# Уводзіны

У якасці курсавога праекта была зроблена прылада для збірання дадзеных, іх хавання і стварэння аналітычных запытаў на іх аснове. Прылада таксама дазваляе ствараць уліковыя запісы для карыстальнікаў і дае магчымасць кіраваць аналітычнымі запытамі з створаных уліковых запісаў. Тэматыкай праекта выбрана кампутарная гульня i 2, з-за наяўнасці вялікіх аб’ёмаў дадзеных, магчымасці іх атрымання і вялікіх патэнцыяльных магчымасцей для стварэння разнастайных аналітычных і прагнастычных запытаў.

Для атрымання дадзеных выкарыстоўваецца інфармацыйная платформа з адкрытым зыходным кодам “OpenDota”. Дзякуючы звычайным Http запытам да гэтага знешняга сховішча праект можа атрымліваць усю неабходную інфармацыю. Запыты фарміруюцца ў часцы прылады, напісанай на мове праграмавання Java, пасля чаго адбываецца выбарка атрыманых дадзеных і пераўтварэнне іх ў адвапедныя структуры дадзеных для далейшай працы з імі. Пасля таго, як дадзеныя атрыманы і пераапрацаваныя прылада захоўвае іх ў базу дадзеных.

Для СКБД была выбрана Oracle. У ходзе распрацоўкі праекта база дадзеных знаходзілася на іншай віртуальнай машыне. Большая частка ўзаемадзеянняў з базай дадзеных з боку напісанай на java праграмай адбываецца цераз выклікі працэдур. База дадзеных нармалізваная, што дазваляе эканоміць фізічна займаемае месца, але запавольвае выкананне аналітычных запытаў з-за размеркаваннасці хаваемых дадзеных. Для больш хуткай апрацоўкі агрэгатных функцый была створана дадатковая база дадзеных, якая іграе ролю сховішча дадзеных. Некаторыя табліцы ў сховішчы не нармалізаваныя а створаны ў выглядзе flat tables, для аптымізацыі запытаў, звязаных з аб’яднаннем некалькіх вялікіх табліц. Дадзеныя з базы дадзенай перыядычна загружаюцца на сховішча, пасля чаго могуць быць выдаленыя з базы дадзеных, каб не займаць лішняе месца на носьбіце. У працэсе загрузкі на сховішча некаторыя дадзеныя агрэгуюцца, такім чынам у табліц сховішча дадзеных знаходзіцца больш палёў чым ў адпаведных табліц базы дадзеных.

Аналітычныя запыты працуюць з дадзенымі захаванымі ў сховішчы дадзеных, якія лепш аптымізаваня для агрэгатных функцый і маюць загадзя вылічаныя значэнні.

Аналітычныя запыты дазваляюць ствараць графічныя дыяграмы з вызначанымі выбаркамі і даследуемымі характарыстыкамі.

Для ўзаемадзеяння карыстальнікаў з прыладай створаны вэб сервер, напісаны на java. Існуюць дзве ролі ўліковых запісаў, якія адрозніваюцца даступнымі для карыстальніка функцыямі.

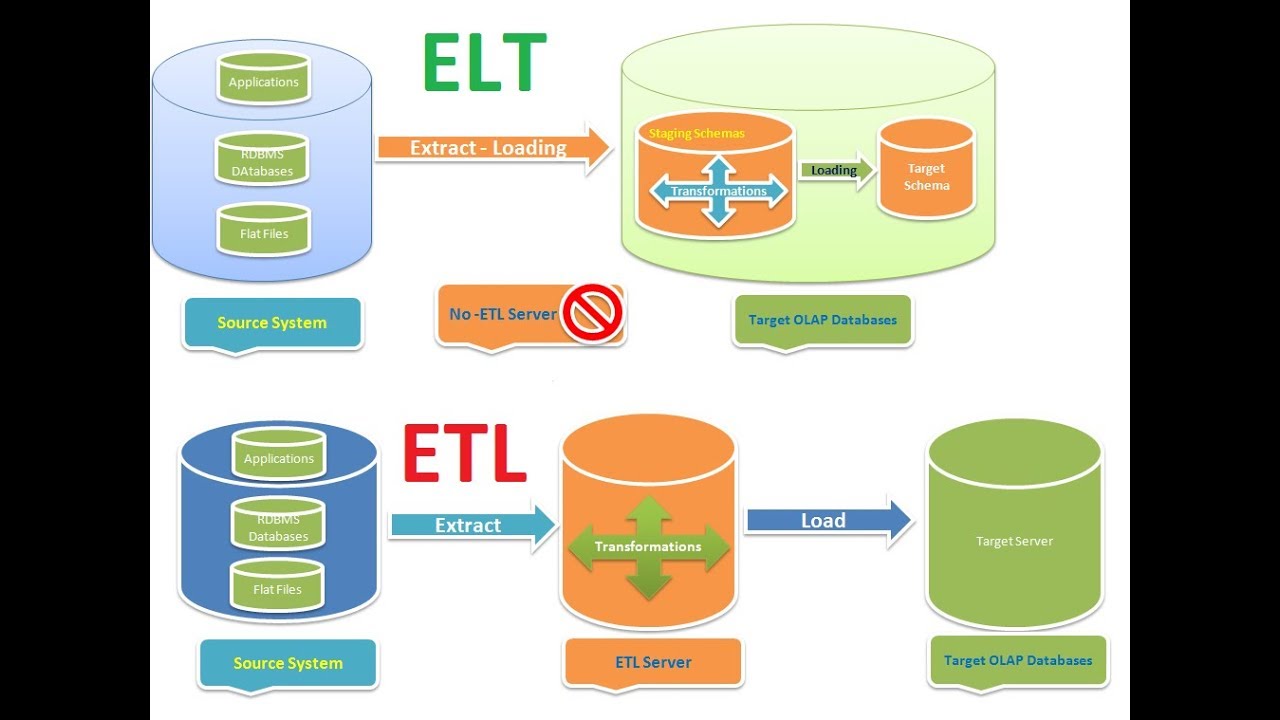
Дзякуючы створанаму праекту карыстальнік можа ствараць, змяняць і выдаляць аналітычныя запыты і выбаркі дадзеных, пасля чаго на іх аснове можа атрымаць графічныя дыяграмы.

## Аналітычны разбор літаратуры

Асноўная мэта праекта – стварэнне і выкананне аналітычных запытаў, на яснове якіх можна рабіць прагнозы альбо адпаведныя стратэгіі развіцця. Для аналітычных запытаў неабходна апрацоўваць вялікія аб’ёмы дадзеных і выклікаць агрэгатныя функцыі. Рэляцыйныя базы дадзеных добра падыходзяць для задачы апрацоўкі і агрэгацыі дадзеных у выпадку іх невялікіх аб’ёмаў, але для аб’ектыўнай статыстыкі неабходна вялікая выбарка, што патрабуе шмат дадзеных. Гэта не падыходзіць для рэляцыйных базаў дадзных, дзе спатрэбіцца для кожнага запыта аб’ядноўваць вялікія табліцы і рабіць агрэгацыю па, часта, аднолькавым дадзеным. У некаторых праектах у якасці рашэння выкарыстоўваецца тэхналогія апрацоўкі дадзеных OLAP, што дазваляе апрацоўваць вялікія аб’ёмы дадзеных значна хучэй, за кошт загадзя вылічаемых значэнняў агрэгатных функцый для некалькіх набораў дадзеных, а таксама выкарыстання плоскіх табліц. Аднак OLAP патрабуе пэўную структуру базы дадзеных, напрыклад Star-schema альбо Snowflake-schema. Такім чынам для OLAP тэхналогіі будзе выкарыстоўвацца большы аб’ём дадзеных з-за загадзя пралічаных і аб’яднаных дадзеных, што дае большую хуткасць апрацоўкі агрэгатных функцый. На аснове star альбо snowflake схемы базы дадзеных будуецца OLAP-куб – структура дадзеных, якая дазваляе рабіць хуткія выбаркі і агрэгатныя па розным палям табліц, але для куба неабходна толькі адна вялікая табліца, таксама вядомая як табліца фактаў, і некалькі адносна малых табліц – табліца вымярэнняў. Такім чынам аб’днанне вялікай табліцы з малымі значна хутчэй адбываецца чым аб’яднанне дзвюх вялікіх табліц. Аднак ў дадзеным праекце гэтая ўмова не можа выконвацца, так як важна ўлічваць не адну асобную сутнасць ў табліцы фактаў, трэба яшчэ ўлічваць ўзаемадзеянні гэтай сутнасці з іншымі такімі ж сутнасцямі табліцы фактаў, што патрабуе аб’аднанняў адной і той жа вялікай табліцы фактаў, што не дае магчымасці выкарыстання OLAP-куба. Яшчэ адной перашкодай з’яўляецца тое, што ў звычайных табліц вымярэнняў, выкарыстоўваемых для куба, можна выдзяліць некалькі іерархічных ўзроўняў, па якім можна аб’ядноўваць дадзеныя табліцы і па якім вылічваюцца агрэгатныя функцыі. У дадзеным праекце складана стварыць іерархію сутнасцей, якая будзе часта выкарыстоўвацца ў запытах, бо сутнасці нельга падзяліць на асобныя катэгорыі. Апошняй прычынай адмаўлення ад выкарыстання OLAP-куба стала сувязі many-to-many у табліцах праекта. Гэтая сувязь змяньшае эфектыўнасць куба, а ў рэалізацыі куба сродкамі Oracle Data Integrator і Oracle Analytic Workspace Manager няма яўнага выкарыстання сувязі m2m, у адрознінне ад Microsoft Sql Server.

У праекце выкарыстоўваецца прамежкавае рашэнне паміж звычайнай рэляцыйнай базай дадзеных і OLAP – сховішча дадзеных. Сховішча дадзеных здольна аб’ядноўваць дадзеныя з розных разнастайных крыніц інфармацыі, у выпадку размеркаваных базаў дадзеных, а таксама сховішча дазваляе пераўтвараць атрыманыя дадзеныя ў адзіны фармат і пераапрацоўваць іх. У дадзеным праекце няма неабходнасці ў прывядзенні дадзеных да адзінага фармата, так як крыніца інфармацыі, пакуль што, адна. Сховішча дадзеных праводзіць вылічэнні з атрыманымі дадзенымі для стварэння новых палёў табліцы, такім чынам утвараюцца загадзя гатовыя значэнні, якія не спатрэбіцца вылічваць занава пры запытах. У сховішчы таксама загадзя аб’яднаны вялікія табліцы для аптымізацыі запытаў за кошт выкарыстання дадатковай памяці для хавання дадзеных.

Для запаўнення сховішча дадзенымі існуе два асноўныя падыходы, агульная схема якіх паказана на малюнку 1.1.



* + - 1. Схема працы ELT

Для запаўнення сховішча выкарыстоўваецца падыход Extract Load Transform, так як для ўзаемадзеяння з базай дадзеных oracle існуе існтрумент для працы са сховішчам дадзеных, які выкарыстоўвае падыход ELT. Гэты падыход прадугледжвае ў першую чаргу выгрузку дадзеных з крыніц інфармацыі. У выпадку некалькіх крыніц інфармацыі перадаваемыя дадзеныя аб’ядноўваюцца. Калі дадзеныя загружаюцца ў сховішча дадзеных яны выкарыстоўваюцца для далейшых вылічэнняў і захавання атрыманых дадзеных таксама ў сховішчы.

Для рэалізацыі сховішча і яго запаўнення дадзенымі разглядаліся два праграмныя сродкі. Analytic Workspace Manager валодае магчымасцямі для лёгкай пабудовы OLAP-куба, але адносна Oracle Data Integrator AWM мае меньш функцый для трнспарціроўкі і пераўтварэння дадзеных, а таксама ў кіраваннямі гэтых працэдур. Таму для рэалізацыі сховішча і яго запаўнення выкарыстоўвася ODI, чаму спрыяла наяўнасць вялікай колькасці афіцыйных вучэбных матэрыялаў, абумоўленай папулярнасцю ODI.

