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

ანა სიხარულიძე

რელაციური მონაცემთა მოდელი
მონაცემთა ბაზის დიზაინის ნორმალიზება და ეფექტურობა

სამაგისტრო პროგრამა: ინფორმაციული ტექნოლოგიები

სამაგისტრო ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების
მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მანანა ხაჩიძე

მაია არჩუაძე

თბილისი, 2013

ანოტაცია

სამაგისტრო ნაშრომის მიზანს წარმოადგენს განიხილოს და უკეთ შეისწავლოს ისეთი აქტუალური საკითხი, როგორიცაა მონაცემთა ბაზის დიზაინის ნორმალიზება და მისი ეფექტურობა. ნაშრომში განხილულია მონაცემთა ბაზის დიზაინის ნორმალიზაციისთვის საჭირო წესები, მითითებები და სხვადასხვა მიდგომები რელაციასთან დაკავშირებით. ასევე ამ წესების დაცვასთან და რეალიზებასთან დაკავშირებული ძირითადი პრობლემები. კვლევის მნიშვნელოვან საკითხს წარმოადგენს რელაცია და ნორმალიზაცია. აღწერილია ამ ტიპის მონაცემთა ბაზის რელაციასთან დაკავშირებული პრობლემები და მათი გადაჭრის მეთოდები. მათ საფუძველზე შექმნილია მომხმარებლის მხარდამჭერი სისტემა, რომელიც დაეხმარება ადმინისტრატორს შექმნას სწორი დიზაინის მქონე მონაცემთა ბაზები.

Annotation

The masters paper aims to discuss and better examine such topical issues as normalization of database design and its effectiveness. The paper discusses the rules, guidelines and various approaches to the relation of database normalization design, the problems related to follow and realize these rules. The important issue is the relation and normalization. described solutions of problems of relation. they are base of user support system, which will help administrators to create correct design of databases.

ს ა რ ჩ ე ვ ი

შესავალი

დღეისთვის მონაცემთა ბაზები ფართოდ არის გავრცელებული და ისინი ფაქტობრივად ყველგან გამოიყენება. თითქმის ყველა გამოყენებით პროგრამას გააჩნია მონაცემთა ბაზებზე დაფუძნებული სტრუქტურა. როდესაც ჩვენ რაიმეს ვყიდულობთ ინტერნეტის საშუალებით “on-line”- რეჟიმში, ვკითხულობთ თვითმფრინავთა ფრენის განრიგს, ჩვენ ვიყენებთ პროგრამებს რომლებიც მონაცემთა ბაზებზეა დაფუძნებული. მონაცემთა ბაზები უზრუნველყოფენ მონაცემთა შენახვის ეფექტურ, უსაფრთხო და მოქნილ ხერხს და გამოიყენებიან სხვადასხვა მიმართულების პროგრამებში: ბიბლიოთეკის ბარათების კატალოგებიდან დაწყებული ქარხნების დაზგების ავტომატიზაციით დამთავრებული.

1950-იანი წლების ბოლოს, როდესაც მეორე თაობის კომპიუტერები გაჩნდა, მაღალი დონის პროგრამირების ენებისა და დიდი მოცულობის დამაგროვებლების გამოყენებამ ძალიან სწრაფად გახადა შესაძლებელი მონაცემთა დიდი კრებულების შექმნა.თავიდან მონაცემები ინახებოდა ცალკეულ ფაილებში, მაგრამ საკმაოდ სწრაფად ცხადი გახდა, რომ ასეთი მიდგომა ქმნიდა მრავალ სიძნელეს. პირველ რიგში, ფაილები რაც უფრო დიდი მოცულობის ხდებოდნენ მით უფრო დიდ დროს მოითხოვდნენ მომხმარებლისათვის საჭირო მონაცემთა ძიებისათვის.

1960-იანი წლების ბოლოს პრობლემების გადასაჭრელად შეიქმნა მონაცემთა ბაზების სისტემები. მონაცემთა ბაზების პირველი სისტემები წარმოადგენდნენ იერარქიული და ქსელური ტიპის სისტემებს. იერარქიული და ქსელური ტიპის მონაცემთა ბაზების სტრუქტურები უზრუნველყოფდნენ როგორც მომხმარებლისათვის საინტერესო ინფორმაციის სწრაფ წვდომას, ასევე მონაცემთა მარტივი სახით განახლებას და უსაფრთხო შენახვას. მაგრამ ისინი საკმაოდ რთულ სტრუქტურებს წარმოადგენდნენ. ამასთან ერთად ნებისმიერი ასეთი სტრუქტურა საკმაოდ ძლიერად იყო დამოკიდებული კონკრეტული ფაილური სისტემის იმპლემენტაციის დეტალებზე, რაც მათ საკმაოდ ხისტ სახეს ანიჭებდა.

1970 წელს IBM-ის თანამშრომელმა ე. კოდმა შეიმუშავა მონაცემთა ბაზების რელაციური მოდელი. რალაციური მოდელი მკაცრად ეფუძნებოდა მათემატიკურ თეორიას. იმ დროისათვის ამ მოდელს არარეალურ მიდგომად მიიჩნევდნენ, რადგანაც მონაცემების შენახვა მოდელში უნდა მომხდარიყო ცხრილებში. ყოველ ცხრილში განთავსდებოდა

ინფორმაცია მხოლოდ ერთი ლოგიკური ობიექტის შესახებ, და ეს ობიექტები ერთმანეთთან დაკავშირებული იქნებოდა არა გარე მაჩვენებლებისა და სხვა ხერხებით არამედ ამ ცხრილებშივე განთავსებული ინფორმაციის საშუალებით. კოდმა ასევე ჩამოაყალიბა მონაცემთა წვდომის ენა, რომელიც დაფუძნებული იყო სიმრავლეთა თეორიაზე მოგვიანებით, 1980-ან წლებში ფირმა IBM-ი მსოფლიოს წარუდგენს სტრუქტურირებული მიმართვების ენას (Structured Query Language = SQL)). იმ დროის პროფესიონალთა უმეტესობის აზრით კოდის იდეების ეფექტური განხორციელება შეუძლებელ ამოცანას წარმოადგენდა.

დღეისათვის მონაცემთა რელაციური მოდელი წარმოადგენს ყველაზე უფრო გავრცელებულ მოდელს. ობიექტებზე ორიენტირებული პროგრამირების განვითარებას მოჰყვა მონაცემთა მართვის ახალი სქემების განვითარება, რომელთაც მონაცემთა ბაზების ობიექტურ-რელაციური ან ობიექტებზე ორიენტირებული მართვა ეწოდება. ეს სისტემები უზრუნველყოფენ მუშაობის ისეთ სახეს, რომელიც მოხერხებულია და შეთანხმებული ობიექტებზე ორიენტირებული პროგრამირების მეთოდებთან.

1. მონაცემთა ბაზა - წარმოდგენის ძირითადი ცნებები და მოდელები

მონაცემთა დამუშავებასთან დაკავშირებული ამოცანები ფართოდაა გავრცელებული ადამიანის მოღვაწეობის ყველა სფეროში. შეუძლებელია წარმოვიდგინოთ რომელიმე თანამედროვე წარმოება-დაწესებულება ავტომატიზირებული ინფორმაციული სისტემების გარეშე. ეს სისტემები წარმოადგენენ ინფორმაციული მოღვაწეობის საფუძველს ყველა სფეროში.

ავტომატიზირებულ ინფორმაციული სისტემებში დაგროვილი ინფორმაციის მასივები უნდა იყვნენ ოპტიმალურად ორგანიზებულნი მათი კომპიუტერში განსათავსებლად და დასამუშავებლად, უნდა იყოს უზრუნველყოფილი მათი მთლიანობა და არაწინააღმდეგობრიობა.

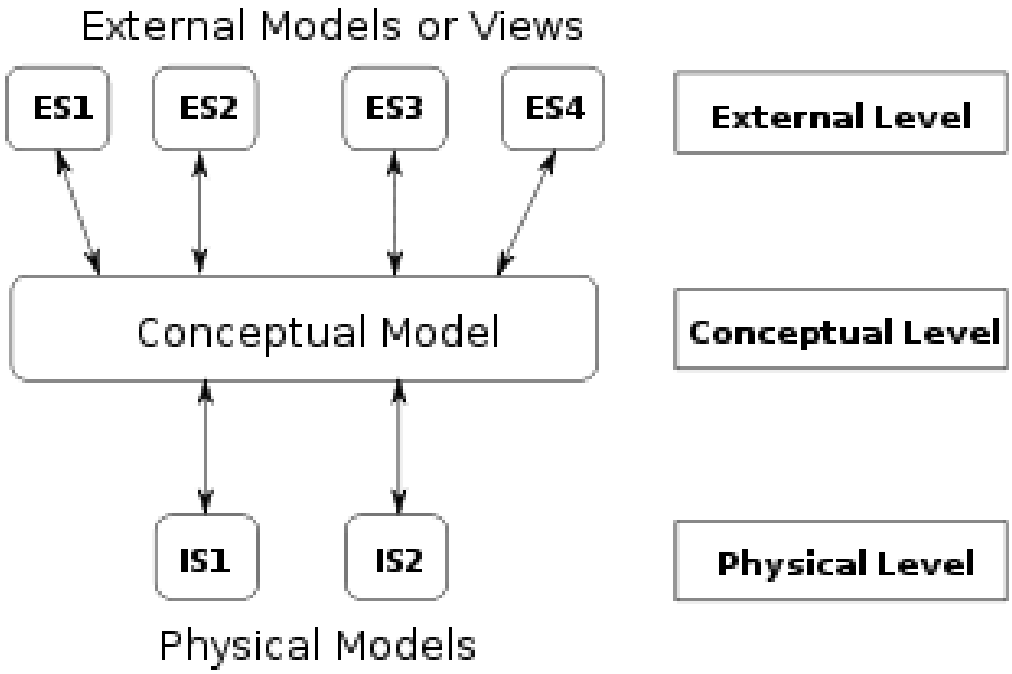
განვმარტოთ რამოდენიმე ძირითადი ცნება:

მონაცემთა ბანკი – ეს არის სპეციალური ხერხით ორგანიზებულ მონაცემთა სისტემა – მონაცემთა, პროგრამების, ტექნიკური, ენობრივი, საორგანიზაციო-მეთოდურ საშუალებების ბაზები – რომელიც განკუთვნილია მონაცემთა ცენტრალური დაგროვებისათვის და კოლექტიური მრავალმიზნობრივი გამოყენებისათვის.

მონაცემთა ბაზა – სახელწოდებულ მონაცემთა ერთობლიობა, რომელიც ასახავს ობიექტების მდგომარეობას და მათ დამოკიდებულებას განსახილველ საგნობრივ არესთან.

მონაცემთა ბაზების მართვის სისტემა – ენობრივი და პროგრამული საშუალებების ერთობლიობა, რომელიც გამიზნულია მონაცემთა ბაზის შესაქმნელად, მრავალი მომხმარებლის მიერ მონაცემთა ბაზის გამოსაყენებლად.

მონაცემთა ბაზების მართვის სისტემების სფეროში სამეცნიერო კვლევებისას, შემოთავაზებული იქნა მონაცემთა ბაზების რეალიზაციის განსხვავებული მეთოდები. ყველაზე სიცოცხლისუნარიანი ხერხი მათ შორის აღმოჩნდა ამერიკის სტანდარტიზაციის კომიტეტის ANSI (American National Standards Institute) მიერ შემოთავაზებული მონაცემთა ბაზის ორგანიზაციის სამდონიანი სისტემა (ნახ.1)



ნახ. 1

ჩვენი კვლევის საგანია **ფიზიკური დონე** – საკუთრივ მონაცემები რომლებიც განლაგებულნი არიან, ფაილებში ან ფურცლურ სტრუქტურებზე ინფორმაციის გარე მატარებლებზე. ეს მონაცემები შედგება ოთხი ძირითადი ელემენტისაგან: მომხმარებლის მონაცემები, მეტამონაცემები, ინდექსები და დანართების მეტამონაცემები. გამოვყოთ პირველი - ელემენტისაგან: მომხმარებლის მონაცემები.

