LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

SOCIĀLO TĪKLU LIETOTNES IZVEIDE IZMANTOJOT NOSQL

MAĢISTRA DARBS

Autors: **Askolds Molotoks**
Stud. Apl. Nr.: am07019
Darba vadītājs: asoc. prof. Ģirts Karnītis

RĪGA 2013

ANOTĀCIJA

Maģistra darbā "*Sociālo tīklu lietotnes izveide izmantojot NoSQL*" tiek izveidotā lietotne, izmantojot *NoSQL* datu bāzi. *NoSQL* – ne tikai *SQL* ir jauna datu bāzes pieeja, kas ļauj izmantot dažādus veidus datu glabāšanai. Maģistra darbā tiek analizētas izstrādes pieejas, izmatojot *SQL* un *NoSQL* datu bāzes tehnoloģijas.

Maģistra darbā ir iekļautas nodaļas, kurās ir aprakstīta sociālo tīklu vēsture, sociālie tīkli un lietotnes. Ir aprakstītas *NoSQL* atšķirības no *SQL*, kā arī apskatītas *NoSQL* datu bāzes. Tiek izskaidrota lietotņu darbība un izstrādes principi. Darbā autors izklāsta, kā tiek izstrādāta lietotne priekš dažādiem sociāliem tīkliem.

Atslēgvārdi: sociālie tīkli, *NoSQL*, *SQL*, lietotnes.

ABSTRACT

In this dissertation the author describes "Social network application development using *NoSQL*", developing applications using a *NoSQL* database. *NoSQL* – not only *SQL* is a new database approach, which allows using different types of data storage. This work explains and analyzes development approach using the *SQL* and *NoSQL* database technology.

Dissertation includes chapters describing the social network history, social networks and applications. *SQL* versus *NoSQL* differences are described as well as view on *NoSQL* database. Also document cover basics of application development. The author describing how the app is being developed for a variety of social networks.

Keywords: social network, *NoSQL*, *SQL*, applications.

АННОТАЦИЯ

В диссертации "Разработка социальных сетевых приложений с использованием *NoSQL*" создается приложение с помощью базы данных *NoSQL*. *NoSQL* – не только *SQL* новый подход к базам данных, который позволяет хранить различные типы данных, а также в работе рассмотрен подход разработки, используя *SQL* и *NoSQL* технологии базы данных.

Диссертация включает в себя главы в которых описаны история социальных сетей, социальные сети и приложения. Описывается *NoSQL* различия от *SQL*, рассмотрены *NoSQL* базы данных. Объясняются основы работы и разработки приложений. Автор описывает, как приложение разрабатывается для различных социальных сетей.

Ключевые слова: социальные сети, *NoSQL*, *SQL*, приложения.

AUTOREFERĀTS

Pirms rakstīt maģistra darbu autora zināšanas par bez relāciju datu bāzēm bija ļoti mazas. Tomēr rakstot darbu autors iepazinās ar bez relāciju datu bāzes vēsturi un tagadējo attīstības situāciju.

Iepazīstinoties ar bez relāciju datu bāzēm tika izvēlēta *MongoDB* datu bāze. Pēdējo divu gadu laikā šī ir viena no populārākam *NoSQL* tipa datu bāzēm. Globālajā tīmekli ir daudz informācijas par *MongoDB* datu bāzi. Kā arī ir vairākas grāmatas, kuras autors lasīja un iepazinās tuvāk ar *MongoDB* datu bāzi.

Maģistra darba gaitā tika izstrādāta lietotne izmantojot *MongoDB* datu bāzi. Lietotne darbojās iekš sociāliem tīkliem *draugiem.lv* un *facebook.com*. Lietotne ievāca dažādus datus, lai varētu aprakstītas lietotnes metrikas.

Maģistra darbs palīdzēja autoram vairāk uzzināt par bez relāciju datu bāzēm un darba ietvaros pāriet no *MySQL* datu bāzes izmantošanas uz *MongoDB* datu bāzes implementāciju projektos.

SATURS

APZĪMĒJUMU SARAKSTS

Saīsinājums	**Skaidrojums**
ActionScript	(ang. *ActionScript*) – programmēšanas valoda, kuru izmanto *Flash* izstrādes rīkos. Pašlaik pēdējā versija ir 3.0, kura sintakse ir ļoti līdzīga *C++* programmēšanas valodai.
API	(ang. *Application program interface*) – lietojumprogrammas interfeiss. Lietojumprocesos izmantojama pilna operētājsistēmas funkciju specifikācija, kā arī šo funkciju izmantošanas procedūru apraksts.
APP	(ang. *Application*) – specifiska uzdevuma programmatūra. Sociālo tīklu lietotnes tiek integrēta sociālajā tīklā, sniedz iespēju ievietot ārējās lapas saturu kā atsevišķas sadaļu.
C, C++	programmēšanas valoda, kas izstrādāta 70. gadu sākumā, lai atvieglotu programmatūras pārnešanu no viena datora uz otru. Tajā apvienotas augsta līmeņa mašīnneatkarīgas valodas un asamblervalodas iespējas.
BSON	(ang. *binary JavaScript Object Notation* - *JavaScript* Objektu Notācija) – binārs datu apmaiņas formāts.
DBVS	(ang. *Database management system* (*DBMS*)) – datu bāžu vadības sistēma
Flash CS6	populārs multimediju un programmatūras izstrādes rīks.
OLAP	(ang. *OnLine Analytical Processing*) – daudzdimensionālas datu bāzes analīzes metode, kas nodrošina specifisku datu bāzu indeksāciju, tādējādi paātrinot piekļuvi datiem gadījumos, kad jāpārskata lieli datu masīvi, kā arī ļaujot analizēt datus daudzos dažādos aspektos.
SQL	(ang. *Structured Query Language*) – strukturēta vaicājumvaloda. Valoda *SQL* tiek izmantota klientservera arhitektūras tīklos, lai nodrošinātu personālajiem datoriem piekļuvi kopīgi izmantojamu datu bāzu resursiem
HTML	(ang. Hyper Text Transfer Protocol) – valoda, kas, izmantojot speciālus kodus, nosaka hiperteksta dokumenta atveidojumu displeja ekrānā gadījumā, ja tiek lietotas tīkla Internet globālā tīmekļa lappuses.
HTTP	(ang. *HyperText Transfer Protocol*) – ir lietojumslāņa protokols, kas paredzēts datu apmaiņai starp tīmekļa serveriem un pārlūkprogrammām. Tas ir galvenais informācijas pārraides veids vispasaules tīmeklī.

NoSQL	(ang. *Not Only SQL*) – ne tikai *SQL*. Strukturēta vaicājumvaloda, kas radīta, lai atbilstu tīmekļa mēroga prasībām.
JSON	(ang. *JavaScript Object Notation* - *JavaScript* Objektu Notācija) – datu apmaiņas formāts. Balstīts uz teksta formātu, viegli lasāms.
MongoDB	(ang. *Mongo Data Base*) – atvērta koda dokument orientētā datu bāze. Viena no vadošajām *NoSQL* datu bāzēm.
MySQL	(ang. *My Structured Query Language*) – viena no pasaules populārākam relāciju datu bāzes vadības sistēmam.
PHP	(ang. *Personal Home Page*) – hiperteksta priekšprocesors, populāra atklātā pirmkoda programmēšanas valoda, kas ļauj izstrādāt tīmekli dinamisku un interaktīvu tīmekļa lapu un sasaistīt to ar datu bāzi
XML	(ang. *eXtensible Markup Language*) – ir speciālas nozīmes iezīmēšanas valoda.

IEVADS

Pēdējo gadu laikā sociālie tīkli ir mūsu dzīves neatņemama daļa. Tā kā to attīstība ir ļoti strauja un ir pieejami dažādi sociālie tīkli un to paveidi, tad katram ir iespējams izmantot sev tīkamāko sociālo tīklu. Pārsvara cilvēki izvēlās lietot kādu pašmāju sociālos tīklus, piemēram, Latvijā "draugiem.lv", "face.lv" vai "one.lv". Pašlaik ir aptuveni ir 200 populāru sociālo tīmekļa vietņu [1], kuros ir reģistrējušies tūkstošiem pat miljoniem lietotāju.

Sociālie tīkli piedāvā ne tikai kontaktēties un sazināties ar citiem lietotājiem, bet tos var uzskatīt par platformām lietotņu izveidei un integrēšanas portālos. Izstrādātājs var izveidot jebkādu lietotni, kas nepārkāpj sociālā tīkla lietotņu izstrādes noteikumus. Parasti lietotnes ir balstītas uz lietotāju datu ievākšanu un turpmāko apstrādi. Izmantojot lietotnes ir iespējams mijiedarboties ar sociālā tīkla lietotājiem, lietotāju informāciju un saturu.

Maģistra darba tēma " *Sociālo tīklu lietotnes izveide izmantojot NoSQL*" tika izvēlēta, lai izveidotu lietotni, kura datus glabā *NoSQL* datu bāzē, kas tieši domāta datu glabāšanai globālajā tīmekli, kur datu struktūra var būt dažāda, bet ierakstu skaits mērāms tūkstošos.

NoSQL izstrādāji apgalvo, kā izstrādāt programmas uz *NoSQL* datu bāzes ir ātrāk par relāciju datu bāzes izstrādes pieeju [2], tāpēc šīs apgalvojums autoram likās apšaubāms un to vajadzētu izpētīt.

NoSQL datu bāzes izvēlei ir vairāki iemesli. Pirmkārt, daudzi sociālie tīkli pamazām pariet uz *NoSQL* datu bāzēm [3], piemēram, *FourSquare* tiešsaistes sociālā tīkla tīmekļa vietne, kas balstīta uz lietotāju atrašanās vietas norādīšanu kartē. Otrkārt, autoram darba vietā ir nepieciešams izstrādāt lietotnes priekš sociāliem tīkliem, tomēr, izmantojot relāciju datu bāzes izstrāde ir laikietilpīgs process, kuru vajadzētu veikt ātrāk. Treškārt autors jau sen gribējis apgūt *NoSQL* datu bāzes izstrādes pieeju. Par *NoSQL* autors bija dzirdējis no apmeklētām prezentācijām par *NoSQL* datu bāzēm un to priekšrocībām par *SQL* datu bāzēm. Tomēr, autoram nebija tik liela vajadzība, lai lietotu *NoSQL* datu bāze. Tāpēc šāda tēmas izvēle palīdzēs izpētīt un saprast *NoSQL* plusus un mīnusus, analizēt *NoSQL* un *SQL* atšķirības, padziļināt zināšanas datu bāzēs un strukturētā vaicājumvalodā.

Darba autors darbā pārsvara izmanto *MySQL* datu bāzi, tomēr palielinoties lietotāju skaitam *MySQL* veiktspēja samazinās. Tā kā resursi ir ierobežoti ir vajadzība pēc jaunām tehnoloģijām, lai atrisinātu radušo problēmu.

Maģistra darba mērķis ir izstrādāt lietotni priekš sociāliem tīkliem *Facebook* un *draugiem.lv* izmantojot *NoSQL* datu bāzi. Šāda tipa lietotnes jau eksistē. Tāpēc darba uzdevums nav radīt jaunu lietotni, kas izmanto *NoSQL* datu bāzi, bet gan izpētīt, kā tiek

izstrādās lietotnes izmantojot šāda tipa datu bāzi kopā ar *PHP* programmēšanas valodu. Analizēt uz kādā tipa datu bāzēm lietotnes izstrādāt ir ātrāk un uz, kuras no tām lietotne strādās ātrāk.

Maģistra darba mērķu sasniegšanai bija veikti šādi soļi:

- doti ieskati par populārākajiem sociālajiem tīkliem;
- analizēti *NoSQL* un *SQL* būtiskākās atšķirības;
- izvēlēta uzdevumam atbilstoša *NoSQL* datu bāze;
- izveidota bibliotēka *PHP* valodā darbā ar *NoSQL* pieprasījumiem;
- aprakstīti lietotnes izstrādes principi priekš sociāliem tīkliem;
- analizēta *NoSQL* un *SQL* ātrdarbība.

Maģistra darbs ir sadalīts nodaļās. Pirmajā darba nodaļā ir aprakstīta sociālo tīklu vēsture, sociālie tīkli un lietotnes. Otrajā darba nodaļā ir aprakstīta datubāžu vēsture, *SQL* un *NoSQL* datu bāzes, izpētīts *NoSQL* un *SQL* būtiskas atšķirības. Trešajā darba nodaļā ir aprakstīta lietotnes projektēšana un realizācijas gaita, izmantojot *NoSQL* datu bāzi. Darbs noslēdzas ar rezultātiem un secinājumiem.

1. SOCIĀLIE TĪKLI

Tiešsaistes sociālais tīkls – tīmekļa vietne, kurā, reģistrējoties un izveidojot individuālo profilu, ir iespējams kontaktēties un sazināties ar citiem lietotājiem (indivīdiem, grupām vai organizācijām). Saziņa sociālajā tīklā notiek dažādos veidos: var paust savu viedokli dienasgrāmatās un forumos, var veidot interešu grupas, apmainīties ar fotogrāfijām, augšupielādēt audio un video failus, sūtīt vēstules u.tml. Lietotājus var vienot intereses, gaume vai paziņas. Pēdējā laikā galvenais uzsvars tiek likts uz savas virtuālas identitātes veidošanu, izkopšanu un popularizēšanu, nevis savstarpējas komunikācijas veidošanu [4].

Visus sociālos tīklus var iedalīt četrās lielās kategorijās:

- atvērtie sociālie tīkli – tādi kā *MySpace,* kur katrs var reģistrēties, komentēt jebkura lietotāja profilā, piedalīties gandrīz visās aktivitātēs, kā arī visa informācija ir publiski pieejama;
- slēgtie sociālie tīkli – tādi kā *draugiem.lv,* kur reģistrācija tikai ar ielūgumiem, gandrīz nekāda informācija nav pieejama nereģistrētajiem lietotājiem;
- mazu emuāru sociālie tīkli – tādi kā *Twitter*, kurā komentāru savā profilā var ierakstīt tikai pats emuāra autors;
- servisi ar sociālo tīklu funkcionalitāti – tādi kā *LinkedIn*, kur visi reģistrētie lietotāji var viens otram nosūtīt ziņojumus, apskatīt *CV* un ieteikt do darba devējiem.

