UDC: 004.42.045 Info M str. 15-21

MERENJE I ANALIZA PERFORMANSI ENTITY CORE OKVIRA I NHIBERNATE OKVIRA
MEASURING AND ANALYZING PERFORMANCE OF ENTYTY CORE AND NHIBERNATE FRAMEWORKS

Tijana Milošević, Saša D. Lazarević

REZIME: Danas postoji dosta alata za objektno-relaciono preslikavanje i s obzirom na tu činjenicu, treba doneti odluku koji alat odabrati. Upravo je ovaj problem svrha pisanja ovog rada. Jasno je da su Entity i Entity Core okviri najpopularniji alati za objektno-relaciono preslikavanje kada je reč o C# programskom jeziku. Međutim, postoji još dobrih alata koji mogu da im pariraju. Rad analizira i poredi podršku za objektno-relaciono preslikavanje i performanse Entity Core okvira koji je danas veoma aktuelan kod .NET aplikacija, i NHibernate koji je .NET verzija popularnog ORM alata Java programskog jezika, Hibernate. Kao metrike u ovom radu odabrane su vreme i alocirana memorija. Za merenje vremena i memorije korišćena je BenchmarkDotNet biblioteka. Merene su performanse učitavanja, upisivanja, izmene i brisanja rekorda iz baze podataka. Cilj rada je uporedna analiza objektno-relacionog preslikavanja Entity Core i NHibernate okvira, ispitivanje performansi Entity Core i NHibernate okvira.
KLJUČNE REČI: Entity Core, NHibernate, objektno-relaciono preslikavanje, .NET, performanse, metrike
ABSTRACT: Today, there are many tools for object-relational mapping, and given that fact, you need to decide which tool to choose. This problem is the purpose of writing this paper. Clearly, the Entity and Entity Core frameworks are the most popular object-relational mapping tools in C# programming language. However, there are other good tools that can match them. The paper analyzes and compares the support for object-relational mapping and performance of the Entity Core framework, which is very relevant today in .NET applications, and NHibernate, which is a .NET version of the popular ORM tool of the Java programming language, Hibernate. Time and allocated memory are selected as metrics in this paper. The BenchmarkDotNet library was used to measure time and memory. The performance of loading, writing, modifying and deleting records from the database were measured. The aim of this paper is a comparative analysis of object-relational mapping of Entity Core and NHibernate framework, examination of Entity Core and NHibernate framework performance.
KEY WORDS: Entity Core, NHibernate, object-relational mapping, .NET, performance, metrics

1. UVOD

Tehnologija baza podataka je u periodu intezivnih promena i inovacija koje su revolucionarne i evolucijske. Revolucionarni aspekt se odnosi na inovaciju objektno-orijentisane tehnologije programiranja. Evolucijski aspekt se odnosi na promociju nove, proširene verzije tehnologije relacionih baza podataka koji se nazivaju objektno-relacioni sistemi za upravljanje bazama podataka. [36] Objektno-relacioni sistemi za upravljanje bazama podataka uključuju proširenje sistema relacionih baza podataka u vidu objektno orijentisanih funkcija ili direktnog predstavljanja aplikativnih objekata u relcionim bazama podataka. [36]

Relacione baze podataka i objektno orijentisani programski jezici su zasnovani na različitim paradigmama i postoje nepoklapanja između njih. Ova nepoklapanja se odnose na neusklađenost impedanse u odnosu na objekat (engl. *impedance mismatch*). [36] Okviri za objektno-relaciono preslikavanje bi trebalo da usaglase objektni i relacioni model i obezbede preslikavanje iz jednog modela u drugi. Neki od ORM alata su Entity Core, NHibernate, Hibernate, Dapper, Django i mnogi drugi.

2. OBJEKTNO-RELACIONO PRESLIKAVANJE

Objektno-relaciono preslikavanje (ORM) je tehnika programiranja za preslikavanje podataka između nekompatibilnih sistema u objektno orijentisanim programskim jezicima. Drugim rečima, to je koncept preslikavanja poslovnih objekata aplikacije u relacione tabele baze podataka, tako da se podacima može lako pristupiti i u potpunosti ih ažurirati kroz objektni model aplikacije. [2]

Neki od najvažnijih benefita korišćenja ORM alata: [21]

- automatizuje konverziju objekat-tabela i tabela-objekat, što olakšava razvoj, i time omogućava brži izlazak na tržište i manje troškove razvoja i održavanja;
- ne zahteva puno koda, nema potrebe za ručnim pisanjem SQL funkcija i procedura ka bazi podataka.

Glavni aspekti dobrog ORM alata: [21]

- Kao osnovni aspekt je da obezbedi preslikavanje entiteta iz objektnog modela u relacioni model, međutim dobar ORM alat pruža podršku za preslikavanje svih tipova veza (obične, jak-slab objekat, agregacije, specijalizacije, direktne i indirektne zavisnosti)
- Mogućnost keširanja objekata, što dopirnosi boljim performansama alata. Na primer, Entity Core okvir ima mogućnost da kešira podatke, ali je potrebno kroz kod to i napomenuti.
- Podrška za više platformi baze podataka.
- Dinamički upiti nad bazom podataka.
- Mogućnost korišćenja lazy loading-a. Ovaj patern omogućava optimizaciju memorijskog prostora time što daje prioritet komponentama koje treba da budu učitane.
- Nenametljiva istrajnost, što podrazumeva da nema potrebe za nasleđivanjem ili proširivanjem funkcije, klase, interfejsa, ili bilo šta što je specifično za provajdera.
- Mogućnost generisanja koda.
- Podrška za više programskih jezika.

