KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programų inžinerijos katedra

Šarūnas Stanskis

TIESIOGINIO ANALITINIO APDOROJIMO KUBO MODELIO SEMANTINĖS ANALIZĖS METODO KŪRIMAS IR TYRIMAS

Programų sistemų inžinerijos magistro baigiamasis darbas

Vadovas:
Dr. Eduardas Bareiša

KAUNAS, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programų inžinerijos katedra

Šarūnas Stanskis

TIESIOGINIO ANALITINIO APDOROJIMO KUBO MODELIO SEMANTINĖS ANALIZĖS METODO KŪRIMAS IR TYRIMAS

Programų sistemų inžinerijos magistro baigiamasis darbas

Vadovas:
Dr. Eduardas Bareiša
2012-05-24

Recenzentas:
Dr. Kęstutis Kapočius
2012-05-24

Atliko:
Šarūnas Stanskis
2012-05-24

KAUNAS, 2012

Creation and research of semantic analysis method of OLAP cube model.

SUMMARY

Report generation is one of the most important functions when developing customer relations management systems. However, in order to display report it is needed to create a data entry form, describe SQL queries and create report templates. Because leaders want to see information about their sales, employees and clients in variety of ways it may require a lot of different reports to satisfy their needs and that in turn may require a lot of resources and time. In order to solve this problem, in was proposed to develop a method and tools to automate this tasks.

It was noticed that there arises another problem after report generation tool was developed. Testing team was manually testing all reports before report generation tool was developed. But it was noticed that after developing means to automate report generation quantity of possible generated reports drastically increased and testing all possible combinations became very difficult and time consuming task. In order to solve this problem it was decided to create a cube model semantic analysis method which would test id given report cube model are correct. Correctness of model syntax is ensured by XML editing tools so it is not concern, but even syntaxically correct model can generate invalid queries. As direct semantic model analysis would be very difficult task due to the fact that users may have some dynamic fields it would be difficult to detect related errors. So it was decided to test generated queries instead of testing model text. In order to ensure that tests can find client specific errors it was decided to use database management system as an oracle. Developed testing tool successfully detected problems in given testing model and based on statistically collected information few assumptions were made on how to improve this test.

Tiesioginio analitinio apdorojimo kubo modelio semantinės analizės metodo kūrimas ir tyrimas

Santrauka

Kuriant ryšių su klientais palaikymo sistemas viena iš svarbiausių sistemos funkcijų tenka ataskaitų generavimui. Tačiau tam, kad būtų galima parodyti ataskaitas reikia sukurti duomenų įvedimo formas, aprašyti SQL užklausas, bei sukurti ataskaitų šablonus. Kadangi vadovai nori pamatyti ir išnagrinėti su jų darbuotojais ir klientais susijusią informaciją įvairiais būdais tam gali prireikti labai daug ataskaitų ir to pasėkoje daug programuotojo ir testuotojo darbo valandų. Dėl to magistrinio projekto metu buvo iškelta idėja šį darbą automatizuoti .

Sukūrus komponentą buvo pastebėta, kad iškyla problema testuojant ar po modelyje padaromų pakeitimų viskas gerai veikia. Paties modelio sintaksės teisingumą užtikrina naudojamo XML kalba parašytų rinkmenų redagavimo įrankis, tačiau, tai, kad modelis yra sintaksiškai teisingas automatiškai nesąlygoja, kad iš jo bus sugeneruotos veikiančios užklausos. Dėl to, tam, kad įsitikinti pakeisto modelio teisingumu, testuojantis asmuo turėtų peržiūrėti visas galimas pasirinkimų kombinacijas, tačiau jei atsižvelgsime, kad pakeitimai gali būti padaryti ne viename modelyje, tai toks išsamus rankinis peržiūrėjimas kainuotų labai daug pastangų, bei užimtų labai daug laiko. Todėl tam, kad išspręsti šią problemą buvo sukurtas kubo modelio semantinės analizės metodas. Tačiau kadangi patį modelį analizuoti būtų labai sudėtinga ir būtų praleisti specifiniai individualių klientų sukurti požymiai, buvo pasirinkta analizuoti tik iš modelio sugeneruotas SQL užklausas. SQL užklausų analizei vietoj orakulo funkcijos buvo nuspręsta panaudoti duomenų bazių valdymo sistemą, taip užtikrinant, kad užklausos veiks su testuojamos duomenų bazės duomenimis. Pasinaudojus šiuo metodų buvo sukurtas testavimo įrankis sėkmingai aptiko modelyje esančias klaidas. Vykdant eksperimentą buvo apžvelgta kaip kinta aptinkamų klaidų kiekis nuo pasirinktus variantus sudarančių elementų aibės dydžio.

TURINYS

PAVEIKSLĖLIŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

FORMULIŲ SĄRAŠAS

1. ĮVADAS

Kuriant ryšių su klientais palaikymo sistemas viena iš svarbiausių sistemos funkcijų tenka ataskaitų generavimui. Tačiau tam, kad būtų galima parodyti ataskaitas reikia sukurti duomenų įvedimo formas, aprašyti SQL užklausas, bei sukurti ataskaitų šablonus. Kad šį procesą paspartinti ir palengvinti buvo suprojektuotas ir sukurtas ataskaitų generavimo komponentas. Sukūrus komponentą buvo pastebėta, kad iškyla problema testuojant ar po modelyje padaromų pakeitimų viskas gerai veikia. Įprastai, prieš sukūrus komponentą, toks testavimas buvo atliekamas rankiniu būdu. Testuotojas atsidarydavo sugeneruotas ataskaitas ir jas nagrinėdamas nuspręsdavo ar viskas gerai veikia. Tačiau automatizavus ataskaitų generavimą labai padaugėjo naudojamų ataskaitų ir pasidarė fiziškai nebeįmanoma patikrinti visas įmanomas kombinacijas, todėl buvo iškelta idėja sukurti modelio testavimo metodą ir pagal jį veikiantį įrankį. Paties modelio sintaksės tikrinti nereikia, nes jos teisingumą užtikrina naudojamas XML kalba parašytų rinkmenų redagavimo įrankis, tačiau, tai, kad modelis yra sintaksiškai teisingas automatiškai nesąlygoja, kad iš jo bus sugeneruotos veikiančios užklausos.

Toliau šis magistrinio darbo aprašas yra suskirstytas į tokius skyrius:

Analitinę dalį, kurioje yra apibudinamos naudojamos technologijos, nagrinėjami esantys įrankiai, kurių funkcionalumą būtų galima panaudoti, bei kitų autorių pasiūlyti tam tikrų sprendžiamo uždavinio sudėtinių dalių sprendimo būdai.

Projektinę dalį, kurioje yra nagrinėjamas sukurtas ataskaitų generavimo įrankis, bei ataskaitų kubo modelis.

Tyrimo dalį, kurioje yra plačiau nagrinėjama iškilusi problema, bei aprašomas pasirinktas konkretus problemos sprendimo būdas.

Eksperimentinę dalį, kurioje nagrinėjami pasirinkto sprendimo būdo rezultatai.

2. ANALITINĖ DALIS

2.1. Apibrėžimai

Testavimas – tai procesas kurio matu yra analizuojama programinė įranga bandant aptikti skirtumus tarp to kaip ji veikia ir kaip ji turėtų veikti pagal specifikaciją.[1]

Ataskaita – tai rašytinis dokumentas pateikiantis grafiškai ar lentelių pavidalu atvaizduotą informaciją. [2]

Užklausa – Užklausos tai yra pagrindinis metodas informacijai iš duomenų bazių nuskaityti. Daugelis duomenų bazių valdymo sistemų naudoja Struktūrinės Užklausų Kalbos (angl. SQL - Structured Query Language) standarto apibrėžiamą užklausų formatą. [3]

Tiesioginio analitinio apdorojimo kubas – Tiesioginio analitinio apdorojimo kubas (angl. OLAP – On Line Analytical Processing) kubas yra n-matė duomenų bazė, kuri yra optimizuota duomenų saugojimui ir greitam analitiniam apdorojimui.[4]

Duomenų bazė – Duomenų baze yra vadinama duomenų struktūra, kuri saugo organizuotą informaciją. Dauguma duomenų bazių turi keletą duomenų lentelę, kurios savyje turi keletą duomenų laukų. [5]

Matas – tai tam tikti kvantuojami duomenys. Skaitinė vertė kuri įdomi analizei. Nors matas nebūtinai privalo būti skaičius. Bet jo reikšmės privalo būti surikiuotos.

Dimensijos – formuoja kubo briaunas ir nusako mato kontekstą. Kiekvienas matas yra susietas bent su viena dimensija. Pavyzdžiui pardavimo laiko matas gali būti susietas su pardavimo diena, pardavimo mėnesiu, pardavimo metais ir t.t. Dimensijos yra organizuotos hierarchijomis. Pavyzdžiui diena -> mėnuo -> metai. [6][7]

Vienetas – programos sudėtinė dalis. Vienetų pavyzdžiais galėtų būti: metodas, klasė, procedūra,[8]

2.2. Programinės įrangos kokybė

Tam, kad kuriama programinė įranga sėkmingai atliktų savo funkcijas, yra būtinas programų kokybės užtikrinimas. Programinės įrangos kokybė yra vienas iš svarbiausių programinės įrangos kūrimo proceso sudėtinių dalių, kurios neužtikrinus kyla rizika, jog išaugs programų kūrimo kaštai, arba bus prarastas vartotojas.

Programinės įrangos kokybė ISO 9126 standarte[9] yra apibrėžiama šiais 6 rodikliais:

- Funkcionalumas

- Naudingumas
- Tinkamumas
- Patikimumas
- Palaikomumas
- Mobilumas.

2.3. Kokybės užtikrinimo būdai

Siekiant užtikrinti programinės įrangos kokybę stengiamasi ją testuoti, tai yra tikrinti ar sukurta programinė įranga veikia, be klaidų, bei atitinka vartotojo reikalavimus. Tam tikslui gali būti panaudojami įvairūs metodai pavyzdžiui: vienetų testai, integracinis testavimas, regresinis testavimas ir kitokie testavimo metodai. Išsamiau apie programų kokybę [10] šaltinyje.

Vienetų testavimas – individualių programos sudėtinių dalių, vienetų, testavimas. Vienetų pavyzdžiais galėtų būti: metodas, klasė, procedūra. Dažnai vienetų testai yra automatizuojami ir vykdomi po padarytų pakeitimų, siekiant įsitikinti, kad pakeistas programinis kodas vis dar veikia teisingai, bei kitose dalyse neatsirado nepageidaujamų šalutinių efektų. Vienetų testavimą dažnai vykdo programuotojai, kurie tą vienetą sukūrė.

Integracinio testavimo metu vienetai yra apjungiami tarpusavyje ir yra testuojama ar programinė įranga gerai veikia kaip kelių komponentų visuma.

Regresinio testavimo metu yra vykdomas testavimas panaudojant jau anksčiau įvykdytų testų duomenis ir palyginant gautus rezultatus su anksčiau gautais rezultatais.

Išsamiau apie testavimo tipus rašoma [11] šaltinyje

2.4. RDL kalba

Ataskaitų aprašymo kalba (angl. RDL – Report Definition Language) yra išplėstine žymų kalba (angl. XML - Extensible Markup Language) aprašytu modeliu atvaizduotas Microsoft SQL Serverio ataskaitų generavimo paslaugos naudojamas ataskaitos aprašas.[12] RDL kalba parašytoje rinkmenoje yra nurodoma:

- Informacija kaip gauti duomenis t.y. užklausos ir duomenų struktūrų aprašymas.
- Išdėstymas ir formatavimas t.y. informacija kuri aprašo kaip duomenys yra pateikiami.
- Papildomi vartotojo suskurti elementai. Pvz.: gali būti sukurtas ir įtrauktas požymis kuris nurodo kurių duomenų įvedimo laukų nerodyti.
- Ataskaitos savybės tokios kaip pavyzdžiui: autorius, parametrai, ataskaitoje naudojami paveikslėliai ir pan.

RDL kalba buvo sukurta tam, kad ją būtų galima naudoti kaip standartą aprašant ataskaitas ir ją palaikantys įrankiai galėtų atvaizduoti sukurtas ataskaitas nežinodami apie programos su kuria šios ataskaitos buvo sukurtos struktūrą.

Plačiau apie RDL kalbą nagrinėjama RDL specifikacijoje [13].

2.5. Duomenų kubo modelis ir galimi testavimo metodai

2.5.1. Kubo samprata

Tiesioginis analitinis apdorojimas (angl. OLAP – On Line Analytical Processing) yra metodas skirtas greitai apdoroti daugiadimensines analitines užklausas. Pavyzdžiui: dažnai asmuo analizuojantis kompanijos ataskaitas kaip pavyzdžiui pardavimus nori su jais susijusius duomenis pamatyti pavaizduotus įvairiais aspektais. Kaip pavyzdžiui pardavimų vadybininkas gali norėti pamatyti šiais metais įvykdytus pardavimus lentelės pavidalu, kurios viršuje būtų rodomo mėnesiai, o šone būtų kompanijos gaminami produktai. (2 lentelė)

1 lentelė. Ataskaitos pavyzdys.

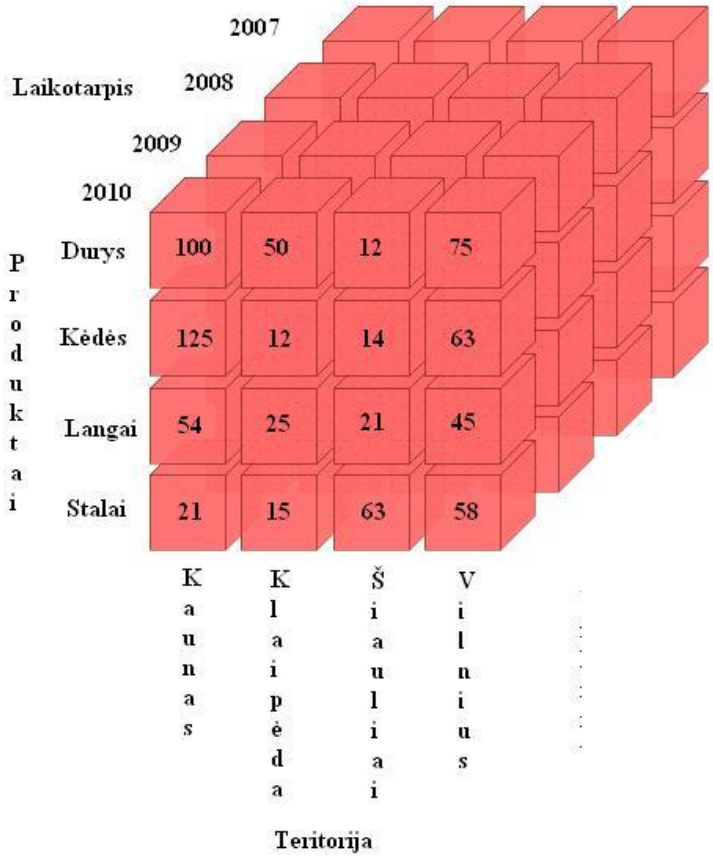
	Sausis	**Vasaris**	**Kovas**	**Balandis**	**Gegužė**	**Birželis**	**Liepa**	**Rugpjūtis**	**Rugsėjis**	**Spalis**	**Lapkritis**	**Gruodis**	***Suma***
Durys	1	2	4	5	5	6	4	5	3	2	1	1	*39*
Kedės	4	5	5	1	2	2	2	2	4	1	8	2	*38*
Langai	1	2	3	0	2	2	3	0	5	4	4	1	*27*
Stalai	1	2	3	1	4	3	4	1	3	5	6	3	*36*
Suma	*7*	*11*	*15*	*7*	*13*	*13*	*13*	*8*	*15*	*12*	*19*	*7*	***140***

Apžvelgęs tokiu pavidalu pateiktą informaciją jis gali norėti tuoj pat pamatyti šiuos duomenis kitokiu pavidalu pavyzdžiui atvaizduotus kaip lentelę kurios viršuje būtų produktai, o šone būtų pavaizduoti firmos padaliniai.

Kadangi gali būti dirbama su labai dideliais duomenų kiekiais tokių užklausų vykdymas su reliacinėje duomenų bazėje saugoma informacija gali užtrukti labai ilgai. O tuo atveju kai yra dirbama ir su istorine informacija pavyzdžiui imami ne tik naujausi duomenys, bet ir senesnė informacija tokių

užklausų apdorojimas gali trukti net iki kelių valandų ar dar ilgiau. O to pasėkoje bus švaistomas analitiko laikas.

Siekiant pagreitinti tokio pobūdžio užklausų apdorojimą yra naudojami tiesioginio analitinio apdorojimo kubai. Arba, kadangi kubas dažnai yra sudarytas iš daugiau nei trijų briaunų, jie kartais yra vadinami hyper kubais. Šio kubo briaunas sudaro dimensijos. Dimensija, tai yra vienas iš būdų apžvelgti duomenims. Galimų dimensijų pavyzdžiai: Prekės, Pardavimai, Kainos, Laikas, Pirkėjai ir pan.



1 pav. Kubo pavyzdys .

Vienas iš tiesioginio analitinio apdorojimo metodo trūkumų yra tas, kad duomenis privalo būti iš anksto apdoroti ir kur nors išsaugoti, o atsižvelgiant į duomenų kiekį jų saugojimui gali būti prireikti labai daug vietos. Taip pat prireikus iš naujo suskaičiuoti kurias nors reikšmes jų perskaičiavimas gali ilgai užtrukti. Dėl to tradiciniai tiesioginio analitinio apdorojimo kubai būna perskaičiuojami iš anksto ir pakeitimai juose yra vykdomi tik kai niekas jais nesinaudos, pavyzdžiui nakties metu. Iš to seka, kad duomenys saugomi kube yra sąlyginai pasenę, o kai kuriais atvejais verslui reikia, kad duomenys būtų kaip galima naujesni. Dėl to norint išspręsti šią problema buvo sukurta tiesioginio analitinio apdorojimo

kubų atmaina vadinama realaus laiko tiesioginio analitinio apdorojimo (angl. RT-OLAP - Real Time On Line Analytical Processing) kubais.

Realaus laiko tiesioginio analitinio apdorojimo kubai nuo tradicinių tiesioginio analitinio apdorojimo kubų skiriasi tuo, kad jie yra sudaromi realiu laiku. Dėl to, kad juose nėra naudojami išankstiniai duomenų suskaičiavimai užklausų vykdymo greitis gali smarkiai nukentėti, bet yra sutaupoma reikiamos vietos kiekis, bei analizuojamos informacijos pasikeitimai yra greitai įtraukiami. Taip pat realaus laiko tiesioginio analitinio apdorojimo kubų duomenys yra saugomi vartotojo kompiuterio operatyviojoje atmintyje ir dėl to šių kubų dydis yra apribotas prieinamos atminties dydžiu.

Plačiau apie tiesioginį analitinį apdorojimą nagrinėjama [14] šaltinyje.