მომხმარებელთა მონაცემების წარმოდგენისათვის მონაცემთა ბაზების განვითარების ეტაპზე შემუშავებული იყო სხვადასხვა მოდელები:

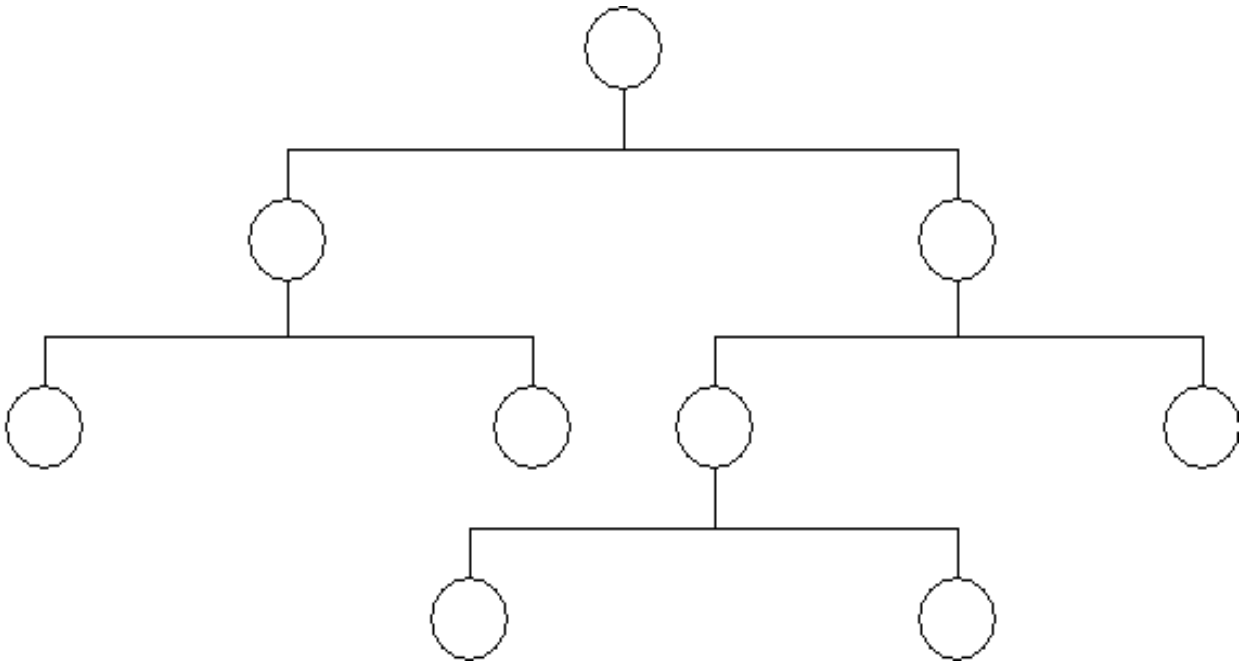
- იერარქიული მონაცემთა მოდელი
- ქსელის მონაცემთა მოდელი
- დამოკიდებული მონაცემთა მოდელი

იერარქიული მონაცემთა მოდელი

იერარქიული მონაცემთა მოდელის სტრუქტურა განისაზღვრება ჩანაწერები ხის სტრუქტურით, ერთ ჩანაწერს შეიძლება ყავდეს ერთი მშობელი და რამოდენიმე შვილი ჩანაწერი.

იერარქიული მონაცემთა მოდელის სტრუქტურა შეიძლება ასე წარმოვიდგინოთ:

ნახაზი 1 - იერარქიული მონაცემთა მოდელი



იერარქიული მონაცემთა ბაზა მოიცავს შემდეგს:

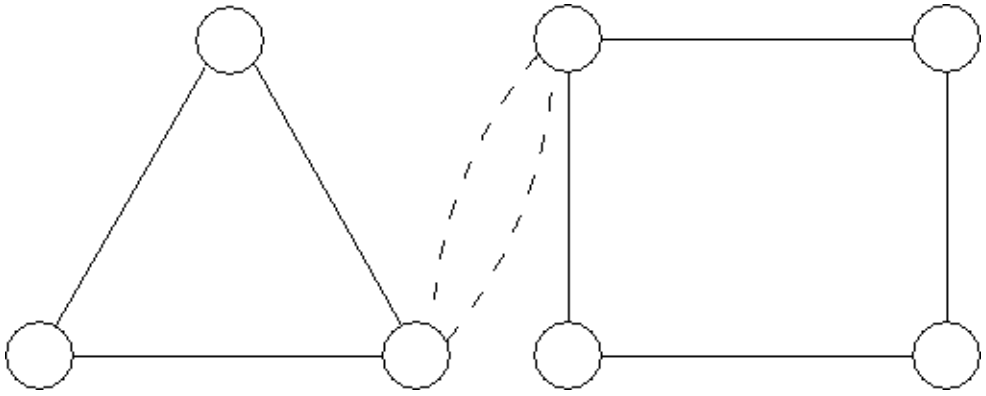
- ის შედგება კვანძებისგან რომლებიც ერთმანეთს უკავშირდება ტოტებით.
- თავში მდგომ კვანძს ფესვს ეძახიან.
- თუ რამოდენიმე კვანძი არის ზედა დონეზე, კვანძებს ეწოდებათ ფესვის სეგმენტები
- n^x კვანძის მშობელი არის კვანძი მის ზემოთ, რომელიც პირდაპირ დაკავშირებულია n^x -თან.
- თითოეულ კვანძს (გარდა ფესვისა) ყავს ზუსტად ერთი მშობელი.
- n^x კვანძის შვილი არის კვანძი მის ქვემოთ, რომელიც დაკავშირებულია n^x -თან.
- ერთ მშობელს შეიძლება ყავდეს რამოდენიმე შვილი

ქსელის მონაცემთა მოდელი

ქსელის მონაცემთა მოდელი იყენებს გისოსებურ სტრუქტურას, რომელშიც ჩანაწერებს შეიძლება ყავდეთ რამოდენიმე მშობელი, ისევე როგორც რამოდენიმე შვილი .

ქსელის მონაცემთა მოდელი შეიძლება ასე წარმოვიდგინოთ:

ნახაზი 2- ქსელის მონაცემთა მოდელი



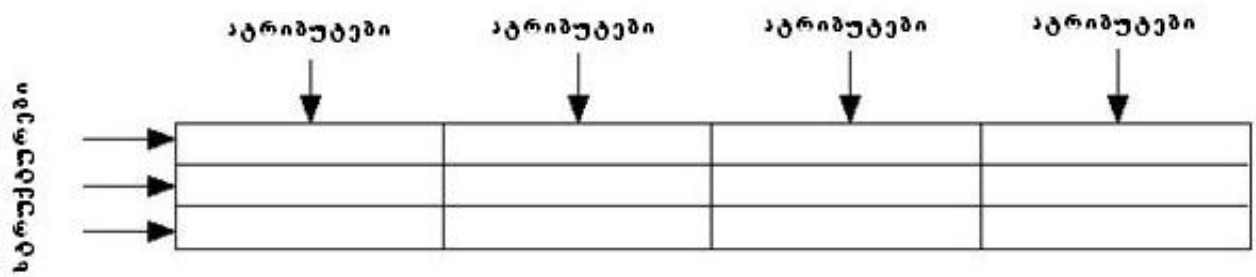
იერარქიული მონაცემთა მოდელის მსგავსად ქსელის მონაცემთა მოდელიც შედგება კვანზებისა და ტოტებისაგან, მაგრამ შვილ კვანძს შეიძლება ყავდეს რამოდენიმე მშობელი კვანძი ქსელის სტრუქტურაში.

რელაციური მონაცემთა მოდელი

დღესდღეობით მონაცემთა ბაზების უმეტესობა წარმოადგენენ რელაციურ მოდელს , ამიტომ განვიხილოთ ის უფრო დეტალურად.

რელაციური მოდელის ძირეული ელემენტია „რელაცია“ (ორგანზომილებიანი ცხრილი) და მასთან დაკავშირებული ძირითადი ცნებები: სტრიქონი _ კორტეჟი (**tuples)** სვეტი – ატრიბუტი (**attributes**). იმისთვის, რომ ცხრილი იყოს დამოკიდებულება იგი უნდა აკმაყოფილებდეს გარკვეულ მოთხოვნებს: ცხრილის უჯრებში მნიშვნელობები უნდა იყოს “ერთობითი” - არ დაიშვება განმეორებადი ჯგუფები, ან მასივები. სვეტში ყველა ჩანაწერი უნდა იყოს ერთი ტიპის. ყოველ სვეტს აქვს უნიკალური სახელი, სვეტების რიგი არსებითი არ არის. დამოკიდებულებაში არ შეიძლება არსებობდეს ორი ერთნაირი სტრიქონი, და სტრიქონთა რიგს მნიშვნელობა არ აქვს.

ნახაზი 3- რელაციები - რელაციურ მონაცემთა მოდელში



რელაციურ მოდელში ერთ-ერთი მნიშვნელოვანი კომპონენტია - გასაღებები, რომელიც წარმოადგენს ერთ ან რამოდენიმე ატრიბუტს, რომლის საშუალებითაც შესაძლებელია კორტეჟის ცალსახა განსაზღვრა.

არსებობს სხვადასხვა ტიპის გასაღებები

- **მარტივი** გასაღები შეიცავს ერთ ატრიბუტს
- **კომპოზიტური** გასაღები არის გასაღები, რომელიც შეიცავს ერთზე მეტ ატრიბუტს
- **კანდიდატი** გასაღები არის ატრიბუტი(ან ატრიბუტები) რომელიც უნიკალურად აიდენტიფიცირებს სტრიქონს. კანდიდატ გასაღებს უნდა ახასიათებდეს შემდეგი:
    1. **უნიკალური იდენტიფიკაცია** - თითოეული სტრიქონისთვის გასაღების მნიშვნელობა უნიკალურად უნდა აიდენტიფიცირებდეს ამ სტრიქონს.
    2. **არაჭარბი** - არცერთი ატრიბუტი გასაღებში არ შეიძლება წაიშალოს ისე რომ არ წაიშალოს უნიკალური იდენტიფიკაცია
- **პირველადი** გასაღები არის კანდიდატი გასაღები რომელიც შეირჩა პრინციპულად უნიკალურ იდენტიფიკატორად. ყოველი რელაცია უნდა შეიცავდეს პირველად გასაღებს. პირველადი გასაღები ჩვეულებრივ არის შერჩეული რათა მოხდეს სტრიქონის იდენტიფიცირება როდესაც მონაცემთა ბაზა არის შექმნილი ფიზიკურად. მაგალითად ნაწილის ნომერი შეირჩევა ნაწილის აღწერის მაგივრად.
- **ზეგასაღები** არის ატრიბუტების ნაკრები ატრიბუტებისა, რომელიც უნიკალურად აიდენტიფიცირებს სტრიქონს. ზეგასაღები განსხვავდება კანდიდატი გასაღებისგან იმით რომ არ მოითხოვს არასიჭარბის თვისებას.
- **გარე** გასაღები არის ატრიბუტი (ატრიბუტთა ნაკრები) რომელიც წარმოადგენს არაგასაღებ ატრიბუტს ერთ ცხრილში და არის პირველადი გასაღები მეორე ცხრილში. ასევე შესაძლებელია რომ გარე გასაღები იყოს მთლიანი ან ნაწილი პირველადი გასაღებისა.
    1. **მრავალი-მრავალთან** კავშირი მიიღწევა გადაკვეთის ან კავშირის ცხრილის შემოღებით, რომელიც შემდეგ ხდება შთამომავალი ორ ერთი-ბევრთან კავშრისა. გადაკვეთის ცხრილს ასევე აქვს გარე გასაღები თითოეული მისი წარმომშობისათვის და მისი პირველადი გასაღები არის ორივე გარე გასაღებისგან

შემდგარი.