## Распрацоўка праграмнага асяроддзя

### Пабудова рэляцыйнай базы дадзеных

База дадзеных рэалізаваная як падключаемая база дадзеных да сервера oracle. Рэляцыйную базу дадзеных, выкарыстоўваемую ў праекце, можна падзяліць на дзьве часткі: адна частка табліц звязаная з гульнявымі дадзенымі і запаўняецца з OpenDota, яна атрымлівае дадзеныя, на падставе якіх будуць вылічвацца аналітычныя запыты; другая частка базы дыдзеных – табліцы патрэбныя для абслугоўвання карыстальнікаў прылады. Падзел частак базы дадзеных рэалізаваны падзелам таблічных прастораў, у якіх знаходзіцца табліцы, а таксама доступ да табліц рэалізаваны цераз выклік працэдур ад розных карыстальнікаў падключаемай базы дадзеных.

#### База з гульнявымі дадзенымі

База дадзеных, выкарыстоўваемая для гульнявых дадзеных, прадстаўлена ў !дадатку. Складаецца з табліц для захавання адносна пастаянных дадзеных – Items, Heroes, Roles, HeroesRoles. Яны захоўваюць гульнявыя дадзеныя, якія змяняюцца толькі з аднаўленнямі самой гульні, што значна радзей за аднаўленне іншых табліц базы дадзеных.

Items – захоўвае інфармацыю аб гульнявых прадметах: назва, апісанне, спсылка на выяву прадмета.

Heroes – захоўвае інфармацыю аб гульнявых героях: імя, спасылкі на выявы.

Roles – захоўвае гульнявыя ролі для герояў. Кожны герой можа валодаць некалькімі ралямі, і наадварот – адна роля можа належыць некалькім героям. Так як тып сувязі паміж табліцамі Heroes і Roles m2m, была створана дадатковая прамежкавая табліца HeroesRoles.

Для табліц Heroes і Items створаны ўнікальныя ідэнтыфікатары, якія бяруцца з OpanDota. Для ролі ў якасці першаснага ключа выкарыстоўваецца імя ролі.

Астатнія табліцы аднаўляюцца часцей – кожны раз пры атрыманні новых дадзеных з OpenDota.

Players – табліца з гульцамі, патрэбна толькі для ідэнтыфікацыі розным гульцоў, таму не ўтрымлівае іншых палёў акрама лічбавага ідэнтыфікатара.

Matches – табліца з агульнай інфармацыяй аб гульнявым матчы: працягласць матча, пераможца, дата гульні і інш.

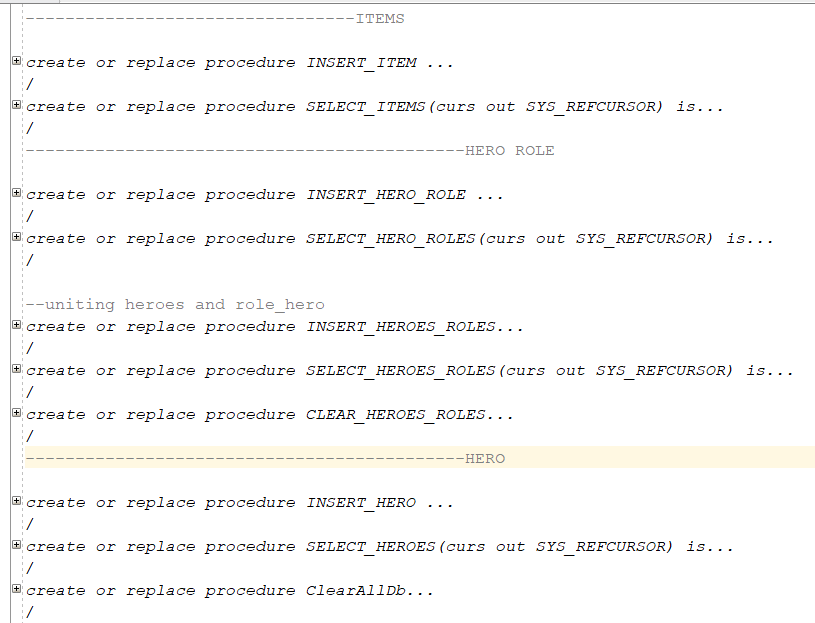
PlayersMatches – табліца для хавання інфармацыі аб гульцы ў канкрэтным матчы: колькасць забойстваў, пашкоджанняў, смярцей за матч і інш. Ідэнтыфікатар генеруецца аўтаматычна.

PlayersMatchTimeStat – табліца з інфармацыяй аб гульцы пад час матча ў канкрэтную хвіліну матча. На падставе гэтай табліцы можна прасачыць за развіццём канкрэтнага гульца на працягу ўсяго матча і зрабіць агульныя вынікі.

Індэксы ў табліцах выкарыстоўваюцца ствроаныя па змаўчанні на ўнікальныя палі і першасныя ключы, а таксама створаныя ўручную індэксы па палям другасных ключоў, для больш хуткых аб’яднанняў табліцы па гэтым палям. Але вялікай неабходнасці ў індэксах гэтай базы дадзеных няма, бо яна невялікая і выйгрыш у вытворчасці будзе невялікім.

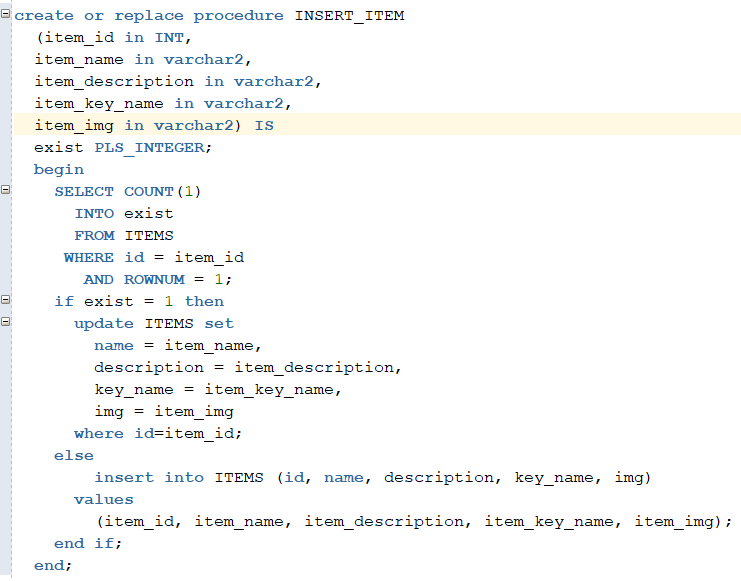
#### Працэдуры для гульнявых дадзеных

Усё ўзаемадзеянне вэб сервера з рэляцыйнай базай дадзеных адбываецца цераз працэдуры. На !малюнку паказаны працэдуры для дабаўлення і атрымання рэдка змяняемых дадзеных.



* + - 1. Працэдуры ўзаемадзення з рэдка аднаўляемымі табліцамі

Гэтыя працэдуры ў асноўным зробленыя для аднаўлення і дабаўлення дадзеных напрыклад INSERT\_ITEM паказаная на !малюнку.

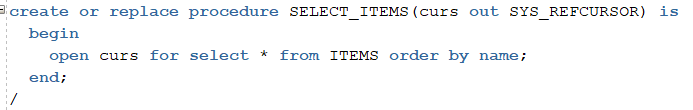


* + - 1. Рэалізацыя працэдуры INSERT\_ITEM

Працэдура прымае параметры якія супадаюць з палямі табліцы Items. У дадзеным выпадку выкарыстоўваецца статычная тыпізацыя зменных, але лепшай прыктыкаў было б выкарыстанне атрыбутаў %TYPE альбо %ROWTYPE, для зручнейшых зменаў тыпаў палёў табліцы.

У цэле прыцэдуры спярша правяраецца існаванне радка з атрыманым ідэнтыфікатарам, калі такога не існуе то дабаўляецца новы радок у табліцу Items з палямі адпавядаючымі параметрам працэдуры, Калі радок з такім ідэнтыфікатарам існуе то гэты радок аднаўляецца згодна з перададзенымі ў працэдуру параметрамі. Дадзеныя дзеянні можна таксама выконваць з дапамогай аператара merge, што зпросціць код.

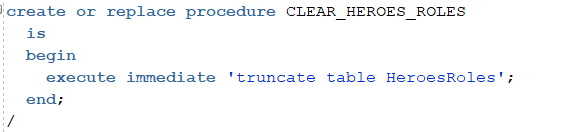
Другі тып працэдур вяртае набор радкоў табліц, напрыклад рэылізацыя працэдуры SELECT\_ITEMS паказана на !малюнку.



* + - 1. Рэалізацыя працэдуры SELECT\_ITEMS

Дадзеная працэдура прымае адзін выходны параметр, які прыме значэнне курсора для разбора адпаведнай табліцы. Для атрымання дадзеных выкарыстоўваецца курсор, пасля чаго, дзякуючы яму, вэб сервер апрацоўвае дадзеныя табліцы.

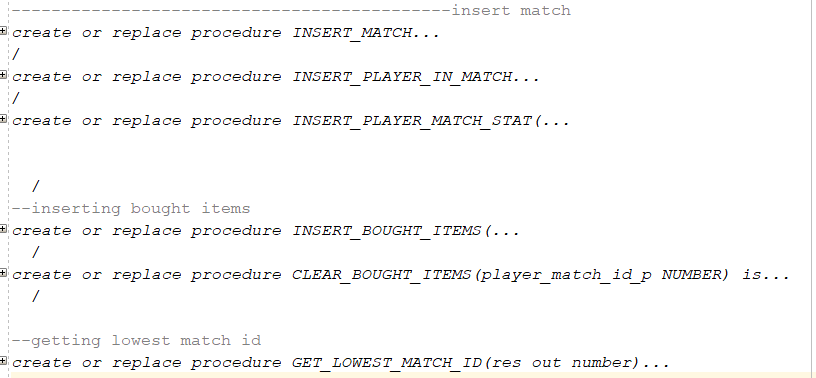
Так як ў базе дадзеных маюцца прамежкавыя табліцы для рэалізацыі сувязі many to many для іх распрацаваныя асобныя працэдуры, напрыклад CLEAR\_HEROES\_ROLES, чыя рыалізацыя паказана на !малюнку.



* + - 1. Рэалізацыя працэдуры CLEAR\_HEROES\_ROLES

Дадзеная працэдура выдаляе ўсё змесціва прамежкавай табліцы HeroesRoles і выклікаецца перад новым яе запаўненнем.

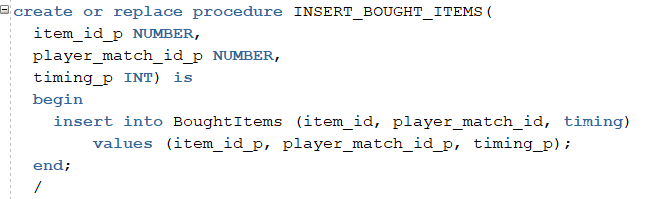
Таксама маюцца працэдуры для ўзаемадзеяння з часта аднаўляемыі табліцамі, паказаны на!малюнку.



* + - 1. Працэдуры для ўзаемадзеяння з часта аднаўляемымі табліцамі

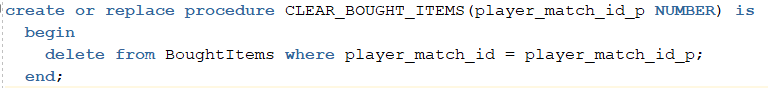
Адсутнасць працэдур атрымання радкоў з табліц тлумачыцца тым, што дадзеная база дадзеных мае мэту часоваха хавання дадзеных, а не пастаяннага ўзаемадзеяння з імі, гэтая функцыя пакладзена на сховішча дадзеных, разглядаемае ніжэй. Асноўныя аперацыі з часта аднаўляемымі табліцамі з’яўляецца дабаўленне новых радкоў, аднаўленне ўжо дабаўленых радкоў не мае вялікага сэнсу, з-за іх нязначнасці адносна вялікага набора асатніх радкоў.

