LATVIJAS UNIVERSITĀTES
DATORIKAS FAKULTĀTES

DATU NOLIKTAVAS IZVEIDES METODES LIELO DATU LAIKMETĀ

PROMOCIJAS DARBS

Autors: **Jānis Zemnickis**
Studenta apliecības Nr.: jz10024
Darba vadītāja: Dr.dat. Laila Niedrīte

RĪGA 2024

ANOTĀCIJA

Darbā tiek piedāvātas jaunas metodes, kas nodrošina daļēji automātisku datu noliktavas kandidātshēmas ģenerāciju, izmantojot datu noliktavas formālās prasības, galveno organizācijas darbības rādītāju (KPI) specificēšanu un datu noliktavas datu modeļa paplašināšanu, izmantojot klientu atsauksmes. Metodes paredz izmantot klientu atsauksmes, kas ir izteiktas brīvā teksta formātā. Šīs atsauksmes tiek apstrādātas ar dabīgās valodas apstrādes rīkiem. No iegūtās informācijas tiek veikti speciāli aprēķini, lai iegūtu informāciju, kas tiek izmantota, lai specificētu jaunus organizācijas galvenos darbības rādītājus un to mērķu vērtības. Tiek noteikts, ka atribūti no galvenajiem darbības rādītājiem ir jāiekļauj organizācijā pastāvošajā datu noliktavas datu modelī, lai būtu iespējams sekot līdzi organizācijas attīstībai.

Atslēgas vārdi: Datu noliktava, brīvā teksta apstrāde, galvenie darbības rādītāji, datu modelis.

ABSTRACT

In this thesis, are presented new methods that allow to generate data warehouse conceptual data model from the data warehouse formal requirements and methods that allow to specify key performance indicators (KPI) and to extend the data warehouse data model by using customer feedback. Methods use customer feedback which is in free text format. Customer feedback is processed by natural language processing tools. In order to specify new key performance indicators and their target values special calculations are performed from processed customer feedback. Attributes from KPI should be included in data warehouse data model to monitor organizations performance.

Keywords: Data warehouse, natural language processing, key performance indicators, data model.

SATURS

Apzīmējumu saraksts

ACID (transakcijas) - *Atomicity, Consistency, Isolation, and Durability*

HDFS - *Hadoop Distributed File System*

SSD - *Solid-State Drive*

ETL - *Extract, Transform, Load*

OLAP - *Online analytical processing*

PHP - *Hypertext Preprocessor*

MVC - *Model View Controller*

CSS - *Cascading Style Sheets*

HTML - *HyperText Markup Language*

JSON - *JavaScript Object Notation*

SQL - *Structured Query Language*

GPU - *Graphics Processing Unit*

API - *Application Programming Interface*

NoSQL - *Not only SQL*

KPI - *Key Performance Indicator*

UML - *Unified Modeling Language*

ER (modelis) – *Entity Relationship*

CSV – *Comma Separated Values*

ML (algoritms) - *Machine Learning*

SoS - *System of Systems*

WEB - *World Wide Web*

NLP - *Natural language processing*

IEVADS

Datu noliktavas organizācijās tiek plaši lietotas visa veida datu analīzes vajadzībām, lēmumu pieņemšanas atbalstam, atskaišu ģenerēšanai, organizācijas galveno darbības rādītāju (turpmāk - KPI (*KPI – key performance indicator*)) uzraudzībai, vēsturisko datu saglabāšanai u.c. Tradicionālajās datu noliktavās kā datu avotus parasti izmanto strukturētus datus, kuri tiek pārveidoti vienotā datu modelī (multidimensionālā shēmā). Attīstoties tādām tehnoloģijām, kā lielie dati un ne relāciju (turpmāk - NoSQL (*NoSQL - not only SQL*)) datubāzes, ir notikušas izmaiņas datu noliktavas izstrādes metodēs un datu noliktavas arhitektūrā. Datu noliktavas izstrādes prasībām ir jāpievērš īpaša uzmanība (Prakash & Prakash, 2019). Ir būtiski, lai saglabātie dati datu noliktavā atbilstu organizācijas ieinteresēto personu vajadzībām. Kā viens no datu noliktavas izstrādes neveiksmes iemesliem, tiek minēts, ka pieejamie dati neatbilst ieinteresēto personu vajadzībām (Benkhaled & Berrabah, 2019). Ieinteresēto personu vajadzības parasti tiek noskaidrotas pirms datu noliktavas izstrādes, veicot prasību inženierijas aktivitāti.

Tēmas aktualitāte un novitāte

Mūsdienās datu apjomi strauji palielinās, pieaugot pieejamajiem atvērtajiem datiem, attīstoties sociālo tīklu platformām un sistēmām, kas ģenerē datus (Bimonte et al., 2021). Datu apjomiem palielinoties, pieaug potenciālās iespējas iegūt vērtīgu informāciju no tiem. Mūsdienās dati ir kļuvuši par vienu no visvērtīgākajiem īpašumiem (Al-Okaily et al., 2022). Organizācijas sāk vairāk orientēt savu darbību vērstu uz datiem (*data-driven*) (Bode et al., 2023). Pastāv viedoklis, ka tradicionāls datu noliktavas modelis un arhitektūra nespēj nodrošināt visas analīzes vajadzības pieaugošo datu apjoma kontekstā (Dehghani, 2022) un (Machado et al., 2022). Tradicionālas datu noliktavas nespēj apmierināt augošās vajadzības, lai integrētu un analizētu dažāda formāta datus no sociālajiem tīkliem, viedierīcēm un sensoriem. Kompānijas arvien vairāk ir ieinteresētas veikt nestrukturētu datu analīzi un integrēt nestrukturētus datus ar strukturētiem datiem (Bouaziz, et al., 2019). Organizācijām nav zināšanu par tās nestrukturētajiem datiem un kā tos apstrādāt, lai iegūtu vērtīgu informāciju (Baviskar et al., 2021).

Pastāv dažāda veida datu noliktavas izstrādes metodes, un viens no veidiem, kā metodes iedala ir:

- Prasību vadītas – metodes, kas nosaka iegūt datu noliktavas informācijas prasības, lai izstrādātu datu noliktavas datu modeli, atbilstoši ieinteresēto

personu vajadzībām, piemēram, metodes (Souza et al., 2012) un (Bimonte et al., 2021);

- Datu vadītas – metodes, kas nosaka ģenerēt datu noliktavas datu modeli, par pamatu izmantojot datu avota datubāzes shēmas struktūru un pieejamos datus, piemēram, metodes (Sellami et al., 2020) un (Romero & Abelló, 2010);
- Kombinētās - metodes, kas nosaka pielietot aktivitātes gan no prasību vadītām metodēm, gan no datu vadītām metodēm, piemēram, metode (Jukic & Nicholas, 2010).

Pastāvošās datu vadītās (dokumentu, kolonu un grafu orientētās) datu noliktavas izstrādes metodes nosaka transformācijas noteikumus, kā pārbūvēt datu avotu shēmas vienotā datu noliktavas datu modelī. Pieaugot datu apjomiem, ir būtiski iegūt datu noliktavas informācijas prasības no nestrukturētajiem datiem, jo, kā liecina pētījums (Rizkallah, 2017), gandrīz 80% procenti organizāciju nezina vai ir tikai neliela nojausma par organizācijas nestrukturētajiem datiem. Joprojām pastāv daudz neatklātas iespējas, ko var sasniegt analizējot nestrukturētus datus (Bach et al., 2019). Datu noliktavas izstrāde ir sarežģīts un laikietilpīgs process (Nordeen, 2020). Pastāv vairāki veidi kā automatizēt datu noliktavas izstrādi (Phipps & Davis, 2002), (Castellanos et al., 2009), (Pilipenko et al., 2019), bet pastāvošās metodes nenodrošina automatizācijas iespējas, lai izstrādātu datu noliktavas konceptuālās shēmas, izmantojot nestrukturētos datus.

KPI nosaka, kādi uzdevumi ir jāveic, lai ievērojami palielinātu organizācijas veiktspēju (Parmenter, 2015). Datu noliktavas tiek izmantotas kā rīks, kas nodrošina KPI uzraudzīšanas funkcionalitāti (Nasiri et al., 2015), (Krstev & Krneta, 2022). Informācija par organizācijas darbības stāvokli ir jāsaglabā datu noliktavā, lai nodrošinātu KPI uzraudzību. KPI nosaka tos atribūtus, kuriem ir jābūt pieejamiem datu noliktavā, bet, aplūkojot pastāvošās datu noliktavas izstrādes metodes, tas nenodrošina KPI izmantošanu datu noliktavas prasību inženierijas aktivitātēs.