2.5.2. XML kalbos bendrinis apibrėžimas

Išplėstinė žymų kalba (angl. XML - Extensible Markup Language) yra lankstus duomenų perdavimui skirtas formatas, kurį galima lengvai praplėsti savo žymėmis. Dėl tokio funkcionalumo yra labai patogu išplėstinę žymų kalbą naudoti modelių aprašymui. Be to kadangi dažnai XML kalba yra naudojama kaip duomenų apsikeitimo formatas, tai paverčia šią kalbą patrauklia panaudojimui kartu su tiesioginio analitinio apdorojimo įrankiais [15]. Kadangi tiesioginio analitinio apdorojimo kubas gali turėti sąsajų ne tik su duomenų baze, bet ir su daugeliu kitų informacijos šaltinių, kurie galbūt yra net nutolusioje sistemoje, kurios mes negalime valdyti, tai išplėstinė žymų kalba tampa labai patraukliu įrankiu. [16]

Išplėstinės žymų kalbos dokumentas pasižymi medžio struktūra. Dokumente būna vienas tėvinis elementas kuris savyje turi kitus vaikinius elementus. Paprasčiausias išplėstinės žymų kalbos dokumento pavyzdys būtų:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Dimensijos>
    <Dimensija>
        <Pavadinimas>Darbuotojai</Pavadinimas >
        <Lentelė>Employees</Lentelė>
        <VertimoŽyma> Darbuotojai</VertimoŽyma >
        <Stulpelis>drabuotojo_vardas +'_' + drabuotojo_pavardė <Stulpelis>
    </ Dimensija >
</ Dimensijos >
```

Jei šiai XML dokumento struktūrai sukuriame analogišką klasių hierarchiją informacijos iš modelio nuskaitymui ir į jį įrašymui galima būtų panaudoti Microsoft .NET karkaso teikiamą automatinį konvertavimo funkcionalumą.

Plačiau apie išplėstinė žymų kalbą [17] šaltinyje.

2.5.3. Kubo modeliavimas

Kadangi n-matis hyper kubas yra pagal susijusius ryšius į vieną visumą sujungta visa duomenų bazėje esanti informacija, galima šiuos ryšius aprašyti sudarant kubo modelį. Tam tikslui galima panaudoti išplėstinės žymų kalbos sintaksę ir tuo būdu apibrėžti kaip viena kubo dimensija siejasi su kitomis dimensijomis, t.y. viena duomenų bazės lentelė siejasi su kitomis. Šiam tikslui ypač pagelbsti tokios XML kalbos savybės, kaip hierarchinės ir lizdinės duomenų struktūros.[18]

Sukūrus tokį modelį, galima juo pasinaudoti ir automatiškai generuoti duomenų įvedimo formas, ataskaitas, bei SQL užklausas reikalingas duomenims užkrauti. Tokiu būdu galima sumažinti laiko sąnaudas reikalingas naujų ataskaitų įtraukimui ar senų ataskaitų modifikavimui.

Kubo modelio pavyzdys:

```
<?xml version="1.0" encoding="utf-8"?>
<ReportingModel name="Activities" xmlns="http://softdent.lt/ReportingModelSchema.xsd">
<Group name="CallGroup" IsRoot="true">
  <Dimension name="CallStatusDim" sourceTable="CallStatus" joinKey="CallStatusUUID">
    <Field Width="1.3" name="CallStatus"
    friendlyName="Status" friendlyNameTag="Status"
    sourceColumn="CallStatus" groupName="Activity"
    groupNameTag="Activity" index="90"/>
    <Parameter name="CallStatusParam"
     friendlyName="Status"  friendlyNameTag="Status"
    sourceColumn="CallStatus"  groupName="Activity"
    groupNameTag="Activity"  index="90"/>
  </Dimension>
</Group>
```

Kaip matome iš pavyzdžio kubo modelis yra sudarytas iš grupių, iš kurių bent viena yra šakninė. Grupės viduje yra dimensijos, kurios atitinka lenteles ar procedūras duomenų bazėje. Dimensijai yra nurodomas jos vardas, požymis pagal kurį bus išverčiamas pavadinimas į naudojamą kalbą. Šaltinis tai

yra duomenų bazės lentelė arba procedūros iškvietimo komanda. Dimensija savyje gali turėti kitas dimensijas ir pagal nutylėjimą jos tarpusavyje sujungiamos naudojant kairįjį išorinį sujungimą pagal *joinKey* parametre nurodytą stulpelį. Tačiau yra galimybė nurodyti specifinius sujungimo tipus panaudojant parametrą *customJoin*, kuriame toks jungimas būtų aprašytas.

Pavyzdys:

<Dimension name="CallStatusDim" sourceTable="CallStatus" joinKey="CallStatusUUID">

<Dimension name="CallSubStatusDim" sourceTable="CallSubStatus"

joinKey="CallSubStatusUUID"/>

</Dimensijon>

Būtų transformuota į:

CallStatus LEFT OUTER JOIN CallSubStatus ON CallStatus.CallSubStatusUUID= CallSubStatusDim .CallSubStatusUUID

Tuo tarpu

<Dimension name="CallStatusDim" sourceTable="CallStatus" joinKey="CallStatusUUID">

<Dimension name="CallSubStatusDim" sourceTable="CallSubStatus"

customJoin="CallSubStatusUUID AND IsMain='1'"/>

</Dimensijon>

Būtų transformuota į:

CallStatus LEFT OUTER JOIN CallSubStatus ON CallStatus.CallSubStatusUUID= CallSubStatusDim .CallSubStatusUUID AND IsMain='1'

Taip pat dimensijos turi laukus. Laukas nurodo galimą vartotojo pasirinkimą ir konkretų duomenų lentelės stulpelį kuriame saugoma vartotojui rodoma informacija. Šis stulpelis apsirašo parametre *sourceColum* ir užklausoje transformuojasi į pasirinkimų sąrašo elementą.

Kitas dimensijos elementas yra Parametras. Parametras nurodo galimą filtravimo informaciją ir kartu su pasirinktų reikšmių sąrašu transformuojasi į atitinkamą WHERE dalies sakinį. Parametro elementui yra nurodomi: pavadinimas, vertimui skirtas požymis, stulpelis iš kurio galima pasiimti reikiamus pasirinkimus, bei jei stulpelyje yra ne vartotojui rodomos reikšmės tarkim identifikatoriai, tai galima nurodyti kuriame stulpelyje yra vartotojui suprantama informacija.

Plačiau kubo modelio elementai aprašyti priede 9.1 Kubo modelio aprašymas, o išsamesnis kubo modelio pavyzdys pateiktas priede 9.3 Kubo modelio pavyzdys.

2.5.4. Kubo modelio testavimo principas

Bet kuri testavimo metodas turi remtis orakulu tam, kad galėtų nustatyti ar vykdomas veiksmas atliktas teisingai arba, kad yra klaidų. Taip pat tam, kad orakulas būtų naudingas jis turi būti mechaninis. Tai yra testavimo metodas negali visą laiką remtis žmogumi, kad jis visada sutikrins gautus rezultatus, nes jų gali būti daug, bei jie gali būti pernelyg sudėtingi, kad žmogus galėtu pateikti greitą įvertinimą.[19]

Tačiau apibrėžti modelio transformacijos orakulo funkciją yra labai sudėtinga vien dėl to, kad transformacijos rezultatas pats savaime gali būti labai sudėtingas. Norint ištestuoti transformacijos rezultatą reikia patikrinti daug struktūros sintaksinių savybių, bei ar rezultatas yra prasmingas semantiškai.[20] Norint supaprastinti testavimo problemos sprendimą galima kaip orakulą panaudoti duomenų bazių valdymo sistemą. Tokiu atveju galima apsiriboti tik SQL užklausų testavimu kuris būtų atliekamas pasitelkiant konkrečia duomenų bazę.

2.6. SQL užklausų analizė

2.6.1. Struktūrizuota užklausų kalba

Struktūrizuota užklausų kalba (angl. SQL – Structurized Query Language) yra ANSI standartizuota bendravimui su duomenų bazėmis skirta kalba. SQL leidžia gauti duomenis, papildyti, bei kitaip manipuliuoti duomenų bazių valdymo sistemose esančiais duomenimis. Tačiau nors SQL yra standartizuota, kiekvienas duomenų bazių valdymo programinės įrangos gamintojas yra į savo sistemas įdiegęs papildomų tarpusavyje nesuderinamų funkcijų, bei apribojimų. Pavyzdžiui Microsoft SQL Serveris nepalaiko užklausų kurias sudaro daugiau nei 4000 simbolių.

2.6.2. Užklausos struktūra

Procesas kurio metu iš duomenų bazės yra atsisiunčiami duomenys arba komanda su kuria yra atliekamas duomenų atsisiuntimo veiksmas yra vadinama užklausa. SQL sistemoje užklausoms apibrėžti yra naudojama SELECT komanda. Bendru atveju SELECT komandos sintaksė yra tokia:

SELECT pasirinkimų_sąrašas FROM lentelių_išraiškos [rūšiavimo_informacija]

Toliau trumpai apžvelgsime komandos sudėtines dalis:

Paprasčiausios užklausos pavyzdys būtų:

*SELECT * FROM lentelė1*

Jei duomenų bazėje egzistuos lentelė pavadinimu *lentelė1* ši užklausa gražins visus joje saugomus įrašus. Simbolis * nurodo, kad reikia gražinti visų lentelėje esančių stulpelių informaciją. Vietoj * galima būtų išvardinti konkrečius lentelės stulpelius kurių reikšmių mums reikia. Taip pat gali būti pasirenkami tam tikrų operacijų, atliekamų su nurodytuose stulpeliuose saugomais duomenimis, rezultatai. Pavyzdžiui:

SELECT pirmas_stulpelis, antras_stulpelis + trečias_stulpelis FROM lentelė1

Tokia užklausa gražintų pirmojo stulpelio reikšmę ir antrojo, bei trečiojo stulpelių reikšmių sumą.

Stulpelių pervardinimas

Kartais gali prireikti pervardinti stulpelius. Pavyzdžiui jei reikia išvesti kelis kartus to paties stulpelio reikšmę. Arba jei yra naudojama kokia nors išraiška ir jos rezultato stulpelis tada neturi pavadinimo. Tam tikslui pasiekti yra naudojamas raktažodis *AS*. Jis yra naudojamas po stulpelio pavadinimo arba po kokios nors išraiškos. Stulpelių pervadinimo pavyzdys būtų:

SELECT stulpelis1 AS 'Pirmas Stulpelis', antras_stulpelis + trečias_stulpelis AS Suma FROM lentelė1

Šiuo atveju gražinamuose rezultatuose *stulpelis1* būtų pavadintas „Pirmas stulpelis“, o antro ir trečiojo stulpelių reikšmių sumai, būtų suteiktas pavadinimas „Suma“.

Lentelių išraiškos

SQL užklausos FROM dalyje yra pateikiamas kableliais atskirtas lentelių nuorodų sąrašas.

FROM lentelės_nuoroda [,lentelės_nuoroda [, ...]]

Lentelės nuoroda gali būti lentelės pavadinimas, kita užklausa, lentelių sujungimas, arba ankščiau išvardintų būdų kombinacija. Jeigu FROM dalyje yra išvardinta daugiau negu viena lentelė, tai jos yra sujungiamos ir gražinama virtuali lentelė su kurios stulpeliais ir yra atliekami užklausos veiksmai.

Lentelių sujungimai

Virtuali sujungta lentelė, yra dviejų lentelių (realių arba virtualių) sujungimo, kuris atliekamas remiantis pasirinkto tipo taisyklėmis produktas. Gali būti keli lentelių sujungimo tipai: vidinis INNER, išorinis OUTER ir kryžminis CROSS. Sujunkimo pavyzdys būtų:

Lentelė1 INNER JOIN Lentelė2 ON Lentelė1.Stulpelis1= Lentelė2.Stulpelis2

Vidiniu sujungimu vadinamas toks sujungimas kurį atlikus su dviem lentelėmis gauname trečią lentelę, kurioje yra visos sujungtos eilutės tenkinančios sujungimo sąlygą.

Vykdant išorinį sujungimą reikia papildomai nurodyti ar tai kairysis ar dešinysis sujungimai. Jei yra pasirinktas kairysis sujungimo tipas, tai sujungimo rezultato lentelėje būna visos sujungtos eilutės tenkinančios sujungimo sąlygą, bei visos pirmosios lentelės eilutės, kurios sujungimo sąlygos netenkina ir į šių eilučių stulpelius kuriuose turėtų būti įrašai iš antrosios lentelės įrašomos tuščios reikšmės. Analogiškai kairiajam sujungimui veikia dešinysis sujungimas, tik į rezultato lentelę yra įtraukiamos antrosios lentelės eilutės kurios netenkina sujungimo sąlygos.

Vykdant kryžminį sujungimą yra atliekami kairysis ir dešinysis sujungimai. Tai sujungimo rezultato lentelėje būna visos sujungtos eilutės tenkinančios sujungimo sąlygą, bei visos pirmosios lentelės eilutės, kurios sujungimo sąlygos netenkina ir į šių eilučių stulpelius kuriuose turėtų būti įrašai iš antrosios lentelės įrašomos tuščios reikšmės, bei visos antrosios lentelės eilutės, kurios sujungimo sąlygos netenkina ir į šių eilučių stulpelius kuriuose turėtų būti įrašai iš pirmosios lentelės įrašomos tuščios reikšmės.

Duomenų filtravimas

Labai dažnai yra norima, kad užklausa gražintų tik tuos duomenis, kurie atitinka tam tikrus kriterijus. Tokiu atveju yra naudojamas filtravimas įrašant filtravimo sąlygą užklausos WHERE dalyje.

SELECT pirmas_stulpelis, antras_stulpelis + trečias_stulpelis FROM lentelė1 WHERE antras_stulpelis > ‘0‘ AND trečias_stulpelis > ‘0‘

Filtravimo sąlygoje nusirodo loginės operacijos kurios gražina rezultatą taip arbe ne, Galimos loginės operacijos:

- =, >, <, <=, >= gražina atitinkamą palyginimo operacijos rezultatą

- IS (NOT) NULL yra tenkinama jei nurodytam stulpeliui ar kintamajam reikšmė nėra (yra) priskirta
- IN (reikšmė1, reikšmė2, reikšmė3,...) yra tenkinama jei nurodyto stulpelio ar kintamojo reikšmė yra tarp išvardintų reikšmių.

Grupavimas

Kartais prireikia, kad rezultatų eilutės būtų sugrupuotos. Kada užklausos SELECT dalyje yra naudojamos funkcijos pavyzdžiui suma SUM(), vidurkis AVERAGE() ir pan. grupavimas yra būtinas, nes jis nurodo kurias reikšmes nurodytas veiksmas apims. Grupavimas nurodomas užklausos gale pridėjus raktažodį GROUP BY ir išvardinus stulpelius. Pastaba: jei stulpelis buvo pervardintas arba yra naudojama išraiška, grupavimo dalyje vis tiek yra naudojamas originalus vardas ar išraiška.

Plačiau apie SQL [21] šaltinyje.

2.7. Galimos sugeneruotų SQL užklausų klaidos

Gali būti sukurta daug SQL užklausų, kurios bus sintaksiškai teisingos, tačiau gali būti semantiškai neteisingos.[22] Dėl to apžvelgsime keltą aktualių galimų semantinių ir sintaksinių klaidų, kurios gali iškilti generuojant iš modelio užklausas:.

2 lentelė. Galimų užklausų klaidų apžvalga.

Klaida	**Šaltinis**
Klaidingi stulpelių pavadinimai	Modelio lauko informacijos šaltinio stulpelio atribute blogai nurodytas stulpelio pavadinimas.
Klaidingi lentelių pavadinimai	Modelio dimensijos šaltinio lentelės atribute blogai nurodytas lentelės pavadinimas ar kreipinys į serveryje saugomą procedūrą.
Trūkstamos ar blogai parašytos sujungimo sąlygos	Modelio dimensijos apraše blogai nurodytas sujungimo tipas arba blogai parašyta specifinis sujungimo sakinys.
Sutampantys pavadinimai	Modelyje *tableAlias* atribute arba kaip šaltinio stulpelis ar lentelė yra nurodytas kur nors jau panaudotas pavadinimas.
Netinkamai naudojamos konversijos	Formulėje bandoma konvertuoti netinkamus duomenų tipus.
Aritmetinės klaidos	Šaltinio atribute naudojamose formulėse esančios klaidos pavyzdžiui formulėse nepatikrinama ar nėra dalinama iš nulio.

Detalesnis galimų SQL klaidų sąrašas pateikiamas [23] šaltinyje.

2.8. SQL užklausų testavimo metodų apžvalga

SQL užklausų tikrinimas panaudojant statinės analizės įrankį

Vienas iš galimų dinamiškai sugeneruotų SQL užklausų testavimo būdų būtų panaudoti statine sakinių analizę pasiūlytą [24] šaltinyje. Pagal šį metodą pirma reiktų atlikti užklausų analizę ir sukurti galimų žymų aprašą iš kurio reikia sudaryti baigtinį automatą ir panaudoti modifikuotą nuo konteksto nepriklausomų kalbų pasiekiamumo algoritmą. Siūloma sudaryti tokių galimų žymų sąrašą į kurį įeitų:

- SQL kalbos raktažodžiai,
- Duomenų bazės objektų pavadinimai,
- Specifiniai simboliai,
- Skaitmenys, sakiniai.

Plačiau apie šį metodą rašoma [25] šaltinyje.

SQL sintaksės validavimas pasinaudojus Microsoft SQL serverio komandomis

Darbo su duomenų baze įrankis Microsoft SQL Server Management Studio turi galimybę patikrinti ar parašyta SQL užklausa yra sintaksiškai teisinga ir gali būti įvykdyta pasirinktoje duomenų bazėje. Nors SQL užklausų vykdymas ir tikrinimas vyksta naudojant šio įrankio grafinę sąsają, pats įrankis užklausų netikrina, o tikrina šias užklausas serveryje. Prieš vykdydamas testuojamos SQL užklausos tekstą įrankis įvykdo komandą „*SET NOEXEC ON*“ , o pabaigęs vykdyti tikrinamą kodą įvykdo komandas „*SET NOEXEC OFF*“ ir „*SET PARSEONLY OFF*“. Kadangi pats testavimas vyksta SQL serveryje, tai šio įrankio vykdomą metodą galima atkartoti bet kurioje programoje kuri turi galimybę prisijungti prie SQL serverio ir įvykdyti užklausą.

Komandų paaiškinimai:

- *SET NOEXEC* – Sukompiliuoja SQL sakinius, tačiau jų neįvykdo.
- *SET PARSEONLY* – Patikrina SQL sakinių sintaksę ir gražina klaidų pranešimus tų sakinių neįvykdydamas.

Pavyzdys testuojant komandą „*SELECT * FROM lentelė1*“:

