ობიექტ-რელაციური მაპირება (ორმ)

გურამ ქოჩლაძე

Email: gugakochladze@gmail.com

**რეზიუმე** — ეს არის პრობლემა, რომელიც დეველოპერს აუცილებლად შეხვდება თავისი მოღვაწეობის საწყის ეტაპებზე. საუბარია ობიექტზე-ორიენტირებულ პროგრამულ ენასა და რელაციურ მონაცემთა ბაზას შორის კავშირზე. თითქოს არაფერი განსაკუთრებული, მაგრამ რელაციური ბაზა ხომ ეწინააღმდეგება ობიექტზე-ორიენტირებული პროგრამირების პრინციპებს?! მაშინ როგორ უნდა გავცვალოთ ინფორმაცია ამ ორ ობიექტს შორის?! ამის გარკვევას საჭიროებდა ყველა შესაძლო მიდგომის მოძიება, შეფასება, დადებითი და უარყოფითი მხარეების გამოვლენა. ასევე მნიშვნელოვანი იყო პროგრამისტების შეფასებები ამ ტექნიკების გამოყენების შემდეგ, ვინაიდან ბევრი პრობლემური დეტალი მხოლოდ მუშაობის პროცესში იჩენს თავს. საბოლოოდ მივედით იმ დასკვნამდე, რომ ობიექტ-რელაციური მაპირების ტექნიკის გამოყენება არის დღეს-დღეობით ყველაზე მიღებული და ამავდროულად პოპულარული მეთოდი. ამ მეთოდს იყენებენ სამუშაოს საგრძნობლად გასამარტივებლად, დეველოპინგის დროის შესამცირებლად, მაგრამ ეს ყველაფერი ხდება ოოპ-ის ზოგიერთი პრინციპის დარღვევის ხარჯზე.

**გასაღები სიტვვები -** ინფორმაციის მიმოცვლა, მაპირება, ობიექტზე-ორიენტირებული პროგრამირება, რელაციური ბაზები

2017-2018 სასწ.წელი თსუ კომპიუტერულ მეცნიერებათა დეპარტამენტი

—————————— ◆ ——————————

# 1 შესავალი

ინფორმაციის მიმოცვლის გარეშე 21-ე საუკუნეში ცხოვრება ფაქტობრივად წარმოუდგენელია. მას შემდეგ, რაც ჩვენს ცხოვრებაში უდიდესი ადგილი დაიკავა ტექნოლოგიამ, ინფორმაციის გაცვლის საკითხი კიდევ უფრო გააქტიურდა. ერთია როდესაც ადამიანი სხვა ადამიანს გადასცემს რაიმე ინფორმაციას, რადგან ეს ორივესთვის გასაგებია, მაგრამ რა ხდება მაშინ, როდესაც ობიექტი მეორე ობიექტს აწვდის ისეთ ინფორმაციას, რომელიც მეორესთვის გაუგებარია? იქმნება პრობლემა, შეუძლებელი ხდება მიღებული ინფორმაციის წაკითხვა და გამოყენება. კონკრეტულ შემთხვევაში კვლევის არეალი წარმოადგენს ინფორმაციის გაცვლას მონაცემთა ბაზასა და ობიექტზე-ორიენტირებულ აპლიკაციას შორის. ამ პრობლემის გადაჭრის ყველაზე პოპულარული გზაა ობიექტ-რელაციური მაპირება.

ობიექტ-რელაციური მაპირება წმინდა გაგებით არის პროგრამირების ტექნიკა რომელიც მხარს უჭერს შეუთავსებელი ტიპების გარდაქმნას ობიექტზე-ორიენტირებული პროგრამირების ენებში, განსაკუთრებით მონაცემთა ბაზისა და პროგრამულ ობიექტებს შორის. შესაძლებელია ორმ ფრეიმვორკის გამოყენება მოდელ ობიექტის გადასაცემად რელაციურ ბაზაში ან პირიქით, ბაზიდან მის დასაბრუნებლად. ორმ ფრეიმვორკი იზრუნებს ინფორმაციის ერთი მდგომარეობიდან სხვა მდგომარეობაში გარდაქმნაზე, რომელიც სხვაგვარად შეუთავსებელია. ობიექტ-რელაციური მაპირების ხელსაწყოების უმეტესობა ძლიერ არის დამოკიდებული მონაცემთა ბაზის და ობიექტების მეტაინფორმაციაზე, ეს ნიშნავს, რომ ობიექტები არ საჭიროებენ ცოდნას ბაზაზე და არც ბაზა საჭიროებს რაიმეს ცოდნას იმის შესახებ თუ როგორია მონაცემთა სტრუქტურა აპლიკაციაში. ორმ უზრუნველყოფს წმინდა განცალკევებას კარგად აწყობილ მონაცემთა აპლიკაციაში, სადაც ბაზა და აპლიკაცია შეძლებენ ცალ-ცალკე იმუშაონ თავიანთთვის გასაგებ ინფორმაციის ფორმატთან.