Pētījuma zinātniskā novitāte

Pētījumos (Kozmina et al., 2018) un (Kozmina et al., 2019) tiek aplūkotas pastāvošās lielo datu apstrādes metodes un informācijas prasību aspekti lielo datu kontekstā. Lielo datu apstrādes metodes izmanto lielo datu tehnoloģijas dažādiem mērķiem - lai analizētu sistēmas lietotāju darbības (Fernandez-Garcia et al., 2015), brīva teksta analīzei (Cheptsov et al., 2014) vai, lai uzlabotu lielo datu pārvaldību, datu analīzi un vizualizācijas risinājumu (Tardio et al., 2015), bet no pētījumā aplūkotajām metodēm neviena metode nenosaka, kā analizēt datus, lai definētu jaunas informācijas prasības. Pētījumā tiek secināts, ka lielo datu apstrādes metodes var tikt attiecinātas uz kādu no datu noliktavas prasību inženierijas

posmiem, jo no 22 aplūkotajām metodēm, 53% gadījumos pastāv "augsta" iespēja piemērot kādu no prasību inženierijas posmiem, 14% gadījumos pastāv "vidēja" iespēja, bet 33% gadījumos no metodes apraksta, šādu informāciju nevarēja izsecināt. Tiek secināts, ka lielo datu apstrādes metodes var nebūt saistītas ar prasību inženieriju, bet, izmantojot lielo datu apstrādes tehnoloģijas, metodes darbības var tikt attiecinātas uz kādu no datu noliktavas prasību inženierijas posmu vai posmiem. Metodēs pielietotās tehnoloģijas var tikt izmantotas automātiskai vai daļēji automātiskai informācijas prasību iegūšanai, bet šobrīd neviena no pētījumos aplūkotajām metodēm nenodrošina šādu iespēju.

Promocijas darbā tika izstrādātas divas jaunas metodes - "KPI noteikšana izmantojot nestrukturētu tekstu" un "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm", ar kuru palīdzību var paplašināt datu noliktavas datu modeli un specificēt jaunus KPI, izmantojot brīvo tekstu – nestrukturētus datus. Metodes nosaka, ka brīvais teksts tiek apstrādāts ar dabīgās valodas apstrādes (turpmāk - NLP (NLP – *Natural Language Processing*)) rīku.

Metode "KPI noteikšana izmantojot nestrukturētu tekstu " ļauj specificēt jaunus KPI. KPI tiek specificēti atbilstoši līdzsvarotajai vadības kartei (*Balanced scorecard*) (Kaplan, 1992). Līdzsvarotā vadības karte sastāv no klientu apmierinātības rādītājiem, kuru noskaidrošanai var tikt izmantoti nestrukturēti dati kā datu avots. Brīvais teksts ir viens no nestrukturētu datu tipiem. Brīvais teksts var saturēt klientu vērtējumus, tai skaitā klientu atsauksmes. Kā vieni no klientu datu avotiem, var tikt izmantoti sociālo platformu dati.

Metode "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm" ļauj daļēji automātiskā veidā ģenerēt organizācijas kvantitatīvos KPI, to mērķa vērtības un paplašināt organizācijā pastāvošu datu noliktavas datu modeli. Metode nosaka, ka no brīvā teksta tiek iegūta informācija par vārdu saistībām teikumā un vārdu noskaņojumiem. Tiek veiktas speciālas kalkulācijas, lai ģenerētu organizācijas KPI un to kvantitatīvās vērtības. Iegūtie atribūti no ģenerētajiem KPI ir jāiekļauj datu noliktavas datu modelī, lai organizācijas varētu uzraudzīt savu veiktspēju atbilstoši KPI mērķa vērtībām. Saglabājot šos atribūtus datu noliktavā, organizācijas varēs sekot līdzi organizācijas attīstībai un uzlabot tās darbību. Metode nenodrošina ģenerēt jaunu datu noliktavas modeli, bet metode sniedz iespēju paplašināt organizācijā jau esošas datu noliktavas datu modeli ar jauniem atribūtiem.

Lai organizācija pielietotu metodes, tai jābūt pieejamiem konkrētās vai citas organizācijas klientu atsauksmju datiem par jomu, kurā tā darbojās. Var izmantot publisku datu avotu, kā tas ir darīts šīs metodes aprobācijas piemērā, kur tiek izmantoti tiešsaistes sociālā medija Twitter dati. Lai no klientu atsauksmēm varētu ģenerēt kvantitatīvus KPI,

klientu datiem ir jāsatur skaitliskas vērtības. Metodes aprobācijas piemērā ir aprakstīta situācija par uzņēmumu, kas darbojās ēdināšanas jomā.

Darba mērķis ir izstrādāt metodes, kas atbalsta datu noliktavas prasību iegūšanas posmu un paplašina datu noliktavas datu modeli, izmantojot nestrukturētus datus un valodas apstrādes tehnoloģijas.

Darba mērķa sasniegšanai tika izvirzīti vairāki **uzdevumi:**

1. Izpētīt datu noliktavas arhitektūru, datu noliktavas datu modeļa izstrādes metodes un to attīstību lielo datu kontekstā;
2. Veikt literatūras izpēti par tradicionālas datu noliktavas prasību iegūšanas metodēm;
3. Izpētīt datu noliktavas informācijas prasību formalizēšanas iespējas;
4. Veikt literatūras izpēti un analīzi par prasību inženierijas aktivitātēm lielo datu izstrādes projektos;
5. Izpētīt dabīgās valodas apstrādes tehnikas un rīkus, izvērtēt dabīgās valodas potenciālu kā datu avotu;
6. Aplūkot organizācijas KPI jēdzienu un darbības principus;
7. Piedāvāt jaunu vai uzlabot eksistējošu metodi, kas nodrošina ģenerēt datu noliktavas kandidātshēmas, izmantojot formālās datu noliktavas prasības;
8. Piedāvāt jaunas metodes, kas nodrošina iespēju sastādīt jaunus organizācijas KPI un paplašināt organizācijas datu noliktavas datu modeli, analizējot nestrukturētus datus;
9. Veikt metožu aprobāciju, uzlabojot datu noliktavas datu modeli un nosakot organizācijai KPI.

Darbā izvirzītās **tēzes**:

- Datu noliktavas prasības ir iespējams formalizēt, un no formālajām prasībām ir iespējams ģenerēt datu noliktavas kandidātshēmas;
- Brīvais teksts var tikt izmantots kā datu avots, lai specificētu jaunus organizācijas KPI, kas var kalpot kā informācijas prasības, lai paplašinātu datu noliktavas datu modeli.

Izvirzīto mērķu sasniegšanai pētījuma gaitā tika **izmantotas sekojošas pētniecības metodes**:

- Analītiskā metode - literatūras avotu analīze, lai aplūkotu un izpētītu konkrētu tēmu, problēmu un atrisinātu to;

- Salīdzinošā metode – veicot literatūras apskatu, salīdzināt vairākas metodes un metožu aprakstos definētās problēmas, izanalizēt trūkumus un problēmas, kas kalpo kā pamats jaunas metodes prasībām;
- Eksperimenta metode – praksē pielietot un pārbaudīt literatūras avotos iegūtās zināšanas, izstrādāt jaunu metodi, veicot praktiskus eksperimentus, uzlabojot metodes darbību balstoties uz eksperimentu rezultātiem;
- Rezultātu novērtējuma metode – izstrādātas metodes testēšana, metodes rezultātu novērtēšana un validācija;
- Aprakstošā metode – apkopojot iegūtos rezultātus, aprakstot jaunās metodes darbību un rezultātus publicējot zinātniskajos rakstos.

Darba rezultāti:

- Izpētītas datu noliktavas prasību formalizēšanas iespējas, izvēlēts pastāvošs metamodelis prasību formalizēšanai, kas tika paplašināts ar jaunām metamodeļa klasēm, kas ļauj saglabāt datu noliktavas formālās prasības, sasaistot tās ar ieinteresētajām personām un biznesa prasībām;
- Izvēlēta eksistējoša datu noliktavas izstrādes metode, kas tika paplašināta ar jaunām komponentēm;
- Atbilstoši paplašinātajai metodei un metamodelim, kas paredz formalizēt datu noliktavas informācijas prasības, tika izstrādāts rīks iReq, kas nodrošina funkcionalitāti - ievadīt grafiskajā saskarnē un saglabāt formālās datu noliktavas prasības, kā arī ģenerēt datu noliktavas kandidātshēmas;
- Izstrādātas metodes “KPI noteikšana izmantojot nestrukturētu tekstu” un “Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm”, kuras paredz analizēt brīvo tekstu, lai nodrošinātu KPI specificēšanas funkcionalitāti, kā arī nodrošina iespējas paplašināt eksistējošas datu noliktavas datu modeli;
- Metožu darbība ir aprobēta ēdināšanas jomā, kur tiek paplašināts uzņēmumā pastāvošs datu noliktavas datu modelis. Aprobācijas rezultātā noskaidroti divi jauni datu noliktavas datu modeļa atribūti un iegūti septiņi KPI un to mērķu vērtības;
- Pētījuma rezultāti ir publicēti zinātniskos rakstos un prezentēti starptautiskajās konferencēs.