1.1. Vēsture

Sociālie tīkli bija radušies pat pirms globāla tīmekļa ēras. Cilvēki saprata kā ar datoru komunikācijas starpniecību var veikt daudzus tiešsaistes pakalpojumus *ARPANET* bija viens no tiem. Daudzas funkcijas, ko mūsdienu sociālie tīkli piedāvā bija izveidotas agrāk dažādās vietnēs kā *America Online, Prodigy, CompuServe, ChatNet,* un *The WELL*. Pirmās vietnes, kuras bija līdzīgas sociāliem tīkliem vairāk izskatījās pēc kopienām. Tās bija radītas jau 1994. gadā *Geocities* un 1995. gadā *Theglobe.com, Tripod.com.*

Mūsdienu sociālo tīklu laikmeta sākums ir meklējams 1997. gada sākumā. Tieši tad jauna Ņujorkas vietne *sixdegrees.com* ieviesa līdz šim nepiedzīvotu pakalpojumu, kurā tika izmantoti reāli vārdi. 2007. gadā divas interneta socioloģes, Dana Boida un Nikola Elisona, kādā rakstu darbā skaidri formulēja īsta sociālā tīkla galvenās iezīmes: tas ir pakalpojums, kura lietotāji var "izveidot atklātu vai daļēji atklātu profilu", "skaidri formulēt citu lietotāju

sarakstu, ar kuriem viņiem ir kopīga saikne" un "sistēmas ietvaros pārskatīt un apceļot gan paši savu, gan citu veidoto kontaktpersonu sarakstu". Nosakot savu pozīciju sarežģītā attiecību tīklā, ar profilu var sevi pozicionēt šo attiecību kontekstā, parasti, lai atklātu citādi slēptus kopīgu interešu vai sarakstu krustpunktus.

Par sociālo tīklu strauju attīstību un pieaugumu var uzskatīt 2003. un 2004. gadu, kad tika palaisti *LinkedIn*, *MySpace* un *Facebook* sociālie tīkli. Latvijā mode uz sociāliem tīkliem sākās 2004. gadā, kad tika palaists pašmāju sociālais tīkls *draugiem.lv*.

Tagad sociālie tīkli ir izplatījušies visā pasaulē un *Facebook* ir lielākais no tiem. *Facebook* vai *MySpace* savā ikdienā neizmanto tikai retais vidusskolēns un koledžas students. Šīs sistēmas ir tik uzmācīgi ielauzušās saziņā, ka daudzi dažāda vecuma cilvēki vairs gandrīz neizmanto elektronisko pastu. Sociālie tīkli, sākot ar *sixdegrees.com* un *Friendster* un beidzot ar *Facebook*, ir kļuvuši par pazīstamu un visuresošu interneta daļu.[5]

1.2. Sociālie tīkli ārzemēs

Ir ļoti daudz dažnedažādu veidu sociālo tīklu ārzemēs. Neapšaubāmi katrs no tiem ir populārs attiecīgajā valstī, bet tomēr daži ir zināmi pasaules līmenī:

- *Facebook* – *Facebook.com*, tika atvērts 2004. gada 4. februārī. Lietotāji var pievienoties tīkliem, kas sakārtoti pa pilsētām, darbavietām, skolām un reģioniem, sazināties ar citiem cilvēkiem. Lietotāji arī var pievienot draugus, sūtīt elektroniskās vēstules un publicēt ziņas par sevi. *Facebook* portālā ikvienam tā lietotājam ir iespējams spēlēt neskaitāmas spēlēs, veidot savas lapas un domu grupas. *Facebook* laika gaitā ir ļoti mainījies un sastāv no daudziem elementiem – lietotāju profiliem, domubiedru grupām, spēlēm un vairāk nekā 90000 lietotņu.

  Sociālā tīkla *Facebook* lietotāju skaits pasaulē uz 2012. gada oktobri bija sasniedzis 1 biljonu [6] un turpina veiksmīgi augt. *Facebook*, skaitļi tiešām ir iespaidīgi. Mēneša laikā *Facebook* 350 miljoni lietotāju apskata 200 miljardus lapu šajā portālā. Portālā esošajā mini čatā katru dienu tiek nosūtīti 1.6 miljardi ziņu un katru sekundi portālā tiek ievietotas 1.4 miljoni fotogrāfijas. Portāls ir uzrakstīts uz *C++* un *PHP* programmēšanas valodām, kā arī izmanto *NoSQL* datu bāzes sistēmas. Attēlā 1.1. ir redzams *Facebook* logo, kas ir uz zila fona, jo tā veidotājs ir Marks Zukerbergs, kas ir daltoniķis un šo krāsu viņš redz bez problēmām. Tieši tāpēc šī krāsa dominē visā portālā;

1.1.att. **Facebook logo**

- *MySpace* – *MySpace*.com, tika atvērts 2003. gada augustā. Lietotājiem portālā ir iespējams veidot interešu grupas, personiskos profilus, rakstīt emuārus, izvietot foto un video saturu un pati galvenā atšķirība no citiem sociāliem tīkliem ir iespēja klausīties populāro izpildītāju dziesmas. Pašlaik *MySpace* lieto apmēram 25 miljonu aktīvo lietotāju, tomēr ar katru gadu šo lietotāju skaits sarūk. Nesen tika palaista jauna versija, ar pilnīgi citu dizainu, kas ir pieejama tiešsaistē *new.MySpace.com*. Portāls ir uzrakstīts izmantojot *ColdFusion* platformu vai arī, kā mēdz teikt - ietvaru. Attēlā 1.2. ir redzami *MySpace* logo pa kreisi vecais, pa labi jaunais logo variants;

1.2.att. **MySpace logo**

- *Google+* – *plus.google.com*, tika palaists 2011. gada 28. jūnijā. Šo tiešsaistes sociālo tīmekļa vietni piedāvā pasaules slavena kompānija *Google*. *Google+* ir vairāku iepriekšējo *Google* kompānijas servisu apkopojums. Tika izveidots jauns serviss ar nosaukumu *Circles*, tulkojumā no angļu valodas - Apļi. *Circles* piedāvā iespēju veidot interešu grupas, atzīmēt draugus un dalīt tos pa grupām. Katram lietotājam ir savs profils, kur var izvietot personīgo saturu. Sociālā tīkla *Google+* lietotāju skaits pasaulē uz 2012. gada decembri bija sasniedzis 500 miljonus. Portāls ir uzrakstīts izmantojo *Java* un *JavaScript* programmēšanas valodās, kā arī tiek izmantotas *NoSQL* datu bāzes. Attēlā 1.3. ir attēlots *Google+* logo;

1.3.att. **Google+ logo**

- *VK –vk.com* jeb oriģinālā *В Контакте*, tika palaists 2006. gada 10. oktobrī. Šī sociāla tīmekļa vietne tika veidota Krievijas globālā tīkla lietotājiem. Tomēr šīs vietnes lietotāji nav tikai Krievijas iedzīvotāji, bet arī citas valstis, kur tiek runāts krieviski. *VK* piedāvā gandrīz tās pašas iespējas ko citi sociālie tīmekļi, tomēr šajā vietnē ir iespējas augšupielādēt un dalīties ar savies failiem (parasti video un audio faili). Aktīvo lietotāju skaits ir 38 miljoni. Attēlā 1.4. ir attēlots *VK* logo. Var saskatīt līdzību ar *Facebook* logo, kā arī pats portāls vismaz saskares ziņā ir *Facebook* klons. Portāls ir uzrakstīts uz *C++* un *PHP* programmēšanas valodām, kā arī izmanto *NoSQL* datu bāzes sistēmas. Tā īpašnieks ir Pavel Durov.

1.4.att. **VK logo**

1.3. Sociālie tīkli Latvijā

Latvijā sociālie tīkli parādījās 2004. gadā, bet to straujš popularitātes līmeņa pieaugums bija 2005. un 2006. gads. Daudzi no tiem ir populāri arī šodien:

- *Draugiem – draugiem.lv* tika palaists 2004. gada 24. martā. Sākumā piedāvāto pakalpojumu skaits bija mazs. Bija iespēja uzlūgt draugus, sākt draudzēties ar citiem draugiem vai paziņam, sūtīt vēstules un augšupielādēt bildes. Portāls nespēja nodrošināt vairāk par pāris simtiem lietotāju tiešsaistē. Neraugoties uz grūtībām portāls attīstījās un strauji pieauga lietotāju skaits, kurš 2007. gadā sasniedza 1 miljonu. Šobrīd ir reģistrēti 2,6 miljonu lietotāju. Katru dienu portālu apmeklē gandrīz pusmiljons lietotāju, bet tiešsaistē atrodas ap 40 līdz pat 80 tūkstošiem lietotāju. Portālā ir iespējams izveidot uzņēmumu jeb biznesa lapu. Pašlaik portālā ir ap 22 tūkstošiem uzņēmumu lapu - to skaits strauji pieaug. Portāls draugiem.lv tiek izmantots kā platforma, uz kuras izstrādātāji var veidot savas lietotnes un piesaistīt jaunus sekotājus uzņēmumu lapām. Lielākā daļa lietotņu ir spēles. Viena no populārākam lietotnēm savulaik bija spēlē *Ferma*. Portāls ir uzrakstīts uz *C++* un *PHP* programmēšanas valodām, izmanto relāciju datu bāzes. Portālam ir divi īpašnieki Lauris Liberts un Agris Tamanis. Attēlā 1.5. ir attēlots *draugiem.lv* logo;

1.5.att. ***Draugiem.lv*** **logo**

- *One – one.lv* tika palaists 1999. gadā. *One.lv* ir lielākā tiešsaistes sociālā tīkla tīmekļa vietne krievvalodīgai auditorijai Latvijā. *One.lv* ir vieta, kur ir iespējams komunicēt ar draugiem, iepazīties un atrast domubiedrus. Lietotāji piedalās diskusijās, veido interešu grupas, apmainās ar fotogrāfijām un gūst jaunas zināšanas. Reģistrēto lietotāju skaits pārsniedza 850 tūkstošus. Diemžēl *One.lv* portāls nespēja konkurēt ar sociāliem tīklu gigantiem *Facebook* un *VK*, kā arī vietne nepārgāja uz platformas veidu. Tādēļ tā šis portāls paziņoja, ka 2013. gada 31. janvārī beidz savu eksistenci. Portāls ir uzrakstīts izmantojot *PHP* programmēšanas valodu, izmanto relāciju datu bāzes. Attēlā 1.6. ir attēlots *one.lv* logo.

1.6.att. **One.lv logo**

1.4. Lietotnes

Lielākoties sociālos tīklus var uzskatīt par platformām, kurās lietotāji paši veido saturu, bet portāls nodrošina šāda veida pakalpojumu. Lietotnes var iedalīt spēlēs un biznesa tipa lietotnes. Spēļu lietotnes neapšaubāmi tiek apmeklētas daudz vairāk par citu veidu lietotnēm, tomēr to uzdevums ir lietotāju izklaide un laika pavadīšana mierīga gaisotnē. Savukārt biznesa lietotnēm ir mērķis vai vēstījums, ko vajag parādīt lietotājam. Mēdz arī būt lietotnes, kas domāts vienam mērķim, piemēram, atgādināt par draugu dzimšanas dienām.

Pārsvarā, lai kādā zīmola biznesa lapa kļūtu populāra vajag, lai to pēc iespējas vairāk apmeklētu lietotāji. Labākais veids kā to panākt, ir izveidot lietotni, kur lietotājam, lai pilnībā varētu izbaudīt sniegtās lietotnes iespējas vajag kļūt par biznesa lapas sekotāju vai nospiest, ka šī lapa viņam patīk. Kad lietotājs ir kļuvis par zīmola fanu, viņa datus var izmantot, lai komunicētu ar lietotāju zīmola vārdā. Attēlā 1.7. ir redzama Ādažu Čipsu "Seko un laimē"

lietotnē, kas ir integrēta *draugiem.lv* portālā, kuras uzdevums ir piesaistīt lapai jaunus sekotājus, un starp tiem izlozēt kasti čipsus.

1.7.att. **Ādažu Čipsu "Seko un laimē" lietotne**

Bieži vien sanāk, ka lietotnes ir ļoti populāra un to apmeklē tūkstošiem lietotāju, lietotne kļūst lēna (servera atbildes laiks uz lietotāja pieprasījumu palielinās). Šādos brīžos vājā vieta ir tieši datu bāze. Parasti šādā situācijā lietotnē ir vairāki datu bāzes pieprasījumi, kas bieži vien ir samērā lēni, dēļ pieprasījumu skaita vai sarežģītības pakāpes dēļ. Tāpēc ir nepieciešamība pēc savādākas, ātrākas datu apstrādes, ko sniedz *NoSQL* datu bāzes.

2. DATU BĀZES

Datu bāze – savstarpēji saistītu informacionālu objektu tematisks kopums, kas ar speciālas pārvaldības sistēmas starpniecību izveidots un organizēts tā, lai nodrošinātu tajā ievadītās informācijas izguvi, veiktu tās atlasi un kārtošanu[7]. Pārsvarā, kad tiek runāts par datu bāzēm prasti domā par relāciju datu bāzēm, bet bez relāciju datu bāzes atstāj novārtā. Tomēr abām šīm datu bāzēm ir savi plusi un mīnusi. Bez relāciju datu bāzes ir jauna pieeja izstrāde ar datu bāzēm, tāpēc, nevajadzētu būt konservatīviem un pamēģināt jaunas tehnoloģijas dažādos izstrādes uzdevumos. Informācija par *SQL* un *NoSQL* 2. nodaļā ir apkopota no vairākiem avotiem [8], [9],[10].

2.1. SQL vēsture

SQL – Structured Query Language jeb tulkojumā no angļu valodas strukturēta vaicājumvaloda. Speciāla programmēšanas valoda, kas ir paredzēta manipulācijai ar datiem relāciju datu bāzes pārvaldības sistēmas. Ar šo valodu var rakstīt funkcijas, procedūras, veidot skatus, trigerus, meklēt datus un vienkārši pievienot, labot un dzēst datus datu bāzē. Pirmo reizi par *SQL* sāka runāt jau 1970. gados, kad kompānija *IBM* izveidoja *SEQUEL* (*Structured English Query Language*). Šī valoda bija paredzēta, lai manipulētu ar datiem priekš *IBM* relāciju datu bāzes pārvaldības sistēmas. Tomēr *SEQUEL* tika nomainīts uz *SQL*, jo *SEQUEL* jau bija reģistrēta aviokompānijas preču zīme. 1970. gadu beigās korporācija *Relational Software* (tagadēja *Oracle* korporācija) saredz potenciālu *SQL* koncepcijas un izveido savu *SQL* bāzētu valodu, ko centās pārdod. 1979. gadā *Relational Software* iepazīstina ar pirmo komerciāli pieejamo *SQL* implementāciju, *Oracle* priekš *Virtual Address eXtension* datoriem.

Pirmais *SQL* valodas standarts *SQL-86* bija izveidots 1986. gadā. Pēc tam ir bijuši daudzi standarta papildinājumi. Šobrīd pēdējā standarta versija ir *SQL-2011*, kurš izveidots 2011. gadā.

2.2. SQL datu bāzes

Populārākas *SQL* datu bāzes:

- *MySQL* – ir atvērtā koda relāciju datu bāzes pārvaldības sistēma. Tiek izmantota gandrīz visos atvērtā koda tīmekļa projektos, kur nepieciešams glabāt datus. Datu bāze ir uzrakstīta, izmantojot *C* un *C++* programmēšanas valodas. Datu glabāšanai tiek izmantoti bināri koki ar indeksu kompresiju augstai veiktspējai. *MySQL* atbalsta sadalīšanu un replikācijas, pilna teksta meklēšanu un procedūras, trigerus skatus, transakcijas utt. Attēlā 2.1. ir attēlots *MySQL* logo;

2.1.att. **MySQL logo**

- *PostgreSQL* – vai vienkārši *Postgres* ir objektu relāciju datu bāzes pārvaldības sistēma. Šī sistēma arī ir atvērtā koda programmatūra. Atbalsta *ACID* transakcijas, replikācijas, ļoti pielāgojamas, savienojumus, procedūras, trigerus skatus utt. Bez tā ir iebūvēts versiju kontroles un rezerves kopiju veidošanas rīks. Datu bāze ir uzrakstīta, izmantojot *C* programmēšanas valodu. Attēlā 2.2. ir attēlots *PostgreSQL* logo;

2.2.att. **Postgre*SQL* logo**

- *Oracle* – vai *Oracle DB* ir objektu relāciju datu bāzes pārvaldības sistēma. Šī datu bāze ir komerciāls produkts. Atbalsta *Real Application Cluster*, datu aizsardzību, privātos virtuālas datu bāzes, *OLAP* rīkus, kā arī procedūras, trigerus skatus utt. Datu bāze ir uzrakstīta, izmantojot *C++* programmēšanas valodu. Attēlā 2.3. ir attēlots *Oracle* logo;

2.3.att. **Oracle logo**

- *MS SQL Server* – ir relāciju datu bāzes pārvaldības sistēma, ko izstrādāja kompānija Microsoft. Pirmās versijas jau bija 1989. gadā. Šī datu bāze ir komerciāls produkts. Izmanto *T-SQL* pieprasījumu valodu. Datu bāze ir uzrakstīta, izmantojot *C++* programmēšanas valodu. Attēlā 2.4. ir attēlots *MS SQL Server* logo.

2.4.att. **MS SQL Server logo**

2.3. NoSQL vēsture

Carlo Storzzi pirmo reizi izmantoja terminu *NoSQL* 1998. gadā, lai nosauktu savu "vieglo" atvērtā koda relāciju datu bāzi, kas neizmanto standarta *SQL* saskarni.

Eric Evans atkal izmantoja terminu *NoSQL* 2009. gada sakumā, kad "Last.fm" apsprieda atvērtā koda izplatītās datu bāzēs.

2011. gadā tika iesākt darbs pie *UnQL* (*Unstructured Query Language*), specifiska vaicājumu valoda priekš *NoSQL* datu bāzēm.

2.4. NoSQL iedalījums

NoSQL datu bāzes var iedalīt trijās lielās grupās:

- *Key-value* – atslēga-vērtība datu bāzes vadības sistēma, kura saglabā datu pārus atslēga = vērtība datu bāzē. Datu bāzē nolasa atslēgas, pēc kurām iegūst attiecīgas vērtības. Ar šādu pieeju var efektīvi saglabāt un nolasīt vērtības, tiek izlaists *SQL* pieprasījuma līmenis, nav indeksācijas;

  Pie atslēga-vērtība tipa datu bāzēm var uzskatīt šādas datu bāzes: *Voldemort, Ringo, Scalaris, Cassandra* un *Hypertable*;

Salīdzinot ar parastu relāciju datu bāzi, piemēram, *MySQL*, lai uztaisītu lietotāja pilnvarošanas sistēma vajag trīs laukus: (*ID* | *login* | *password*). Kad notiek reģistrācija pārbaudīta vai *login* (lietotāja vārds) jau nav tabulā, ja, nav pieliek jaunu ierakstu. Pie pilnvarošanās pēc *login* tiek atrasta parole (*password*) un pārbaudīta ar ievadīto. Šim uzdevumam netiek izmantots *MySQL* potenciāls, respektīvi ir citi efektīvāki risinājumi kā *key-value* vadības sistēmas. Šis pats uzdevums izmantojot *key-value* DBVS. Tātad *login* ir unikāls sistēmas ietvaros un parole ir sajaukta (*hash*). *Login* būs atslēga, bet parole attiecīgi - vērtība. Veicot pilnvarošanu, sistēma paroli varēs uzzināt pēc unikālas atslēgas. Ja ir nepieciešams ātrs efektīvs risinājums un datu kvantums nav liels, tad labāk izmantot *key-value* DBVS;

Lielāki *key-value* datu bāzes plusi:

- ātrāks un efektīvāks par *SQL* "dziņiem", kurus grūtāk mērogot;
- labāk apstrādā paralēlus pieprasījumus;
- glabājot mazus datu pārus aizņem mazāk vietas nekā RDBMS;
- neizmanto *SQL* vaicājumus, indeksus, trigerus, saglabātas procedūras, pagaidu tabulas. skatus utt.;
- *key-value* DBVS viegli mērogot un ar augstu efektivitātes līmeni, dēļ savas vienkāršības.

Kolonu-orientētas – datu bāzes, kas glabā datus tabulas, balstoties uz kolonnām, nevis kā parastās relāciju datu bāzes balstoties uz rindām. Salīdzinājumā starp rindu orientētu un kolonnu orientētu sistēmu prasti attiecās uz pieprasījuma ātrumu un cik daudz ir nepieciešams izmantot cieto disku, lai sniegtu atbildi uz pieprasījumu. Šāda veida sistēmu, labāk izmantot sistēmās, kuras būs daudz jānolasa dati, piemērām, datu noliktavās, *OLAP* rīkos un klientu attiecību vadība (*CRM*) utt.;

Kolonu-orientēta tipa datu bāzēm var uzskatīt šādas datu bāzes: *Teradata*, *Sybase IQ*, *C-Store*, *MonetDB*, *EXASOL* un *Vertica*;

Lielāki kolonu-orientētas datu bāzes plusi:

- kolonnu orientētas sistēmas ir daudz efektīvākas, kad ir jāiegūst dati no daudzām rindām, bet tikai no dažām kolonnām, tāpēc ka lasīt attiecīgas kolonnas ir daudz ātrāk nekā lasīt visus tabulu;
- kolonnu orientētas sistēmas ir efektīvākas, ja jaunās vērtības kolonnā tiek ievietotas visām rindām uzreiz, jo netiek mainīti un aiztikti iepriekšējie kolonnu dati;

- nav domāts, kad jāmaina atsevišķu rindu visi lauki;
- tiek izmantoti dažādi datu saspiešanas mehānismi, lai aizņemtu mazāk vietas uz cietā diska.

- Dokumentu orientētas – datu bāzes domātas darbā ar dokumentu orientētām sistēmām vai programmām. Šāda veida datu bāzes galvenais objekts ir dokuments, kurs sevī ietver pārus atslēga un vērtība, kas ir sasaistīti sava starpā. Tāds dokuments datu bāze parasti glabājas *JSON* vai *XML* formātā. Katram dokumenta ir sava unikāla atslēga, ko sauc par URI. Datus no servera parasti iegūst, padot ar *HTTP* pieprasījumiem, tādiem kā *GET, POST, PUT, DELETE*. Atšķirībā no relāciju datu bāzē, kur katrs ieraksts būtu tāds pats komplekts no laukiem un vērtībām, tomēr, ja vērtības nav, tad ieraksts jāveic ar *null* vērtību. Turpretī dokumentu orientētā datu bāzē vērtības var izlaist, jo nav jēgas glabāt atslēgu, kurai nav vērtības. Pēdējā laikā dokumentu orientētās datu bāzes ir kļuvušas populāras globālajā timekļi izstrādes lietotnēs, tīmekļa lapās, kuras ir jāveic meklēšana, kurai jābūt ļoti ātrai, piemēram, meklējot sociālos tīklos kādu cilvēku, tieši *Facebook* meklētos datus par cilvēkiem atgriež *JSON* formātā, kurš tiek apstrādāts ar *JavaScriptu* un atrādīts lietotājam saprotamā veidā;

  Par dokumentu orientēta tipa datu bāzēm var uzskatīt šādas datu bāzes: *MongoDB, Clusterpoint, SisoDB, ThruDB, OrientDB* un *Terrastore*.

2.5. NoSQL datu bāzes

Pagaidām ir vairāk nekā 10 populāras *NoSQL* datu bāzes, tomēr maģistra darbā aplūkotas tikai autoram zināmas datu bāzes un tās datu bāzes par, kurām globālajā tīmekli var atrast vairāk informācijas un piemērus. Aplūkosim pāris no tām:

- *MongoDB* – atvērta koda dokumentu orientēta datu bāze. 2009. gada tika izlaista pirmā versija, tagad aktuāla versija ir 2.2.2. Dati tiek glabāti kā dokumenti ar *JSON* līdzīgu formātu, tikai binārā veidā *BSON*. Šī datu bāze neizmanto struktūrās jeb ir bez struktūrām, kas atvieglo datu glabāšu. Šobrīd *MongoDB* ir vispopulārākā *NoSQL* datu bāzes pārvaldības sistēma. Datu bāze ir uzrakstīta, izmantojot *C++* programmēšanas valodu. Attēlā 2.5. ir attēlots *MongoDB* logo;

2.5.att. **mongoDB logo**

- *Clusterpoint* – augstas veiktspējas, bez struktūras dokumentu orientēta datu bāze. Datu bāze ir gan komerciāls, gan brīvs produkts, atkarība kādām vajadzībām tas tiek lietots. 2006. gada tika izlaista pirmā versija, tagad aktuāla versija ir 2.0.3. Datu bāze pārvalda *XML* dokumentu kolekcijas, kas tiek glabāta *XML* līdzīgā datu formātā. Datu bāze glabā datus dabiski cilvēk saprotamā veidā, kas dod iespēju viegli saprast datus, ar kuriem tiek strādāts. Datu bāzes saturs tiek automātiski indeksēts pat priekš nestrukturētiem datiem. *Clusterpoint* lielākais pluss ir ātra meklēšana lielos tekstos. Datu bāze ir uzrakstīta, izmantojot *C++* programmēšanas valodu. Attēlā 2.6. ir attēlots *Clusterpoint* logo.

2.6.att. **Clusterpoint logo**

- *FatDB* – atslēga-vērtība jeb *key-value* kā arī dokument orientēta datu bāze vienlaicīgi. Līdz ar to izvēlēties datu bāzes struktūru priekš izstrādātas lietojumprogrammas nav sarežģīti. Datu bāze ir komerciāls produkts. *FatDB* ir pieejama no 2012. gada. Šī datu bāze ļoti labi piemērotā mobilām ierīcēm. Datu bāze ir integrēta iekš *.NET* un *Windows Communication Foundation*. Pagaidām *FatDB* ir pieejams tikai priekš *Windows* operētai sistēmām. Attēlā 2.7. ir attēlots *FatDB* logo.

2.7.att. **FatDB logo**

2.6. SQL un NoSQL atšķirības

SQL koncepts ir datu relācija. Par relāciju var nosaukt datu bāzes tabulu, kur glabājas dati. Relāciju datu bāze atlasa datus pēc noteiktas datu kopas. Rezultāti tiek atgriezti strukturētu pieprasījumu datu veidā. Tabulas ir sadalīta pa rindām un kolonām. Lai atgrieztu tabulas rindu(as) tiek izmantots pieprasījums. Tabulas var glabāt datus ar iepriekš definētu datu tipu vai struktūru. Katra tabula satur vienu vai vairākas strukturētas kolonnas. Katra rinda satur unikālus datus, ko definē kolonnas struktūra. Lietotājs var piekļūt tabulu datiem nezinot tabulas iekšējo struktūru. Mērogojamība ir sarežģīta, citreiz pat neiespējam. It īpaši, kad vajag vienlaicīgi manipulēt ar datiem uz diviem dažādiem serveriem vai kad ir vajadzība pārnest datu bāzi uz citu serveri. Iegūt datus no vairākām tabulām arī mēdz būt ļoti sarežģīts process, ir bijuši gadījumi, kad vienu pieprasīju nākas rakstīt vairākas stundas.

Savukārt *NoSQL* pieejā nav vajadzīga datu struktūra. Attiecīgi tas nozīmē, ka datus var ievietot datu bāze pirms tam nedefinējot datu struktūru. Respektīvi, jebkurā brīdi ir iespējams sākt ievietot jaunus datus vai pat izmainīt datu formātu, bez aizkaves skatoties no biznesa viedokļa. Šī iespēja automātiski padara *NoSQL* risinājumus daudz elastīgākus. Izmantojot *NoSQL* mākoņlietotnēs, dati automātiski tiek glabāti pa vairākiem serveriem. Šos serverus var gan pielikt, gan noņemt, neapstādinot pašu lietotni. *NoSQL* izmanto integrētu kešatmiņu, kura datus saglabā sistēmas atmiņā.

Tiek uzskatīts, ka *NoSQL* datu bāzes:

- apstrādā datus daudz ātrāk par relāciju datu bāzēm;
- ir ātrākas par relāciju datu bāzēm tāpēc ka datu modelis ir vienkāršāks;
- ir elastīgākas, tās ir vieglāk mainīt un pārveidot izstrādes laikā, ka āri pēc tam;
- ir vieglāk mērogot.

2.7. NoSQL datu bāze MongoDB

Pēc iepazīšanas ar datu bāzēm, izvēle krita uz *NoSQL MongoDB* pārvaldības sistēmu. *MongoDB* projekts ir pieejams pēc saites http://www.mongodb.org. Šajā vietnē ir pieejama visa svaigākā un jaunākā informācija par *MongoDB* iespējām un attīstību. Viens no plusiem, kāpēc tika izvēlēts tieši *MongoDB*, ir ne tikai augsta veiktspēja un viegla mērogojamība, bet arī iespēja sadarboties ar *PHP* valodu, kura ir pieejama uz izstrādes servera. *MongoDB* ir atvērta koda dokument orientēta datu bāze, kas ļauj bez papildus naudas līdzekļiem izmantot,

to komerciālos projektos. Dažādas iespējas, ko sniedz *MongoDB* gan mācīšanas nolūkos, gan izmantojot projektu izstrādē:

- viegli iemācīties, tas ir šo sistēmu apgūt ir vieglāk par citām *NoSQL* sistēmām. Kolonu-orientētās datu bāzes sistēmas ievieš radikāli jaunas idejas, kuras izstrādātajiem, lai saprastu vajag ļoti iedziļināties dokumentācijā. Tomēr *MongoDB* ir līdzīgi pamatjēdzieni kā relāciju datu bāzēs, kas atvieglina to mācīšanos. Tiem, kas visu laiku ir izmantojuši relāciju datu bāzes iespējams būs mazliet problēmas pielāgojoties *MongoDB*;
- izmanto elastīgu datu struktūru (nestrukturēti dati). Tas nozīme pirms sākt ķerties pie datu glabāšanas datu bāzē nav nepieciešams definēt datu struktūru, kas ļauj viegli glabāt nestrukturētus datus;
- plaši mērogojama datu bāzē. *MongoDB* ir iebūvēta lieliska funkcija, kas nodrošina optimālu datu bāzes veiktspēju, kad datu kvantums un plūsma aug. Šādās situācijās prasti nevajag veikt pārmaiņas lietotnes līmenī.

MongoDB pamatjēdzieni – datu bāze, kolekcija un dokuments:

- *MongoDB* var uzskatīt par serveri uz kura vienlaicīgi var atrasties vairākas datu bāzes. Datu bāzes darbojas autonomi kā datu konteineri, kas nav atkarīgi viens no otra. Datu bāze sevi iekļauj vienu vai vairākas kolekcijas. Piemēram, datu bāzē priekš ziņu portālā varētu iekļaut šādas kolekcijas: raksti, autori, komentāri, kategorijas utt.;
- Kolekcija ir dokumentu kopums. Analoģiski relāciju datu bāzē kolekcija ir tā pati tabula. Bet atšķirībā no tabulām nav nepieciešams definēt struktūru pirms datu saglabāšanas kolekcijā;
- Dokuments ir saglabāta datu vienība kolekcijā. Dokuments satur lauku pārus vai atslēga-vērtība pārus. Atslēgas vērtība ir simbolu virkne, bet vērtības tips varbūt: simbolu virkne, vesels skaitlis, peldošais skaitlis, laiks, datums utt. Tik pat labi kā vērtību var izmantot citu dokumentu;

MongoDB dokuments ir redzams attēla 2.8.1.


```
{
   "_id": ObjectId("518cfba36803fa7e2e000001"),
   "uid": NumberInt(325453),
   "avatar": "http:\/\/i3.ifrype.com\/profile\/325\/453\/v1367418712\/i_325453.jpg",
   "name": "Askolds",
   "surname": "Molotoks",
   "date": "2013-05-10 16:52:35"
}
```


2.8.1.att. **MongoDB dokuments**

Relāciju datu bāzes ieraksts, kas ir ekvivalents ar *MongoDB* dokumentu no 2.8.1 attēla ir redzams attēla 2.8.2..

id	uid	avatar	name	surname	date
1	325453	http://i3.ifrype.com/profile/325/453/v1364998760/i...	Askolds	Molotoks	2013-04-25 17:32:44

2.8.2.att. **MySQL datu bāzes ieraksts**

Attēlā 2.8. redzamais dokuments satur septiņus laukus. Struktūra ir ļoti līdzīga *JSON* sintaksei. Pirmā lauka "_id" vērtība automātiski ģenerēta unikāla simbolu virkne. *MongoDB* automātiski piešķir katram dokumentam "_id", tomēr to var arī noņemt pēc vajadzības. Tā kā uzģenerētā simbolu virkne vienmēr būs unikāla tas nozīmē ,kā otra dokumenta ar datu pašu unikālu identifikatoru kolekcijas ietvaros nebūs. Otrais lauks "uid" ir vesels skaitlis, kas apzīmē lietotāja identifikācijas numuru sociālajā tīklā. Lauki "avatar", "name", "surname" un "date" satur simboliskas virknes, kas attiecīgi glabā informāciju par lietotāja bildi, vārdu, uzvārdu un reģistrācijas laiku.

Tabulā 2.1. ir *SQL* un *MongoDB NoSQL* atbilstošo pamatjēdzienu saraksts [11].

2.1. tabula

SQL un MongoDB NoSQL pamatjēdzienu saraksts.

***SQL* term**	***SQL* pamatjēdziens**	***NoSQL* pamatjēdziens**
database	datu bāze	datu bāze
table	tabula	kolekcija
row	rinda	dokuments vai objekts
column	kolonna	lauks
index	indekss	indekss
table joins	tabulas relācijas	iegults dokuments vai saite
primary key	primāra atslēga	primāra atslēga, kas *MongoDB* ir automātiski ģenerēts lauks *_id*
aggregation	apkopošana (grupēšana, summas iegūšanā, vidējas vērtības aprēķins utt.)	apkopošanas ietvars *MapReduce*

MongoDB atbalsta visus *JSON* datu tipus, kā arī uztur speciālus sistēmas datu tipus. Sarakta ir aprakstīti iespējamie *MongoDB* datu tipi [12]:

- *null* – bez vērtības, var tikt izmantos reprezentējot bez vērtības vērtību vai neesošu lauku;

```
{
   "_id": ObjectId("519372426803fa316a000012"),
   "tukša_vērtība": null
}
```

- *boolean* – Bula loģikas tips, kas varbūt patiess (*true*) vai nepatiess (*false*);

```
{
   "_id": ObjectId("519373646803fadd6500005a"),
   "Bula_vērtība": true
}
```

- *32-bit integer* –32 bitu vesels skaitlis fiziski netiek attēlots, jo *JavaScript* nodrošina darbu tikai priekš 64 bitu peldošiem skaitļiem, tas nozīme ka saglabājot veselu skaitlī, tas tiek konvertēts uz peldošu skaitli;
- *64-bit integer* – 64 bitu vesels skaitlis fiziski netiek attēlots, tāpāt kā 32 bitu vesels skaitlis;
- *64-bit floating point number* – skaitlis ar peldošu komatu;

```
{
   "_id": ObjectId("51937f2f6803fa316a0000b5"),
   "skaitlis_1": 3.14,
   "skaitlis_2": 3
}
```

- *string* – simbolu virkne *UTF-8* kodējuma. Jāpiebilst, ka viens simbols automātiski tiek konvertēts uz simbolu virkni;

```
{
   "_id": ObjectId("5193801f6803fa7d0a000030"),
   "string": "Simbolu virkne"
}
```

- *object id* – dokumenta unikāls identifikators, kas aizņem 12 bitus;

```
{
   "_id": ObjectId("5192a3236803fabe7000004f")
}
```

- *date* – datums, tiek glabāts milisekundes no unix ēras sākuma. Laika joslas netiek glabātas;

```
{
   "_id": ObjectId("519383956803fa7d0a0000a6"),
   "datums": ISODate("2013-05-15T12:46:13.347Z")
}
```

- *regular exoression* – regulāra izteiksme, glabā regulāras izteiksmes izmantojot *JavaScript* sintaksi;
- *code* – kods, var saglabāt *JavaScript* kodu;

```
{
   "_id": ObjectId("5193888c6803fa56140000ba"),
   "kods": function () {
     /* anonīma funkcija */
   }
}
```

- *binary data* – binārs datu tips;
- *maximum value* – maksimāli iespējama *BSON* mainīga vērtība;
- *minimum value* – minimāli iespējama *BSON* mainīga vērtība;
- *undefined* – nedefinēts mainīga tips vai uzskatīt arī par *null* vērtību;

```
{
   "_id": ObjectId("519387906803fa5614000081"),
   "nedefinēts": undefined
}
```

- *array* – masīvs, mainīgo kopums tiek glabāts masīvos;

```
{
   "_id": ObjectId("5193874c6803fa561400007c"),
   "masīvs": {
     "0": 1,
     "1": 2,
     "2": "a",
     "3": "b"
   }
}
```

- *embedded document* – iegults dokuments, dokumentā var glabāt citu dokumentu.

```
{
   "_id": ObjectId("519386196803fa5614000066"),
   "iegults_dokuments": {
     "dati": "dati iekšā"
   }
}
```

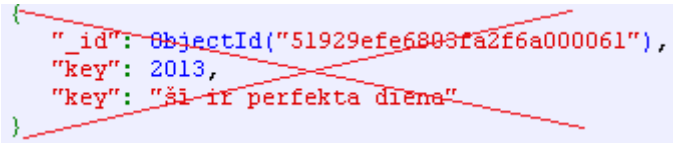
MongoDB ir reģistrjutīga (attēls 2.9.) un datjūtīga (attēls 2.10.) datu bāze. Veidojot atslēgas ir jāpieturas pie dažiem principiem. Pirmkārt, atslēgas vārds nevar saturēt virknes beigu simbolu "\0". Otrkārt, . un $ ir jāizmanto tikai speciālos gadījumos. Jāpiebilst, kā dokuments nevar saturēt divas identiskas atslēgas, viena dokumenta ietvaros, kas ir redzams attēlā 2.11..

```
{
   "_id": ObjectId("51929d596803fa2f6a00004c"),
   "atslēga": "vērtība",
   "Atslēga": "vērtība"
}
```

2.9.att. **Dokuments ar dažādām atslēgām**

```
{
   "_id": ObjectId("51929efe6803fa2f6a000062"),
   "key_1": 3,
   "key_2": "3"
}
```

2.10.att. **Dokuments ar dažādiem datu tipiem**



2.11.att. **Neeksistējošs dokuments ar vienādām atslēgām**

Tā kā *MongoDB* dokumenta dati varbūt nestrukturēti, piemērām, attēla 2.12. ir redzami divi dažādi dokumenti, kas ir saglabāti vienā kolekcijā. Tomēr balstoties uz autora pieredzi glabāt dažāda veida strukturētus datus viena kolekcijā nav optimāli, to apstrāde paliek grūta un neloģiska.

```
#1 {
     "_id": ObjectId("5192a3236803fabe7000004f"),
     "virkne": "Teksts varbūt jebkurš UTF-8 simbols !"
   }

#2 {
     "_id": ObjectId("5192a3596803fabe70000050"),
     "Pi": 3.14159265359
   }
```

2.12.att. **Atšķirīgas struktūras dokumenti**

Ir labi redzams ka dokumenta struktūras sintakse ir *JSON* viedīga. Tomēr, kad dokuments tiek saglabāts datu bāze, tas tiek sarealizēts speciāli bināri kodēta formātā *BSON*. *BSON* pēc noklusējuma skaitās *MongoDB* datu apmaiņas formāts. *BSON* galvenā priekšrocība ir tā, ka to lietot ir efektīvāk nekā parastus formātus, piemēram, *XML* vai *JSON*, jo tas aizņem mazāk atmiņas un apstrādes laiks ir ātrāks. *BSON* datu tips atbalsta visus datu tipus, ko atbalsta *JSON* (simbolu virknes, veselus skaitļus, peldošus skaitļus, loģiskus tipus, masīvus, objektus, null vērtības) plus tam atbalsta dažus speciālus datu tipus kā regulāras izteiksmes, objekta identifikatoru, datumus, binārus datus un kodu. Programmēšanas valodas kā *PHP*, *Python*, *Java* satur bibliotēkas, kas konvertē valodas specifiskas datu struktūras, piemēram, asociatīvus masīvus priekš *BSON* un atpakaļ uz sākotnējo programmēšanas valodu. Šāda programmēšanas valodas iespēja dod vieglu datu mijiedarbību ar *MongoDB* un atpakaļ.

Kolekcijas var atšķirt pēc to nosaukumiem. Kolekcijās nosaukums varbūt jebkurš *UTF-8* simbols ar pāris nosacījumiem:

- kolekcijas nosaukums nevar būt tukša simbolu virkne;
- kolekcijas nosaukums nevar saturēt simbolu virknes beigu simbolu "\0";
- nav ieteicams veidot kolekcijas, kurās nosaukuma ir vārds "*system.*", jo šāds vārds ir rezervēts priekš paša *MongoDB* datu bāzes kolekcijām, piemērām, *system.users*, kas glabā *MongoDB* lietotāju datus;
- nav ieteicams izmantot $ simbolu iekš kolekcijas nosaukuma, jo dažas datu bāzes kolekcijas satur šādu simbolu un pirmkārt ir domāti priekš sistēmas.

Dokumenti tiek grupēti kolekcijās, bet kolekcijas ir grupētas datu bāzēs. Viena *MongoDB* instance var saturēt vairākas datu bāzes, kas pilnība ir savstarpēji neatkarīgas. Datu bāzei ir savas tiesības, tā tiek glabāta pa dažādiem failiem uz diska. Laba prakse ir glabāt vienas lietotnes datus tikai vienā datu bāzē. Gluži kā kolekcijas datu bāzes identificē pēc nosaukumiem. Datu bāzes nosaukums varbūt jebkurš *UTF-8* simbols ar pāris nosacījumiem:

- datu bāzes nosaukums nevar būt tukša simbolu virkne;
- datu bāzes nosaukums nevar saturēt simbolus: tukšumu (atstarpi), punktu, $, /, \ vai virknes beigu simbolu "\0";
- datu bāzes nosaukums jābūt ar maziem burtiem;
- datu bāzes nosaukuma garums ir limitēts uz 64 baitiem.

Vērts atcerēties datu bāze ir parasti faili, kas atrodas uz cietā diska. Tieši tāpēc, ir izskaidrojami tik daudz nosacījumi pie datu bāzes nosaukumu izveides. Bez šiem nosacījumiem vēl ir trīs rezervēti datu bāzes nosaukumi: *admin, local* un *config*.

Tabulā 2.2. ir *SQL* un *MongoDB NoSQL* vaicājumu paraugi, lai saprastu sintaksi un būtu vieglāk pāriet no vienas vaicājuma valodas uz otru.

2.2. tabula

SQL un MongoDB NoSQL vaicājumu paraugi.

***SQL* vaicājums**	***NoSQL* vaicājums**
Tabulas struktūras izveides vaicājums	

`CREATE TABLE users (`
`  id MEDIUMINT NOT NULL`
`  AUTO_INCREMENT,`
`  user_id Varchar(30),`
`  age Number,`
`  status char(1),`
`  PRIMARY KEY (id)`
`)`
	`db.users.insert( {`
	`  user_id: "abc123",`
	`  age: 55,`
	`  status: "A"`
	`} )`

Tabulas dzēšanas vaicājums	
`DROP TABLE users`	`db.users.drop()`
Tabulas visu datu atrādīšanas vaicājums	
`SELECT * FROM users`	`db.users.find()`
Tabulas ieraksta izmaiņas vaicājums	

`UPDATE users SET status = "C"`
`WHERE age > 25`
	`db.users.update(`
	`  { age: { $gt: 25 } },`
	`  { $set: { status: "C" } },`
	`  { multi: true }`
	`)`

Tabulas ieraksta dzēšanas vaicājums	
`DELETE FROM users WHERE status = "D"`	`db.users.remove( { status: "D" } )`

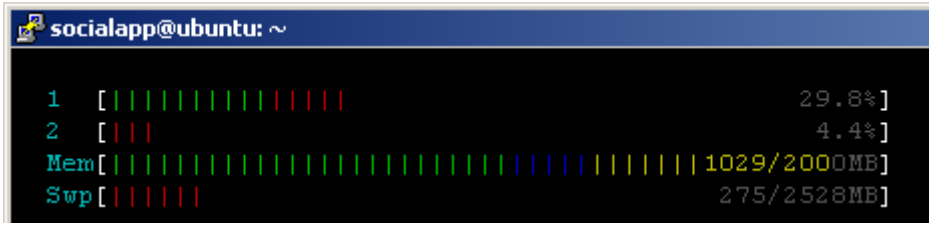
3. PROBLĒMAS APRAKSTS

Autora darbs ir saistīts ar sociālo tīklu lietotņu un portālu izveidi. Parasti uz servera vienlaicīgi darbojās vairāki projekti. Projekti varbūt dažādi konkursi, kas var ilgt no dienas līdz pat mēnesim, tīmekļa spēles vai portāli, kurus dienā apmeklē daudz lietotāji. Lietotāju skaitam strauji pieaugot, piemēram, kad ir palaista jauna lietotne, kur lietotājam ir jāizveido komanda uzaicinot savus draugus. Šādas lietotnes ātri kļūst populāras un pieprasījumu skaits strauji palielinās, līdz ar to lietotnes ielādē paliek ilgāka un serveris sāk strādāt lēnāk un pat vairākas reizes ir pilnība apstājies apturot visu lietotņu pieejamību. Protams, var likties kā tā nav liela problēma un to var atrisināt palielinot servera jaudu tas ir izīrējot spēcīgāku serveri, tomēr naudas resursi ir ierobežoti un ir jāatrod problēmas risinājums izmantojot esošo serveri.

3.1. Serveris

Lietotnes tiek izstrādātas uz virtuāla servera, kas atvieglo vairāku lietotņu vienlaicīgu uzturēšanu un izstrādi programmētājiem. Tā kā izstrādātas lietotnes un portāli strādā vienlaicīgi servera veiktspēja samazinās atkarība no tā cik daudz ir aktīvo lietotņu un portālu. Bez datu bāzēm serveris ir noslogots ar failu augšuplādi, dokumentu ģenerēšanu un bilžu apstrādi.

Uz virtuālais servera ir uzinstalēta Ubuntu 10.04 operētājsistēmā. Serverim ir divi Intel Xeon E5506 procesori. Katram procesoram ir četri kodoli un to takts frekvence ir 2.13GHz. Viena no lielākam problēmām ir servera brīvpiekļuves atmiņas trūkums. Tagad uz servera ir tikai 2 gigabaitu atmiņas, kas ir ļoti maz priekš tik daudzām lietotnēm. Attēla 3.1. ir redzams procesoru un atmiņas noslodzē.



3.1.att. **Procesoru un atmiņas noslodzē**

3.2. SQL datu bāze MySQL

Pārsvara autors izstrādājot lietotnes un portālus izmanto *SQL* tipa *MySQL* 5.1. datu bāzi. Izmantojot *MySQL* datu bāzi, autors jau ir izstrādājis vismaz 100 lietotnes priekš *draugiem.lv* un *Facebook* sociāliem tīkliem. *MySQL* datu bāzēs glabājas dati par lietotāju, konkursa rezultāti, lietotāju darbības un cita vajadzīga informācija priekš lietotnēm. Lietotnes atspoguļojās lietotāju dati, veiktās darbības, tiek ģenerēti topi ar rezultātiem, statistika un citas lietotnē paredzētas lietas.

Tā kā palielinās lietotņu un lietotāju skaits, aizvien biežāk tiek novērots kā *MySQL* nespēj ātri apstrādāt lietotāju pieprasījumus, līdz ar to lapas atbildes (ielādes) laiks palielinās. Kad datu bāzē nevar apstrādāt vienlaicīgi daudz pieprasījumu, tie tiek kārtoti rindā un lietotājam ir jāgaida, kad serveris un datu bāzē apstrādās viņa pieprasījumu. Līdz ar to lapas ielādes laiks ir pārāk liels un lietotāji kļūst neapmierināti.

Pēc vairāku lietotāju sūdzību izskatīšanas tika veikta datu bāzes stresa testēšana, kuras laikā atklājas, ka sistēmas vāja vieta ir datu bāzes pieprasījumu apstrādes laiks. Vairākas situācijas pieprasījumu apstrādes laiks ilga vairākas minūtes. Tāpēc tika meklētas alternatīvas datu bāzes.

4. LIETOTNES APRAKSTS

Maģistra darba uzdevums ir izveidot lietotni priekš sociāliem tīkliem, kas izmanto *NoSQL* datu bāzi jeb lietotni, kas visus datus glabā datu bāze neizmantojot relācijas. Šajā nodaļā ir aprakstīts lietotnes vispārējs raksturojums. Funkcionālas un nefunkcionālas prasības. Plašāka lietotnes tehniskā izstrādes detalizācija sekos nākamajās nodaļās.

4.1. Lietotnes vispārīgs raksturojums

Lietotnei jābūt integrētai sociālos tīklos un jāspēj datus glabāt *NoSQL MongoDB* datu bāzē. Lietotnes izstrādei ir vairāki mērķi:

- biznesa mērķis, popularizēt lietotnes piedāvātos produktus;
- piesaistīt biznesa lapai (vietnei) vairāk fanus;
- pārbaudīt praksē cik ilgi aizņems apgūt un izstrādāt lietotni uz *MongoDB*;
- sākumā izveidot analogu lietotni, kas datus glabā relāciju *MySQL* datu bāzē.

Lietotnes pamatideja ir spēle jeb konkurss, kur lietotāji sacenšas sava starpā, kurš vairāk un ātrāk iegūs punktus. Spēles laikā no laukuma augšas uz leju pamazām slīdēs dažādi produkti no dažādām preču kategorijām. Laukuma apakšā ir novietoti 3 preču kategoriju maisi, katrs ir domāts savas produktam. Spēles uzdevums ir ievirzīt katru produktu tam paredzētajā maisā. Par katru pariezi ieliktu produktu tiek saņemts 10 punkti. Jā produkts nav ielikts pareizā maisā vai neievietos maisā vispār, tad spēle beidzās un tiek paziņots spēlētajā rezultāts. Spēlēs laikā produkti pamazām sāk krist arvien ātrāk. Iegūto rezultātu drīkst uzlabot spēlējot spēli atkārtoti. Attēla 4.1. ir redzama spēles saskarne.

4.1.att. **Spēles saskarne**

Pati spēle ir uzrakstīta uz *ActionScript 3.0* programmēšanas valodas un izveidota izmantojot *Flash CS6* izstrādes rīku. Spēles izstrāde nav aprakstīta maģistra darba, jo neatbilst maģistrā darba tēmai. Neskatoties uz to autors ir pilnība izstrādājis lietotni ieskaitot spēles elementu. Spēle nosuta datus par izspēlēto spēli uz serveri, kas apstrādā iegūtos datus un saglabā *MongoDB* datu bāzē.

4.2. Lietotnes funkcionālas prasības

Lietotnei jāstrādā divos sociālos tīklos *draugiem.lv* un *facebook.com* portālos. Lietotne *draugiem.lv* portālā ir latviešu valodā, bet *facebook.com* portālā ir lietuviešu valodā attiecīgi ir divas lietotnes versijas. Lietotnei ir jābūt šādai funkcionalitātei:

- lietotājam obligāti jākļūst par biznesa lapas fanu;
- pēc spēles beigām ir jābūt iespējai izvelēties jaunu spēli vai apskatīt topu;
- attēlot labākos top divpadsmit spēlētājus;
- attēlot lietotāja vietu topā;
- attēlot nedēļas labākos spēlētājus;
- attēlot balvas un noteikumu skatu;
- datus glabāt *MySQL* datu bāzē;
- datus glabāt *MongoDB* datu bāzē;
- izvadīt statistiku, kas redzamā tikai lietotnes administratoram.

4.3. Lietotnes nefunkcionālas prasības

Izstrādājot lietotni pēc iespējas vairāk ir jāizmanto atvērtā pirmkoda programmatūra. Tas attiecās gan uz izstrādēs vidi, gan datu bāzes apstrādes sistēmu un pašu datu bāzi. Lietotnes pirmkodam jābūt viegli lasāmam, lai to būtu viegli labot vai papildināt. Tā kā ir dažādas pārlūkprogrammas[13] (*Internet Explorer*, *Firefox*, *Chrome*, *Safari* un *Opera*) ir jānodrošina lietotnes saderīgumu ar tām, lai lietotāji varētu izmantot lietotni uz savas iemīļotākas pārlūkprogrammas. Lietotnei ir jābūt šādām nefunkcionālam prasībām:

- glabāt skripta izpildes laiku datu bāzē;
- paredzēt aizsargmehānismu pret spēles rezultāta viltošanu;
- izmantot *PHP* programmēšanas valodu.

5. LIETOTNES IZSTRĀDE

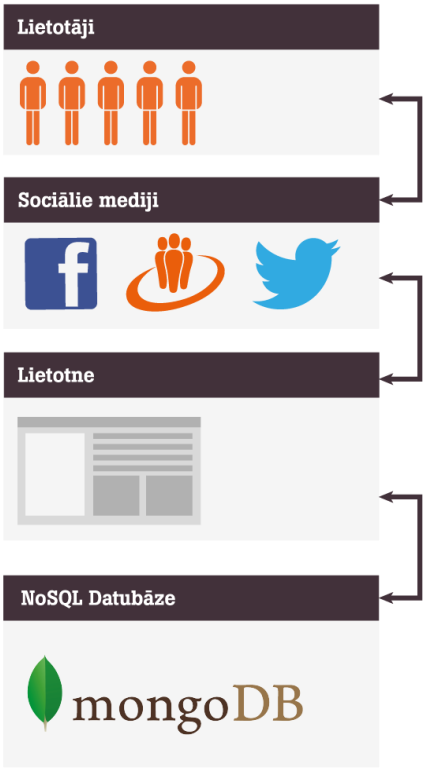
Šajā nodaļā ir aprakstīts lietotnes integrācijas algoritms sociālos tīklos, risinājuma tehnoloģijas, izstrādes vide un lietotnes administrēšanas iespēja. Vizuāli attēlots un aprakstīts datu modelis gan *MySQL*, gan *MongoDB* datu bāzei.

5.1. Lietotnes integrācijas algoritms

Veidojot lietotni, tas nosaukums sociālā tiklā ietvaros ir unikāls. Unikāls arī ir lietotnes numurs jeb *ID*. Katrai lietotnei ir sava *API* atslēga, kuru nedrīkst uzzināt lietotāji. *API* atslēga ir vajadzīga, lai lietotājs varētu pilnvaroties lietotnē un tiktu atpazīts no lietotnes un sociālā tīkla puses.

Lietotāji lieto sociālo tīklu, kurā ir ievietota lietotne. Jā lietotājs piekrīt savu datu padošanai lietotnei, tad lietotāja dati tiek saglabāti datu bāze. Ik reizi, kad lietotājs ienāk lietotnē tas tiek atpazīts kā lietotnes lietotājs un attiecīgi var lietot šo lietotni. Lietotnē izmantojot portāla *API* ir iespējams mijiedarboties ar lietotāja datiem, sūtīt vēstules, sūtīt atgādinājumus, ieteikt ziņas un daudz ko citu.

Lietotnes integrēšanas shēmas vizualizācija ir apskatāma attēlā 5.1.



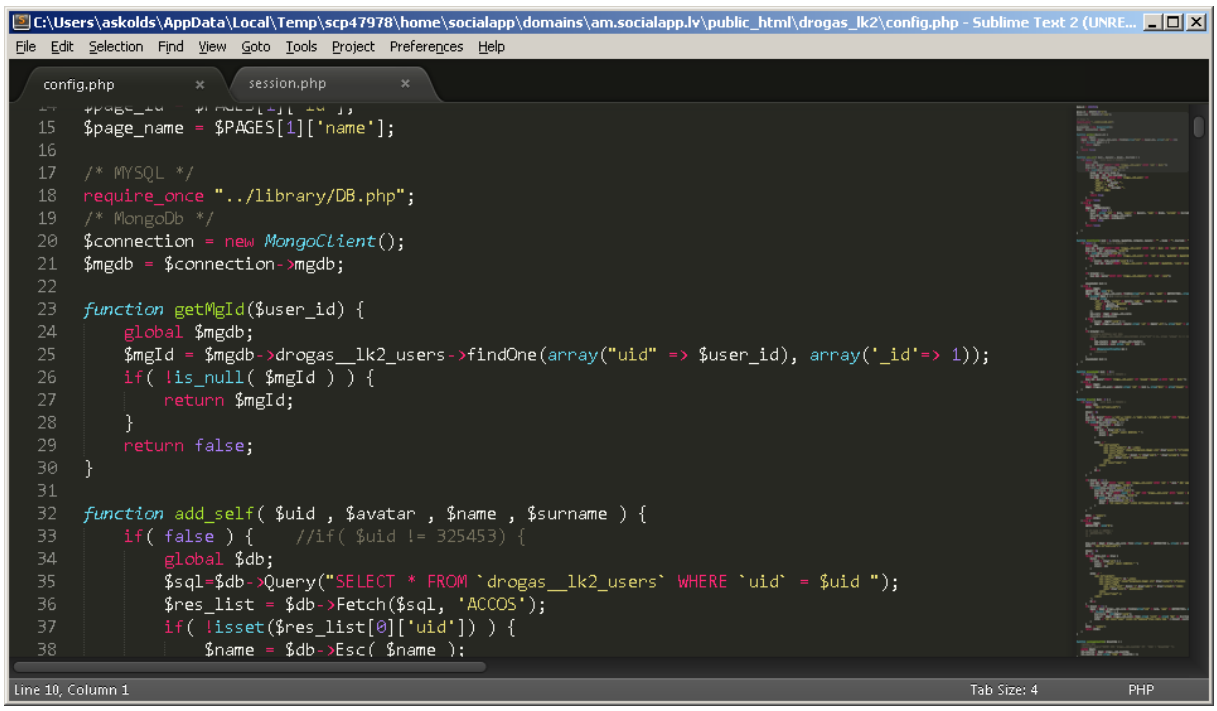
5.1.att. **Lietotnes integrēšanas shēma**

5.2. Risinājuma tehnoloģijas

Izstrādājot sociālo tīkla lietotni vajag izvēlēties tīmekļa programmēšanas valodu, kas spētu komunicēt ar serveri un dinamiski attēlot informāciju. Ir daudzas tīmekļa programmēšanas valodas: *ASP.NET*, *Python*, *Java*, *PHP* u.c. Visas šīs pieminētās valodas ir ļoti populāras, tomēr tika izvēlēta *PHP* programmēšanas valoda, jo izstrādes serveris atbalsta *PHP* valodu, izmantojot *Apache* serveri. *PHP* ir atklātā pirmkoda programmēšanas valoda, kas ir viens no galvenajiem plusiem. Ar *PHP* palīdzību tiks ģenerēts HTML kods, kas attēlosies tīmekļa pārlūkprogrammā. Komunikācija ar *MongoDB* ir izmantots *MongoDB* paplašinājums priekš *PHP*.

5.3. Izstrādes vide

PHP ir interpretatora veida programmēšanas valoda. Tas nozīmē, ka uzrakstīto programmas kodu nav vajadzības kompilēt. Programmas kods izpildās automātiski uz servera, katru reizi palaižot attiecīgo skriptu jeb programmas koda rindas. Līdz ar to, lai sāktu programmēt *PHP* valodā, ir vajadzīgs tikai teksta redaktors. Tomēr, lai būtu vieglāk rakstīt kodu un sintakse attēlotos krāsaini, tika izmantots *Sublime Text 2* ar *PHP* spraudni. *Sublime Text 2* automātiski piedāvā funkciju nosaukumus un liek atstarpes kodā, lai palielinātu koda lasāmību. Attēlā 5.2. ir redzams *Sublime Text 2* izmantošanas piemērs lietotnes izstrādes laikā.



5.2.att. **Sublime Text 2**

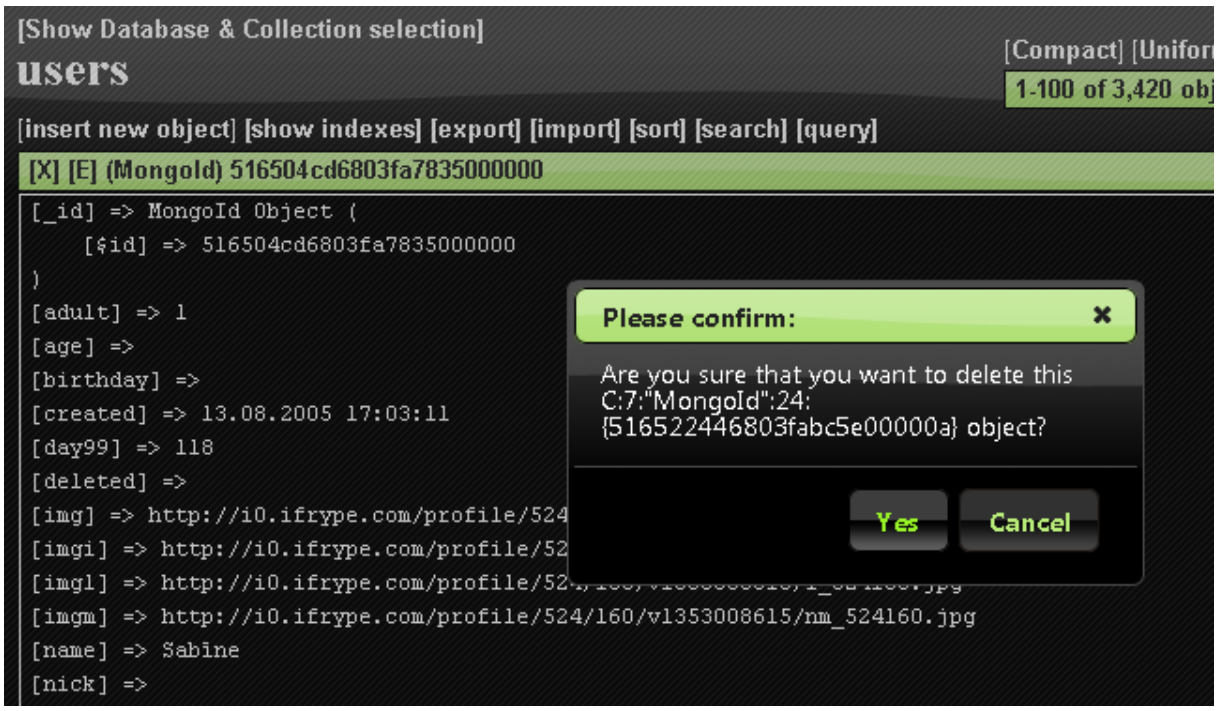
5.4. Lietotnes administrēšana

MySQL tiek administrēts ar *phpMyAdmin* [14] datu bāzes pārvaldības rīku, kas sniedz izstrādātājam dažādas iespējas, kas atvieglo izstrādes gaitu un dot administrēšanu iespēju.

MongoDB nepiedāvā datu bāzes administrēšanas rīku izņemot var veikt vaicājumus caur komandrindu, kas aizņem daudz laika un ir viegli pielaist kļūdu rakstot komandas un vaicājumus manuāli (ar roku).

Pirmais administrēšanas rīks, ko autors izpētīja ir *phpMoAdmin* [15]. Attēla 5.3. ir redzama *phpMoAdmin* rīka saskarne. *PhpMoAdmin* grafisks lietotāja saskarnes rīks, kas paredzēts *MongoDB* datu bāzes administrēšanai. Tas ir izstrādāts *PHP* programmēšanas valoda un strādā uz *Apache* servera. Šis rīks ir izstrādāts uz *Vork* ietvara. Viss rīks sastāv no viena faila un tā izmērs ir ap 100 kilobaitiem. *PhpMoAdmin* ir atvērtā pirmkoda programmatūra zem *GPLv3 FOSS* licences.

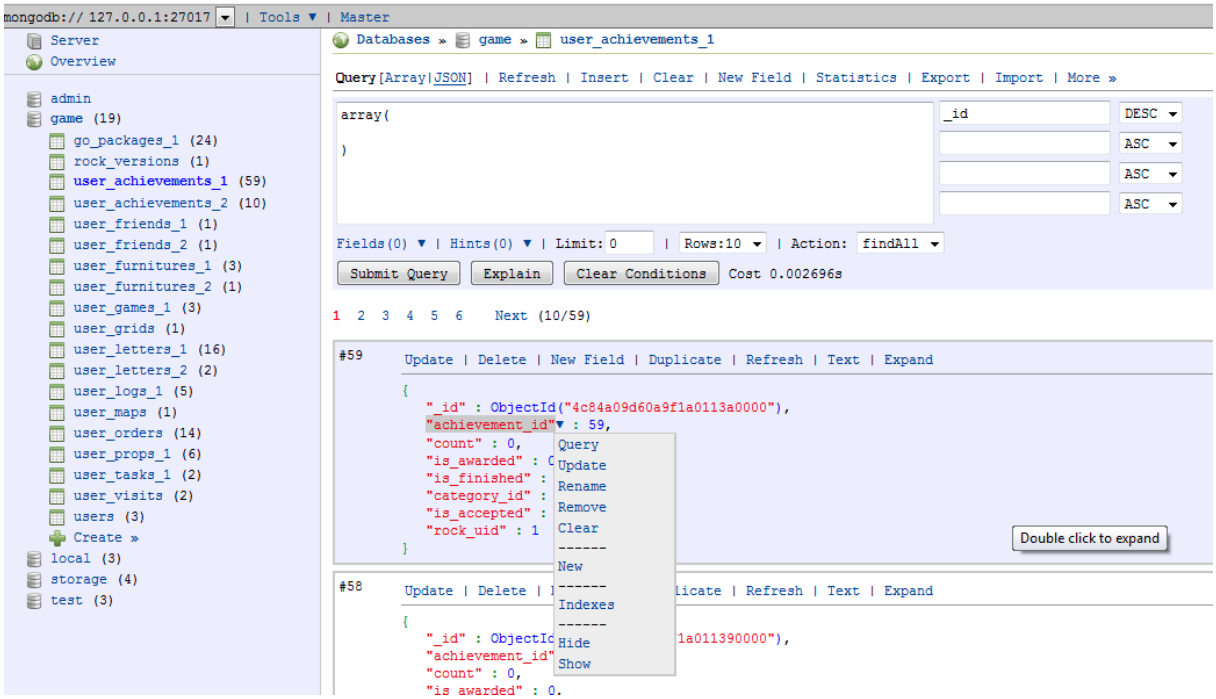
Pie daudziem ierakstiem šīs rīks nav tik pārskatāms, tā lietošana paliek apgrūtinātā. Veicot dažādus uzdevums saskarne mainās un paliek ne tik intuitīva, līdz ar to uzdevumu veikšana paliek sarežģītāka. Alternatīvs rīks ir *RockMongo* [16], kas ir ļoti līdzīgs *phpMyAdmin* rīkam, tikai priekš *MongoDB* datu bāzes administrēšanas.



5.3.att. **PhpMoAdmin**

Otrais administrēšanas rīks, ko izpētīja autors ir *RockMongo*. Attēla 5.4. ir redzama *RockMongo* rīka saskarne. *RockMongo* grafisks lietotāja saskarnes rīks, kas paredzēts *MongoDB* datu bāzes administrēšanai un vieglai pārlūkošanai. Tas ir izstrādāts *PHP 5* programmēšanas valoda. To var uzskatīt par tīmekļa lietotni, kas strādā uz *Apache* servera, kaut gan to var uzstādīt uz jebkura servera, kas atbalsta *PHP 5*. Šis rīks sastāv no vairākiem failiem, kas kopā aizņem ap vienu megabaitu vietas uz cieta diska. *RockMongo* ir atvērtā pirmkoda programmatūra zem *BSD* licences.

RockMongo rīka saskarne ir ļoti līdzīga *phpMyAdmin* rīka saskarnei, kas padara strādāšanu ar šo rīku ļoti intuitīvu. *RockMongo* rīkam var pielikt jaunus spraudņus, kas atvieglo pieprasījumu rakstīšanu un testēšanu.



*5.4.att.***RockMongo**

PhpMoAdmin un *RockMongo* abus rīkus var izmantot *MongoDB* administrēšanai, veidot jaunas datu bāzes un kolekcijas, veikt vaicājums, saglabāt un dzēst dokumentus, apskatīt statistiku un citas iespējas. Tā kā iespēju ir daudz, lai vieglāk saprast kādu rīku izmantot atkarība no vajadzīgā uzdevuma tabulā 5.1. ir šo rīku funkciju salīdzinājums.

Pēc abu rīku lietošanas un to salīdzināšanas var redzēt ka *RockMongo* vairāk paredzēts nopietnai administrēšanai, kad datu bāzes un kolekciju skaits ir liels. *RockMongo* funkcijas atļauj veikt vairāk datu manipulācijas iespēju nekā *PhpMoAdmin* rīkā un tā uztvere ir intuitīvāka. Tomēr, ja tiek strādāts pie mazas lietotnes izstrādes un administrēšanas, kad negribās daudz konfigurēt datu bāzi, bet ķerties pie ātrākas lietotnes izstrādes, tad noderēs *PhpMoAdmin* rīks.

Abiem rīki vēl ir izstrādes procesā un jaunas versijas tiek izstrādātas bieži, līdz ar to rīki tiek nepārtraukti uzlaboti.

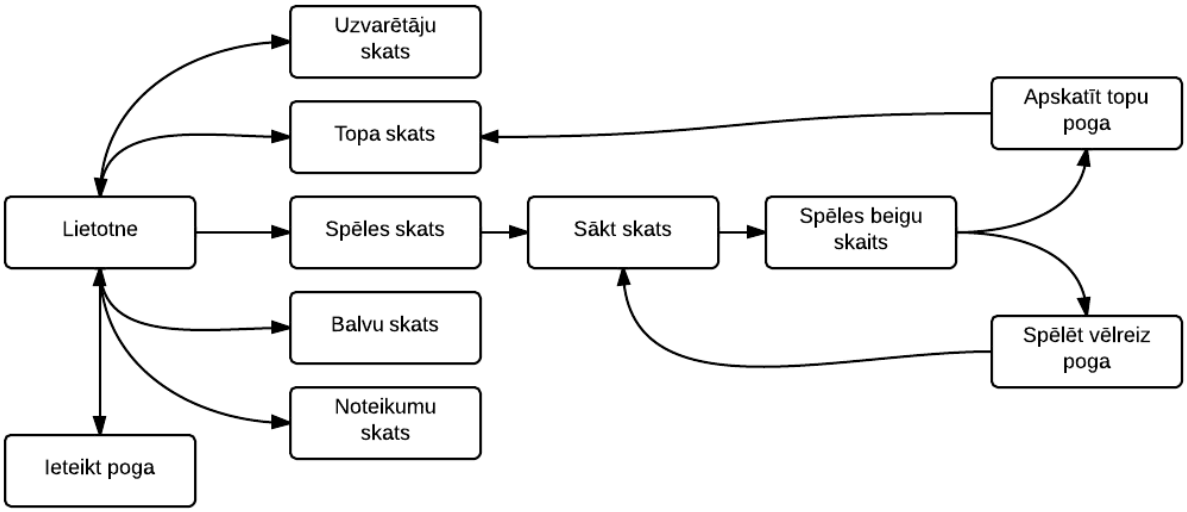
5.1. tabula

PhpMoAdmin un RockMongo funkciju salīdzinājums

Funkcija	***PhpMoAdmin***	***RockMongo***
Instalācija	Instalācija ir ļoti vienkārša. Augšupielādēt skripta failu uz servera un var sākt administrēt datu bāzi.	Augšupielādēt failus uz servera. Konfigurācijas failā uzlikt iespēju pilnvaroties sistēma ar paroli. Pēc tam var sākt lietot.
Konfigurācija	Pēc noklusējuma nav konfigurācijas. Lietotājs pats var izmanīt skriptu.	Lietotājs var izmanīt konfigurācijas iestatījums, kas atrodas config.php failā.
Lietotāja pilnvarošanā	Tikai lietotnes pilnvarošana. Nevar rediģēt *MongoDB* lietotājus.	Atbalsta gan lietotnes, gan *MongoDB* lietotāju pilnvarošanu. Ir iespēja rediģēt *MongoDB* lietotājus.
Saskarne	Saskarne nav intuitīva veicot dažādus uzdevumus.	Saskarne ir izveidotā līdzi kā *phpMyAdmin* rīka. Vaicājumu rezultāti tiek izvadīti saprotamā veida izmantojot dažādas krāsas palielinot dokumentu lasāmību.
Datu imports un eksports	Atbalsta datu importēšanu un eksportēšanu *JSON* formāta. Ir iespējams eksportēt vaicājuma rezultātus. Pie importēšanas ir iespēja atrisināt problēmu ar datu duplikāciju.	Atbalsta datu importēšanu un eksportēšanu *JSON* formāta. Pilna datu bāzes eksportēšana. Pie importēšanas veic datu duplikāciju.
Statistika	Atbalsta servera statistikas, bet tā ir grūti uztverama.	Atbalsta servera statistikas attēlošanu lietotājam viegli saprotamā veidā..

5.5. Lietotnes saskarne

Lietotnes saskarnei ir jābūt intuitīvi skaidrai, lai lietotājam būtu viegli mijiedarboties ar lietotni. Lietotnes augšēja daļā atrodas piecas izvēlnes (Spēle, Tops, Uzvarētāji, Balvas, Noteikumi) un poga ieteikt. Izvēlnes pogas ļauj pārslēgties starp lietotnes skatiem, kad attiecīgais skaits ir aktīvs izvēlnes poga ir iekrāsotā. Ieteikt poga ļauj lietotājam pastāstīt viņa draugiem par lietotni. Spēles skatā atrodas spēlē. Topa skatā ir lietotāju rezultātu attēlošana. Uzvarētāju skatā ir apskatāmi spēles uzvarētāji un ir iespēja pārslēgties starp nedēļas uzvarētājiem. Balvas un noteikumu skatā ir informācija ar balvām un spēles noteikumiem. Attēla 5.5. ir redzama skatu navigācijas iespējas.

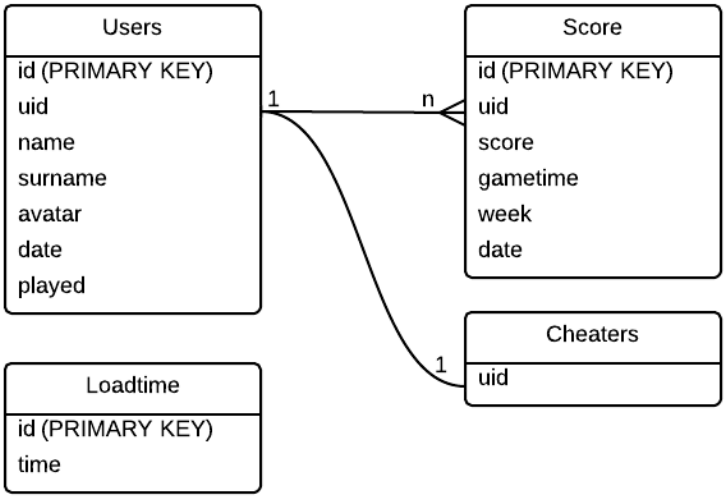


*5.5.att.***Skatu navigācija**

5.5. MySQL datu modelis

Lai realizētu visas lietotnes funkcionālas prasības entītiju relācijas modelī tika izveidotas četras entītijas, kas attēlotas 5.6. datu modelī – *Users, Score, Loadtime* un *Cheaters*. Entītija *Users*, kas sīkāk aprakstīta tabulā 5.2., tiek glabāta informācija par lietotnes lietotājiem un spēļu skaitu. Entītija *Score*, kas sīkāk aprakstīta tabulā 5.3., satur informāciju par lietotāju izspēlētām spēlēm – rezultātu un laiku. Entītija *Loadtime* satur datus par lietotnes ielādes laikiem. Tabulā 5.4. aprakstīta *Loadtime* struktūra. Entītija *Cheaters* satur informāciju par negodīgiem spēlētājiem. Tabulā 5.5. aprakstīta, tās struktūra.

Lietotni palaižot datu bāzes tabulas ir tukšas un aizpildās hronoloģiskā secībā, lai vieglāk veiktu atkļūdošanu.



*5.6.att.***MySQL ER modelis**

5.2. tabula

Entītija Users

Nosaukums	**Datu tips**	**Apraksts**
id	integer(10)	Primāra atslēga. Identifikators.
uid	big integer(20)	Unikāla atslēga. Lietotāja identifikācijas numurs sociālā tīmeklī.
name	varchar(50)	Lietotāja vārds.
surname	varchar(50)	Lietotāja uzvārds.
avatar	varchar(255)	Saite uz lietotāja bildi.
date	datetime	Datums un laiks, kad lietotājs ir reģistrējies lietotnē.

5.3. tabula

Entītija Score

Nosaukums	**Datu tips**	**Apraksts**
id	integer(10)	Primāra atslēga. Identifikators.
uid	big integer(20)	Unikāla atslēga. Lietotāja identifikācijas numurs sociālā tīmeklī.
score	integer(5)	Spēles rezultāts.
gametime	integer(10)	Spēles laiks sekundēs.
week	integer(5)	Gada nedēļas numurs.
date	datetime	Datums un laiks, kad lietotājs ir izspēlējis.

5.4. tabula

Entītija Loadtime

Nosaukums	Datu tips	Apraksts
id	integer(10)	Primāra atslēga. Identifikators.
time	float	Lietotnes ielādes laiks.

5.5. tabula

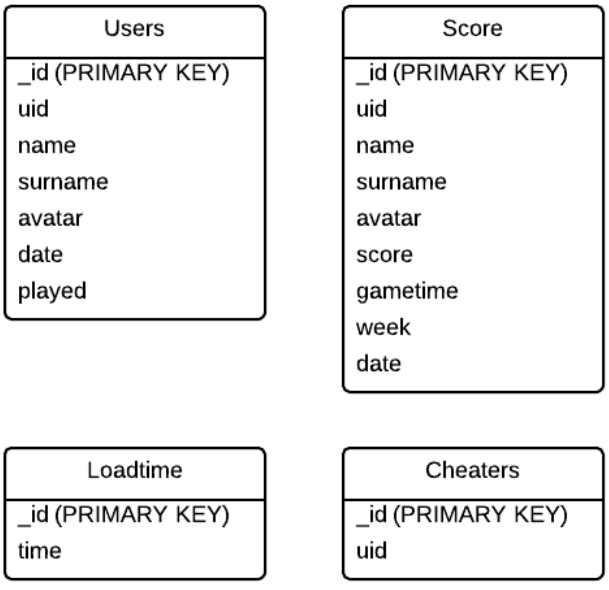
Entītija Cheaters

Nosaukums	Datu tips	Apraksts
uid	big integer(20)	Lietotāja identifikācijas numurs sociālā tīmeklī.

5.6. MongoDB datu modelis

Atkarība no risināmas problēmas *MongoDB* datus var glabāt normalizēta un denormalizēta veidā. Lietotāju rezultātu tops tiek ģenerēšanai tiešsaistē. Tas nozīme, ka to vajag darīt pēc iespējas ātri. Nolasīt no datu bāzes spēlētāju rezultātus, sakārtot tos un izvadīt, lai to viss ātrāk izdarītu dati datu bāzē ir jāglabā denormalizēta veidā. Protams, rodas datu duplikācija, tomēr šāda pieeja ir ātrāka nekā iegūt datus normalizētā veidā.

Datu modelis ir līdzīgs *MySQL* datu modelim, tomēr *MongoDB* datu modelī varēja iztikt bez kolekcijas *Users*, lai būtu analoģija ar *MySQL* datu modeli kolekcija *Users* tika atstāta. Attēla 5.7. ir attēlots *MongoDB* datu modelis.



*5.7.att.***MongoDB datu modelis**

Kā redzams datu modeļa attēlā 5.7 datu bāze sastāv no četrām kolekcijām – *Users*, *Score*, *Loadtime* un *Cheaters*. Šīs kolekcijas veic tās pašas funkcijas, kas ir aprakstītās 5.5 nodaļā, tomēr to struktūra ir savādāka un ir aprakstīta tabulās 5.6. *Users*, 5.7. *Score*, 5.8. *Loadtime* un 5.9. *Cheaters* attiecīgi.

5.6. tabula

Kolekcija Users

Nosaukums	**Datu tips**	**Apraksts**
_id	object id	*MongoDB* dokumenta identifikators.
uid	integer	Unikāla atslēga. Lietotāja identifikācijas numurs sociālā tīmeklī.
name	string	Lietotāja vārds.
surname	string	Lietotāja uzvārds.
avatar	string	Saite uz lietotāja bildi.
date	string	Datums un laiks, kad lietotājs ir reģistrējies lietotnē.

5.7. tabula

Kolekcija Score

Nosaukums	**Datu tips**	**Apraksts**
_id	object id	*MongoDB* dokumenta identifikators.
uid	integer	Unikāla atslēga. Lietotāja identifikācijas numurs sociālā tīmeklī.
name	string	Lietotāja vārds.
surname	string	Lietotāja uzvārds.
avatar	string	Saite uz lietotāja bildi.
date	string	Datums un laiks, kad lietotājs ir ieguvis pēdējo rezultātu spēlē.
score	integer	Spēles rezultāts.
gametime	integer	Spēles laiks sekundēs.
week	string	Gada nedēļas numurs.

5.8. tabula

Kolekcija Loadtime

Nosaukums	Datu tips	Apraksts
_id	object id	*MongoDB* dokumenta identifikators.
time	float	Lietotnes ielādes laiks.

5.9. tabula

Kolekcija Cheaters

Nosaukums	Datu tips	Apraksts
_id	object id	*MongoDB* dokumenta identifikators.
uid	integer	Lietotāja identifikācijas numurs sociālā tīmeklī.

6. LIETOTNES METRIKAS

Šajā nodaļā ir aprakstīts lietotnes dažādas metrikas. Metrikas tika analizētas lietotnes izstrādes un lietošanas laikā. Dati tika ievākti no diviem sociālā tīkla portāliem *draugiem.lv* un *facebook.com*, kuros darbojās izstrādāta lietotne. Lai iegūtos datus būtu vieglāk salīdzināt un tie pēc iespējas būtu vienlīdzīgāki, lietotne vienādu laika posmu izmantoja *MySQL* un *MongoDB* datu bāzi. Abās datu bāzēs tika glabāta informācija par vaicājumu izpildes laikiem.

6.1. Lietotnes izstrādes laiks

Izstrādes laiks tika lietota programma *DeskTime* [17] – darba laika uzskaites programmatūra, kas dod iespēju uzlabot darba produktivitāti un apskatīties izstrādes laiku.

Izstrādājot lietotni izmatojot *MySQL* datu bāzi tika patērēts par 20% vairāk laika, nekā izmantojot *MongoDB* attiecīgi 10h un 8h stundas. Tas ir izskaidrojams ar to, kā *MySQL* datu bāzei pirms lietotnes izstrādes vajadzēja izveidot četrām tabulām struktūru, bet *MongoDB* datu bāzei definēt struktūru iepriekš nevajadzēja.

6.2. Lietotnes ielādes laiks

Lietotnes ielādes laiks ir laiks, kas ir nepieciešams, lai lietotne pilnībā būtu ielādējusies (ir ielādēti visi lietotnes elementi – grafika, skripti un flash spēle).

Lietotnes ielādes laiks tika mērīts izmantojot *Google Analytics* rīku. *Google Analytics* rīks veic tīmekļa lapas detalizētu statistikas ievākšanu. Tabulā 6.1. attēloti lietotnes ielādes laiki izmantojot dažādas datu bāzes. Ir ļoti labi redzams ka lietotne, kas atrodas *facebook.com* sociāla tīkla portālā ielādes laiks ir lielāks par *draugiem.lv* portālu. Ielādes laiks ir lielāks, jo *facebook.com*, pirmkārt, serveris atrodas ārzemes, otrkārt, visi dati tiek šifrēti caur drošligzdu slāni jeb *SSL*. Neskatoties uz to *facebook.com* gadījumā lietotnes laiks izmantojot *MongoDB* datu bāzi ir samazinājies par 29%.

6.1. tabula

Vidējie lietotnes ielādes laiki

	Izmatojot MySQL	**Izmatojot MongoDB**	
Draugiem.lv	1.72 sekundes	1.66 sekundes	3% ātrāk
Facebook.com	4.60 sekundes	3.27 sekundes	29% ātrāk

6.3. Vaicājuma izpildes laiks

Vaicājuma izpildes laiks ir laiks, kas ir patērēts, lai datu bāze spētu apstrādāt vaicājumu un atgriezt rezultātu.

Tika uzrakstīta funkcija, kas veica vaicājumu izpildes laika uzskaitīšanu un tabulā 6.2. redzami ievāktie statistikas dati. Katrā gadījuma vidējais laiks bija aprēķināts izmantojot ap 100 tūkstošiem ierakstu. Lietojot *MongoDB* datu bāzi abos variantos vaicājuma izpildes laiks ir ātrāks. Viens lietotājs atšķirību sekundes simtdaļas nepamanīs, tomēr, jā tie būs tūkstotis lietotāju vienlaicīgi, tad ieguvums ir neapšaubāms – vairāk par divām minūtēm.

6.2. tabula

Vidējie vaicājuma izpildes laiki

	Izmatojot MySQL	**Izmatojot MongoDB**	
Draugiem.lv	0.155 sekundes	0.031 sekundes	80% ātrāk
Facebook.com	1.263 sekundes	0.774 sekundes	49% ātrāk

6.4. Lietotāju statistika

Šobrīd *draugiem.lv* portālā aplikāciju izmantoja 8474 lietotāji, bet *facebook.com* portālā 7830 lietotāji. Portāla *draugiem.lv* lietotāji ir izspēlējuši 116675 spēles, bet *facebook.com* lietotāji ir izspēlējuši 79729 reižu. Lietotnes lietotāju skaits pamazām pieaug abos portālos, līdz ar to izspēlēto spēļu skaits ar būs lielāks.

NOBEIGUMS UN SECINĀJUMI

Bez sociālie tīkliem nav iedomājama mūsdienu dzīve. Tajos mēs pavadām laiku komunicējot viens ar otru, lasot ziņas, spēlējot spēles, skatoties video vai klausoties dziesmas. Katrs sociālais tīkls piedāvā savus pakalpojumus un servisus, lietotājam atliek tik izvēlēties sev tīkamāko un sākt veidot vai lietot tā saturu.

Relāciju datu bāzes ir tehnoloģijas, kas zināmas gandrīz jau 45. gadus. Pa šiem gadiem relāciju datu bāzes kļuvušas par neatņemamu izstrādes daļu, dēļ plašām iespējam un *SQL* pieprasījumu valodas. Tomēr relāciju datu bāzes vislabāk ir piemērotas noteiktu uzdevumu risināšanai, kur dati ir normalizēti, zināmas datu struktūras un dati tiek mērogojami vertikāli.

NoSQL ir jauna elpa datu bāzes vēsture. Šīs datu bāzes sniedz savādākas pieejas datu glabāšana, meklēšanā un mērogojamībā. Šāda veida datu bāzes nav sarežģītas, jo tās strādā ar nestrukturētiem datiem, tas domāts, lai panāktu stabilitāti, elastīgumu un ātrdarbību darbā ar lieliem datu apjomiem. *NoSQL* datus ir viegli mērogot horizontāli *NoSQL* datu bāzes iedalās vairākos tipos, respektīvi, katram uzdevumam var atrast atbilstošāko datu bāzi..

Ir pieejams datu bāzes sistēmas ar atvērtu kodu. Tādas datu bāzes var lietot gan mācību, gan komerciālos nolūkos. Tieši tāpēc *MongoDB* atbilst abiem šiem nosacījumiem.

Maģistra darba sākumā ir izvirzīti vairāki mērķi, kurus veiksmīgi ir sanācis realizēt darba gaitā.

