Georgian Scientists
ქართველი მეცნიერები
Vol. 7 Issue 2, 2025
https://doi.org/10.52340/gs.2025.07.02.47

ობიექტურ–ორიენტირებული მონაცემთა ბაზების შესახებ

ცაცა ნამჩევაძე[1], ელზა ბიწაძე[2]

[1]აკაკი წერეთლის სახელმწიფო უნივერსიტეტი, ასოცირებული პროფესორი,
tsatsanamg@gmail.com,
[2]აკაკი წერეთლის სახელმწიფო უნივერსიტეტი, ასისტენტ პროფესორი,
elza.bitsadze@atsu.edu.ge

რეზიუმე

სტატიაში განხილულია მონაცემთა ბაზების გამოყენების მნიშვნელობა; რელაციურ მონაცეთა ბაზების ნაკლოვანი მხარეები დიდი მოცულობის მონაცემთა დამუშავებისას. აღნიშნულია, რომ ამ ნაკლოვანებების თავიდან აცილების ერთ–ერთ საშუალებას წარმოადგენს ობიექტურ–ორიენტირებული მონაცემთა ბაზები. ასეთი ბაზების არსი განისაზღვრება ობიექტურ–ორიენტირებული მიდგომით. ობიექტურ–ორიენტირებული მიდგომა გულისხმობს ობიექტურ–ორიენტირებულ პროექტირებას და ობიექტურ ორიენტირებულ პროგრამირებას. ჩამოყალიბებულია ობიექტურ–ორიენტირებული პროექტირებისა და ობიექტურ–ორიენტირებული პროგრამირების ძირითადი ცნებები.

მონაცემთა ბაზების აგების ობიექტურ–ორიენტირებული მიდგომისათვის აუცილებელია: 1).მონაცემთა ინკაფსულაცია ე.ი. კლასებისა და ობიექტების გამოყოფა; 2).ცხრილების სტრუქტურის განსაზღვრა; 3).მონაცემებისათვის კლასების მემკვიდრეობის შექმნა; 4).პოლიმორფიზმის უზრუნველყოფა და სხვა.

აღნიშნულია, რომ ობიექტურ–ორიენტირებულ მონაცემთა ბაზებს მიეკუთვნება მხოლოდ ის მონაცემთა ბაზები, რომელთა რეალიზაციის ყველა სტრუქტურული ელემენტი აგებულია ობიექტურ–ორიენტირებული მიდგომით. თუ ერთი მაინც სტრუქტურული ელემენტი არ გამოიყენებს ობიექტურ–ორიენტირებულ მიდგომას, მაშინ საქმე გვაქვს ობიექტურ–რელაციურ მონაცემთა ბაზებთან.

ნაშრომში ჩამოყალიბებულია რელაციურ მონაცემთა ბაზებთან შედარებით ოომბ–ის უპირატესობები. ესენია: 1).სისტემის მოდელირების უკეთესი შესაძლებლობები; 2).სტრუქტურის ადვილად გაფართოების უნარი, რაც განპირობებულია ახალი ტიპის მონაცემების (თვისებების) შექმნით, მემკვიდრეობით, ახალი კავშირების დამყარებით და მეთოდების კორექტირების საშუალებებით. 3).რეკურსული მეთოდების გამოყენების შესაძლებლობა დიდი მოცულობის მონაცემებზე ნავიგაციური მეთოდით მიმართვისას;

4).მწარმოებლობის ამაღლება 3–20–ჯერ და მეტად; 5).გამოყენების ფართო სპექტრი (მაგალითად, გამოიყენება მულტიმედიურ სისტემებში, გრაფებში) და სხვა.

აღსანიშნავია, რომ ობიექტურ–ორიენტირებული მონაცემთა ბაზები რელაციურ მონაცემთა ბაზებთან შედარებით საჭიროებს 4-5–ჯერ მეტ მეხსიერებას.

