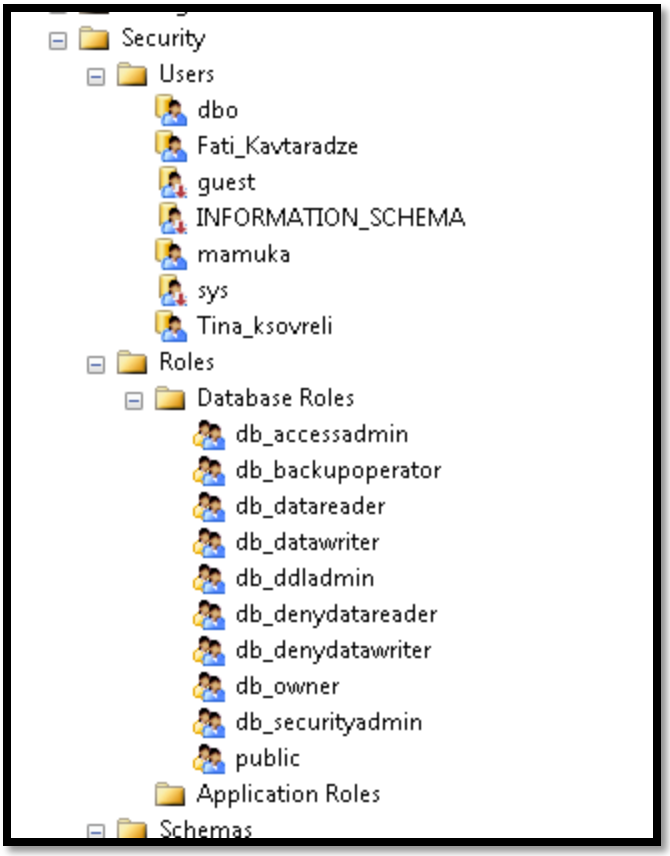
სურათი 1.5 ზე ნაჩვენებია სერვერის როლები, ხოლო 1.6 სურათზე ის როლები რომელიშიც გაერთიანებულია STATISTICS/Demna იუზერი

მონაცემთა ბაზების როლები

მონაცემთა ბაზების როლები (database role) საშუალებას გვაძლევს მომხმარებლები გავაერთიანოთ ერთ ადმინისტრაციულ ერთეულში და მასთან ვიმუშაოთ როგორც ჩვეულებრივ მომხმარებელთან. განვსაზღვრავთ რა მონაცემთა ბაზის ობიექტებთან მიმართვის უფლებებს კონკრეტული როლისთვის, ჩვენ ამით ავტომატურად ვაძლევთ იგივე უფლებებს ამ როლის ყველა წევრს. თუ მომხმარებელს უნდა მიენიჭოს იგივე უფლებები, როგორიც მინიჭებული აქვს როლს, მაშინ ეს მომხმარებელი ამ როლში უნდა ჩავრთოთ.

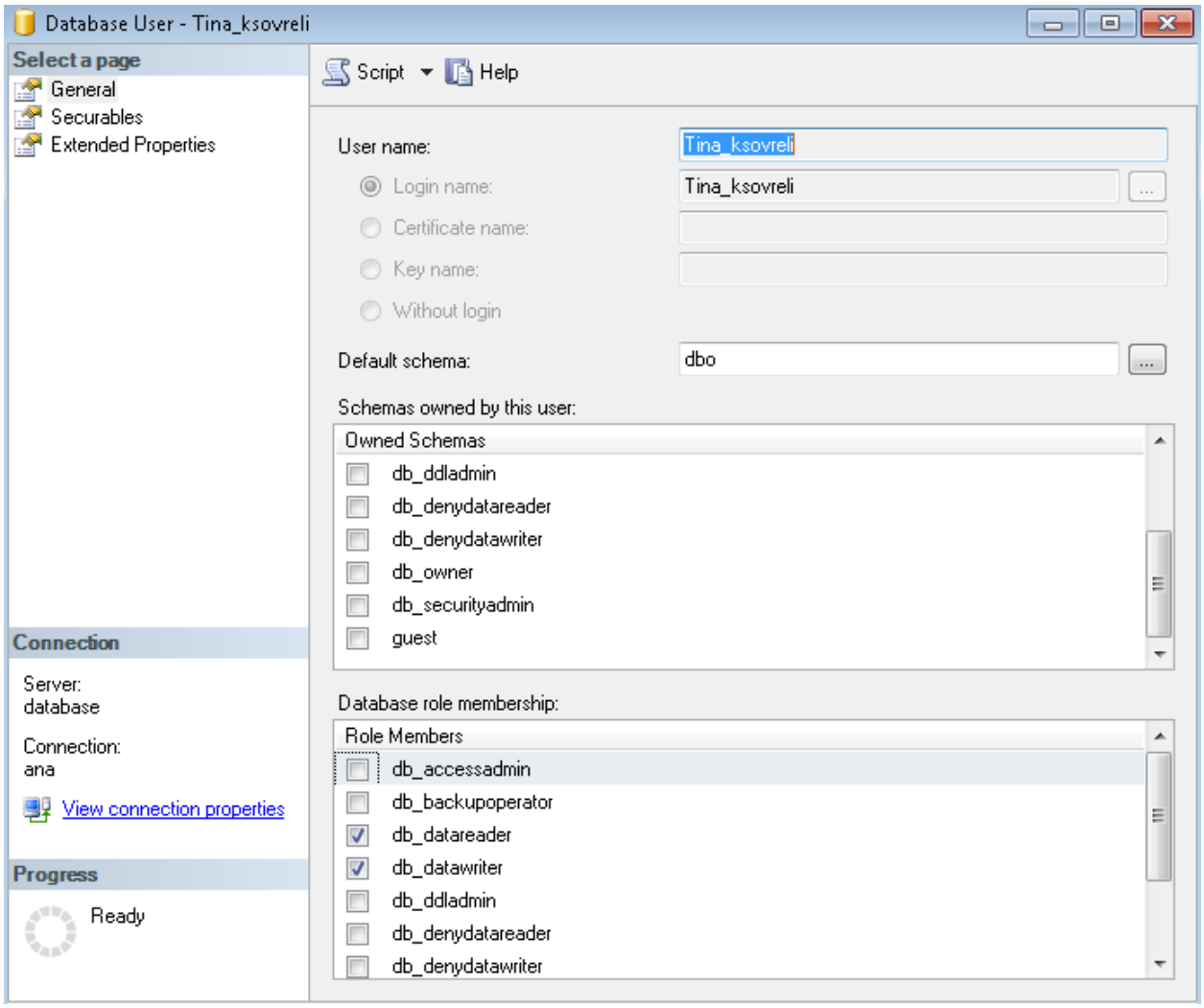
მონაცემთა ბაზის სტანდარტული როლები (fixed database role) და მათი უფლებები მოყვანილია ცხრილში 1.2.

db_owner	აქვს ყველა უფლება მონაცემთა ბაზაში
db_accessadmin	შეუძლია მომხმარებლების დამატება და წაშლა
db_securityadmin	მართავს ყველა ნებართვას, ობიექტს, როლს და როლების წევრებს
db_ddladmin	შეუძლია DDL ბრძანებების შესრულება, გარდა GRANT, DENY და REVOKE ბრძანებებისა
db_backupoperator	შეუძლია DBCC, CHECKPOINT და BACKUP ბრძანებების შესრულება
db_datareader	შეუძლია ნახოს ნებისმიერი მონაცემები მონაცემ-თა ბაზის ნებისმიერ ცხრილში
db_datawriter	შეუძლია შეცვალოს ნებისმიერი მონაცემები მონა-ცემთა ბაზის ნებისმიერ ცხრილში
db_denydatareader	ეკრძალება ნახოს ნებისმიერი მონაცემები მონა-ცემთა ბაზის ნებისმიერ ცხრილში
db_denydatawriter	ეკრძალება შეცვალოს ნებისმიერი მონაცემები მონაცემთა ბაზის ნებისმიერ ცხრილში



მოცემულ სურათზე ნაჩვენებია მონაცემთა ბაზის როლები და ის იუზერები რომელთაც ამ მონაცემთა ბაზაზე აქვთ გარკვეული წვდომის უფლებები.

სურათი 1.4 ზე ნაჩვენებია რომ იუზერი რომლის სახელიცაა Tina_ksovreli არის db_datareader და db_datawriter როლის წევრი.



სისტემაში შესვლის პროცედურა

მომხმარებლებმა უნდა წარადგინონ საკუთარი თავი - ავტორიზაციის პროცედურა

იმისათვის, რომ მოიცვას სისტემის უსაფრთხოება, DBMS-ს სჭირდება რომ იცოდეს კონკრეტულად ვინ იყენებს მონაცემთა ბაზას დროის ნებისმიერ მონაკვეთში. ამისათვის, თითქმის ყველა კომერციული SQL DBMS ეყრდნობა ავტორიზაციის ID-ს კონცეპციას. ავტორიზაციის ID არის ნიშანი, რომლის დახმარებითაც SQL ახდენს იმ პირების იდენტიფიკაციას, რომლებსაც აქვთ უფლება მიმართონ ბრძანებებით DBMS-ს (უნდა

აღინიშნოს რომ, მომხმარებელთა ჯგუფს შესაძლოა ჰქონდეს მსგავსი მოთხოვნები და შესაძლოა შესაბამისად იზიარებდნენ ავტორიზაციის ID-ს, მაგრამ ეს არ ხდება ხშირად.როგორც წესი, თითოეულ მომხმარებელს აქვს თავისი პირადი ავტორიზაციის ID). ავტორიზაციის ID შესაძლებელია აგრეთვე გამოყენებულ იქნას იმისათვის, რომ მოხდეს იმის გარჩევა თუ ვინ ახდენს მიმართვას SQL ბრძანებებით - პროგრამა თუ პიროვნება.

SQL რეკომენდაცია

SQL სერვერს და Sybase-ს აქვთ ჯგუფური ID-ის მხარდაჭერა, რომელიც შესაძლოა გამოვიყენოთ მსგავსი საჭიროებების მქონე მომხმარებელთა ჯგუფის ინდენტიფიკაციისთვის.

სისტემაში ახალი მომხმარებლების რეგისტრაცია ხდება მონაცემთა ბაზის ადმინისტრატორის მიერ, რომელმაც უნდა მიაწოდოს ინფორმაცია DBMS-ს, რომ ახალი მომხმარებლის ავტორიზაციის ID და პაროლი დაამატოს აქტიური მომხმარებლების სიას.

ზოგიერთი კომერციული SQL სტრუქტურა, რომელიც მოიცავს Ingres და Informix იყენებს მომხმარებლის სახელს რომელიც განსაზღვრულია მასპინძელი კომპიუტერის სისტემაში შესვლის პროცედურით, როგორც ამ მომხმარებლის ავტორიზაციის ID. სხვა სისტემები, მათ შორის Oracle, მომხმარებლისგან მოითხოვს მომხმარებლის სახელისა და აგრეთვე ასოცირებული პაროლის განსაზღვრას ინტერაქციული SQL სესიის დასაწყისში. მომხმარებლის სახელი SQL-ში გამოიყენება როგორც ავტორიზაციის ID, მაგრამ პაროლი არ გამოიყენება.

SQL რეკომენდაცია

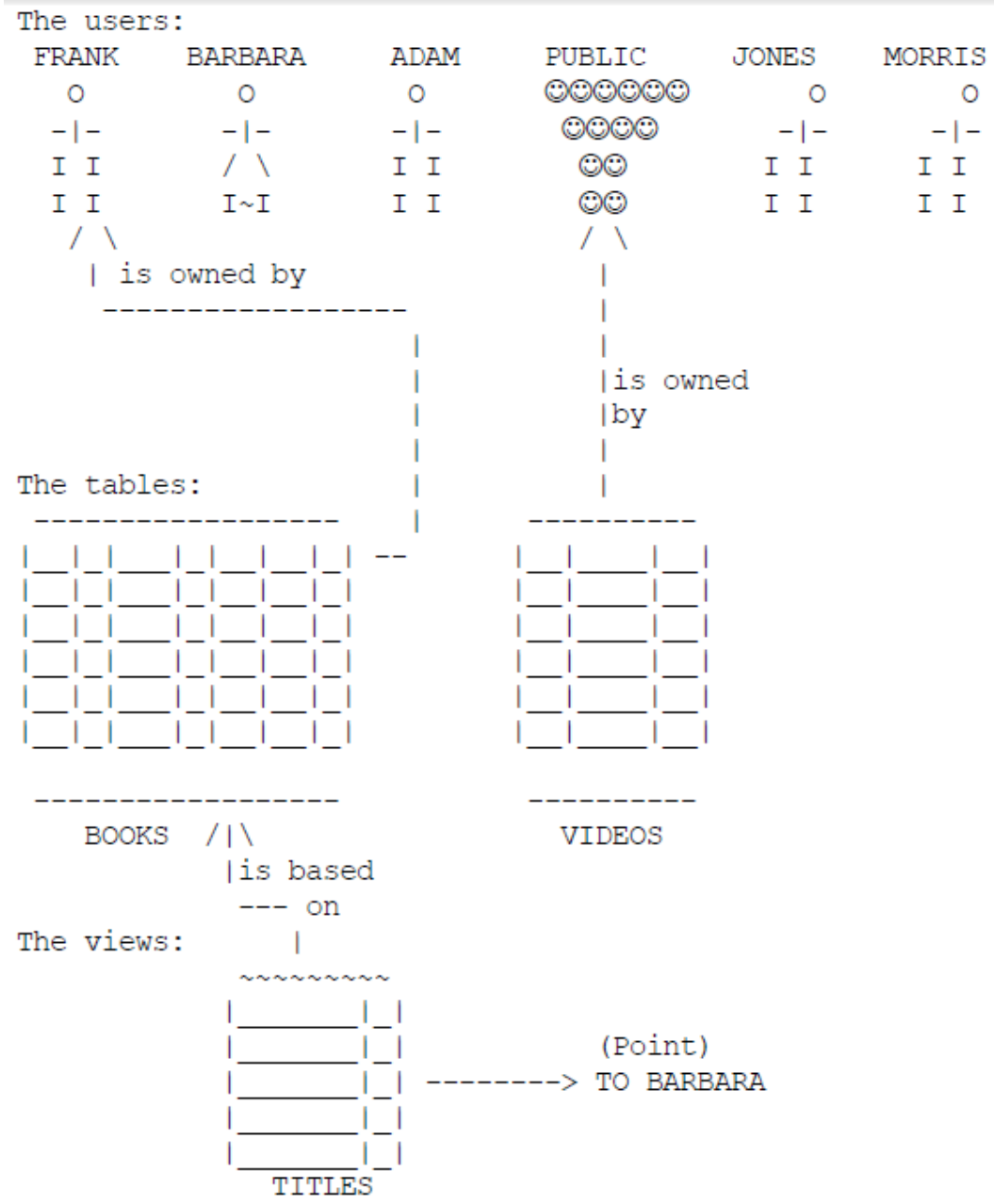
ANSI/ISO სტანდარტი იყენებს ტერმინს ავტორიზაციის ID, მომხმარებლის ID-ს ნაცვლად.

ბიბლიოთეკის მონაცემთა ბაზა

სისტემის მაგალითი.

ნებისმიერ ჩვენს მიერ შექმნილ ცხრილზე, SQL ვანიჭებთ ცხრილის მფლობელის უფლებებს. ცხრილის მფლობელობა ნიშნავს, რომ ჩქვენ ავტომატურად გვენიჭებათ ყველა პრივილეგია (უფლება) ამ ცხრილზე (ეს არის 4 სტანდარტული ANSI/ISO პრივილეგია; SELECT, INSERT, UPDATE, და DELETE). საბოლოოდ, მონაცემთა ბაზის ყველა სხვა მომხმარებელს არ ექნება პრივილეგია თქვენს ახალ ცხრილზე. სურათი 1.1 გვაჩვენებს

მომხმარებელთა ჯგუფს და ბიბლიოთეკის მონაცემთა ბაზას, რომელსაც ისინი იყენებენ.



სურათი 1.1

The Contents of Videos:

(‘Star Trek’, ’Universal’, ‘PG’)

(‘Duck Tales’, ‘Disney’, ‘U’)

The Contents of Books:

('English', 'U.K.Author', 'IN')

('French, 'E.E.C.Author', 'OUT')

The Contents of Titles View:

('German', 'Otto Matic')

SQL რეკომენდაცია

ANSI/ISO სტანდარტი უფლებას აძლევს ავტორიზაციის ID-ს, იყოს 18 სიმბოლომდე, მაგრამ ბევრი კომერციული სტრუქტურა არ მოქმედებს ამ წესით.

მომხმარებელი Frank შედის სისტემაში ავტორიზაციის ID-ით - FRANK და ქმნის ცხრილს სახელწოდებით BOOKS შემდეგი მეთოდით:

CREATE TABLE BOOKS (
TITLE CHAR(10),
AUTHOR CHAR(15),
STATUS CHAR(3)) ;
Table BOOKS successfully created (ცხრილის BOOKS შექმნა განხორციელდა წარმატებით).

FRANK ახლა არის BOOKS ცხრილის მფლობელი. სხვა მომმხარებლებს არ აქვთ ცხრილზე წვდომის უფლებები. თუ FRANK თავის ცხრილს დაამატებს მონაცემთა ორ სტრიქონს, მაშინ SQL მიიღებს ამ ბრძანებას, რადგან FRANK-ს აქვს INSERT პრივილეგია ამ ცხრილზე.

INSERT INTO books

VALUES ('English','U.K.Author', 'IN');

INSERT INTO books

VALUES ('French','E.E.C.Author', 'OUT');

2 rows successfully inserted (2 სტრიქონის ჩამატება განხორციელდა წარმატებით).

მაგრამ, თუ მომხმარებელი ADAM შეეცდება დაამატოს ახალი სტრიქონი, მაშინ SQL უარყოფს მის ბრძანებას რადგან მას არ აქვს შესაბამისი პრივილეგია.

INSERT INTO BOOKS

VALUES ('Spanish','E.S.Panya', 'OUT') ;

Error 136: User does not have INSERT Privileges (მომხმარებელს არ გააჩნია INSERT პრივილეგია).

ცხრილი VIDEOS მფლობელია PUBLIC (ეს არის სპეციალური ავტორიზაციის განმსაზღვრელი, რომელიც ნიშნავს რომ ყველა მომხმარებელი ფლობს მას) და VIEW TITLES მფლობელია ბარბარა. აღსანიშნავია, რომ ბარბარა არაა იმ ცხრილის (BOOKS) მფლობელი რომელზეც დაფუძნებულია VIEW TITLES.

SQL VIEW-ს შექმნის საშუალებას იმ შემთხვევაში იძლევა თუ თქვენ გაქვთ ბრძანება SELECT-ის გამოყენების პრივილეგია VIEW-ში გამოყენებულ ყველა ცხრილზე. შესაბამისად VIEW -ს მფლობელობა გულისხმობს მასზე SELECT-ის გამოყენების პრივილეგიას. სხვა ბრძანებების გამოყენების საშუალება გექნებათ თუ თქვენ გაგაჩნიათ ამ ბრძანების შესრულების პრივილეგია VIEW-ში გამოყენებულ ყველა ცხრილზე. ასე რომ თუ თქვენ ფლობთ INSERT ბრძანების შესრულების უფლებას ყველა ცხრილზე მაშინ თქვენ გაქვთ INSERT-ის პრივილეგია VIEW-ზე. ბიბლიოთეკის მონაცემთა ბაზაში მომხმარებელ BARBARA -ს BOOKS ცხრილზე მინიმუმ უნდა გააჩნდეს SELECT-ის და INSERT-ის შესრულების უფლება რათა შექმნას ამ ცხრილზე დაფუძნებული VIEW და ჩაამატოს ახალი სტრიქონი.

როგორ გადაიცემა პრივილეგიები

GRANT (მინიჭება) და REVOKE (გაუქმება) ბრძანებები

აქამდე ნახსენები SQL-ის ყველა ბრძანებიდან გამომდინარე ვასკვნით, რომ მომხმარებელი სარგებლობს წვდომით მხოლოდ იმ ცხრილებზე, რომელსაც ფლობს ან საჭირო პრივილეგია მინიჭებული აქვს ცხრილის მფლობელისაგან. მონაცემთა ბაზების მოქმედ სისტემაში, რეალურად, ძალიან მცირეა იმ მომხმარებელთა რიცხვი, რომლებიც ფლობენ აღნიშნულ ცხრილებს.

SQL რეკომენდაციები

ANSI/ISO სტანდარტი ცხრილებისათვის და ხედვისთვის ითვალისწინებს მხოლოდ 4 პრივილეგიას (უფლებას): SELECT (არჩევა), INSERT (ჩასმა), UPDATE (განახლება) და DELETE (წაშლა).

მონაცემთა ბაზების მომხმარებლებს ენიჭებათ წვდომა ცხრილებზე, ხედვაზე (ერთობლივად - მონაცემთა ბაზების ობიექტები) და სვეტებზე GRANT ბრძანებით. რაც გახლავთ SQL DDL-ის (მონაცემთა დეფინიციის ენა) და ANSI/ISO სტანდარტების ნაწილი. GRANT-ის საპირისპირო დატვირთვა აქვს ბრძანებას REVOKE. GRANT ბრძანებით უზრუნველყოფილი პრივილეგიები ანულირდება ბრძანებით REVOKE. REVOKE არ წარმოადგენს ANSI/ISO სტანდარტის ნაწილს, თუმცა ფართო გამოყენების მოპოვების შედეგად, თითქმის უკვე დე ფაქტო სტანდარტად მიიჩნევა. GRANT და REVOKE ბრძანებათა სინტაქსი წარმოდგენილია 1.2 და 1.3 სურათებზე.

```
GRANT rights ON tbl_name TO auth_id ;
[ WITH GRANT OPTION ]
```

rights
The rights GRANTed may be:
ALL PRIVILEGES | SELECT | INSERT | UPDATE | DELETE
{ , **SELECT | INSERT | UPDATE | DELETE** }*

tbl_name
The table or view name on which privilege(s) are to be GRANTed. This may be up to 24 characters.

auth_id
The authorization identifier to which privilege(s) are to be GRANTed.

სურათი 1.2

```
REVOKE rights ON tbl_name FROM auth_id ;
```

rights
The rights REVOKEd may be:
ALL PRIVILEGES | SELECT | INSERT | UPDATE | DELETE
{ , **SELECT | INSERT | UPDATE | DELETE** }*

tbl_name
The table or view name on which privilege(s) are to be REVOKEd. This may be up to 24 characters.

auth_id
The authorization identifier from which privilege(s) are to be REVOKEd.

სურათი 1.3

ცხრილის მფლობელი ვალდებულია მიანიჭოს პრივილეგია ყველა იმ მომხმარებელს, ვისაც აღნიშნული ცხრილის გამოყენება სჭირდება. მაგალითად, თუ ფრენკს სურს, რომ ბარბარას ჰქონდეს წვდომა BOOKS ცხრილზე, მან უნდა მიანიჭოს SELECT პრივილეგია აღნიშნულ ცხრილზე:

GRANT SELECT ON BOOKS TO BARBARA ;

Privileges successfully granted.

view-ს გამოყენება სვეტებზე წვდომის შეზღუდვით

ANSI/ISO სტანდარტი არ იძლევა საშუალებას შერჩევით მოხდეს წვდომა სვეტებზე. ეს ნიშნავს, რომ SELECT (არჩევა) უფლებებით თქვენ გადასცემთ პრივილეგიას ან მთლიან ცხრილზე, ან საერთოდ არაფერზე. თქვენ შეგიძლიათ ამ პირობას თავი აარიდოთ, თუ მიუთითებთ ისეთ ხედვას, რომელშიც აისახება მონაცემები და მომხმარებელს ექნება მათი დათვალიერების საშუალება, მაგრამ არ ექნება წვდომა ბაზისურ ცხრილთან. მაგალითად, თუ ბარბარა მიანიჭებს ADAM-ს SELECT პრივილეგიას ხედვაზე TITLES შემდეგნაირად:

GRANT SELECT ON TITLES TO ADAM ;

Privileges successfully granted.

ამ შემთხვევაში ადამს ექნება შესაძლებლობა ნახოს მონაცემები რამდენჯერაც მოისურვებს, თუმცა ვერ შეძლებს მათში ცვლილებების შეტანას და არ ექნება წვდომა საბაზისო ცხრილთან. SQL-ი შესაბამისად, ასეთ მოთხოვნას უკუაგდებს ადამისგან.

SELECT *

FROM BOOKS;

Error 137: User does not have SELECT

privileges.

სერიოზულ ნაკლოვანებად ხედვათა გამოყენებაში, მხოლოდ უსაფრთხოების სტრუქტურის უზრუნველსაყოფად ითვლება ის საკმაოდ დიდი დრო, რაც პროცესის დამუშავებას სჭირდება. ხედვათა არასისტემური გამოყენება საგრძნობლად აქვეითებს მთელს მონაცემთა ბაზაზე წვდომის დროს.

შეუზღუდავი პრივილეგიები და საჯარო (public) სიტყვა-გასაღებები

GRANT გაძლევთ საშუალებას მოახდინოთ ერთზე მეტი უფლებით უზრუნველყოფა, მაგრამ ერთზე მეტ ავტორიზაციის ID-ს (იდენტიფიკატორი) ვერ განსაზღვრავთ. შესაბამისად, ფრენკს შეუძლია გადასცეს ბარბარას ოთხივე პრივილეგია BOOKS ცხრილზე:

```
GRANT SELECT, INSERT, UPDATE, DELETE

ON BOOKS

TO BARBARA ;

Privileges successfully granted.
```

ზოგიერთი DBMS (მონაცემთა ბაზის მართვის სისტემა) ემიჯნება ANSI/ISO სტანდარტს და გაძლევთ საშუალებას მიუთითოთ მთელი რიგი ავტორიზაციის ID-ებისა და პრივილეგიებისა GRANT ბრძანების ფარგლებში. მაგალითად, აღნიშნული დაშვებულია DB2 SQL დიალექტში:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TO BARBARA, MORRIS ;

Privileges successfully granted.
```

თუ გსურთ მომხმარებელს ყველა ხელმისაწვდომი პრივილეგია მიანიჭოთ, ANSI/ISO SQL გთავაზობთ shortcut-ის სახით ALL PRIVILEGES (შეუზღუდავი პრივილეგიები) გამოიყენოთ. ასე რომ, თუ ფრენკს სურს BOOKS ცხრილებზე შეუზღუდავი პრივილეგია გახადოს მაგალითად PUBLIC (საჯარო), მას შეუძლია მიუთითოს:

```
GRANT ALL PRIVILEGES ON BOOKS TO PUBLIC ;

Privileges successfully granted.
```

სიტყვა-გასაღები PUBLIC გახლავთ სპეციალური ავტორიზაციის ID (იდენტიფიკატორი), რომელიც ყველა მომხმარებელზე ვრცელდება. შესაბამისად,

ზემოთხსენებული GRANT ბრძანება ყველას ანიჭებს თავისუფალ წვდომას BOOKS ცხრილზე. ყველა მომხმარებელს აქვს უფლება განახორციელოს ნებისმიერი შესაძლო ოპერაცია BOOKS-ის ფარგლებში. სხვა საკითხია, რომ ამას კარგ იდეად ვერ მოვიაზრებთ, ვინაიდან, ახლა ფრენკი ვეღარ გააკონტროლებს თავის ცხრილში მონაცემების შეცვლისა და დამატების შეტანის უფლებამოსილ პირთა ვინაობას. იმ შემთხვევაში, თუ გამოუცდელი მომხმარებელი ცვლის რომელიმე რიგს ცხრილში არსებული კავშირების გათვალისწინების გარეშე, მონაცემთა მთლიანობა შესაძლოა ადვილად დაირღვეს.

SQL რეკომენდაციები

Oracle და IBM-ის DB2 და SQL/DS მხარდაჭერა და ALTER TABLE-ის (ცხრილის შეცვლა) და CREATE INDEX-ის (ინდექსის შექმნა) პრივილეგია.

UPDATE პრივილეგიით შერჩევითი უზრუნველყოფა

SELECT, INSERT და DELETE პრივილეგიები უნდა მიენიჭოს ცხრილის (თუ ხედვის) ან ყველა სვეტს ან საერთოდ არცერთს. ANSI/ISO სტანდარტი გაძლევთ საშუალებას UPDATE პრივილეგია ცხრილის, ან ხედვის ცალკეულ სვეტებზე შერჩევითად განახორციელოთ. შესაბამისად, ბიბლიოთეკის მაგალითში, თუ მომხმარებელი ადამი უფლებამოსილია BOOKS ცხრილში განაახლოს სვეტი STATUS (სტატუსი), მაგრამ არა სვეტები TITLE (სახელწოდება) ან AUTHOR (ავტორი), ფრენკს, როგორც ცხრილის მფლობელს შეუძლია ამ განსაზღვრული განახლების შეტანა:

```
GRANT UPDATE(STATUS)

ON BOOKS

TO ADAM;

Privileges successfully granted.
```

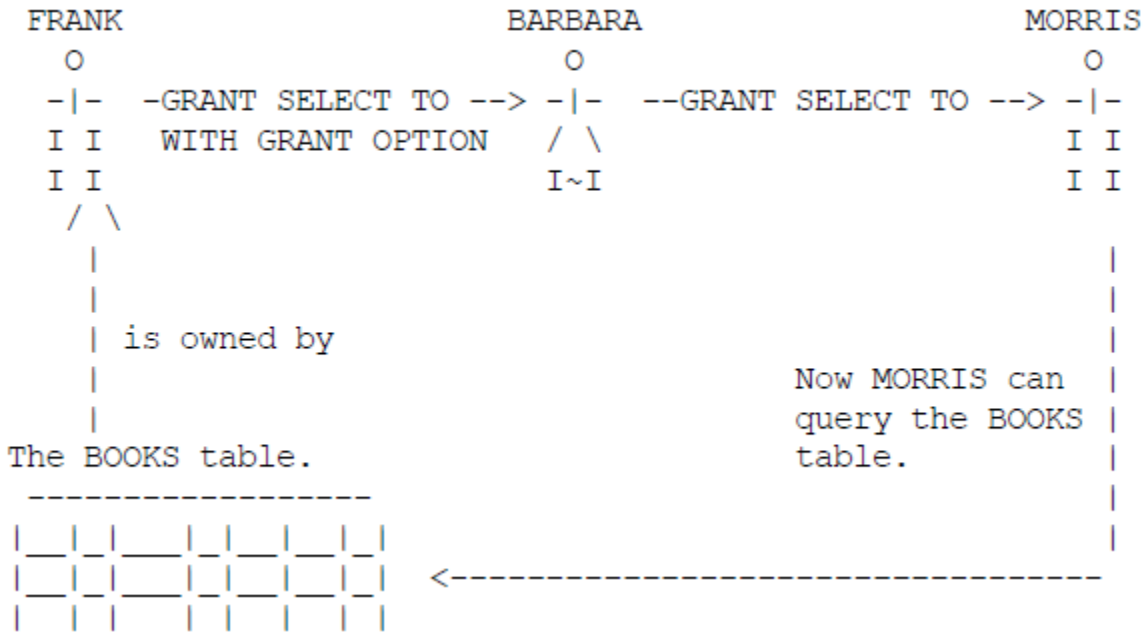
ახლა, ადამს შეუძლია წიგნების სტატუსის შეცვლა, გატანილია ისინი, თუ ნაყიდია, მაგრამ ვერ შეცვლის TITLE-ის და AUTHOR-ის სვეტების მონაცემებს. ამ სვეტთა ჩამონათვალი UPDATE-ში არჩევითია. თუ იგი გამოტოვებულია, SQL ასკვნის, რომ UPDATE ეხება ყველა სვეტს. მაგალითად, აღნიშნული ბრძანება აძლევს ბარბარას განახლების უფლებას BOOKS ცხრილის ყველა სვეტის ფარგლებში:

```
GRANT UPDATE

ON BOOKS

TO BARBARA ;

Privileges successfully granted.
```

პრივილეგიის მინიჭებისას მომხმარებლისთვის ამ პრივილეგიის სხვისთვის მინიჭებაზე ნების დართვა.

აქამდე ყველა პრივილეგია გადაიცემოდა შესაბამისი ცხრილებისა და ხედვის მფლობელების მიერ მომხმარებელზე. მაგრამ, რა ხდება იმ შემთხვევაში, როდესაც მომხმარებელს, რომელსაც თავად მინიჭებული აქვს უფლებები და, შესაბამისად, წარმოადგენს უფლებათა მიმღებს, სურს ეს უფლებები სხვა მომხმარებლებს გადასცეს? მფლობელს შეუძლია ასეთი შემთხვევების უზრუნველყოფა, თუ უფლებათა მინიჭების საწყის ეტაპზევე მიუთითებს WITH GRANT OPTION (უფლებათა გადაცემის შესაძლებლობა) პირობას. მაგალითად, სურათი 8.4 ასახავს პრივილეგიათა იმ ჯაჭვს, რომელიც გვინდა, რომ განხორციელდეს. BOOKS ცხრილის მფლობელია ფრენკი, რომელსაც სურს SELECT პრივილეგია გადასცეს ბარბარას. ბარბარას კი, თავის მხრივ, სურს აღნიშნული SELECT პრივილეგია მიანიჭოს მორისს. ამის განსახორციელებლად, ფრენკი, პირველ რიგში, ანიჭებს ბარბარას SELECT

უფლებას BOOKS-ზე, შემდეგ WITH GRANT OPTION-ით ანიჭებს მას უფლებამოსილებას გადასცეს თავისი პრივილეგიები ასევე სხვა მომხმარებლებს.



GRANT SELECT

ON BOOKS

TO BARBARA

WITH GRANT OPTION;

Privileges successfully granted.

ახლა ბარბარას უკვე აქვს SELECT პრივილეგია BOOKS-ზე და ასევე იმის საშუალება რომ გადასცეს ეს უფლება სხვა მომხმარებელსაც. აღსანიშნავია, რომ WITH GRANT OPTION ვრცელდება მხოლოდ GRANT უფლების ფარგლებში, ანუ იმ პრივილეგიაზე და იმ ცხრილზე (ხედვაზე) რომელიც აღნიშნულია GRANT–ის ბრძანებაში. შესაბამისად, ბარბარა ვერ შეძლებს INSERT, DELETE თუ UPDATE უფლებების გადაცემას ვინმეზე (როგორ მოახერხებს ამას, თუ თვითონ არ გააჩნია). ბარბარას შეუძლია მიანიჭოს SELECT პრივილეგია მორისს (ან სხვა ნებისმიერ მომხმარებელს) შემდეგნაირად:

GRANT SELECT

ON BOOKS

TO MORRIS ;

Privileges successfully granted.

ახლა მორისიც ფლობს BOOKS-ზე SELECT უფლებებს. თუ ბარბარამ მიუთითა WITH GRANT OPTION მორისისთვის, ამ უკანასკნელსაც ექნება SELECT უფლებების გადაცემის საშუალება. BOOKS-ის პირველ მფლობელს - ფრენკს არაფერი ეცოდინება მომხმარებელთა შორის მიმდინარე პრივილეგიათა ჯაჭვზე და, შესაბამისად, დაკარგავს BOOKS ცხრილზე კონტროლს. უფლებათა წვდომაზე მკაცრი კონტროლის შესანარჩუნებლად, კარგი იქნება თუ გავაკონტროლებთ, თუ ვინ იღებს აღნიშნულ WITH GRANT OPTION-ს (უფლებათა გადაცემის შესაძლებლობა).

„პრივილეგიათა ანულირება“: REVOKE (გაუქმება) ბრძანება

ყველა კომერციული SQL მომწოდებელი გვთავაზობს ბრძანება REVOKE-ს, როგორც GRANT-ის (მინიჭება) მიერ მინიჭებული პრივილეგიების ანულირების მეთოდს. ბრძანება REVOKE რეალურად წარმოადგენს ANSI/ISO სტანდარტის დამატებას. ANSI/ISO SQL ეყრდნობა ვარაუდს, რომ მონაცემთა ბაზის დიზაინერი ჯერ ქაღალდზე აყალიბებს დიზაინს და შემდეგ იყენებს DDL ბრძანებებს ყველა ცხრილის, ხედვის თუ უსაფრთხოების უფლებების შესაქმნელად. სტანდარტი დიზაინერს ძალიან ზღუდავს ერთხელ დანერგილი დიზაინის გადაკეთების შემთხვევებში. შესაბამისად, ANSI-ს ფარგლებში ერთხელ მინიჭებული უფლებების ცვლილება დიდ სირთულეებს უკავშირდება.

<u>SQL რეკომენდაციები</u>

ბრძანება REVOKE საერთოდ არ ფიგურირებს ANSI/ISO სტანდარტებში.

REVOKE-ის ფორმატი, რომელიც დანერგილია, კომერციული SQL სისტემების უმრავლესობაში, ნაჩვენებია სურათში 1.3. მისი ფორმატი ძალიან ჰგავს GRANT ბრძანებას.

მაგალითად, ეს ბრძანება აუქმებს INSERT და UPDATE პრივილეგიებს მორისისთვის ხედვაზე TITLES:

```
REVOKE INSERT, UPDATE

ON TITLES

FROM MORRIS ;
```

Privileges successfully revoked.

თქვენ, ასევე, შეგიძლიათ პირობა ALL PRIVILEGES (შეუზღუდავი პრივილეგია) გამოიყენოთ ბრძანებაში REVOKE. მაგალითად, იმისათვის, რათა ადამმა ვეღარ გამოიყენოს BOOKS ცხრილი, ფრენკმა უნდა მიუთითოს:

```
REVOKE ALL PRIVILEGES

ON BOOKS

FROM ADAM ;
```

Privileges successfully revoked.

REVOKE უფლებას გაძლევთ გააუქმოთ მხოლოდ ის პრივილეგიები, რაც თავად მიიღეთ. ეს ნიშნავს, რომ თუ განვიხილავთ ზემოაღნიშნულ პრივილეგიათა ჯაჭვს, რომელიც ნაჩვენებია სურათზე 1.4, ფრენკი ვერ შეძლებს BOOKS ცხრილზე SELECT უფლების გაუქმებას მორისისთვის, მიუხედავად იმისა, რომ ცხრილის მფლობელია. სამაგიეროდ მას შეუძლია გააუქმოს ბარბარას პრივილეგია:

```
REVOKE SELECT

ON BOOKS

FROM BARBARA ;
```

Privileges successfully revoked.

სისტემათა უმრავლესობაში ეს გამოიწვევს DBMS-ში (მონაცემთა ბაზის მართვის სისტემა) შესაბამისი პრივილეგიების გაუქმებას ყველა უფლებამინიჭებული პირისგან უფლებათა ჯაჭვის ქვევით. აქედან გამომდინარე, მორისიც დაკარგავს SELECT უფლებებს BOOKS-ზე, რადგან ის მინიჭებულია ბარბარას მიერ. ვინაიდან REVOKE არასტანდარტული პარამეტრია, SQL-ის სხვადასხვა დიალექტი მას სხვადასხვაგვარად ნერგავს. REVOKE-ის კასკადური ეფექტი შესაძლოა არ იყოს პირადად თქვენი SQL დიალექტის პარამეტრი.

დასკვნა

სამაგისტრო ნაშრომის მიზანს წარმოადგენდა მომხმარებლებისთვის უკეთ გაეცნო მონაცემთა ბაზის უსაფრთხოების საკითხი და ამ მიზნით განხილულ იქნა სხვადასხვა მეთოდები, რომელთა საშუალებითაც შასაძლებელი იქნება მაქსიმალურად უსაფრთხო სისტემის შექმნა.

სწორად დაგეგმილმა უსაფრთხოების სისტემამ მომხმარებელს არ უნდა მისცეს უფლება შეასრულოს ის მოქმედებები, რომლებიც სცილდება მის უფლებამოსილებას. მაგალითად, შეგვიძლია ავკრძალოთ მონაცემთა ბაზიდან იმ მონაცემების წაშლა, რომელთა შენახვის ვადა არ გასულა, ახალ თანამშრომლებს მივცეთ მონაცემების მხოლოდ წაკითხვის უფლება. შემდგომში მათ შეგვიძლია გავუფართოვოთ უფლებები და მივცეთ მონაცემების შეცვლის, წაშლის ან ჩამატების უფლება.

ადმინისტრატორმა თვალყური უნდა ადევნოს, აგრეთვე, თანამშრომლების გადასვლას ერთი განყოფილებიდან მეორეში. თანამდებობის შეცვლა დაუყოვნებლივ უნდა აისახოს მიმართვის უფლებებზე. დროულად უნდა წავშალოთ ის მომხმარებლები, რომლებიც აღარ მუშაობენ ფირმაში. თუ თანამშრომელი შვებულებაში ან ხანგრძლივ მივლინებაში მიდის, მაშინ მისი სააღრიცხვო ჩანაწერი დროებით უნდა დავბლოკოთ.

პაროლების შექმნისას უნდა დავიცვათ სტანდარტული მოთხოვნები. სასურველია, რომ პაროლი სიმბოლოების გარდა ციფრებსაც შეიცავდეს. პაროლი არ უნდა იყოს ოჯახის წევრის სახელი, დაბადების წელი, პასპორტის ნომერი და ა.შ. პაროლს უნდა ჰქონდეს მოქმედების შეზღუდული ვადა, რომლის გასვლის შემდეგ მომხმარებელმა ის უნდა შეცვალოს. უსაფრთხოების გამო სასურველია პაროლები დაშიფრული იყოს . იმ შემთხვევაში თუ ჩვენი ბაზის "გატეხვა" მაშინ იქიდან მონაცემების ამოღება შეუძლებელი იქნება, რადგან ჩვეულებრივი პაროლების მაგივრად გაურკვეველი სიმბოლოები იქნება ბაზაში. მაგალითად:5f6955d227a320c7f1f6c7da2a6d96a851a8118f

	FirstName	LastName	UserName	Password
	ნიკოლოზი	ფცქიალაძე	nfcqialadze	5f6955d227a320c7f1f6c7da2a6d96a851a8118f
	გიორგი	კაკაშვილი	gkakashvili	2a8539f906ea9825622b41920943554410183240
	დალი	ქარჩავა	dqarchava	603ae0f0fee3d423c77bf8c432d413ad87ae0b44
	ხათუნა	დევაძე	xdevadze	ab4c55d451bac6557678b2aa9d602da8ab001265
	ნათია	თავართქილაძე	ntavartqiladze	dbf0c012065b013321c8be6dd8a4d1d0b9eab6de
	ვახტანგ	ბოხაშვილი	vboxashvili	5994f427108559d2100231a8b8bce768f78dbcbf
	ლიღი	ფოჩხუა	lfochxua	cfa1150f1787186742a9a884b73a43d8cf219f9b
	მაყვალა	ფარტენაძე	mayvala.fartenadze	5f6955d227a320c7f1f6c7da2a6d96a851a8118f
	ნანი	ცინცაძე	nani.cincadze	dc28119d07b55c33ce29aa11b1f1eccdf6751f83
	გულნაზი	შომახია	gulnazi.shomaxia	edd2c1a13c2a3b402cafd6edfaa78175f5fbf4a3
	ნონა	ცინცაძე	nona.cincadze	cf121f4b3e1424ccd5b07ab7e823262dc0c84a23
	მიხეილ	ეფრემაშვილი	mikheil.efremashvili	fc7a734dba518f032608dfeb04f4eeb79f025aa7
	მაკა	ნოდია	maka.nodia	a72f9df2b74c15d6133eba3c6c07405eaf853c42
	მედეა	ბარამიძე	medea.baramidze	5ad140d5f4b7cad9d45ac9c0d051724ab2960e88
	დავით	ღლონტი	davit.glonti	7f8f0d635d897bb12cd7a341abc424bdceb63c58

დღესდღეობით, თითქმის ყველა თანამედროვე მონაცემთა ბაზების მართვის სისტემაში არსებობს მონაცემთა ბაზების სარეზერვო ასლების შექმნისა და მათი აღდგენის ინტეგრირებული იარაღები, თანაც სარეზერვო ასლების შექმნა მონაცემთა სარეზერვო ფიზიკურ მატარებლებზე მაშინაც კი არის შესაძლებელი, როდესაც მართვის სისტემა ერთდროულად მრავალ მომხმარებელს ემსახურება.

ლიტერატურა:

1. Structured Query Language (SQL) – A Practical Introduction – AKEEL I DEEN
2. http://citforum.ru/security/articles/safe_db/
3. http://pyramidin.narod.ru/php42/database.html
4. http://infosecmd.narod.ru/gl6.html