2. **ერთი-ერთთან** კავშირი მოითხოვს რომ შთამომავალ ცხრილს ქონდეს მხოლოდ ერთი წარმომშობი ჩანაწერი, რომელიც ასევე შეიძლება მიიღწეოდეს რომ გარე გასაღები იყოს პირველადი გასაღები.

- **სემანტიკური** ან ბუნებრივი გასაღები არის გასაღები რომლის მნიშვნელობა თავიდანვე არის გასაგები მომხმარებლისთვის. მაგალითად, სემანტიკური გასაღები ქვეყნისათვის შეიძლება შეიცავდეს მნიშვნელობას საქართველო რომელიც აღწერს საქართველოს. მნიშვნელობა გასაგებია მომხმარებლისთვის.
- **ტექნიკური** ან სუროგატი ან ხელოვნური გასაღები არის გასაღები, რომლის მნიშვნელობებსაც მომხმარებლისთვის არ აქვს მნიშვნელობა. ისინი გამოიყენება სემანტიკური გასაღებების მაგივრად, შემდეგ შემთხვევებში:
  1. როდესაც სემანტიკური გასაღების მნიშვნელობა შეიძლება შეიცვალოს მომხმარებლის მიერ და მიიღოს დუბლიკატი მნიშვნელობები. მაგალითად, პერსონა ცხრილში არ არის რეკომენდირებული რომ გვარი იყოს როგორც გასაღები, რადგან შეიძლება იყოს ერთზე მეტი, ან შეიძლება შეიცვალოს ქორწინების შემდეგ.
  2. როდესაც არცერთი ატრიბუტი არ იძლევა უნიკალურობის გარანტიას, ემატება ატრიბუტი, რომლის მნიშვნელობები გენერირდება სისტემის მიერ. მაგალითად, რიცხვების მიმდევრობა, არის ერთადერთი გზა რომ მივიღოთ უნიკალური მნიშვნელობები. ტიპიური მაგალითებია შეკვეთის ნომერი და ინვოისის ნომერი. მნიშვნელობას '12345' არ აქვს არანაირი აზრი მომხმარებლისთვის, რადგან იგი არ ატარებს აზრს, იმ არსის შესახებ, რომელსაც შეესაბამება.
- გასაღების ფუნქციონალურად განსაზღვრავს სხვა ატრიბუტებს სტრიქონში, ანუ ის არის ყოველთვის დეტერმინირებული.
- მნიშვნელოვანია, რომ ტერმინი 'გასაღები' DBMS ძრავებში წარმოდგენილია როგორც ინდექსი, რომელიც არ იძლევა დუბლიკატების ჩამატების საშუალებას.

ერთი ცხრილი შეიძლება დაკავშირებული იყოს სხვასთან, რასაც **რელაცია** ეწოდება. რელაციები შეიძლება იყოს ჩაშენებული მონაცემთა ბაზების სტრუქტურაში რომ შესრულდეს რელაციური გაერთიანება შესრულების მიმდინარეობისას.

- რელაციები ორ ცხრილს შორის, ცნობილია როგორც ერთი-მრავალთან ან მშობელი-შვილთან მიმართება, სადაც შემთხვევა ერთის, მშობელის მხარეს შეიძლება ქონდეს ნებისმიერი რაოდენობა ბევრის ან შვილის მხარეს. ამის მისაღწევად შვილი ცხრილი უნდა შეიცავდეს ატრიბუტს, რომელიც მიემართება პირველადი გასაღებისკენ მშობელ ცხრილში. ეს ატრიბუტები ცნობილია როგორც გარე გასაღები, და მშობელი ცხრილი მიემართება გარე გასაღებით.

- ჩანაწერს მშობელ ცხრილში შესაძლებელია არ ქონდეს შესაბამისი ჩანაწერი შვილ ცხრილში, მაგრამ არ არის დასაშვები რომ შვილი ცხრილში იყოს ჩანაწერი რომელსაც არ ექნება შესაბამისი მშობელ ცხრილში.

- შვილი ჩანაწერი, რომელიც არ შეესაბამება მშობელ ჩანაწერს ცნობილია როგორც ობოლი.

- შესაძლებელია ცხრილი მიუთითებდეს თავის თავზე. ამ შემთხვევაში მას უნდა ქონდეს გარე გასაღები, რომელიც მიუთითებს ისევ პირველად გასაღებს. ამ შემთხვევაში, გასაღებები არ უნდა უთითებდნენ ზუსტად ერთმანეთს, რადგან ჩანაწერი მიუთითებს ყოველთვის მხოლოდ თავის თავზე.

- ცხრილში შეიძლება იყოს ნებისმიერი რაოდენობის რელაციები, და შეიძლება იყოს მშობელი და შვილი ერთდროულად.

- ზოგიერთი მონაცემთა ბაზის ძრავა ნებას რთავს მშობელ ცხრილს რომ დაუკავშირდეს კანდიდატი გასაღებით, მაგრამ ამის შეცვლის შემთხვევაში, კავშირი შეიძლება დაირღვეს შვილთან.

- ზოგიერთი მონაცემთა ბაზის ძრავა ნებას იძლევა კავშირი შედგეს წესებით, რომელიც ცნობილია მიმართების ერთიანობით ან გარე გასაღების შეკავებით. ეს თავიდან აგვაშორებს რომ შვილი ჩანაწერები არ შეიქმნას იმ შემთხვევაში თუ არ გვაქვს მშობელი

ჩანაწერები, ან გაასწორებს შთამომავალ ჩანაწერს, როდესაც წარმომშობი შეიცვლება ან წაიშლება.

რელაციური კავშირი

კავშირის ოპერატორი (join) გამოიყენება რომ გავაერთიანოთ მონაცემები ორი ან მეტი ცხრილიდან რათა დავაკმაყოფილოთ განსაზღვრული მოთხოვნა. ორი ცხრილი შეიძლება დაკავშირდეს როდესაც მათ აქვთ ერთი მაინც საერთო ატრიბუტი. კავშირი განსაზღვრულია თითოეული სტრიქონისთვის ცხრილში. სტრიქონი ცხრილში R1 უკავშირდება სტრიქონს ცხრილში R2 როდესაც როდესაც საერთო ატრიბუტის მნიშვნელობა ემთხვევა ორივე ცხრილში. ორი ცხრილის კავშირს უწოდებენ ბინარულ კავშირს.

ორი ცხრილის კავშირი ქმნის ახალ ცხრილს. **R1 x R2** ნოტაცია ნიშნავს R1 და R2 ცხრილების კავშირებს, მაგალითად.

რელაცია		R1
A	B	C
1	5	3
2	4	5
8	3	5
9	3	3
1	6	5
5	4	3
2	7	5
რელაცია		R2
B	D	E
4	7	4
6	2	3
5	7	8
7	2	3
3	2	2

R1 და R2 ცხრილები შეიცავს ერთნაირ მნიშვნელობას B ატრიბუტისთვის. მონაცემთა ნორმალიზაცია წარმოდგენილია R (A,B,C,D,E) რელაციის უფრო მცირე რელაციებად

დაყოფით (R1 და R2). მონაცემთა მნიშვნელობები ატრიბუტი B-სთვის ამ კონტექსტში იქნება იდენტური ორივესთვის. R1 და R2 არიან პროექციები R(A,B,C,D,E) ატრიბუტებზე (A,B,C) და (B,D,E). პროექცია არ გაანადგურებს მნიშვნელობებს - დუბლიკატი სტრიქონები წაიშლება, მაგრამ ეს არ წაშლის მონაცემებს ატრიბუტებიდან.

ორი ცხრილის R1 და R2 გაერთიანება შესაძლებელია რადგან მათ აქვთ საერთო B ატრიბუტი.

რელაცია R1 x R2				
A	B	C	D	E
1	5	3	7	8
2	4	5	7	4
8	3	5	2	2
9	3	3	2	2
1	6	5	2	3
5	4	3	7	4
2	7	5	2	3

სტრიქონი (2 4 5 7 4) ჩამოყალიბდა ორი სტრიქონის გაერთიანებით (2 4 5) და (4 7 4). ორი სტრიქონი გაერთიანდა რადგან ისინი შეიცავდნენ ერთნაირ მნიშვნელობებს. სტრიქონი (2 4 5) არ გაერთიანდა სტრიქონთან (6 2 3) რადგან საერთო ატრიბუტი (4 6) არ არის ერთნაირი.

კავშირების შეერთება მაგალითში იზიარებდა ზუსტად ერთ საერთო ატრიბუტს. მაგრამ შესალებელია რომ ქონდეთ მრავალი საერთო ატრიბუტი. ყველა უნდა გამოყენებული იყოს გაერთიანებაში. მაგალითად R1 და R2 ობიექტების კავშირი შემდეგ მაგალითში კავშირდება B და C ატრიბუტით.

გაერთიანებამდე:

რელაცია R1		
A	B	C
6	1	4
8	1	4
5	1	2
2	7	1

რელაცია R2		
B	C	D
1	4	9
1	4	2
1	2	1
7	1	2
7	1	3

გაერთიანების შემდეგ:

Relation R1 x R2			
A	B	C	D
6	1	4	9
6	1	4	2
8	1	4	9
8	1	4	2
5	1	2	1
2	7	1	2
2	7	1	3

სტრიქონი (6 1 4 9) ჩამოყალიბდა ორი სტრიქონის გაერთიანების შედეგად (6 1 4) და (1 4 9) - დან. გაერთიანება შეიქმნა საერთო ატრიბუტებიდან B და C, რომლებიც შეიცავდნენ ერთნაირ მნიშვნელობებს (1 4). სტრიქონი (6 1 4) არ შეუერთდა სტრიქონს (1 2 1) რადგან საერთო ატრიბუტები არ დაემთხვა ერთმანეთს.

გაერთიანების ოპერაცია გვაძლევს საშუალებას ავაგოთ რელაცია რომელიც იყო გახლეჩილი ორ ცხრილად ნორმალიზაციის პროცესისას. ორი სტრიქონის გაერთიანებას შეუძლია შექმნას ერთი ახალი სტრიქონი, რომელიც არ იყო ორიგინალი რელაციის წევრი. მაგრამ გაერთიანების დროს შეიძლება შეიქმნას არასწორი ინფორმაცია.

რელაციები აკმაყოფილებს უდანაკარგო კავშირების თვისებას, თუ ობიექტები შეიძლება გაერთიანდეს არასწორი მონაცემების შექმნის გარეშე.

არასწორი ინფორმაციის მაგალითის მოსაყვანად განვიხილოთ შემდეგი სტრუქტურა:

R(student, course, instructor, hour, room, grade)

როდესაც მხოლოდ ერთი კლასი უნდა გამოეყოს სემესტრის განმავლობაში, შეგვიძლია განვსაზღვროთ შემდეგი დამოკიდებულებები:

- (HOUR, ROOM) ⟶ COURSE
- (COURSE, STUDENT) ⟶ GRADE
- (INSTRUCTOR, HOUR) ⟶ ROOM
- (COURSE) ⟶ INSTRUCTOR
- (HOUR, STUDENT) ⟶ ROOM

ავიღოთ მარტივი ცხრილი:

STUDENT	COURSE	INSTRUCTOR	HOUR	ROOM	GRADE
ანა	მათემატიკა	მანანა	8:00	100	A
მარი	ინგლისური	მაია	8:00	200	B
ლაშა	ინგლისური	მაია	8:00	200	A
ნიკა	ალგებრა	მანანა	9:00	400	C

მომდევნო ოთხი კავშირი, თითოეული 4-ე ნორმალურ ფორმაში, შეიძლება დაგენერირდეს მოცემული და ნაგულისხმევი დამოკიდებულებებიდან.

- **R1(STUDENT, HOUR, COURSE)**
- **R2(STUDENT, COURSE, GRADE)**
- **R3(COURSE, INSTRUCTOR)**
- **R4(INSTRUCTOR, HOUR, ROOM)**

ყურადსაღებია, რომ დამოკიდებულებები **(საათი, ოთახი) -> კურსი** და **(საათი, სტუდენტი) - > ოთახი** არ არიან განსაზღვრული ცხადად შემდეგ ჩაშლაში. ჩვენი მიზანია რომ მივაღწიოთ მეოთხე ნორმალურ ფორმას, რომელიც შესაძლებელი იქნება დავუკავშიროთ სწორად პასუხს. მიზანი შეიძლება მიღწეულ იქნას ისე, რომ არ წარმოვადგინოთ თითოეული ფუნქციონალური დამოკიდებულება როგორც კავშირი. უფრო მეტიც, რამდენიმე რელაციამ შეიძლება დააკმაყოფილოს მიზანი.