ობიექტურ-ორიენტირებულ მონაცემთა ბაზებში მონაცემებთან წვდომა და მოთხოვნების შექმნა ხდება პროგრამული კოდით, რომელიც დაწერილია ობიექტურ-ორენტირებულ ენაზე (მაგ. Java, C++, Python). ზოგიერთ თანამედროვე ობიექტურ-ორენტირებულ მონაცემთა ბაზას აქვს SQL ენის-ის გაფართოებული ვერსია.

სტატიაში მოცემულია თანამედროვე პირობებში ფართოდ გამოყენებული რამოდენიმე ობიექტურ–ორიენტირებული მონაცემთა ბაზების მართვის სისტემის სახეები. მოყვანილია პროგრამის ფრაგმენტები ერთ-ერთი პოპულარული მონაცემთა ბაზების მართვის სისტემისათვის-**Versant Object Database.**

საკვანძო სიტყვები: ობიექტურ–ორიენტირებული მონაცემთა ბაზები, ობიექტურ–ორიენტირებული მონაცემთა ბაზების მართვის სისტემა, მონაცემთა ობიექტური მოდელი, ობიექტების განსაზღვრის ენა, მოთხოვნების ობიექტური ენა.

თანამედროვე პირობებში ძალიან დიდი მნიშვნელობა აქვს მონაცემთა ბაზების გამოყენებას. იგი ეხება საზოგადოების საქმიანობის თითქმის ყველა სფეროს. ესენია: მეცნიერება, საბანკო სექტორი, ბიზნესი, ჯანდაცვა, სახელმწიფო მართვა და სხვა.

მონაცემთა ბაზები უზრუნველყოფს დიდი რაოდენობის ინფორმაციის სისტემატიზაციას, სწრაფ დამუშავებას, ანალიზს და სხვა. მისი საშუალებით შესაძლებელია ინფორმაციის დაცვა უნებართვო წვდომისგან, პარალელურად კი - მრავალმომხმარებლიანი წვდომის უზრუნველყოფა. მონაცემთა ბაზები ხელს უწყობს პროცესების ავტომატიზაციას - მცირდება ხელით შეყვანის საჭიროება, რაც ამცირებს შეცდომებს და ზრდის პროდუქტიულობას.

მონაცემთა რელაციური მოდელის სერიოზულმა ნაკლოვანებებმა გამოიწვია სხვა მოდელების ძებნის აუცილებლობა. მონაცემთა პროგრესულ და პერსპექტიულ მოდელს წარმოადგენს მონაცემთა ობიექტურ–ორიენტირებული მოდელი. მასში მონაცემთა ბაზა, მომხმარებლის ინტერფეისი და დანართების ალგორითმები აგებულია ობიექტურ–ორიენტირებული მიდგომის გამოყენებით.

მონაცემთა რელაციური მოდელის ავტორმა **ედგარ კოდმა** თავდაპირველად ჩამოაყალიბა 12 მოთხოვნა, რომლებსაც უნდა აკმაყოფილებდეს **მონაცემთა ბაზები (მბ).** თანამედროვე პირობებში იგი უნდა აკმაყოფილებდეს დამატებით 15–20-ზე მეტ ტექნიკურ, უსაფრთხოების, ოპტიმიზაციის და ინტეგრაციის მოთხოვნასაც. აღნიშნულის რეალიზება დაკავშირებულია რიგ სირთულეებთან.

დიდი მოცულობის მონაცემების დამუშავებისას ვლინდება რელაციურ მბ–ის ნაკლოვანებები. ამ ნაკლოვანებებს მიეკუთვნება:

1) მბ–ის სტრუქტურის სირთულე, რომელიც განპირობებულია ნორმალიზაციის ჩატარების აუცილებლობით;
2) დაბალი მწარმოებლობა, რომელიც გამოწვეულია გასაღების მიხედვით ძებნით. აღნიშნული 3–5–ჯერ ზრდის მიმართვის ოპერაციებს;
3) მონაცემთა ტიპების შეზღუდული რაოდენობა;