# 2 განხორციელებული სამუშაო (არსებული მდგომარეობა)

1. პირველ რიგში საჭიროა გავიგოთ თუ რა საშუალებებით ხდება ინფორმაციის მიმოცვლა რელაციურ ბაზასა და ობიექტზე-ორიენტირებულ ენას შორის.
2. გავეცნოთ არსებული საშუალებების კარგ და ცუდ მხარეებს.
3. გავეცნოთ თითოეულის სწორ ადგილას გამოყენების პრაქტიკას და არეალს.
4. გავეცნოთ ამ მეთოდებზე დაფუძნებულ პროგრამირების ტექნიკებს და შევაფასოთ ისინი.

ხალხის ნაწილი ფიქრობს, რომ ორმ არის ანტი-პატერნი, რომელიც არღვევს ობიექტზე-ორიენტირებული პროგრამირების ყველა პრინციპს, ანაწევრებს ობიექტებს და მონაცემთა პასიურ „გროვად“ გარდაქმნის. მიიჩნევენ, რომ ეს მეთოდი სხვა მეთოდებთან შედარებით ნელია, რთულდება მონაცემთა ბაზის განახლება და იმის ნაცვლად რომ მოახდინოს ბაზათან კავშირის ინკაფსულაცია ობიექტის შიგნით, მას პირიქით ობიექტის გარეთ გამოაქვს. ობიექტის ერთ ნაწილი ინახავს ინფორმაციას, მეორე ნაწილი კი იმპლემენტირებული ორმ ძრავში, პასუხისმგებელია ამ ინფორმაციის გამოყენებასა და რელაციურ ბაზაში გადაგზავნაზე. ორმ-ს ალტერნატიული გზებია: ობიექტების გამოყენება ინფორმაციის შესანახად და SQL-ის გამოყენება მოდელში. ეს უკანასკნელი გულისხმობს, რომ მოდელის ფენის გამოყენებით მოვახდინოთ ინფორმაციის ინკაფსულაცია კოდის ერთ ნაწილში.

ჩატარებული კვლევების შედეგად ნათლად გამოჩნდა, რომ უარყოფითი მხარეების მიუხედავად ორმ-ს გააჩნია უფრო მეტი დადებითი ფუნქცია, რაც გადაწონის ჩემს არჩევანს თავისისკენ.

# 3 სისტემური მოდელი

როდესაც ვსაუბრობთ მეთოდის შერჩევაზე რაიმეს დავალების შესასრულებლად, საჭიროა განვიხილოთ არსებული ვარიანტები და შევარჩიოთ ჩვენთვის სასურველი პრინციპების გამოყენებით. შესაძლოა მეთოდი ყველა პრინციპს აკმაყოფილებდეს ერთის გარდა, მაგრამ ჩვენთვის ეს უკანასკნელი იყოს ყველაზე მნიშვნელოვანი.

თუ ავირჩევთ ორმ-ს გასათვალისწინებელია რომ მოგვიწევს: ობიექტზე-ორიენტირებული პროგრამირების პრინციპების დარღვევა, რელაციურ ბაზასთან მიმთითებლით წვდომა, რასაც თავად რელაციური ბაზა გმობს, სტრუქტურის და ერთიანობის განსხვავება. მაგრამ, ამასთანავე გვაქვს უფრო მეტი დადებითი მხარე.

თუ ავირჩევთ ტრადიციულ ტექნიკას, მაშინ ჩვენ შევინარჩუნებთ ობიექტზე-ორიენტირებული პროგრამირების პრინციპებს, მაგრამ ამასთანავე დავკარგავთ იმ სიკეთეებს რის გამოც ორმ შეიქმნა.

მეთოდის შერჩევის შემდეგ საჭიროა განვიხილოთ ის პროგრამული ტექნიკები, რომლებიც ამ მეთოდზეა დაფუძნებული. ამ შემთხვევაშიც უნდა შევარჩიოთ ის ტექნიკა, რომელიც ჩვენს კონკრეტულ მოთხოვნებს აკმაყოფილებს. თითოეულ პროგრამული ენისთვის მაგალითად როგორიცაა: C++, Java, .Net პლატფორმა, PHP, Python, Ruby და ა.შ, არსებობს არაერთი ტექნოლოგია, რაც არჩევანის დიდ თავისუფლებას იძლევა.

# 4 ამოცანის დასმა

ჩვენი ამოცანაა რელაციური ბაზიდან ობიექტზე-ორიენტირებულ ენაში ინფორმაციის მიმოცვლა. ამ ამოცანას კი გარკვეულ წინააღმდეგობებთან მივყავართ ვინაიდან არ არსებობს ერთი კონკრეტული და დომინანტური გზა. არჩევანის გაკეთება გვიწევს ჩვენი მოთხოვნებიდან, შესასრულებელი სამუშაოდან და პრინციპებიდან გამომდინარე.

# 5 პრობლემის გადაწყვეტა

ამ პრობლემის გადაწყევტას მე ვხედავ ობიექტ-რელაციური მაპირების გამოყენებაში. მიუხედავად იმისა, რომ იგი არღვევვს ობიექტზე-ორიენტირებული პროგრამირების პრინციპებს, მე უპირატესობას ვანიჭებ მის დადებით მხარეებს. ორმ საგრძნობლად აჩქარებს დეველოპმენტს, თავიდან გვარიდებს SQL კოდების ხელახლა დაწერას და გამოყენებას, ამცირებს დეველოპმენტზე დახარჯულ თანხას და ა.შ

მას შემდეგ რაც ჩვენ ავირჩევთ ამ მეთოდს, კიდევ ერთი დიდი პლიუსი არის ის რომ თითქმის ყველა პროგრამულ ენაზე უკვე შექმნილია არაერთი პროგრამული ტექნიკა. გვეძლევა საშუალება გამოვიყენოთ ის, რომელიც ყველაზე უკეთ იქნება ჩვენზე მორგებული. ეს იგივეა რაც დეველოპერისთვის პროგრამული ენის შერჩევა.

მაგალითად, თუ დეველოპერი მუშაობს .NET პლატფორმასთან და ესაჭიროება რელაციურ ბაზასთან ინფორმაციის მიმოცვლა, შეუძლია გამოიყენოს საკმაოდ პოპულარული ორმ ფრეიმვორკები როგორებიცაა: Entity Framework, LINQ, Dapper, NHibernate. ან სურვილისამებრ ნაკლებად პოპულარული: nHydrate, XPO, SqlMapper, MapLess და ა.შ

# 6 ანალიზი

განვიხილოთ ის მახასიათებლები, რომლებიც გააჩნია ობიექტ-რელაციურ მაპირებას:

დომეინის მოდელის იმპლემენტაცია - ეს ნიშნავს, რომ ჩვენი მოდელები დაფუძნებულია რეალურ ბიზნეს-კონცეფციებზე და არა ბაზის კონკრეტულ სტრუქტურაზე.

კოდის რაოდენობის შემცირება - ორმ ხელსაწყოები გვთავაზობენ სერვისებს, რომლებიც გვეხმარებიან ფოკუსირება მოვახდინოთ ბიზნეს ლოგიკაზე და დრო არ ვხარჯოთ ერთი და იგივე კოდის თავიდან წერაზე.

ობიექტ-მოდელის ცვლილება - ეს ცვლილება ხდება ერთ კონკრეტულ ადგილას. როდესაც, ვანახლებთ ობიექტს, ორმ ავტომატურად იყენებს უკვე განახლებულ ობიექტს და არ საჭიროებს ცვლილებების შეტანას აპლიკაციის სხვადასხვა ფენაზე.

მოთხოვნათა შესაძლებლობების სიმდიდრე - ორმ ხელსაწყოები გვთავაზობენ ობიექტზე-ორიენტირებულ მოთხოვნათა ენას, რომელიც გამოიყენება მონაცემთა ბაზასთან სამუშაოდ. ეს დეველოპერს ეხმარება მეტი დრო დაუთმოს ობიექტ-მოდელებს, რადგან ხელსაწყო თავად გადათარგმნის მოთხოვნათა ენას ბაზასთან სამუშაო სინტაქსში.

ინფორმაციის ჩატვირთვა - საშუალება გვეძლება სრულიად ვაკონტროლოთ თუ რა ინფორმაცია ჩაიტვირთება თითოეული სცენარისთვის.

ქეშის მართვა - ობიექტები ქეშირებულია მეხსიერებაში, რაც ამცირებს დატვირთვას მონაცემთა ბაზაზე.

ტრანზაქციების მართვა და იზოლირება - ყველა ობიექტის ცვლილება ხდება ერთ დახურულ ტრანზაქციაში, რომელიც შეგვიძლია გადავაწოდოთ ან უკან დავაბრუნოთ. ასევე შესაძლებელია ერთდროულად რამოდენიმე აქტიური ტრანზაქციის ქონა მეხსიერებაში, რომლებიც ერთმანეთისგან სრულიად იზოლირებულია.

# დასკვნა

ობიექტ-რელაციური მაპირების გამოყენებით ჩვენ შევძლებთ ვიურთიერთოთ მონაცემთა ბაზასთან ობიექტზე-ორიენტირებული პროგრამირების ნებისმიერი ენით. გვექნება საშუალება ავტომატურად გარდავქმნათ ინფორმაცია ერთი მდგომარეობიდან მეორეში, ვისარგებლოთ გამზადებული სერვისებით, შევამციროთ დეველოპმენტის დრო და საწერი კოდი, ზოგადად გავიმარტივოთ სამუშაო. ამასთანავე არჩევანის გაკეთების დროს გასათვალისწინებელია უარყოფითი მხარეები.

ობიექტზე-ორიენტირებული პროგრამირების პრინციპების დარღვევა - ობიექტ-რელაციური მოდელი არ ანსხვავებს ინფორმაციის წვდომადობას, არღვევს ინკაფსულაციას, ბაზასთან კავშირი ხდება ობიექტის გარეთ, გამოიყენება შვილეულობის პრინციპი რასაც რელაციური ბაზა არ ემხრობა.

ინფორმაციის ფორმატის განხვავება - რელაციური მოდელი მკაცრად ეწინააღმდეგება მიმთითებლით წვდომას ინფორმაციასთან, რასაც პირიქით ოოპ ემხრობა.

ორმ ხელსაწყოს შესწავლა - სანამ ამ მეთოდს გამოვიყენებთ საჭიროა შევისწავლოთ ერთი-ერთი ორმ ხელსაწყო, რომელიც ჩვენს პროგრამულ ენაზე მუშაობს.

შრომისუნარიანობის შემცირება - როდესაც, ვაპირებთ დიდი ზომის მოთხოვნათა კოდის გამოყენებას, უკეთესი იქნება თუ ამას ჩვენით გავაკეთებთ, რადგან ორმ ხელსაწყომ შეიძლება დააყოვნოს ამ დავალების შესრულება.

ინფორმაციის ფორმატის განხვავება - რელაციური მოდელი მკაცრად ეწინააღმდეგება მიმმართველით წვდომას ინფორმაციასთან, რასაც პირიქით ოოპ ემხრობა.

# ბიბლიოგრაფია

|  |  |
| --- | --- |
| [1] | "Hibernate," [Online]. Available: http://hibernate.org/orm/what-is-an-orm/. |
| [2] | S. M. CRAIG RUSSELL, "ACM," May/June 2008. [Online]. Available: http://delivery.acm.org/10.1145/1400000/1394139/p16-russell.pdf?ip=176.221.129.25&id=1394139&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&CFID=844812143&CFTOKEN=77758532&\_\_acm\_\_=1514324428\_330a60d92e81fd6d5ce195a3. |
| [3] | I. O. Likhatskyi, "TAAC," [Online]. Available: http://taac.org.ua/files/a2013/proceedings/UA-1-Igor%20Likhatskyi-338.pdf. |
| [4] | M. Makai, "Full Stack Python," [Online]. Available: https://www.fullstackpython.com/object-relational-mappers-orms.html. |
| [5] | C. Richardson, "ACM Digital Library," April 2004. [Online]. Available: https://dl.acm.org/citation.cfm?id=1498783. |
| [6] | M. S., "LearnNowOnline," 28 August 2012. [Online]. Available: http://blogs.learnnowonline.com/2012/08/28/4-benefits-of-object-relational-mapping-orm/. |