წინა რელაციები შეიძლება განვიხილოთ შემდეგნაირად:

R1

STUDENT	HOUR	COURSE
ანა	8:00	მათემატიკა
მარი	8:00	ინგლისური
ლაშა	8:00	ინგლისური
ნიკა	9:00	ალგებრა

R2

STUDENT	COURSE	GRADE
ანა	მათემატიკა	A
მარი	ინგლისური	B
ლაშა	ინგლისური	C
ნიკა	ალგებრა	A

R3

COURSE	INSTRUCTOR
მათემატიკა	მანანა
ინგლისური	მაია
ალგებრა	მანანა

R4

INSTRUCTOR	HOUR	ROOM
მანანა	8:00	100
მაია	8:00	200
მანანა	9:00	400

წარმოვიდგინოთ, რომ საჭიროა კურსების სია შესაბამისი ოთახის ნომრებით. რელაცია R1 და R4 შეიცავს საჭირო ინფორმაციას და შეიძლება დაკავშირდეს ატრიბუტი 'საათი'-ს მეშვეობით.

R1 x R4				
STUDENT	**COURSE**	**INSTRUCTOR**	**HOUR**	**ROOM**
ანა	მათემატიკა	მანანა	8:00	100
ანა	მათემატიკა	მაია	8:00	200
მარი	ინგლისური	მანანა	8:00	100
მარი	ინგლისური	მაია	8:00	200
ლაშა	ინგლისური	მანანა	8:00	100
ლაშა	ინგლისური	მაია	8:00	200
ნიკა	ალგებრა	მანანა	9:00	400

ეს კავშირი ქმნის შემდეგ არასწორ ინფორმაციას:

- ანა, მარი, ლაშა- მათ აქვთ ერთი გაკვეთილი ერთსადაიმავე დროს ორი სხვადასხვა ინსტრუქტორისგან, სხვადასხვა ოთახში.
- მანანა (მათემატიკის მასწავლებელი) ასწავლის ინგლისურს.
- მაია (ინგლისურის მასწავლებელი) ასწავლის მათემატიკას.
- ორივე ინსტრუქტორი ასწავლის სხვადასხვა კურსს ერთდროულად.

კიდევ ერთი შესაძლებლობა კავშირისთვის, არის R3 და R4 ცხრილების გადაბმა:

R3 x R4			
COURSE	**INSTRUCTOR**	**HOUR**	**ROOM**
მათემატიკა	მანანა	8:00	100
მათემატიკა	მანანა	9:00	400
ინგლისური	მაია	8:00	200
ალგებრა	მანანა	8:00	100
ალგებრა	მანანა	9:00	400

ეს კავშირი ქმნის შემდეგ არასწორ ინფორმაციას:

- მანანა ასწავლის მათემატიკას და ალგებრას ერთად, 8 და 9 საათზე.

სწორი მიმდევრობა იქნება R1 და R3 კავშირების შექმნა (კურსის გამოყენებით) და შემდეგ დაკავშირება R4-თან (ინსტრუქტორი და საათი):

R1 x R3			
STUDENT	**COURSE**	**INSTRUCTOR**	**HOUR**
ანა	მათემატიკა	მანანა	8:00
მარი	ინგლისური	მაია	8:00
ლაშა	ინგლისური	მაია	8:00
ნიკა	ალგებრა	მანანა	9:00

(R1 x R3) x R4				
STUDENT	**COURSE**	**INSTRUCTOR**	**HOUR**	**ROOM**
ანა	მათემატიკა	მანანა	8:00	100
მარი	ინგლისური	მაია	8:00	200
ლაშა	ინგლისური	მაია	8:00	200
ნიკა	ალგებრა	მანანა	9:00	400

გავშალოთ კურსი და ოთახი ატრიბუტები (და წავშალოთ დუბლიკატი სტრიქონები წარმოქმნილი ინგლისურის მიერ) რომელიც გვაძლევს საჭირო შედეგს.

COURSE	**ROOM**
მათემატიკა	100
ინგლისური	200
ალგებრა	400

სწორი შედეგი მიიღწევა როდესაც მიმდევრობა (r1 და r3) და r4 აკმაყოფილებს უდანაკარგო გაერთიანების თვისებას.

რელაციური მონაცემთა ბაზა არის 4-ე ნორმალურ ფორმაში როდესაც შესაძლებელია უდანაკარგო გაერთიანებამ უპასუხოს განუსაზღვრელ მოთხოვნებს. კავშირების უნდა იყოს სწორად წარმოქმნილი. სხვადასხვა მიმდევრობა დაგვიბრუნებს რელაციის ობიექტს (ცხრილს). ზოგიერთი მიმდევრობა პრიორიტეტულია, რადგან ის ქმნის ნაკლებ არასწორ მონაცემებს კავშირისას.

წარმოვიდგინოთ, რომ რელაცია დაშლილია ფუნქციური დამოკიდებულებებით და მრავალმნიშვნელობიანი დამოკიდებულებებით. მაშინ, აუცილებლად არსებობს ერთი მაინც კავშირი, რომელიც შექმნის ორიგინალ ცხრილს არასწორი მონაცემების გარეშე.

R1 x R3
(R1 x R3) x R2
((R1 x R3) x R2) x R4

ან

R1 x R3
(R1 x R3) x R4
((R1 x R3) x R4) x R2

მაგალითად, წარმოვიდგინოთ რომ ხარისხები განსაზღვრულია ოთახის ნომრის მიხედვით. ეს არის კითხვა, რომელიც სავარაუდოდ არ დაისვა მონაცემთა ბაზის დიზაინის დროს, მაგრამ შეიძლება პასუხი გაეცეს არასწორი მონაცემების შექმნის გარეშე შემდეგი ორი კავშირის ოპერაციით.

მოთხოვნილ ინფორმაციას შეიცავს რელაციები R2 და R4, მაგრამ ეს რელაციები არ შეიძლება დაკავშირდნენ. ამ შემთხვევაში, სწორი გადაწყვეტილებაა დაკავშირდეს ოთხივე რელაცია.

მონაცემთა ბაზამ შეიძლება მოითხოვოს უდანაკარგო კავშირის რელაცია, რომელიც აწყობილია რათა დავრწმუნდეთ რომ პასუხი გაეცემა მოთხოვნას სწორად. ეს რელაცია შეიძლება შეიცავდეს ატრიბუტებს, რომელიც არ არიან ერთმანეთთან ლოგიკურად დაკავშირებული. ეს ხდება იმის გამო რომ რელაცია უნდა იყოს ხიდი სხვა რელაციებს შორის ბაზაში. მაგალითად, უდანაკარგო კავშირი მოიცავს ყველა ატრიბუტს, რომელიც არის მარცხენა მხარეს ფუნქციონალური დამოკიდებულებისა. სხვა ატრიბუტები შეიძლება ასევე იყოს მოთხოვნილი.

რელაციური სქემა R(A, B, C, D), A⟶B და C⟶D. რელაციები Rl(A, B) და R2(C, D) არიან მეოთხე ნორმალურ ფორმაში. მესამე რელაცია R3(A, C)-ც უნდა აკმაყოფილებდეს უდანაკარგო კავშირის თვისებას. ეს რელაცია შეიძლება გამოყენებულ იქნას რათა დაკავშირდეს ატრიბუტები B და D. რაც მიიღწევა R1 და R3 რელაციების დაკავშირებით და შემდეგ შედეგის გადაბმით R2 რელაციაზე. ამ შემთხვევაში არ შეიქმნება არასწორი მონაცემები. რელაცია R3(A, C) არის უდანაკარგო ბაზის დიზაინისთვის.

რელაცია ძირითადად წარმოდგენილია ატრიბუტების გაერთიანებით განსაზღვრულ ობიექტზე. უდანაკარგო კავშირი კი წარმოადგენს კავშირს სხვადასხვა რელაციებს შორის. უდანაკარგო კავშირი შეიძლება იყოს რთული - მივიღოთ ისეთი შედეგი, რომელიც ლოგიკურად არ ასოცირდება ერთმანეთთან.

უდანაკარგო კავშირში ატრიბუტები ხშირად შეიცავენ მრავალმნიშვნელოვან დამოკიდებულებებს. მეოთხე ნორმალური ფორმის გათვალისწინება ამ სიტუაციაში მნიშვნელოვანია. უდანაკარგო კავშირი შეიძლება ზოგჯერ დაიშალოს მრავალმნიშვნელოვანი დამოკიდებულებების წაშლის გზით. მცირე რელაციები უფრო ადვილი შესავსები და მოსავლელია.

განმსაზღვრელი და დამოკიდებული

ტერმინი შეიძლება განისაზღვროს შემდეგნაირად:

- გამოსახულება x->y ნიშნავს, თუ ვიცი x, მაშინ შემიძლია გავიგო y.
- გამოსახულებაში x განმსაზღვრელია, y დამოკიდებული ატრიბუტი.
- X-ის მნიშვნელობა განსაზღვრავს y-ს.
- Y -ის მნიშვნელობა დამოკიდებულია x-ზე.

ფუნქციონალური დამოკიდებულება განისაზღვრება შემდეგნაირად:

- ატრიბუტი არის ფუნქციონალურად დამოკიდებული თუ მისი მნიშვნელობა განისაზღვრება სხვა ატრიბუტის მეშვეობით, რომელიც არის გასაღები.
- თუ ჩვენ ვიცით ერთი ან რამდენიმე ცვლადის მნიშვნელობა, შეგვიძლია ვიპოვოთ მეორე ცვლადი.
- ფუნქციონალური დამოკიდებულება გამოხატულია x -> y, სადაც x დეტერმინანტია და y ფუნქციონალურად დამოკიდებული ატრიბუტი.
- თუ A -> (B, C), მაშინ C->B და A->C.
- თუ (A,B) ->C =, მაშინ არ არის აუცილებელი რომ A->C და B->C მართალი იყოს.
- თუ A->B და B->A, მაშინ A და B არიან 1:1 კავშირში.
- თუ A->B, მაშინ A-სთვის მხოლოდ ერთი B-ს მნიშვნელობა შეიძლება იყოს.

ტრანზიტული დამოკიდებულება შეიძლება აღიწეროს შემდეგნაირად:

- ატრიბუტი არის ტრანზიტურალ დამოკიდებული, თუ მისი მნიშვნელობა განისაზღვრება ატრიბუტის მეშვეობით, რომელიც არ არის გასაღები.
- თუ x->y და x არ არის გასაღები, მაშინ ეს არის ტრანზიტული დამოკიდებულება.
- ტრანზიტული დამოკიდებულება არსებობს მხოლოდ A->B->C და არა A->C შემთხვევაში.

მრავალმნიშვნელობიანი დამოკიდებულება შეიძლება აღიწეროს:

- ცხრილი შეიცავს მრავალმნიშნელოვან დამოკიდებულებას, თუ ის შეიცავს მრავალ მნიშვნელობას არისთვის.
- ასეთი დამოკიდებულება შეიძლება გამოჩნდეს პირველი ნორმალური ფორმის გამოყენებისას
- X->>y, თუ x მრავალ მნიშვნელობიანად განსაზღვრავს y-ს, თუ ყოველი x_ისთვის შეიძლება გვქონდეს ერთზე მეტი y.
- თუ A->>B და A->>C მაშინ გვაქვს ერთი ატრიბუტი A, რომელიც მთავალ მნიშვნელობიანად განსაზღვრავს ორ სხვა დამოუკიდებელ ატრიბუტს, B და C.
- თუ A->>(B, C) მაშინ გვაქვს ატრიბუტი A, რომელიც მრავალმნიშვნელობიანად განსაზღვრავს შესაბამის ატრიბუტეს, B და C.