Rezultātu aprobācija, publikācijas

Darba pētījumu rezultāti ir publicēti 5 zinātniskos rakstos, kas ir iekļauti SCOPUS datubāzē:

1. Zemnickis, J. (2023). Data Warehouse Data Model Improvements from Customer Feedback. Baltic Journal of Modern Computing, 11(3). (ieguldījums 100%);
2. Zemnickis, J., Niedrite, L., & Kozmina, N. (2020). A Little Bird Told Me: Discovering KPIs from Twitter Data. In Databases and Information Systems: 14th International Baltic Conference, DB&IS 2020, Tallinn, Estonia, June 16–19, 2020, Proceedings 14 (pp. 161-175). Springer International Publishing. (ieguldījums: 60%);
3. Kozmina, N., Niedrite, L., & Zemnickis, J. (2018). Information requirements for big data projects: A review of state-of-the-art approaches. In Databases and Information Systems: 13th International Baltic Conference, DB&IS 2018, Trakai, Lithuania, July 1-4, 2018, Proceedings 13 (pp. 73-89). Springer International Publishing. (ieguldījums: 33%);
4. Kozmina, N., Niedrite, L., & Zemnickis, J. (2019). Perspectives of Information Requirements Analysis in Big Data Projects. In Databases and Information Systems X (pp. 109-124). IOS Press. (ieguldījums: 33%)
5. Kozmina, N., Niedrite, L., & Zemnickis, J. (2017, April). Gathering Formalized Information Requirements of a Data Warehouse. In International Conference on Enterprise Information Systems (Vol. 2, pp. 217-224). SCITEPRESS. (ieguldījums: 60%).

Ar citu tematu saistīti zinātniskie raksti:

- Arnicans, G., Niedrite, L., Solodovnikova, D., Virbulis, J., & Zemnickis, J. (2021, June). Towards a system to monitor the virus's aerosol-type spreading. In International Conference on Computer Science Protecting Human Society Against Epidemics (pp. 95-106). Cham: Springer International Publishing.

Darba rezultātus darba autors prezentēja starptautiskās konferencēs:

1. "*Databases and Information Systems: 13th International Baltic Conference, DB&IS 2018*" (Lietuva), prezentēts pētījums "*Perspectives of Information Requirements Analysis in Big Data Projects*";
2. "*Databases and Information Systems: 14th International Baltic Conference, DB&IS 2020*" (Igaunija), prezentēts pētījums "*A Little Bird Told Me: Discovering KPIs from Twitter Data*";

3. "*82nd International Scientific Conference of the University of Latvia, 2024*" (Latvija), prezentēts pētījums "*Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm*".

Darbs sastāv no ievada, pamatdaļas ar 5 nodaļām, rezultātiem un secinājumiem:

1. **nodaļā** aplūkots datu noliktavas jēdziens, arhitektūra, datu noliktavas modelēšanas metodes, kā arī datu noliktavas attīstība lielo datu un NoSQL datu bāžu kontekstā. Veicot literatūras apskatu, tiek aplūkotas jaunākās metodes un identificētas problēmas datu noliktavas izstrādē. Tiek analizēti trūkumi un nepilnības datu noliktavas izstrādes metodēs lielo datu kontekstā;
2. **nodaļā** tiek aplūkotas tradicionālas datu noliktavas prasību vākšanas metodes. Tiek aplūkots informācijas prasību jēdziens. Izvērtētas iespējas formalizēt datu noliktavas informācijas prasības un aplūkotas pastāvošās metodes datu noliktavas prasību iegūšanā. Šajā nodaļā autors paplašina eksistējošas datu noliktavas izstrādes metodi, formālo prasību metamodeli un apraksta rīku "iReq", ar kura palīdzību ir iespējams ģenerēt datu noliktavas kadidātshēmas, izmantojot formālās datu noliktavas prasības;
3. **nodaļā** tiek aplūkots lielo datu un lielo datu avotu jēdziens. Tiek veikta literatūras izpēte un analīze par lielo datu apstrādes metodēm un iespējām to darbību attiecināt uz kādu no datu noliktavas prasību inženierijas aktivitātēm. Tiek analizēti trūkumi un nepilnības par prasību inženierijas aktivitātēm lielo datu apstrādes metodēs;
4. **nodaļā** tiek aplūkots temats par dabīgo valodu un NLP rīkiem. Veicot literatūras pārskatu, tiek novērtētas kopīgas problēmas apstrādājot dabīgo valodu. Tiek izvērtēta iespēja izmantot dabīgo valodu kā datu avotu, iegūstot datu noliktavas informācijas prasības, un aplūkots nestrukturētu datu potenciāls kā informācijas avots. Otrs temats, kas tiek aplūkots šajā nodaļā, ir organizācijas KPI. Tiek aplūkotas KPI īpašības un aplūkota iespēja KPI izmantot kā datu noliktavas informācijas prasības;
5. **nodaļā** tiek prezentētas jaunas metodes. Metode "KPI noteikšana izmantojot nestrukturētu tekstu", kas aprakstīta nodaļā 5.1., paredz apstrādāt nestrukturētus datus, nodrošināt datu analīzei nepieciešamo informāciju, lai specificētu jaunus organizācijas KPI, izmantojot speciālus grafus. Metode "Datu noliktavas datu modeļa uzlabojumi no klientu

atsauksmēm", kas aprakstīta nodaļā 5.2., līdzīgi kā pirmā metode, paredz apstrādāt nestrukturētus datus. Otrā metode nodrošina daļēji automātiskā veidā ģenerēt organizācijas kvantitatīvos KPI. Izmantojot atribūtus no KPI, tiek paplašināts organizācijā pastāvošās datu noliktavas datu modelis. Abu metožu darbība tiek aprobēta ar piemēru ēdināšanas nozarē, sastādot organizācijas KPI vai uzlabojot organizācijā pastāvošās datu noliktavas datu modeli.

1 DATU NOLIKTAVAS ARHITEKTŪRA UN MODEĻI

Šajā nodaļā tiks apskatīts tradicionālas datu noliktavas jēdziens, arhitektūra un datu noliktavas datu modeļi, kā arī tiek aplūkots temats par datu noliktavas attīstību lielo datu laikmetā.

1.1 Datu noliktava

Datu noliktava ir subjekt-orientētu, integrētu, vēsturisku datu kopums, lai atbalstītu organizāciju svarīgu lēmumu pieņemšanā (Kimball & Ross, 2011). Datu noliktavas ir pazīstamas jau kopš deviņdesmitajiem gadiem, tās nodrošina organizāciju ar attīstībai svarīgu informāciju. Datu noliktavas ir plaši izmantotas dažādās nozarēs, kā piemēram, banku, apdrošināšanas, pārdošanas u.c. Datu noliktavas sniedz organizācijas lēmumu pieņēmējiem vērtīgu informāciju (Kimball & Ross, 2011). Datu noliktavās tiek saglabāti liela apjoma attīrīti dati, kas ielasīti no avotu sistēmām un transformēti formātā, kas sniedz maksimālo vērtību analīzes vajadzībām un organizācijas lēmumu pieņēmējiem (Halim et al., 2020).

Datu noliktavas datu modelis parasti sastāv no kubiem un dimensijām, ko sauc par multidimesionālu datu modeli. *Kubs* multidimensionālajā datu modelī reprezentē kādu *faktu*, piemēram, pirkums vai rezervācija. Fakta *notikums* ir atgadījums, ko var raksturot ar *mērījumiem*, piemēram, cena, daudzums. Fakta notikumus un mērījumus analizē kopā ar saistītajām *dimensijām*, piemēram, produkts vai klients.