Прыклад рэалізацыі працэдуры для дабаўлення радкоў у табліцу паказаны на !малюнку.



* + - 1. Рэалізацыя працэдуры INSERT\_BOUGHT\_ITEMS

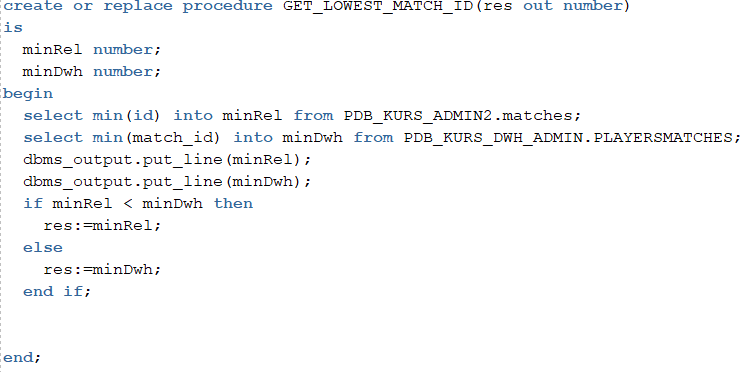
Працэдура прымае параметры для палёў новага радка дабаўляемага ў табліцу. Тут адсутнічае праверка на існавання падобных радкоў, таму што іх наяўнасць дапушчальная і абумоўлена асаблівасцямі гульні. Так як дапушчальна выкарыстанне аднолькавых радкоў важна сачыць за тым, каб іх колькасць строга адпавядала рыяльным дадзеным, для чаго зроблена працэдура паказаная на !малюнку.



* + - 1. Рэалізацыя працэдуры CLEAR\_BOUGHT\_ITEMS

У выпадку памылкі пры дабаўленні новых радкоў у табліцы ёсць небяспечнасць дабаўлення вялікай колькасці аднолькавых дадзеных ў табліцу BoughtItems, таму перад кожным дабаўленнем ў гэтую таблічу дадзеных аб гульцы пад час матчу з табліцы BoughtItems выдалаецца адпаведная інфармацыя.

Рэалізацыя яшчэ адной праэцдуры паказана на !малюнку



* + - 1. Рэалізацыя працэдуры GET\_LOWEST\_MATCH\_ID

Дадзеная працэдура вяртае найменьшы ідэнтыфікатар матча, які ўтрымліваецца ў базе дадзеных альбо ў сховішчы дадзеных. Выкарыстоўваецца для атрымання толькі новых дадзеных з OpenDota.

Усе вышэй адзначаныя працэдур не маюць апрацоўшчыкаў памылак, ўсе памылкі, якія ўзнікаюць пад час выканання працэдур, апрацоўваюцца ўжо ў вэб серверы, так як на ўзроўні працэдур базы дадзеных выконваецца вельмі прымітыўная логіка.

Працэдуры запаўнення часта аднаўляемых табліц выклікаюцца па ініцыятыве карыстальнікаў з адпаведнымі правамі.

#### База з дадзенымі карыстальнікаў прылады

Табліцы, выкарыстоўваемыя для захавання дадзеных карытсальнікаў, паказаны ў !дадатку. Гэтая частка базы дадзеных складаецца з наступных табліц:

* App\_users захоўвае дадзеныя аб ўліковых запісах карыстальнікаў прылады;
* Selections захоўвае дадзеныя неабходныя для выбарак, ствараемых карыстальнікамі;
* Selection\_heroes захоўвае дадзеныя аб героях, якія ўтрымлівае выбарка;
* Selection\_items захоўвае прадметы, якія ўтрымлівае выбарка;
* LineChart захоўвае дадзеныя аб лінейных графіках карыстальнікаў;
* BubbleChart прадстаўляе дадзеныя аб кропкавых дыяграмах карыстальнікаў;
* BarChart захоўвае дадзеныя для слупковых дыяграм карыстальнікаў;
* BarChartLabels захоўвае вымяраемыя значэнні, неабходныя для слупковых дыяграм;
* Selections\_barChart адлюстроўвае выбаркі, з якіх складаецца слупковая дыяграма;
* Selections\_bubbleChart захоўвае выбаркі, якія выкарыстоўваюцца ў кропкавай дыяграме.

Як і ў папярэдніх табліцах, у табліцах з дадзенымі карыстальнікаў не ўтрымліваюцца індэксы, апрача індэксаў першасных ключоў і абмежавання на ўнікальнасць, так як аб’ёмы дадзеных не велікія і патрэбы ў астатніх індэксах няма.

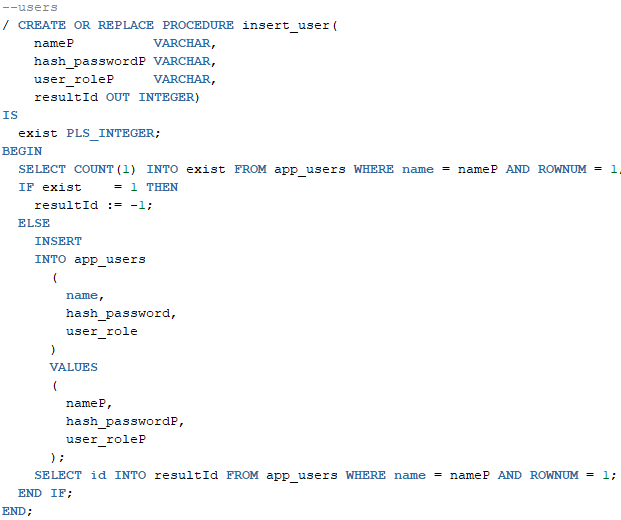
Змесціва дадзенай часткі базы дадзеных аднаўляецца толькі па ініцыятыве карыстальнікаў і не залежыць ад знешніх крыніц інфармацыі, у адрознінне ад табліц з гульнявым змесцівам. Усе ўнікальныя ідэнтыфікатары аўтаматычна генеруюцца.

На дыяграме можна заўважыць сувязі некаторых табліц з базай гульнявых дадзеных прадстаўленыя ў выглядзі другасных ключоў. На самай справе сувязь гэтых частак базы дадзеных рэалізуецца ўжо на ўзроўні вэб сервера.

Для забеспячэння бяспечнасці прылады ва ўліковых запісах карыстальнікаў захоўваюцца захэшыраваныя значэнні пароляў. Такім чынам пры ўцечцы дадзеных табліцы доступ да ўліковых запісаў карыстальнікаў будзе па-ранейшаму не магчымы.

#### Працэдуры для карыстальніцкіх дадзеных

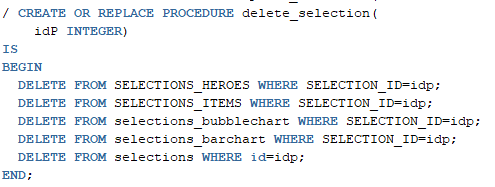
У ходзе працы прылады неабходна захоўваць, змяняць і выдаляць мноства карыстальніцкіх дадзеных, для чаго неабходна ўзаемадзеянне за базай дадзеных. Працэдуры для ўзаемадзеяння вэб сервера з базай дадзеных прадстаўлена ў !дадатку. З-за наяўнасці мноства табліц, звязаных паміж сабой, ёсць неабходнасць стварыць мноства працэдур, для аперацый дабаўлення, аднаўлення, выдалення і атрымання дадзеных. Прыклад рэалізацыі працэдуры дабаўлення дадзеных прадстаўлена на !малюнку.



* + - 1. Рэалізацыя працэдуры insert\_user

Працэдура дабаўляе радок з уваходнымі параметрамі ў выпадку адсутнасці такога радка ў табліцы, калі імя дабаўляемага карыстальніка ўжо знахдзіцца ў табліцы, новы карыстальнік н едабаўляецца і вяртаецца адпаведнае значэнне выклікаемай вэб серверам функцыі. Для ўзаемадзеяння з ўліковымі запісамі карыстальнікаў існуюць яшчэ дзьве працэдуры: для атрымання ўліковага запісу па яго імені і праверка наяўнасці ўліковага запісу з дадзеным іменем. Працэдуры для аднаўлення інфармацыі аб радку рэалізаваны аналагічным чынам.

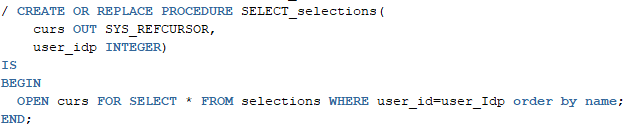
Прыклад рэалізацыі працэдуры для выдалення запісаў прадстаўлены на !малюнку.



* + - 1. Рэалізацыя працэдуры delete\_selection

Працэдура прымае ў якасці параметра ідэнтыфікатар выдаляемага радка. З-за наяўнасці вялікай колькасці сувязей паміж табліцамі выдаленне радка з адной табліцы часта патрабуе закранання змесціва астатніх звязаных табліц. У дадзенай працэдуры закранаюцца 5 табліц.

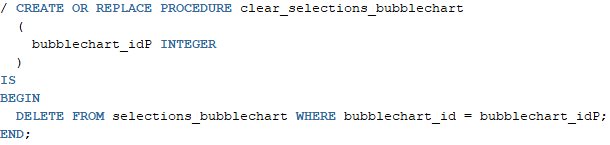
Прыклад рэалізацыі працэдуры для атрымання набора радкоў з табліц прадстаўлены на !малюнку.



* + - 1. Рэалізацыя працэдуры Select\_selections

Вяртанне набора радкоў, як і ў папярэдніх выпадках, рэалізавана цераз выходны параметар тыпа курсора. У дадзенай працэдуры вяртаюцца выбаркі, якія належаць карыстальніку з дадзеным ідэнтыфікатарам.

Для рэалізацыі сувязей many to many былі створаны прамежкавыя табліцы, з-за чаго пры дабаўленні новых дадзеных у табліцы трэба захоўваць змеціва прамежкавых табліц у адноўленым стане, для чаго былі створаны працэдуры для выдалення пэўных радкоў прамежкавых табліц пры аднаўленні звязаных табліц. Прыклад рэалізацыі адной з такіх працэдур паказаны на !малюнку.

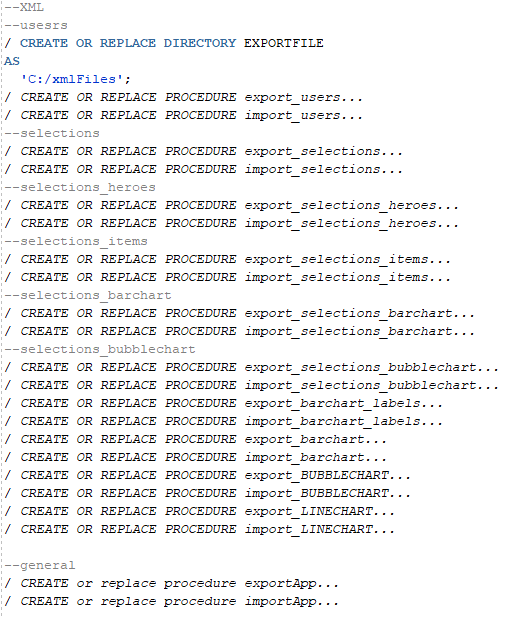


* + - 1. Рэалізацыя працэдуры clear\_selections\_bubblechart

У дадзенай працэдуры выдаляюцца сувязі дадзенай кропкавай дыяграмы з выбаркамі, па якім дыяграма будуецца. Працэдура выклікаецца пры аднаўленні і выдаленні дыяграмы.

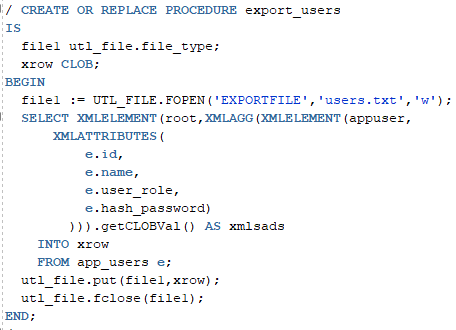
#### Рэалізацыя імпарта і экспарта дадзеных

У прыладзе рэалізавана функцыя экспарта і імпарта дадзеных, выкарыстоўваючы фармат xml. Гэтыя функцыі даступныя толькі для карыстальнікаў з пэўнымі правамі доступа. Захоўваюцца толькі дадзеныя карыстальнікаў, так як гульнявыя дадзеныя надта вялікія для адначасовага іх хавання ў базе дадзеных і ў экспартаваным файле, а таксама па прычыне магчымасці атрымаць ўсе гульнявыя дадзеныя зноў са знешняй крыніцы інфармацыі. Працэдуры, выкарыстоўваемыя пры экспарце і імпарце паказаны на !малюнку.



* + - 1. Працэдуры экспарта і імпарта

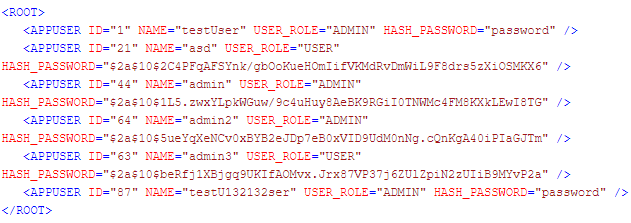
Для кожнай з табліц была зроблена працэдура для экспарта і імпарта з фармата xml. Прыклад рэалізацыі адной з працэдур экспарта паказана на !малюнку.



* + - 1. Рэалізацыя працэдуры export\_users

У працэдуры ў першую чаргу адчыняецца для запісу файл на серверы. У ходзе перабора ўсіх радкоў табліцы ўліковых запісаў карыстальнікаў фарміруецца xml аб’ект, з адным карнявым элементам і па аднаму элементу на кожны запіс ў табліцы. Атрыманы xml дакумент захоўваецца ў зменнай тыпу CLOB, пасля чаго гэтая зменная запісваецца ў адчынены раней файл. У канцы працэдуры файл зачыняецца. Дадзены падыход да рэалізацыі можа працаваць не правільна ў выпадках вялікіх аб’ёмаў дадзеных у табліцы. Дадзеная заўвага датычыцца ўсіх працэдур экспарта і імпарта, рэалізаваных ў праекце.

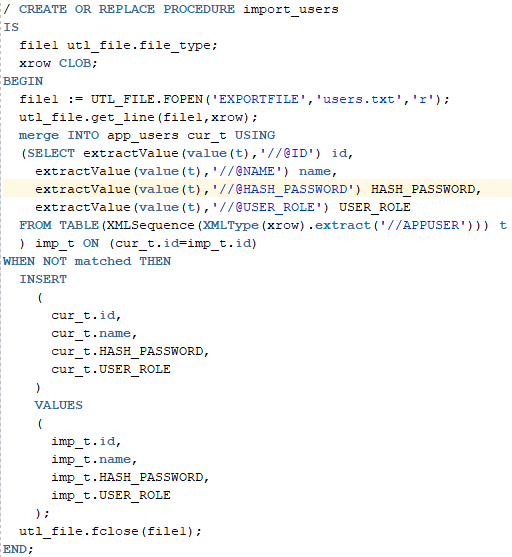
Структура атрыманага xml паказана на !малюнку.



* + - 1. Прыклад атрыманага xml файла

З-за асаблівасцей дэмаршалізацыі сродкамі oracle ўсе палі табліцы захоўваюцца ў якасці атрыбутаў xml элемента.

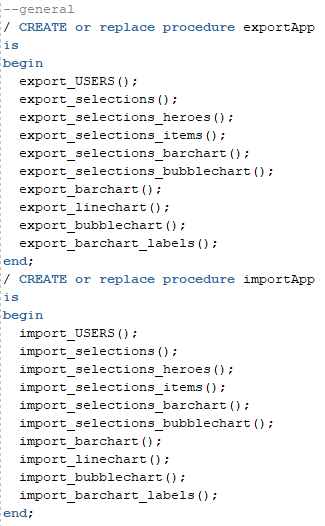
Прыклад рэалізацыі працэдуры імпарта дадзеных табліцы з xml фармата паказаны на !малюнку.



* + - 1. Рэалізацыя працэдуры import\_users

Працэдура адчыняе файл з xml дакументам і запісвае яго змесціва ў зменную, пасля чаго дэмаршалізуе атрыманы xml дакумент, стварае часовую табліцу з элементамі атрыманага xml дакумента і аб’ядноўвае дадзеныя з xml і дадзеныя з цякучай табліцы. Пры існаванні радка ў табліцы радок не будзе адноўлены з дакумента, бо лічыцца, што экспартаваныя дадзеныя больш старыя чым дадзеныя ў табліцы. У канцы працэдуры файл зачыняецца.

Для экспарта і імпарта дадзеных карыстальнік выклікае працэдуры, якія аб’ядноўваюць працэдуры адпаведных аперацый для кожнай табліцы. Рэалізацыя працэдур экспарта і імпарта паказана на !малюнку.



* + - 1. Рэалізацыя агульных працэдур імпарта і экспарта

Такім чынам для рэзервовага хавання дадзеных карыстальнікаў карыстальнікі з адпаведнымі правамі могуць рабіць экспарт дадзеных, каб у выпадку страты інфармацыі мець магчымаць імпарта з раней створаных файлаў.

### Пабудова сховішча дадзеных

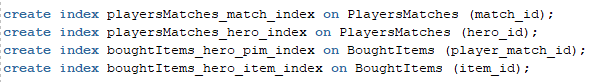
Апрача рэляцыйнай базы дадзеных у праекце выкарыстоўваецца дадатковая савакупнасць табліц, утвараючых сховішча дадзеных. Табліцы сховішча аддзелены ад табліц рэляцыйнай базы дзякуючы розным таблічным прасторам і карыстальнікам падключаемай базы дадзеных oracle.

#### База з гульнавымі дадзенымі

Схема табліц сховішча дадзеных паказана ў !дадатку.

Большасць табліц сховішча дадзеных супадае з адпаведнымі табліцамі ў рэляцыйнай базе дадзеных, аднак у сховішчы аб’яднаныя дзьве табліцы PlayersMatches і Matches, а таксама выдалены табліцы Players і PlayersMatchesTimeStat так як яны не ўдзельнічаюць у падліку аналітычных запытаў. Замест PlayersMatchesTimeStat у табліцу PlayersMatches дабаўляюцца новыя палі, дзякуючы якім можна атрымліваць больш аналітычнай інфармацыі. Запаўненне дадатковых палёў апісана ніжэй.

У дадатак да новых палёў табліц сховішча дадзеных ўтрымліаве таксама індэксы, для паскарэння пошука па табліцам, так як табліцы ў сховішчы знана большага аб’ёму, чым ў рэляцыйнай базе дадзеных. Створаныя індэксы паказаны на !малюнку.



* + - 1. Індэксы ў сховішчы

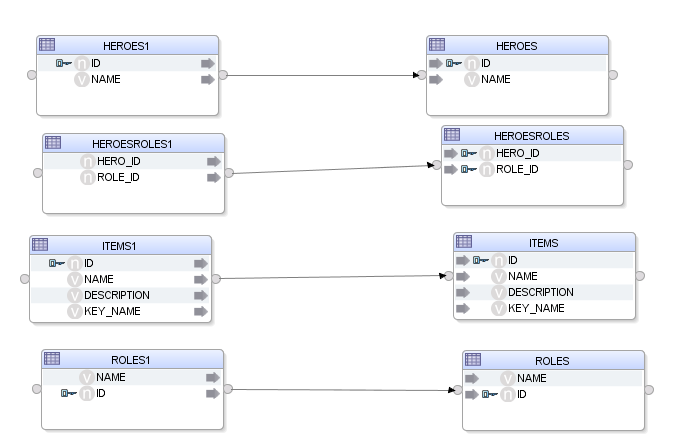
Індэксы свтораны па палям, якія выкарыстоўваюцца ў якасці ўмоў фільтрацыі пры выбарцы набораў радкоў з табліц.

#### Запаўненне сховішча дадзеных

Сховішча дадзеных аднаўляецца перыядычна, ў дадзеным праекце – кожны дзень. Усе гульнявыя дадзеныя, якія знаходзяцца ў рыляцыйнай базе дадзеных, заносяцца на сховішча. Сховішча дадзеных, як і працэс яго аднаўлення рэалізаваны сродкамі Oracle Data Integrator (ODI).

Працэс папаўнення сховішча дадзеных адбываецца па тэхналогіі Extract, Load and Transform (ELT), пад гэтай тэхналогіяй разумеецца атрыманне дадзеных з знешніх крыніц інфармацыі (у дадзеным выпадку гэта адзіная рэляцыйная база дадзеных), пасля атрымання дадзеных яны загружаюцца ў сховішча, дзе потым яны выкарыстоўваюцца ў вылічэннях для стварэння новых палёў табліц сховішча. Прамежкавыя дадзеныя, атрыманыя ў вылічэннях, захоўваюць у сховішчы, а самі пераўтварэнні адбываюцца згодна з згенераванымі сродкамі ODI запытамі.

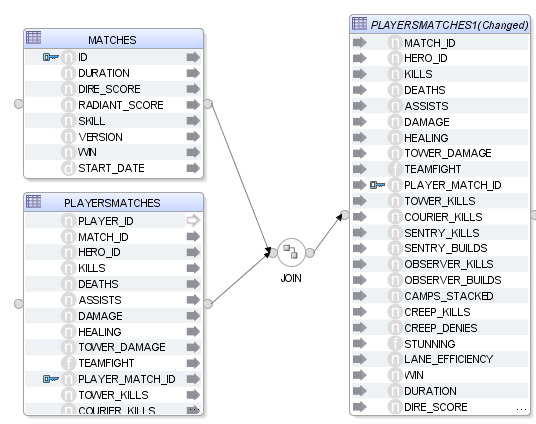
Аднаўленне сховішча адбываецца з аднаўлення табліц для рэдка змяняемых дадзеных, схема гэтага аднаўлення паказана на !малюнку.



* + - 1. Схема запаўнення рэдка змяняемых табліц сховішча

У дадзеным выпадку запаўненне табліц рэалізуецца як звычайнае капіяванне табліц з рэляцыйнай базы дадзеных, так як гэтыя дадзеныя не патрабуюць дадатковых вылічэнняў. Пры дабаўленні гэтых дадзеных у сховішча адпаведныя табліцы сховішча цалкам выдаляюць сваё змесціва.

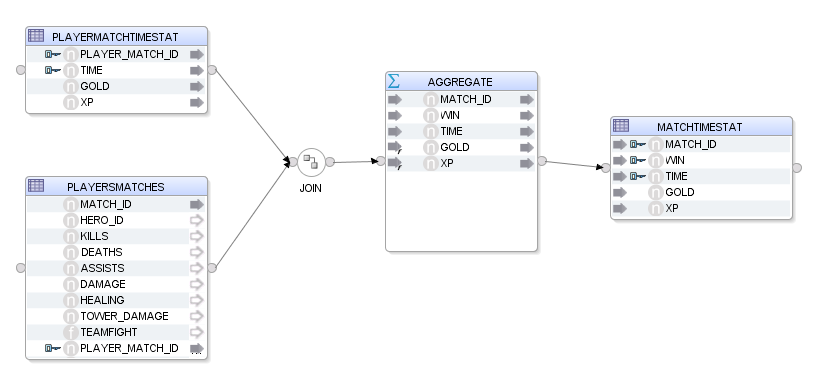
Пасля запаўнення рэдка аднаўляемых табліц запаўняецца табліца PlayersMatches як паказана на !малюнку.



* + - 1. Схема запаўнення табліцы PlayersMatches сховішча

Пад час запаўнення дзьве табліцы з рэляцыйнай базы дадзеных аб’ядноўваюцца і атрыманая плоская табліца заносіцца ў сховішча, без аднаўлення радкоў пры супадзенні ідэнтыфікатараў.

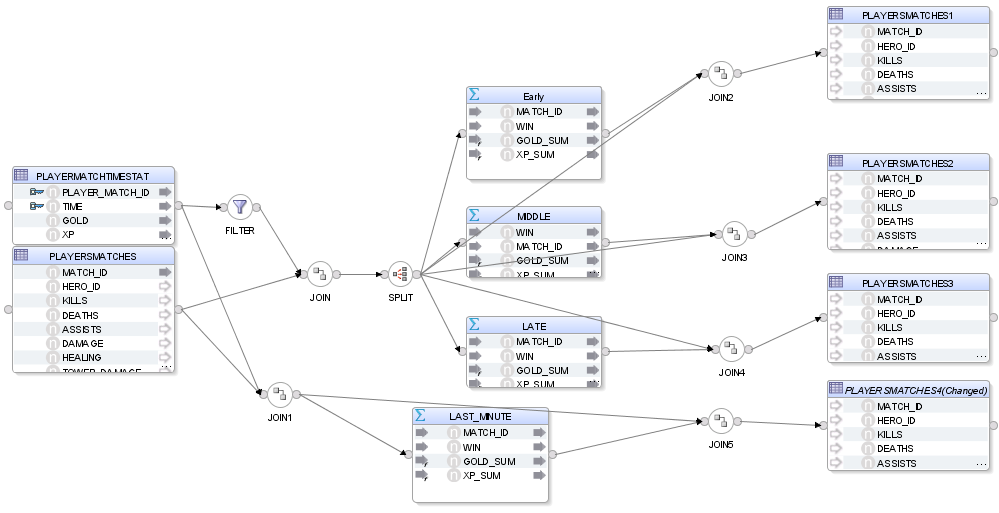
Пасля пераноса ўсёй неабходнай інфармацыі з рэляцыйнай базы дадзеных адбываецца вылічэнне і запаўненне асаттніх палёў табліцы PlayersMatches, які адбываецца ў два этапы. Спярша вылічваюцца і запаўняюцца палі аднолькавыя для гульцоў адной каманды ў адным матчы. Прыклад рэалізацыі запаўнення табліцы MatchTimeStat паказана на !малюнку.



* + - 1. Схема запаўнення табліцы MatchTimeStat

У дадзеным пераўтварэнні бяруцца дадзеныя з рэляцыйнай базы дадзеных у выглядзе таблі PlayersMatches і PlayersMatchesTimesStat, якія аб’ядноўваюцца, пасля чаго агрэгуюцца па пэўным палям і заносяцца ў прамежкавую табліцу MatchTimeStat, якая потым быдзе выкарыстоўвацца ў далейшым запаўненні сховішча дадзеных.

Пасля аднаўлення MatchTimeStat ідзе канчатковае запаўненне табліцы PlayersMatches, схема запаўнення паказана на !малюнку.



* + - 1. Схема канчатковага запаўнення табліцы PlayersMatches

У дадзеным этапе аб’ядноўваюцца дзьве табліцы з рэляцыйнай базы дадзеных, пасля чаго атрыманая табліца фільтруецца, падзяляецца і на падставе яе вылічваюцца агрэгатныя функцыі, якія служаць для выніковага запаўнення табліцы PlayersMatches.

Такім чынам пасля праходжання ўсіх пераўтварэнняў дадзеныя з рэляцыйнай базы заносяцца ў сховішча, пасля чаго яны могуць быць выдалены з базы дадзеных па ініцыятыве карыстальніка. Цяпер дадзеныя з сховішча дадзеных могуць быць выкарыстаны для выканання аналітычных запытаў.

### Распрацоўка вэб сервера

Для ўзаемадзеяння карыстальніка з прыладай быў распрацаваны вэб сервер, які апрацоўвае запыты карыстальніка, сфармуляваныя з дапамогай браўзера на баку кліента.

У якасці мовы праграмавання для рэалізацыі вэб сервера была выбрана java, разам з тэхналогіяй Spring па архітэктуры layered MVC.

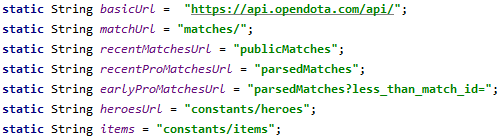
#### Узаемадзеянне з OpenDota

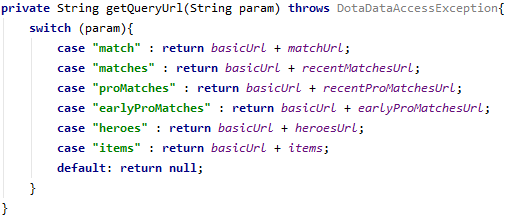
OpenDota выкарыстоўваецца як знешняя крыніца гульнявых дадзеных. Так як OpenDota з’яўляецца вэб серверам ўзаемадзеянне з ім адбываецца дзякуючы http запытам. Адпаведна з абранай архітэктурай ўзаемадзеянне ажыццяўляе слой сервісаў. Рэалізацыя адпраўкі запытаў на сервер OpenDota паказана на !малюнку.



* + - 1. Адпраўка запытаў да OpenDota

Для ўзаемадзеяння з http запытамі выкарыстоўваецца пакет org.springframework.http. Ствараецца аб’ект класа HttpEntity з вызначанымі загалоўкамі запыта пасля чаго яму надаецца спасылка url. Фарміраванне спасылкі паказана на !малюнку.

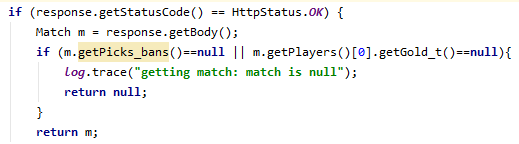




* + - 1. Рэалізацыя фарміравання спасылкі запытаў

Згодна з уваходным параметрам функцыя getQueryUrl фарміруе адпаведны url.

Большая частка работы па апрацоўцы адказаў з OpenDota бярэ на сябе выкарыстоўваемы пакет функцый. Рэалізацыя апрацоўкі адказаў паказана на !малюнку.



* + - 1. Рэалізацыя апрацоўкі адказаў з OpenDota

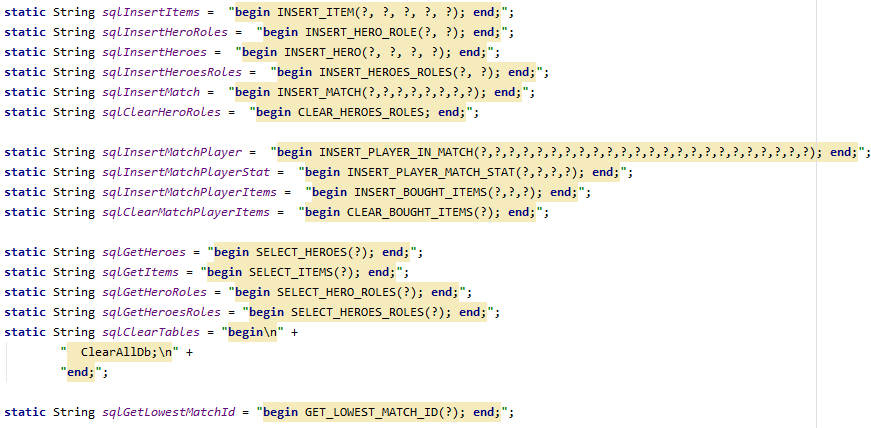
Дадзеныя ад сервера вяртаюцца ў фармаце JSON пасля чаго атрыманыя аб’екты аўтаматычна дэсэрыялізуюцца да аб’ектаў java. З-за таго, што сігнатуры аб’ектаў вяртаемыя серверам не заўсёды супадаюць з сігнатурамі выкарыстоўваемых унутраных класаў прылады былі створаныя прамежкавыя класы, якія ствараюцца дзякуючы адказам сервера і на падставе іх ствараюцца патрэбныя аб’кты ўнутраных класаў.

Такім чынам, дзякуючы загадзя прыгатаваным http запытам, адбываецца ўзаемадзеянне з OpenDota, аднак, з-за абмежаванняў сервера могуць ўзнікаць памылкі звязаныя з колькасцю адпраўляемых запытаў, у выніку чаго аперацыі атрымання дадзеных з сервера будуць прырывацца, што будзе патрабаваць паўторнага запыту выкліку функцый прылады. Таксама не ўсе дадзеныя, вяртаемыя з сервера, знаходзяцца ў правільным фармаце, таму перад занясеннем дадзеных ў базу дадзеных прылады адбываецца іх фільтрацыя.

#### Узаемадзеянне з базай дадзеных

Узаемадзеянне з унутранай базай дадзеных і сховішчам па большай часцы адбываецца дзякуючы працэдурам oracle. Узаемадзеянне на баку сервера рэалізавана на слаі рэпазіторыя, дзе і адбываецца асноўная апрацоўка памылак, вяртаемых з працэдур.

Для выканання запытаў спярша ствараецца злучэнне з базай дадзеных. Для гульнявой базы дадзеных, базы дадзеных з карыстальніцкімі дадзенымі і сховішча злучэнні розныя. Злучэнні ствараюцца дзякуючы драйверу OracleJDBC. Прыклады запытаў да гульнявой базы паказаны на !малюнку.



* + - 1. Запыты да гульнявой базы дадзеных

Для выкліку запытаў бярэцца статычны sql код, які запаўняецца параметрамі, пасля чаго перадаецца серверу oracle на выкананне. Прыклад рэалізацыі выкліка працэдуры для аднаўлення базы дадзеных паказаны у !дадатку. Пасля выкліка працэдуры, апрача апрацоўкі адказу, адбываецца апрацоўка памылак, пры іх наяўнасці. Пры наяўнасці памылак транзакцыя, у якой выконваўся запыт, адкаціцца, для чаго ў злучэнні, у якім адбываецца выклік працэдур, адключаны рэжым аўтападцвярджэння транзакцый. Калі памылак пад час выканання працэдур не было, то адбываецца захаванне транзакцыі.

Для атрымання дадзеных з базы выкарыстоўваюццы выходныя параметры працэдур. Прыклад рэалізацыі Апрацоўкі адказу з oracle сервера паказана ў !дадатку. Перад выклікам працэдуры ў запыце рэгіструецца выходны параметр тыпа курсор. Пасля выканання запыта, дзякуючы курсору, адбываецца перабор набор радкоў і пераўтварэнне атрыманых дадзеных ў аб’екты класаў java для далейшага ўзаемадзеяння з праграмай.

#### Выкананне аналітычных запытаў

Агульная схема працы прылады па выкананні аналітычных запытаў паказана ў !дадатку.

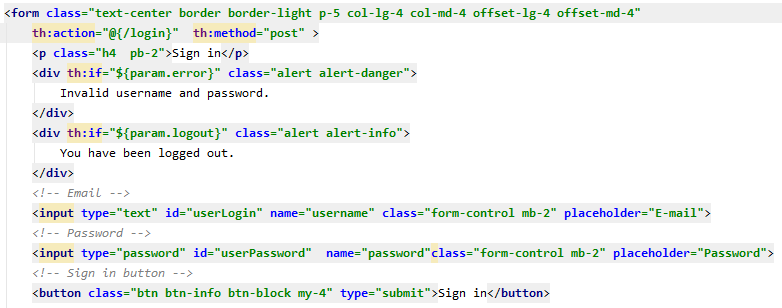
Для пабудовы графічных дыяграм неабходна атрымацаь адпаведныя дадзеныя з сховішча. Так як дыяграмы могуць патрабаваць мноства камбінацый разнастайных дадзеных выкананне аналітычных запытаў выконваецца без працэдур, а дзякуючы аператару select з адпаведнымі ўмовамі, фільтрамі і сартыроўкамі. Прыклад рэалізацыі фарміравання select запытана для пабудовы слупковай дыяграмы паказана ў !дадатку. У першую чаргу вызначаюцца палі, якія будуць вяртацца select запытам, гэта будуць значэнні восі абсцыс і ардынат а таксама колькасць матчаў для кожнай атрыманай кропкі, для магчымасці аналізу дакладнасці атрыманых статыстычных дадзеных. Пасля вызначэння выходных палёў вызначаюцца табліцы, з якіх будзе адбывацца выбарка. Калі ёсць неабходнасць браць дадзеныя з іншай табліцы, напрклад з-за ўмоў выбаркі альбо пэўнага віда дыяграмы, неабходныя табліцы аб’ядноўваюцца з асноўнай PlayersMatches па адпаведным палям. Пасля таго, як ўсе неабходны для дыяграмы дадзеныя былі аб’яднаныя ў адну табліцу адбываецца фільтрацыя дадзеных па пэўных ўмовах, прапісаных ў выбарках і дыяграмах. Пасля фільтрацыі атрыманыя дадзеныя групіруюцца па адпаведным палям з неабходнымі ўмовамі having. Атрыманы select запыт у выглядзе радка адпраўляецца на oracle сервер дзякуючы рэпазіторыю пасля чаго запыт выконваецца і вяртае вынікі для апрацоўкі ў выглядзе курсора для абыхода.

З-за фарміравання select запыта сродкамі java па зададзеным ад карыстальніка параметрам неабходна выключыць магчымасць sql injections, дзеля чаго параметры, уводзімыя карыстальнікамі, абмежаваныя такім чынам, што не могуць прымаць значэнні sql кода. Таму магчымасць ін’екцый выключана.

Пасля апрацоўкі атрыманых з сховішча дадзеных яны перасылаюцца кліенту ў якасці сэрыялізаванага ў фармат JSON аб’екта, які пасля дэсэрыялізацыі на баку кліента будзе пераўтвораны ў графічную дыяграму, выкарыстоўваючы падключаемую бібліятэку ChartJS. У залежнасці ад тыпа графіка перадаваемыя дадзеныя будуць адрознівацца.

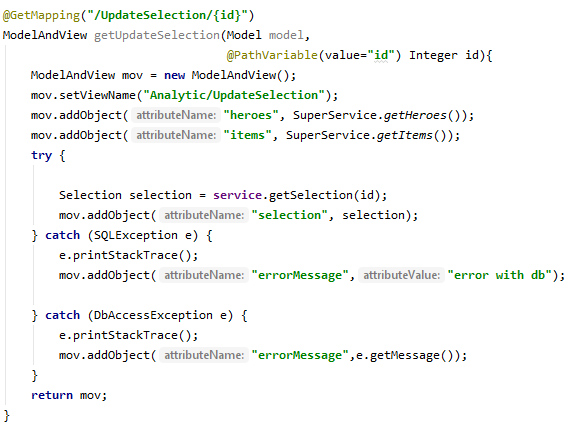
#### Апрацоўка запытаў карыстальнікаў

Карыстальнік узаемадзейнічае з прыладай дзякуючы http запытаў, якія фарміруюцца на баку кліента ў браўзеры. Для фарміравання запытаў выкарыстоўваюцца формы разам з шабланізатарам Theamleaf. Прыклад формы паказаны на !малюнку.



* + - 1. Прыклад формы для фарміравання запытаў

Уведзеныя дадзеныя ў палі формы змяшчаюцца ў запыт па ўказаным адрасе, пасля чаго дадзеныя апрацоўваюцца. Прыклад апрацоўкі запытаў з боку карыстальнікаў паказана на !малюнку.



* + - 1. Прыклад апрацоўкі запыта карыстальніка

Слой кантролераў, які прыдназначаны для апрацоўкі апытаў карыстальнікаў, прыводзяць увходныя параметры ў адпаведны фармат, пры неабходнасці, пасля чаго выклікаюць патрэбныя функцыі слоя сервіса. У дадзеных выпадку кантролер, атрымлівючы ідэнтыфікатар выбаркі для аднаўлення, атрымлівае дадзеныую выбарку дзяякуючы сервісу, пасля чаго вяртае карыстальніку старонку для рэдагаванне адпаведнай выбаркі.

## Абаснаванне тэхнічных прыёмаў праграмавання

#### Хаванне дадзеных

Для хавання дадзеных выкарыстоўваюцца рэляцыйныя базы дадзеных. Дзякуючы шырокаму распаўсюджванню выкарыстання базаў дадзеных сродкі для іх кіравання лёгка знайсці і ўжыць ў праекце. Таксама дзякуючы базам дадзеных выдаткі на працэсарны час і памяць пры захаванні і выкарыстанні дадзеных значна змяньшаюцца.

У якасці сістэмы кіравання базай дадзеных выкарыстоўваецца Oracle, так як ён падтрымлівае асноўныя магчымасці пабудовы базы дадзеных – табліцы, першасныя і другасныя ключы, абмежаванні палёў табліц, таблічныя прасторы, індэксы, карыстальнікі з рознымі правамі доступу для ўзаемадзеяння з табліцамі і працэдурамі.

Табліцы ў рэляцыйнай базе дадзеных нармалізаваныя, што дае выйгрыш у хуткасці выкарыстання і аб’ёме месца захавання дадзеных.

Сховішча дадзеных рэалізавана таксама ў выглядзе базы дадзеных для лёгкага ўзаемадзеяння з дадзенымі. Для памяньшэння часу выканання запытаў да дадзеных табліцы былі загадзя аб’аднаныя і былі дабаўлены дадатковыя індкэсы. Запаўняецца сховішча дзякуючы Oracle Data Integrator з тэхналогія ELT, так як ODI мае вялікі набор інструментаў для пабудовы і выкарыстання сховішча дадзеных. У дадзеным праекце ODI бярэ дадзеныя з базы дадзеных, пераўтварае іх і запаўяняе імі сховішча дадзеных. Таксама ODI дае магчымасць выконваць запаўненне табліц перыядычна.

Выкарыстанне ODI дае магчымасць далейшага развіцця праекта шляхам выкарыстання OLAP-куба ў быдучым, так як ODI утрымлівае ўбудаваныя магчымасці для пабудовы куба. Таксама дзякуючы ODI колькасць баз дадзеных, выкарыстоўваемых прыладай, можа павялічвацца без вялікіх зменаў у прыладзе.

Размеркаванне дадзеных дае большую гібкасць для выкарыстання праекта. Пры наўянасці некалькіх размеркаваных oracle сервераў праект дае магчымасць павялічыць эфектыўнасць ў хаванні і выкарыстанні дадзеных.

#### Вэб сервер

У якасці сродка пабудовы вэб сервера была выбрана тэхналогія Spring, так як выбраная тэхналогія шырока выкарыстоўваецца, што дае шмат магчымасцей для пабудовы прылады і балейшага яе развіцця. Дзякуючы сродкам, выкарыстоўваемымі Spring, такім як SpringSecurity, SpringContext, Mappings колькасць распрацоўваемага для праекта кода значна змяньшаецца, адпаведна змяньшаецца час распрацоўкі і складанасць аднаўлення праекта.

Аўтарызацыя і рэгістрацыя карыстальнікаў рэалізуецца Spring Security, дзякуючы простаму выкарыстанню, магчымасцю хэшыравання пароляў і аўтаматычнаму кантролю доступа да старонак згодна з правамі карыстальніка.

Архітэктурай распрацоўкі сервера была выбрана Layered MVC. Гэтая архітэктура деае размеркаванасць модуляў прылады, што павышае лёгкасць распрацоўкі модуляў і дае магчымасць лёгкага змянення асобных модуляў прылады не закранаючы ўсю прыладу цалкам. Згодна з архітэктурай функцыі апрацоўкі запытаў кліентаў, бізнэс-функцыі і функцыі ўзаемадзеяння з базай дадзеных выконваюцца класамі, змешчанымі ў розных архітэктурных слаях.

У якасці прадстаўленняў у архітэктуры MVC выкарыстоўваюцца HTML старонкі з Theamleaf. Такім чынам прадстаўленне будуецца дынамічна ў залежнасці ад уваходных параметраў, што таксама дае магчымасць фарміраваць гібкія http запыты на баку кліента.

Для графічнага адлюстравання пабудаваных дыяграм выкарыстоўваецца спецыялізаваная бібліятэка ChartJS. Дзякуючы шырокім магчымасцям пры пабудове графікаў і лёгкага выкарыстання графічнае адлюстраванне дыяграм цалкам выконваецца бібліятэкай.

## Тэставання праекта

### Тэставанне бяспекі

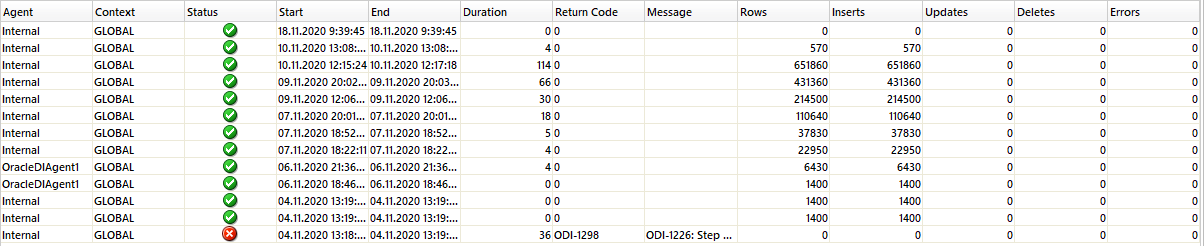
Для праверкі бяспекі прылады правяраўся доступ да магчымасцей прылады карыстальнікаў, якія ўвайшлі пад уліковымі запісамі з рознымі правамі доступа. Пры спробе задаць ў адрасе браўзера старонку, для якой аўтарызаваны карыстальнік не мае правоў, вэб сервера вяртае памылку. Усе запыты, якія адпраўляюцца на вэб сервер, патрабуюць токен аўтарызацыі, без якога нельга выканаць непрыдназначаных функцый. Так звычайны карыстальнік ўзаемадзейнічае з серверам толькі дзякуючы спасылкам на старонках і формам, ён не бачыць недаступных яму магчымасцей.

За бяспеку прылады адказвае кампанент Spring Security.

Бяспека базы дадзеных забяспечваецца немагчымасцю ін’екцый, таму небяспека для дадзеных можа існаваць толькі з-за не дастаткова бяспечных спосабаў выкарыстання базы дадзеных.

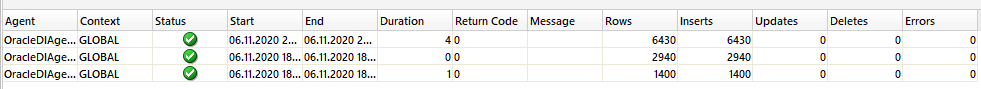
### Тэставанне сховішча

Так як за сховішча адказвае ODI былі пратэставаны асноўныя выкарыстоўваемыя функцыі ODI. Журнал правядзення імпарта дадзеных з базы ў сховішча прыведзены ў !малюнку.



* + - 1. Журнал выкананне імпарта ў сховішча

На малюнку бачна, што сховішча запаўняецца дадзенымі паспяхова, нават пры вялікіх аб’ёмах дадзеных. Вынікі выканання перыядычнага імпарта, ініцыяванага службай ODI паказана на !малюнку.



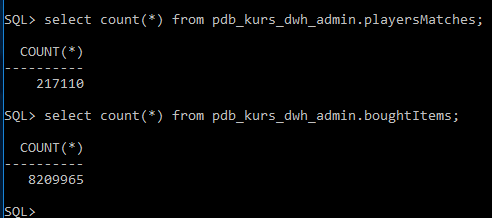
* + - 1. Журнал выканання перыядычнага імпарта

У журнале таксама відаць паспяховыя выкананні імпарту.

Такім чынам была праведзена праверка працаздольнасці праекта з ODI і з сховішчам дадзеных.