კავშირის დამოკიდებულება შეიძლება აღიწეროს: თუ ცხრილი შეიძლება დაიშალოს სამ ან მეტ უფრო მცირე ცხრილად, ის შეიძლება აეწყოს ისევ, გასაღებებით ორიგინალი ცხრილიდან.

2. მოდიფიკაცია, ანომალია, ნორმალიზაცია

მონაცემთა ბაზების ექსპლოატაციის განუყოფელი ნაწილია მისი მოდიფიკაცია, რომელიც გულისხმობს: ჩაწერის, წაშლის და განახლების ანომალიების წინააღმდეგ ბრძოლასა და ნორმალიზაციას.

როგორც წესი ამ პროცესს თან სდევს მოდიფიკაციის ანომალია, რომელიც გულისხმობს აღმოფხვრის პროცესს, რომელსაც გვკარნახობს ნორმალიზაციის წესები, სახელდობრ მონაცემთა ნორმალიზაციის მიზანია რომ თავი ავარიდოთ მოდიფიკაციის ანომალიებს. ისინი ორი სახისაა:

- ჩაწერის ანომალია არის წარუმატებლობა ახალი ინფორმაციის შეტანისა ბაზაში სადაც საჭიროა რომ ეს ჩანაწერი არსებობდეს. სწორად ნორმალიზებულ მონაცემთა ბაზაში, ინფორმაცია ახალი ჩანაწერის შესახებ უნდა ჩაიწეროს მხოლოდ ერთ ცხრილში. არაადეკვატურად ნორმალიზებულ ბაზაში, ინფორმაცია შეიძლება ჩაიწეროს ერთზე მეტ ადგილას, და ადამიანის შეცდომის გამო შეიძლება დამატებითი ჩაწერა არ მოხდეს.
- წაშლის ანომალია არის ინფორმაციის წაშლის წარუმატებლობა არსებული ბაზიდან, როდესაც ჩანაწერი უნდა წაიშალოს. სწორად ნორმალიზებულ ბაზაში ის უნდა წაიშალოს მხოლოდ ერთი ადგილიდან, ხოლო არასწორად ნორმალიზებულში კი რამდენიმე ადგილიდან, ასევე გასათვალისწინებელია ადამიანის შეცდომაც.
- განახლება ბაზისა მოიცავს მოდიფიკაციებს რომელიც შეიძლება იყოს დამატება, წაშლა ან ორივე. განახლების ანომალიები არის ერთ-ერთი ზემოთ განხილულიდან.
- სამივე ანომალია არის არასასურველი, მაგრამ მათი არსებობა წარმოქმნის ბაზის პრობლემებს. სწორად ნორმალიზებული ბაზა არის მდგრადი ასეთი ანომალიების მიმართ.

დაკავშირება არის მეთოდი რომ შეიქმნას მონაცემები ორი ან მეტი ცხრილიდან. როდესაც დარდება ცხრილების ინფორმაცია შეიძლება გამოიკვეთოს შემდეგი პირობები:

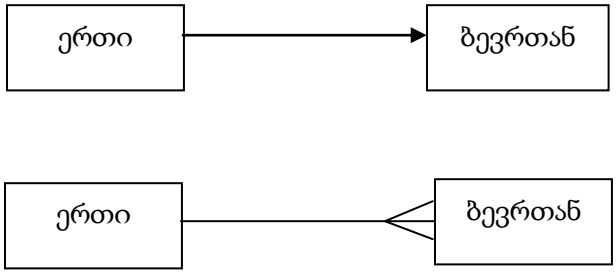
- ყოველ სტრიქონს ერთ ცხრილში აქვს შესაბამისი მეორე ცხრილში
- რელაცია R1 შეიცავს სტრიქონებს, რომელთაც არ აქვთ შესაბამისი R2 რელაციაში.
- რელაცია R2 შეიცავს სტრიქონებს, რომელთაც არ აქვთ შესაბამისი R1 რელაციაში.

არსი-კავშირის დიაგრამა არის მონაცემების მოდელირების ტექნოლოგია, რომელიც ქმნის არსების გრაფიკულ წარმოდგენას და კავშირებს მათ შორის, ინფორმაციული სისტემის შიგნით. არსი-კავშირის დიაგრამას აქვს შესაბამისი რელაციური ცხრილი და ნებისმიერ რელაციურ ცხრილს აქვს შესაბამისი არსი-კავშირის დიაგრამა. არსი-კავშირის დიაგრამის გაკეთება არის შეუფასებელი დახმარება ინჟინრებისთვის დიზაინში, ოპტიმიზაციაში და შეცდომების გასწორებაში.

- არსი არის პერსონა, ობიექტი, ადგილი ან ხდომილება რომლისთვისაც მონაცემები იკრიბება. ის ექვივალენტურია ცხრილისა. არსი შეიძლება განისაზღვროს მისი თვისებებით, ანუ ატრიბუტებით. მაგალითად, არსის კლიენტს შეიძლება ქონდეს ატრიბუტები, როგორიცაა სახელი, მისამართი და ტელეფონის ნომერი.
- რელაცია არის არსებს შორის ურთიერთქმედება. შეიძლება აღიწეროს შემდეგნაირად:
    - მომხმარებელი აკეთებს შეკვეთას
    - გამყიდველი ემსახურება კლიენტს
    - შეკვეთა შეიცავს პროდუქტს
    - საწყობი ყიდის პროდუქტს

არსი-კავშირი დიაგრამაში არსები გამოსახულია როგორც მართკუთხედი, კავშირები როგორც ისრები, რომლებიც აერთებს მართკუთხედებს. ისრის თავი მიმართული მშობლიდან შვილისაკენ.

მეორე მეთოდით შეგვიძლია ასე წარმოვადგინოთ:

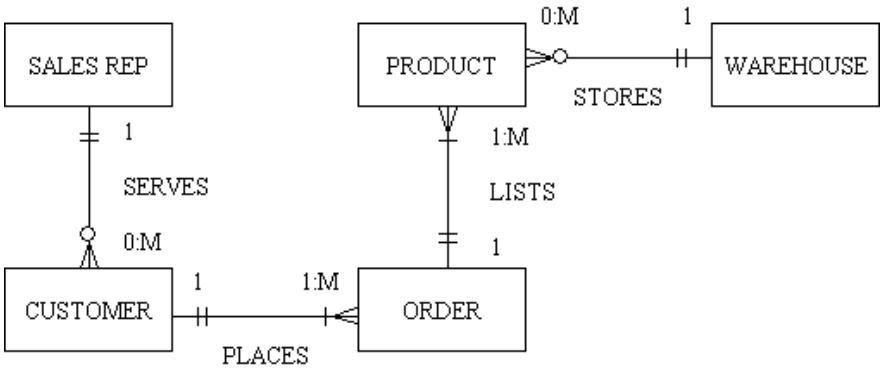


კავშირის ხაზი გამოიყენება კარდინალობის აღსანიშნავადაც არსებს შორის. არსი შეიძლება იყოს 0 ან მეტი, ან იყოს მისი არსებობა სავალდებულო 1 ან მეტი.

- ერთი შტრიხი აღნიშნავს 1-ს.
- ორი შტრიხი აღნიშნავს ზუსტად 1-ს
- წრე აღნიშნავს 0-ს.
- ისრის თავი აღნიშნავს ბევრს.

ასევე რიცხვებითაც შეიძლება აღინიშნოს კარდინალობა:

- ერთი-ერთთან გამოიხატება 1:1
- ნოლი - ბევრთან გამოიხატება 0:M
- ერთი-ბევრთან გამოიხატება 1:N
- ბევრი-ბევრთან N:M



რელაციური მონაცემთა ბაზების თეორია და ნორმალიზაციის პრინციპები პირველად ააგო ხალხმა, რომელსაც ჰქონდა ძლიერი მათემატიკური ბაზა. ისინი წერდნენ ბაზების შესახებ ისეთი ტერმინოლოგიით, რომელიც არ იყო მარტივი გასაგები მათემატიკის ცოდნის გარეშე.

მონაცემთა ნორმალიზაცია არის წესების კრებული:

- მოახდინეთ კავშირების იდენტიფიკაცია ატრიბუტებს შორის;
- შეაერთეთ ატრიბუტები რათა ჩამოყალიბდეს რელაცია;
- შეაერთეთ რელაციები, რათა ჩამოყალიბდეს ბაზა.

ამას მოყვება კოდის წესები, რომელიც შეიქმნა 1970 წელს. ნორმალიზებული რელაციური მონაცემთა ბაზა გვაძლევს შემდეგ უპირატესობებს:

- ჭარბი მონაცემების არარსებობა;
- სიახლოვე რეალურ არსებთან, პროცესებთან და მათ კავშირთან;
- მონაცემთა ისე სტრუქტურირება, რომ მოდელი იყოს მოქნილი.

არსებობს რამდენიმე ნაბიჯი ნორმალიზაციის პროცესში 1NF, 2NF, 3NF, BCNF, 4NF, 5NF და DKNF. უამრავი ბაზის დიზაინერი თვლის რომ მესამე ნორმალურ ფორმის იქით წასვლა არ ღირს. ეს არ ნიშნავს რომ სხვა ფორმები არ არის მნიშვნელოვანი, უბრალოდ შემთხვევები, რომლებისთვისაც ის მიიღეს ძალიან იშვიათია. მაგრამ ყველა დიზაინერი უნდა ამოწმებდეს ნორმალიზაციის ყველა წესს, ასე უფრო მარტივად გამოირკვევა თუ რომელიმე დაირღვა და მიიღებს შესაბამის ზომებს.

მესამე ნორმალური ფორმის დირექტივები ასეთია:

- განსაზღვრეთ ატრიბუტები;
- დააჯგუფეთ ლოგიკურად დაკავშირებული ატრიბუტები რელაციაში;
- დაასახელეთ კანდიდატი გასაღები თითოეული რელაციისთვის;
- აირჩიეთ პირველადი გასაღები თითოეული რელაციისთვის;
- დაასახელეთ და წაშალეთ განმეორებადი ჯგუფები;
- გააერთიანეთ რელაციები იდენტური გასაღებებით (პირველი ნორმალური ფორმა);
- დაასახელეთ ყველა ფუნქციონალური დამოკიდებულება;
- დაშალეთ რელაცია ისე რომ არაგასაღები ატრიბუტი დამოკიდებული იყოს გასაღებში არსებულ ყველა ატრიბუტზე;
- გააერთიანეთ რელაციები იდენტური პირველადი გასაღებით (მეორე ნორმალური ფორმა);

- დაასახელეთ ყველა ტრანზიტული დამოკიდებულება:
    - შეამოწმეთ რელაციები დამოკიდებულებისთვის ერთი არაგასაღები ატრიბუტი მეორე არაგასაღები ატრიბუტით.
    - შეამოწმეთ ყველა დამოკიდებულება თითოეული პირველადი გასაღებით (გასაღების ერთი ატრიბუტის დამოკიდებულება მეორეზე).
- დაშალეთ რელაციები ისე რომ არ იყოს ტრანზიტული დამოკიდებულებები;
- გააერთიანეთ რელაციები იდენტური პირველადი გასაღებით თუ არ შეიქმნება ტრანზიტული დამოკიდებულებები.

პირველი ნორმალური ფორმა

ცხრილი არის პირველ ნორმალურ ფორმაში თუ ყველა გასაღები ატრიბუტი განსაზღვრულია და არ შეიცავს განმეორებად ჯგუფებს.

განვიხილოთ ცხრილი ORDER, რომელსაც აქვს ასეთი ატრიბუტების განლაგება

ORDER				
order_id	**customer_id**	**product1**	**product2**	**product3**
123	456	abc1	def1	ghi1
456	789	abc2		

ეს სტრუქტურა ქმნის შემდეგ პრობლემებს

- შეკვეთა 123 არ აქვს ადგილი 3-ზე მეტი პროდუქტისთვის
- შეკვეთა 456 აქვს ზედმეტი ადგილი

იმისთვის რომ შევქმნათ ცხრილი რომელიც არის პირველ ნორმალურ ფორმაში უნდა გამოვაცალკევოთ განმეორებადი ჯგუფები და გავიტანოთ ცალკე ცხრილში.