4) მონაცემთა წარმოდგენა მხოლოდ 2–განზომილებიანი ცხრილის სახით;
5) შეუძლებელობა იმისა, რომ მონაცემები განხილულ იქნას ფენებად (მაგალითად, „თანამშრომლები“ ერთ ფენად, ხოლო „მეცნიერ-მუშაკები“ და „მასწავლებლები“ დაქვემდებარებულ ფენად);
6) მონაცემთა დანაკარგები მრავალრიცხოვანი განახლებისას მე–3 ნორმალურ ფორმაში;
7) მონაცემთა შენახვის სხვა პარადიგმებთან შერწყმის სირთულე და სხვა.

ჩამოთვლილი ნაკლოვანებების თავიდან აცილების ერთ–ერთ საშუალებას წარმოადგენს **ობიექტურ–ორიენტირებული მონაცემთა ბაზები (ოომბ).** იგი მოითხოვს დიდი მოცულობის მეხსიერებას (20 გიგაბაიტზე მეტი) სისტემის კონსტრუირებისათვის.

1989 წელს გამოქვეყნდა „ოომბ–ის მანიფესტი“, რომელშიც გამოიყენებოდა ფორმულა:

ოომბმს=მბმს+ოოპე

სადაც ოომბს ნიშნავს ობიექტურ–ორიენტირებულ მონაცემთა ბაზების მართვის სისტემას;

ოოპე – ობიექტურ–ორიენტირებულ პროგრამირების ენას.

ოომბმს უნდა უზრუნველყოფდეს არაწრფივი სტრუქტურის ობიექტებს, სისტემის ადვილად გაფართოებას, გამოთვლების შესრულებას, მოთხოვნების ენებს. აღნიშნული მიიღწევა ინკაფსულაციითა და მემკვიდრეობით.

1991 წელს შეიქმნა ჯგუფი Object Database Managment Group (ODMG), რომელსაც უნდა ჩამოეყალიბებია ოომბ–ის სტანდარტი. ეს სტანდარტი ჯგუფმა წარადგინა 1993 წელს სახელწოდებით ODMG-3. მასში შედიოდა:

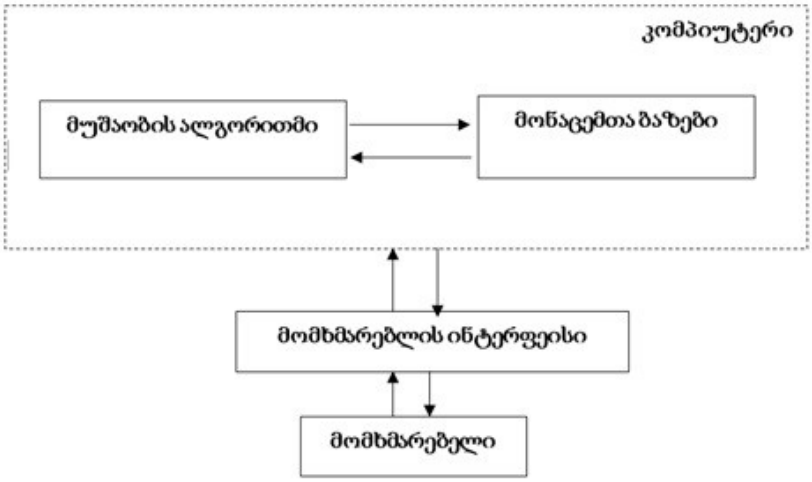
1) მონაცემთა ობიექტური მოდელი – Object Data Model (ODM);
2) ობიექტების განსაზღვრის ენა – Object Definition Language (ODL), რომელიც გამოყენებულია მონაცემთა მანიპულირებისათვის;
3) მოთხოვნების ობიექტური ენა – Object Query Language (ODL), რომელიც გამოყენებულია მოთხოვნების დამუშავებისათვის;
4) პროგრამირების ენების ინტერფეისები (C^{++} და სხვა).