### Тэставанне з вялікім наборам дадзеных

Для прылады наяўнасць вялікіх набораў дадзеных з’яўляецца нормай. Колькасць радкоў некаторых табліц з сховіша дадзеных паказана на !малюнку.

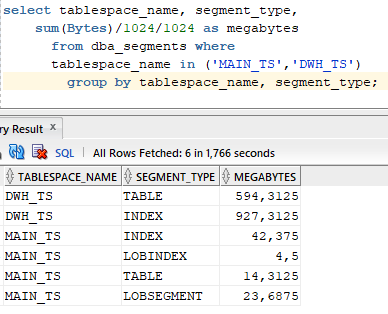


* + - 1. Колькасць радкоў табліц сховішча

Дзякуючы выкарыстанням індэксаў і прыстасаванай апрацоўцы oracle серверам вялікіх набораў дадзеных ўзаемадзеянне з сховішчам дадзеных адбываецца за задавальняючы перыяд часу, асаблівы ў выпадку выканання простых аналітычных запытаў.

Для тэставання максімальнага часу выканання аналітычнага запыту выкарыстоўваўся лінейная дыяграма з выбаркаў з мноствам параметраў. У выінку тэставання было вызначына, што час выканання запыта раўняецца каля 20 секунд, што з’яўляецца вялікай велічынёй, аднак такія складаныя запыты будуць рэдка выкарыстоўвацца. У сярэднім выкананне запытаў займае 9 секунд, што таксама шмат, аднак гэта выклікана вялікімі аб’ёмамі дадзеных. Хуткасць выканання таксама тлумачыцца абмежаванымі вытворчымі магчымасцямі сервера, які апрацоўвае сховішча дадзеных. Вышэй падлічаныя дадзеныя былі падлічаны для сервера, які выкарыстоўвае толькі некалькі ядзер працэсара з абмежаванымі магчымасцямі чытання/запісу на жорсткі дыск. Але трэба адзначыць, што вылічэнні адбываліся адразу пасля запуска прылады, што азначае адсутнасць дадзеных ў буферах сервера oracle, пад час рэальнай працы найбольш часта выкарыстоўваемыя дадзеныя для запытаў будзе захоўвацца ў больш хуткадаступнай памяці, што зменьшыць час выканання запытаў.

Аб’ёмы памяці, занятыя базай дадзеных і сховішчам паказаны на !малюнку.



* + - 1. Аб’мёы займаемага месца для базы і сховішча дадзеных

Ня гледзячы на тое, што сховішча захоўвае больш 20000 матчаў, займаемае ім месца адносна невялікае. Для базы дадзеных лічбы значна меньшыя, так як яны не захоўваюцць пастаянна інфармацыю пра матчы.

### Заключэнне

Пад час распрацоўкі вэб сервера, базы дадзеных і сховішча дадзеных праводзілася тэставанне на працаздольнасць распрацоўваемых модуляў прылады. Кожная працэдура ў базе дадзеных і функцыя ў вэб серверы адпавядае закладзеным патрабаванням.

Магчымасці кліента па ўзаемадзеянню з базай дадзеных і вэб серверам абмежаваныя формамі html, цераз якія ствараюцца запыты на сервер. Таму запыты былі пабудаваныя такім чынам, каб кліент не мог здзясняць забароненыя функцыі. За гэта адказвае Spring Security так і слой кантролераў увэб серверы.

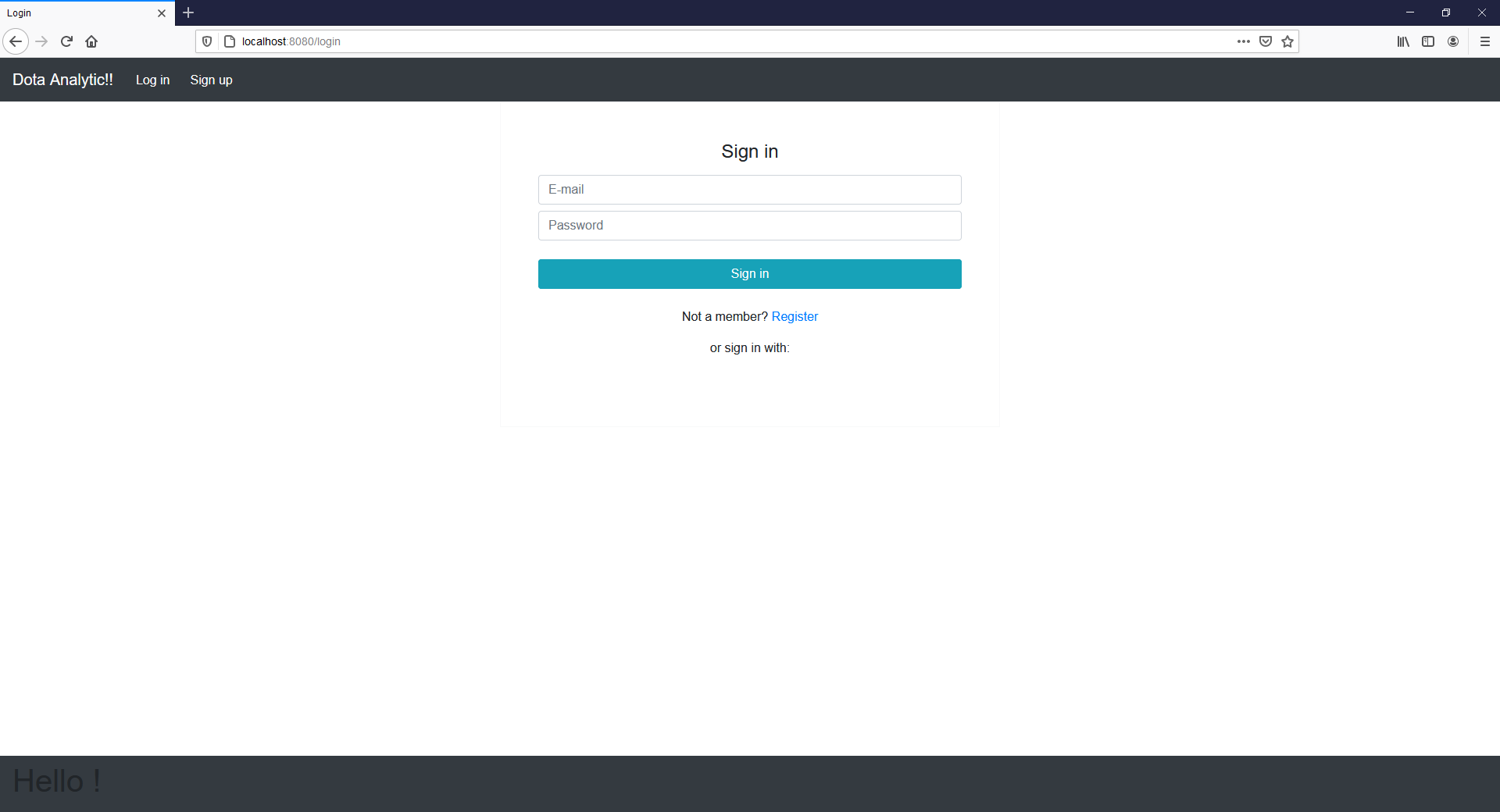
У выніку тэставання прылады была паказана яе працаздольнасць і адпаведнасць закладзеным умовам. Па-ранейшаму застаюцца магчымасці для злачынцаў у выглядзе DDOS атак і іншых. Таксама час выканання запытаў варты паляпшэння. Аднак праект выконвае закладзеныя на яго функцыі.

## Методыка выкарыстання праграмнага сродка

Так як карыстальнік узаемадзейнічае з прыладай цераз html старонку, у дадзеным раздзеле будзе разглядацца менавіта такое ўзаемадзеянне.

Для выкарыстання прыладай неабходна мець любы сучасны і папулярны браўзер, напрыклад Google Chrome.

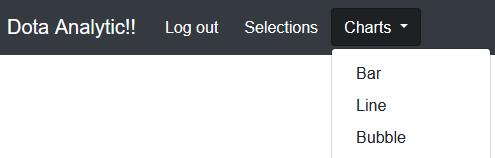
Каб пачаць карыстацца функцыямі прылады трэба ўвайці ў прыладу пад ўліковым запісам, альбо стварыць новы. Старонка аўтарызацыі паказана на !малюнку.



* + - 1. Старонка аўтарызацыі

Старонка аўтарызацыі з’яўляецца стандартнай. Пры ўвядзенні няправільных дадзеных будзе вяртацца памылка. Пасля ўвядзення правільных дадзеных – кліент будзе аўтарызаваны і зможа далей выкарыстоўваць прыладу.

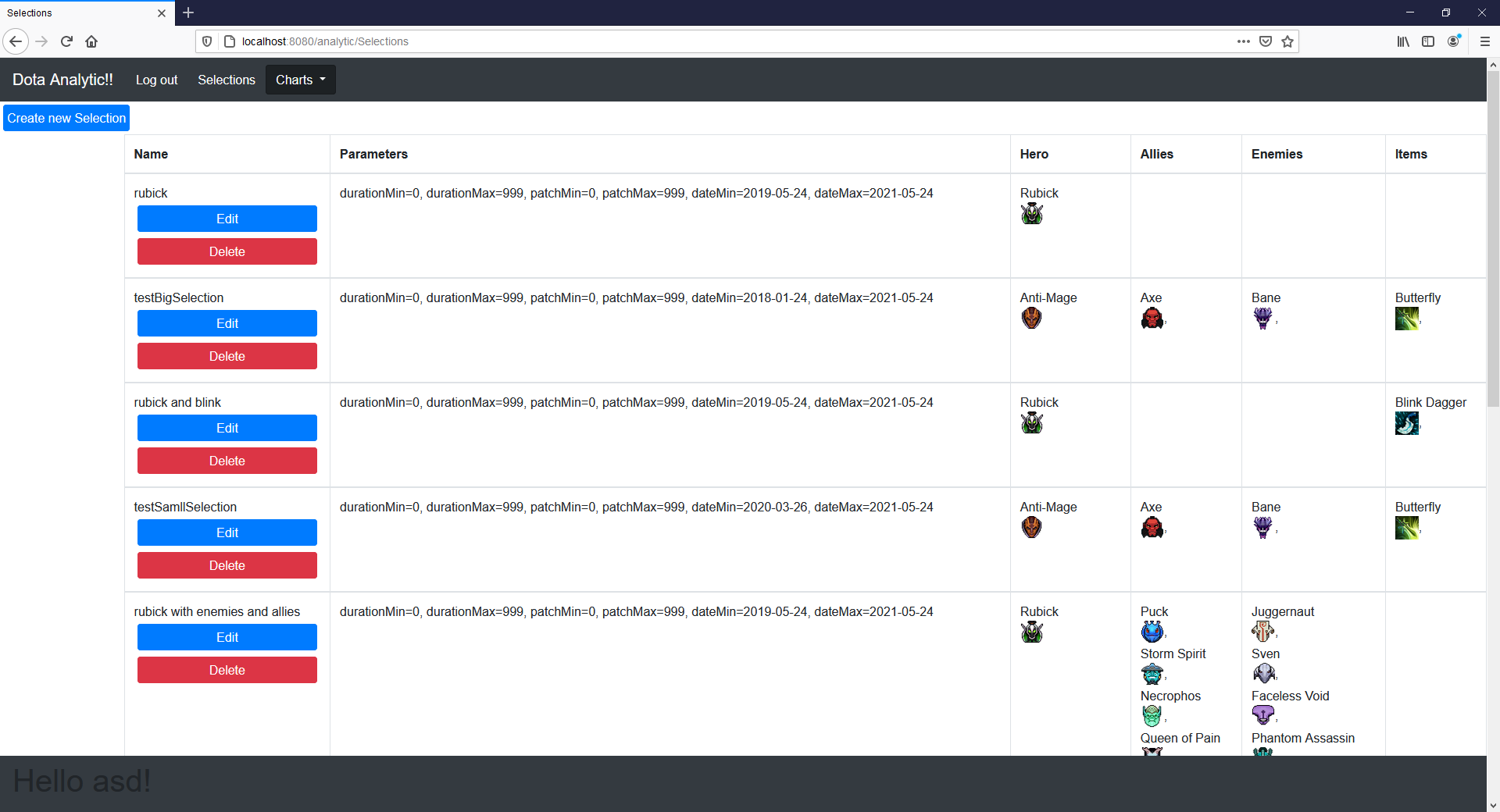
Меню з для ўзаемадзеяння звычайнага кліента з прыладай паказана на !малюнку.



* + - 1. Меню для ўзаемадзеяння

Карыстальніку даступна рэдагаванне выбарак, для чаго трэба націснуць кнопку “Selections”, і для рэдагавання розных відаў дыяграм, для чаго трэба націснуць на адпаведную кнопку выпадаючага меню.

Прыклад старонкі для прагляду выбарак карыстальніка паказаны на !малюнку.



* + - 1. Прыклад спіса выбарак для карыстальніка

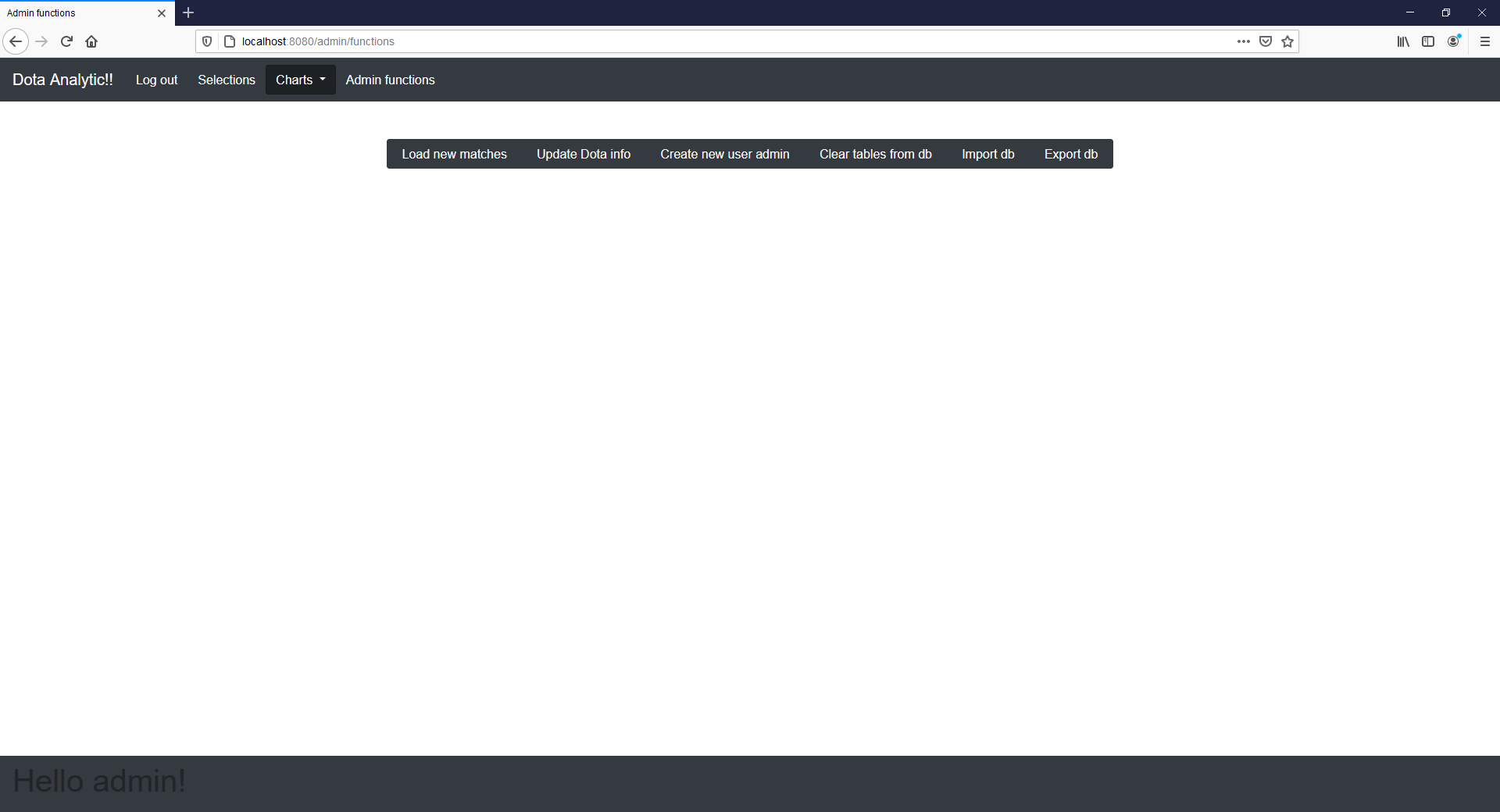
На гэтай старонцы карыстальнік бачыць створаныя ім выбаркі, інфармацыю аб іх і кнопкі для іх рэдагавання альбо стварэння новай выбаркі.

Для пабудовы дыяграмы неабходна стварыць патрэбныя выбаркі, пасля чаго стварыць патрэбную дыяграму, выбраўшы адпаведную опцыю з выпадаючага меню “Charts” на верхняй панэлі. Пасля стварэння дыяграмы і адпаведных выбарак можна графічна адлюстраваць створаную дыяграму, для чго трэба выбраць спіс дыяграм, пасля чаго націснуць кнопку “Build Chart”. Пасля чакання адказу ад сервера кліент убачыць дыяграму. Прыклад выгляду дыяграму паказаны на !малюнку.



* + - 1. Прыклад дыяграмы

У выпадку, калі кліент зайшоў з уліковым запісам адміністратара, кліент атрымае доступ да дадатковых функцый. Дадатковыя функцыі паказаны на !малюнку.



* + - 1. Функцыі адміністратара

Функцыі адміністратара:

* загрузіць новыя гульнявыя дадзеныя;
* аднавіць рэдка аднаўляемыя дадзеныя базы;
* стварыць новы ўліковы запіс адміністратара;
* Ачысціць табліцы гульнявых баз дадзеных, для вызвалення месца;
* Імпарт базы карыстальніцкіх дадзеных;
* Экспарт базы карыстальніцкіх дадзеных.

# Заключэнне

У выніку выканання работы была база дадзеных, сховішча дадзеных і java прылада ў якасці вэб сервера для ўзаемадзеяння з імі. Для сістэмы кіравання базамі дадзеных выкарыстоўваўся oracle сервер, што дае вялікію надзейнасць і вытворчасць узаемадзеяння з дадзенымі.

Для распрацоўкі сховішча дадзеных былі даследаваны некалькі рашэнняў праблемы вялікіх аб’ёмаў дадзеных, на падставе якіх плянуецца ствараць аналітычныя запыты. З-за асаблівасцей выбранай тэматыкі і адпаведным дадзеным рашэнне з выкарыстаннем OLAP-куба не ўжывалася, хаця пры правільнай рэалізацыі такое рашэнне магло даць патэнцыяльна большую вытворчасць пры апрацоўцы аналітычных запытаў, але ў гэтым выпадку не выкарыстоўваліся б некаторыя магчымасці OLAP-куба. Рашэнне з выкарыстанням звычайнай рэляцыйнай базы дадзеных таксама не выкарыстоўвалася, бо, нягледзячы на прастату рэалізацыі такога рашэння, яно не задавальняла закладзеным ўмовам, так як запыт выконваліся б значна даўжэй. У выніку ў рэалізацыі праекта выкарыстоўвалася сховішча дадзеных, што дае выйгрыш у вытворчасці і з’яўляецца лёгка рэалізуемым не выкарыстоўваючы лішнія функцыі.

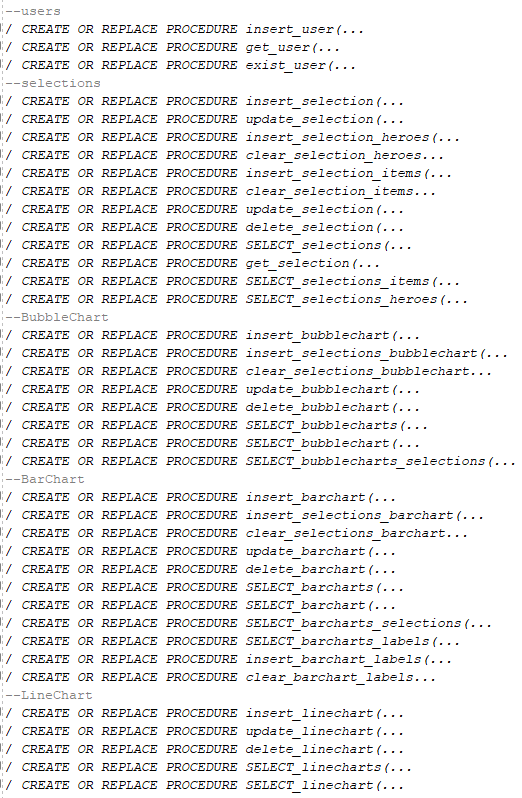
Распрацаваны праект мае вялікія патэнцыяльныя магчымасці для далейшага развіцця і дабаўлення новых магчымасцей, дзякуючы вырастынні oracle сервера, layered MVC архітэктуры і ODI для арганізацыі сховішча. Таксама для магымасці будучага пашырэння прылады былі дабаўлены дадатковыя табліцы і палі, якія пукль што не выкарыстоўваюцца.

Мэтай распрацоўкі праекта з дадзенай тэматыкай была магчымасць атрымання аналітычных дыяграм на падставе мноства рэяльных дадзеных. Такім чынам, праект дае магчымасць пабудоў разнастайных прагнозаў з высокай ступенню ўпэўненасці ў іх. Яшчэ адной прычынай стварэння дадзенага праекта была адсутнасць праектаў, якія рашалі б такія ж задачы. Такім чынам, распрацаваны праект можа быць натхняльнікам для распрацоўкі буйнамаштабнай прылады па аналізу дадзеных з гульні Dota2, улічваючы папулярнасць гульні, падобны праект мае вялікія перспектывы.

З наяўным праектам карыстальнік мае магчымасць атрымаць карысныя дадзеныя ў выглядзе графічных дыяграм. Улічваючы той факт, што раней гулец папулярнай гульні Dota2 не меў такіх магчымасцей, распрацоўка дадзенага праекта выглядае мэтазгоднай.

# Спіс літаратуры

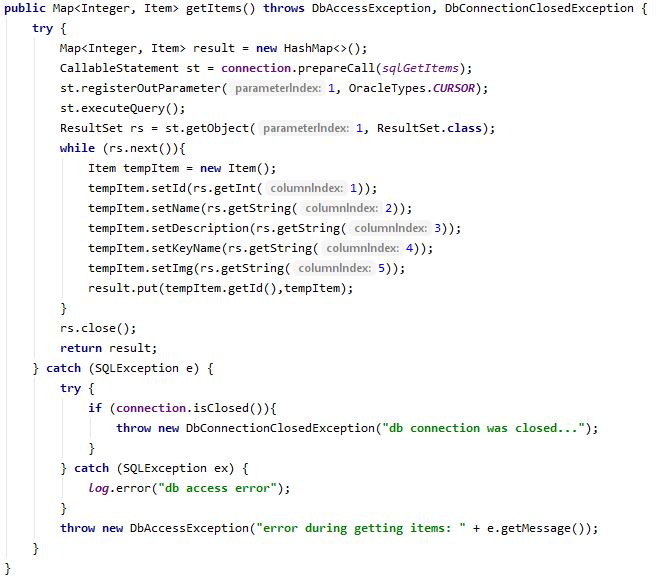
# Дадатак



Выклік працэдур для аднаўлення базы дадзеных



Апрацоўка запытаў бд



Фармірванне select-a