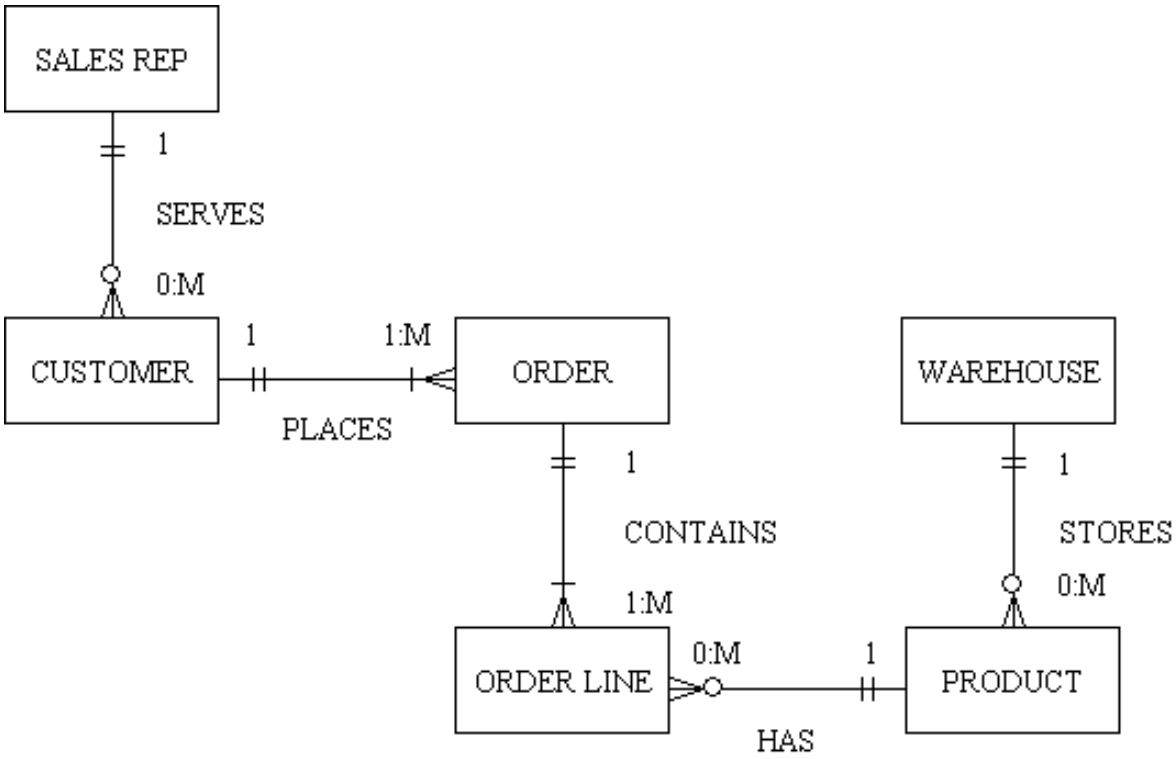
ORDER	
order_id	**customer_id**
123	456
456	789

ORDER ცხრილიდან წავშალეთ პროდუქტი1, პროდუქტი2 და პროდუქტი3 ატრიბუტები, ასე რომ აღარ არის განმეორებადი ჯგუფები

ORDER_LINE	
order_id	**product**
123	abc1
123	def1
123	ghi1
456	abc2

თითოეული სტრიქონი ORDER_LINE ში შეიცავს ერთ პროდუქტს შეკვეთისათვის, ეს საშუალებას იძლევა რომ შეკვეთაში შევიდეს ნებისმიერი რაოდენობის პროდუქტი.

ამას მივყავართ არსი-კავშირის დიაგრამის ახალ შედეგამდე:



ახალი კავშირი შეიძლება ჩამოყალიბდეს შემდეგნაირად:

- 1 ობიექტს ORDER_სას აქვს ერთი ან მეტი ORDER_LINE
- 1 ობიექტს PRODUCT-სას აქვს 0 ან მეტი ORDER_LINE

მეორე ნორმალური ფორმა

ცხრილი მეორე ნორმალურ ფორმაშია მაშინ და მხოლოდ მაშინ, თუ ის არის პირველ ნორმალურ ფორმაში და ყველა არაგასაღები ატრიბუტი ფუნქციონალურად დამოკიდებულია პირველად გასაღებზე.

- ანომალიები შეიძლება მოხდეს როდესაც ატრიბუტები დამოკიდებულია მხოლოდ ნაწილზე კომპოზიტური გასაღებისა.
- რელაცია მეორე ნორმალურ ფორმაშია, როცა ყველა არაგასაღები ატრიბუტი

დამოკიდებულია მთლიან გასაღებზე. ატრიბუტი არ უნდა იყოს გასაღების ნაწილზე დამოკიდებული.

- ნებისმიერი ცხრილი, რომელსაც აქვს ერთ ატრიბუტიანი გასაღები არის მეორე ნორმალურ ფორმაში.

მაგალითად, მოცემულია შემდეგი სტრუქტურის ცხრილი:

order(order_id, cust, cust_address, cust_contact, order_date, order_total)

აქ ჩვენ უნდა გავიგოთ რომ cust_address, cust_contact არიან ფუნქციონალურად დამოკიდებულები cust და არა order_id-ზე. მითუმეტეს ისინი არ არიან დამოკიდებული მთლიან გასაღებზე. ეს ცხრილი რომ იყოს მეორე ნორმალურ ფორმაში ეს ატრიბუტები უნდა ამოვიღოთ და გადავიტანოთ სხვაგან.

მესამე ნორმალური ფორმა

ცხრილი არის მესამე ნორმალურ ფორმაში მაშინ და მხოლოდ მაშინ როცა ის არის მეორე ნორმალურ ფორმაში და ყველა არაგასაღები ატრიბუტი არატრანზიტულად არის დამოკიდებული პირველად გასაღებზე.

- ანომალიები შეიძლება მოხდეს როცა ცხრილი შეიცავს ერთ ან მეტ ტრანზიტულ დამოკიდებულებას.
- რელაცია არის მესამე ნორმალურ ფორმაში თუ ის არის მეორე ნორმალურ ფორმაში და არ აქვს ტრანზიტული დამოკიდებულებები.
- ცხრილი არის მესამე ნორმალურ ფორმაში როცა ყველა არაგასაღები ატრიბუტი დამოკიდებულია მხოლოდ და მხოლოდ გასაღებზე.

მაგალითად განვიხილოთ შემდეგი სტრუქტურის ცხრილი

order(order_id, cust, cust_address, cust_contact, order_date, order_total)

აქ cust_address, cust_contact არიან ფუნქციონალურად დამოკიდებული cust-ზე, რომელიც არ არის გასაღები. ეს ცხრილი რომ მესამე ნორმალურ ფორმას შევუსაბამოთ, ეს ატრიბუტები უნდა წავშალოთ ამ ცხრილიდან და წავიღოთ სხვაგან.

ასევე უნდა გავითვალისწინოთ გამოთვლილი სვეტები. ავიღოთ, მაგალითად, ცხრილი, რომელიც შეიცავს ფასს, რაოდენობას და საბოლოო ფასს, სადაც საბოლოო ფასი გამოითვლება [რაოდენობა * ფასი]. იქიდან გამომდინარე რომ ამ ატრიბუტების მნიშვნელობები შეიძლება გამოითვალოს სხვა ორიდან, ისინი არ უნდა იყოს ბაზის ცხრილში.

მაგალითი:

- თანხა - მონეტარული ღირებულება სახელმწიფო ვალუტა, მძიმის შემდეგ 2 ადგილი
- გადახურდავების კოეფიციენტი - მძიმის შემდეგ 9 ციფრი
- ვალუტის თანხა - გამოხატული უცხოურ ვალუტაში, გამოითვლება [თანხა * კოეფიციენტზე].

მესამე ნორმალური ფორმის მიღწევა არის კარგი უმეტესობა პრაქტიკული მიზნებისთვის, მაგრამ შეიძლება იყოს ისეთი შემთხვევები, როდესაც უფრო შორსაც ღირს წასვლა.

ბოის-კოდის ნორმალური ფორმა

ცხრილი ბოის-კოდის ნორმალურ ფორმაშია მაშინ და მხოლოდ მაშინ თუ ის არის მესამე ნორმალურ ფორმაში და ყველა დეტერმინანტი არის კანდიდატი გასაღები.

- ანომალიები ხდება თუ ცხრილი მესამე ნორმალურ ფორმაშია და არის კომპოზიტური გასაღები, რომლის ნაწილიც არის დეტერმინანტი რომელიც არ არის კანდიდატი გასაღები.
- ეს შეიძლება გამოიხატოს R(A,B,C), C⟶A სადაც:
    - რელაცია შეიცავს ატრიბუტებს A, B და C
    - A და B არიან კანდიდატი გასაღებები
    - C არის დეტერმინანტი A -სთვის (A ფუნქციონალურად დამოკიდებულია C -ზე)
    - C არ არის არცერთი გასაღების ნაწილი
- ანომალიები შეიძლება მოხდეს როდესაც რელაცია შეიცავს რამდენიმე კანდიდატ გასაღებს, სადაც:
    - გასაღებები შეიცავს ერთზე მეტ არტიბუტს
    - ატრიბუტი არის საერთო ერთზე მეტი გასაღებისთვის

განვიხილოთ შემდეგი ცხრილი:

schedule(campus, course, class, time, room/bldg)

campus	course	class	time	room/bldg
East	English 101	1	8:00-9:00	212 AYE
East	English 101	2	10:00-11:00	305 RFK
West	English 101	3	8:00-9:00	102 PPR

ყურადღება მიაქციეთ რომ არცერთ შენობას უნივერსიტეტისას არ აქვს ერთნაირი სახელები, მაგრამ კამპუსები დამოკიდებულია room/bldg-ზე. როგორც დეტერმინანტი არ არის კანდიდატი გასაღები ეს ცხრილი არ არის ბოის-კოდის ნორმალურ ფორმაში.

ეს ცხრილი უნდა დაიშალოს შემდეგ რელაციებად:

R1(course, class, room/bldg, time)

R2(room/bldg, campus)

განვიხილოთ შემდეგი მაგალითი:

ცხრილს აქვს შემდეგი კანდიდატი გასაღები:

- (სტუდენტი#, კურსი#)
- (სტუდენტი#, კურსის_სახელი)
- (სტუდენტის_სახელი, კურსი#) - ეს ნიშნავს რომ სტუდენტის სახელი არის უნიკალური იდენტიფიკატორი
- (სტუდენტის_სახელი, კურსის_სახელი) - ეს ნიშნავს რომ კურსის სახელი არის უნიკალური იდენტიფიკატორი.

ეს რელაცია არის მესამე ნორმალურ ფორმაში მაგრამ არა ბოის-კოდის, შემდეგი დამოკიდებულებების გამო

- სტუდენტი# - >სტუდენტის სახელი
- კურსის# -> კურსის სახელი

enrol(student#, s_name, course#, c_name, date_enrolled)

მეოთხე ნორმალური ფორმა

ცხრილი არის მეოთხე ნორმალურ ფორმაში მაშინ და მხოლოდ მაშინ თუ ის არის ბოის-კოდის ფორმაში და შეიცავს მხოლოდ ერთ მრავალმნიშვნელოვან დამოკიდებულებას.

- ანომალიები შეიძლება წარმოიქმნას ცხრილებში ბოის-კოდის ფორმაში, თუ არის ერთზე მეტი მრავალმნიშვნელოვანი დამოკიდებულება
- თუ A↠B და A↠C, მაგრამ B და C არიან დაუკავშირებელი, მაგ. A↠(B,C) არის არასწორი, მაშინ ჩვენ გვაქვს მრავალმნიშვნელობიანი დამოკიდებულება
- A ცხრილი არის მეოთხე ნორმალურ ფორმაში თუ ის არის ბოის-კოდის ფორმაში და არ აქვს მრავალმნიშვნელოვანი დამოკიდებულება.