```
SET NOEXEC ON
* FROM lentelė1
```

SET NOEXEC OFF

SET PARSEONLY OFF

Plačiau šis metodas aprašytas [26] šaltinyje.

2.9. Išvados

Atlikus srities analizę galima padaryti tokias išvadas:

1. Panaudojus XML kalbą galima sėkmingai aprašyti duomenų kubo modelį kuriuo pasinaudojant galima generuoti kitus nuo modelio priklausomus komponentus: užklausas, ataskaitas, duomenų įvedimo formas.
2. Sukurto XML modelio sintaksę įprastai automatiškai patikrina teksto redaktorius su kuriuo yra modelis redaguojamas, tačiau validi sintaksė neužtikrina, kad bus gerai sugeneruotos užklausos.
3. Kadangi tiesiogiai tikrinti modelio tekstą gali būti sudėtinga, bei toks metodas nebūtinai užtikrins, kad testuojamas užklausų generavimo komponentas sugeneruos identiškas užklausas buvo pasirinkta sugeneruotas užklausas testuoti SQL serveryje.
4. Dėl to, kad nėra jau sukurtų tinkamų bibliotekų tenka pasinaudoti vienu iš kelių internete pateikiamų sprendimų ir parašyti SQL užklausų sintaksę tikrinantį, komponentą,
5. Dauguma sprendimų siūlo suformuotas užklausas panaudoti standartinį c# programavimo kalbos teikiamą klaidų gaudymo mechanizmą ir tiesiog bandyti įvykdyti SQL užklausas, tačiau kadangi, gali būti, jog bus bandoma modeli sutikrinti su klientų turimomis duomenų bazėmis, bet kokia duomenų praradimo rizika yra nepriimtina, dėl to pasirinktas metodas kuris išjungia užklausų vykdymą.

3. PROJEKTINĖ DALIS

Šioje dalyje apžvelgsime magistratūros studijų metu sukurtą programinę įrangą – Ataskaitų ir įvedimo formų generavimo komponentą.

3.1. Sukurta sistema

Vykdant magistrinį projektą buvo sukurtas ataskaitų generavimo komponentas.

3.1.1. Projekto tikslas ir adresatas

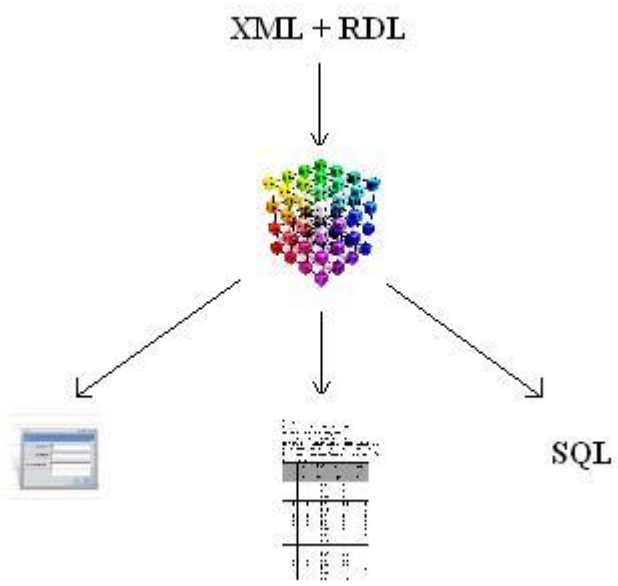
Lietuvoje ir užsienio šalyse veikia daug farmacijos įmonių, kurios siūlo labai didelę įvairovę medikamentų. Šios įmonės samdo reklamos agentus, kad jie pristatinėtų jų gaminama produkciją, rengtų seminarus medicinos tematika, vykdytų akcijas (pasiūlytų nemokamų naujų mėginių), vykdytų apklausas. Tam, kad sėkmingai šią veiklą organizuoti įmonėms reikia turėti galimybę gauti veiklos ataskaitas.

Ataskaitų sudarymas, tai yra viena iš dažniausiai verslo atstovų pageidaujamų funkcijų, kuri yra įgyvendinama daugelyje verslo sektoriuje naudojamų programų. Kaip keletą iš pavyzdžių galima paminėti dažniausiai verslo atstovus dominančias pardavimų ar išlaidų ataskaitas.

Duomenys šioms ataskaitoms yra gaunami iš duomenų bazių prieš tai paprašius vartotoją įvesti tam tikrą informaciją. Pavyzdžiui laikotarpį, prekes, darbuotojus ir t.t. Tam įvedimui yra kuriamos atskyros formos, rašomos užklausos. Šio magistrinio darbo metu bus sukurtas generatorius kuris perskaitys duomenis iš RDL tipo rinkmenos ir ją atitinkančios XML duomenų schemos ir išgaus:

- Prisijungimo duomenis
- Ataskaitos išvaizdą
- Duomenų struktūras
- Užklausas reikalingas duomenims išgauti
- Užklausoms reikalingus parametrus kuriuos turi įvesti vartotojas

Gavęs šiuos duomenis generatorius susieja reikiamų įvesti duomenų tipus su C# WindowsForms esamais įvedimo laukais ir sugeneruoja formą, kad vartotojas galėtų įvesti duomenis. Vartotojui įvedus duomenis komponentas turės juos nuskaityti ir sugeneruoti SQL užklausas. Tada gavęs iš serverio duomenis komponentas turės sugeneruoti ataskaitą ir parodyti ją vartotojui.

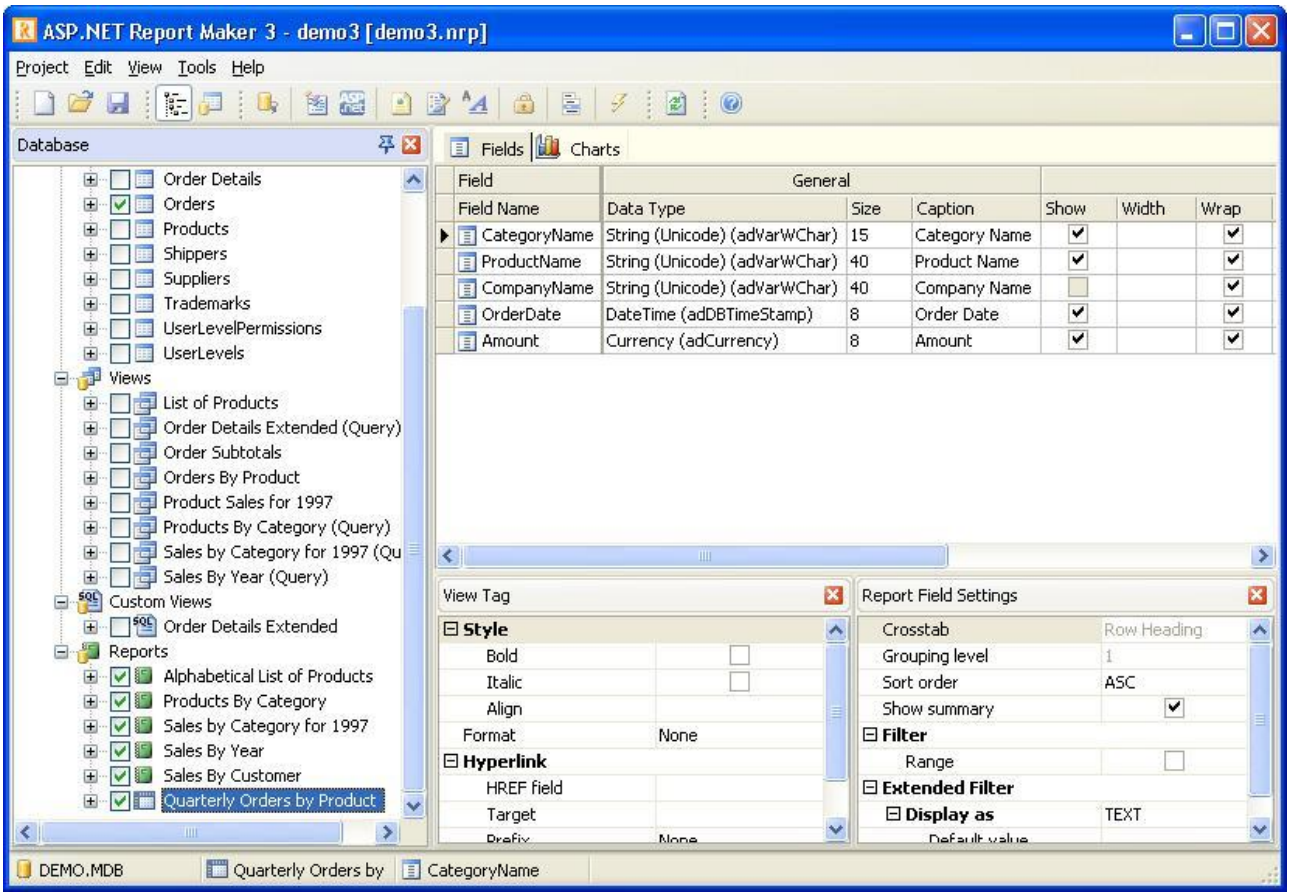


2 pav. Komponento paskirtis

Taigi automatizuodamas dalį dažnai pasitaikančių užduočių šis komponentas sutaupys laiko ir išteklių reikalingų sukurti programinei įrangai.

3.1.2. Problemos sprendimas pasaulyje

Report Maker 3



3 pav. ASP.NET Report Maker 3 išvaizda.

Kūrėjas: Hkvstore

Produktas: ASP.NET Report Maker 3

Internetinės svetainės adresas: http://www.hkvstore.com/aspnetreportmaker/

Apžvalga:

Kompanijos Hkvstore sukurtas ataskaitų demonstravimo įrankis ASP.NET Report Maker 3 yra skirtas sugeneruoti ASP.NET (C# arba VB.NET) ataskaitas panaudojant duomenis iš Microsoft Access duomenų bazės ar kitokio ADO (ActiveX duomenų objekto) duomenų šaltinio. Su šiuo įrankiu galima sukurti detalias ar suvestinių ataskaitas internetiniams puslapiams. Taip pat šis įrankis gali piešti

įvairiausius grafikus. Šio įrankio sugeneruotas kodas yra parašytas gryna ASP.NET kalba, tad tam kad jis veiktų nereikia papildomai instaliuoti papildomų dinaminių bibliotekų.

Catchysoft Report Generator

Kūrėjas: Catchysoft

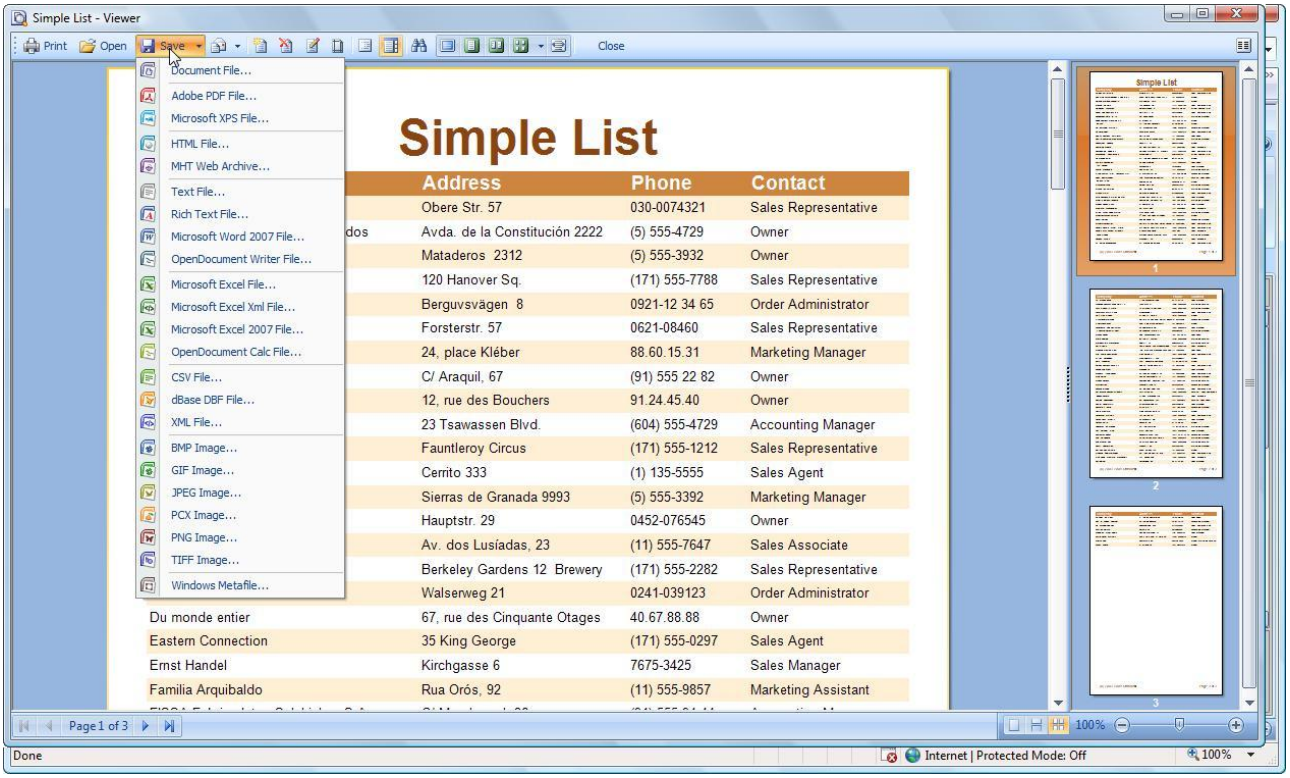
Produktas: Catchysoft Report Generator

Internetinės svetainės adresas: http://www.catchysoft.com/report_generator_pro.html

Apžvalga:

Kompanijos Catchysoft produktas Catchysoft Report Generator Pro yra ataskaitų generavimo biblioteka, kuri panaudoja ActiveX (dar žinomą kaip COM) technologiją. Todėl ataskaitų generavimo galimybės gali būti panaudotos iš bet kurios programavimo kalbos, kuri tik gali panaudoti ActiveX komponentus. Unikali šios bibliotekos savybė yra ta, kad ją galima lengvai ir greitai integruoti į bet kurį projektą. Taip pat šis produktas turi automatinio ataskaitos turinio išdėstymo galimybę, kuri leidžia greitai sukurti ataskaitos struktūrą.

Stimulsoft Reports Designer



4 pav. Stimulsoft Reports.Net yra .NET išvaizda.

Kūrėjas: Stimulsoft

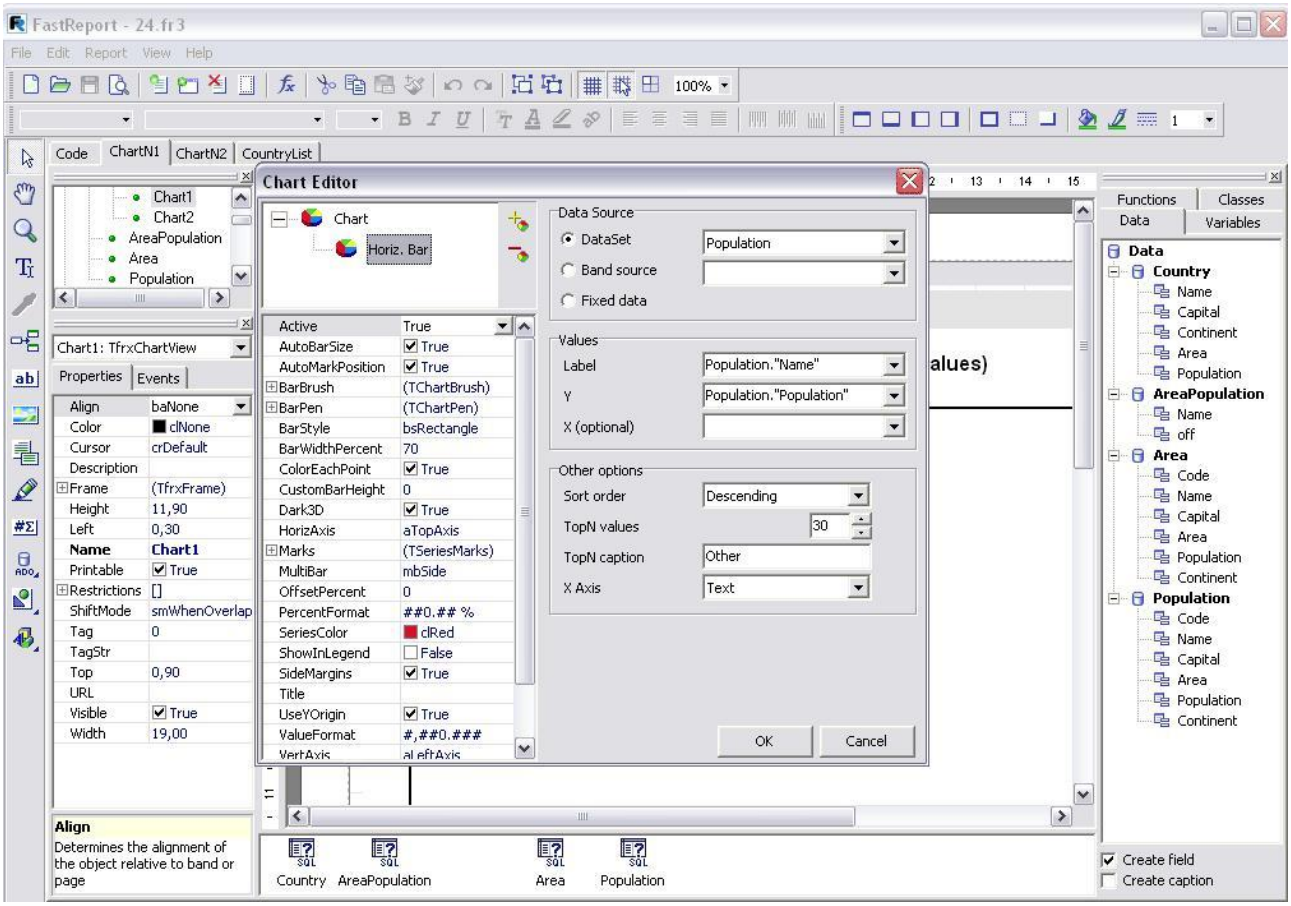
Produktas: Stimulsoft Reports.Net yra .NET

Internetinės svetainės adresas: http://www.stimulsoft.com/ReportsNet.aspx

Apžvalga:

Kompanijos Stimulsoft produktas Stimulsoft Reports.Net yra .NET technologija paremtas ataskaitų generatorius kuris padeda sukurti lanksčias ataskaitas. Visos ataskaitos yra sukuriamos ataskaitų kūrimo įrankiu kuris turi naudingą ir vartotojams patogią sąsają. Taip pat ataskaitų kūrimo įrankį galima panaudoti tiek programos kūrimo metu, tiek ir programos veikimo metu. Taip pat šis įrankis gali eksportuoti gautas ataskaitas į daugelį įvairiausių rinkmenų.

FastReport® Studio for business



5 pav. FastReport® Studio for business išvaizda.

Kūrėjas: Fast Reports Inc.

Produktas: FastReport® Studio for business

Internetinės svetainės adresas:

http://www.fast-report.com/en/products/report-generator-tool-for-business-fast-report-studio.html

Apžvalga:

Kompanijos Fast Reports Inc. Sukurtas produktas FastReport Studio for business yra atskyra programa skirta ataskaitų kūrimui, planiniam ataskaitų parengimui, išsaugojimui ir paskirstymui. Įrankyje tai pat yra daug demonstracinių ataskaitų pavyzdžių, bei jis gali dirbti su Microsoft Access ir Office programomis.

Programų sistemų savybių kiekybinis ir/ arba kokybinis palyginimas

3 lentelė. Programų sistemų savybių kokybinis palyginimas.

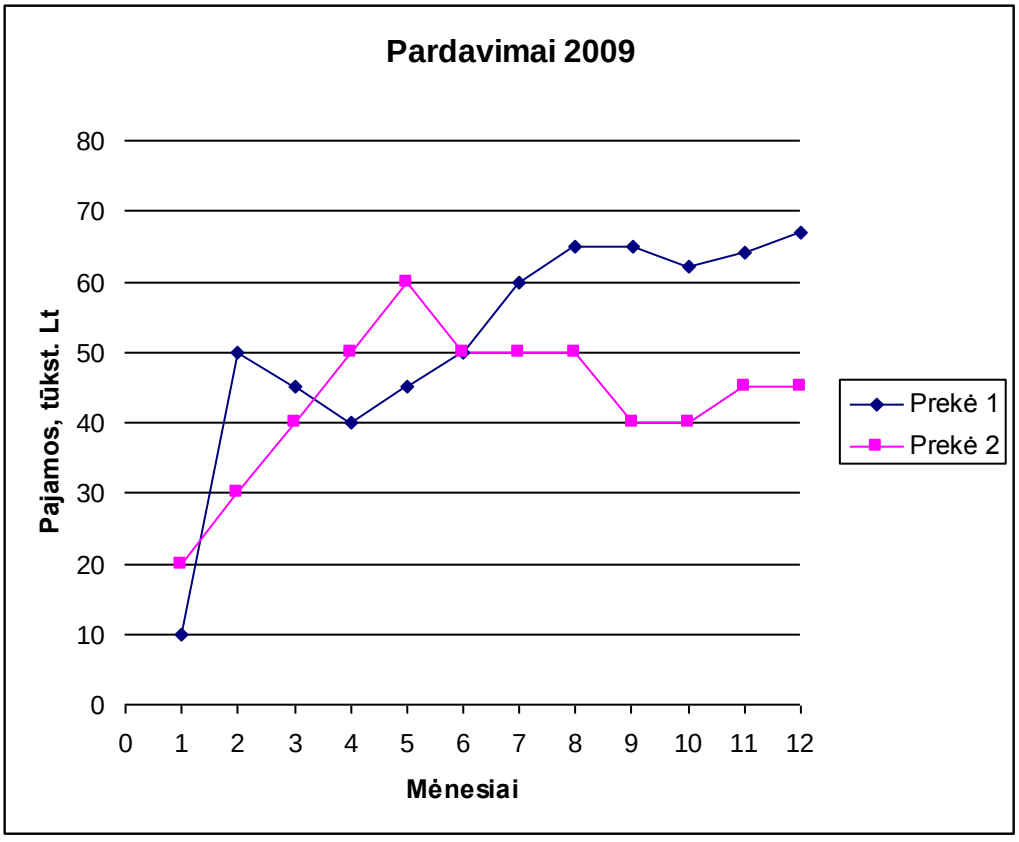
Produktas	Ataskaitų dizaineris	Galimybė rodyti grafikus	Ataskaitų eksportavimas	Tipas	Įvedimo formų generavimas
Report Maker 3	yra	Yra	yra	kompo - nentas	nėra
Catchysoft Report Generator	yra	Nėra	nėra	kompo - nentas	nėra
Stimulsoft Reports.Net yra .NET	yra	Yra	yra	kompo - nentas	nėra
FastReport® Studio for business	yra	Yra	yra	kompo - nentas	nėra

Kaip matome iš palyginimo nei vienas iš išnagrinėtų sprendimų negali automatiškai sugeneruoti parametrų įvedimo formos. Tad nei vienas iš jų nėra tinkamas.

3.1.3. Situacijos Lietuvoje įvertinimas

Remiantis statistikos departamento duomenimis Lietuvoje yra 18 bendrovių besiverčiančių vaistų gamyba. Tačiau kaip galimi užsakovo klientai yra ir kitos Europos Sąjungoje veikiančios kompanijos. Taip pat vertėtų atsižvelgti į tai, kad dauguma kompanijų turi atstovybes, keliose Europos Sąjungos šalyse. Tad gavus palankų produkto įvertinimą ir gerus atsiliepimus iš vienoje šalyje esančios atstovybės gali būti lengviau sudominti kitose šalyse esančias tos pačios įmonės atstovybes.

Taip pat statistikos departamento 2009 m. duomenimis Lietuvoje dirba 13 228 gydytojai, bei 1 504 vaistinės, kurias periodiškai lanko užsakovo agentai tikrindami kainas, atsiliepimus arba pristatinėdami naują produkciją. Tad veiklos yra didelis kiekis ir galimybė greitai ir efektyviai gauti informaciją apie konkretų gaminamą produktą, reklamos darbuotoją ar kokią nors kitą su įmonės veikla susijusią informaciją.



6 pav. Pardavimų ataskaitos pvz.

Duomenys šioms ataskaitoms yra gaunami iš duomenų bazių prieš tai paprašius vartotoją įvesti tam tikrą informaciją. Pavyzdžiui laikotarpį, prekes, darbuotojus ir t.t. Tam įvedimui yra kuriamos atskyros formos, rašomos užklausos. Šio magistrinio darbo metu bus sukurtas generatorius kuris perskaitys duomenis iš RDL tipo rinkmenos ir ją atitinkančios XML duomenų schemos ir išgaus:

- Prisijungimo duomenis,
- Ataskaitos išvaizdą,
- Duomenų struktūras,
- Užklausas reikalingas duomenims išgauti,
- Užklausoms reikalingus parametrus kuriuos turi įvesti vartotojas.

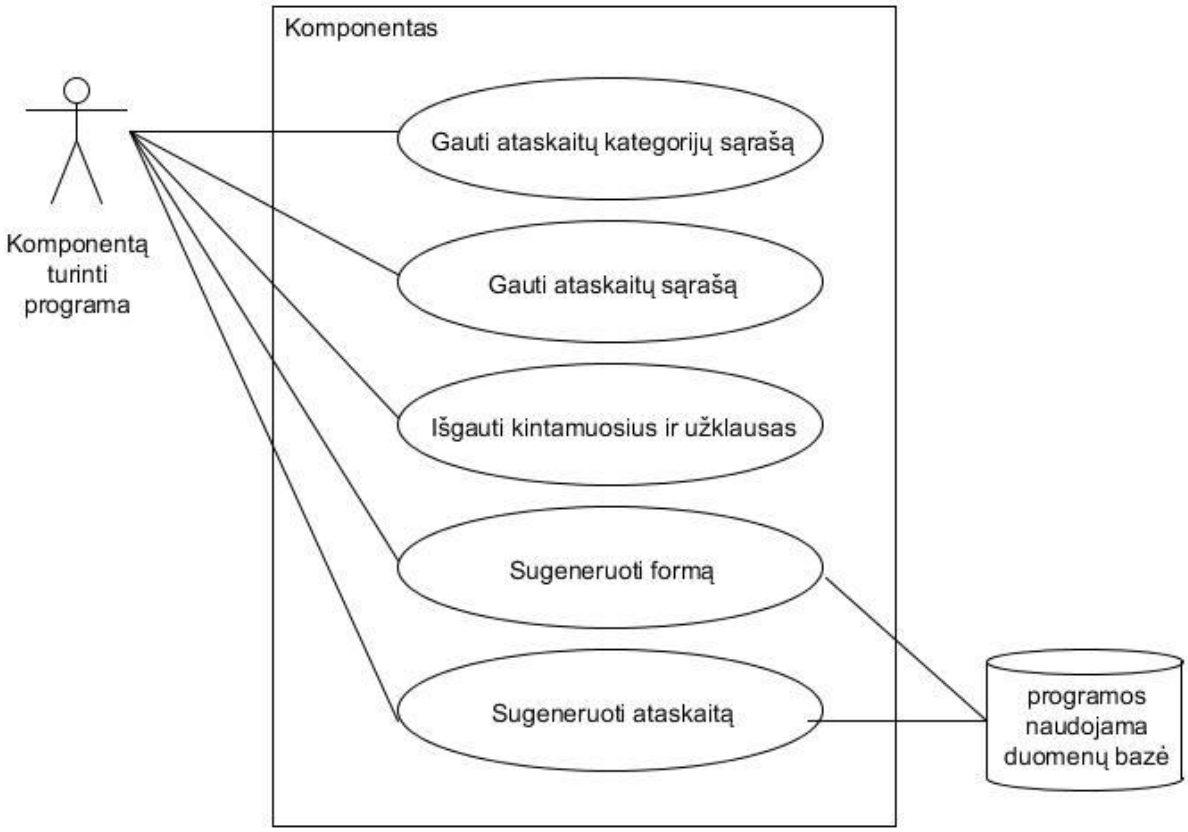
Gavęs šiuos duomenis generatorius susieja reikiamų įvesti duomenų tipus su C# WindowsForms esamais įvedimo laukais ir sugeneruoja formą, kad vartotojas galėtų įvesti duomenis. Vartotojui įvedus duomenis komponentas turės juos nuskaityti ir sugeneruoti SQL užklausas. Tada gavęs iš serverio duomenis komponentas turės sugeneruoti ataskaitą ir parodyti ją vartotojui.

Taigi automatizuodamas dalį dažnai pasitaikančių užduočių šis komponentas sutaupys laiko ir išteklių reikalingų sukurti programinei įrangai.

3.1.4. Sprendimo įgyvendinimas

Panaudojimo atvejų vaizdas

Pateikiami panaudojimo atvejai gauti Reikalavimų analizės metu:



7 pav. Panaudojimo atvejai.

Sistemos atliekamos funkcijos

1. PANAUDOJIMO ATVEJIS: Gauti ataskaitų kategorijų sąrašą

Aprašas: Vartotojas peržiūri galimas ataskaitų kategorijas.

2. PANAUDOJIMO ATVEJIS: Gauti ataskaitų sąrašą

Aprašas: Vartotojas peržiūri galimas pasirinkti ataskaitas priskirtas nurodytai kategorijai.

3. PANAUDOJIMO ATVEJIS: Išgauti kintamuosius ir užklausas

Aprašas: Apima procesą, kurio metu iš rinkmenų yra išgaunami kintamieji, jų tipai ir SQL užklausos reikalingos jiems užkrauti.

4. PANAUDOJIMO ATVEJIS: Sugeneruoti formą

Aprašas: Sugeneruoti formos langą kuriame būtų visi laukai kuriuos privalo užpildyti vartotojas.

5. PANAUDOJIMO ATVEJIS: Filtruoti ataskaitos duomenis

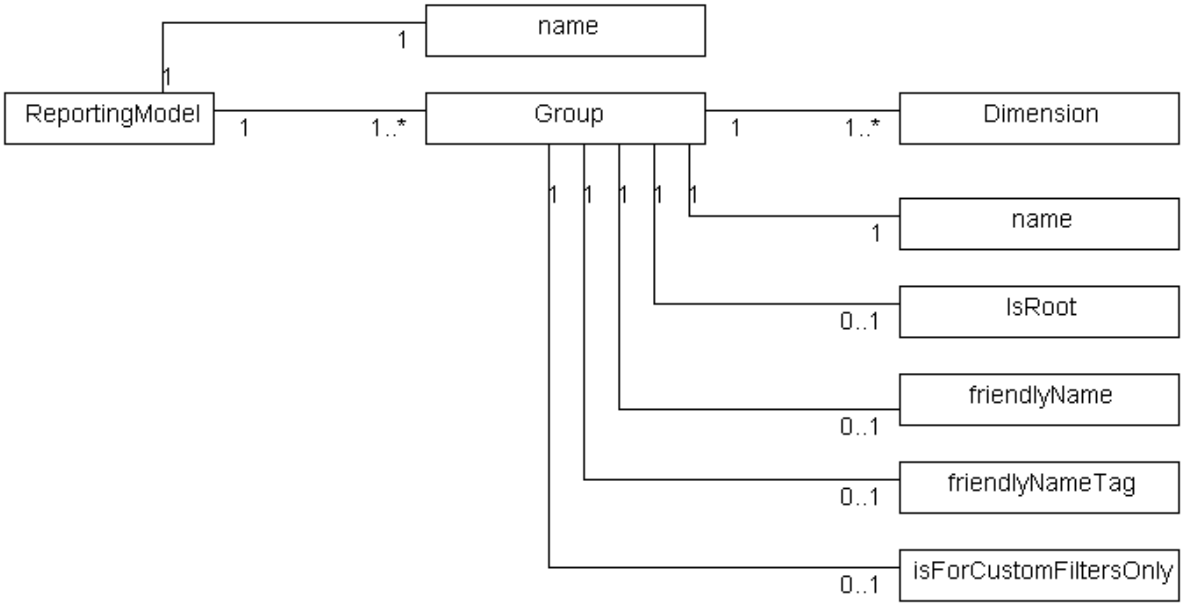
Aprašas: Programa turi gebėti filtruoti formos duomenis, bei turėti reikiamas formas filtrams atvaizduoti.

6. PANAUDOJIMO ATVEJIS: Sugeneruoti ataskaitą

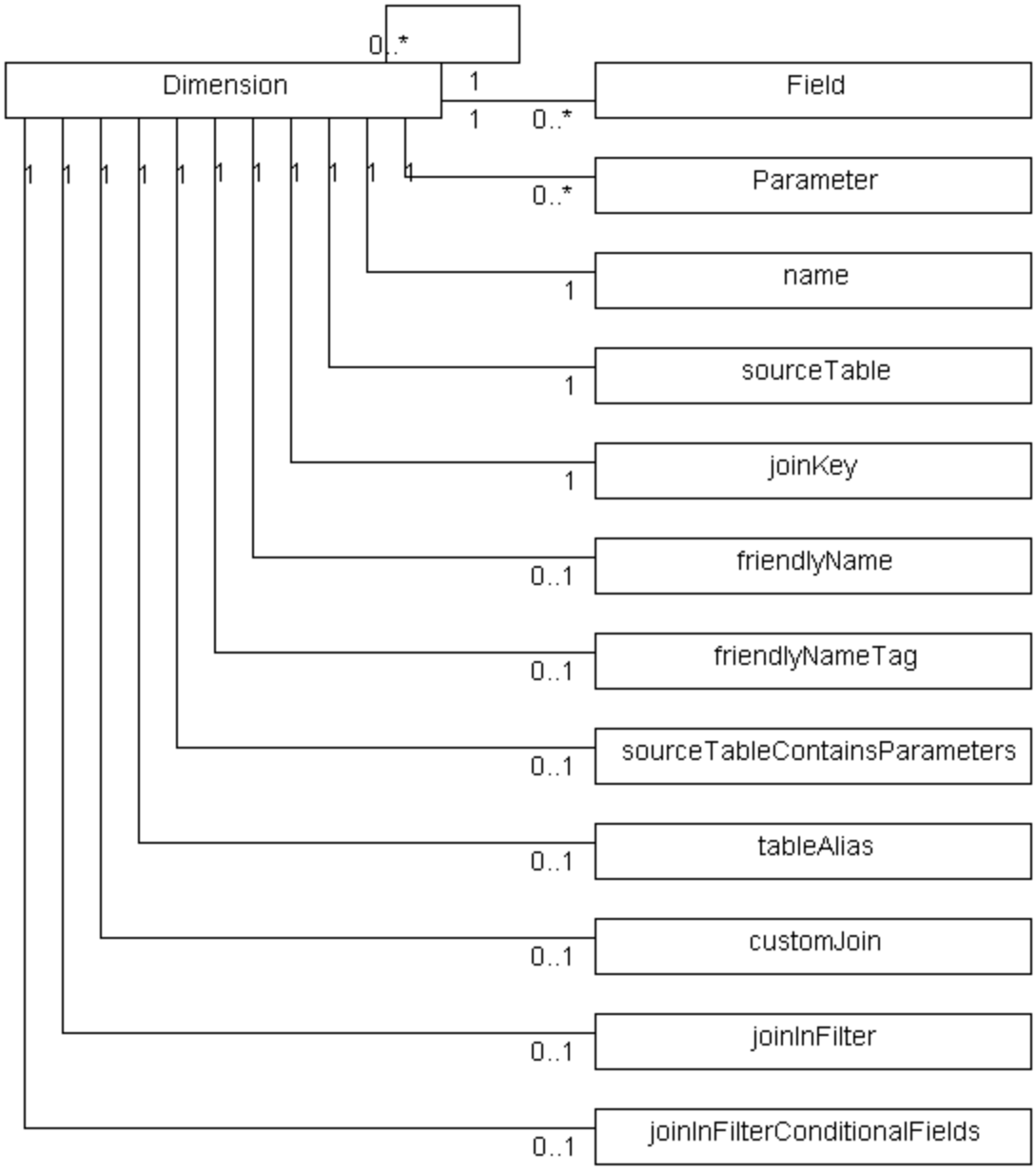
Aprašas: Sugeneruoti formos langą kuriame būtų visi laukai kuriuos privalo užpildyti vartotojas.

3.1.5. Kubo XML Metamodelis

Kadangi įrankis dirba su bet kokio tipo duomenų baze, jei tik ji aprašyta XML modelyje, tai duomenų bazės schemos pateikti negalima. Dėl to šiame skyriuje bus pateiktas XML modelis iš kurio yra generuojami duomenys ataskaitoms.

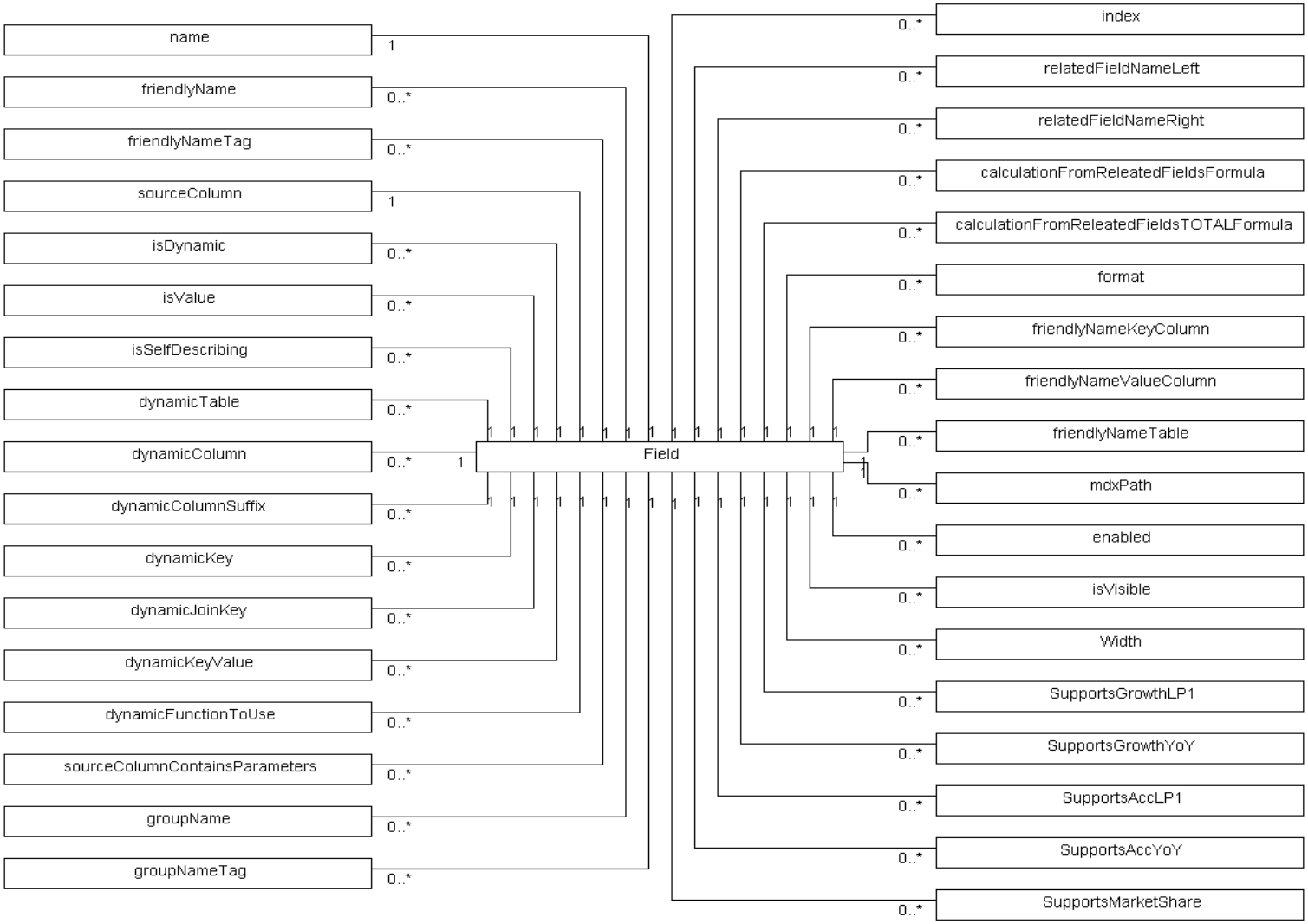


8 pav. Reporting Model schema.

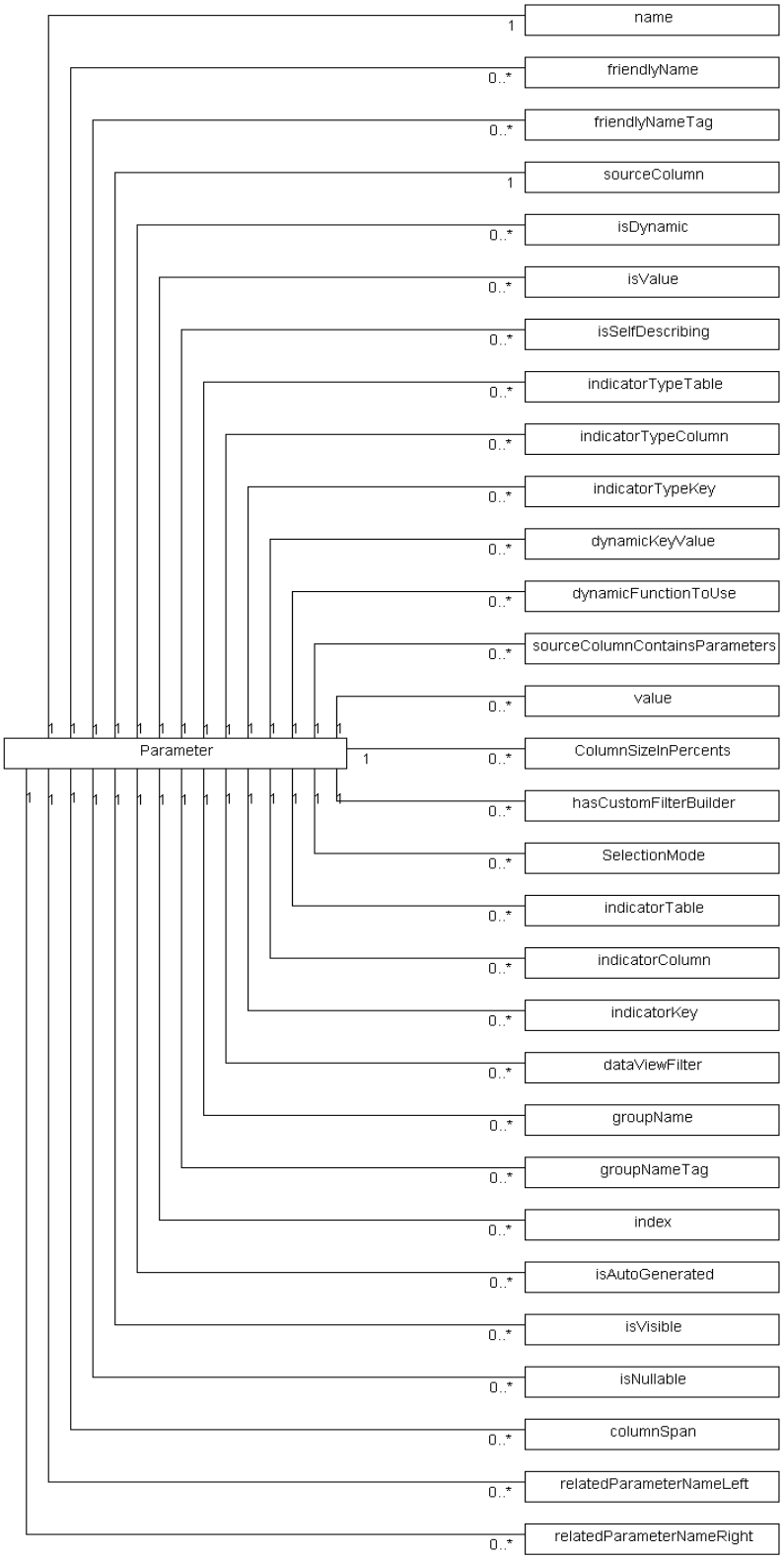


9 pav. Lauko Dimension schema.

.



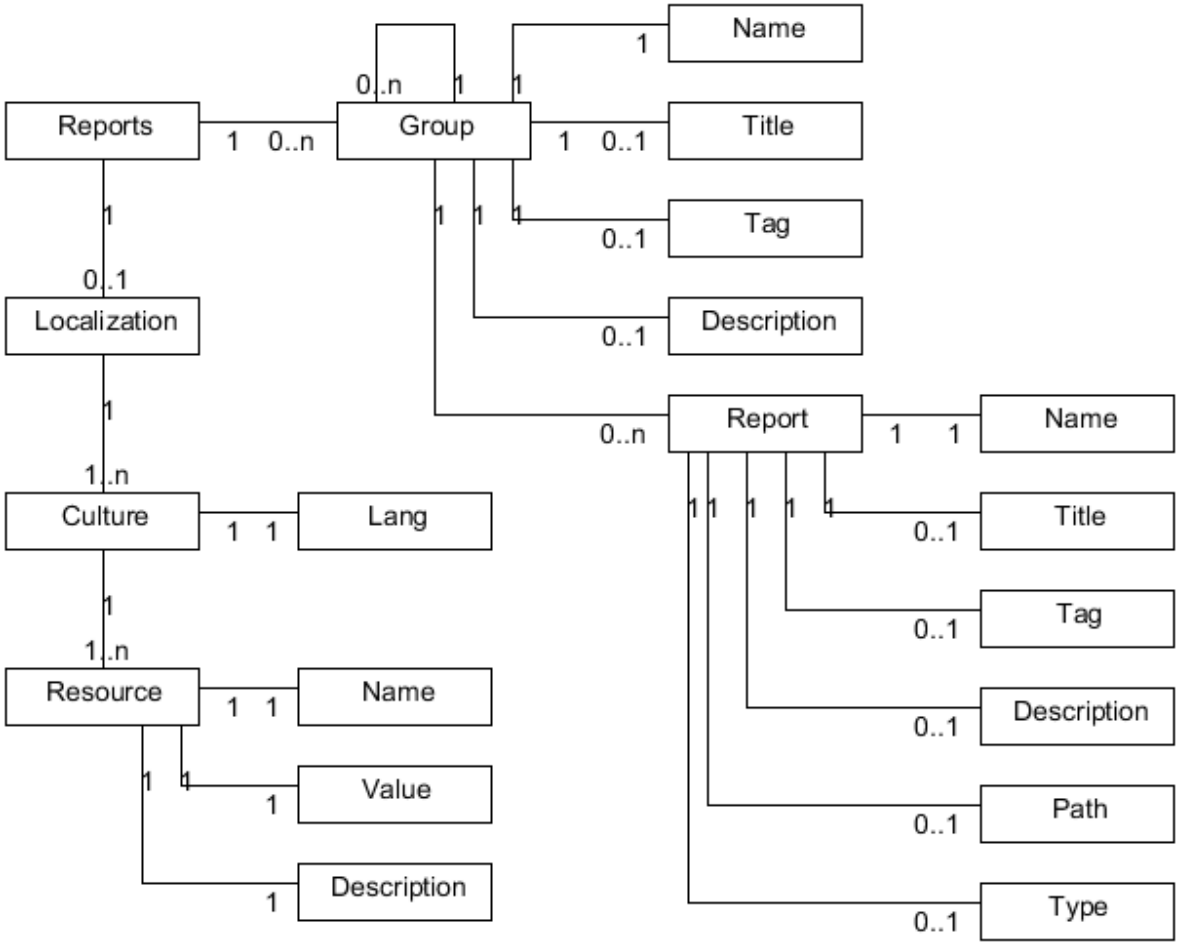
10 pav. Lauko Filter schema.



11 pav. Lauko Parameter schema.

3.1.6. Ataskaitų sąrašo XML metamodelis

Toliau pateikiamas ataskaitų sąrašo XML modelis. Pagal kurį generuojami ataskaitų sąrašai ir jų vertimai į kitas kalbas.

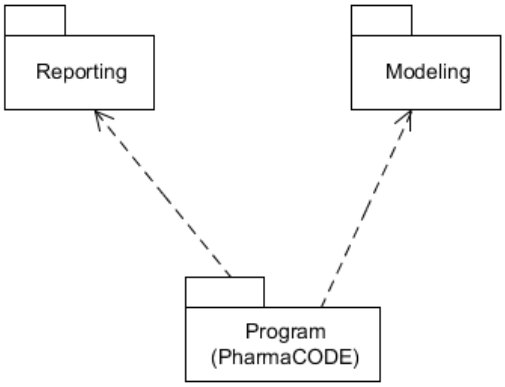


12 pav. Ataskaitų sąrašo modelio schema.

3.2. Architektūra

Įrankis suskaidytas į du pagrindinius paketus: Reporting ir Modeling. Paveikslėlyje taip pat pavaizduoti ryšiai su šį įrankį naudojančia programa. Šiuo atveju PharmaCODE.

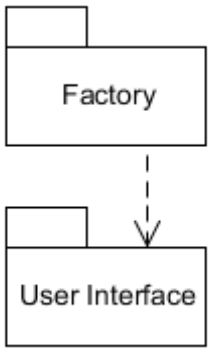
.



13 pav. Aukščiausio lygio paketų diagrama.

3.3. Paketų detalizavimas

3.3.1. Paketas Reporting



14 pav. Paketo Reporting žemesnio lygio paketų diagrama.

Pakete pateigti paketai atsakingi už grafinės sąsajos (ataskaitų peržiūros ir įvedimo formų) generavimą ir jų sukeltų įvykių apdorojimą.

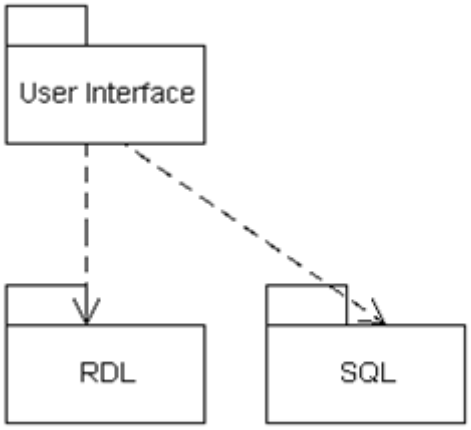
Paketas Factory

Pakete esančios klasės valdo formų ir SQL užklausų generavimą.

Paketas User Interface

Pakete pateigtos klasės susijusios su grafinės sąsajos generavimu.

3.3.2. Paketas Modeling



15 pav. Paketo Modeling žemesnio lygio paketų diagrama.

Paketas skirtas RDL rinkmenų generavimui

Paketas User Interface

Paketas realizuoja sąsają skirta duomenų eksportavimui.

Paketas RDL

Paketas skirtas generuoti RDL rinkmenas reikalingas ataskaitų generavimui. Jame esančios klasės aprašo generuojamų ataskaitų išvaizdą, pagal pasirinktus laukus sukuria ataskaitos šabloną, kurį paskui yra panaudojamas ataskaitos parengimui.

Paketas SQL

Paketas skirtas SQL komandų generavimui, bei analizei. Paketo klasės iš nurodytų laukų, bei papildomų parametrų (lentelių slapyvardžių, sujungimų tipų, filtruojamų duomenų) sugeneruoja SQL užklausas, kurios užkrauna ataskaitoms reikalingus duomenis.

3.4. Išvados

Įgyvendinus magistrinį projektą galima padaryti tokias išvadas:

1. Atliekant magistrinį projektą buvo sėkmingai įgyvendinta ataskaitų generavimo iš ataskaitų kubo modelio koncepcija.

2. Sukurtas įrankis sėkmingai sugeneruoja įvedimo formas, bei galima formoms suteikti papildomo funkcionalumo, pavyzdžiui paslėpti nuo vartotojo kuriuos nors laukus į modelį įvedus papildomą požymį.
3. Sukūrus ataskaitų generavimo iš modelio algoritmą užsakovas nesunkiai galėjo savo klientams suteikti galimybę peržiūrėti jį dominančius duomenis įvairiu formatu.
4. Kadangi galimų skirtingų ataskaitų kiekis labai ženkliai išaugo iškilo testavimo problema.
5. Tam, kad išspręsti su testavimu iškilusias problemas nuspręsta sukurti automatinio testavimo metodą ir juo paremtą įrankį.

4. TYRIMO DALIS

Šioje dalyje apžvelgsime ataskaitų modelio testavimo metodą

Magistratūros studijų metu buvo kuriamas ataskaitų ir duomenų įvedimo formų generavimo komponentas. Sėkmingai įgyvendinus projektą buvo pastebėta, kad dėl padidėjusio galimų ataskaitų kiekio labai buvo apsunkintas testavimo procesas ir dėl to nuspręsta, kad reiktų sukurti metodą ir įrankį kuris galėtų bent dalinai sutrumpinti testavimui skirtą laiką. Tam tikslui yra siūlomas modelio testavimo metodas.

4.1. Apibrėžimai

Komponento generuojamos SQL užklausa sudaro šios dalys:

- Pasirinkimų sąrašas;
- Lentelių rodyklės;
- Filtravimo informacija;
- Grupavimo informacija.

SQL užklausos ir jų sudėtinės dalys plačiau paaiškintos skyriuje 2.6 SQL užklaus.

4.2. Problema

Parašius naują ataskaitų kubo modelį arba pakeitus jau esantį reikia jį ištestuoti. Kubo modelio sintaksės testavimą atlieka pats įrankis kuriuo jis yra kuriamas. Didžioji dalis galimų įrankių gali patikrinti ar padaryti pakeitimai atitinka XML sintaksę ir paryškinti blogai parašyta tekstą.

Tačiau validi sintaksė negarantuoja, kad bus sugeneruotos užklausos kurias bus galima įvykdyti pasirinktoje duomenų bazėje. Be to kubo modelyje gali būti panaudotos dinaminės dimensijos pavyzdžiui: produkto pakuotės indikatorius, asmens tipas ar organizacijos tipas, kuriuos susikuria pats vartotojas ir dėl to skirtingose duomenų bazėje gali neveikti tos pačios užklausos. Tad norint ištestuoti iš kubo modelio generuojamas užklausas yra tik viena galimybė: pereiti visas galimų pasirinkti laukų kombinacijas, kurių gali būti labai daug. Iš viso galimų variantų kiekį galima apskaičiuoti pagal formulę:

Lygtis 1. Galimų variantų kiekis

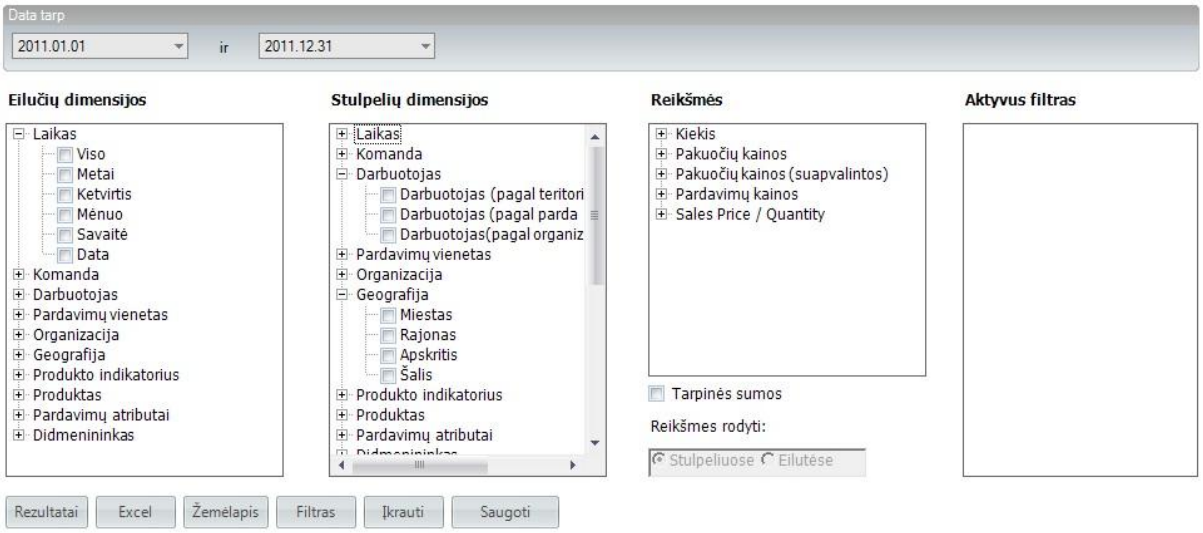
$$n_v = n_d * (n_d - 1) * n_r;$$

n_v – visų galimų variantų kiekis;

n_d – galimų dimensijų kiekis;

n_r – galimų reikšmių kiekis.

Pavyzdžiui: Pagal žemiau pateiktą modelį paveikslėlį: pateiktame pavyzdyje yra 50 laukų ir 33 galimos reikšmės. Tad išviso yra n=50*(50-1)*33 = 80850 galimi variantai.



16 pav. Dimensijų pavyzdys.

Taigi peržiūrėti visas galimas kombinacijas prireiktų daug pastangų ir, atsižvelgiant į tai, kad užklausų apdorojimas užtrunka, tai testavimui gali prireikti labai daug laiko ir greičiausiai išvis neįmanoma to atlikti rankiniu būdu. Žinoma, pasinaudojant ankstesnių testavimų patirtimi, bei žinant asmenis kurie vykdė pakeitimus galima būtų atsisakyti visų variantų peržiūros ir apsiriboti tik tam tikrų pasirinkimų poaibiu, tačiau taip kyla rizika, kad kas nors gali likt nepastebėta. Be to atliekant testavimą rankiniu būdu testuotojas turi suvedinėti klaidų aprašymus, kas irgi užtrunka. Taip pat vykdant testavimą rankiniu būdu nėra galimybės išsaugoti vykdomų SQL užklausų tekstą, nes jos nėra parodomos, o tokia galimybė leistų jas pateikus programuotojų komandai paspartinti tolesnę problemos analizę.

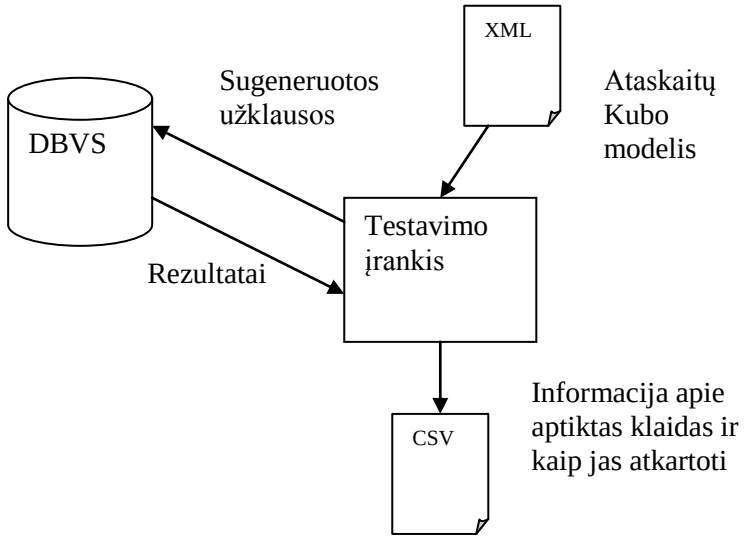
4.3. Tikslai

Ataskaitų kubo modelio testavimo įrankis turėtų atlikti šias funkcijas:

- Iš modelio išgauti visus galimus pasirinkti laukus;
- Automatiškai generuoti pasirinktų laukų kombinacijas;
- Patikrinti ar iš jam pateikto modelio yra sugeneruojamos validžios SQL užklausos.
- Automatiškai parengti duomenis galimam regresiniam nepavykusių pasirinkimų testavimui:
    - Įsiminti pasirinktus laukus.
- Fiksuoti užklausų įvykdymo greitį ir su vykdymo trukme susijusias klaidas.

4.4. Sprendimo būdas

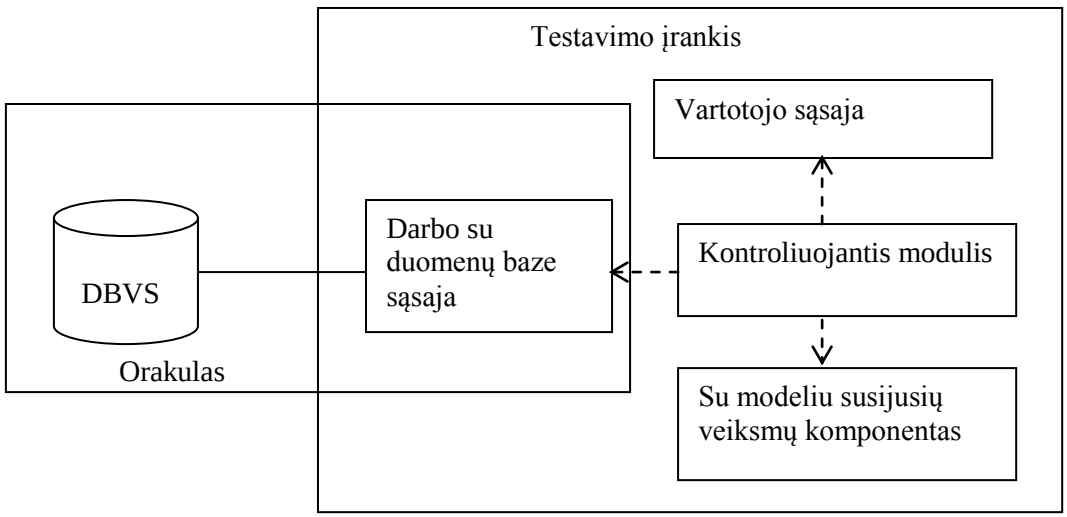
Siekiant išspręsti anksčiau įvardintą problemą buvo nuspręsta sukuri atskyrą įrankį, kuris galėtų atlikti testavimą. Įrankiui kaip duomenys būtų paduodamas kubo modelis, o rezultatai būtų informacija apie aptiktas klaidas su informacija kaip jas atkartoti. Orakulo vaidmenį atliktų duomenų bazių valdymo sistema.



17 pav. Testavimo įrankis.

Testavimo įrankis susidės iš keturių dalių:

- išorinės duomenų bazės,
- darbo su duomenų baze komponento,
- vartotojo sąsajos,
- testavimo procesą kontroliuojančio modulio,
- manipuliavimo su modeliu komponento.



18 pav. Testavimo įrankio struktūra.

Išorinė duomenų bazė

Testavimui atlikti buvo nuspręsta panaudoti išorinę duomenų bazę, kurios informaciją galėtų pateikti vartotojas. Toks sprendimas buvo pasirinktas todėl, kad sukurti modeliai gali būti naudojami daugelyje skirtingų programos versijų, kurios gali palaikyti skirtingą duomenų bazės struktūrą ar naudoti skirtingus užklausų vykdymo algoritmus. Nors įprastai tą pačią programos versiją naudojančių serverių duomenų bazių struktūra sutampa, atnaujinat versijas arba diegiant duomenų bazę naujam klientui gali atsirasti klaidų. Dėl to toks sprendimas suteiks galimybę patikrinti ar modelis atitinka konkrečią pasirinktą duomenų bazę ir gali būti naudingas identifikuojant su ataskaitų generavimu susijusias problemas ar norint įsitikinti, kad išleidus naują versija pas visus klientus viskas veiks.

Su duomenų baze susijusių veiksmų atlikimo komponentas

Šis komponentas skirtas atlikti veiksmus su pasirinkta duomenų baze. Jis turi galėti prisijungti prie duomenų bazės, įvykdyti nurodytą užklausą ir atsijungti nuo duomenų bazės. Šiuos veiksmus buvo pasirikta realizuoti kaip atskyrą komponentą, tam kad prireikus būtų galima įrankį papildyti galimybe dirbti su kitokio tipo duomenų bazių valdymo sistemomis.

Vartotojo sąsaja

Vartotojo sąsajos komponento paskirtis priimti ir interpretuoti vartotojo pasirinkimus ir komandas. Kadangi testavimas gali būti atliekamas tiek paleidus įrankį ir nurodžius visus veiksmus vartotojo sąsajoje, tiek ir paleidus į fizinę rinkmeną įrašytą komandų seką norint ištestuoti kelis modelius.

Kontroliuojantis modulis

Kontroliuojančiojo modulio paskirtis yra parinkti ir įkelti reikiamą vartotojo sąsajos tipą, iš modelio komponento paimti galimus laukus ir iš jų suformuoti testavimui skitus įėjimus, persiųsti duomenų bazės komponentui sugeneruotas užklausas ir išsaugoti rezultatus.

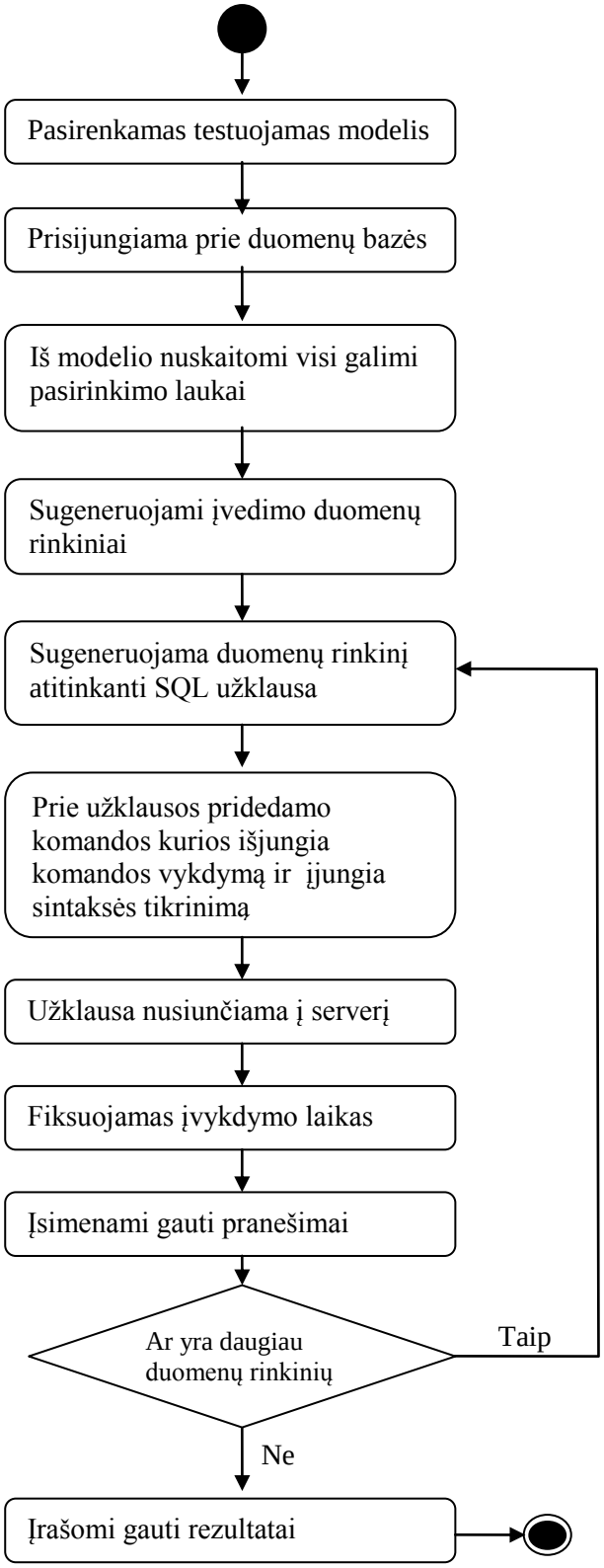
Su modeliu susijusių veiksmų atlikimo komponentas

Su modeliu susijusių veiksmų atlikimo komponentas turi iš kubo modelio išgauti galimas pasirinkti reikšmes, bei jam pateikus pasirinktas reikšmes sugeneruoti SQL užklausą.

4.5. SQL užklausų testavimas

SQL užklausų testavimui buvo pasirinktas SQL užklausų kompiliavimo serveryje su užklausų vykdymą sustabdančiomis komandomis metodas, kadangi norint jį panaudoti nereikia jokių papildomų

bibliotekų, bei jis nesukels pavojaus, kad gali būti pakenkta duomenims. Žemiau pateikiamas testavimo veikimo algoritmas:



19 pav. Testavimo algoritmas.

4.6. Keliamos Hipotezės

Tyrimo metu buvo iškeltos dvi hipotezės kurias bus bandoma priimti arba atmesti remiantis Skyriuje 5. Eksperimentinė dalis, aprašomo eksperimento rezultatais:

Hipotezė 0

Tikrinant SQL užklausas negalima greitai ir patikimai ištestuoti XML modelio.

Hipotezė 1

Vykdant galimų pasirinkimų perrinkimą ir testuojant iš jų sugeneruotas iš užklausas galima sėkmingai ir per priimtiną laiko tarpą ištestuoti visą kubo modelį.

4.7. Išvados

Atlikus tyrimą galima padaryti tokias išvadas:

1. Ataskaitų kūrimo proceso palengvinimas sąlygojo įvairiausių skirtingų ataskaitų tipų kiekio padidėjimą. Savo ruožtu toks ataskaitų kiekio padidėjimas sukėlė testavimo problemą: Perrinkti visas galimas pasirinkti kombinacijas ir įsitikinti, kad ataskaita yra sugeneruojama pasidarė fiziškai neįmanoma, dėl to prireikė automatizuoto testavimo metodo.
2. Projektuojamas testavimo įrankis buvo išskaidytas į atskyras dalis. Tai ateityje padės lengviau pridėti papildomo funkcionalumą tiesiog vieną komponentą pakeičiant kitu. Tarkime su MS SQL serveriu dirbantį komponentą pakeičiant į su Postgre SQL dirbantį komponentą.
3. Vietoj to, kad būtų rašomas Orakulo metodas, buvo nuspręsta pasinaudoti SQL duomenų bazių valdymo sistema ir taip sutaupyti laiko, bei užtikrinti, kad testuojamos užklausos bus semantiškai teisingos su ten naudojama duomenų baze.
4. Tam, kad nebūtų pakenkta duomenų bazėje esantiems duomenims, buvo atkartota komercinio įrankio užklausų tikrinimo logika, kai užklausų vykdymas yra sustabdomas panaudojus Serverio komandas.

5. EKSPERIMENTINĖ DALIS

Šiame skyriuje pateikiamas vykdytų modelio testavimo eksperimentų aprašymas, bei gautų rezultatų analizė.

5.1. Eksperimento tikslas

Pagrindinis atliekamo eksperimentinio tyrimo tikslas yra išnagrinėti sukurto testavimo metodo ir įrankio veikimo charakteristikas. Tyrimas bus atliekamas su analogišku realiai naudojamam ataskaitų kubo modeliu, kuriame yra aprašyta 50 dimensijų, kurios gali būti atvaizduotos stulpeliuos arba eilutėse, bei aprašytos 33 galimos skirtingos reikšmės.

5.2. Eksperimento vykdymo planas

Eksperimento vykdymui buvo paruošta aplinka: įdiegta duomenų bazių valdymo sistema, nukopijuota programų kūrimui ir testavimui naudojama vietinė duomenų bazė ir paruoštas testuojamas kubo modelis.

Paleidus testavimo įrankį buvo nustatyti duomenų bazės serverio parametrai ir patikrinta ar pavyksta prisijungti prie duomenų bazės. Nurodytas testuojamas modelis ir paleidžiama vykdyti testavimo procedūra.

Eksperimentas atliekamas su kompiuteriu kurio charakteristikos pateikiamos lentelėje žemiau:

4 lentelė. Eksperimentams skirto kompiuterio informacija.

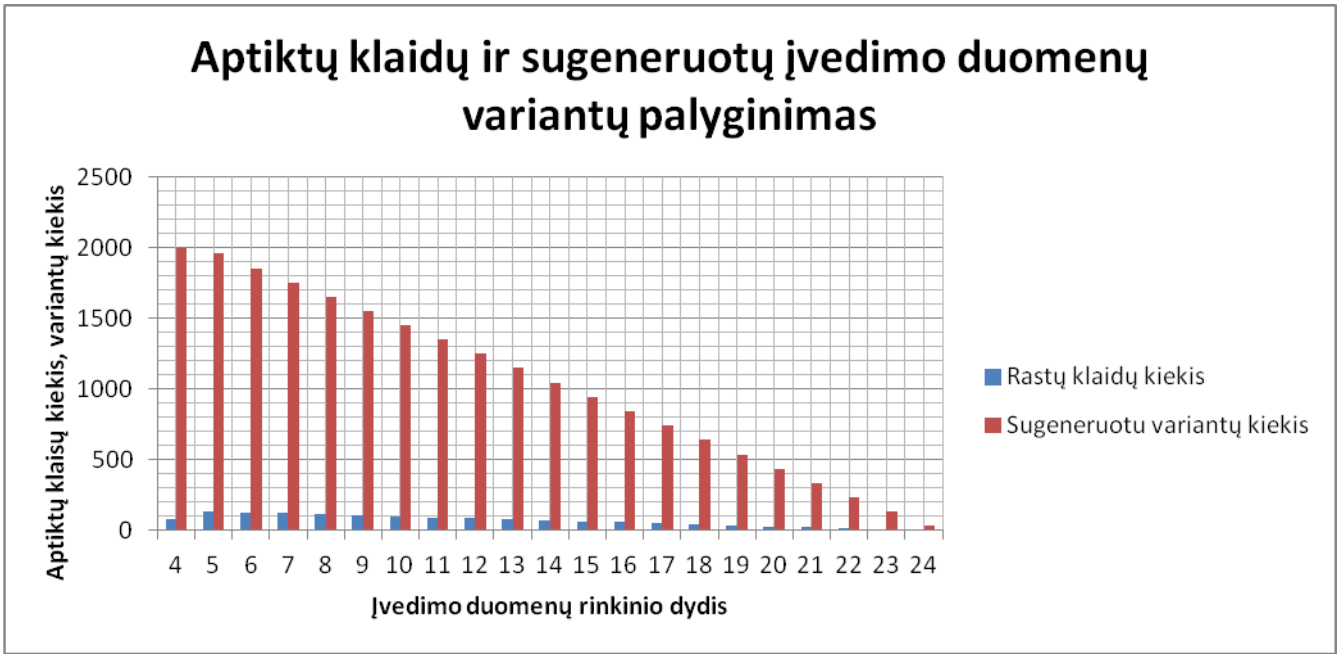
Procesorius	2 branduoliai, 3,2 Ghz
Operatyvioji atmintis	4 GB
Laisvos vietos kietajame diske	200 GB
Operacinė sistema	Microsoft Windows 7
DBVS	Microsoft SQL Server 2008 Express

Eksperimento metu buvo renkama tokia informacija:

- įvedimo rinkinių elementų aibės ilgis,
- aptiktų klaidų kiekis,
- sugeneruotu rinkinių kiekis,
- testavimo trukmė,
- bendras užtrukęs laikas, valandomis,
- bendras rastas klaidų kiekis,
- bendras patikrintas rinkinių kiekis,
- aptiktų klaidų ir sugeneruotų rinkinių kiekio santykis.

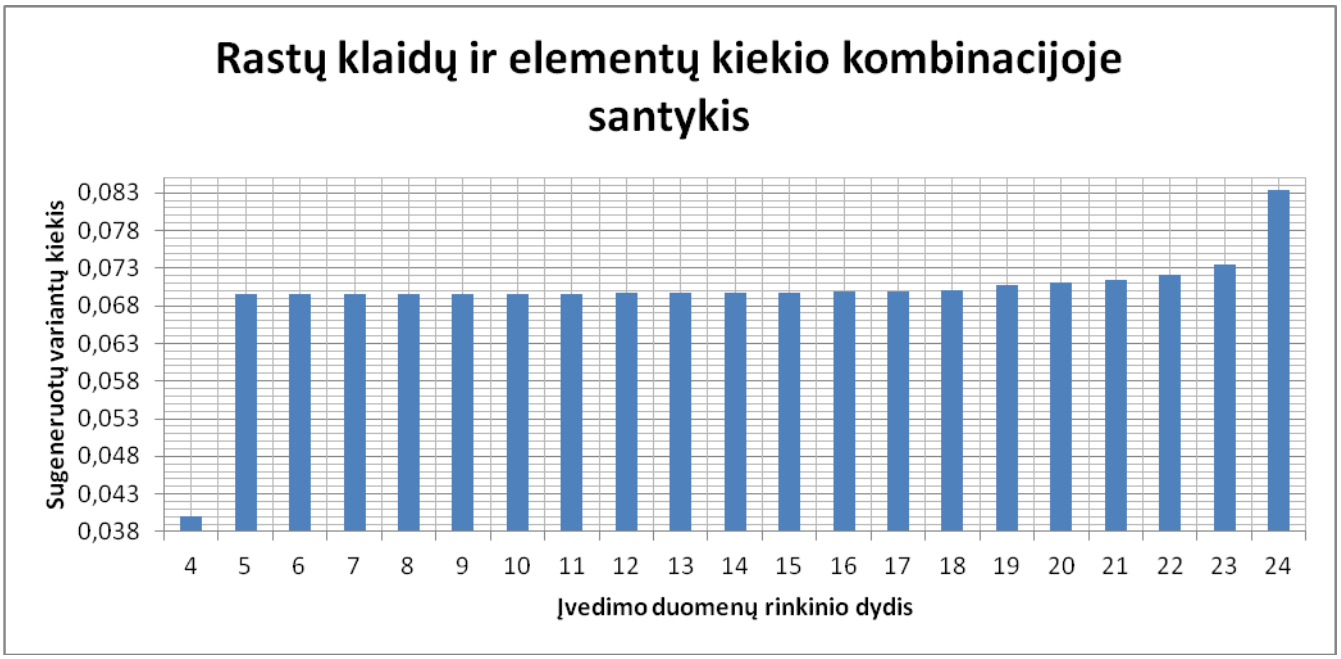
5.3. Eksperimento rezultatų analizė

Bendras atlikto modelio testavimo trukmės laikas 6 valandos. Apibendrinus testavimo rezultatus gavome:



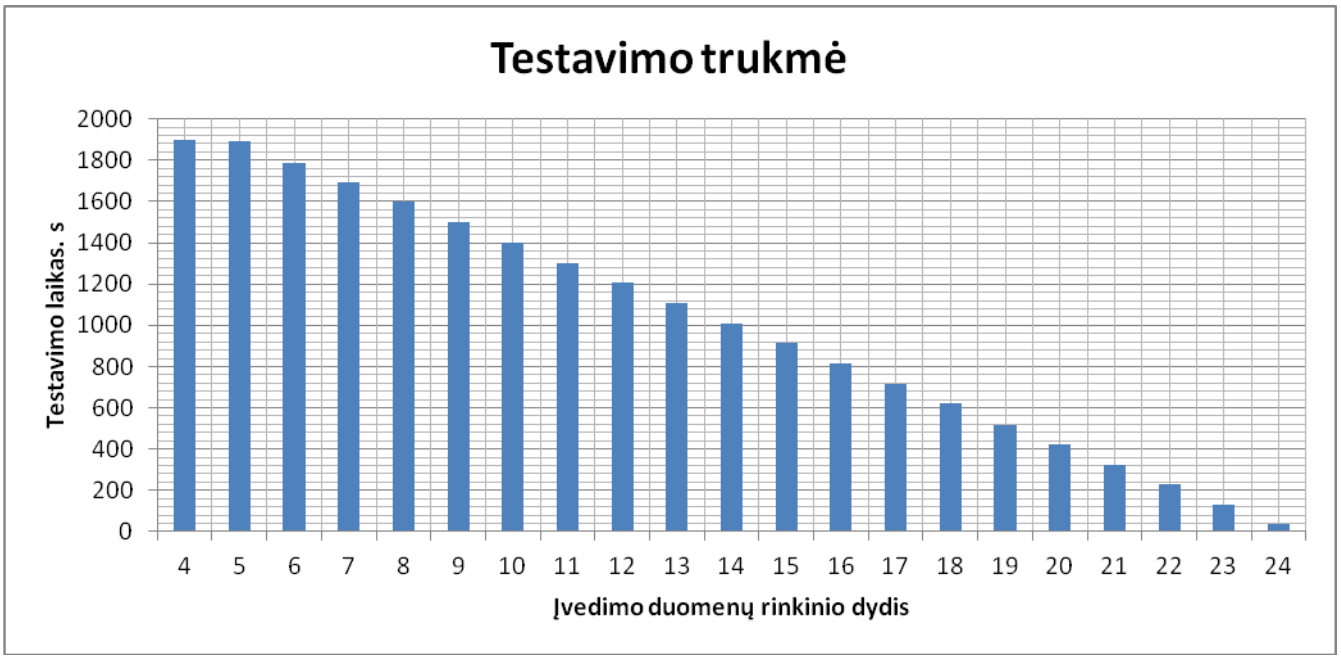
20 pav. Aptiktų klaidų ir sugeneruotos variantų kombinacijos.

21 pav. Aptiktų klaidų kiekio priklausomybė nuo ištestuotų rinkinių.



22 pav. Rastų klaidų ir elementų kiekio įvedimo duomenų rinkinyje santykis.

Kaip matome iš aukščiau pateikto grafiko, sugeneruotų įvedimų kombinacijų kiekis yra tiesiškai priklausomas nuo elementų kiekio kombinacijoje ir didėjant elementų kiekiui gaunamų kombinacijų kiekis mažėja. Tačiau taip pat galima pastebėti, jog padidinus elementų kiekį kombinacijoje nuo 4 iki 5 padidėja aptiktų klaidų kiekis. Taip nutinka todėl, kad kai kurios klaidos pasireiškia tik esant didesniam elementų kiekiui. Pavyzdžiui sutampantys pavadinimai ar blogai aprašyti sujungimai. Taip pat galima pastebėti, kad absoliutus aptiktų klaidų kiekis, bei aptiktų klaidų ir elementų kiekio įvedimo duomenų rinkinyje santykis nors ir labai nežymiai, bet nuolat auga, todėl, kad yra aptinkamos anksčiau aptiktos klaidos. Taip pat galima pastebėti, kad nėra tikslinga naudoti rinkinius su mažu elementų kiekiu, nes aptinkamų klaidų ir kiekio santykis yra žemesnis negu rinkinių su daugiau elementų. Taip pat, kaip matome, iš eksperimento trukmės ir aptiktų klaidų buvo sėkmingai paneigta Hipotezė 0, kad negalima laiku ir patikimai perrinkti galimus variantus ir taip ištestuoti kubo modelį ir patvirtinta Hipotezė 1, kad testavimą galima įvykdyti laiku ir sėkmingai.



23 pav. Testavimo laikas.

24 pav. Testavimo trukmės priklausomybė nuo įvedimo duomenų kiekio.

Kaip matome iš aukščiau pateiktų grafikų testvaimui reikalinga trukmė tiesiškai priklauso nuo sugeneruotų įvedimo elementų kiekio ir nepriklauso nuo gaunamų užklausų sudėtingumo, nes didžią ją dalį testavimui reikalingo laiko užima su modeliu susijusių veiksmų atlikimas.
Išsamųs eksperimento rezultatai pateikiami priede 9.2 Kubo modelio testo rezultatai.

5.4. Išvados

Atliekant testavimo įrankio patikrinimo eksperimentą buvo:

1. Įsitikinta, kad iš sintaksiškai teisingo kubo modelio gali būti sugeneruotos klaidingos užklausos.

2. Patvirtinta hipotezė teigianti, kad panaudojant pasirinktą algoritmą galima sėkmingai ir per priimtiną laiko tarpą aptikti modelyje esančias klaidas.
3. Pastebėta, kad aptiktų klaidų kiekis yra priklausomas nuo sugeneruotų pasirinktų įvedimo laukų kiekio.
4. Pastebėta, kad negalima apsiriboti vien tiktai mažos apimties duomenų rinkiniais, nes kai kurios klaidos susijusios su lentelių sujungimais ar vienodais pavadinimais pasireiškia tik esant didesniam duomenų kiekiui.
5. Pastebėta, kad testavimo laikas nežymiai priklauso nuo užklausų sudėtingumo, tad tai parodo, kad didžioji laiko dalis yra sugaištama veiksmams su modeliu atlikti ir padarius kelis pakeitimus algoritme būtų galima testavimo procesą pagreitinti.
6. Taip pat pastebėta, kad didėjant elementų užklausose kiekiui didėja ir santykinis aptiktų klaidų kiekis,

6. IŠVADOS

1. Panaudojus XML kalbą galima sėkmingai aprašyti duomenų kubo modelį kuriuo pasinaudojant galima generuoti kitus nuo modelio priklausomus komponentus: užklausas, ataskaitas, duomenų įvedimo formas.
2. Panaudojus automatinį ataskaitų generavimą galimų skirtingų ataskaitų kiekis labai ženkliai išaugo ir dėl to iškilo testavimo problema.
3. Tam, kad išspręsti su testavimu iškilusias problemas nuspręsta sukurti automatinio testavimo metodą ir juo paremtą įrankį.
4. Redaguojamo XML modelio sintaksę įprastai automatiškai patikrina teksto redaktorius su kuriuo yra modelis redaguojamas, tačiau validi sintaksė neužtikrina, kad modelis yra semantiškai prasmingas ir kad bus sugeneruotos veikiančios užklausos.
5. Kuriant testavimo metodą buvo išanalizuota iškilusi testavimo problema ir jos galimi sprendimai, bei sukurtas testavimo algoritmas ir panaudojant jį suprojektuotas testavimo įrankis, kurio veikimas buvo ištestuotas vykdant eksperimentą.
6. Taip pat buvo pastebėta, kad aptiktų klaidų kiekis yra priklausomas nuo sugeneruotų pasirinktų įvedimo laukų kiekio, bei, kad negalima apsiriboti vien tiktai mažos apimties duomenų rinkiniais, nes kai kurios klaidos susijusios su lentelių sujungimais ar vienodais pavadinimais pasireiškia tik esant didesniam duomenų kiekiui.
7. Taip pat pastebėta, kad testavimo laikas nežymiai priklauso nuo užklausų sudėtingumo, tad tai parodo, kad didžioji laiko dalis yra sugaištama veiksmams su modeliu atlikti ir padarius kelis pakeitimus algoritme būtų galima testavimo procesą optimizuoti.

7. LITERATŪRA

[1] **Software Testing Stuff: Software Testing Dictionary**, *Software Testing Dictionary* [Interaktyvus] - 2007.09.17 - [Žiūrėta 2012.05.22] http://www.softwaretestingstuff.com/2007/09/software-testing-dictionary.html

[2] **WebFinance, Inc**, *Report* [interaktyvus], 2012, [Žiūrėta 2012.05.19] Prieiga per internetą: http://www.businessdictionary.com/definition/report.html

[3] **Chapple M.**, *Query Definition* [interaktyvus], 2012, [Žiūrėta 2012.05.18] Prieiga per internetą: http://databases.about.com/cs/administration/g/query.htm

[4] **Rouse M.**, *OLAP cube* [interaktyvus], 2012, [Žiūrėta 2012.05.21] Prieiga per internetą: http://searchdatamanagement.techtarget.com/definition/OLAP-cube

[5] **TechTerms**, *Database* [interaktyvus], 2009, [Žiūrėta 2012.05.22] Prieiga per internetą: http://www.techterms.com/definition/database

[6] **Oracle,** *OLAP Application Developer's Guide, 10g Release 1* [interaktyvus], 2003. Prieiga per internetą: http://download.oracle.com/products/bi/pdf/9204_olap_appdev_guide.pdf

[7] **Karayannidis, N., Vassiliadis, P., Tsois, A., ir Sellis, T. ERATOSTHENES**: *Design and Architecture of an OLAP System*, Athens, 2001

[8**] Ammann P., Offutt J.,** *Introduction to Software Testing*, Cambridge University Press, 2008

[9**] International Organisation for Standardization,** ISO9126 *Information Technology Software Product Evaluation Quality* . Geneva, 1992.

[10] **Jones C. C,**, *Software Quality: Analysis and Guidelines for Success*, 1997, ISBN:1850328676

[11] **Glenford J. Myers**, *The Art of Software Testing*, Hoboken, New Jersey, 2004, ISBN 0-471-46912-2

[12] **Microsoft Corporation**, *Report Definition Language (SSRS)* [interaktyvus],, 2012, [Žiūrėta 2012.05.20] Prieiga per internetą: http://msdn.microsoft.com/en-us/library/ms155062.aspx

[13] **Microsoft Corporation**, *Report Definition Language Specification THIRD VERSION*., 2008

[14] **Chaudhuri, S. ir Dayal, U**. *An Overview of Data Warehousing and OLAP Technology*, ACM SIGMOD Record 1997, Volume 26 Issue 1, p. 65 – 74

[15] **Näppilä, T.; Järvelin, J.; ir Niemi , T.** *Tool for Data Cube Construction from Structurally Heterogeneous XML Documents*. *Journal of the American Society for Information Science and Technology* Vasaris, 2008, Volume 59 Issue 3, p. 435-449

[16] **Jensen, M.R.; Møller, T.H. ir Pedersen, T.B.** *Specifying OLAP Cubes On XML Data*. *Journal of Intelligent Information Systems* Gruodis, 2001, Volume 17 Issue 2-3, p. 255-280

[17] **W3C**, Extensible Markup Language (XML), 2006

[18] **Jensen M.IR., Møller T.H., Pedersen T.B.**, Specifying OLAP Cubes On XML Data, 2001

[19] **Hamlet, R.** 2002. *Random Testing. Encyclopedia of Software Engineering*.

[20] **Mottu J.M., Baudry B., Traon. Y.** *Model Transformation Testing : oracle issue*, In proceedings of MoDeVVa workshop, colocated with ICST'08, Lillehammer, Norway, 2008

[21] **Oracle**, *The SQL Language* [Interaktyvus], 2012, [Žiūrėta 2012.05.22] http://www.postgresql.org/docs/8.2/static/sql.html

[22] **Brass S., Goldberg C.,** *Detecting Logical Errors in SQL Queries*, Halle Universitetas, 2004

[23] **Goldberg C., Brass S.,** *Semantic Errors in SQL Queries: A Quite Complete List*, Halle Universitetas, 2004

[24] **Gould C., Su Z., Devanbu P.**, *JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications*, Davis, 2004

[25] **Gould C., Su Z., Devanbu P.**, *Static Checking of Dynamically Generated Queries in Database Applications*, Davis, 2004

[26] **CodeProject,** *Check Validity of SQL Server Stored Procedures, Views and Functions*, 2006

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

ActiveX – tai yra karkasas skirtas nepriklausomai nuo naudojamos programavimo kalbos aprašyti pakartotinai panaudojamus programinius komponentus.

ADO (angl. ActiveX Data Objects) – Microsoft's ActiveX Duomenų Objektai skirti priėjimui prie duomenų šaltinių rinkinys.

COM (angl. COM Component Object Model) – Komponentinis Objektinis Modelis – tai komponentų sąsajos standartas.

OLAP Kubas – OLAP įrankių naudojama duomenų struktūra.

OLAP (angl. Online Analytical Processing) – analitinio duomenų apdorojimo technologija.

MDA (angl. Model driven architecture) – Modeliais paremta architektūra.

MDE (angl. Model driven engineering) – Modeliais paremta inžinerija.

RDL (angl. Report Definition Language) – Ataskaitų apibrėžimo kalba.

RTOLAP (angl. Real Time OLAP) – Realaus laiko OLAP.

SQL (angl. Structured Query Language) – Struktūrizuota užklausų kalba.

UML (angl. Unified modeling language) – Unifikuota Modeliavimo Kalba.

XML (angl. Extensible Markup Language) – XML yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba

XPath – kelio standartas duomenų struktūrai XML dokumente pasiekti

9. PRIEDAI

9.1. Kubo modelio aprašymas

Dimensijos aprašymas

Dimensijos aprašo pavyzdys:

<Dimension name="DimensijosPavadinimas" sourceTable="DuomenųLentelė" joinKey="StulpelioPavadinimas">

</Dimension>

Laukų aprašymas

5 lentelė. Dimensijos aprašo laukų aprašymas.

Laukas	Paaiškinimas	Pavyzdys
Name	Dimensijos pavadinimas	CallDim
sourceTable	Lentelės pavadinimas iš kurios bus imami duomenys	AllActivitiesReportVW
joinKey	Stulpelis pagal kurį bus jungiama su aukščiau hierarchijoje esančia lentele	CallUUID
Neprivalomi laukai		
tableAlias	Alternatyvus lentelės pavadinimas.	KitiDarbuotojai
customJoin	Specialus jungimo tipas, naudojamas kai reik lenteles tarpusavyje sujungti ne standartiniu būdu.	Dimensijoslentele.ID = KitiDarbuotojai.ID AND KitiDarbuotojai.Statusas = ‘Dirba‘

Filtro aprašymas

Filtro aprašo pavyzdys:

<Field Width="0.8" name="FiltroLaukas" friendlyName="MatomasPavadinimas" friendlyNameTag="LaukoPavadinimpŽymė" sourceColumn="StulpelioPavadinimas" groupName="FitroGrupė" groupNameTag=" FitroGrupėsŽymė" index="40"/>

Laukų aprašymas

6 lentelė. Filtro aprašo laukų aprašymas.

Laukas	Paaiškinimas	Pavyzdys
Name	Filtro pavadinimas	FiltroLaukas
friendlyName	Užrašas matomas ekrane.	MatomasPavadinimas
friendlyNameTag	Požymis pagal kurį bus ieškomas vertimas į kitas kalbas.	LaukoPavadinimpŽymė
sourceColumn	Lentelės stulpelis iš kurio bus imami duomenys	StulpelioPavadinimas
groupName	Filtrų grupės pavadinimas	FitroGrupė
groupNameTag	Filtrų grupės pavadinimo požymis.	FitroGrupėsŽymė
Index	Indeksas pagal kurį nustatomas eiliškumas	40
Neprivalomi laukai		
Width	Plotis	0.8

Parametro aprašymas

Parametro aprašo pavyzdys:

<Parameter name="ParametroPavadinimas" friendlyName="Parametras" friendlyNameTag="ParametroPavadinimoPožymis" sourceColumn=" StulpelioPavadinimas " groupName="ParametroGrupė" groupNameTag=" ParametroGrupėsPožymis" index="80"/>

Laukų aprašymas

7 lentelė. Parametro aprašo laukų aprašymas.

Laukas	Paaiškinimas	Pavyzdys
name	Parametro pavadinimas	ParametroPavadinimas
friendlyName	Užrašas matomas ekrane.	Parametras
friendlyNameTag	Požymis pagal kurį bus ieškomas vertimas į kitas kalbas.	ParametroPavadinimoPožymis
sourceColumn	Lentelės stulpelis iš kurio bus imami duomenys	StulpelioPavadinimas
groupName	Filtrų grupės pavadinimas	ParametroGrupė
groupNameTag	Filtrų grupės pavadinimo požymis.	ParametroGrupėsPožymis
Index	Plotis	80

9.2. Kubo modelio testo rezultatai

8 lentelė. Kubo modelio testo rezultatai.

Įvedimo rinkinių elementų aibės ilgis	4	5	6	7	8	9	10	11	12
Aptiktų klaidų kiekis	80	136	129	122	115	108	101	94	87
Sugeneruotų rinkinių kiekis	2000	1956	1855	1754	1653	1552	1451	1350	1249
Kiek laiko truko testavimas	1900	1892	1789	1694	1598	1497	1402	1303	1206
Bendras užtrukęs laikas, h	0,527778	1,053333	1,550278	2,020833	2,464722	2,880556	3,27	3,631944	3,966944
Bendras rastas klaidų kiekis	80	216	345	467	582	690	791	885	972
Bendras patikrintas rinkinių kiekis	2000	3956	5811	7565	9218	10770	12221	13571	14820
Aptiktų klaidų ir sugeneruotų rinkinių kiekio santykis	0,04	0,06953	0,069542	0,069555	0,06957	0,069588	0,069607	0,06963	0,069656

Įvedimo rinkinių elementų aibės ilgis	13	14	15	16	17	18	19	20	21
Aptiktų klaidų kiekis	80	73	66	59	52	45	38	31	24
Sugeneruotų rinkinių kiekis	1148	1047	946	845	744	643	537	436	336
Kiek laiko truko testavimas	1110	1011	913	816	718	621	519	420	325
Bendras užtrukęs laikas, h	4,275278	4,556111	4,809722	5,036389	5,235833	5,408333	5,5525	5,669167	5,759444
Bendras rastas klaidų kiekis	1052	1125	1191	1250	1302	1347	1385	1416	1440
Bendras patikrintas rinkinių kiekis	15968	17015	17961	18806	19550	20193	20730	21166	21502
Aptiktų klaidų ir sugeneruotų rinkinių kiekio santykis	0,069686	0,069723	0,069767	0,069822	0,069892	0,069984	0,070764	0,071101	0,071429

Įvedimo rinkinių elementų aibės ilgis	22	23	24
Aptiktų klaidų kiekis	17	10	3
Sugeneruotų rinkinių kiekis	236	136	36
Kiek laiko truko testavimas	228	131	35
Bendras užtrukęs laikas, h	5,822778	5,859167	5,868889
Bendras rastas klaidų kiekis	1457	1467	1470
Bendras patikrintas rinkinių kiekis	21738	21874	21910
Aptiktų klaidų ir sugeneruotų rinkinių kiekio santykis	0,072034	0,073529	0,083333

9.3. Kubo modelio pavyzdys