Maģistra darba rezultātā ir apkopota informācija par sociāliem tīkliem un datu bāzēm. Ir aprakstīta sociālo tīklu un datu bāzes vēsture. Ir aprakstītas populārākas sociālo tīklu vietnes. Ir dots saraksts ar *SQL* un *NoSQL* datu bāzēm. Tiek salīdzinātas *SQL* un *NoSQL* atšķirības, kā arī ir doti sintakses pieraksta piemēri. Tiek aprakstīti trīs *NoSQL* datu glabāšanas tipi un doti datu bāzes piemēri. No izpētītām *NoSQL* datu bāzēm ir izvēlēta *MongoDB NoSQL* datu bāze, kurā tiks glabāti lietotāju un lietotnes dati. *MongoDB* datu bāze pēdējo divi gadu laikā ir kļuvusi ļoti populārā dēl, pieejamās informācijas globālajā tīmekli un dēļ, tā ka to ir viegli apgūt. Pielikumā ir dots lietotnes *PHP* koda fragments, kurš strādā izmantojot *MongoDB NoSQL* datu bāzi.

Tiek paskaidrots, kas ir lietotne un kādam nolūkam tā ir domāta sociāla tīklā. Aprakstīts lietotnes darbības princips. Attēlota lietotnes integrācijas sociālajā tīkla un datu bāzes mijiedarbība ar to.

Izstrādājot maģistra darbu autors iemācījās veidot datu bāzi izmantojot *MongoDB*. Sākotnēji autors pat nezināja *MongoDB* sintaksi, tomēr lasot [18] informāciju par *MongoDB* iepazina to tuvāk. Paralēli izstrādei tika izpētīta programmatūra *MongoDB* administrēšanai.

Maģistra darba rezultāta ir izstrādātā lietotne, kas sākotnēji izmanto *MySQL* datu bāzi, bet pēc tam *MongoDB*. Analizējot lietotnes metrikas neradās šaubas, kā *MongoDB* praksē ir ātrāks un spēj strādāt ar lieliem datu apjomiem.

Darba izstrādes gaitā apgūtajām zināšanām ir praktisks pielietojums. Iegūtās zināšanas var pielietot, izstrādājot lietotnes izmantojot *NoSQL* datu bāzes.

Katram uzdevumam varbūt vairāki risinājumi, tāpēc pirms sākt jaunu lietotnes izstrādi, to vajag labi analizēt un saprast kādu risinājuma tehnoloģiju pielietot. Darba autors pēc iespējas vairāk sāks lietot *MongoDB*, lai padziļinātu savas zināšanas *NoSQL* datu bāzes un izstrādātu kvalitatīvāku programmatūru.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. *List of social networking websites.* [tiešsaiste]. Vikipēdija [atsauce 19.05.2013.]. Pieejams: http://en.wikipedia.org/wiki/List_of_social_networking_websites
2. *Learn more about NoSql databases.* [tiešsaiste]. 10gen [atsauce 19.05.2013.]. Pieejams: http://www.10gen.com/nosql
3. *Migrating from a relational to a NoSQL cloud database.* [tiešsaiste]. TechRepublic. [atsauce 19.05.2013.]. Pieejams: http://www.techrepublic.com/blog/datacenter/migrating-from-a-relational-to-a-nosql-cloud-database/5904
4. *Tiešsaistes sociālais tīkls.* [tiešsaiste]. Vikipēdija [atsauce 19.05.2013.]. Pieejams: http://lv.wikipedia.org/wiki/Tiešsaites_sociālais_tīkls
5. Kērkpatriks, D. *Facebook efekts.* Zvaigzne ABC, 2012. p. 408
6. *Retrieved October 4* [tiešsaiste] Amazonaws [atsauce 19.05.2013.].Pieejams: https://s3.amazonaws.com/OneBillionFB/*Facebook*+1+Billion+Stats.docx
7. *Akadēmiskā terminu datu bāze* [tiešsaiste] Termini.lza.lv [atsauce 19.05.2013.]. Pieejams: http://termini.lza.lv/term.*php*?term=database&list=&lang=EN&h=yes
8. *SQL.* [tiešsaiste]. Vikipēdija [atsauce 19.05.2013.]. Pieejams: http://en.wikipedia.org/wiki/*SQL*
9. *NoSQL.* [tiešsaiste]. Vikipēdija [atsauce 19.05.2013.]. Pieejams: http://en.wikipedia.org/wiki/*NoSQL*
10. *Top 5 Best Databases.* [tiešsaiste]. Thegeekstuff [atsauce 19.05.2013.]. Pieejams: http://www.thegeekstuff.com/2010/03/top-5-best-databases
11. *SQL to MongoDB Mapping Chart.* [tiešsaiste]. MongoDB [atsauce 19.05.2013.]. Pieejams: http://docs.mongodb.org/manual/reference/sql-comparison/
12. E. Plugge, P. Membrey, T. Hawkins, *The Definitive Guide to MongoDB*, Apress, 2010, p 328.
13. *Internet Browser Software Review 2013.* [tiešsaiste]. Compare Best Internet Browsers – TopTenREVIEWS [atsauce 19.05.2013.]. Pieejams: http://internet-browser-review.toptenreviews.com
14. *PhpMyAdmin Home.* [tiešsaiste]. PhpMyAdmin [atsauce 19.05.2013.]. Pieejams: http://www.phpmyadmin.net/home_page/index.php
15. *phpMoAdmin - MongoDB GUI administration tool for PHP.* [tiešsaiste]. PhpMoAdmin [atsauce 19.05.2013.]. Pieejams: http://www.phpmoadmin.com

16. *RockMongo - Best PHP MongoDB Administrator*. [tiešsaiste]. RockMongo [atsauce 19.05.2013.]. Pieejams: http://rockmongo.com
17. *DeskTime - Laika uzskaites un produktivitātes programmatūra*. [tiešsaiste] DeskTime [atsauce 19.05.2013.]. Pieejams: http://desktime.com
18. S. Francia, *MongoDB and PHP*, O`Reilly, 2012, p 80.

DOKUMENTĀRĀ LAPA

Maģistra darbs: **Sociālo tīklu lietotnes izveide izmantojot *NoSQL***

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____

(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **piemērotu** / **nepiemērotu** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____

(Vadītāja paraksts)

Darbs iesniegts **maģistrantūras sekretariātā** _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: _____

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)

PIELIKUMI

1. pielikums

1. Config.php fails

Pielikumā ir *config.php* faila *PHP* programmēšanas valodas koda fragments, kur var redzēt *MySQL* un *MongoDB* izmantošanu.

```
<?php
session_start();
require_once "../library/DraugiemApi.php";
require_once "../library/PAGES.php";

$app_key = '57cae1db0545f4514090ea35f05f6d07';
$app_id = 15013311;

$page_id = $PAGES[1]['id'];
$page_name = $PAGES[1]['name'];

/* MYSQL */
require_once "../library/DB.php";
/* MongoDb */
$connection = new MongoClient();
$mgdb = $connection->mgdb;

function getMgId($user_id) {
    global $mgdb;
    $mgId = $mgdb->users-
>findOne(array("uid" => $user_id), array('_id'=> 1));
    if( !is_null( $mgId ) ) {
        return $mgId;
    }
    return false;
}

function add_self( $uid , $avatar , $name , $surname ) {
    if( false ) {
        global $db;
        $sql=$db->Query("SELECT * FROM `users` WHERE `uid` = $uid ");
        $res_list = $db->Fetch($sql, 'ACCOS');
        if( !isset($res_list[0]['uid']) ) {
            $name = $db->Esc( $name );
```

```
            $surname = $db->Esc( $surname );
            $sql=$db->Query("INSERT INTO `users` SET
                `uid` = '".$uid."',
                `avatar` = '".$avatar."',
                `name` = '".$name."',
                `surname` = '".$surname."',
                `date` = NOW()
            ");
            return true;
        }
        return false;
    } else {
        global $mgdb;
        $mgId = getMgId($uid);
        if( $mgId === false ) {
            $user = array( 'uid' => $uid, 'avatar' => $avatar, 'name' => $n
ame, 'surname' => $surname, 'played' => 0, 'date' => date('Y-m-d H:i:s') );
            $db_users = $mgdb->users;
            $data = $db_users->save($user);
            return true;
        }
        return false;

    }
}


function insertScore( $uid = 0, $score, $gametime, $cheater, $avatar = "" ,
 $name = "", $surname = "") {
    if( false ) {
        global $db;
        $sql=$db-
>Query("SELECT * FROM `score` WHERE `uid` = $uid  AND `week`= WEEKOFYEAR(NO
W()) ");
        $res_list = $db->Fetch($sql, 'ACCOS');
        if( !isset($res_list[0]['uid']) ) {
            $sql=$db-
>Query("INSERT INTO `score` SET  `uid` = $uid, `gametime`= $gametime, `scor
e`= $score, `date` = NOW(), `week`= WEEKOFYEAR(NOW()) ");
        } else {
            if( $score > $res_list[0]['score'] ) {
                $sql=$db-
>Query("UPDATE `score` SET `gametime`= $gametime, `score`= $score, `date` =
```

```
 NOW() WHERE `uid` = $uid  AND `week`= WEEKOFYEAR(NOW()) ");
            }
        }

        if( $cheater ) {
            $sql=$db->Query("INSERT INTO `cheaters` SET  `uid` = $uid");
        }

        playedGame( $uid );
    } else {
        global $mgdb;
        $WEEKOFYEAR = date('W');
        $mgId = $mgdb->score-
>findOne(array("uid" => $uid, 'week' => $WEEKOFYEAR), array('_id'=> 1, 'sco
re' => 1 ));
        if( is_null( $mgId ) ) { //veicam jaunu ierakstu
            $score = array(
                'uid' => $uid, 'avatar' => $avatar,'name' => $name, 'surnam
e' => $surname,
                'score' => $score, 'gametime' => $gametime,
                'week' => $WEEKOFYEAR,
                'date' => date('Y-m-d H:i:s')
            );
            $db_score = $mgdb->score;
            $db_score->save($score);
        } else {
            if( $score > $mgId['score'] ) {
                $mgdb->score-
>update( array( '_id' => $mgId['_id'] ), array('$set' => array('score' => $
score, 'gametime' => $gametime )));
            }
        }
        if( $cheater ) {
            //indeksa uzlikšana caur kodu
            //$mgdb->cheaters-
>ensureIndex( array("uid" => 1), array( 'unique' => 1 ) );
            try {
                $db_cheaters = $mgdb->cheaters;
                $db_cheaters->save( array( 'uid' => $uid ) );
            }
            catch (MongoCursorException $e) {
                //duplikāts
            }
```

```
        }
        playedGame( $uid );
    }
}


function playedGame( $uid  = 0) {
    if( false ) {
        global $db;
        $sql=$db-
>Query("UPDATE `users` SET `played`=`played`+1 WHERE `uid` = $uid ");
    } else {
        global $mgdb;
        $mgdb->users-
>update( array( 'uid' => $uid ), array('$inc' => array('played' => 1 ) ) );
    }
}


function drawTop( $uid = 0 ) {
    if( false ) {
        global $db;
        $html = '<div id="users_box">';

        $top12 = 0;
        $i = 1;
        $sql=$db-
>Query("SELECT a.`uid`,a.`score`, b.`name`, b.`surname`, b.`avatar` FROM `s
core` as a, `users` as b WHERE a.`uid` = b.`uid` AND `week`= WEEKOFYEAR(NOW
()) ORDER BY a.`score` DESC, a.`gametime` ASC LIMIT 12 ");
        $res_list = $db->Fetch($sql, 'ACCOS');
        if( isset($res_list[0]['uid']) ) {
            foreach ($res_list as $row) {
                $color = '';
                if( $uid == $row['uid'] ) {
                    $color = ' style=" color: #FBF719; " ';
                    $top12 = $i;
                }

                $html .= '
                    <div id="userbox">
                        <div class="number">'.$i.'.</div>
                        <div class="avatar" style="background-
```

```
image: url('.$row['avatar'].');"></div>
                        <div class="name">
                            <div class="nick" '.$color.'>'.$row['name'].' '
.$row['surname'].'</div>
                            <div>'.$row['score'].' punkti</div>
                        </div>
                        <br class="clear" />
                    </div>
                ';
                $i++;
            }
        }

        if( $top12 == 0 ) {
            $sql=$db-
>Query("SELECT `score` FROM `score` WHERE `uid` = ".$uid." AND `week`= WEEK
OFYEAR(NOW()) LIMIT 1 ");
            $res_list = $db->Fetch($sql, 'ACCOS');
            if( isset($res_list[0]['score']) ) {
                $currScore = $res_list[0]['score'];
                $sql=$db-
>Query("SELECT count(*) as `cnt` FROM `score` WHERE `score` > $currScore AN
D `week`= WEEKOFYEAR(NOW()) ");
                $res_list = $db->Fetch($sql, 'ACCOS');
                if( isset($res_list[0]['cnt']) ) {
                    $place = $res_list[0]['cnt']+1;
                    $html .= '<br class="clear" /><div id="topplace">Tava v
ieta topā '.($place).'.</div>';
                }
            }
        }

        $html .= '</div>';
        return $html;
    } else {
        global $mgdb;
        $WEEKOFYEAR = date('W');

        $res_list = $mgdb->score-
>find( array( 'week' => $WEEKOFYEAR ), array() )->limit(12)-
>sort( array('score'=> -
1, 'gametime' => 1 ) ); //print_r( iterator_to_array($res_list) );
        $html = '<div id="users_box">';
```

```
        $top12 = 0;
        $i = 1;
        foreach ($res_list as $row) {
            $color = '';
            if( $uid == $row['uid'] ) {
                $color = ' style=" color: #FBF719; " ';
                $top12 = $i;
            }

            $html .= '
                <div id="userbox">
                    <div class="number">'.$i.'</div>
                    <div class="avatar" style="background-
image: url('.$row['avatar'].');"></div>
                    <div class="name">
                        <div class="nick" '.$color.'>'.$row['name'].' '.$ro
w['surname'].'</div>
                        <div>'.$row['score'].' punkti</div>
                    </div>
                    <br class="clear" />
                </div>
            ';
            $i++;
        }
        if( $top12 == 0 ) {
            $mgId = $mgdb->score-
>findOne(array("uid" => $uid, 'week' => $WEEKOFYEAR), array('_id'=> 1, 'sco
re' => 1 ));
            if( !is_null( $mgId ) ) {
                $currScore = (int)$mgId['score'];
                $result = $mgdb->score-
>find( array( 'score' => array( '$gt' => $currScore ), 'week' => $WEEKOFYEA
R ) );
                $html .= '<br class="clear" /><div id="topplace">Tava vieta
 topā '.( $result->count()+1 ).'.</div>';
            }
        }
        $html .= '</div>';
        return $html;
    }
}
```

```
function savepageloadtime( $loadtime ) {
    if( false ) {
        global $db;
        $sql=$db-
>Query("INSERT INTO `loadtime` SET  `time` = '$loadtime' ");
    } else {
        global $mgdb;
        $db_loadtime = $mgdb->loadtime;
        $db_loadtime->save( array( 'time' => $loadtime ) );
    }
}

?>
```