ავიღოთ შემდეგი ცხრილი:

info(employee#, skills, hobbies)

Employee #	Skills	Hobbies
1	პროგრამირება	გოლფი
1	პროგრამირება	ბოულინგი
1	ანალიზი	გოლფი
1	ანალიზი	ბოულინგი
2	ანალიზი	გოლფი
2	ანალიზი	ბილიარდი
2	მენეჯმენტი	გოლფი
2	მენეჯმენტი	ბილიარდი

ეს ცხრილი არის რთულად მართვადი, რადგან ახალი ჰობის დამატება ითხოვს რამდენიმე ახალ სტრიქონს თითოეულ თვისებასთან დაკავშირებულს. ეს პრობლემა იქმნება ორი მრავალმნიშვნელოვანი დამოკიდებულების გამო EMPLOYEE# ↠SKILLS და

EMPLOYEE#↠HOBBIES.ბევრად უკეთესი იქნებოდა დაგვეშალა ცხრილი შემდეგ ნაწილებად:

```
skills(employee#, skill)
hobbies(employee#, hobby)
```

მეხუთე ნორმალური ფორმა

ცხრილი მეხუთე ნორმალურ ფორმაშია თუ ის არის მეოთხე ნორმალურ ფორმაში და არ შეიძლება ქონდეს უდანაკარგო ჩაშლა ნებისმიერი რაოდენობის უფრო პატარა ცხრილებში.

- ... და ყველა კავშირის დამოკიდებულება არის კანდიდატი გასაღების შედეგი.
- ... არ არის დაწყვილებული ციკლური დამოკიდებულებები პირველად გასაღებში რომელიც შედგენილია სამი ან მეტი ატრიბუტისგან.

- ანომალიები შეიძლება მოხდეს ცხრილში რომელიც არის მეოთხე ნორმალურ ფორმაში თუ პირველადი გასაღები შედგება სამზე მეტი ატრიბუტისგან;
- მეხუთე ნორმალური ფორმა დაფუძნებულია კავშირის დამოკიდებულების კონცეპტზე - თუ ცხრილი არ შეიძლება იყოს დაშლილი მეტად, მაშინ ის არის მეხუთე ნორმალურ ფორმაში.
- დაწყვილებული ციკლური დამოკიდებულება ნიშნავს:
  - უნდა იცოდე ორი მნიშვნელობა;
  - ყოველი ერთისთვის უნდა იცოდე დანარჩენი ორი.

განვიხილოთ შემდეგი მაგალითი:

```
buying(buyer, vendor, item)
```

buyer	vendor	item
მაკა	ილია	მაისური
ნინო	ილია	მაისური
მაკა	ვაჟა	ჯინსი
ნინო	ვაჟა	ჯინსი
მაკა	ვაჟა	ფეხსაცმელი

პრობლემა იმაში მდგომარეობს რომ პირველად გასაღებში არის ციკლური დამოკიდებულებები. იმისთვის რომ გავიგოთ ნივთი, უნდა ვიცოდეთ მყიდველი და გამყიდველი, და რომ გავიგოთ გამყიდველი, უნდა ვიცოდეთ მყიდველი და ნივთი და ა.შ.

ამის გასასწორებლად ეს ცხრილი უნდა დაიშალოს სამ ნაწილად, მყიდველი-გამყიდველი, მყიდველი-ნივთი, გამყიდველი-ნივთი.

მეექვსე ნორმალური ფორმა

ცხრილი მეექვსე ნორმალურ ფორმაშია თუ ის არის მეხუთე ნორმალურ ფორმაში და ყველა კონსტრეინტი და დამოკიდებულება რომელიც უნდა ახლდეს ცხრილს შეიძლება იძულებით განისაზღვროს დომეინ კონსტრეინტების და გასაღების კონსტრეინტების დაზუსტებით.

თუ ყველა კონსტრეინტი ცხრილზე არის ლოგიკურ ურთიერთკავშირში განსაზღვრის გასაღებების და დომენების.

- დომეინ კონსტრეინტი არის კონსტრეინტი რომელიც იღებს A არტიბუტს R-დან რაიმე მოცემული დომენისთვის.
- გასაღები კონსტრეინტი არის კონსტრეინტი რომელიც ადგენს R-ის გასაღებს A,B,...,C სიმრავლიდან.
- ეს სტანდარტი შემოიღო რონ ფაგინმა 1981 წელს, მაგრამ მას არ გაუკეთებია არანაირი

ჩანაწერი მრავალმნიშვნელობიანი დამოკიდებულებების, კავშირების დამოკიდებულებების ან ფუნქციონალური დამოკიდებულებების შესახებ და არც დემონსტრირება მოუხდენია როგორ უნდა მიიღწეს ეს ნორმალური ფორმა. მაგრამ მან აღნიშნა რომ ფორმის მიღწევა ხშირად შეუძლებელია.

- თუ რელაცია R არის მეექვსე ნორმალურ ფორმაში, მაშინ საკმარისია რომ დომეინ და გასაღებების კონსტრეინტები შემოვიტანოთ R-ში და ყველა სხვა ავტომატურად შემოვა. ამ კონსტრეინტების შემოტანა ძალზედ ადვილია. უფრო ზუსტად, ამ კონსტრეინტების შემოტანა ნიშნავს, შემოწმდეს მათი მნიშვნელობები, თუ არიან დასაშვები. გასაღები კონსტრეინტების შემოტანა კი მნიშვნელობების უნიკალურობას გულისხმობს.
- სამწუხაროდ უმეტესობა რელაციები არ არის ამ ნორმალურ ფორმაში პირველ ადგილზე. მაგალითად, წარმოიდგინეთ არსებობს კონსტრეინტი R-ზე, რომელიც აიძულებს რომ R უნდა შეიცავდეს 10 სტრუქტურას მაინც. ეს კონსტრეინტი არ არის არც დომენის და არც გასაღების კონსტრეინტებისგან წარმოქმნილი, ე.ი. R არ არის მეექვსე ნორმალურ ფორმაში. ფაქტია რომ ყველა რელაცია არ დაიყვანება ამ ნორმალურ ფორმამდე.

დე-ნორმალიზაცია

დე-ნორმალიზაცია არის პროცესი, რომელიც კეთდება უნაკლოდ ნორმალიზებული ბაზის ასწრაფების მიზნით. დე-ნორმალიზაცია არის ბუნებრივი და საჭირო მონაცემთა ბაზის დიზაინი, მაგრამ უნდა მიყვეს სწორ ნორმალიზაციას.

ნორმალიზაციის ძირითადი იდეა მდგომარეობს იმაში რომ, მონაცემთა ბაზის დიზაინერმა უნდა დაისახოს მიზნად მეხუთე ნორმალური ფორმა. მაგრამ ეს რეკომენდაცია არ უნდა იყოს დაკანონებული. ზოგჯერ არის კარგი მიზეზები რომ გადავუხვიოთ ნორმალიზაციის პროცესს... მოთხოვნა არის რომ რელაციები იყო მინიმუმ პირველ ნორმალურ ფორმაში მაინც. მონაცემთა ბაზის დიზაინი არის ძალიან რთული ამოცანა. ნორმალიზაციის თეორია არის კარგი დამხმარე ამ პროცესში, მაგრამ ეს არ არის

უნივერსალური საშუალება. ყველას ვინც აკეთებს ბაზის დიზაინს რეკომენდირებულია იცნობდეს ნორმალიზაციის ძირითად ტექნოლოგიებს. მაგრამ არ არის აუცილებელი რომ დიზაინი იყოს დაფუძნებული ნორმალიზაციის მხოლოდ ამ ტექნიკაზე.

1970 -80 წლებში როდესაც კომპიუტერის ტექნიკური შესაძლებლობები იყო ძალიან მცირე, ძვირი და ნელი, ხშირად იყო რეკომენდირებული ბაზის დე-ნორმალიზაცია, რათა მიეღწია საჭირო სისწრაფემდე, მაგრამ ამ შემთხვევაში ხდებოდა რაღაცის დათმობა (მოდიფიკაციის ანომალიები). შედარებისთვის 21 საუკუნეში კომპიუტერები არის ძალიან კომპაქტური, იაფი და სწრაფი. როდესაც ეს უკავშირდება რელაციური ბაზის ძრავას, ნორმალიზებული ბაზა არის ხშირად დასაშვები და ნაკლები საჭიროებაა დე-ნორმალიზაციისა.

Company	**City**	**State**	**Zip**
Acme Widgets	New York	NY	10169
ABC Corporation	Miami	FL	33196
XYZ Inc	Columbia	MD	21046

ეს ცხრილი არ არის მესამე ნორმალურ ფორმაში, რადგან ქალაქი და შტატი არის დამოკიდებული ზიპ კოდზე. ის რომ ჩაისვას მესამე ნორმალურ ფორმაში, ორი ცაკლე ცხრილი უნდა შეიქმნას, ერთი შეიცავდეს კომპანიის სახელს და ზიპ კოდს, მეორე ქალაქს, შტატს, ზიპ კოდს.

წინასწარ განზრახული დე-ნორმალიზაცია არის ადგილი, სადაც სისწრაფის ოპტიმიზაცია ხდება. თუ ხშირად იღებთ ინფორმაციას მხოლოდ ერთი ცხრილიდან, შესაძლებელია მოხდეს მონაცემთა დუბლიკაცია სიჭარბით. დე-ნორმალიზაცია ყოველთვის ხდის თქვენს სისტემას პოტენციურად ნაკლებ ეფექტურს და მოქნილს, ე.ი. დე-ნორმალიზაცია არ არის მაინც და მაინც საჭირო, მაგრამ არც უმნიშვნელო.

არსებობს ტექნიკა სისწრაფის გაუმჯობესებისთვის, რომელიც ზეგავლენას ახდენს ჭარბ ან გამოთვლილ მონაცემებზე. ზოგიერთი ამ ტექნიკიდან არღვევს ნორმალიზაციის წესებს, ზოგიერთი არა. ზოგჯერ რეალური სამყაროს მოთხოვნები გვაიძულებს დავარღვიოთ წესები. სწორად დარღვევა ნორმალიზაციის წესებისა სისწრაფისთვის დასაშვებია, მაგრამ უნდა იყოს ასევე მისაღები.

შედგენილი ველი არის ველი რომლის მნიშვნელობა არის ორი ან მეტი ველის ნაერთი. ასეთი ველების ღირებულება არის ადგილი, რომელსაც ისინი იკავებენ და კოდი, რომელიც ამუშავებს მათ.

მაგალითად, თუ ბაზას აქვს ცხრილი მისამართით, რომელიც შეიცავს ქალაქს და შტატს, შეგიძლიათ შექმნათ შედგენილი ველი (ქალაქი_შტატი) რომელიც შეიცავს ქალაქის და შტატის სახელების კონკატენაციას. დალაგება და მოთხოვნები ქალაქი_შტატი-ზე არის ბევრად სწრაფი ვიდრე იგივე დალაგება და მოთხოვნა ცალ-ცალკე ველებზე - ზოგჯერ 40ჯერ სწრაფიც.

შედგენილი ველის უარყოფითი მხარე არის ის, რომ გჭირდებათ დაწეროთ კოდი, რომელიც განაახლებს ამ ველს როდესაც შეიცვლება ქალაქი ან შტატის სახელი. ეს არ არის რთული, მაგრამ მნიშვნელოვანია არ იყოს შეცდომები, ან სიტუაცია სადაც ინფორმაცია იცვლება ხოლო შედგენილი ველი არა.

შეჯამებული ველი არის ველი ცხრილში რომლის მნიშვნელობა დაფუძნებულია დაკავშირებულ რამდენიმე ცხრილის ჩანაწერთან. შეჯამებული ველები ანადგურებს განმეორებად და დროის ხარჯვად გადაკვეთის ცხრილების კალკულაციას და ქმნის გამოთვლილ მნიშვნელობებს, რომელიც პირდაპირ არის ხელმისაწვდომი მოთხოვნისას, დალაგებისას და რეპორტებისას ახალი პროგრამის შექმნის გარეშე. ერთი ცხრილის ველი რომელიც აჯამებს მნიშვნელობებს მრავალი დაკავშირებული ცხრილიდან არის ძლიერი ოპტიმიზაციის საშუალება. შეჯამებული ველები არ არღვევს ნორმალიზაციის წესებს. ნორმალიზაცია ხშირად არასწორად წარმოადგენს გამოთვლით მნიშვნელობებს, რომელიც აიძულებს ხალხს გვერდი აუარონ შეჯამებულ ველებს.