ოომბ-ის არსი განისაზღვრება ობიექტურ–ორიენტირებული მიდგომით **(სურ.1).** იგი გულისხმობს ობიექტურ–ორიენტირებულ პროექტირებას და ობიექტურ ორიენტირებულ პროგრამირებას.

სურ. 1. ობიექტურ–ორიენტირებული მიდგომა

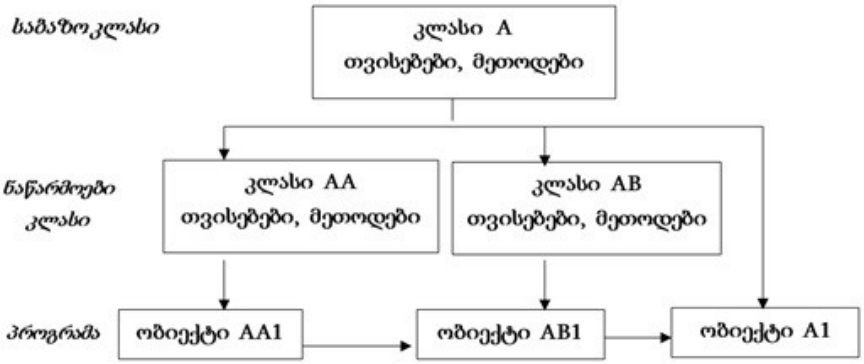
ობიექტურ–ორიენტირებული პროექტირება **(სურ.2)** არის პროექტირების მეთოდოლოგია, რომელიც თავის თავში აერთიანებს ცოდნის ბაზების ან მონაცემთა ბაზების დეკომპოზიციის (დაყოფის) პროცესს შემადგენელ ნაწილებად. ამასთან იგი მოიცავს

დასაპროექტებელი სისტემის ლოგიკურ და ფიზიკურ წარმოდგენებს, სტატიკურ და დინამიურ მოდელებს.



სურ. 2. ობიექტიურ–ორიენტირებული პროექტირება

ობიექტურ–ორიენტირებული პროგრამირება **(სურ.3)** არის პროგრამირების მეთოდოლოგია, რომელიც ემყარება პროგრამის წარმოდგენას ობიექტების ერთობლიობის სახით. მასში ყოველი ობიექტი წარმოადგენს კლასის ეგზეპლიარს. კლასები კი იქმნება იერარქიულად მემკვიდრეობის პრინციპის მიხედვით.



სურ. 3. ობიექტიურ–ორიენტირებული პროგრამირება

განვიხილოთ ოოპ–ის რამოდენიმე დამახასიათებელი ცნება. კლასი არის ლოგიკური კონსტრუქცია, ხოლო ობიექტი მისი ფიზიკური რეალიზაცია. კლასს გააჩნია თვისებები (მონაცემები), მეთოდები (ალგორითმები).

ოოპ ეყრდნობა 3 ძირითად კონცეფციას ესენია: **ინკაფსულაცია, მემკვიდრეობა და პოლიმორფიზმი.**

ინკაფსულაცია არის კლასის თვისებების ჩაკეტვა მასზე წვდომის (მიმართვის) მიხედვით. კლასის თვისებებზე წვდომა შესაძლებელი უნდა იყოს მხოლოდ კლასის მეთოდების საშუალებით. ამ მეთოდებს დაშვების ან წვდომის მეთოდებს უწოდებენ.

მემკვიდრეობა ნიშნავს, რომ კლასები იქმნება იერარქიული პრინციპის მიხედვით რომელიმე საბაზო (საწყისი) კლასიდან. წინაპარი კლასი მიჩნეულია როგორც საბაზო კლასი, ხოლო მემკვიდრე კლასი-როგორც წარმოებული კლასი. მემკვიდრე კლასი იძენს თვისებებსა და მეთოდებს წინაპარი კლასისაგან, ამასთან მასში შეიძლება გამოცხადდეს დამატებითი თვისებები და მეთოდები.

პოლიმორფიზმი საშუალებას იძლევა გამოვიყენოთ მეთოდები ერთი და იგივე სახელით როგორც საბაზო კლასში, ასევე წარმოებულ კლასებში. პოლიმორფიზმის გამოყენება განპირობებულია პროგრამაში დიდი რაოდენობით (100 და მეტი) კლასების არსებობით.

თუ გავავლებთ პარალელს რელაციურ მონაცემთა ბაზებთან, მაშინ კლასი (ობიექტი) შეესაბამება ცხრილს, ხოლო თვისება ველს (სვეტს).

თვისება შეიძლება იყოს მუდმივა, გადაგზავნა ობიექტზე, ჩაშენებული ობიექტები, მასივი (Array) და სია (List), მრავალგანზომილებიანი თვისება. მას მიეკუთვნება აგრეთვე ობიექტებს შორის კავშირები. თვისებას აქვს ტიპი. ძირითადი ტიპებია: String, Numeric, Float, Currency, Date, List, Status.

ქვემოთ მოცემულია ოომბ–ის ძირითადი პრინციპები, რომლებიც ეყრდნობა ობიექტურ–ორიენტირებულ მიდგომას:

1) სვეტის როლში შეიძლება გამოყენებულ იქნას ნებისმიერი კორტეჟი. სხვანაირად რომ ვთქვათ, სვეტი შეიძლება იყოს სხვა ცხრილი.
2) მონაცემთა მანიპულაციის პროცედურები საშუალებას იძლევა გაერთიანდეს რამოდენიმე პროცედურა;
3) სვეტების მონაცემები შეიძლება ექვემდებარებოდეს მემკვიდრეობითობის პრინციპს;
4) ელემენტებს შორის დამოკიდებულებაში შესაძლებელი იქნება არა მარტო ცალკეული ელემენტის მონაწილეობა (როგორც რელაციურ მონაცემთა ბაზებში), არამედ მრავალისაც.

ოომბ-ს მიეკუთვნება მხოლოდ ის მონაცემთა ბაზები, რომელთა რეალიზაციის ყველა სტრუქტურული ელემენტი აგებულია ობიექტურ–ორიენტირებული მიდგომით. თუ ერთი მაინც სტრუქტურული ელემენტი არ გამოიყენებს ობიექტურ–ორიენტირებულ მიდგომას, მაშინ საქმე გვაქვს ობიექტურ–რელაციურ მონაცემთა ბაზებთან [1-4].

თანამედროვე პირობებში ობიექტურ-ორიენტირებული მონაცემთა ბაზების ტექნოლოგიის განვითარებაში ერთმანეთს კონკურენციას უწევს ორი ძირითადი მიმართულება: 1). Distributed Object Linking and Embedding (OLE), 2). Common Object Request Broker Architecture (CORBA). ისინი არის ობიექტზე ორიენტირებული დისტრიბუციული ტექნოლოგიები, რომლებიც ხშირად გამოიყენება ობიექტურ-ორიენტირებული მონაცემთა ბაზების ინტეგრაციისთვის. ქვემოთ ჩამოყალიბებულია ამ ტექნოლოგიების ძირითადი არსი:

1).Distributed Object Linking and Embedding (OLE)-ტექნოლოგია, რომელიც შემუშავებულია კომპანია *Microsoft*-ის მიერ და ძირითადად ორიენტირებულია Windows პლატფორმებზე. იგი უზრუნველყოფს ობიექტების გაზიარებას და ურთიერთქმედებას სხვადასხვა პროგრამას შორის.

OLE-ს მეშვეობით სხვადასხვა პროგრამას (მაგალითად, MS Word-სა და Excel-ს) შეუძლია ერთმანეთთან **ობიექტების გაზიარება** ისე, რომ ისინი მუშაობდნენ **სინქრონულად.**

მაგალითად, Excel-ში შექმნილი ცხრილი ჩაშენდეს Word დოკუმენტში ისე, რომ საჭიროების შემთხვევაში მოხდეს პირდაპირ Word-დან განახლება.

2).Common Object Request Broker Architecture (CORBA)-პლატფორმებისაგან დამოუკიდებელი სტანდარტი, რომელიც შემუშავებულია *Object Management Group (OMG)*-ის მიერ და მხარდაჭერილია ისეთი კომპანიების მიერ, როგორებიცაა *IBM, Novell, DEC* და სხვები. CORBA უზრუნველყოფს სხვადასხვა სისტემებსა და პროგრამულ გარემოში არსებული ობიექტების ურთიერთქმედებას ქსელის საშუალებით.

მაგალითად, ერთი პროგრამა შეიძლება დაწერილი იყოს Java-ზე Linux-ში, მეორე - C++-ზე Windows-ში. CORBA იძლევა მათ შორის კავშირისა და მონაცემთა გაცვლის შესაძლებლობას.

ამგვარად, ობიექტურ-ორიენტირებულ მონაცემთა ბაზებში მონაცემები ინახება ობიექტების სახით და არა ცხრილებად. მონაცემებთან წვდომა და მოთხოვნების შექმნა ხდება პროგრამული კოდით, რომელიც დაწერილია ობიექტურ-ორენტირებულ ენაზე (მაგ. Java, C++, Python). ზოგიერთ თანამედროვე ობიექტურ-ორენტირებულ მონაცემთა ბაზას აქვს SQL ენის გაფართოებული ვერსია.

თანამედროვე პირობებში ფუნქციონირებს 300-ზე მეტი სახის ოომბმს. ზოგიერთი მათგანი მოცემულია ცხრ.1–ში. ცხრილიდან ჩანს, რომ მათი შექმნისათვის გამოყენებულია დამუშავების სხვადასხვა მიდგომა:

ცხრილი 1

რამოდენიმე ოომბმს–ის დახასიათება

მწარმოებელი ფირმა	მართვის სისტემა (OODBMS)	დამუშავების ენა/ენები	დამუშავების მიდგომა	შენიშვნა / განსაკუთრებული მახასიათებლები
ObjectDB Software Ltd.	ObjectDB	Java, JPA	ობიექტების პირდაპირი შენახვა, ORM მხარდაჭერა	მაღალსიჩქარიანი JPA რეალიზაცია, მხარდაჭერა Hibernate-ის ალტერნატივად
Actian (Versant Corp.)	Versant Object Database	Java, C++	ობიექტების პირდაპირი რუკირება	გამოიყენება ტელეკომუნიკაციებში, რეალურ დროში დამუშავება
db4objects Inc.	db4o	Java, .NET (C#)	ობიექტების პირდაპირი შენახვა, არ იყენებს ORM-ს	გამარტივებული დანერგვა და მინიმალური კონფიგურაცია
GemTalk Systems	GemStone/S	Smalltalk	ობიექტების პირდაპირი შენახვა,	სრული ინტეგრაცია Smalltalk გარემოსთან

მწარმოებელი ფირმა	მართვის სისტემა (OODBMS)	დამუშავების ენა/ენები	დამუშავების მიდგომა	შენიშვნა / განსაკუთრებული მახასიათებლები
			გამართულობა ობიექტურ გარემოში	
Zope Corporation	ZODB	Python	ობიექტების პირდაპირი შენახვა	გამოიყენება Web აპლიკაციებში (Zope, Plone)
RunTime, Inc.	ODABA	C++	ობიექტური მოდელი + სუსტი რელაციური მხარდაჭერა	ძლიერი ტიპიზაცია, ინტეგრირებული GUI
Oracle Corporation	Oracle Berkeley DB (Java Edition)	Java	ჩაშენებული ობიექტური შენახვა Java-ში	გამოიყენება ჩაშენებულ აპლიკაციებში, მაღალი შესრულება
Franz Inc.	AllegroGraph	Java, Lisp, Python, C#	გრაფებზე დაფუძნებული ობიექტური მოდელი + RDF მონაცემების მხარდაჭერა	გამოიყენება knowledge graph-ებში, AI/ML სისტემებში

Object-Relational Mapping (**ORM-ობიექტურ-რელაციური რუკირება)** არის ტექნოლოგია, რომელიც გამოიყენება **ობიექტურ-ორიენტირებულ პროგრამებსა** და **რელაციურ მონაცემთა ბაზებს** შორის მონაცემების გადასაყვანად. ORM ავტომატურად გარდაქმნის ობიექტებს მონაცემთა ბაზის ცხრილებად და პირიქით - ცხრილების მონაცემებს პროგრამულ ობიექტებად SQL-ენის გამოყენების გარეშე.

ოომბმს–ის არჩევას განსაზღვრავს: ოოპე და SQL–ენის გაფართოება; ადმინისტრირების და დამუშავების საშუალებანი; მონაცემებზე წვდომა **(ODBC-**Open Database Connectivity-ღია კავშირი მონაცემთა ბაზებთან); სხვადასხვა პლატფორმით მუშაობის შესაძლებლობა.

ერთ-ერთი პოპულარული ოომბმს არის **Versant Object Database.** იგი იყენებს ენას C++ და SQL-ენის გაფართოებას **VQL-ენას**. **VQL-ენა** არის SQL-ენის მსგავსი, მაგრამ მორგებულია ობიექტებზე მანიპულირებისათვის [5].

1.1.1. *ქვემოთ მოცემულია პროგრამების ფრაგმენტები ობმბმს-თვის - Versant Object Database. (მონაცემთა ბაზაში შექმნილია ორი კლასი: 1.კლასი Person, 2.კლასი Department).*

1) პროგრამამ უნდა განაახლოს **ერთი თანამშრომლის ასაკი და წაშალოს მეორე თანამშრომლის მონაცემი.**

```
//გავაახლოთ Saba-ს ასაკი
const char* updateQuery="select from Person where name=' Saba '";
VOD_Query* q1= new VOD_Query(db, updateQuery);
Person* s;
if ((s=(Person*)q1->next()) != NULL) {
std::cout<<"მოიძებნა Saba. ასაკი: "<< s->age <<" -> ";
s->age = 40;
std::cout<<s->age<<" განახლდა"<<endl;
} else {
std::cout<<" ვერ მოიძებნა Saba."<<endl;
}
delete q1;

//წავშალოთ Akaki
const char* deleteQuery="select from Person where name = 'Akaki '";
VOD_Query* q2=new VOD_Query(db, deleteQuery);
Person* a;
if ((a = (Person*)q2->next())!=NULL) {
std::cout<<"მოიძებნა Akaki. იშლება."<<endl;
db->remove(a);
} else {
std::cout<<" ვერ მოიძებნა Akaki."<<endl;
}
```

1.1.2. *2) პროგრამამ უნდა **გამოთვალოს თანამშრომლების საშუალო ასაკი თითოეული განყოფილებისათვის.***

```
const char* vql = "select from Person";
VOD_Query* query = new VOD_Query(db, vql);
std::map<VOD_String, AgeAccumulator>ageByDept;
Person* p;
while ((p=(Person*)query->next()) !=NULL) {
if (p->department) {
VOD_String deptName = p->department->name;
ageByDept[deptName].totalAge +=p->age;
ageByDept[deptName].count +=1;
}
```

```
}
std::cout << "საშუალო ასაკი განყოფილებების მიხედვით:"<<endl;
for (const auto& entry : ageByDept) {
double avg=(double)entry.second.totalAge /entry.second.count;
std::cout<<"განყოფილება: "<<entry.first<<", საშუალო ასაკი: "<< avg<<std::endl;
}
```

1.1.3.