```
<?xml version="1.0" encoding="utf-8"?>
<ReportingModel name="Activities" xmlns="http://softdent.lt/ReportingModelSchema.xsd">
  <Group name="CallGroup" IsRoot="true">
    <Dimension name="CallDim" sourceTable="AllActivitiesReportVW" joinKey="CallUUID">
      <Field Width="0.8" name="DateDayField" friendlyName="Date" friendlyNameTag="Date"
sourceColumn="Convert(Char(10),CallDate,102)" isSelfDescribing="true" groupName="Time" groupNameTag="Time"
index="60"/>
      <Field Width="0.9" name="DateWeekField" friendlyName="Week" friendlyNameTag="Week"
          sourceColumn="CONVERT(nchar(4), DATEPART(yyyy, CallDate)) + ' week ' + RIGHT('00' + CONVERT(varchar,
dbo.GetWeekNumber(CallDate)),2)"
          isSelfDescribing="true" groupName="Time" groupNameTag="Time" index="50"/>
      <Field Width="0.8" name="DateMonthField" friendlyName="Month" friendlyNameTag="Month"
          sourceColumn="CONVERT(nchar(4), DATEPART(yyyy, CallDate)) + ' ' + RIGHT('00' + CONVERT(varchar,
DATEPART(mm, CallDate)), 2)"
          isSelfDescribing="true" groupName="Time" groupNameTag="Time" index="40"/>
      <Field Width="0.8" name="DateQuarterField" friendlyName="Quarter" friendlyNameTag="Quarter"
          sourceColumn="STR(DATEPART(yyyy, CallDate),4) + ' ' + STR(DATEPART(qq, CallDate),1) + 'Q'"
          isSelfDescribing="true" groupName="Time" groupNameTag="Time" index="30"/>
      <Field Width="0.8" name="DateYearField" friendlyName="Year" friendlyNameTag="Year"
sourceColumn="STR(DATEPART(yyyy, CallDate),4)" isSelfDescribing="true" groupName="Time" groupNameTag="Time"
index="20"/>
      <Field Width="0.8" name="CallDateTotal" friendlyName="Total" friendlyNameTag="Total" sourceColumn="CallDateTotal"
isSelfDescribing="true" groupName="Time" groupNameTag="Time" index="10"/>
      <Field name="CallCount" friendlyName="All Activities" friendlyNameTag="AllActivities" format="auto"
          sourceColumn="COUNT(DISTINCT AllActivitiesReportVW.CallUUID)" isValue="true" isSelfDescribing="true"
groupName="Activity " groupNameTag="Activity" index="100"/>
      <Field name="PersonCalls" friendlyName="Person Calls" friendlyNameTag="PersonCalls" format="auto"
          sourceColumn="COUNT(DISTINCT CallPersonUUID)" isValue="true" isSelfDescribing="true" groupName="Activity "
groupNameTag="Activity" index="110"/>