Dimensiju tabulas ir denormalizētas un tiek izmantotas, lai atfiltrētu nepieciešamos faktus. *Hierarhijas* sastāv no diskrētiem dimensiju atribūtiem, kas saistīti ar viens pret viens attiecību un nosaka, kā fakti var tikt apkopoti (*aggregated*) vai atlasīti (Golfarelli et al., 1998). Fakta mērījumi tiek aprēķināti izmantojot *agregācijas* funkcijas, piemēram, SUM vai AVG. Kubi tiek analizēti izmantojot OLAP funkcijas:

- Augšupejošās (*roll-up*) – notikumi tiek agregēti;
- Lejupejošās (*drill-down*) – notikumi tiek detalizēti;
- Dalīšanas (*slice-and-dice*) – faktu dalīšana apakšfaktos.

OLAP funkcijas, izmantojot atskaišu rīkus, sniedz vienkāršu un intuitīvu datu analīzes funkcionalitāti (Eggert & Alberts, 2020). Pastāv vairāki multidimensonāla modeļa veidi, viens no vienkāršākajiem multidimensionāla modeļa veidiem ir zvaigznes shēma (Giovinazzo, 2000). Zvaigznes shēma sastāv no fakta, kurš ir saistīts ar dimensijām.

Dati datu noliktavas datubāzē parasti tiek ielasīti no organizācijas iekšējām vai ārējām avota sistēmām, izmantojot *ETL tehnoloģiju*. ETL tehnoloģija apzīmē trīs galvenās darbības (Souibgui et al., 2019):

1. Datu iegūšana (*Extract*) – datu iegūšanu no avota sistēmas;
2. Datu transformācija (*Transform*) – datu transformēšana (attīrīšana, agregācija, konsolidācija u.c.) pareizā formātā un struktūrā, lai tā būtu izmantojama analīzes vajadzībām;
3. Datu saglabāšana (*Load*) – transformētie dati tiek ielasīti datu noliktavā.

ETL rīki nodrošina transformācijas, agregācijas un visa veida izņēmumu un noteikumu pārvaldību (Michael & Ahirao, 2020).

1.2 Datu noliktavu veidi un attīstība

Pastāv vairākas metodes un arhitektūras, kā implementēt datu noliktavu. Laikam ejot, ir notikušas inovācijas izstrādes paņēmienos un datu noliktavas arhitektūrā (Yang et al., 2019). Šajā nodaļā tiks aplūkotas tradicionālas, vairāku līmeņu datu noliktavas arhitektūras, kā arī arhitektūras, kas kļuvušas zināmas relatīvi nesen. Datu noliktavas arhitektūra ir tehnika, kas specificē datu komunikācijas plūsmas, datu kopu struktūru un piekļuvi datiem (Jameel et al., 2022).

1.2.1 Tradicionālas viena un vairāku līmeņu datu noliktavas

Kā viens no senāk zināmajiem veidiem, kā iedalīt datu noliktavas arhitektūru, ir noteikt līmeņu skaitu (Devlin, 1996) un (Golfarelli & Rizzi, 2009).

Viena līmeņa arhitektūrā dati netiek fiziski ielasīti datu noliktavā. Datu noliktava tiek implementēta, veidojot multidimensionālus datubāzes skatus, par pamatu izmantojot operacionālās sistēmas datubāzes tabulas vai skatus (Golfarelli & Rizzi, 2009). Par uz operacionālu sistēmu tiek saukta datu noliktavas avota sistēma. Viena līmeņa arhitektūra praksē ir reti pielietota. Būvējot datu noliktavu pēc šāda tipa arhitektūras, var tikt ietaupīta servera vieta uz diska, bet ir arī trūkumi, kā piemēram, grūti atšķirt operacionālos sistēmas datus no analītiskajiem datiem. Lielas datu noliktavas noslodzes gadījumā operacionālās sistēmas datu bāzes servera darbība var tikt negatīvi ietekmēta.

Divu līmeņu arhitektūra sastāv no datu avotu sistēmu līmeņa un datu noliktavas līmeņa (Golfarelli & Rizzi, 2009). Datu noliktavā dati ir ielasīti izmantojot ETL rīkus. Datu noliktavas līmenī var pastāvēt speciālas datuves, kas ir apkšķopa no datiem datu noliktavā. Datuves tiek veidotas specifisku servisu vajadzībām, pielietojot speciālas kalkulācijas konkrētā servisa vajadzībām. Atskaitēs un OLAP rīkos izmanto datus no datu noliktavas vai no datuvēm. Divu līmeņu arhitektūrā dati fiziski tiek ielasīti datu noliktavā un datuvēs.

Trīs līmeņu arhitektūra ir līdzīga divu līmeņu arhitektūrai. Šis līmenis sastāv no avota sistēmu līmeņa, datu noliktavas līmeņa un rekonsilācijas līmeņa. Starp avotu sistēmu līmeni

un datu noliktavas līmeni pastāv rekonsilācijas līmenis, kas dažkārt tiek dēvēts arī kā operacionālais datu glabāšanas (*operational data store*) līmenis. Rekonsilācijas līmenī dati tiek integrēti, laboti, attīrīti un sagatavoti transformācijai datu noliktavas līmenī.

1.2.2 Datu ezera arhitektūra

Datu ezers (*Datalake*) ir centralizēta datu glabātuve, kas nodrošina iespēju saglabāt neapstrādātus, orģināla formāta, dažāda tipa, nestrukturētus, daļēji strukturētus, strukturētus liela apjoma datus. Datu ezers dod iespēju organizācijām pieņemt labākus lēmumus organizācijas attīstībai piedāvājot interaktīvus atskaišu, vizualizācijas, lielo datu rīkus, kā arī mašīnmācīšanās un reālā laika analīzes iespējas (Nambiar & Mundra, 2022).

Ar tradicionāliem relāciju datubāžu datu tipiem tiek saprasti *integer, varchar, decimal*, u.c., bet 21. gadsimta sākumā parādījās jauni daudzveidīgāki datu tipi kā *teksts, video, audio, attēli* u.c. Internetam kļūstot aizvien populārākam un pieejamākam, pieauga arī datu apjomi. Tradicionālas datu noliktavas ar tradicionāliem ETL rīkiem un datu glabātuvēm vairs efektīvi netika galā ar dažādajiem datu tipiem un lieliem datu apjomiem (Hai, 2021), (Zagan & Danubianu, 2020) un (Cherradi & Haddadi, 2022).

Datu noliktavas un datu ezers ir savstarpēji aizvietojamas tehnoloģijas, bet tās nav vienādas tehnoloģijas (Nambiar & Mundra, 2022). Datu ezers un datu noliktavas ir veidotas, lai iegūtu vērtīgu informāciju no datiem un atbalstītu organizācijas lēmumu pieņemšanā (Inmon, 2016), (Llave, 2018) un (Madera & Laurent, 2016). Datu noliktavās dati tiek apstrādāti un filtrēti, lai tos varētu saglabāt vienotā, iepriekš definētā datu modelī, bet datu ezers atļauj saglabāt neapstrādātus un nestrukturētus datus. Datu noliktavas parasti tiek būvētas ar konkrētu izmantošanas mērķi un atbilstoši konkrētām prasībām, bet datu ezera izmantošanas mērķis nav stingri definēts, ideālā gadījumā tas var tikt izmantots jebkurām vajadzībām (Nambiar & Mundra, 2022). Datu ezeram, salīdzinot ar datu noliktavām, nav stingri noteikta datu modeļa, struktūras un arhitektūras, abiem risinājumiem ir savas priekšrocības un trūkumi konkrētos gadījumos.

Dodot iespēju organizācijām saglabāt datus centralizētā veidā, datu ezers kļuva aizvien populārāks. Datu ezers var saturēt dažādus datu formātus, tai skaitā arī nestrukturētus un daļēji strukturētus formātus. Tas dod iespēju saglabāt datus bez jebkādām transformācijām (Hai, 2021). Vairumā gadījumu organizācijās nav izmantots nestrukturētu un daļēji strukturētu datu potenciāls (Hai, 2021). Datu ezera arhitektūra nodrošina centralizētu, efektīvu un elastīgu datu apstrādi no dažādiem datu avotiem, kas dod priekšrocības datu-vadītai (*data driven*) pārvaldībai organizācijā. Datu ezera arhitektūra ir viegli mērogojama, jo ir nodalīta datu apstrāde un datu saglabāšana.