- Podrška za uskladištene procedure. Mogućnost pisanja uskladištenih procedura se najbolje ogleda u performansama upita nad bazom podataka, na primer ukoliko je potrebno prikazati neki izveštaj.
- Ostali aspekti u koje spadaju cena, dokumentacija, jednostavnost upotrebe, performanse, podrška.

Kada se govori o objektno-relacionom preslikavanju, treba spomenuti i pojam perzistentnosti (postojanosti) objekta. Za neki proces ili objekat se kaže da je perzistentan ako nastavi da postoji i onda kada je njegov roditelj/glavni proces prestao da postoji. [29] Relacioni sistemi baze podataka čuvaju podatke u vidu rekorda i tabela, gde se može reći da su podaci perzistentni, tj. postojani. To bi značilo da ukoliko nas zanimaju informacije o nekom objektu, morali bi smo da svaki put pozivamo bazu podataka. Stalno pozivanje baze podataka je skupa operacija, koja oduzima dosta vremena i mermorijskog prostora. Zbog toga se javila potreba za mehanizmom koji će jednom učitan rekord iz baze u vidu objekta čuvati i pratiti kroz njegova stanja. Upravo su ORM alati ti koji ovo omogućavaju.

Između objektnog i relacionog modela postoji neusklađenost (engl. paradigm mismatch). Prilikom upisivanja i učitavanja podataka iz baze nailazi se na nekoliko problema prilikom njihovog predstavljanja u objektnom/relacionom modelu: [4]

- nasleđivanje,
- identitet (engl. identity), gde kod sistema za upravljanje bazama podataka imamo identitet po primarnom ključu, dok kod objektnog modela se može koristiti identitet objekta (a==b) ili jednakost korišćenjem Equals metode,
- veze između entiteta, gde se u relacionom modelu veze predstavljaju spoljnim ključevima, i ta veza je jednosmerna, dok se u objektnom modelu veze predstavljaju dvosmerno, kod prvog i kod drugog objekta koji čine vezu
- kretanje kroz podatke, u objektnom modelu se od prvog objekta do informacija o drugom objektu dolazi preko veze između njih, dok se u relacionom modelu preko spajanja tabela dolazi do informacija.

ORM alat bi trebalo da usaglasi objektni i relacioni model i obezbedi preslikavanje iz jednog modela u drugi.

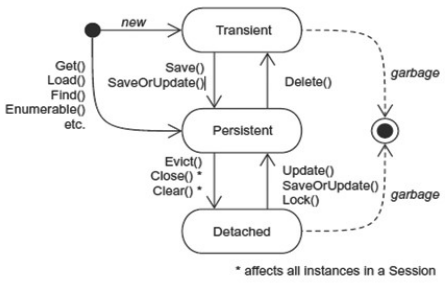
3. ENTITY CORE OKVIR

Entity Core je okvir za objektno-relaciono preslikavanje. Napisan je u C# programskom jeziku, i za razliku od Entity okvira, koji radi na .NET Framework-u, Entity Core je napravljen da radi na .NET Core-u. Prva verzija je realizovana u avgustu 2008. godine. Entity Core okvir je platformski nezavistan, tj, može se koristiti na različitim platformama kao što su Windows, Linux, macOS, iOS i Android. Sistemi za upravljanje bazama podataka koje su podržane od strane Entity Core okvira su Microsoft SQL Server, SQLite, MySQL, DB2, PostgreSQL i drugi. [19] Komunikacija sa odabranim sistemom za upravljanje bazama podataka se ostvaruje preko DbContext-a. Entity Core okvir razlikuje 5 različitih stanja entiteta: dodat, obrisan, izmenjen, nepromenjen i nezavisan.

4. NHIBERNATE OKVIR

NHibernate je alat za objektno-relaciono preslikavanje. Na zvaničnom sajtu ovog okvira stoji da je on alat otvorenog koda za objektno- relaciono preslikavanje za .NET okvir, koji se redovno razvija i koristi u hiljadama uspešnih projekata. NHibernate je .NET verzija veoma popularnog alata za objektno-relaciono preslikavanje Java programskog jezika, Hibernate. Ukoliko pogledamo izvorni kod (https://github.com/nhibernate/nhibernate-core), može se videti da je ovaj okvir napisan u C# programskom jeziku. Prva realizacija ovog okvira bila je 2007. godine, a trenutno najnovija verzija je 5.3.8. (u trenutku pisanja ovog rada, jul 2021. godine). NHibernate ima podršku za mnoge baze podataka poput SQL Server-a, Oracle, DB2, MySQL, SQLite. NHibernate komunicira sa bazom podataka preko sesija. Za rad sa jednom bazom podataka konfiguriše se jedan Session Factory. Ukoliko bi u sistemu postojale dve ili više baze podataka, za svaku treba da postoji Session Factory. Za svaki zahtev korisnika Session Factory kreira sesiju.

Kada je reč o postojanosti (perzistentnosti) entiteta, NHibernate okvir opisuje tri stanja entiteta: tranzientno, perzistentno i nezavisno stanje. Prilikom kreiranja entieta on postaje tranzientan, i baza podataka ga još uvek nije svesna (on ne postoji u tabeli). Nakon poziva metoda Save ili SaveOrUpdate, nov entitet se dodaje u tabelu i postaje perzistentan. Dalje se on može brisati ili menjati. Prilikom brisanja entiteta, on prelazi u tranzientno stanje. Ukoliko se tokom jedne sesije radi sa nekim entitetom, nije moguće u toku iste sesije raditi sa nekim drugim entietom. Prvo je potrebno da se tekući entetet dovede u nezavisno stanje metodama Evict, Close ili Clear. Učitavanjem entita iz baze, on je u perzistentnom stanju. Dijagram prelaza stanja entiteta dat je na slici 1.



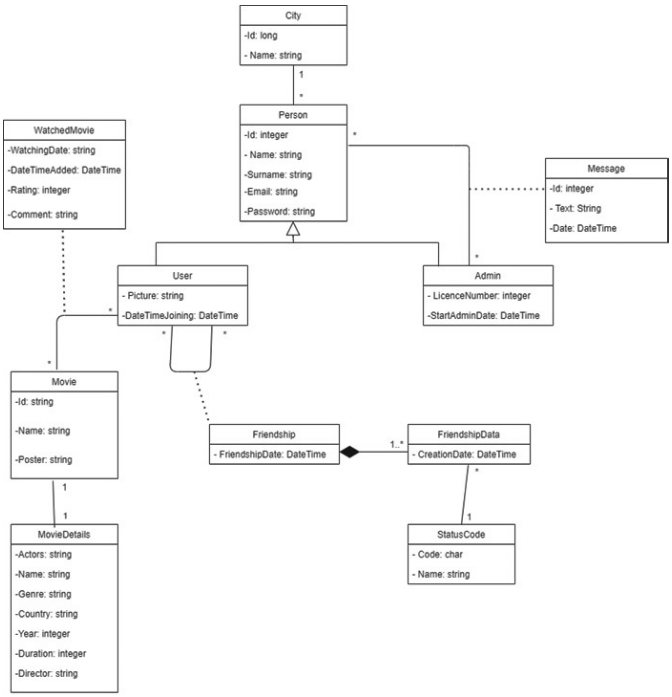
Slika 1. NHibernate dijagram prelaza stanja

5. STUDIJSKI PRIMER

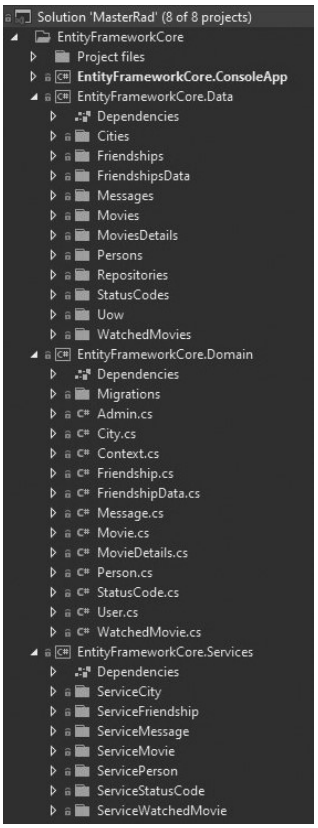
Entity Core i NHibernate su korišćeni u studijskom primeru kao alati za preslikavanje objektnog modela u relacioni i obrnuto. Domenskim modelom studijskog primera objašnjen je sistem za praćenje odgledanih filmova. Model je osmišljen tako da pokrije sve tipove veza- obične, jak/slab objekat, agregacije, specijalizacije, direktne i indirektne zavisnosti. Konceptualni dijagram klasa domenskog modela dat je na slici 2.

Implementacija studijskog primera rađena je u C# programskom jeziku, u Visual Studio 2019 razvojnom okruženju, dok je kao sistem za upravljanje bazama podataka korišćen

SQL Server. Projekat je podeljen u dva dela- prvi koji se tiče Entity Core okvira, a drugi NHibernate okvira. Oba dela projekta su isto struktuirana. Arhitekturu čine tri sloja- domenski, sloj podataka i servisni sloj i na kraju konzolna aplikacija iz koje se pokreću metode servisa. Struktura projekta prikazana je na slici 3.



Slika 2. Konceptualni dijagram klasa domenskog modela



Slika 3. Struktura projekta

6. MERENJE PERFORMANSI ORM ALATA

U poglavlju 2 pobrojani su aspekti koje dobar ORM alat treba da poseduje. Jedan od tih aspekata su i performanse alata. Performanse predstavljaju meru koliko dobro osoba, mašina, sistem, itd. dobro obavljaju neku aktivnost, operaciju. To može biti vreme za koje se obavlja ta operacija, koliko resursa (novca, ljudstva, memorije...) je potrošeno, snaga motora za vozila itd. S obzirom da je alat za objektno-relaciono preslikavanje direktno vezan za bazu podataka, jasno je da je glavni pokazatelj performansi vreme upisivanja i učitavanja podataka. Međutim on nije i jedini pokazatelj. Memorija koja je potrebna za učitavanje/upisivanje podataka u bazu takođe je bitan pokazatelj performansi nekog ORM alata.

Metrike koje su odabrane u ovom radu, kao pokazatelji performansi ORM alata su vreme izvršenja operacije i alocirana memorija prilikom izvršenja operacije. Jasno je da je vreme odgovora kako neke aplikacije tako i kod ORM alata bitno iz ugla korisnika i da se očekuje odgovor što pre. Iz tog razloga je vreme uzeto u razmatranje kada je reč o merenju performansi ORM alata. Pored vremena izvršenja kao metrika je odabrana i alocirana memorija jer se može desiti da alat bude brz, ali da za izvršenje svojih operacija troši mnogo memorije. Memorija je skup resurs. Zbog toga se u ovom radu performanse ORM alata sagledaju i sa aspekta memorije.

7. PERFORMANSE ENTITY CORE OKVIRA I NHIBERNATE OKVIRA

U nastavku je opisan postupak merenja performansi Entity Core i NHibernate okvira, kao i rezultati merenja i njihova analiza.

7.1 *Merenje*

Za merenje performansi korišćena je biblioteka- BenchmarkDotNet verzije 0.12.1. Merene su performanse (brzina izvršenja operacije i alocirana memorija) Entity Core i NHibernate okvira, i to performanse dodavanja, brisanja i izmene rekorda tabela svih tipova veza oba okvira, prvo za jedan rekord u bazi, a zatim i za 100, 1000 i 10000 rekorda. Svaka metoda je testirana u 10 iteracija, iz razloga što je primećena oscilacija u vremenu izvršenja metoda. Prvi poziv metode je uvek najsporiji, dok je svaki sledeći brži (međutim, pad brzine izvršenja metode nije linearan). Takođe, ispitane su performanse učitavanja podataka nad tabelom City ukoliko ona ima 2000, 25000 i 200000 rekorda u dve iteracije. Performanse su merene na računaru koji ima operativni sistem Windows 10, karakteristika Intel Core i7-6500U CPU 2.50GHz, 8GB RAM.

7.2 *Rezultati i analiza*

U tabeli 1 prikazana su prosečna vremena za unos jednog novog rekorda u tabelu za svaki tip veze oba okvira, dok je u tabeli 2 alocirana memorija. Sa slika 4 i 5 se može videti da ubedljivo bolje performanse ima NHibernate, za svaki

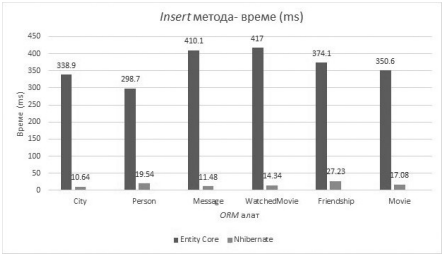
tip veze. Kod Entity Core-a, dodavanje odgledanog filma se najduže izvršavalo (417 ms), kod NHibernate-a dodavanje novog prijateljstva (27.23 ms), dok je memorijski najzahtevnije dodavanje novog prijateljstva kod oba okvira (141.04 KB EF Core i 40.31 KB NHibernate), što je i očekivano, jer se prilikom dodavanja prijateljstva dodaju i detalji o prijateljstvu.

Tabela 1. Prosečno vreme dodavanja novog rekorda u bazu

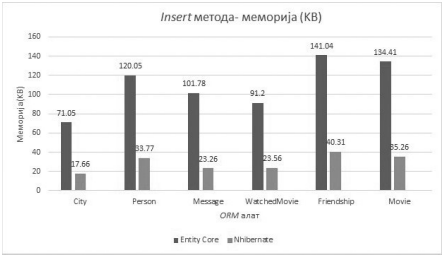
Entitet (veza)/vremena	Insert metoda- vreme (ms)	
	Entity Core	NHibernate
City	338.9	10.64
Person	298.7	19.54
Message	410.1	11.48
WatchedMovie	471	14.34
Friendship	374.1	27.23
Movie	350.6	17.08

Tabela 2. Alocirana memorija dodavanja novog rekorda u bazu

Entitet (veza)/memorija	Insert metoda- memorija (KB)	
	Entity Core	NHibernate
City	71.05	17.66
Person	120.05	33.77
Message	101.78	23.26
WatchedMovie	91.2	23.56
Friendship	141.04	40.31
Movie	134.41	35.26



Slika 4. Prosečno vreme dodavanja novog rekorda u bazu



Slika 5. Alocirana memorija dodavanja novog rekorda u bazu

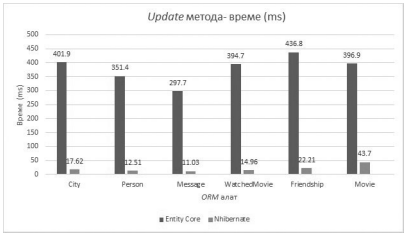
U tabelama 3 i 4 prikazane su performanse (vreme i alocirana memorija) izmene jednog rekorda za svaki od tabela (tipova veza) za Entity Core i NHibernate okvir. Prosečno vreme je dato za 10 iteracija. Sa slika 6 i 7 se može videti da ubedljivo bolje performanse ima NHibernate, za svaki tip veze. Kod EF Core-a, izmena prijateljstva se najduže izvršavalo (436.8 ms), kod NHibernate-a izmena filma (43.7 ms), dok je memorijski najzahtevnije izmena prijateljstva kod oba okvira (274.98 KB EF Core i 68.61 KB NHibernate), što je i očekivano, jer se prilikom izmena prijateljstva menjaju i detalji o prijateljstvu.

Tabela 3. Prosečno vreme izmene jednog rekorda u bazi

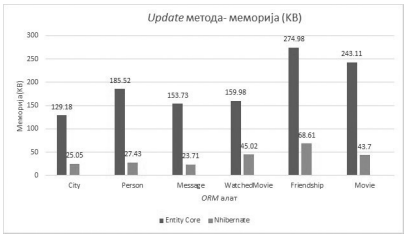
Entitet (veza)/vremena	Update metoda- vreme (ms)	
	Entity Core	NHibernate
City	404.9	17.62
Person	351.4	12.51
Message	297.7	11.03
WatchedMovie	394.7	14.96
Friendship	436.8	22.21
Movie	396.9	43.7

Tabela 4. Alocirana memorija izmene jednog rekorda u bazi

Entitet (veza)/memorija	Update metoda- memorija (KB)	
	Entity Core	NHibernate
City	129.18	25.05
Person	185.52	27.43
Message	153.73	23.71
WatchedMovie	159.98	41.02
Friendship	274.98	68.61
Movie	243.11	43.7



Slika 6. Prosečno vreme izmene jednog rekorda u bazi



Slika 7. Alocirana memorija izmene jednog rekorda u bazi

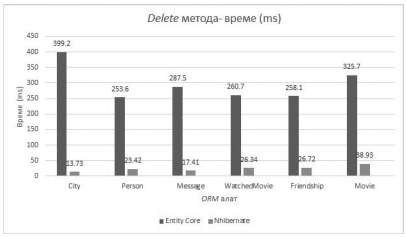
U tabelama 5 i 6 prikazane su performanse (vreme i alocirana memorija) brisanja jednog rekorda za svaki od tabela (tipova veza) za Entity Core i NHibernate okvir. Prosečno vreme je dato za 10 iteracija. Sa slika 8 i 9 se može videti da ubedljivo bolje performanse ima NHibernate, za svaki tip veze. Kod EF Core-a, brisanje grada se najduže izvršavalo (399.2 ms), kod NHibernate-a brisanje filma (38.93 ms), dok je memorijski najzahtevnije brisanje osobe kod EF Core-a (80.91 KB), a kod NHibernate-a brisanje filma što je i očekivano, jer se prilikom brisanja filma brišu i njegovi detalji.

Tabela 5. Prosečno vreme brisanja jednog rekorda iz baze

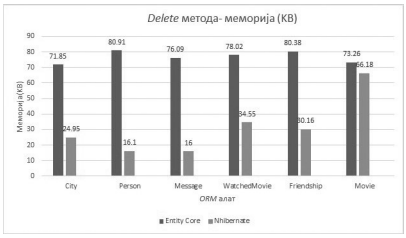
Entitet (veza)/vremena	Delete metoda- vreme (ms)	
	Entity Core	NHibernate
City	399.2	13.73
Person	253.6	23.42
Message	287.5	17.41
WatchedMovie	260.7	26.34
Friendship	258.1	26.72
Movie	325.7	38.93

Tabela 6. Alocirana memorija brisanja jednog rekorda iz baze

Entitet (veza)/memorija	Delete metoda- memorija (KB)	
	Entity Core	NHibernate
City	71.85	24.95
Person	80.91	16.1
Message	76.09	16
WatchedMovie	78.02	34.55
Friendship	80.38	30.16
Movie	73.26	66.18



Slika 8. Prosečno vreme brisanja jednog rekorda iz baze



Slika 9. Alocirana memorija brisanja jednog rekorda iz baze

U tabelama 7 i 8 prikazane su performanse učitavanja rekorda iz baze podataka, ukoliko ona ima 2000, 25000 ili 200000 rekorda. Vremena prikazana u tabeli 7 su prosečna vremena za dve iteracije. Formirani su različiti upiti kako bi se ispitale njihove performanse. Testirani su sledeći upiti: običan select bez uslova, select sa uslovom, select sa spajanjem sa tabelama i uslovom. Po pitanju brzine vraćanja odgovora, bolji je NHibernate. Pri vraćanju svih rekorda iz tabele ukoliko ona ima 2000 rekorda, NHibernate je brži za 1,6 sekundi u odnosu na Entity Core, za 2 sekunde za 25 000 rekorda i za 13 sekundi za 200 000 rekorda. Kod ostalih select upita brži je za oko sekundu.

Kada je reč o memoriji koja se alocira prilikom izvršavanja upita, pri vraćanju svih rekorda iz tabele ukoliko ona ima 2000 rekorda, NHibernate alocira 1 MB manje u odnosu na Entity Core, 13,8 MB manje za 25 000 rekorda i oko 50 MB manje za 200 000 rekorda. Pri pozivanju upita pri kome se do podataka dolazi spajanjem tabela, Entity Core alocira manje memorije nego NHibernate. Međutim ta razlika je minimalna, oko 0,05 MB.

Tabela 7. Prosečno vreme učitavanja rekorda iz baze

Select upit/vremena	Select metoda- vreme (s)					
	Entity Core			NHibernate		
	2000	25000	200000	2000	25000	200000
Učitavanje svih mesta	1.757	2.326	16.884	0.1034	0.332	3.7539
Učitavanje mesta po Id-ju	1.414	1.584	1.199	0.0667	0.0918	0.0791
Učitavanje mesta po imenu osobe	1.493	1.505	1.489	0.1574	0.1932	0.1549
Učitavanje mesta po oceni odgledanog filma	1.566	1.659	1.377	0.1871	0.1944	0.3578
Učitavanje mesta po žanru odgledanog filma	1.449	1.547	1.373	0.1844	0.1536	0.2156

Tabela 8. Alocirana memorija učitavanja rekorda iz baze

Select upit/memorija	Select metoda- memorija (MB)					
	Entity Core			NHibernate		
	2000	25000	200000	2000	25000	200000
Učitavanje svih mesta	1.5154	17.7855	80.854	0.4181	3.9695	30.0397
Učitavanje mesta po Id-ju	0.0395	0.0006	0.0006	0.0165	0.0007	0.0004
Učitavanje mesta po imenu osobe	0.0387	0.03926	0.0392	0.0856	0.0856	0.0856
Učitavanje mesta po oceni odgledanog filma	0.0507	0.0495	0.0501	0.1553	0.1144	0.1143
Učitavanje mesta po žanru odgledanog filma	0.0577	0.0578	0.0577	0.1663	0.1619	0.1626

ZAKLJUČAK

Dobar alat za objektno-relaciono preslikavanje bi trebalo da obezbedi preslikavanje iz objektnog modela u relacioni, i obrnuto, uz podršku za sve tipove veza. Ovaj aspekt dobrog ORM alata poseduju i Entity Core i NHibernate okviri, s obzirom da poseduju mehanizme koji omogućavaju programeru da svoj domenski model uspešno prevede u relacioni, i obrnuto. Oba okvira imaju podršku za preslikavanje svih tipova veza (obična, veza agregacije, jak-slab objekat, specijalizacija, indirektna i direktna veza). Entity Core je tek od poslednje verzije uveo podršku za vezu više-više, što je do tog trenutka bio velika mana ovog okvira. U tabeli 9 prikazani su neki aspekti dobrog ORM alata i da li Entity Core i NHibernate poseduju te aspekte.

Tabela 9. Aspekti dobrog ORM alata

Aspekt	Entity Core	NHibernate
Preslikavanje iz objektnog modela u relacioni, i obrnuto, sa podrškom svih tipova veza	da	da
Podrška za različite sisteme za upravljanje bazama podataka	da	da
Keširanjae entiteta	da	da
Lazy loading	da	da
Pisanja i izvršavanja uskladištenih procedura	da	ne
Posedovanje dokumentacije	da	da

Podršku za različite sisteme za upravljanje bazama podataka, kao još jedan aspekt dobrog ORM alata, imaju i Entity Core i NHibernate okvir, s tim što NHibernate okvir ima širu podršku za različite sisteme za upravljanje bazama podataka. Mogućnost keširanja entiteta poseduju oba okvira, što dodatno doprinosi boljim performansama. Učitavanje podataka u vidu lazy loading-a, još jedan je bitan aspekt dobrog ORM alata koji poseduju i Entity Core i NHibernate okvir. Kod NHibernate okvira je takav način učitavanja generički postavljen na true, dok je kod Entity Core okvira potrebno navesti da se želi ovakav način učitavanja podataka. Kada je reč o uskladištenim procedurama, Entity Core okvir daje mogućnost pisanja i izvršavanja uskladištenih procedura, dok je kod NHibernate okvira samo moguće pokretati uskladištene procedure. Oba okvira kreirana su isključivo za C# programski jezik, dok je Entity Core moguće koristiti i na Linux, iOS i drugim operativnim sistemima, pored Windows-a.

Oba okvira su besplatna i dostupna svima, poseduju dokumentaciju, s tim što je dokumentacija NHibernate okvira nešto slabija u odnosu na dokumentaciju Entity Core okvira. Što se tiče jednostavnosti upotrebe, oba okvira su praktična i jednostavna, osim što je konfiguracija NHibernate okvira sa okruženjem i bazom podataka nešto složenija nego kod Entity Core okvira.