არის ორი ღირებულება რომელიც უნდა გავითვალისწინოთ ასეთი ველების შექმნისას: კოდირების დრო, რომელიც მოუვლის ამ მონაცემებს სწორად და ადგილი სადაც უნდა შეინახოს ეს ველი:

- ინვოისისთვის ინვოისის თანხა არის მთლიანი ღირებულება ყველა ინვოისის ჩანაწერისთვის
- ანგარიშისთვის ანგარიშის ბალანსი იქნება ჯამი თანხა გადამრავლებული ინვოისზე და გადასახადის ჩანაწერები ამ ანგარიშისთვის.

შეჯამებული ცხრილები არის ცხრილი რომლის ჩანაწერები ჯამავს დიდი რაოდენობის დაკავშირებულ ჩანაწერებს. ეს ცხრილი გამოიყენება რეპორტინგის ოპტიმიზაციისთვის, მოთხოვნებისთვის და გადაკვეთის ცხრილის სელექციისთვის. ასეთი ცხრილები შეიცავენ რამდენიმე ჩანაწერის მიერ წარმოშობილ მონაცემებს და არ არის აუცილებელი იცავდეს ნორმალიზაციის წესებს. ხალხი ფიქრობს რომ ეს ჩანაწერები აუცილებლად უნდა იყოს დენორმალიზებული.

შეჯამებული ცხრილი რომ იყოს გამოყენებადი ის უნდა იყოს სწორი. ეს ნიშნავს რომ გჭრიდებათ განახლება ჯამური ჩანაწერისა, როდესაც მასთან დაკავშირებული ჩანაწერები იცვლება. ეს ამოცანა შეიძლება შესრულდეს პროგრამული კოდის მეშვეობით, ან მონაცემთა ბაზის ტრიგერით (სასურველია), ან სკრიპტით. ასევე აუცილებელია განახლოთ ჩანაწერები, როდესაც იცვლება რაიმე კოდში. ამ მონაცემების სისწორე მოითხოვს დამატებით სამუშაოს და ქმნის კოდის შეცდომებს, გასათვალისწინებელია როდესაც გადაწყვეტთ გამოიყენოთ ეს ტექნიკა.

როგორც მესამე ნორმალური ფორმის წესებში იყო აღწერილი ყველა რელაცია, რომელსაც აქვს საერთო პირველადი გასაღები უნდა იყოს ერთ ცხრილში. არის შემთხვევები, როდესაც ეს არ არის სასურველი. განვიხილოთ შემდეგი მაგალითი:

- ფინანსური კომპანია იძლევა სესხს და ჩანაწერი ინახება თითოეული კლიენტის გადახდების მიხედვით.
- თუ კლიენტი არ იხდის პერიოდულად, მაშინ მისი ანგარიში საჭიროებს დაჯარიმებას და სპეციალური ღონისძიებების გატარებას
- დაახლოებით 5% კლიენტებისა არის დაჯარიმებული ერთდროულად.

ეს ნიშნავს რომ 100 000 კლიენტიდან არის უხეშად 5 000 დაჯარიმებული. თუ ჯარიმა გატარდება იგივე ჩანაწერზე, მოითხოვება რომ მოიძებნოს 100 000 ჩანაწერში. ეს არ არის ეფექტური. ერთი მეთოდია განისაზღვროს ანგარიშის სტატუსი, რომელიც აიდენტიფიცირებს ანგარიში არის თუ არა დაჯარიმებული, მაგრამ გაუმჯობესება არის მინიმალური.

გადაწყვეტილება ამ შემთხვევაში არის ამ მონაცემების ცალკე ცხრილში გადატანა. თუ არის 5 000 დაჯარიმებული, გვექნება ცხრილი 5 000 ჩანაწერით. ჯარიმების ცხრილი არის კლიენტების შვილობილი. ეს იქნება შესაძლებელი რომ ჯარიმებს მიეცეს სხვა პირველადი გასაღები და დაუკავშირდეს გარე გასაღებით კლიენტების ცხრილს. რაც წარმოშობს ერთი ბევრთან კავშირს ერთი-ერთთან კავშირის მაგივრად. ეს რომ არ დაირღვეს გარე გასაღები და პირველადი გასაღები უნდა იყოს იგივე

ეს აღიწერება შემდეგი სტრუქტურით:

R (K, A, B, C, X, Y, Z) სადაც:

- ატრიბუტი K არის პირველადი გასაღები;
- ატრიბუტები A B C არის ყოველთვის
- ატრიბუტები X Y Z არის ხანდახან (მაგრამ ყოველთვის ჯგუფად, 3ვე ერთად)
- ატრიბუტები X Y Z მოითხოვს სპეციალურ დამუშავებას

დენორმალიზაციის შემდეგ ხდება ორი განცალკევებული რელაციის წარმოქმნა:

- **R1 (K, A, B, C)**
- **R2 (K, X, Y, Z)** სადაც K ასევე გარე გასაღებია

3. მონაცემთა ბაზის ნორმალიზების მხარდაჭერის სისტემა

მონაცემთა ბაზის ნორმალიზების პროცესის სირთულე შესაძლებელია დაძლეულ იქნეს მომხმარებლის მხარდამჭერი სისტემის შექმნით, რომელიც დაეხმარება ნაკლებად კვალიფიციურ მომხმარებელს, მოახდინოს მისი საკუთარი ბაზის ნორმალიზება ერთიანობის დარღვევის გარეშე.

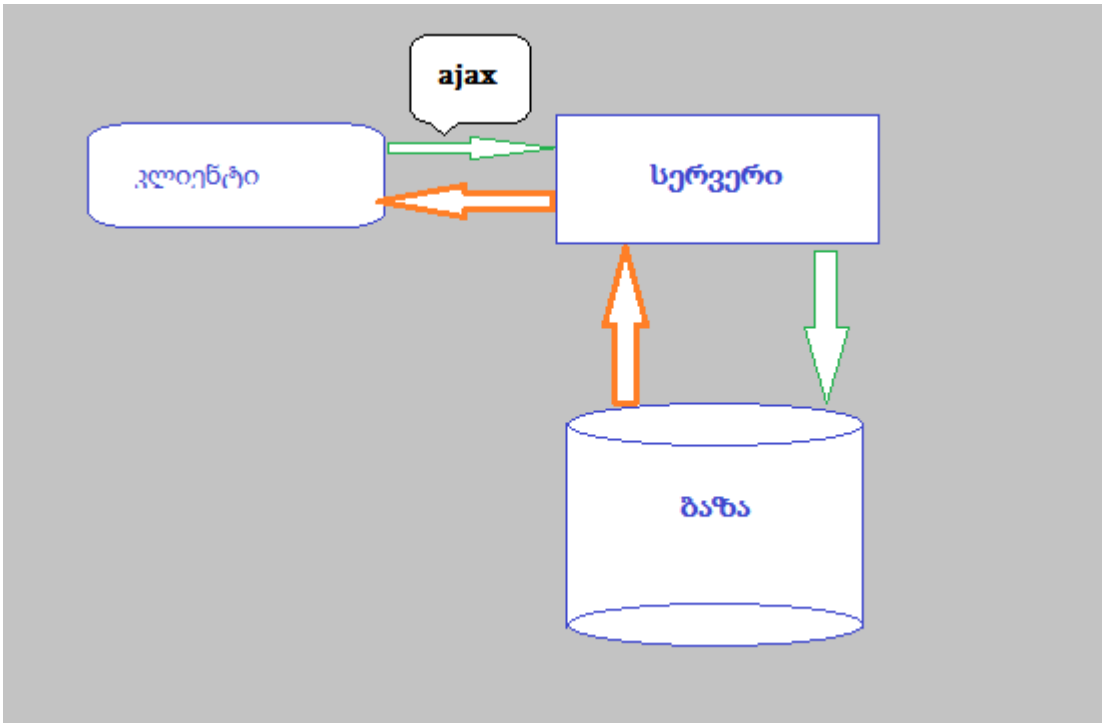
ამ მიზნით შემუშავდა სისტემა, რომელიც შეიცავს რელაციური ბაზის ნორმალიზებასთნ დაკავშირებულ ყველა პროცედურებს, თავისი ილუსტრაციებით.

სისტემის არქიტექტურა შედგება სამი დონისგან: კლიენტის, სერვერის და ბაზის დონეებისგან.

კლიენტის დონეზე გამოყენებული ტექნოლოგიები მოიცავს, ჰიპერტექსტური მონიშვნების ენას (HTML), კლიენტის მხარის სკრიპტს : JavaScript, jQuery. JavaScript-ის მეშვეობით ხორციელდება ajax მიმართვების შესრულება, სერვერის დონეზე.

სერვერის დონე მოიცავს სერვისს, რომელიც რეალიზებულია სერვერის მხარის სკრიპტულ ენაზე (PHP). მას აქვს საშუალება მიიღოს და დაამუშავოს მოთხოვნები კლიენტის დონიდან.

სერვერის დონე მიმართავს ბაზის დონეს საიდანაც ხდება ინფორმაციის ამოკითხვა და



გადაგზავნა კლიენტთან. ბაზად გამოყენებულია MySql მონაცემთა ბაზა.

მოცემული სისტემა მომხმარებელს აძლევს მითითებებს შექმნას ნორმალიზებული მონაცემთა ბაზა და მართოს ის.

მაგალითად, მომხმარებელს უნდა გააკეთოს შეერთება ორ ცხრილს R1 და R2-ს შორის.

ამისათვის ის ირჩევს შესაბამის წესს. სადაც გამოდის ამ წესთან დაკავშირებული ყველა შესაძლო ვარიანტი. აირჩევს შესაბამის კავშირს და პროგრამა კარნახობს მას თუ როგორი უნდა იყოს მოცემული კავშირი.

ასევე მოყვანილია მაგალითი რათა მომხმარებელს გაუადვილდეს მისი გამოყენება.

დასკვნა

პროექტის მიზანი იყო შეგვექმნა პროგრამული უზრუნველყოფა რომელიც დაეხმარებოდა მომხმარებელს რელაციური ბაზის აგებაში, მის ნორმალიზაციაში და ბაზის დიზაინის უკეთ წარმოდგენაში. ამიტომ ჩვენ შევქმენით პროგრამული უზრუნველყოფა რომელიც იყენებს PHP, JavaScript, jQuery, Ajax, MySql ტექნოლოგიებს არაპროფესიონალ მომხმარებელს ეხმარება მონაცემთა ბაზის დიზაინის აგებასა და მის მართვაში. სისტემა შეიძლება შემდგომში განვითარდეს და მოერგოს სხვადასხვა პლატფორმებს.

ლიტერატურა

1. E.F. Codd (1970). "A relational model of data for large shared data banks". In: *Communications of the ACM archive*. Vol 13. Issue 6(June 1970). pp.377-387.
2. *Introducing databases* by Stephen Chu, in Conrick, M. (2006) *Health informatics: transforming healthcare with technology*, Thomson, ISBN 0-17-012731-1, p. 69.
3. Date, C. J. (June 1, 1999). "When's an extension not an extension?". *Intelligent Enterprise* **2** (8).
4. Zhuge, H. (2008). *The Web Resource Space Model*. Web Information Systems Engineering and Internet Technologies Book Series **4**.Springer. ISBN 978-0-387-72771-4.
5. *Data Modeling*

http://www.liberty.edu/media/1414/%5B6330%5DERDDataModeling.pdf

6. *The Relational Data Model*

http://www.tonymarston.co.uk/php-mysql/database-design.html

7. *Database normalization*

https://en.wikipedia.org/wiki/Database_normalization