რელაციურ მონაცემთა ბაზებთან შედარებით ოომბ–ს აქვს შემდეგი უპირატესობები:

1) სისტემის მოდელირების უკეთესი შესაძლებლობები;
2) სტრუქტურის ადვილად გაფართოების უნარი, რაც განპირობებულია ახალი ტიპის მონაცემების (თვისებების) შექმნით, მემკვიდრეობით, ახალი კავშირების დამყარებით და მეთოდების კორექტირების საშუალებებით;
3) რეკურსული მეთოდების გამოყენების შესაძლებლობა დიდი მოცულობის მონაცემებზე ნავიგაციური მეთოდით მიმართვისას;
4) მწარმოებლობის ამაღლება 3–20–ჯერ და მეტად;
5) გამოყენების ფართო სპექტრი (მაგალითად, გამოიყენება მულტიმედიურ სისტემებში, გრაფებში).

აღსანიშნავია, რომ ოომბ რელაციურ მონაცემთა ბაზებთან შედარებით საჭიროებს 4–5–ჯერ მეტ მეხსიერებას.

ნაშრომში განხილულია ოომბ და ოომბმს-ის სახეები. ჩამოყალიბებულია ობიექტურ–ორიენტირებული პროექტირებისა და ობიექტურ–ორიენტირებული პროგრამირების არსი. გაკეთებულია რელაციურ მბ-ისა და ოომბ-ის შედარებითი ანალიზი.

ლიტერატურა:

1. Советов Б., Цахановский В., Чертовской В. Базы дных. 2013; сс. 168-166.
2. McFarland G., Rudmik A., Lange D. Object-Oriented Database Management Systems Revisited. Modus Operandi, Inc.122 Fourth Ave., Indialantic, FL 32903. 1997; pp. 4-43.
3. https://www.owox.com/blog/articles/object-oriented-data-models#build-scalable-data-models-with-owox-bi
4. https://www.mongodb.com/resources/basics/databases/what-is-an-object-oriented-database?
5. Versant Software LLC and Actian Corporation. *Versant C++ Programmer's Guide: Release 9*, Versant Server 9.3. 2020; pp. 325-346.

About Object-Oriented Databases

Tsatsa Namchevadze[1], Elza Bichadze[2]
[1]Akaki Tsereteli State University, Associate Professor, tsatsanamg@gmail.com
[2]Akaki Tsereteli State University, Assistant Professor, elza.bitsadze@atsu.edu.ge

Summary

The article discusses the importance of using databases and highlights the shortcomings of relational databases when processing large volumes of data. It is noted that one way to overcome these limitations is through the use of object-oriented databases. The essence of such databases is defined by the object-oriented approach. This approach involves object-oriented design and object-oriented programming. The article outlines the key concepts of object-oriented design and programming.

To construct databases using the object-oriented approach, the following are essential:

1. Data encapsulation, i.e., the definition of classes and objects;
2. Specification of the structure of tables;
3. Creation of class inheritance for data;
4. Ensuring polymorphism, among other aspects.

It is noted that only those databases in which all structural elements are implemented using the object-oriented approach are considered object-oriented databases. If even one structural element does not use the object-oriented approach, the database is classified as an object-relational database.

The paper presents the advantages of object-oriented databases (OODB) compared to relational databases. These include:

1. Better system modeling capabilities;
2. Easier structural extensibility enabled by the creation of new data types (attributes), inheritance, establishing new relationships, and method modification capabilities;
3. The ability to use recursive methods when accessing large datasets via navigational methods;
4. 3 to 20 times or more performance improvement;
5. A wide range of applications (e.g., in multimedia systems, graphics), among others.

It is worth noting that object-oriented databases require 4 to 5 times more memory compared to relational databases.

In object-oriented databases, data access and query execution are performed using program code written in an object-oriented programming language (e.g., Java, C++, Python). Some modern object-oriented databases provide an extended version of the SQL language.

The article also describes several widely used object-oriented database management systems available today. Code fragments are provided for one of the most popular systems-Versant Object Database.

Keywords: object-oriented databases, object-oriented database management system, object data model, object definition language, object query language.