Kada je reč o performansama Entity Core i NHibernate okvira, prilikom upisivanja, izmene i brisanja jednog ili više entiteta, bolje performanse ima NHibernate okvir (brže izvrši operaciju i alocira manje memorije u odnosu na Entity Core okvir). U tabelama 10, 11 i 12 prikazana su procentualna poboljšanja dodavanja, izmene i brisanja jednog rekorda iz tabele za NHibernate okvir, u odnosu na Entity Core okvir.

Tabela 10. Procentualno poboljšanje dodavanja novog rekorda u bazu za NHibernate okvir u odnosu na Entity Core okvir

Entitet (veza) / procentualno poboljšanje	Insert metoda- NHibernate	
	Vreme	Memorija
City	96.8%	75.1%
Person	93.4%	71.8%
Message	97.2%	77.1%
WatchedMovie	96.9%	74.2%
Friendship	92.7%	71.4%
Movie	95.1%	73.7%

Tabela 11. Procentualno poboljšanje izmene jednog rekorda u bazi za NHibernate okvir u odnosu na Entity Core okvir

Entitet (veza)/procentualno poboljšanje	Update metoda- NHibernate	
	Vreme	Memorija
City	95.6%	80.6%
Person	96.4%	85.2%
Message	96.3%	84,6%
WatchedMovie	96.2%	74.4%
Friendship	94.9%	75.0%
Movie	89.0%	82.0%

Tabela 12. Procentualno poboljšanje brisanja jednog rekorda iz baze za NHibernate okvir u odnosu na Entity Core okvir

Entitet (veza)/procentualno poboljšanje	Delete metoda- NHibernate	
	Vreme	Memorija
City	96.6%	65.3%
Person	90.8%	80.1%
Message	93.9%	79.0%
WatchedMovie	89.9%	55.7%
Friendship	89.6%	62.5%
Movie	88.0%	9.7%

Ukoliko bi u sistemu bilo tabela sa puno podataka (>200000), i potrebno je da se učitaju svi podaci tabele, bolje performase za takvu operaciju ima NHibernate, i u takvoj situaciji bi trebalo njega primeniti. Takođe, za učitavanje jednog rekorda iz baze (po Id-ju), po perfomansama, bolje je koristiti NHibernate koji brže vraća traženi rekord i alocira manje memorije bez obzira na to koliko rekorda imala tabela. Ukoliko se pri učitavanju rekorda spaja sa jednom ili više tabela, EF Core je sporiji po pitanju vraćanja podataka, međutim alocira manje memorije nego NHibernate. U ovakvoj situaciji, može se uzeti u obzir implementacija upita i spajanja sa drugim tabelama. NHibernate nudi mnogo elegantnije rešenje u tom slučaju. U tabeli 13 prikazana su procentualna poboljšanja učitavanja rekorda iz tabele za NHibernate okvir, u odnosu na Entity Core okvir. Iz tabele se vidi da za slučajeve alocirane memorije prilikom učitavanja rekorda, gde je neophodno spajanje sa drugim tabelama, NHibernate ima negativan rezultat u odnosu na Entity Core.

Tabela 13. Procentualno poboljšanje učitavanja rekorda iz baze za NHibernate okvir u odnosu na Entity Core okvir

Select upit/procentualno poboljšanje	Select metoda- NHibnernate					
	Vreme			Memorija		
	2000	25000	200000	2000	25000	200000
Učitavanje svih mesta	94.1%	85.7%	77.8%	72.4%	77.7%	62.8%
Učitavanje mesta po Id-ju	95.3%	94.2%	93.4%	58.2%	-14.3%	33.3%
Učitavanje mesta po imenu osobe	89.5%	87.2%	89.6%	-54.8%	-54.1%	-54.2%
Učitavanje mesta po oceni odgledanog filma	88.1%	88.3%	74%	-67.4%	-56.7%	-56.2%
Učitavanje mesta po žanru odgledanog filma	87.3%	90.1%	84.3%	-65.3%	-64.3%	-64.5%

Na osnovu svega izloženog, dolazi se do zaključka da oba okvira poseduju aspekte dobrog ORM alata, neke u manjoj, a neke u većoj meri.

LITERATURA

[1] Manning Publications, 2017: https://livebook.manning.com/book/entity-framework-core-in-action/chapter-12/v-8/1 (приступљено 30.07.2021.)

[2] Tutorials Point: https://www.tutorialspoint.com/nhibernate/nhibernate_orm.htm (приступљено 30.07.2021.)

[3] Entity Framework: https://entityframework.net/ef-vs-nhibernate (приступљено 30.07.2021.)

[4] NHibernate.info: https://nhibernate.info/doc/index.html (приступљено 31.07.2021.)

[5] Microsoft: https://docs.microsoft.com/en-us/ef/core/performance/ (приступљено30.07.2021.)

[6] Allen, J. (2020, јануар 30.). InfoQ: https://www.infoq.com/news/2020/01/EF-Core-Performance-Collections/ (приступљено 30.07.2021.)

[7] Boake, A., Zyl, P. V., Kourie, D. G. (2015). *Comparing the Performance of Object Databases and ORM Tools.*

[8] Chatekar, S. (2015). *Learning NHibernate 4.* Birmingham, UK: Packt Publishing.

[9] Gerr, P. (2020, јун 10.). Think texture: https://www.thinktecture.com/en/entity-framework-core/execution-plans-in-3-1/ (приступљено 29.07.2021.)

[10] Goos, G., Hartmanis, J., Leeuwen, J. v. (2010). Lecture Notes in Computer Science. *Third International Conference, ICOODB.* Frankfurt/Main, Germany: Springer.

[11] Jones, M. (2019), Exception not found: https://exceptionnotfound.net/dapper-vs-entity-framework-core-query-performance-benchmarking-2019/ (приступљено 15.06.2021.)

[12] Korzh, S. (2019, август 6.), Gitconnected: https://levelup.gitconnected.com/3-ways-to-improve-the-ef-core-performance-in-your-net-core-app-d9b6295188cc (приступљено 15.07.2021.)

[13] Kuate, P. H., Harris, T., Bauer, C., King, G. (2009). *NHibernate in Action.* Greenwich, CT: Manning Publications.

[14] Kumar, A. (2012, јул, 30.), C# Corner: https://www.c-sharpcorner.com/blogs/nhibernate-vs-entity-framework1 (приступљено 15.08.2021.)
[15] Penniman, J. M. (2019). *Using NHibernate in Asp.Net Core.* Massachusetts, United States.
[16] Peres, R. (2014). *NHibernate Succinctly.* Morrisville, NC: Syncfusion.
[17] Peres, R. (2016). *Entity Framework Core Cookbook.* Birmingham, UK: Packt Publishing Ltd.
[18] Peres, R., (2018, март,21). *Stackify*. https://stackify.com/entity-framework-core-nhibernate/ (приступљено 14.08.2021.)
[19] Schwichtenberg, H. (2018). *Modern Data Access with Entity Framework Core.* Essen, Germany: Apress.
[20] Smith, J. P. (2018). *Entity Framework Core in Action.* Shelter Island, NY: Manning Publications Co.
[21] Vijay P. Mehta (2008). *Pro LINQ Object Relational Mapping with C#.*
[22] Andrew B. (2015). *Comparing the Performance of Object Databases and ORM Tools*, Stellenbosch University
[23] Cvetkovic S., Jankovic S. D. (2019). *A Comparative Study of the Features and Performance of ORM Tools in a .NET Environment*, University of Nis
[24] Microsoft: https://docs.microsoft.com/en-us/ef/core/performance/performancediagnosis?tabs=simple-logging%2Cload-entities (приступљено 01.05.2021.)
[25] Microsoft: https://docs.microsoft.com/en-us/ef/core/logging-events-diagnostics/event-counters?tabs=windows (приступљено 01.05.2021.)
[26] Zmaranda D., Lucian-Laurentiu Pop-Fele, Győrödi C., Gyorodi R. (2020), *Performance Comparison of CRUD Methods using NET Object Relational Mappers: A Case Study*, University of Oradea
[27] Microsoft: https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/(посећено 25.06.2021.)
[28] Microsoft: https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key(посећено 25.06.2021.)
[29] Vignesh S. (2020, септембар, 18). Study: https://study.com/academy/lesson/object-persistence-definition-overview.html (посећено 21.06.2021.)
[30] Hibernate: https://hibernate.org/orm/what-is-an-orm/(посећено 22.06.2021.)
[31] Benchmark DotNet: https://benchmarkdotnet.org/articles/overview.html(посећено 02.05.2021.)
[32] Microsoft: https://docs.microsoft.com/en-us/ef/core/querying/ (посећено 22.07.2021.)
[33] Microsoft: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history(посећено 23.07.2021.)
[34] Microsoft: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/configure-language-version (посећено 23.06.2021.)
[35] Microsoft: https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.entitystate?view=efcore-5.0(посећено 22.06.2021.)
[36] Ogheneovo, Edward Erhieyovwe, Asagba, Prince Oghenekaro, Ogini, Nicholas Oluwole (2013), *An Object Relational Mapping Technique for Java Framework*
[37] Gošić, B., Lazarević, S. D., (2021). Komparativna analiza razvoja jednostraničnih aplikacija korišćenjem Redux paterna i konvencionalnom metodom razvoja. Info M
[38] Jevtić, D., Lazarević, S. D., Stojanović T. (2020). Razvoj softverskog sistema za generisanje cross-platform korisničkog interfejsa. Info M

Tijana Milošević, master inženjer organizacionih nauka, softverski inženjer u GiveSense d.o.o
Kontakt: tijana.milosevic@saga.rs
Oblasti interesovanja: softversko inženjerstvo, .NET platforma, .NET Core

prof. dr Saša D. Lazarević, Univerzitet u Beogradu – Fakultet organizacionih nauka
Kontakt: sasa.lazarevic@fon.bg.ac.rs
Oblast interesovanja: softversko inženjerstvo, informacioni sistemi, baze podataka, sistemi za upravljanje dokumentacijom, .NET platforma