      <Field name="OrganisationCalls" friendlyName="Organisation Calls" friendlyNameTag="OrganisationCalls" format="auto"
          sourceColumn="COUNT(DISTINCT CallOrganisationUUID)" isValue="true" isSelfDescribing="true"
groupName="Activity " groupNameTag="Activity" index="120"/>
      <Field name="Calls" friendlyName="Calls" friendlyNameTag="Calls" format="auto"
          sourceColumn="( COALESCE(COUNT(DISTINCT CallPersonUUID), 0)+COALESCE(COUNT(DISTINCT
CallOrganisationUUID),0) )" isValue="true" isSelfDescribing="true" groupName="Activity " groupNameTag="Activity"
index="130"/>
      <Field name="Meetings" friendlyName="Meetings" friendlyNameTag="Meetings" format="auto"
          sourceColumn="COUNT(DISTINCT CallGroupUUID)" isValue="true" isSelfDescribing="true" groupName="Activity "
groupNameTag="Activity" index="140"/>
      <Field name="CallsAndMeetings" friendlyName="Calls+Meetings" friendlyNameTag="CallsPlusMeetings" format="auto"
          sourceColumn="( COALESCE(COUNT(DISTINCT CallPersonUUID), 0)+COALESCE(COUNT(DISTINCT
CallOrganisationUUID),0)+COALESCE(COUNT(DISTINCT CallGroupUUID),0) )" isValue="true" isSelfDescribing="true"
groupName="Activity " groupNameTag="Activity" index="140"/>
      <Field name="DoubleVisitCount" friendlyName="Double Visits" friendlyNameTag="DoubleVisits" format="auto"
          sourceColumn="COUNT(DISTINCT AllActivitiesReportVW.DoubleVisitUUID)" isValue="true" isSelfDescribing="true"
groupName="Activity " groupNameTag="Activity" index="150"/>
      <Field name="DurationField" friendlyName="Duration (min.)" friendlyNameTag="DurationInMinutes" format="auto"
sourceColumn="SUM(DATEDIFF(minute, dbo.AllActivitiesReportVW.TimeFrom, dbo.AllActivitiesReportVW.TimeTo))"
isValue="true" isSelfDescribing="true" groupName="Duration " groupNameTag="Duration"/>
```

```
<Field Width="1.5" name="CallOwnerEmployeeFullName" friendlyName="Employee" friendlyNameTag="Employee"
sourceColumn="FullName" isSelfDescribing="true" groupName="Created by" groupNameTag="CreatedBy" index="71"/>
<Dimension name="AllActivitiesPersonAnOrganisationCountDim"
sourceTable="AllActivitiesPersonAnOrganisationCountReportVW" joinKey="CallUUID">
<!--Persons group-->
<Field name="PersonsInCallsField" friendlyName="Persons in Calls" friendlyNameTag="PersonsInCalls"
sourceColumn="COUNT(DISTINCT PersonInCall)" isValue="true" isSelfDescribing="true" groupName="Persons"
groupNameTag="Persons" index="2010" />
<Field name="PersonsInMeetingsField" friendlyName="Persons in Meetings" friendlyNameTag="PersonsInMeetings"
sourceColumn="COUNT(DISTINCT PersonInMeeting)" isValue="true" isSelfDescribing="true" groupName="Persons"
groupNameTag="Persons" index="2020" />
<Field name="PersonsInCallsAndMeetingsField" friendlyName="Persons in Calls and Meetings"
friendlyNameTag="PersonsInCallsAndMeetings" sourceColumn="COUNT(DISTINCT PersonInCallAndMeeting)" isValue="true"
isSelfDescribing="true" groupName="Persons" groupNameTag="Persons" index="2030" />
<!--eof Persons group-->
<!--Organisations group-->
<Field name="OrganisationsInCallsField" friendlyName="Organisations in Calls"
friendlyNameTag="OrganisationsInCalls" sourceColumn="COUNT(DISTINCT OrganisationInCall)" isValue="true"
isSelfDescribing="true" groupName="Organisations" groupNameTag="Organisations" index="2110" />
<Field name="OrganisationsInPersonCallsField" friendlyName="Organisations in Person Calls"
friendlyNameTag="OrganisationsInPersonCalls" sourceColumn="COUNT(DISTINCT OrganisationInPersonCall)" isValue="true"
isSelfDescribing="true" groupName="Organisations" groupNameTag="Organisations" index="2120" />
<Field name="OrganisationsInOrganisationsCallsField" friendlyName="Organisations in Organisations Calls"
friendlyNameTag="OrganisationsInOrganisationsCalls" sourceColumn="COUNT(DISTINCT OrganisationInOrganisationCall)"
isValue="true" isSelfDescribing="true" groupName="Organisations" groupNameTag="Organisations" index="2130" />
<!--eof Organisations group-->
<!--Activities per customer group-->
<Field name="CallsPerPersonField" friendlyName="Calls per Person" friendlyNameTag="CallsPerPerson"
sourceColumn="CASE WHEN COUNT(DISTINCT PersonInCall) > 0 AND COUNT(DISTINCT CallPersonUUID) > 0
THEN (COUNT(DISTINCT PersonInCall) * 1.0) / (COUNT(DISTINCT CallPersonUUID) * 1.0) ELSE 0.0 END"
isValue="true" isSelfDescribing="true" groupName="ActivitesPerCustomer" groupNameTag="ActivitesPerCustomer"
index="2210" />
<Field name="OrganisationCallsPerOrganisationField" friendlyName="Organisation Calls per Organisation"
friendlyNameTag="OrganisationCallsPerOrganisation"
sourceColumn="CASE WHEN COUNT(DISTINCT OrganisationInOrganisationCall) > 0 AND COUNT(DISTINCT
CallOrganisationUUID) > 0 THEN (COUNT(DISTINCT OrganisationInOrganisationCall) * 1.0) / (COUNT(DISTINCT
CallOrganisationUUID) * 1.0) ELSE 0.0 END"
isValue="true" isSelfDescribing="true" groupName="ActivitesPerCustomer" groupNameTag="ActivitesPerCustomer"
index="2220" />
<Field name="CallsPerOrganisationField" friendlyName="Calls per Organisation"
friendlyNameTag="CallsPerOrganisation"
sourceColumn="CASE WHEN COUNT(DISTINCT OrganisationInCall) > 0 AND ( COALESCE(COUNT(DISTINCT
CallPersonUUID), 0)+COALESCE(COUNT(DISTINCT CallOrganisationUUID),0) ) > 0 THEN (COUNT(DISTINCT
OrganisationInCall) * 1.0) / (( COALESCE(COUNT(DISTINCT CallPersonUUID), 0)+COALESCE(COUNT(DISTINCT
CallOrganisationUUID),0) ) * 1.0) ELSE 0.0 END"
isValue="true" isSelfDescribing="true" groupName="ActivitesPerCustomer" groupNameTag="ActivitesPerCustomer"
index="2230" />
<Field name="CallsPerCustomerField" friendlyName="Calls per Customer" friendlyNameTag="CallsPerCustomer"
sourceColumn="CASE WHEN (COUNT(DISTINCT PersonInCall) + COUNT(DISTINCT
OrganisationInOrganisationCall)) > 0 AND ( COALESCE(COUNT(DISTINCT CallPersonUUID),
0)+COALESCE(COUNT(DISTINCT CallOrganisationUUID),0) ) > 0 THEN ((COUNT(DISTINCT PersonInCall) +
```

```
COUNT(DISTINCT OrganisationInOrganisationCall)) * 1.0) / (( COALESCE(COUNT(DISTINCT CallPersonUUID),
0)+COALESCE(COUNT(DISTINCT CallOrganisationUUID),0) ) * 1.0) ELSE 0.0 END"
            isValue="true" isSelfDescribing="true" groupName="ActivitesPerCustomer" groupNameTag="ActivitesPerCustomer"
index="2240" />
        <!--eof Activities per customer group-->
      </Dimension>
      <Dimension name="ActivityTypeDim" sourceTable="[GetActivityTypesList]()" joinKey="ActivityType"
tableAlias="_activityTypesList">
        <Field Width="1.3" name="ActivityType" friendlyName="Activity Type" friendlyNameTag="ActivityType"
sourceColumn="ActivityType" groupName="Activity" groupNameTag="Activity" index="80"/>
        <Parameter name="ActivityTypeParam" friendlyName="Activity Type" friendlyNameTag="ActivityType"
sourceColumn="ActivityType" groupName="Activity" groupNameTag="Activity" index="80"/>
      </Dimension>
      <Dimension name="CallEmployeeFullNameVWDim" sourceTable="EmployeeFullNameVW" joinKey="EmployeeUUID">
        <Parameter name="CallEmployeeParam" friendlyName="Employees" friendlyNameTag="Employees"
sourceColumn="EmployeeFullName" ColumnSizeInPercents="50" groupName="Created by" groupNameTag="CreatedBy"
index="200"/>
        <Dimension name="UserEmployeesDim" sourceTable="UserEmployee" joinInFilter="true" joinKey="EmployeeUUID">
          <Dimension name="UserListVWDim" sourceTable="UserListVW" joinInFilter="true" joinKey="UserUUID">
            <Parameter name="UserParam" friendlyName="Users" friendlyNameTag="Users" sourceColumn="FullName"
isVisible="false" groupName="Created by" groupNameTag="CreatedBy" index="200"/>
          </Dimension>
        </Dimension>
      </Dimension>
      <Dimension name="CompaniesByEmployeesDim" sourceTable="CompanyEmployee" joinKey="EmployeeUUID">
        <Dimension name="CompaniesDim" sourceTable="Company" joinKey="CompanyUUID">
          <Field Width="1.5" name="CompanyNameField" friendlyName="Company" friendlyNameTag="Company"
sourceColumn="CompanyName" groupName="Created by" groupNameTag="CreatedBy" index="65"/>
        </Dimension>
      </Dimension>
      <Dimension name="CompaniesByEmployeesFilterDim" sourceTable="CompanyAllEmployeesVW"
joinKey="EmployeeUUID" joinInFilter="true" tableAlias="CompanyEmployeeFilter">
        <Dimension name="CompaniesFilterDim" sourceTable="Company" joinKey="CompanyUUID" joinInFilter="true"
tableAlias="CompanyFilter">
          <Parameter name="CompanyNameParam" friendlyName="Company" isNullable="true" friendlyNameTag="Company"
sourceColumn="CompanyName" groupName="Created by" groupNameTag="CreatedBy" index="195"/>
        </Dimension>
      </Dimension>
      <Dimension name="SalesLineEmployeeDim" sourceTable="SalesLineEmployee" joinKey="EmployeeUUID">
        <Dimension name="SalesLineDim" sourceTable="SalesLine" customJoin="SalesLine.SalesLineUUID =
SalesLineEmployee.SalesLineUUID AND (SalesLine.IsActive = '1' OR SalesLine.IsActive IS NULL)" joinKey="SalesLineUUID">
          <Field Width="1.5" name="SalesLineField" friendlyName="Team" friendlyNameTag="Team"
sourceColumn="SalesLineName"  groupName="Created by" groupNameTag="CreatedBy" index="70"/>
          <Parameter name="SalesLineParam" friendlyName="Teams" friendlyNameTag="Teams"
sourceColumn="SalesLineName" dataViewFilter = "IsActive= 'true'" groupName="Created by" groupNameTag="CreatedBy"
index="200"/>
        </Dimension>
      </Dimension>
      <Dimension name="CallStatusDim" sourceTable="CallStatus" joinKey="CallStatusUUID">
        <Field Width="1.3" name="CallStatus" friendlyName="Status" friendlyNameTag="Status" sourceColumn="CallStatus"
groupName="Activity" groupNameTag="Activity" index="90"/>
```

```
        <Parameter name="CallStatusParam" friendlyName="Status" friendlyNameTag="Status" sourceColumn="CallStatus"
groupName="Activity" groupNameTag="Activity" index="90"/>
      </Dimension>
      <Dimension name="CallTypeDim" sourceTable="CallPersonalType" joinKey="CallPersonalTypeUUID">
        <Field Width="1.3" name="CallTypeParam" friendlyName="Call Type" friendlyNameTag="CallType"
sourceColumn="CallPersonalType" groupName="Activity" groupNameTag="Activity" index="100"/>
        <Parameter name="CallTypeParam" friendlyName="Call Type" friendlyNameTag="CallType"
sourceColumn="CallPersonalType" groupName="Activity" groupNameTag="Activity" index="100"/>
      </Dimension>
      <Dimension name="MeetingTypeDim" sourceTable="CallGroupType" joinKey="CallGroupTypeUUID">
        <Field name="MeetingEquivalentsField" friendlyName="Meeting Equivalents" friendlyNameTag="MeetingEquivalents"
format="auto"
          sourceColumn="SUM(CallEquivalent)" isValue="true" isSelfDescribing="true" groupName="Activity "
groupNameTag="Activity" index="150"/>

        <Field name="MeetingEquivalentsPlusCallCountField" friendlyName="Calls+Meeting Equivalents"
friendlyNameTag="CallsPlusMeetingEquivalents"
          sourceColumn="( COALESCE(SUM(CallEquivalent),0)+COALESCE(COUNT(DISTINCT CallPersonUUID),
0)+COALESCE(COUNT(DISTINCT CallOrganisationUUID),0) )" isValue="true" isSelfDescribing="true" groupName="Activity "
groupNameTag="Activity" index="160"/>
        <Field Width="1.3" name="MeetingTypeParam" friendlyName="Meeting Type" friendlyNameTag="MeetingType"
sourceColumn="CallGroupType" groupName="Activity" groupNameTag="Activity" index="110"/>
        <Parameter name="MeetingTypeParam" friendlyName="Meeting Type" friendlyNameTag="MeetingType"
sourceColumn="CallGroupType" groupName="Activity" groupNameTag="Activity" index="110"/>
      </Dimension>
      <Dimension name="TaskTypeDim" sourceTable="CallOtherType" joinKey="CallOtherTypeUUID">
        <Field Width="1.5" name="TaskTypeParam" friendlyName="Other Activity Type" friendlyNameTag="OtherActivityType"
sourceColumn="CallOtherType" groupName="Activity" groupNameTag="Activity" index="120"/>
        <Parameter name="TaskTypeParam" friendlyName="Other Activity Type" friendlyNameTag="OtherActivityType"
sourceColumn="CallOtherType" groupName="Activity" groupNameTag="Activity" index="120"/>
      </Dimension>
      <Dimension name="TownDim" sourceTable="Town" joinKey="TownUUID" >
        <Field Width="1" name="OrganisationTown" friendlyName="Town" friendlyNameTag="Town" sourceColumn="Town"
groupName="Geography" groupNameTag="Geography" index="153"/>
        <Parameter name="TownParam" friendlyName="Towns" friendlyNameTag="Towns" sourceColumn="Town"
ColumnSizeInPercents="25" groupName="Geography" groupNameTag="Geography" index="200"/>
        <Dimension name="RegionDim" sourceTable="Region" joinKey="RegionUUID">
          <Field Width="1" name="Region" friendlyName="Region" friendlyNameTag="Region" sourceColumn="Region"
groupName="Geography" groupNameTag="Geography" index="154"/>
          <Parameter name="RegionParam" friendlyName="Regions" friendlyNameTag="Regions" sourceColumn="Region"
ColumnSizeInPercents="33" groupName="Geography" groupNameTag="Geography" index="200"/>
          <Dimension name="DistrictDim" sourceTable="District" joinKey="DistrictUUID">
            <Field Width="1" name="District" friendlyName="District" friendlyNameTag="District" sourceColumn="District"
groupName="Geography" groupNameTag="Geography" index="155"/>
            <Parameter name="DistrictParam" friendlyName="Districts" friendlyNameTag="Districts" sourceColumn="District"
ColumnSizeInPercents="33" groupName="Geography" groupNameTag="Geography" index="200"/>

            <Dimension name="CountryDim" sourceTable="Country" joinKey="CountryUUID">
              <Field Width="1" name="Country" friendlyName="Country" friendlyNameTag="Country" sourceColumn="Country"
groupName="Geography" groupNameTag="Geography" index="156"/>
              <Parameter name="CountryParam" friendlyName="Countries" friendlyNameTag="Countries"
sourceColumn="Country" ColumnSizeInPercents="33" groupName="Geography" groupNameTag="Geography" index="200"/>
```

```
            </Dimension>
          </Dimension>
        </Dimension>
      </Dimension>
      <Dimension name="CallBrandSupportingMessagesVWDim" sourceTable="CallBrandSupportingMessagesVW"
joinKey="CallUUID">
        <Dimension name="BrandSupportingMessages" sourceTable="BrandSupportingMessage"
joinKey="BrandSupportingMessageUUID">
          <Parameter name="MessageParam" friendlyName="Topics" friendlyNameTag="Topics"
sourceColumn="BrandSupportingMessage" groupName="Brand" groupNameTag="Brand" index="200"/>
        </Dimension>
      </Dimension>
    </Dimension>
    <Dimension name="IsDoubleVisitDim" sourceTable="[GetIsDoubleVisitList]()" joinKey="IsDoubleVisit"
tableAlias="_IsDoubleVisitList">
      <Parameter name="IsDoubleVisitParam" friendlyName="Double Visit" friendlyNameTag="DoubleVisit"
SelectionMode="One" sourceColumn="DoubleVisitCaption" groupName="Double Visit" groupNameTag="DoubleVisit"
index="200"/>
    </Dimension>
    <Dimension name="DoubleVisitEmployeeDim" sourceTable="EmployeeFullNameVW" tableAlias="dve"
customJoin="dve.EmployeeUUID = AllActivitiesReportVW.DoubleVisitEmployeeUUID" joinKey="EmployeeUUID">
      <Field Width="1.5" name="DoubleVisitEmployeeField" friendlyName="Double Visit Employee"
friendlyNameTag="DoubleVisitEmployee" sourceColumn="EmployeeFullName" groupName="Double Visit"
groupNameTag="DoubleVisit" index="151"/>
      <Parameter name="DoubleVisitEmployeeParam" friendlyName="Double Visit Employee"
friendlyNameTag="DoubleVisitEmployee" sourceColumn="EmployeeFullName" groupName="Double Visit"
groupNameTag="DoubleVisit" index="200"/>
    </Dimension>
    <Dimension name="DoubleVisitSalesPersonDim" sourceTable="EmployeeFullNameVW" tableAlias="dvsp"
customJoin="dvsp.EmployeeUUID = AllActivitiesReportVW.SalesPersonUUID" joinKey="EmployeeUUID">
      <Field Width="1.5" name="DoubleVisitSalesPersonField" friendlyName="Sales Person" friendlyNameTag="SalesPerson"
sourceColumn="EmployeeFullName" groupName="Double Visit" groupNameTag="DoubleVisit" index="152"/>
      <Parameter name="DoubleVisitSalesPersonParam" friendlyName="Sales Person" friendlyNameTag="SalesPerson"
sourceColumn="EmployeeFullName" groupName="Double Visit" groupNameTag="DoubleVisit" index="200"/>
    </Dimension>
  </Group>
  <Group name="PersonGroup">
    <Dimension name="PersonDim" sourceTable="Person" joinKey="PersonUUID">
      <Field Width="1.5" name="FullName" friendlyName="Person" friendlyNameTag="Person"
sourceColumn="COALESCE(Person.LastName+' ','')+COALESCE(Person.FirstName,'')" isSelfDescribing="true"
groupName="Person" groupNameTag="Person" index="170"/>
      <Field Width="1" name="PersonIndicator"
          sourceColumn="PersonSingleIndicator"
          isDynamic="true"
          dynamicTable="PersonSingleIndicatorType"
          dynamicColumn="PersonSingleIndicatorType"
          dynamicKey="PersonSingleIndicatorTypeUUID"
          dynamicJoinKey="PersonSingleIndicatorTypeUUID"
          dynamicFunctionToUse=" dbo.GetPersonSingleIndicator(Person.PersonUUID, '{0}') "
          groupName="Person" groupNameTag="Person" index="210"/>
      <Parameter name="PersonSingleIndicatorParam"
            sourceColumn="PersonSingleIndicator"
```

```
            isDynamic="true"
            indicatorTypeTable="PersonSingleIndicatorType"
            indicatorTypeColumn="PersonSingleIndicatorType"
            indicatorTypeKey="PersonSingleIndicatorTypeUUID"
            indicatorTable="PersonSingleIndicator"
            indicatorColumn="PersonSingleIndicator"
            indicatorKey="PersonSingleIndicatorUUID"
            dynamicFunctionToUse=" dbo.GetPersonSingleIndicatorUUID(Person.PersonUUID, '{0}') " groupName="Person
Indicators" groupNameTag="PersonIndicators" index="200"/>

    <Dimension name="PersonOrganisationDim" sourceTable="PersonMainWorkplaceVW" joinKey="PersonUUID">
      </Dimension>
      <Dimension name="PersonSpecialityDim" sourceTable="PersonMainSpecialityVW" joinKey="PersonUUID">
        <Field Width="1.5" name="PersonMainSpeciality" friendlyName="Speciality" friendlyNameTag="Speciality"
sourceColumn="Speciality" groupName="Person" groupNameTag="Person" index="190"/>
        <Dimension name="PersonMainSpecialityFilterDim" sourceTable="Speciality" joinKey="SpecialityUUID">
          <Parameter name="PersonMainSpecialityParam" friendlyName="Person Specialities"
friendlyNameTag="PersonSpecialities" sourceColumn="Speciality" ColumnSizeInPercents="50" groupName="Person"
groupNameTag="Person" index="200"/>
        </Dimension>
      </Dimension>
      <Dimension name="PersonTargetsDim" sourceTable="dbo.GetAllPersonTargets('{0}','{1}',{2},'{3}',{4},{5})"
joinKey="PersonUUID" sourceTableContainsParameters="true" tableAlias="AllPersonTargets">
        <Field Width="0.5" name="PersonTargetStatusName"
            friendlyName="Person Target"
            friendlyNameTag="PersonTarget"
            sourceColumn ="[PersonTargetStatusName]"
            isSelfDescribing ="true"
            groupName="Person" groupNameTag="Person" index="200"/>
      </Dimension>
      <Dimension name="PersonStatusDim" sourceTable ="PersonStatus" joinKey="PersonStatusUUID">
        <Field Width="1.1" name="PersonStatus" friendlyName="Person Status" friendlyNameTag="PersonStatus"
sourceColumn="PersonStatus" groupName="Person" groupNameTag="Person" index="180"/>
     <Parameter name="PersonStatusParam" friendlyName="Person Statuses" friendlyNameTag="PersonStatusess"
sourceColumn="PersonStatus" ColumnSizeInPercents="50" groupName="Person" groupNameTag="Person" index="200"/>
      </Dimension>
      <Dimension name="TargetByPersonFilter_PersonBrandTargetDim" sourceTable="PersonBrandTarget"
joinKey="PersonUUID" joinInFilter="true">
        <Dimension name="TargetByPersonFilter_TargetStatusDim" sourceTable="PersonTargetStatus"
joinKey="PersonTargetStatusUUID" joinInFilter="true">
          <Parameter name="TargetByPersonFilter_PersonTargetStatusParam" friendlyName="Person Target Statuses"
friendlyNameTag="PersonTargetStatuses" sourceColumn="PersonTargetStatusName" ColumnSizeInPercents="50"
groupName="Targeting" groupNameTag="Targeting" index="160"/>
        </Dimension>
        <Dimension name="TargetByPersonFilter_TimePeriodDim" sourceTable="TimePeriod" joinKey="TimePeriodUUID"
joinInFilter="true">
          <Parameter name="TargetByPersonFilter_TimePeriodParam" friendlyName="Time Period (Target)"
friendlyNameTag="TimePeriodOfTarget" sourceColumn="TimePeriodName" SelectionMode="One" groupName="Targeting"
groupNameTag="Targeting" index="170"/>
        </Dimension>

        <Dimension name="TargetByPersonFilter_BrandDim" sourceTable="Brand" joinKey="BrandUUID" joinInFilter="true">
```

```
          <Parameter name="TargetByPersonFilter_BrandParam" friendlyName="Brands (Target)"
friendlyNameTag="BrandsOfTarget" sourceColumn="BrandName" ColumnSizeInPercents="50" dataViewFilter="IsUsed=true"
groupName="Targeting" groupNameTag="Targeting" index="180"/>
        </Dimension>
        <Dimension name="TargetByPersonFilter_BrandIndicatorDim" sourceTable="BrandIndicator"
joinKey="BrandIndicatorUUID" tableAlias="personTargetingBrandIndicators" joinInFilter="true">
          <Parameter name="TargetByPersonFilter_BrandIndicatorParam" friendlyName="Brand Indicators (Target)"
friendlyNameTag="BrandIndicatorsOfTarget" sourceColumn="BrandIndicator" ColumnSizeInPercents="50"
groupName="Targeting" groupNameTag="Targeting" index="190"/>
        </Dimension>
        <Dimension name="TargetByPersonFilter_SalesLineDim" sourceTable="SalesLine"
customJoin="PersonBrandTarget.SalesLineUUID = personTargetingSalesLines.SalesLineUUID AND
(personTargetingSalesLines.IsActive = '1' OR personTargetingSalesLines.IsActive IS NULL)" joinKey="SalesLineUUID"
tableAlias="personTargetingSalesLines" joinInFilter="true">
          <Parameter name="TargetByPersonFilter_SalesLineParam" friendlyName="Teams" friendlyNameTag="Teams"
sourceColumn="SalesLineName" dataViewFilter = "IsActive= 'true'" ColumnSizeInPercents="50" groupName="Targeting"
groupNameTag="Targeting" index="200"/>
        </Dimension>
        <Dimension name="TargetByPersonFilter_PersonTargetTypeDim" sourceTable="PersonTargetType"
joinKey="PersonTargetTypeUUID" tableAlias="personTargetingPersonTargetTypes" joinInFilter="true">
          <Parameter name="TargetByPersonFilter_PersonTargetTypeParam" friendlyName="Person Target Types"
friendlyNameTag="PersonTargetTypes" sourceColumn="PersonTargetTypeName" ColumnSizeInPercents="50"
groupName="Targeting" groupNameTag="Targeting" index="210"/>
        </Dimension>
      </Dimension>
    </Dimension>
  </Group>
  <Group name="OrganisationGroup">
    <Dimension name="OrganisationDim" sourceTable="Organisation" joinKey="OrganisationUUID">
      <Field Width="1.5" name="OrganisationName" friendlyName="Organisations" friendlyNameTag="Organisations"
sourceColumn="OrganisationName" groupName="Organisation" groupNameTag="Organisation" index="220"/>
      <Field Width="1.5" name="AddressLine1" friendlyName="Address Line 1" friendlyNameTag="AddressLine1"
sourceColumn="AddressLine1" groupName="Organisation" groupNameTag="Organisation" index="230"/>
      <Field Width="1.5" name="AddressLine2" friendlyName="Address Line 2" friendlyNameTag="AddressLine2"
sourceColumn="AddressLine2" groupName="Organisation" groupNameTag="Organisation" index="240"/>
      <Field Width="1" name="OrganisationIndicator"
          sourceColumn="OrganisationSingleIndicator"
          isDynamic="true"
          dynamicTable="OrganisationSingleIndicatorType"
          dynamicColumn="OrganisationSingleIndicatorType"
          dynamicKey="OrganisationSingleIndicatorTypeUUID"
          dynamicJoinKey="OrganisationSingleIndicatorTypeUUID"
          dynamicFunctionToUse=" dbo.GetOrganisationSingleIndicator(Organisation.OrganisationUUID, '{0}') "
          groupName="Organisation" groupNameTag="Organisation" index="280"/>
      <Parameter name="OrganisationNameParam" friendlyName="Organisations" friendlyNameTag="Organisations"
sourceColumn="OrganisationName" ColumnSizeInPercents="50" groupName="Organisation" groupNameTag="Organisation"
index="200"/>
      <Parameter name="OrganisationSingleIndicatorParam"
            sourceColumn="OrganisationSingleIndicator"
            isDynamic="true"
            indicatorTypeTable="OrganisationSingleIndicatorType"
            indicatorTypeColumn="OrganisationSingleIndicatorType"
```

```
            indicatorTypeKey="OrganisationSingleIndicatorTypeUUID"
            indicatorTable="OrganisationSingleIndicator"
            indicatorColumn="OrganisationSingleIndicator"
            indicatorKey="OrganisationSingleIndicatorUUID"
            dynamicFunctionToUse=" dbo.GetOrganisationSingleIndicatorUUID(Organisation.OrganisationUUID, '{0}') "
groupName="Organisation Indicators" groupNameTag="OrganisationIndicators" index="200"/>
      <Dimension name="OrganisationTypeDim" sourceTable="OrganisationType" joinKey="OrganisationTypeUUID">
        <Field Width="1.2" name="OrganisationTypeName" friendlyName="Organisation Type"
friendlyNameTag="OrganisationType" sourceColumn="OrganisationTypeName" groupName="Organisation"
groupNameTag="Organisation" index="250"/>
        <Parameter name="OrganisationTypeNameParam" friendlyName="Organisation Types"
friendlyNameTag="OrganisationTypes" sourceColumn="OrganisationTypeName" ColumnSizeInPercents="25"
groupName="Organisation" groupNameTag="Organisation" index="200"/>
      </Dimension>
      <Dimension name="OrganisationStatusDim" sourceTable="OrganisationStatus" joinKey="OrganisationStatusUUID">
        <Field Width="1.2" name="OrganisationStatus" friendlyName="Organisation Status"
friendlyNameTag="OrganisationStatus" sourceColumn="OrganisationStatus" groupName="Organisation"
groupNameTag="Organisation" index="260"/>
        <Parameter name="OrganisationStatusParam" friendlyName="Organisation Statuses"
friendlyNameTag="OrganisationStatuses" sourceColumn="OrganisationStatus" ColumnSizeInPercents="25"
groupName="Organisation" groupNameTag="Organisation" index="200"/>
      </Dimension>
      <Dimension name="OrganisationTargetsDim" sourceTable="dbo.GetAllOrganisationTargets('{0}','{1}',{2},'{3}')"
joinKey="OrganisationUUID" sourceTableContainsParameters="true" tableAlias="AllOrganisationTargets">
        <Field Width="0.5" name="OrganisationTargetStatusName"
            friendlyName="Organisation Target"
            friendlyNameTag="OrganisationTarget"
            sourceColumn ="[OrganisationTargetStatusName]"
            isSelfDescribing ="true"
            groupName="Organisation" groupNameTag="Organisation" index="270"/>
      </Dimension>
    </Dimension>
  </Group>
  <Group name="BrandGroup">
    <Dimension name="CallBrandDim" sourceTable="CallBrand" joinKey="CallUUID">
      <Dimension name="BrandDim" sourceTable="Brand" joinKey="BrandUUID" customJoin="CallBrand.BrandUUID =
Brand.BrandUUID AND Brand.IsUsed = 1">
        <Field Width="1.5" name="BrandName" friendlyName="Brand" friendlyNameTag="Brand" sourceColumn="BrandName"
groupName="Product" groupNameTag="Product" index="290"/>
        <Field Width="1" name="BrandCode" friendlyName="Brand Code" friendlyNameTag="BrandCode"
sourceColumn="BrandCode" groupName="Product" groupNameTag="Product" index="291"/>
        <Field Width="1.5" name="BrandIndicator" groupName="Product" groupNameTag="Product"
            sourceColumn="BrandIndicator"
            isDynamic="true"
            dynamicTable="BrandIndicatorType"
            dynamicColumn="BrandIndicatorType"
            dynamicKey="BrandIndicatorTypeUUID"
            dynamicJoinKey="BrandIndicatorTypeUUID"
            dynamicFunctionToUse=" dbo.GetBrandIndicator(Brand.BrandUUID, '{0}') " index="300"/>
        <Field Width="2" name="BrandSupportingMessage"
            isSelfDescribing ="true"
            friendlyName="Message"
```

```
            friendlyNameTag="Message"
sourceColumn="dbo.GetBrandSupportingMessages(CallBrand.CallUUID, CallBrand.BrandUUID)"
            groupName="Product" groupNameTag="Product" index="310"/>
        <Parameter name="BrandNameParam" friendlyName="Brands" friendlyNameTag="Brands"
sourceColumn="BrandName" ColumnSizeInPercents="30" dataViewFilter="IsUsed=true" groupName="Brand"
groupNameTag="Brand" index="198"/>
        <Parameter name="BrandIndicatorParam"
            sourceColumn="BrandIndicator"
            isDynamic="true"
            indicatorTypeTable="BrandIndicatorType"
            indicatorTypeColumn="BrandIndicatorType"
            indicatorTypeKey="BrandIndicatorTypeUUID"
            indicatorTable="BrandIndicator"
            indicatorColumn="BrandIndicator"
            indicatorKey="BrandIndicatorUUID"
            dynamicFunctionToUse=" dbo.GetBrandIndicatorUUID(Brand.BrandUUID, '{0}') " groupName="Brand"
groupNameTag="Brand" index="200"/>
      <Dimension name="PackageDim" sourceTable="Package" joinKey="PackageUUID" customJoin="Brand.BrandUUID =
Package.BrandUUID">
        <Parameter name="PackageNameParam" isVisible="false" friendlyName="Packages" friendlyNameTag="Packages"
sourceColumn="PackageName" ColumnSizeInPercents="25" groupName="Products" groupNameTag="Products"
index="120"/>
      </Dimension>

      <Dimension name="ProducerDim" sourceTable="Producer" joinKey="ProducerUUID">
        <Parameter name="ProducerNameParam" friendlyName="Producer" friendlyNameTag="Producer"
sourceColumn="ProducerName" ColumnSizeInPercents="25" groupName="Brand" groupNameTag="Brand" index="200"/>
      </Dimension>
       </Dimension>
      <Dimension name="DetailWeightDim" sourceTable="(SELECT CallUUID, COUNT(BrandUUID) AS CallBrandCount FROM
CallBrand GROUP BY CallUUID)" joinKey="CallUUID" tableAlias="cbc" customJoin="CallBrand.CallUUID = cbc.CallUUID">
        <Dimension name="DetailWeightInnerDim" sourceTable="DetailWeight" customJoin="DetailWeight.Position =
CallBrand.Position AND DetailWeight.DetailNo = cbc.CallBrandCount" joinKey="CallUUID">
          <Field Width="1.5" name="DetailWeightField"
            friendlyName="Detail Weight"
            friendlyNameTag="DetailWeight"
            sourceColumn ="SUM(COALESCE (DetailWeight.Weight,0))"
            isSelfDescribing ="true"
            groupName="Activity " groupNameTag="Activity" index="200" isValue="true"/>
        <Field Width="1.5" name="DetailWeightPercentageField"
            friendlyName="Detail Weight %"
            friendlyNameTag="DetailWeightPercentage"
            sourceColumn ="SUM(COALESCE (DetailWeight.Weight,0))"
            isSelfDescribing ="true"
            relatedFieldNameLeft="DetailWeightPercentageField"
          calculationFromReleatedFieldsFormula="=IIF(SUM(DetailWeightPercentageField, &amp;quot;Matrix1&amp;quot;)=0, 0,
SUM(DetailWeightPercentageField)*100/SUM(DetailWeightPercentageField, &amp;quot;Matrix1&amp;quot;))"
          calculationFromReleatedFieldsTOTALFormula="=IIF(SUM(DetailWeightPercentageField, &amp;quot;Matrix1&amp;quot;)=0, 0,
SUM(DetailWeightPercentageField)*100/SUM(DetailWeightPercentageField, &amp;quot;Matrix1&amp;quot;))"
          groupName="Activity " groupNameTag="Activity" index="200" isValue="true"/>
        </Dimension>
      </Dimension>
```

```
      <Dimension name="BrandPositionDim" sourceTable="CallBrandPositionVW" joinKey="PositionID"
customJoin="CallBrand.Position = CallBrandPositionVW.PositionID">
        <Parameter name="BrandPositionParam" friendlyName="Brand Position" friendlyNameTag="BrandPosition"
sourceColumn="PositionName" ColumnSizeInPercents="30" groupName="Brand" groupNameTag="Brand" index="199"/>
      </Dimension>
    </Dimension>
  </Group>
  <Group name="SamplesGiveawaysGroup">
    <Dimension name="CallSamplesDim" sourceTable="CallSample" joinKey="CallUUID">
      <Dimension name="SamplesDim" sourceTable="Package" joinKey ="PackageUUID">
        <Parameter name="SampleNameParam" friendlyName="Samples" friendlyNameTag="Samples"
sourceColumn="PackageName" groupName="Samples/Giveaways" groupNameTag="SamplesAndGiveaways" index="200"/>
      </Dimension>
    </Dimension>
    <Dimension name="CallGiveawaysDim" sourceTable="CallGiveaway" joinKey="CallUUID">
      <Dimension name="GiveawaysDim" sourceTable="Giveaway" joinKey ="GiveawayUUID">
        <Parameter name="GiveawayNameParam" friendlyName="Giveaways" friendlyNameTag="Giveaways"
sourceColumn="GiveawayName" groupName="Samples/Giveaways" groupNameTag="SamplesAndGiveaways" index="200"/>
      </Dimension>
    </Dimensijon>
  </Group>
 </ReportingModel>
```