Datu ezera arhitektūra ir laika gaitā attīstījusies, sākotnēji datu ezera arhitektūra sastāvēja no viena slāņa. Šī viena slāņa arhitektūra paredzēja izmantot kopīgu vidi un saglabāt vienuviet neapstrādātus un apstrādātus datus. Šāda arhitektūra attīstījās līdztekus Hadoop videi, kas nodrošināja dažāda formāta liela apjoma datu saglabāšanu par zemām izmaksām. Taču šāda arhitektūra neļāva veikt datu apstrādi un lietotāju darbību žurnalēšanu. Datu ezeram attīstoties, tiek nodalīti atsevišķi datu slāņi (Ravat & Zhao, 2019):

- Neapstrādātu datu slānis – visa tipa dati bez apstrādes tiek ielasīti un saglabāti to dabiskajā formātā uz servera. Datu ielasīšana var notikt ar pakešapstrādes tehnoloģiju (*batch*) reālā laikā vai hibrīda veidā. Neapstrādāto datu slānī lietotājiem ir iespēja atrast datus bez modifikācijām;
- Datu apstrādes slānis – šīs slānis ir paredzēts, lai lietotāji varētu veikt sev nepieciešamās manipulācijas ar datiem atbilstoši vajadzībām. Lietotāji iegūto rezultātu un starprezultātu var saglabāt šajā slānī. Datu apstrāde var notikt pa partijām vai reālā laikā. Datu apstrādes slānī var veikt pieprasījumus, datu kopu apvienošanu un agregācijas;
- Datu pieejas slānis – nodrošina pieeju analītiskajiem datiem. Šajā slānī pēc pašapkalpošanās principa var tikt veidoti objekti nepieciešamajām vajadzībām – atskaitēm, analīzei, mašīnmācīšanās uzdevumiem, u.c.;
- Datu pārvaldības slānis – šī slāņa funkcionalitāte ir saistīta ar visiem pārējiem slāņiem. Šis slānis nodrošina datu drošību, kvalitāti, dzīvesciklu, pieeju un metadatu pārvaldību.

Līdzīgi kā tradicionālas datu noliktavas, arī datu ezera risinājumam ir savi trūkumi (Nargesian et al., 2019) un (Gupta & Giri, 2018):

- Augstas ieviešanas un uzturēšanas izmaksas. Dažas izmaksas sākotnēji var samazināt izvēloties mākoņrisinājumu, bet tas var sadārdzināt vēlāk uzturēšanas izmaksas. Ir iespēja datu ezeru ieviest lokāli, izmantojot atvērtā koda programmatūru Hadoop;
- Pārvaldības grūtības – lai pārvaldītu datu ezeru ir jānodrošina vairāki aspekti, kā piemēram, datu dublēšana, datu drošība un servera ietilpība;
- Speciālistu trūkums – lai pārvaldītu un attīstītu datu ezeru ir nepieciešami datu zinātnieki (*data scientists*) un atbilstoši programmatūras izstrādātāji;
- Datu drošība – ir jāveido speciāli mehānismi, lai pareizi pārvaldītu datus un datu piekļuvi;

- Servera resursi – lai arī datu ezers efektīvi apstrādā lielus datu apjomus, līdztekus datu apjomi aug ātrāk nekā servera resursu pieejamība.

1.2.3 "Lakehouse" arhitektūra

Lakehouse (Armbrust et al., 2021) ir datu bāzes vadības sistēma, kas darbojās izmantojot tiešas pieejas un salīdzinoši lēto diska atmiņu. Tai pašā laikā, tā nodrošina tradicionālas datu bāzes vadības sistēmas analītiskās funkcijas - ACID transakcijas, datu versionēšanu, auditēšanu, indeksēšanu, kešošanu un vaicājumu optimizēšanu. Lakehouses apvieno labākās īpašības no datu ezera un datu noliktavas, tai ir zemas atmiņas izmaksas un iespēja saglabāt datus atvērtos, brīvi pieejamos formātos, kā arī tai ir spēcīga datu pārvaldības un optimizācijas funkcionalitāte. Lakehouse risinājums ir īpaši piemērots mākoņa platformām, kuras nodala skaitļošanas un datu saglabāšanas vides. Dažādas skaitļošanas programmas var darboties pēc pieprasījuma uz nodalītiem servera mezgliem (*node*). Viens no piemēriem ir GPU (*graphical processing unit*) klasteris priekš mašīnmācīšanās uzdevumiem, kas izmanto GPU un kopējo atmiņu, lai piekļūtu datiem no lokāla HDFS servera.

Tradicionālas datu noliktavas arhitektūrā tiek izmantotas tehnoloģijas un paņēmieni kā uzlabot veiktspēju, piemēram, izmantojot ātras darbības SSD diskus, saglabāt un pārvaldīt statistiku par datiem, veidojot indeksus un optimizējot datu formātus. Lakahouse nenodrošina iespēju mainīt datu formātus, bet pastāv citas iespējas kā uzlabot veiktspēju – kešošanas tehnoloģija, jaunu datu struktūru izveide, indeksu pievienošana un statistikas saglabāšana. Lakehouse nodrošina DataFrame[1] lietojumprogrammu saskarni (*API - Application Programming Interface*). Lietojumprogrammu saskarne DataFrame nodrošina lietotājam plaši pazīstamās datu bāzes tabulas abstrakcijas lietošanas iespēju. SparkSQL sistēmas nodrošina pazīstamo SQL sintakses lietošanu.

1.2.4 "Data mesh" arhitektūra

Data mesh arhitektūra liek mainīt esošo tradicionālo paradigmu, lai organizācijas patiesi kļūtu datu orientētas, izstrādājot datu modeļus, kas ir konceptuāli pretēji tradicionālajiem datu modeļiem. Data mesh arhitektūra ir īpaši piemērota lielām organizācijām ar lielām struktūrvienībām un nodalītiem domēniem. Data mesh datu modeļi primāri ir orientēti uz produktu (Machado et al., 2022). Data mesh datu modelis sastāv no vairākiem domēniem. Domēni parasti reprezentē kādu organizācijas struktūrvienību. Data mesh galvenais mērķis ir nodrošināt decentralizētu datu arhitektūru, kas nodrošina liela apjoma analītisko datu pieejamību. Data mesh balstās uz četriem galvenajiem principiem:

[1] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

- Domēni datu modelī ir decentralizēti, veidoti produkta vai servisa vajadzībām;
- Dati ir kā produkts – datiem tiek piešķirta galvenā nozīme un tie kalpo kā produkts attiecīgajam biznesa procesam;
- Pašapkalpojoša datu platforma (*self-serve data platform*) - servisa orientētas komandas ir atbildīgas par platformas uzturēšanu un pārvaldību;
- Sadalīta kalkulāciju pārvaldība – katrs serviss un atbildīgā komanda veido biznesa vajadzībām nepieciešamās kalkulācijas.

Data mesh nosaka, ka domēniem ir jābūt sadalītiem atbilstoši organizācijas struktūrai un produktiem. Izstrādes komandas ir atbildīgas par konkrētu domēnu, šīs izstrādes komandas ir konkrētas biznesa struktūrvienības sastāvdaļa. Šādā veidā komandām ir lielāka kompetence par konkrēto produktu un biznesa pārstāvjiem ir tiešas iespējas pieprasīt vajadzīgās informācijas prasības. Saistībā ar datiem Data mesh darbojās pēc "produkta orientētā domāšana" principa, šādi datu plūsmas, saglabāšanas aspekti tiek atstāti apakšplānā. Galvenās priekšrocības, ko nodrošina data mesh ir:

- Decentralizētas komandas, kas sastāv no konkrēta domēna pārstāvjiem un skaidri pārredzamas īpašumtiesības un atbildība pār datu produktiem;
- Izstrādes vajadzībām pašapkalpojošas platformas, lai pārvaldītu un uzturētu infrastruktūru;
- Vadlīnijas, kā nodrošināt kvalitāti, prasību analīzi un drošību;
- Domēnu atšķirīgu tehnoloģiju savietojamība.

Pastāv vairākas data mesh implementācijas. Zalando implementācijā (WEB, l) tiek pielietota data mesh arhitektūra lielai organizācijai, kas darbojās apģērbu industrijā. Organizācijā pastāvēja datu ezera risinājums, bet pieaugot datu avotiem un prasībām, izstrādes komandas kļuva par vājo posmu, jo nevarēja apkalpot visas vajadzības. Zalando risinājums bija pārveidots no vienotas datu arhitektūras uz sadalītu, darboties spējīgu datu arhitektūru. Šajā risinājumā tika ieviests princips, ka lietotājs vai kāds biznesa serviss var pats piekļūt un veidot jaunus kalkulācijas procesus atbilstoši biznesa vajadzībām. Lietotājiem tiek piešķirtas tiesības izmantot klasterus, rīkus un pieeja datu apstrādes videi, kas dod iespēju bez infrastruktūras komandas iesaistes ieviest vajadzīgās biznesa prasības.

1.2.5 Datu fabrikas arhitektūra

George Kurian 2015. gadā pirmo reizi prezentēja datu fabrikas jēdzienu, ieviešot organizācijā "NetApp" datu fabrikas datu stratēģiju (WEB, e). Uzņēmums Gartner (Zaidi rt al., 2019) vēlāk attīstīja datu fabrikas jēdzienu - datu fabrikas arhitektūra ļauj pielietot un kombinēt dažādas integrācijas tehnikas, datu saglabāšanas platformas. Datu fabrika ir

projektēta, lai sniegtu iespēju vairākkārtīgi izmantot izstrādātos integrācijas risinājumus, nodrošinātu sematiski vienotu pieeju datiem un vienotu datu piegādi. Datu fabrikas datu noliktavas arhitektūra ir balstīta uz loģiskās datu noliktavas principu (tradicionālas viena līmeņa arhitektūras). Loģiska datu noliktava nosaka izmantot avota sistēmu datus tos fiziski neielasot datu noliktavā, kas ļauj apvienot vairākas tehnoloģijas, bet saglabā lietotājiem vienotu saskarni datu piekļuvei.

Galvenais datu fabrikas uzsvars ir uz metadatiem. Datu fabrikas gadījumā metadati sastāv no divām komponentēm:

- Datu kataloga;
- Informācijas grafa.

Tiek noteikts, ka ir jāizmanto "aktīvie metadati", tas nozīmē, ka metadati tiek automātiski ģenerēti un pārvaldīti izmantojot mākslīgā intelekta paņēmienus, piemēram, mašīnmācīšanas paņēmienus. Analīzes un atskaišu rīki nav daļa no datu fabrikas arhitektūras.

1.3 Datu noliktavas modelēšana

Lai izstrādātu datu noliktavu, nepieciešams izvēlēties datu noliktavas modelēšanas metodi, kas nosaka modelēšanas un izstrādes principus, kā arī projekta un risku pārvaldību (Yessad & Labiod, 2016). Metodes jeb metodoloģijas jēdziens atšķīrās dažādu autoru skatījumā. Autori (Avison & Fitzgerald, 2003) definē metodoloģiju kā "Metodoloģija ir likumu, darbības posmu, dokumentācijas, procedūru, tehniku, rīku, dokumentācijas, pārvaldības un apmācības kopums, ko izmanto kādas sistēmas izstrādei.", bet grāmatā (Guide, 2008) metode tiek aprakstīta kā sistēma, ko izmanto nozarē strādājošie, kas sastāv no tehnikām, procedūrām un likumiem.

Pastāv pieeja iedalīt datu noliktavas datu modeļa izstrādes metodes trīs grupās (Bimonte et al., 2021):

- Datu vadīta pieeja, kas balstīta uz datu avota shēmas analīzi;
- Prasību vadīta pieeja, kurā datu modeļa prasības tiek iegūtas no lietotājiem;
- Jauktā, kas kombinē prasību vadītu pieeju ar datu vadītu pieeju.

Tradicionālas datu noliktavas parasti tiek modelētas izmantojot divu principu metodes - Inmon (Inmon, 1996) un Kimball (Kimball & Ross, 1996).

Inmona metode ir balstīta uz entītiju saistību diagrammām, kas iegūtas no operacionālajām sistēmu datu bāzēm. Organizācijas dati tiek ielasīti datu noliktavā nezinot ieinteresēto personu informācijas prasības. Inmona metode tiek saukta par "datu vadītu" metodi. Inmon metode nosaka, ka datu noliktava un datuves ir fiziski nodalītas. Datu noliktavām un datuvēm var būt atšķirīgas struktūras un prasības. Lietotāji izmanto datus no

datuvēm un datuves tiek veidotas konkrētu servisu vajadzībām. Lietotāji datus no datuvēm iegūst izmantojot analītiskus rīkus. Lai izstrādātu datu noliktavu, Inmon nosaka veikt izstrādi iteratīvi pēc spirāles projekta pārvaldības principa. Lai izstrādātu datu noliktavas modeli, Inmon paredz šādus posmus:

- Veidot entītiju attiecību diagrammas – tas tiek darīts, lai izveidotu augstas abstrakcijas datu modeļus;
- Veidot vidējā līmeņa modeli – šajā posmā tiek veidots otrā līmeņa modelis. Otrā līmeņa modelī tiek meklēta detalizēta informācija par katru biznesa objektu, tiek noskaidrotas entītiju atslēgas, atribūti, atribūtu grupas un to saistības;
- Veidot fizisko modeli – augstākā līmeņa modelis. Šis modelis tiek veidots par pamatu izmantojot otrā līmeņa modeli.

Kimbal metode datu noliktavas modelēšanā balstās uz dimensiju modelēšanu par pamatu izmantojot kritiskos biznesa procesus, biznesa prasības un analizējot biežāk uzdotos jautājumus par organizācijas attīstību. Metode paredz ieinteresēto personu iesaisti datu noliktavas izstrādes sākotnējos posmos, iegūstot datu noliktavas prasības. Kimlall metodi sauc par "prasību vadītu" metodi. Kimball uzskata, ka datu noliktavai jāsastāv no vairākām datuvēm. Kimball metode nosaka veidot multidimensionālu datu modeli, kas galvenokārt sastāv no divām komponentēm – faktu un dimensiju tabulām.

Kimball datu noliktavas multidimensonālo modeli ir jāizstrādā četros posmos:

- Biznesa procesu identificēšana – datu noliktavai jāatbalsta organizācijas biznesa procesi un jāsniedz noderīga informācija lietotājiem;
- Datu granulitātes noteikšana – noteikt, kādā granulitātē veidot datu modeli un saglabāt datus. Kimball ieteikums ir vienmēr veidot datu modeli pēc iespējas detalizētāk. Pat, ja sākotnēji nav nepieciešami detalizēti dati, tos ir iespējams agregēt nepieciešamajā detalizācijā, līdz ar to, parādoties jaunām prasībām, dati būs pieejami arī zemākā detalizācijā;
- Dimensiju identificēšana - šajā posmā ir jānosaka dimensijas, kas aprakstīs faktu tabulas, respektīvi, atribūtus pēc kuriem var filtrēt un grupēt faktus. Dimensiju atribūti var tikt noskaidroti aprakstot biznesa procesus, uzdodot jautājumus – Kas? Kur? Kad? Kāpēc? Kā?
- Faktu identificēšana – faktu tabulās tiek galvenokārt saglabātas skaitliskās vērtības. Faktu tabulas sniedz atbildes uz tādiem biznesa jautājumiem, kā piemēram – Kāda ir vidējā peļņa katrai produktu grupai?

Inmon un Kimball metodes ir pazīstamākās un lietotākās metodes datu noliktavas modeļa izstrādē (Yessad & Labiod, 2016). Turpmākajās apakšnodaļās tiks aplūkotas jaunākās metodes datu noliktavas modeļa izstrādē.

1.3.1 Datu noliktavas modeļa izstrādes metode balstoties uz rakursa tabulām

Prasību vadīta datu noliktavas izstrādes metode (Bimonte et al., 2021) par pamatu prasību analīzei un datu noliktavas datu modeļa izveidei izmanto rakursa tabulas. Šī metode balstās uz četriem iteratīviem soļiem.

Mērķu modelēšana – izmantojot intervēšanas metodi, izveido datu noliktavas mērķu modeli. Šo soli ir paredzēts iteratīvi atkārtot. Mērķus ir paredzēts uzklausīt no visām ieinteresētajām personām. Lai identificētu mērķus, tiek veiktas intervijas brīvā formātā, lai noskaidrotu nepieciešamo informāciju analīzes vajadzībām. Kad ir iegūtas prasības no visām ieinteresētajām pusēm, tiek paredzēts veikt prasību specificēšanu, izmantojot mērķu vadītu pieeju prasību analīzē, kas aprakstīta rakstā (Giorgini et al., 2008). Iegūtais mērķu modelis tiek apspriests un apstiprināts ar ieinteresētajām personām pēc iepriekš definētas jautājumu kopas par tipiskām problēmām, kas saistītas ar datu noliktavas izstrādi un prasību analīzi.

Agregāciju tabulu modelēšana – tiek pielietotas speciālas agregācijas un veiktas daļēji strukturētas intervijas, lai apstrādātu mērķu modeli. Šo soli paredzēts veikt iteratīvi, lai apstrādātu iegūto mērķu modeli. Rakursa tabulas tiek veidotas, jo tiek apgalvots, ka ieinteresētajām personām mēdz pietrūkt pieredzes, lai izteiktu prasības multidimensionāla modeļa kontekstā vai, dažos gadījumos, nav pieredzes prasību definēšanā. Rakursa tabulas palīdz ieinteresētajām personām definēt vajadzības. Tiek noteikts iegūt no ieinteresētajām personām detalizētas prasības par katru mērķi vai apakšmērķi, veidojot e-rakursa tabulas. E-rakursa tabulas ir uzlabotas parastās rakursa tabulas (rakursa tabulas piemērs redzams tabula 1.1.), bet tās atšķiras ar grafisku attēlojumu, lai atainotu neregulāras hierarhijas. Pēc e-rakursa tabulu iegūšanas datu noliktavas eksperti, izmantojot daļēji strukturētas intervijas, precizē neskaidrības un iespējamās kļūdas.

Rakursa tabulas iedala:

- Stingras (*strict*) hierarhijas – pastāv tikai "viens pret daudz" attiecības;
- Vājas (*non-strict*) hierarhijas – var pastāvēt "daudz pret daudz" attiecības;
- Tukšas (*non-onto*) hierarhijas – var pastāvēt pirmā līmeņa elements bez nevienas instances;
- Trūkstošās (*non-covering*) hierarhijas – kāds no elementiem var trūkt starp hierarhijas līmeņiem;

- "Daudz pret daudz" faktu un dimensiju attiecības (*many-to-many fact-dimension*) – hierarhijā pastāv fakti, kas saistīti ar vairākām dimensijām;
- Vairāku granulitātes faktu (*multigranular fact*) hierarhijas – kāds no fakta notikumiem var būt nesaistīts ar visām dimensijām;

tabula 1.1 var redzēt stingras hierarhijas rakursa tabulas piemēru, kur pastāv "viens pret daudz" relācija starp lokāciju un laiku, bet nav trūkstošu elementu.

tabula 1.1

Lokācija		**Laiks**		**avg(ieņēmumi)**
Pilsēta	**Novads**	**Gads**	**Mēnesis**	
Rīga	Rīgas novads	2022	Maijs	15000
Ogre	Ogres novads	2023	Janvāris	12000
Ķegums	Ogres novads	2022	Marts	10000

Multidimensionāla modelēšana – prasību specifikācijas posmā iegūtā informācija tiek transformēta datu noliktavas multidimensionālā modelī. Katra e-rakursa tabula automātiskā veidā tiek transformēta multidimensionālā modelī atbilstoši ICSOLAP (Bimonte, et al., 2013) standartam. Atbilstoši metodei (Giorgini et al., 2008), shēmas, kas saistītas ar kopīgu apakš mērķi, tiek apvienotas. Galvenais princips, kā tiek veidots multidimensionālais modelis, ir e-rakursa tabulu kolonu nosaukumu transformācija multidimesionāla modeļa elementos – hierarhijas un dimensijās. Pēc datu noliktavas shēmas iegūšanas, datu noliktavas eksperti pārskata datu modeli un pārliecinās, ka visi atribūti ir pieejami avota sistēmās.

Datu noliktavas implementācija – šajā solī tiek veidots datu noliktavas prototips un izstrādāta datu noliktava. Sākotnēji tiek paredzēts, ka datu noliktavas eksperti ieviesīs datu noliktavas prototipu, kas tiks prezentēts ieinteresētajām personām.

Ierobežojumi metodes (Bimonte et al., 2021) darbībā:

- Pastāv ierobežojumi gadījumos, kad ieinteresētajai personai ir jāapraksta prasības, kas sastāv no sarežģītām agregācijām un atribūtiem, kas tiek aprēķināti, izmantojot e-rakursa tabulas;
- Mērķu modelēšanas solī tiek paredzētas intervijās ar ieinteresētajām personām, pastāv iepriekš definētas jautājumu kopas par konkrētām problēmām, tomēr pastāv bažas, ka šīs jautājumu kopas nespēs novērst visas neskaidrības, kas ir saistītas ar prasību analīzes posmu;

- Mērķu modelēšanas solī nav ļoti stingri noteikts kā atšķirt mērķi no apkašmērķa, kas var negatīvi ietekmēt datu noliktavas multidimensionālā modeļa faktu tabulu detalizāciju.

1.3.2 NoSQL datu noliktavu modeļi

Daudzas kompānijas, pieaugošo datu apjomu dēļ, ir mainījušas savas datu bāzes pārvaldības sistēmas veidu uz NoSQL (Bouaziz, et al., 2019). NoSQL datubāzes ir sadalīta datu bāzes sistēma, kas ļauj saglabāt nestrukturētus datus un to ir iespējams mērogot (*scale*) vertikāli. NoSQL datu bāzes nenodrošina standarta relāciju datu bāžu īpašības – atomitāti, konsistenci, izolāciju un noturību (Baralis et al., 2017). NoSQL datubāzes ir plaši izmantotas tādās lielās kompānijās kā Google, Amazon u.c. Lielākoties, izmantojot NoSQL datubāzes, nav nepieciešams definēt stingru un nemainīgu datubāzes struktūru. Autori (Akid et al., 2022) NoSQL datu noliktavas modeļus iedala trīs grupās – *dokumentu orientētās*, *kolonnu bāzētās*, *grafu bāzētās*.

Dokumentu orientētas. Šīs metodes var iedalīt pie datu vadītas datu noliktavas modelēšanas grupas.

Autori (Chevalier et al., 2015) definē noteikumus un saistības, kā multidimensionālu modeli transformēt dokumentu orientētā modelī. Katrs multidimensionāls fakts tiek pārveidots par dokumenta kolekciju (*collection*), kura satur mērījumus. Katra dimensija tiek pārveidota dokumenta kolekciju, kura satur atribūtus.

Metodē (ABDELHEDI et al., 2022) tiek noteikts, kā relāciju datu bāzi pārveidot NoSQL datu noliktavā – multidimensionālā modelī. Lai veiktu pārveidošanu, tiek veikts modelēšanas un datu pārvaldības solis. Modelēšanas solis sastāv no apakš soļiem:

1. Relāciju datu bāzes modelēšana – relāciju datubāzes datu noliktava var sastāvēt no vairākām shēmām, tabulām, atribūtiem un agregācijām. Par pastāvošo relāciju datu modeli ir jāizveido metamodelis;
2. Dokumentu orientēta NoSQL modelēšana – NoSQL datubāze sastāv no klasēm. Katrā klasē ietilpst objekti, kas ir identificēti. Objekts sastāv no pāriem - *atribūts* un tā *vērtība*. Dokumentu orientētai NoSQL datubāzei tiek veidots metamodelis.

Datu pārvaldības solis sastāv no:

1. Datu noliktavas modeļu veidošanas – lai veidotu šos medeļus tiek noteikti divi noteikumi. Pirmais noteikums liek katru tabulu no relāciju datu bāzes transformēt kā klasi NoSQL modelī. Lai izvairītos no dublikātiem, ir jāiekļauj klašu nosaukumā orģinālās datubāzes un shēmas nosaukums. Otrais noteikums ir, ka

katru rindiņu no relāciju datu bāzes ir jātransformē kā ierakstu atbilstošajā klasē. Katrs ieraksts sastāv no pāra - *atribūta* un tā *vērtības*.

2. Objektu un modeļu saistību veidošana – relāciju datubāze sastāv no ārējām atslēgām, NoSQL datubāzēs saistība starp objektiem tiek veidota ar referenču palīdzību. Referenču veidošana notiek klases ierakstu līmenī.
3. Klašu apvienošanas modulis – līdzīgu klašu apvienošana. Lai iegūtu informāciju par līdzīgajām klasēm, tiek izmantota ontoloģija, kas ir ģenerēta no relāciju datu bāzes.

Metodē (Bouaziz, et al., 2019) tiek aprakstīts, kā no dokumenta orientētas datu bāzes var modelēt multidimensionālo shēmu, izmantojot NoSQL datubāzi. Tas dod iespēju izstrādāt datu noliktavu atbilstoši multidimensionālam projektēšanas principam, izmantojot NoSQL datubāzi. Šī metode liek uzvaru uz NoSQL shēmas ieguvi grafa formātā, ar mērķi izveidot datu noliktavas multidimensionālo modeli. Tie paredzēta iespēja projektētājam noteikt datu noliktavas konceptus manuāli. Metodes darbība tiek iedalīta piecos soļos:

1. NoSQL dokumentu orientētas datubāzes izvēle;
2. Struktūras ieguve – NoSQL datubāzēm nepastāv vienota struktūra. Ir nepieciešams iegūt katra NoSQL datubāzes ieraksta struktūru no kā vēlāk var iegūt kopējo datubāzes struktūru;
3. Kopējās struktūras identificēšana – šajā solī tiek identificēts grafs par pamatu izmantojot struktūru, kas iegūta no NoSQL datubāzes vērtībām. Šis grafs palīdzēs nākamajos soļos projektētājam atrast multidimensionālos elementus;
4. Multidimensionālo elementu noteikšana – faktu, mērījumu, dimensiju un hierarhiju. Šo darbību veic datu noliktavas projektētājs manuāli;
5. Datu noliktavas modelēšana – tiek projektēts datu noliktavas multidimensionālais modelis. Šajā solī tiek savienoti iegūtie multidimensionālie elementi.

Kolonu bāzētās. NoSQL datu noliktavas multidimensionāla modeļa ieguvi apraksta autori (Ferreira, 2022), nosakot par pamatu izmantot četrus soļus:

1. Zvaigznes shēmas transformācijas solis, kas balstās uz šādiem principiem:
    a. Izstrādāt vai pilnveidot esošo modeli, atbilstoši multidimensonāla modeļa standartam (Kimball & Ross, 2011);
    b. Denormalizēt visu zvaigznes shēmas modeli, transformējot to vienā tabulā (Chevalier et al., 2015).
2. Nodalījumu izvēles solis, kas balstās uz šādiem principiem:
    a. Identificēt dimensijas, kuras ir visvairāk izmantotas filtros un vaicājumos, piemēram, laiks, lokācija un organizācijas dati;

b. Identificēt filtrus pēc Pareto principa, kas aprakstīts (Zhu & Xiang, 2016);
c. Identificēt dimensijas, kurās ir vairāk atšķirīgas vērtības saistībā ar faktiem, piemēram, lokācijas dimensijā - pilsētas;
d. Samazināt partīciju kolonu skaitu līdz piecām (Chebotko et al., 2015);
e. Neiekļaut partīcijās kolonas, kuras nav izmantotas filtros un vaicājumos.

3. Grupēšanas izvēles solis, kas balstās uz šādiem principiem:
   a. Identificēt filtrus, kuri ir bieži pielietoti, bet nav nepieciešami;
   b. Identificēt kārtošanas funkcijas lietojumus;
   c. Identificēt grupēšanas un detalizācijas (*drill-down*) funkcijas;
   d. Noteikt dimensijām granulitāti, sakārtot tās secībā no mazāk granulārās uz vairāk granulāro;
4. Hierarhiju izvēles un veiktspējas solis, kas balstās uz šādiem principiem:
   a. Vaicājumos jāparādās kolonnām, kas ir izvēlētas kā partīciju kolonnas un grupēšanas kolonnas;
   b. Hierarhiju gadījumā vaicājumos jāizmanto detalizētākā kolonna.

Lai sniegtu biznesa vajadzībām nepieciešamo informāciju, tiek noteikts veidot vairāku veidu skatus – pastāvošā (orģinālā) zvaigznes modeļa, biežāk lietoto filtru, biežāk grupēto datu.

1.3.3 Grafu bāzēts datu noliktavas modelis

Autori (Sellami et al., 2020) ir aprakstījuši, kā izstrādāt datu noliktavu par pamatu izmantojot grafu bāzētu datubāzes pārvaldības sistēmu. Autori apgalvo, ka pēdējos gados tīmekļa tehnoloģijās izstrādātās programmatūras sāk vairāk izmantot grafu datubāzes informācijas saglabāšanai. Tādas sociālās platformas, kā Facebook, Twitter un LinkedIn, aizvien plašāk izmanto grafu orientētās datubāzes pārvaldības sistēmas. Šo metodi var iedalīt pie datu vadītas datu noliktavas modelēšanas metodes.

Autori (Sellami et al., 2020) piedāvā veidot grafu bāzētu datu noliktavu, izmantojot loģisko modeli - "grafu dimensionālo modeli" (GDM – "*Graph Dimensional Model*"). Grafu orientētas datubāzes pārvaldības sistēmas sastāv no virsotnēm (*node*), šķautnēm (*edges*) un atribūtiem (*properties*). Katrai virsotnei ir nosaukums (*label*) un ir atribūti. Šķautnes saista virsotnes, šķautnēm var būt savi atribūti. Tiek piedāvāts, kā veikt "denormalizētu transformāciju", šī transformācija nosaka saistību starp NoSQL modeļa elementiem un multidimensionāla modeļa elementiem. Transformācija nenosaka, kā veidot hierarhijas. Aprakstītā transformācija nosaka, kā katru faktu un dimensiju pārveidot virsotnēs, sekojot šādiem principiem:

1. Faktu un mērījumu transformācija – katrs fakts tiek pārveidots par virsotni. Virsotnei tiek pievienots priedēklis “fact” un nosaukums – fakta nosaukums no multidimensionālā modeļa. Katrs fakta mērījums tiek pievienots kā virsotnes atribūts;
2. Dimensiju un parametru transformācija – katra dimensija tiek pārveidota kā virsotne. Virsotnei tiek pievienots priedēklis “dimension” un dimensijas nosaukums no multidimensionālā modeļa. Dimensijas atslēgas un kolonnas tiek pievienotas kā virsotnes atribūti;
3. Saites starp faktu un dimensiju transformācijas – saites starp faktu un dimensiju tiek pārveidotas šķautnēs. Tiek pievienots priedēklis “relation fact - dimension”.

Lai būtu iespējams izsekot objektu saitēm starp orģinālo multidimensionāla modeļa objektu un izveidoto grafa objektu, tiek veidotas speciālas atbilstības tabulas, kurās tiek saglabāta informācija par multidimensionāla modeļa objektu un atbilstošo grafa objektu.

1.4 Secinājumi

Šajā nodaļā tika aplūkots datu noliktavas jēdziens, tradicionālas datu noliktavas arhitektūra, jaunākās pieejas datu noliktavu arhitektūrā, kā arī datu noliktavas modelēšanas metodes. Tradicionālas datu noliktavas arhitektūra tipiski tiek iedalīta pēc to līmeņu skaita. Attīstoties jaunām tehnoloģijām un biznesa vajadzībām, tiek pielietota atšķirīgāka datu noliktavas arhitektūra, piemēram, izmantojot lielo datu tehnoloģijas un veidojot “loģisku datu noliktavu”, fiziski datus neielasot uz datu noliktavas servera, bet veidojot datu noliktavas datu bāžu skatus. Pastāv nostāja, ka tradicionālas datu noliktavas ar tradicionāliem ETL rīkiem un datu glabātuvēm vairs efektīvi netiek galā ar dažādajiem datu tipiem un lieliem datu apjomiem (Hai, 2021), (Zagan & Danubianu, 2020) un (Cherradi & Haddadi, 2022).

Nodaļā arī tika aplūkotas datu noliktavas modelēšanas metodes. Datu noliktavas modelēšanas metodes var iedalīt trīs grupās – datu vadītās, prasību vadītās un kombinētās. Lai arī, attīstītos tādām jaunām tehnoloģijām kā NoSQL un grafu datu bāzes, kas piedāvā noteikumu kopu, kā nestrukturētus datus transformēt datu noliktavas modelī, joprojām ir svarīgi balstīt datu noliktavas modeļa izstrādi uz ieinteresēto personu prasībām, jo pastāv nostāja (Benkhaled & Berrabah, 2019), ka datu noliktavas izstrāde cieš neveiksmi, ja tā netiek izstrādāta atbilstoši ieinteresēto personu vajadzībām – prasībām, tāpēc nākamajā nodaļā tiek aplūkotas datu noliktavas prasību vākšanas metodes un informācijas prasību jēdziens.

2 PRASĪBU IEGŪŠANA TRADICIONĀLĀM DATU NOLIKTAVĀM

Šajā nodaļā tiek apskatītas tradicionālu datu noliktavu izstrādes prasību specifika, datu noliktavas prasību iegūšanas metodes, aplūkots informācijas prasību jēdziens un prasību formalizēšanas iespējas datu noliktavas kontekstā. Šajā nodaļā tiek aprakstīa metode datu noliktavas kandidātshēmu ģenerācijai. Šī metode nodrošina iespēju, ģenerēt datu noliktavas kandidātshēmas daļēji automātiskā veidā, izmantojot datu noliktavas formālo prasību metamodeli. Tiek aprakstīts rīks iReq, kas praktiski ievieš metodes darbību. