ივანე ჯავახიშვილის სახელობის თბილისის
სახელმწიფო უნივერსიტეტი

მარიამ ფუხაშვილი

ტრანზაქციის მართვის თავისებურებები მონაცემთა ბაზებში

სამაგისტრო ნაშრომი შესრულებულია საინფორმაციო
ტექნოლოგიების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მანანა ხაჩიძე
ტექნიკის მეცნიერებათა კანდიდატი
აკადემიური დოქტორი
თანახელმძღვანელი: მაია არჩუაძე

თბილისი
2013

ანოტაცია

დღეისათვის მონაცემთა ბაზის გამოყენება იმდენად აქტუალურია , რომ პრაქტიკულად ყველა ტექნოლოგია ეყრდნობა ბაზებისა და მონაცემთა ბაზის მართვის სისტემების განვითარებას. ტრანზაქციებს მონაცემთა ბაზის მართვის სისტემაში უმთავრესი ადგილი უჭირავს. სამაგისტრო ნაშრომის მიზანიც სწორედ ისეთი სისტემის შექმნაა, რომელიც არაპროფესიონალ მომხმარებელს დაეხმარება უზრუნველყოს ტრანზაქციის მართვა საკუთარ მონაცემთა ბაზაში. ნაშრომი ხაზს უსვავს იმ ძირითად პრინციპებს თუ რაოდენ მნიშვნელოვანია ტრანზაქციის მართვის თავისებურებების ცოდნა მონაცემთა ბაზაში მონაცემების მდგრადობისა და მთლიანობის შენარჩუნებისთვის.მასში ასევე წარმოდგენილია ტრანზაქციის მართვასთან დაკავშირებული პრობლემები და მოყვანილია მათი გადაჭრის გზები. ნაშრომში შემუშავებული მოდელი სრულად აღწერს ტრანზაქციის პროცესში ყველაზე ხშირად გამოყენებულ ოპერაციებს.

annotation

The utilization of databases is now so widespread that virtually every technology and product relies on databases and DBMSs for its development and commercialization.

Transaction is one of the most important part for developer to be master in development. Master's work aim is to establish a system which will help to ensure that non-professional users to manage their transaction database. This work underlines the importance of the transaction to the basic principles of knowledge management features of a database for maintaining the stability and integrity. It also provides transaction management-related problems and their solutions presented. Research developed a model to describe the transaction process is the most commonly used operations.

სარჩევი

შესავალი 5
ტრანზაქცია - მისი არსი და ძირითადი ცნებები 7
მოდელის აღდგენის კონფიგურაცია 16
ტრანზაქციის ძირითადი თვისებები 18
მონაცემებთან მიმართვის პრობლემები და მათი გადაჭრა ტრანზაქციის პროცესში 21
მონაცემთა ბაზებში ტრანზაქციის მართვის მხარდამჭერი სისტემა 32
დასკვნა 36
ლიტერატურა 37

შესავალი

მონაცემთა ბაზა ელექტრონული ინფორმაციის მართვის სისტემაა. მისი მთავარი დანიშვნულებაა ინფორმაციის დაცულად შენახვა და მომხმარებლისთვის სათანადო ფორმით მიწოდება.მონაცემთა ბაზები მნიშვნელოვან როლს თამაშობს ჩვენს ცხოვრებაში,გვხდება თითქმის ყველა სფეროში, სადაც კომპიუტერი გამოიყენება. ტერმინი ბაზა ნიშნავს დაკავშირებული მონაცემების კოლექციას.იგი შეიძლება იყოს სხვადასხვა ზომის და სხვადასხვა სირთულის და შეიცავდეს რამდენიმე ასეულ და მილიონ ჩანაწერს. ეს ინფორმაცია უნდა იყოს ორგანიზებული და გამოყენებადი ისე, რომ მომხმარებელს შეეძლოს მოძებნა და განახლება მონაცემების, როგორც მას სჭირდება.ასევე, ორგანიზაციები და კომპანიები, მცირე თუ დიდი, თავისი საქმიანობის გამო დამოკიდებულია მონაცემთა ბაზებზე.ორგანიზაციას უნდა ჰქონდეს ზუსტი და საიმედო მონაცემები ეფექტური გადაწყვეტილებების მიღებაში.

მონაცემთა ბაზის სისტემა გულისხმობს, რომ მასში არსებული მონაცემები უნდა აკმაყოფილებდეს გარკვეული დონის ხარისხს(როგორიცაა სიზუსტე, ხელმისაწვდომობა, გამოყენებადობა და მდგრადობა). ამ დონის მისაღწევად გამოიყენება მონაცემთა ბაზის მართვის სისტემა(DBMS). DBMS პროგრამული სისტემაა რომელიც აკმაყოფილებს მონაცემთა ბაზის მოთხოვნებს რომელიც ხშირ შემთხვევაში ძალიან დიდი და კომპლექსურია. მონაცემთა ბაზების მართვის სისტემას შეუძლია ერთდროულად მოემსახუროს ათი ათასობით მომხმარებელს, თანაც ამასთან ერთად უზრუნველყოს ბაზებში არსებული მონაცემების საიმედო უსაფრთხოება და მთლიანობა.მაგალითად ავიღოთ კომერციული ორგანიზაცია www.Amazon.com, რომელთანაც მსოფლიოს სხვადასხვა ადგილებში მყოფი მრავალი მომხმარებელი ერთდროულად აქტიურ დიალოგს ახორციელებს: გზავნიან სხვადასხვა შინაარსის კონკრეტულ მიმართვებს, ახორციელებენ მათთვის საინტერესო ინფორმაციის ძიებას, აფორმებენ შეკვეთებსა და ხელშეკრულებებს, გადარიცხავენ ფულად თანხებს და ა.შ.. მონაცემთა ბაზების მართვის სისტემა ისეთი სახით ახორციელებს ასეთი მრავალ-მომხმარებლური წვდომის კოორდინაციას, რომ მუშაობის პროცესში დაცულია ბაზებში განთავსებული მონაცემების მთლიანობა და მათი ცვლილებების დაფიქსირების

ადეკვატურობა. ნებისმიერი კონკრეტული მომხმარებლის მიერ განხორციელებული ნებისმიერი სახის ოპერაციები არავითარ ზეგავლენას არ ახდენენ სხვა მომხმარებლების ურთიერთქმედებაზე მონაცემთა ბაზებთან. მონაცემთა ბაზების მართვის სისტემები უზრუნველყოფენ მონაცემთა ბაზების დროებით ჩაკეტვებს, რითაც ახდენენ მოსალოდნელი კონფლიქტების აცილებას.ყველა ზემოთ ჩამოთვლილი მიზეზების გამო მონაცემთა ბაზების მართვის სისტემებმა მოიპოვეს ფართო გავრცელება მთელ მსოფლიოში. დღეისათვის საკმაოდ რთულია რომელიმე მნიშვნელოვანი გამოყენებითი პროგრამის დასახელება, რომელიც არ მოიცავს მონაცემთა ბაზას ან არ გაჩნია მასთან რაიმე კავშირი.

მონაცემთა ბაზების მართვის სისტემები იძლევიან შესაძლებლობას მონაცემთა მთლიანობის უზრუნველსაყოფად. ამ შესაძლებლობას ტრანზაქცია (transaction) ეწოდება.

ტრანზაქციებს მონაცემთა ბაზის მართვის სისტემაში დიდი როლი უჭირავს,ჩვენ შეიძლება გვქონდეს ისეთი შემთხვევები, როცა მონაცემთა ბაზაში ერთზე მეტ კომბინაციას ვწერთ (INSERT / UPDATE / DELETE), სრულიად შესაძლებელია რომ რომელიმე ოპერაცია ვერ შესრულდეს და დაგვჭირდეს ცვლილებების აღდგენა იმ მდგომარეობაში, რაც მათ შეცდომის მოხდენამდე ჰქონდათ, რათა შევინარჩუნოთ მონაცემთა მდგრადობა და მთლიანობა. ამ პრობლემის გადაწყვეტაში გვეხმარება მონაცემთა ბაზის ტრანზაქცია, რომელსაც ბაზა მოყავს ACID (ატომარობა, მთლიანობა, იზოლირებულობა, მდგრადობა) შესაბამისობაში.
სტანდარტული ტრანზაქციის პროცესის პროგრამული უზრუნველყოფა შეიქმნა 1960 წელს და მჭიდრო კავშირშია მონაცემთა ბაზების მართვის სისტემებთან. პირველი ტრანზაქციების დამუშავების სისტემა იყო ავიაკომპანია SABRE სისტემა, რომელმაც ფუნქციონირება 1960 წელს დაიწყო. იგი ამუშავებდა 83,000 -ზე მეტ ტრანზაქციას დღეში. სისტემა გაშვებული იყო ორ IBM 7090 კომპიუტერზე.1972 წელს SABRE-მ მიიღო პირველი IBM-ის პროდუქტი ავიახაზების კონტროლის პროგრამა (ACP) , მოგვიანებით კი ტრანზაქციის დამუშავების მოწყობილობა (TPF). გარდა ამისა, TPF გამოიყენება დიდი ბანკებს, საკრედიტო ბარათების კომპანიებსა და სასტუმროს ქსელებში.

ტრანზაქციის კონცეფციის გაგება ძალიან მნიშვნელოვანია მონაცემთა ბაზასთან მუშაობის დროს, რადგან იგი თავიდან გვარიდებს ისეთ პრობლემებს რომელიც შეიძლება წარმოიქმნას მაშინ , როდესაც ძალიან ბევრი მომხმარებელი ერთდროულად მუშაობს მონაცემთა ბაზასთან. ტრანზაქცია იცავს მონაცემებს და უზრუნველყოფს მათ თანმიმდევრულ შენახვას.

ტრანზაქცია - მისი არსი და ძირითადი ცნებები

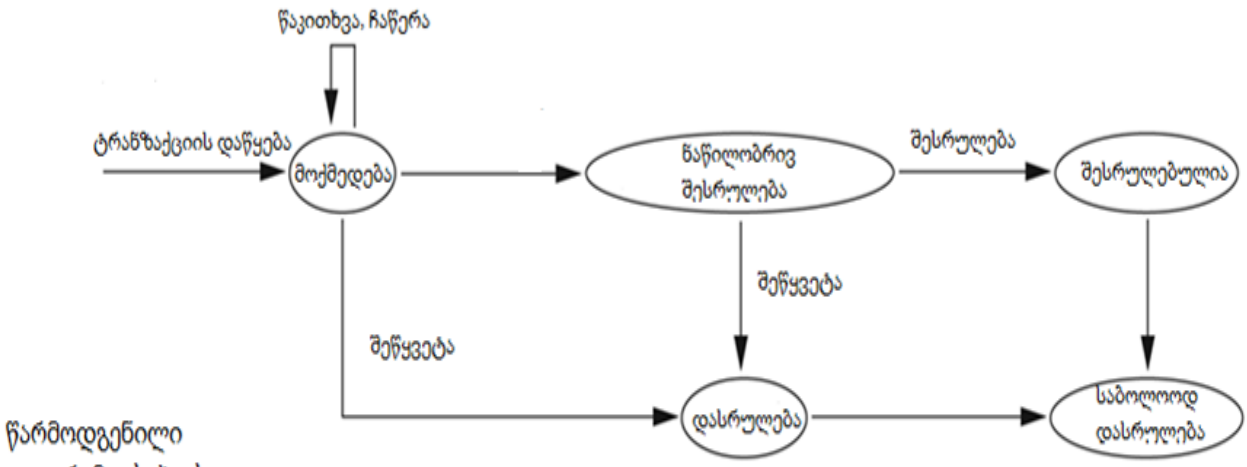
ტრანზაქცია არის პროცესი, როცა სრულდება მასში შემავალი ყველა ბრძანება ან არც ერთი მათგანი. ეს ბრძანებები ერთმანეთთან იმყოფებიან მჭიდრო კავშირში.ტრანზაქცია შეიძლება შედგებოდეს როგორც ერთი, ისე ასობით ბრძანებისაგან.

ტრანზაქციის ცნობილი მაგალითია ფონდების მოხსნა სადეპოზიტო ანგარიშიდან და მათი განთავსება მობილურ ანგარიშზე. ჩვენ გვჭირდება ორივე ოპერაციის (მოხსნა და განთავსება) წარმატებულად შესრულება, მაგრამ თუ ფონდების გამოტანის ოპერაცია შესრულდა, ხოლო განთავსების ოპერაცია ვერ განხორციელდა, მაშინ ჩვენ ინტერესებში იქნება საწყისი გამოტანის ოპერაციის გაუქმება და ფონდების დაბრუნება უკან სადეპოზიტო ანგარიშზე. ეს ორი ოპერაცია ორივე ერთად წარმოადგენს ერთ ტრანზაქციას, რომელიც ან უნდა შესრულდეს მთლიანად, ან საერთოდ არ უნდა შესრულდეს რათა არ მოხდეს ფულადი თანხების დაკარგვა. მონაცემთა ბაზების მართვის სისტემები იძლევიან მონაცემებზე მოქმედების ოპერაციათა ტრანზაქციების სახით დაჯგუფების საშუალებას.

ტრანზაქციის მაგალითია ასევე , როცა მომხმარებელი იძენს ტურს ონლაინით ტურისტულ სააგენტოში.მას შეუძლია მოითხოვოს შეკვეთები ავიაკომპანიაზე, სასტუმროზე, მანქანის დამქირავებელ კომპანიაზე და ა.შ. დამკვეთის თვალში მთელი ერთი პაკეტი არის ერთი შეკვეთა, მაგრამ ეს შეკვეთა ბევრი ტრანზაქციებისგან შედგება. ერთი ტრანზაქცია სჭირდება ავიაკომპანიის დაჯავშნას, ერთი-სასტუმროს ნომრის დაჯავშნას და მანქანის დაქირავებაც ცალკე ტრანზაქციაა. ტრანზაქციაა ასევე თითოეული ბილეთის დაბეჭდვა და საბოლოო ინვოისი.

მონაცემთა მართვის სისტემა ნებას აძლევს მომხმარებელს (ან პროგრამისტს) განსაზღვროს ტრანზაქციის საზღვრები. თუ ყველა ბრძანება წარმატებით შესრულდა, მაშინ მოხდება ტრანზაქციის ფიქსირება (Commit). თუ ტრანზაქციის რომელიმე ბრძანება ვერ შესრულდა, მაშინ მოხდება ტრანზაქციის უკუქცევა (Rollback), ანუ გაუქმდება ყველა ცვლილება. სისტემა აღდგება საწყის მდგომარეობაში, რომელიც მას ჰქონდა ტრანზაქციის დაწყებამდე.

მოცემული სქემა გვიჩვენებს თუ როგორი პრინციპით მუშაობს ტრანზაქცია მონაცემთა ბაზაში:



წარმოდგენილი დიაგრამა ასახავს ტრანზაქციის შესრულებას

ტრანზაქციას გააჩნია საწყისი და საბოლოო წერტილები. მათ დასაფიქსირებლად გამოიყენება ოპერატორები:

- **BEGIN TRANSACTION**
- **COMMIT TRANSACTION**
- **ROLLBACK TRANSACTION**
- **SAVE TRANSACTION**

- **BEGIN TRANSACTION** გვიჩვენებს საიდან იწყება მუშაობის უწყვეტი პროცესი, რომელიც განიხილება როგორც ერთი ტრანზაქცია ანუ ფიქსირდება ტრანზაქციის საწყისი წერტილი. თუ რაიმე მიზეზით არ მოხდა ტრანზაქციის დაფიქსირება, მაშინ

ყველა მოქმედება უქმდება და მონაცემთა ბაზა ბრუნდება იმ მდგომარეობაში, რომელიც მას ეკავა ოპერატორის გამოძახების წერტილში.

- **COMMIT TRANSACTION** უზრუნველყობს ტრანზაქციის ფიქსაციას ანუ ტრანზაქციის პროცესის დასრულებას. ამ ოპერატორის შესრულების შემდეგ ტრანზაქციის შედეგები ფიქსირდება, ტრანზაქცია ხდება მუდმივი და ინახება სამუშაოს ავარიული დასრულების შემთხვევაშიც.

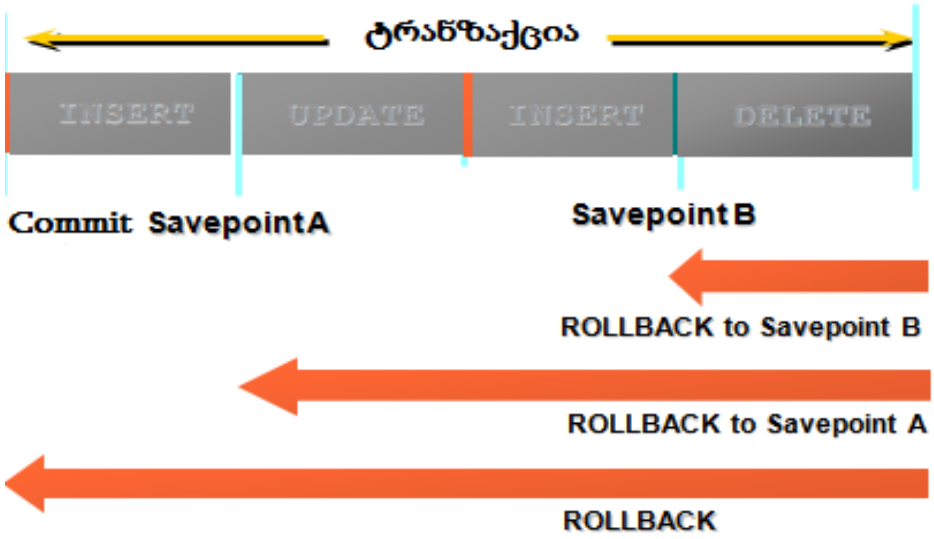
```
BEGIN TRAN

UPDATE  authors
SET     au_fname = 'John'
WHERE   au_id = '172-32-1176'

UPDATE  authors
SET     au_fname = 'Marg'
WHERE   au_id = '213-46-8915'

COMMIT TRAN
```

- **ROLLBACK TRANSACTION** -ეს ოპერატორი შეიძლება გამოვიყენოთ წარუმატებელი მოქმედების შესრულების უარყოფის მიზნით. მისი საშუალებით ბაზა შეიძლება დავაბრუნოთ იმ მდგომარეობაში, რომელიც მას ეკავა ტრანზაქციამდე ან დაბრუნება მოხდეს ერთ-ერთ საკონტროლო წერტილში.

```
BEGIN TRAN

UPDATE  authors
SET     au_fname = 'John'
WHERE   au_id = '172-32-1176'

UPDATE  authors
SET     au_fname = 'JohnY'
WHERE   city = 'Lawrence'

IF @@ROWCOUNT = 5
  COMMIT TRAN
ELSE
  ROLLBACK TRAN
```

- **SAVE TRANSACTION** -სვავს საკონტროლო წერტილებს. ეს ოპერატორი თავის კოდში გვიჩვენებს თუ სად, რა მდგომარეობაში უნდა დაბრუნდეს ბაზა ტრანზაქციის წარუმატებლად განხორციელების შემთხვევაში. ოპერატორის ერთხელ შესრულების შემდეგ ეს საკონტროლო წერტილები იშლება.

  მოცემულ სურათზე მოცემულია ტრანზაქციის კონტროლი commit,rollback da savepoint ბრძანებების გამოყენებით



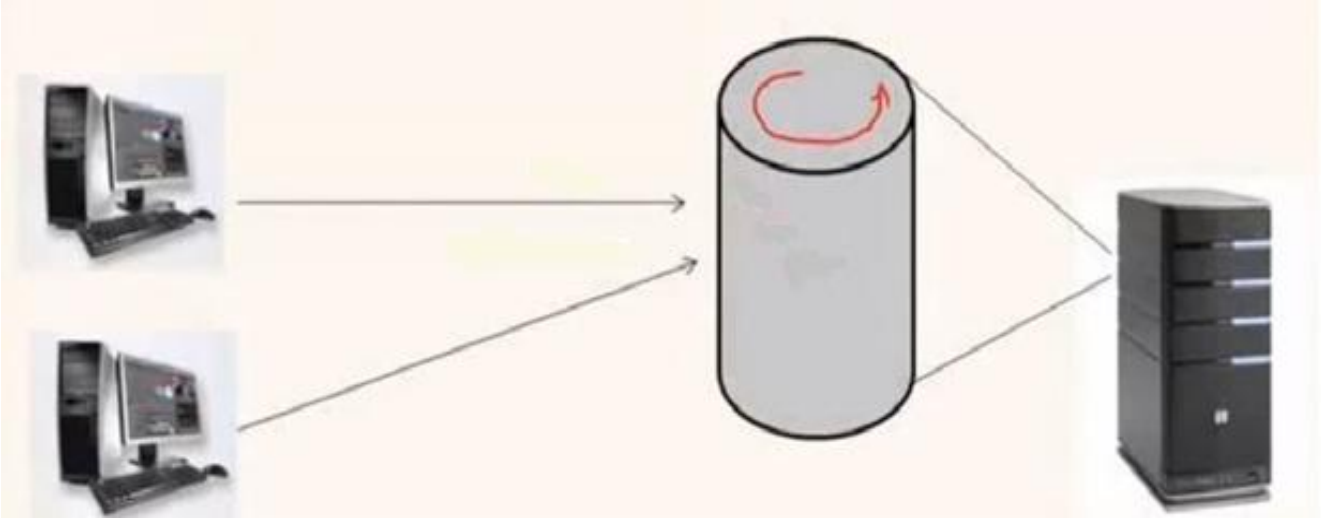
ინფორმაცია სისტემის საწყისი მდგომარეობის შესახებ ინახება ტრანზაქციების ჟურნალში.

ტრანზაქციების იმპლიმენტაცია (განხორციელება) ეფუძნება მონაცემთა წინასწარ რეგისტრაციას. ტრანზაქციის დასაწყისში მონაცემთა ბაზებში შესატანი ახალი მნიშვნელობები (ანუ ცვლილებები) და ბაზაში არსებული საწყისი მნიშვნელობები (ანუ ის მნიშვნელობები, რომლებიც უნდა შეიცვალოს ახალი მნიშვნელობებით) ჩაიწერება სარეგისტრაციო ჟურნალში და არა მონაცემთა ბაზაში. როდესაც ტრანზაქცია მთლიანად

წარმატებულად შესრულდება, მაშინ ის ფიქსირდება (ანუ მონაცემთა მართვის სისტემა აფიქსირებს წარმატებულ ტრანზაქციას). ამ ფაზაში ის ცვლილებები, რომლებიც იყო ჩაწერილი სარეგისტრაციო ჟურნალში, ფაქტიურად გადადის მონაცემთა ბაზაში, და შესაბამისი ცვლილებები ხილვადი ხდება სხვა მომხმარებლებისათვის. მეორეს მხრივ, თუ ტრანზაქციის რომელიმე ნაწილი ვერ განხორციელდა (ანუ მისი განხორციელება ჩავარდა) ნებისმიერი მიზეზის გამო, მაშინ ცვლილებები ბრუნდება უკან, ანუ არცერთი ცვლილება ჩაწერილი სარეგისტრაციო ჟურნალში ფაქტიურად არ გადადის მონაცემთა ბაზაში. მისი ასეთი დიდი როლიდან გამომდინარე იგი ინახება ერთ ან რამდენიმე ფაილში, რომლებიც მოშორებულია მონაცემთა ბაზის ფაილებისგან. ტრანზაქციის ჟურნალის ჩანაწერები იწერება იქამდე სანამ ყოველგვარი ცვლილება ბუფერული მეხსიერებიდან ჩაიწერება მონაცემთა ბაზის ფაილებში. ბაზას შეიძლება გააჩნდეს რამდენიმე ტრანზაქციის ჟურნალი.

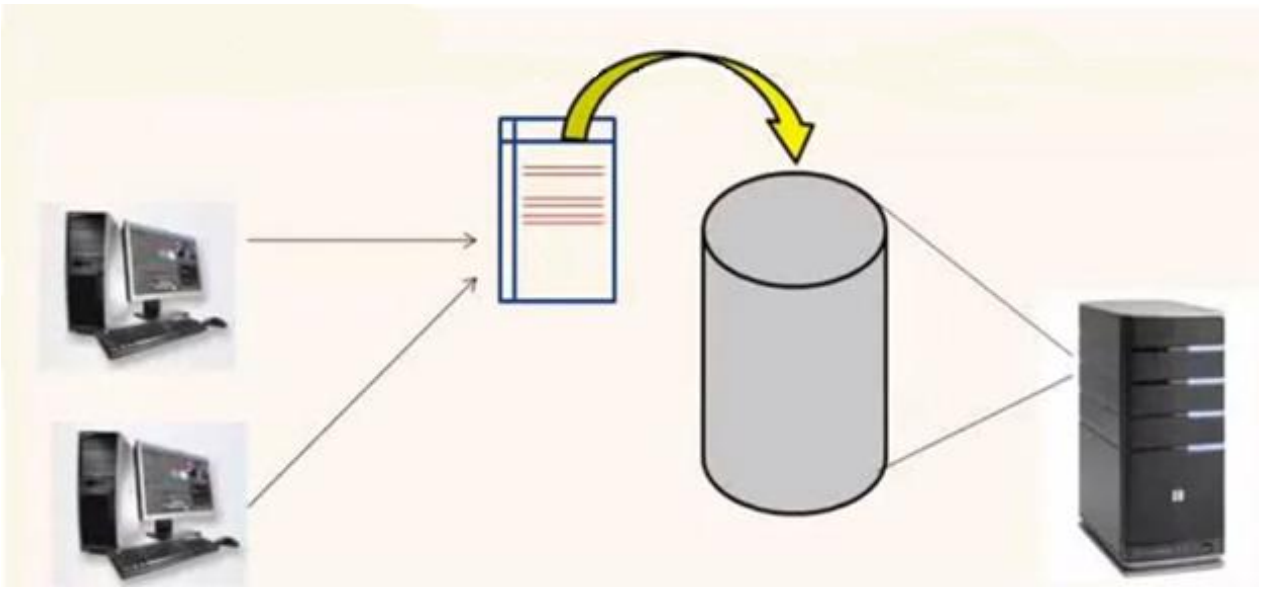
განვიხილოთ შემთხვევა როცა ორი მომხმარებელი ერთდროულად მუშაობს მონაცემთა ბაზაში , ცვლის წერს ან შლის მონაცემებს. ერთი მომხმარებლის მიერ ჩატარებულმა ნებისმიერმა ცვლილებამ შეიძლება დააზიანოს მეორე მომხმარებელი ან მთლიანად მონაცემთა ბაზა. შეიძლება მომხმარებელს წაეშალოს და შეცვალოს ძალიან მნიშვნელოვანი ინფორმაცია , რომლის აღდგენასაც ვეღარ შეძლებს:

მოცემულ სურათზე ორი მომხმარებელი ერთდროულად მუშაობს მონაცემთა ბაზასთან



ამ პრობლემების თავიდან ასაცილებლად შემოდის ტრანზაქციის ჟურნალის ცნება და მომხმარებლებს საშუალება ეძლევათ ტრანზაქციის ჟურნალის დახმარებით ცვალონ ან შალონ მონაცემები და ბაზის დაზიანების შემთხვევაში აღადგინონ საწყის მდგომარეობაში.მომხმარებელი ოპერაციებს ატარებს ტრანზაქციის ჟურნალში და არა უშუალოდ მონაცემთა ბაზაში. იმ დროს როცა მომხმარებელი დარწმუნდება ბრძანებების სისწორეში ტრანზაქციის ჟურნალში არსებული ინფორმაცია გადაიწერება მონაცემთა ბაზაში.

როგორც ეს ქვემოთ მოყვანილ სურათზეა წარმოდგენილი:



წინასწარი რეგისტრაცია ასევე სასარგებლოა მონაცემთა ბაზის აღდგენისას ავარიული სიტუაციის შემდეგ. სარეგისტრაციო ჟურნალი (log) მოიცავს მონაცემთა ბაზაში განხორციელებულ ყველა ცვლილებას, იმ ინფორმაციის ჩათვლით დაფიქსირდა თუ უკან დაბრუნდა თითოეული ტრანზაქცია. მონაცემთა ბაზის აღსადგენად ადმინისტრატორს შეუძლია აღადგინოს მონაცემთა ბაზის წინამორბედი სარეზერვო ასლი და გაიმეოროს დაფიქსირებული ტრანზაქციები. ამას ეწოდება მონაცემთა ბაზის აღდგენა დაფიქსირებული ტრანზაქციების გამეორებით (roll forward recovery). ზოგიერთი მონაცემთა ბაზის მართვის სისტემა იყენებს წინასწარ რეგისტრაციას (ცვლილებების წინსწრებით რეგისტრაციას), მაგრამ ამასთან ერთად ახორციელებს მონაცემთა ბაზის ფაქტიურ ცვლილებას ტრანზაქციის ფორმალურ დაფიქსირებამდე. ასეთ სისტემებში ავარიული სიტუაციის შემდეგ აღდგენა შესაძლებელია განხორციელდეს მონაცემთა ბაზების მართვის სისტემის გადატვირთვით და სარეგისტრაციო ჟურნალში არსებული ყველა იმ ტრანზაქციების გაუქმებით, რომლებიც არ დაფიქსირდა. ასეთ მიდგომას დაუფიქსირებელი ტრანზაქციების გაუქმებით აღდგენას უწოდებენ (rollback recovery).

მონაცემთა ბაზაში ტრანზაქიის ჟურნალმა უნდა უზრუნველყოს შემდეგი ოპერაციები:

1. დააბრუნოს ინდივიდუალური ტრანზაქცია, იმ შემთხვევაში თუ ბაზა გასცემს „ROLLBACK“ ბრძანებას ან ბაზის ძრავაში აღმოჩნდება შეცდომა.
2. უნდა მოხდეს ყველა ტრანზაქციის უკან დაბრუნება, SQL Server-ის გადატვირთვის დროს
3. აღადგინოს დაუსრულებელი ტრანზაქციის ჩანაწერები ჟურნალში და არა ბაზის ფაილებში,ტრანზაქციის შედეგები იწერება ბაზის ფაილებში SQL Server-ის გადატვირთვის დროს
4. მხარს უჭერს ტრანზაქციების რეპლიკაციის

სისტემის ავარიული გათიშვის შემთხვევაში სარეზერვო ასლი შეიცავს ინფორმაციას ყველა განხორციელებულ ცვლილებაზე, გარდა მასიური ოპერაციების, ამიტომ საჭირო ხდება მათი ხელახალი შესრულება.

ტრანზაქციის ჟურნალის ფაილები დაყოფილია ვირტუალურ ფაილებად , რომელთა ზომას ფიზიკურ ფაილში განსაზღვრავს მონაცემთა ბაზის ძრავა, რომელიც ასევე განსაზღვრავს თუ როდის და რომელი ფაილის ზომა შეამციროს. თუმცა ჩვენ შეგვიძლია განვსაზღვროთ ფიზიკური ფაილების მინიმალური და მაქსიმალური ზომა, ასევე დავამატოთ ფიზიკური ფაილები ჟურნალში ან წავშალოთ. მოცემული მაგალითი გვიჩვენებს თუ როგორ უნდა ვმართოთ ტრანზაქციის ჟურნალი SQL Server-ში, სატესტოდ შევქმნათ EmployeeDB მონაცემთა ბაზა.

```
USE master;

IF EXISTS
(
  SELECT name FROM sys.databases
  WHERE name = 'EmployeeDB'
)
DROP DATABASE EmployeeDB;

CREATE DATABASE EmployeeDB
ON
(
  NAME = EmployeeDB_dat,
  FILENAME = 'C:\SqlData\EmployeeDb.mdf'
)
LOG ON
(
  NAME = EmployeeDB_log,
  FILENAME = 'C:\SqlData\EmployeeDb.ldf'
);
```

ბაზის შექმნის შემდეგ გამოვიყენოთ SELECT…INTO ბრძანება და შევქმნათ Employees ცხრილი:

```
USE EmployeeDB;

IF OBJECT_ID ('Employees', 'U') IS NOT NULL
DROP TABLE dbo.Employees;

SELECT BusinessEntityID,
  FirstName,
  LastName,
  JobTitle,
  PhoneNumber,
  EmailAddress,
  AddressLine1,
  AddressLine2,
  City,
  StateProvinceName,
  PostalCode,
  CountryRegionName
INTO dbo.Employees
FROM AdventureWorks2008.HumanResources.vEmployee;
```

მოდელის აღდგენის კონფიგურაცია

SQL-Server-ის თითოეული ბაზა შეიცავს მოდელის აღდგენის თვისებას, რომელიც თავის მხრივ განსაზღვრავს არის თუ არა შესაძლებელი ტრანზაქციის ჟურნალიდან აღდგენა. ჩვეულებრივ, ახალი მონაცემთა ბაზა მემკვიდრეობით იღებს ბაზის მოდელის აღდგენის თვისებას მოდელის ბაზიდან.თუმცა ჩვენ შეგვიძლია გადავფაროთ ეს თვისება სხვადასხვა აღდგენის მეთოდის გამოყენებით.

SQL Server -ში არსებობს სამი ტიპის აღდგენის მოდელი:

- **Full**
- **Simple**
- **Bulk-logged**

Full-ამ მოდელის გამოყენებისას ჟურნალში იწერება სრული ინფორმაცია ყველა განხორციელებულ ოპერაციაზე. ავარიული გათიშვისას მონაცემების დაკარგვა არ ხდება,(რა თქმა უნდა თუ შექმნილია მონაცემთა ფაილების და ტრანზაქციის ჟურნალის ფაილების სარეზერვო ასლები)

ამ მოდელში შესაძლებელია არა მხოლოდ სრული ასლების შექმნა, არამედ იმ ცვლილებების ასახვაც, რომელიც მოხდა ბოლო სრული სარეზერვო ასლის შექმნის შემდეგ. გარდა ამისა შესაძლებელია მოხდეს ბაზის სრული აღდგენა კონკრეტული დროის შესაბამისი მდგომარეობისათვის. ამ მოდელის შემთხვევაში ტრანზაქციის ჟურნალში ადგილი თავისუფლდება მხოლოდ მას შემდეგ, როცა უკვე შესრულდება სარეზერვო კოპირება Transaction Log და ყველა ცვლილება გადავა სარეზერვო ასლში, ხოლო მათ მიერ დაკავაბული ადგილი ჟურნალში გამოთავისუფლედება ახალი ჩანაწერებისათვის. ამისათვის მონაცემთა ბაზებს, რომელთა ექსპლუატაციაც ხდება ამ რეჟიმში, უნდა გააჩნდეთ ტრანზაქციის ჟურნალისთვის საკმარისი ადგილი, რათა მოხდეს ყველა ტრანზაქციის შენახვა, რომელიც სრულდება სარეზერვო ასლებს შორის.

არარეგისტრირებული ოპერაციები დაუშვებელია.

Simple-ეს მოდელი იდეალურია ისეთი ბაზებისათვის, რომლებშიც მოითხოვება ტრანზაქციის ატომარობის დაცვა. ამ მოდელის არსებობისას რეგულარულად ხდება ჟურნალის გასუფთავება საკონტროლო წერტილებში და ფაქტიურად იშლება ინფორმაცია ყველა დასრულებულ ტრანზაქციაზე. ინახება მხოლოდ ბაზაში ჯერ კიდევ დაუფიქსირებელი ტრანზაქცია. ასე, რომ ხდება ჟურნალის ფორმირება, თავისუფლდება ადგილი ახალი ჩანაწერებისათვის. ჟურნალის მოცულობა პატარაა. მაგრამ იგი გამოუყენებელია აღდგენისათვის სისტემის ავარიული გამორთვის შემთხვევაში.

ამგვარად, Simple მოდელში არ ხდება ჟურნალის ასლის შექმნა. შესაძლებელია მხოლოდ მოხდეს ბაზის სრული ან ნაწილობრივი (დიფერენცირებული) აღდგენა. მარტივ მოდელში იმ ცვლილებების აღდგენა , რომელიც მოხდა ბოლოს შესრულებული სრული სარეზერვო კოპირების შემდეგ შეუძლებელია.

დასაშვებია არარეგისტრირებული ოპერაციები, მაგ. მასიური კოპირებები.

Bulk-Logged-ეს მოდელი ზემოთ განხილულ ორ მოდელს შორისაა. შესაძლებელია მონაცემთა ბაზის თანმიმდევრული სარეზერვო ასლის შექმნა და ამასთანავე Transaction log მუშავდება ისევე, როგორც სრულ მოდელში. მხოლოდ ტრანზაქციის ჟურნალში არ ხდება შემდეგი ოპერაციები:

- მასიური ჩასმები;

- ინსტრუქცია SELECT *INTO
- ოპერაციები WRITETEXT UPDATEETEXT
- ინსტრუქცია GREATE INDEX

ჩვენ შეგვიძლია გადავრთოთ აღდგენის მოდელი ALTER DATABASE ბრძანების გაშვებით და შემდეგ SET RECOVERY პუნქტის არჩევით, როგორც ნაჩვენებია შემდეგ მაგალითში:

```
USE master;

ALTER DATABASE EmployeeDB
SET RECOVERY FULL;
```

როგორც ამ მაგალითიდან ჩანს ჩვენ EmployeeDB მონაცემთა ბაზას განვუსაზღვრეთ FULL აღდგენის მოდელი , რაც იმას ნიშნავს რომ იგი ავტომატურად დაკონფიგურირდება ანუ მემკვიდრეობით მიიღებს აღდგენის მოდელისგან თვისებებს.

ტრანზაქციის ძირითადი თვისებები

ტრანზაქცია ეს არის სამუშაოს ერთიანი ლოგიკური ბლოკი, რომელიც ხასიათდება მკვეთრად გამოხატული თვისებებით მაგ. ბაზაში ერთდროულად რამდენიმე სტრიქონის ჩამატება ან განახლება, წაშლა და ა. შ.

ბაზის მუშაობის ხარისხი დამოკიდებულია იმაზე, თუ რამდენად შეესაბამება მისი შესაძლებლობები ტრანზაქციის პროცესის შესრულებისას ACID პრინციპებს. ეს პრინციპებია: ატომარობა, მთლიანობა, იზოლირებულობა, მდგრადობა (Atomicity,Consistency, Isolation, Durability)

- **ატომარულობა**: უნდა შესრულდეს ტრანზაქციაში მოთავსებული ყველა მოქმედება ან არც ერთი.
- **შეთანხმებულობა:** ტრანზაქციის შესრულების შემდეგ მონაცემები უნდა იმყოფებოდეს შეთანხმებულ მდგომარეობაში, ე.ი. დაცული უნდა იყოს მთლიანობის შეზღუდვები. ტრანზაქციის დამთავრების შემდეგ მონაცემების გარე სტრუქტურები (მაგალითად, ინდექსები) ასევე უნდა იყოს კორექტულ მდგომარეობაში. შეთანხმებულობის მოთხოვნა შეიძლება დაცული იყოს როგორც ტრანზაქციის თითოეული ბრძანებისთვის, ისე მთლიანად ტრანზაქციისთვის. პირველ შემთხვევაში გარანტირებული იქნება ის, რომ ტრანზაქციის თითოეული ბრძანება არ დაარღვევს მთლიანობის შეზღუდვებს, წინააღმდეგ შემთხვევაში, შესრულდება ტრანზაქციის უკუქცევა. მეორე შემთხვევაში ტრანზაქციის შესრულების პროცესში შეიძლება დროებით დაირღვეს წესები და მთლიანობის შეზღუდვები, მაგრამ გარანტირებული იქნება ის, რომ ტრანზაქციის ფიქსირების მომენტისათვის მონაცემების მთლიანობა არ იქნება დარღვეული.
- **იზოლირებულობა :** როდესაც მონაცემთა ბაზას ერთდროულად მიმართავს მრავალი მომხმარებელი, არსებობს იმის შესაძლებლობა, რომ ერთი პიროვნების მიერ შესრულებული ცვლილებები ზემოქმედებას მოახდენენ სხვა პიროვნების მუშაობაზე. სხვადასხვა ტრანზაქციების მიერ მონაცემების ცვლილება უნდა იყოს ერთმანეთისაგან იზოლირებული. ანუ ერთი ტრანზაქციის მიერ მონაცემების ცვლილება არ უნდა იყოს დამოკიდებული მეორე ტრანზაქციის მიერ მონაცემების ცვლილებაზე. წინააღმდეგ შემთხვევაში, ორივე ტრანზაქციის მუშაობა შეიძლება დაიბლოკოს, ანუ ადგილი ჰქონდეს „მკვდარ“ ბლოკირებებს. ტრანზაქციას შეუძლია მონაცემებს მიმართოს მხოლოდ მაშინ, როცა ამ მონაცემებთან სხვა ტრანზაქცია არ მუშაობს. ამიტომ, ტრანზაქციას არ შეუძლია ნახოს სხვა ტრანზაქციის მუშაობის შუალედური შედეგები. თუ ტრანზაქცია ერთსა და იმავე მონაცემებს რამდენიმეჯერ კითხულობს, მაშინ ეს მონაცემები არ უნდა შეიცვალოს. თუ პირველი ტრანზაქცია ირჩევს გარკვეული ლოგიკური პირობის შესაბამის სტრიქონებს, მაშინ მეორე ტრანზაქციამ არ უნდა ჩასვას ამავე ლოგიკური პირობის შესაბამისი სტრიქონები.ასეთ ქცევას სერიალიზებადობა ეწოდება.მაგალითად წარმოვიდგინოთ, რომ ორი

პიროვნება ერთდროულად იმყოფებიან ავიაბილეთების შეკვეთათა ცხელი ხაზის სისტემაში (on-line flight reservation system), ორივე ხედავს, რომ ფანჯარასთან მდებარე ადგილი მე-18 რიგში ჯერ კიდევ თავისუფალია, და ორივე უკვეთავს ამ ადგილს თითქმის ერთდროულად. შესაბამისი კონტროლის გარეშე ორივე პიროვნება დარწმუნებული იქნება, რომ მათი ადგილი შეკვეთილია, მაგრამ ერთ-ერთი მათგანი იმედგაცრუებული დარჩება. მოყვანილი მაგალითი წარმოადგენს დამთხვევათა პრობლემების ერთ-ერთ სახეს, რომელსაც დაკარგულ განახლებათა პრობლემა ეწოდება (the lost update problem).

- **მდგრადობა:** ტრანზაქციის წარმატებით დამთავრების შემდეგ ხდება მისი ფიქსირება და სისტემა წინა მდგომარეობაში ვეღარ დაბრუნდება. ეს ძალაშია სისტემაში წარმოქმნილი შეფერხებების დროსაც. ცვლილებები დაფიქსირებული იქნება იმ შემთხვევაშიც თუ ტრანზაქციის დამთავრებისთანავე ადგილი ექნა კომპიუტერის, ოპერაციული სისტემის ან სერვერის შეცდომას. სერვერი, მომდევნო გაშვებისას, შეასრულებს მონაცემების აღდგენას ტრანზაქციების ჟურნალის გამოყენებით. თუ საჭიროა მონაცემების აღდგენა ტრანზაქციის შემდეგ, მაშინ აღდგენა უნდა შევასრულოთ მონაცემთა ბაზის ან ტრანზაქციების ჟურნალის სარეზერვო ასლიდან.

ოთხივე აღნიშნული მოთხოვნის დაცვას სერვერი უზრუნველყოფს. ჩვეულებრივ, სერვერი ავტომატურად ბლოკავს იმ მონაცემებს, რომლებთანაც ტრანზაქცია მუშაობს. ამიტომ, მცირე ზომის ტრანზაქციების გამოყენების შემთხვევაში სერვერის მუშაობის ეფექტურობა იზრდება დიდი ზომის ტრანზაქციების გამოყენებასთან შედარებით. დიდი ზომის ტრანზაქციების გამოყენების შემთხვევაში ხანგრძლივი დროით ხდება მონაცემების დაბლოკვა და შესაბამისად, ადგილი აქვს ხანგრძლივ მოცდენებს. მაგალითად, თუ მომხმარებლები იყენებენ შენახულ პროცედურას, რომელიც მონაცემების შეცვლას ასრულებს როგორც ერთ დიდ ტრანზაქციას, მაშინ ადგილი ექნება ხანგრძლივ მოცდენებს. ასეთი შენახული პროცედურა ისე უნდა დაიწეროს, რომ მან მონაცემების შეცვლა შეასრულოს როგორც რამდენიმე მცირე ზომის, სწრაფად შესრულებადმა და მცირე მონაცემების დამბლოკავმა

ტრანზაქციამ. ამ შემთხვევაში, დიდი რაოდენობის მომხმარებლები ეფექტურად გამოიყებენ ასეთ შენახულ პროცედურას მონაცემებთან სამუშაოდ.

მონაცემებთან მიმართვის პრობლემები და მათი გადაჭრა ტრანზაქციის პროცესში

არსებობს მონაცემებთან მიმართვის ოთხი პრობლემა:

- *უკანასკნელი შეცვლის პრობლემა.* როცა რამდენიმე მომხმარებელი ერთდროულად ცვლის ერთი სტრიქონის მონაცემებს, მაშინ მონაცემების ნაწილი იკარგება, რადგან ყოველი მომდევნო ტრანზაქცია ცვლის წინა ტრანზაქციის მიერ შესრულებულ ცვლილებებს. დავუშვათ, რომ ორმა მომხმარებელმა სერვერიდან თავის კომპიუტერში გადაწერა ერთი და იგივე ფაილის ასლი და მასში შეიტანა ცვლილებები. შემდეგ, ორივე მომხმარებელი ამ ასლს სერვერზე ინახავს. თუ ჯერ პირველი მომხმარებელი ინახავს ფაილს, შემდეგ კი - მეორე, მაშინ პირველი მომხმარებლის მიერ შეტანილი ცვლილებები დაიკარგება, რადგან შემდეგ ამ ფაილში ჩაიწერება მეორე მომხმარებლის მიერ შეტანილი ცვლილებები. ამ სიტუაციიდან გამოსავალია ცვლილებების მიმდევრობით შეტანა. ეს იმას ნიშნავს, რომ მეორე მომხმარებელმა ფაილის ასლი სერვერიდან უნდა გადაწეროს და შეცვალოს მხოლოდ მაშინ, როცა პირველი მომხმარებებლი დაამთავრებს ამ ფაილთან მუშაობას.
- *შეცდომითი კითხვის პრობლემა.* დავუშვათ, მომხმარებელი ასრულებს მონაცემების ხშირ ცვლილებას მანამ, სანამ მონაცემები გადავა ლოგიკურად სწორ მდგომარეობაში. თუ მონაცემების შეცვლის მომენტში სხვა მომხმარებელი წაიკითხავს ამ მონაცემებს, მაშინ ის მიიღებს ლოგიკურად არასწორ ინფორმაციას. ამის თავიდან ასაცილებლად, მონაცემების წაკითხვა უნდა შესრულდეს მხოლოდ მათი დამუშავების შემდეგ.

- *არაგამეორებადი კითხვის პრობლემა.* ეს პრობლემა წამოიჭრება მაშინ, როცა ტრანზაქცია რამდენიმეჯერ კითხულობს ერთსა და იმავე მონაცემებს. პირველი ტრანზაქციის შესრულების დროს თუ მეორე ტრანზაქციამ მონაცემები შეცვალა, მაშინ

პირველი ტრანზაქცია მონაცემების განმეორებით წაკითხვისას სხვა მნიშვნელობებს მიიღებს.

- *ფანტომების კითხვის პრობლემა.* ამ პრობლემას ადგილი აქვს მაშინ, როცა პირველი ტრანზაქცია ცხრილიდან ირჩევს სტრიქონებს, მეორე ტრანზაქცია კი ახდენს ამავე ცხრილში სტრიქონების ჩასმას პირველი ტრანზაქციის მუშაობის დამთავრებამდე. შედეგად, პირველი ტრანზაქციის მიერ წასაკითხი სტრიქონები არაკორექტული იქნება. თუ თავისი მუშაობის დროს ერთი ტრანზაქცია კითხულობს ჩანაწერების რომელიმე სიმრავლეს ორჯერ, მაშინ მან შესაძლებელია წაიკითხოს ახალი ჩანაწერები მეორე წაკითხვისას. ეს შესაძლებელია მოხდეს იმ შემთხვევაში, თუ პარალელურად მონაცემთა ბაზაში სხვა ტრანზაქციას ახალი ჩანაწერები შეაქვს

აღნიშნული პრობლემების გადასაწყვეტად სტანდარტიზების ინსტიტუტმა შეიმუშავა სპეციალური ANSI სტანდარტი, რომელიც განსაზღვრავს დაბლოკვის 4 დონეს. ყოველი მომდევნო დონე უზრუნველყოფს წინა დონის მოთხოვნებს და მას დამატებით შეზღუდვებს ადებს. თითოეულ დონეს თავისი ნომერი აქვს. პირველი დონის ნომერია 0. ნულოვანი დონე არის დაბლოკვის ყველაზე ნაკლებად მკაცრი დონე და მოითხოვს სისტემური რესურსების მინიმუმს. ეს დონეებია:

- დონე 0. მონაცემების „დაბინძურების“ აკრძალვა. ეს დონე ითხოვს, რომ მხოლოდ ერთ ტრანზაქციას შეეძლოს მონაცემების შეცვლა. თუ მეორე ტრანზაქციას უნდა ამ მონაცემების შეცვლა, მაშინ ის უნდა დაელოდოს პირველი ტრანზაქციის დამთავრებას.
- დონე 1. „შეცდომითი“ კითხვის აკრძალვა. თუ ტრანზაქციამ დაიწყო მონაცემების შეცვლა, მაშინ სხვა ტრანზაქციებს არ შეეძლებათ ამ მონაცემების წაკითხვა მანამ, სანამ პირველი ტრანზაქცია არ დამთავრდება.

- დონე 2. „არაგამეორებადი“ კითხვის აკრძალვა. თუ ტრანზაქცია კითხულობს მონაცემებს, მაშინ სხვა ტრანზაქციას არ შეუძლია ამ მონაცემების შეცვლა. შედეგად, როცა პირველი ტრანზაქცია მონაცემებს განმეორებით წაიკითხავს, მათ იგივე მნიშვნელობები ექნება.
- დონე 3. ფანტომების აკრძალვა. თუ ტრანზაქცია მონაცემებს მიმართავს, მაშინ სხვა ტრანზაქცია ვერ შეცვლის ამ მონაცემებს. ბლოკირების ამ დონის რეალიზება სრულდება გასაღებების დიაპაზონის ბლოკირებების გამოყენებით. იბლოკება ცხრილის ის სტრიქონები, რომლებიც ლოგიკურ პირობას აკმაყოფილებენ.

ტრანზაქცია განისაზღვრება შეერთების დონეზე. შეერთების დახურვის დროს ავტომატურად იხურება ტრანზაქციაც. არსებობს სამი სახის ტრანზაქცია: აშკარა, ავტომატური და არააშკარა (ნაგულისხმევი).

ტრანზაქცია არის ლოკალური თუ ის სრულდება ერთი მონაცემთა ბაზის შიგნით. განაწილებული არის ტრანზაქცია, რომელიც მიმართავს რამდენიმე მონაცემთა ბაზას, რომლებიც შეიძლება სხვადასხვა სერვერებზე იყოს განთავსებული.

აშკარა ტრანზაქციები

აშკარა ტრანზაქციების გამოყენების შემთხვევაში აშკარად უნდა მიუთითოთ ტრანზაქციის დაწყება და დამთავრება. ამისათვის, გამოიყენება შემდეგი ბრძანებები: BEGIN TRANSACTION, COMMIT და ROLLBACK. BEGIN TRANSACTION ბრძანება განსაზღვრავს ტრანზაქციის დასაწყისს. ამ დროს ტრანზაქციების ჟურნალში ფიქსირდება შესაცვლელი მონაცემების საწყისი მნიშვნელობები და ტრანზაქციის დაწყების მომენტი.

ავტომატური ტრანზაქციები

ჩვეულებრივ, სერვერი მუშაობს ტრანზაქციის ავტომატურად დაწყების რეჟიმში (autocommit transaction). ამ რეჟიმში თითოეული ბრძანება განიხილება როგორც ცალკეული ტრანზაქცია. თუ ბრძანება წარმატებით შესრულდა, მაშინ მის მიერ შესრულებული ცვლილებები ფიქსირდება. თუ ბრძანების შესრულების დროს ადგილი ექნა შეცდომას, მაშინ შესრულებული ცვლილებები უქმდება და სისტემა უბრუნდება საწყის მდგომარეობას.
თუ საჭიროა ტრანზაქციის შექმნა, რომელიც რამდენიმე ბრძანებას შეიცავს, მაშინ ტრანზაქცია აშკარად უნდა მივუთითოთ.
ჩვეულებრივ, სერვერი მუშაობს ტრანზაქციის განსაზღვრის ავტომატურ ან არააშკარა. რეჟიმში. სერვერი არ შეიძლება იმყოფებოდეს ტრანზაქციის მხოლოდ აშკარა განსაზღვრის რეჟიმში.
ტრანზაქციის ავტომატური განსაზღვრის რეჟიმის დასაყენებლად გამოიყენება ბრძანება:
SET IMPLICIT_TRANSACTION OFF

არააშკარა ტრანზაქციები

ტრანზაქციის არააშკარა (ნაგულისხმევი) დაწყების (implicit transaction) რეჟიმში მუშაობის დროს სერვერი ავტომატურად იწყებს ახალ ტრანზაქციას, როგორც კი დამთავრდება წინა. მომხმარებელი არაფერს არ უთითებს ტრანზაქციის დასაწყებად. ტრანზაქცია გრძელდება მანამ, სანამ მომხმარებელი აშკარად არ მიუთითებს ტრანზაქციის უკუქცევის ან ფიქსირების ბრძანებას. ამის შემდეგ, სერვერი ავტომატურად იწყებს ახალ ტრანზაქციას.
მას შემდეგ, რაც შეერთებისათვის დაყენებულია ტრანზაქციის არააშკარა დაწყების რეჟიმი, სერვერი ავტომატურად ამთავრებს მიმდინარე ტრანზაქციას და იწყებს ახალს, თუ გვხვდება შემდეგი ბრძანებებიდან ერთ-ერთი:

- ALTER TABLE - ცხრილის სტრუქტურის შეცვლა;
- CREATE - მონაცემთა ბაზის ობიექტის შექმნა;
- DELETE - ცხრილიდან სტრიქონების წაშლა;
- DROP - მონაცემთა ბაზის ობიექტის წაშლა;
- FETCH - კურსორიდან მითითებული სვეტის მნიშვნელობის მიღება;
- GRANT - მონაცემთა ბაზის ობიექტთან მიმართვის ნების დართვა;

- INSERT - ცხრილში სტრიქონების დამატება;
- OPEN - კურსორის გახსნა;
- REVOKE - მონაცემთა ბაზის ობიექტებთან მიმართვის არააშკარა გადახრა (отклонение);
- SELECT - ცხრილიდან მონაცემების ამირჩევა;
- TRUNCATE TABLE - ცხრილის ჩამოჭრა;
- UPDATE - ცხრილში მონაცემების შეცვლა.

ტრანზაქციის არააშკარა განსაზღვრის რეჟიმის დასაყენებლად გამოიყენება ბრძანება:
SET IMPLICIT_TRANSACTION ON

განაწილებული ტრანზაქციები

როცა საჭიროა სხვადასხვა მონაცემთა ბაზაში მოთავსებულ რესურსებთან მიმართვა, აუცილებელია განაწილებული ტრანზაქციის (distributed transaction) გამოყენება. განაწილებული ტრანზაქცია წარმოადგენს რამდენიმე ცალკეულ ტრანზაქციას, რომლებიც ლოკალურად სრულდება ცალკეულ მონაცემთა ბაზაში.
მონაცემების წყაროს, რომელსაც მომხმარებელი მიმართავს, რესურსების მენეჯერი (resource manager) ეწოდება. რესურსების მენეჯერი ასრულებს ლოკალური ტრანზაქციების ფიქსირებას, უკუქცევასა და რესურსების სხვა მენეჯერებთან თავისი მოქმედებების კოორდინირებას. სერვერი იყენებს X/OPEN XA სპეციფიკაციას განაწილებული მოთხოვნების დამუშავებისათვის (distributed transaction processing). თუ მონაცემთა ბაზის მართვის სისტემა უზრუნველყოფს ამ სპეციფიკაციას, მაშინ ის შეიძლება გამოყენებული იყოს სერვერის განაწილებულ ტრანზაქციებში. შედეგად, განაწილებული მოთხოვნების შესრულების დროს შეგვიძლია მივმართოთ არა მარტო SQL სერვერებს, არამედ მონაცემების ისეთ წყაროებს, როგორიცაა Oracle, Access, ODBC და ა.შ.
განაწილებული ტრანზაქციების ფიქსირება და უკუქცევა იმართება ტრანზაქციების მენეჯერის (transaction manager) მიერ. ის ახდენს ლოკალური რესურსების მენეჯერების მუშაობის კოორდინირებას და იძლევა იმის გარანტიას, რომ ყველა ლოკალური ტრანზაქცია

დაფიქსირებული ან უკუქცეული იქნება. არ შეიძლება ტრანზაქციის ერთი ნაწილის ფიქსირება და მეორე ნაწილის უკუქცევა. სერვერზე ტრანზაქციების მენეჯერად გამოიყენება განაწილებული ტრანზაქციების კოორდინატორი , ის იწყებს და ამთავრებს ლოკალურ ტრანზაქციებს, აგრეთვე ახდენს მათ უკუქცევას.

განაწილებული ტრანზაქციების მუშაობის ყველაზე რთული ეტაპი დაკავშირებულია მათ დამთავრებასთან. ზოგიერთ განაწილებულ ტრანზაქციას, რომელიც მიმართავს რესურსების რამდენიმე მენეჯერს და ცვლის მონაცემების დიდ რაოდენობას, შეიძლება დასჭირდეს დიდი დრო თავისი ფიქსირებისათვის.

განაწილებული ტრანზაქციების კოორდინატორმა უნდა უზრუნველყოს ყველა ლოკალური ტრანზაქციის ფიქსირება ან უკუქცევა, იმ შემთხვევაშიც კი, როცა ადგილი აქვს სერვერის ან ქსელის შეცდომას. ამისათვის, სერვერზე ტრანზაქციის დამთავრების ორფაზიანი პროტოკოლი (2PC) გამოიყენება. ის მომზადების ფაზისა და დამთავრების ფაზისაგან შედგება:

- მომზადების ფაზა (Prepare phase). როცა განაწილებული ტრანზაქციების კოორდინატორი იღებს განაწილებული ტრანზაქციის დამთავრების მოთხოვნას, ის ლოკალური ტრანზაქციების მენეჯერებს უგზავნის ლოკალური ტრანზაქციების დამთავრების მოთხოვნას. ტრანზაქციების მენეჯერები ასრულებენ ყველა აუცილებელ მოქმედებას ცვლილებების ფიქსირებისათვის და განაწილებული ტრანზაქციების კოორდინატორს უგზავნიან უწყებას ტრანზაქციის წარმატებით ან წარუმატებლად დამთავრების შესახებ. ტრანზაქციის შესრულების ფაქტი ამ ეტაპზე არ ფიქსირდება.
- დამთავრების ფაზა (Commit phase). იღებს რა ტრანზაქციების ყველა ლოკალური მენეჯერისაგან შეტყობინებას მონაცემების წარმატებით ფიქსირების შესახებ, განაწილებული ტრანზაქციების კოორდინატორი უგზავნის მათ ტრანზაქციის ფიქსირების მოთხოვნას. თუ ერთმა მენეჯერმა ვერ შეძლო მომზადების ფაზის შესრულება ან ტრანზაქციის დაფიქსირება, მაშინ განაწილებული ტრანზაქციების კოორდინატორი ყველა ლოკალურ მენეჯერს უგზავნის მოთხოვნას ტრანზაქციების უკუქცევის შესახებ და გასცემს შეტყობინებას შეცდომის შესახებ.

არსებობს განაწილებული ტრანზაქციის დაწყების რამდენიმე გზა:

- თუ პროგრამა-დანართი ლოკალურ ტრანზაქციაში იყენებს განაწილებულ მოთხოვნას მონაცემებთან მიმართვისათვის, მაშინ სერვერი ავტომატურად იწყებს განაწილებული ტრანზაქციის შესრულებას.
- თუ პროგრამა-დანართი იწყებს ლოკალურ ტრანზაქციას და მისგან იძახებს დაშორებულ შენახულ პროცედურას დაყენებული REMOTE_PROC_TRANSACTION პარამეტრის შემთხვევაში, მაშინ ეს ტრანზაქცია ავტომატურად ფართოვდება განაწილებულ ტრანზაქციამდე.
- პროგრამა-დანართს განაწილებული ტრანზაქციის დაწყება შეუძლია OLE DB-ის მეთოდების ან ODBC-ის ფუნქციების გამოყენებით.
- სერვერი იწყებს განაწილებული ტრანზაქციის შესრულებას BEGIN DISTRIBUTED TRANSACTION ბრძანების საშუალებით.

ჩადგმული ტრანზაქციები

ჩადგმული ტრანზაქციები (nested transaction) ეწოდება ტრანზაქციებს, რომელთა შესრულება ინიცირდება (იწყება) აქტიური ტრანზაქციის კოდიდან.

ჩადგმული ტრანზაქციების გამოყენება მისაწვდომია მხოლოდ აშკარა ტრანზაქციებში. ავტომატური ან არააშკარა ტრანზაქციების შესრულებისას, ახალი ტრანზაქციის დაწყებამდე, წინა ტრანზაქცია უნდა იყოს დამთავრებული. ამიტომ, ამ ტრანზაქციებიდან შეუძლებელია ჩადგმული ტრანზაქციების ინიცირება. ჩადგმული ტრანზაქციების ძირითადი დანიშნულებაა იმ ტრანზაქციების უზრუნველყოფა, რომლებიც სრულდებიან ჩადგმული პროცედურების მიერ. ჩვენ შეგვიძლია მივმართოთ შენახულ პროცედურას როგორც უკვე დაწყებული ტრანზაქციიდან, ისე უშუალოდ (არა ტრანზაქციის კოდიდან). სერვერმა ორივე შემთხვევაში უნდა უზრუნველყოს მონაცემების მთლიანობა და ACID მოთხოვნების დაცვა.
ჩადგმული ტრანზაქციის შესაქმნელად უნდა დავიწყოთ ახალი ტრანზაქცია წინა ტრანზაქციის დახურვის გარეშე. ზედა დონის ტრანზაქციის დამთავრება გადაიდება ჩადგმული ტრანზაქციების დამთავრებამდე. თუ ყველაზე დაბალი დონის ტრანზაქცია წარუმატებლად დამთავრდა და უკუქცეულია, მაშინ ზედა დონის ყველა ტრანზაქცია

უკუიქცევა პირველი დონის ტრანზაქციის ჩათვლით. გარდა ამისა, თუ ქვედა დონის რამდენიმე ტრანზაქცია წარმატებით დამთავრდა (მაგრამ არ დაფიქსირდა), მაგრამ საშუალო დონეზე ტრანზაქცია წარუმატებლად დამთავრდა, მაშინ უკუიქცევა ყველა დონის ტრანზაქცია იმ ტრანზაქციების ჩათვლით, რომლებიც წარმატებით დამთავრდა. როცა ყველა დონის ტრანზაქცია წარმატებით დამრთავდება, მაშინ ყველა ცვლილება დაფიქსირდება იმ შემთხვევაში, როცა წარმატებით დამთავრდება პირველი დონის ტრანზაქცია.

თითოეული COMMIT TRANSACTION ან COMMIT WORK ბრძანება მუშაობს მხოლოდ უკანასკნელად დაწყებულ ტრანზაქციასთან. ჩადგმული ტრანზაქციის დამთავრებისას COMMIT ბრძანება გამოიყენება ყველაზე „ღრმად|| ჩადგმული ტრანზაქციის მიმართ. იმ შემთხვევაშიც კი, თუ COMMIT TRANSACTION ბრძანებაში ტრანზაქციის_სახელი მიმართავს უფრო მაღალი დონის ტრანზაქციას, ჯერ მაინც დამთავრდება უკანასკნელად დაწყებული ტრანზაქცია.

თუ ROLLBACK TRANSACTION ბრძანება გამოიყენება ჩადგმულობის ნებისმიერ დონეზე ტრანზაქციის სახელის მითითების გარეშე ან გამოიყენება ROLLBACK WORK ბრძანება, მაშინ უკუიქცევა ყველა ჩადგმული ტრანზაქცია, ყველაზე ზედა (პირველი) დონის ტრანზაქციის ჩათვლით. ROLLBACK TRANSACTION ბრძანებაში დასაშვებია მხოლოდ ზედა დონის ტრანზაქციის სახელის მითითება. ნებისმიერი ჩადგმული ტრანზაქციის სახელი იგნორირდება და მისი მითითება გამოიწვევს შეცდომას. ამრიგად, ნებისმიერი დონის ტრანზაქციის უკუქცევისას ნებისმიერ შემთხვევაში ხდება ყველა ტრანზაქციის უკუქცევა. თუ საჭიროა ტრანზაქციების ნაწილის უკუქცევა, შეგვიძლია გამოვიყენოთ SAVE TRANSACTION ბრძანება, რომლის საშუალებითაც იქმნება შენახვის წერტილი. მოგვიანებით შეგვეძლება სისტემის აღდგენა იმ მდგომარეობაში, რომელშიც ის იყო SAVE TRANSACTION ბრძანების შესრულების მომენტში.

ჩადგმული ტრანზაქციის უკეთ დემონსტრირებისთვის შევქმნათ ცხრილი people

```
CREATE TABLE People
(
    Name VARCHAR(30)
)
```

შემდეგ შევქმნათ ტრანზაქცია რომლის შიგნით გამოვიძახებთ მეორე ტრანზაქციას. ტრანზაქციით People ცხრილში ჩავწეროთ სტრიქონი „tom“ , შემდეგ ჩადგმიული ტრანზაქციის საშუალებით ჩავწეროთ სხვა სტრიქონი სახელწოდებით „dick“.

```
BEGIN TRAN

INSERT INTO People VALUES ('Tom')

BEGIN TRAN

INSERT INTO People VALUES ('Dick')

ROLLBACK TRAN

COMMIT TRAN
```

მაგრამ ჩვენ თუ ამ პროცედურას გავუშვებთ , იგი დაგვიწერს შეცდომას:

```
The COMMIT TRANSACTION request has no corresponding BEGIN TRANSACTION.
```

შეცდომა დაფიქსირდა იმიტომ რომ ბოლო სტრიქონი “commit tran” -ში დაფიქსირდა შეცდომა, ვინაიდან გარე ტრანზაქცია აღარ არსებობს. ეს იმიტომ მოხდა რომ ჩადგმულ ტრანზაქციაში rollback ბრძანება აბრუნებს ყველა აქტიურ ტრანზაქციას.

ამ შემთხვევის თავიდან ასაცილებლად გამოვიყენოთ @@TRANCOUNT ცვლადი, რომელიც აბრუნებს აქტიური ტრანზაქციების რიცხვს. ნულოვანი მნიშვნელობა მიუთითებს რომ აქტიური ტრანზაქცია არ არსებობს.

გავუშვათ შემდეგი სკრიპტი, რომელიც შეცდომას აღარ დააფიქსირებს:

```
SET NOCOUNT ON

BEGIN TRAN
PRINT 'First Transaction: ' + CONVERT(VARCHAR,@@TRANCOUNT)

INSERT INTO People VALUES ('Tom')

BEGIN TRAN
PRINT 'Second Transaction: ' + CONVERT(VARCHAR,@@TRANCOUNT)

INSERT INTO People VALUES ('Dick')

ROLLBACK TRAN
PRINT 'Rollback: ' + CONVERT(VARCHAR,@@TRANCOUNT)

/* MESSAGES

First Transaction: 1
Second Transaction: 2
Rollback: 0

*/
```

ამ პრობლემის გადაწყვეტა ასევე შეგვეძლო SAVEPOINT (საკონტროლო წერტილების) ჩასმით . ჩვენ შეგვიძლია შევქმნათ მრავალი SAVEPOINT ტრანზაქციებისთვის და შემდეგ ინდივიდუალურად დავაბრუნოთ უკან. მაგალითად , თუ შევქმნით საკონტროლო წერტილებს სახელწოდებით „S1“, “S2” და “S3”. S2-ის უკან დაბრუნება გამოიწვევს S3 -ის წაშლას. S1 დარჩება აქტიური.

ქვემოთ მოყვანილი მაგალითის შესრულების შემდეგ დავინახავთ , რომ ცხრილში „tom“ სტრიქონი ჩაემატება, ხოლო “dick“ არა, რადგან სრულდება მხოლოდ ერთი ტრანზაქცია:

```
SET NOCOUNT ON

BEGIN TRAN
PRINT 'First Transaction: ' + CONVERT(VARCHAR,@@TRANCOUNT)

INSERT INTO People VALUES ('Tom')

SAVE TRAN Savepoint1
PRINT 'Second Transaction: ' + CONVERT(VARCHAR,@@TRANCOUNT)

INSERT INTO People VALUES ('Dick')

ROLLBACK TRAN Savepoint1
PRINT 'Rollback: ' + CONVERT(VARCHAR,@@TRANCOUNT)

COMMIT TRAN
PRINT 'Complete: ' + CONVERT(VARCHAR,@@TRANCOUNT)

/* MESSAGES

First Transaction: 1
Second Transaction: 1
Rollback: 1
Complete: 0

*/
```

მონაცემთა ბაზებში ტრანზაქციის მართვის მხარდამჭერი სისტემა

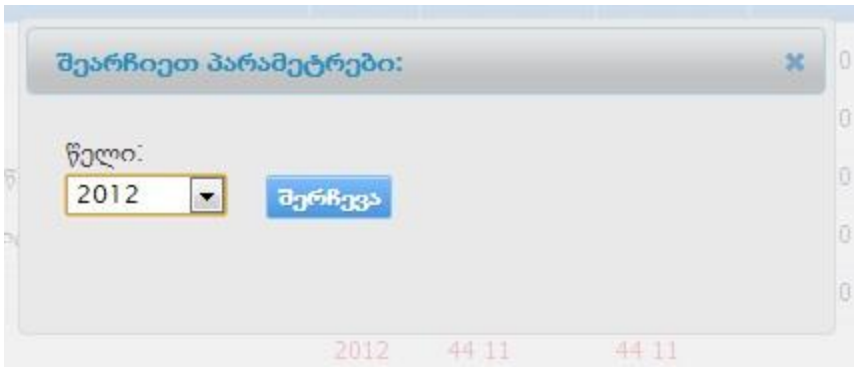
მონაცემთა ბაზის გამოყენების არეალის ზრდა და მის დამუშავებასთან დაკავშირებული პროცესების მართვის სირთულე, მწვავედ აყენებს საკითხს შეიქმნას ისეთი მხარდამჭერი, სისტემა, რომელიც არაპროფესიონალ მომხმარებელს დაეხმარება უზრუნველყოს საკუთარი მონაცემთა ბაზის ტრანზაქციის მართვა.

პროექტის მიზანია შეიქმნას ისეთი პროგრამული უზრუნველყოფა, რომელიც უზრუნველყოფს ერთი ბაზიდან (MySql) მონაცემების გადატანას მეორე ბაზაში(SQL Server) ტრანზაქციის მეთოდის საშუალებით.

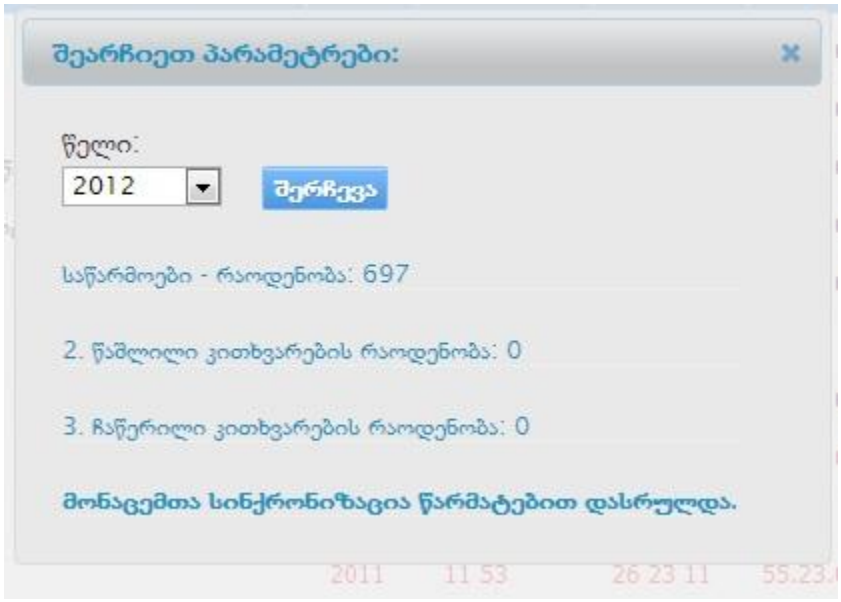
სისტემის შესაქმნელად უნდა შემუშავდეს მოდელი, რომელიც სრულყოფილად აღწერს ყველაზე ხშირად გამოყენებულ ოპერაციებს ტრანზაქციის პროცესში. მაგალითისათვის განვიხილოთ ონლაინ კითხვარების გადატანა სასტუმროების წლიური კვლევის მაგალითზე.

მოცემულ სურათზე წარმოდგენილი სინქრონაზიის ღილაკზე დაჭერისას , კითხვარები გადადის SQL Server-ის ბაზაში.

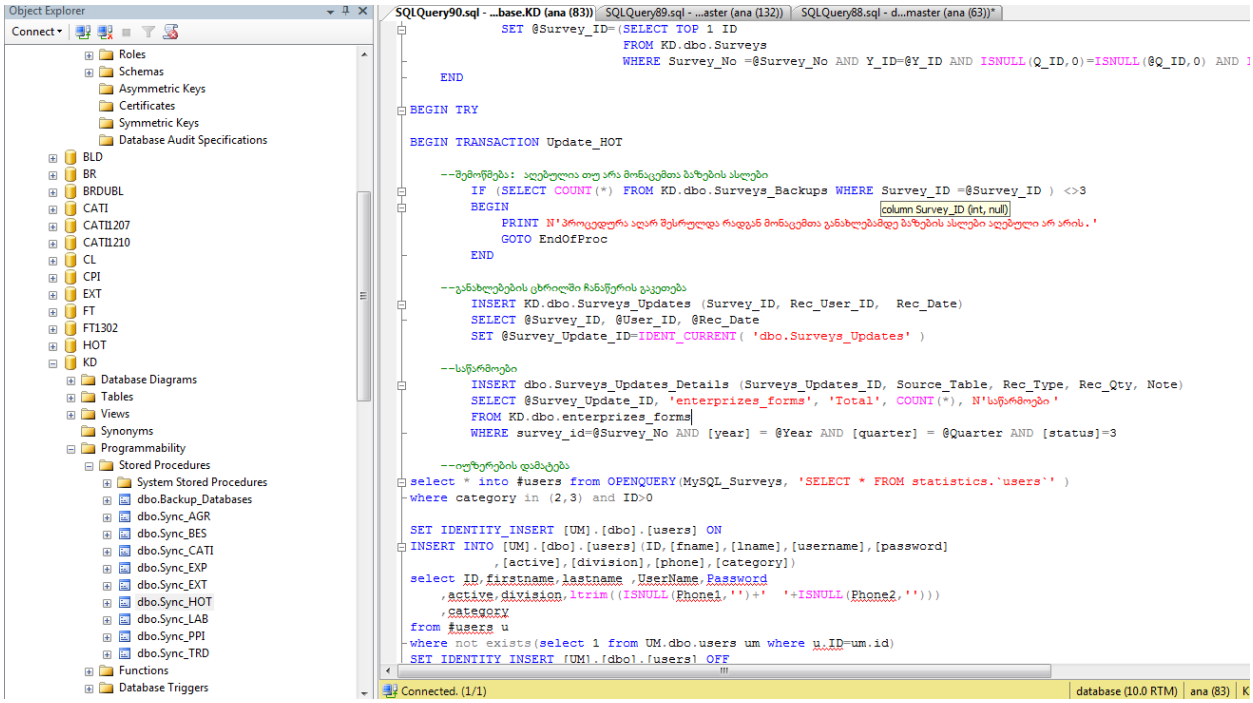
თავიდან ხდება პერიოდის არჩევა:



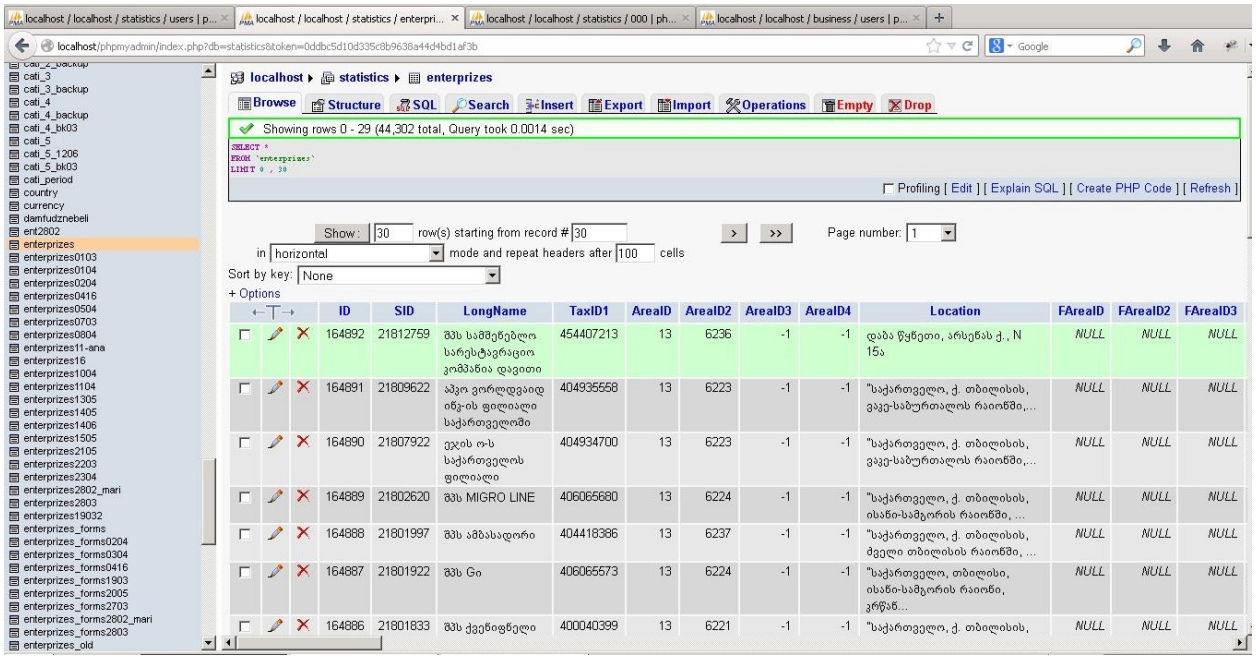
და თუ სინქრონიზაციის პროცედურამ სწორად იმუშავა , გამოვა შემდეგი ფანჯარა:



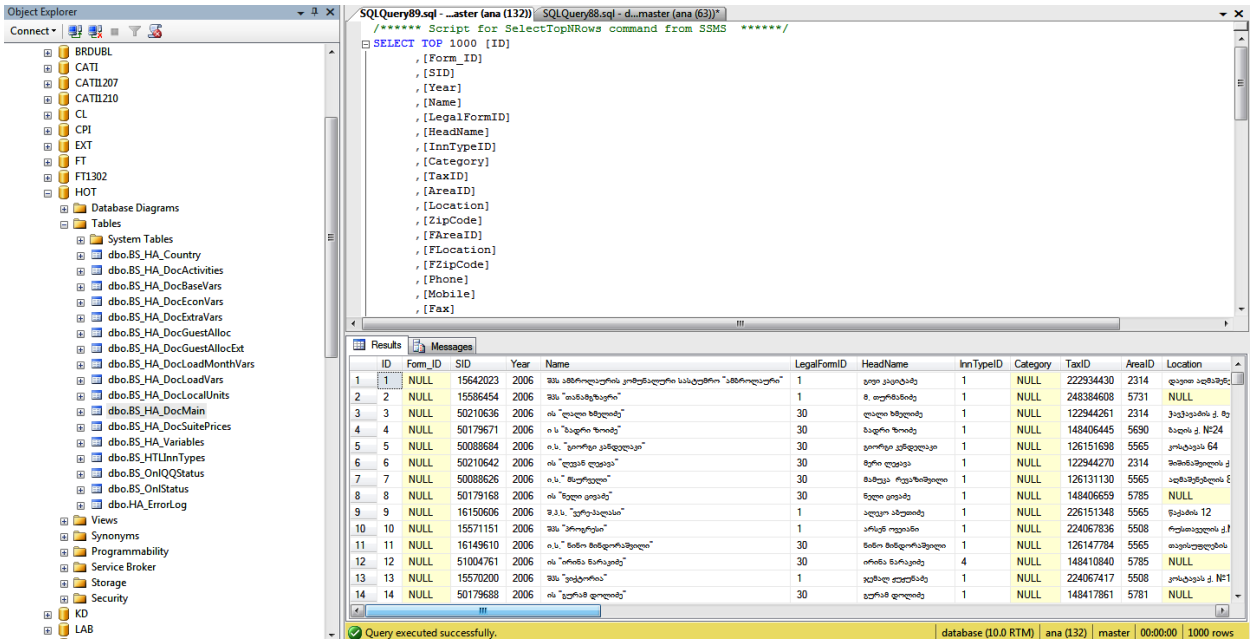
თუ მოხდა შეცდომა კითხვარების პორტირებისას ამუშავდება ტრანზაქციის ბლოკი, რომელიც შეცდომის წერტილიდან უკან დააბრუნებს ყველა ჩატარებულ პროცედურას, შედეგად მონაცემთა ბაზაში ვუზრუნველყოფთ მონაცემების სისწორესა და მთლიანობას. ტრანზაქციის პროცედურა SQL Server-ზე:



MySql ბაზაში არსებული მონაცემები:



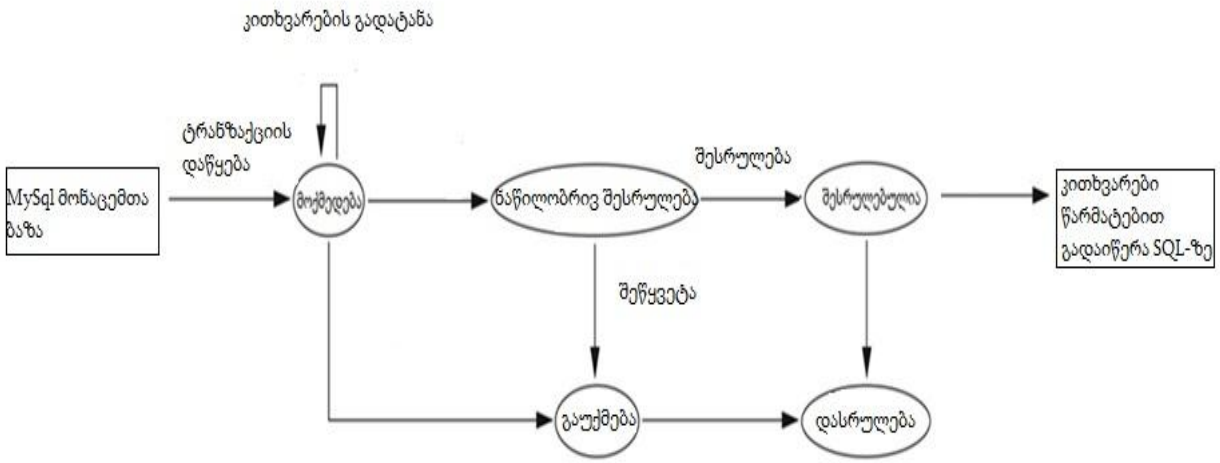
Sql Server-ზე გადაწერისას მიიღებს სახეს:



სისტემა წარმოდგება ფუნქციონალური ბლოკის საშუალებით:

1. ტრანზაქციული ოპერაციების ბაზა
2. მეთოდები , რომლებიც უზრუნველყოფენ ტრანზაქციის პროცესს
3. სისტემაში არსებული პროცედურა, რომელშიც განსაზღვრულია ტრანზაქციის პროცესი.

მოცემული ესკიზური სქემა ასახავს კითხვარების გადატანის მიმდინარე პროცესს:



დასკვნა

ნაშრომში შემუშავებულია ისეთი მხარდამჭერი სისტემის მოდელი, რომელიც ნებისმიერ მომხმარებელს, არის ის მონაცემთა ბაზის მართვის პროფესიონალი თუ არა, საშუალებას აძლევს მართოს ტრანზაქციის პროცესი, უკეთ გაერკვეს მის თავისებურებებში და გაიაზროს ტრანზაქციის მეთოდის მნიშვნელობა მონაცემთა ბაზის მთლიანობის დაცვაში.

ასევე ნაშრომში მოყვანილი ტრანზაქციის პროცესთან დაკავშირებული პრობლემების გადაჭრის გზები, მომხმარებელს უადვილებს თავიდან აიცილოს ნებისმიერი გამონაკლისი სიტუაცია, რომელიც შეიძლება წარმოიქმნას ტრანზაქციის პროცედურის მართვისას.

წარმოდგენილი ლოგიკური სქემა კი მომხმარებელს უადვილებს ტრანზაქციის პროცესის უკეთ აღქმას.

ლიტერატურა

Philip A. Bernstein, Eric Newcomer (2009): Principles of Transaction Processing, 2nd Edition, Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4

Gerhard Weikum, Gottfried Vossen (2001), Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery, Morgan Kaufmann, ISBN 1-55860-508-8

Database Transaction Models for Advanced Applications (The Morgan Kaufmann Series in Data Management Systems), by Ahmed K. Elmagarmid, Marek Rusinkiewicz and Amit Sheth (Oct 27, 1998)

http://www.tutorialspoint.com/sql/sql-transactions.htm

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

http://en.wikipedia.org/wiki/Transaction_processing#Description

http://wiki.answers.com/Q/Describe_data_base

http://www.essays.org/index.php?Page=Detail&Esej=3282&Title=Database+is+important+in+our+daily+files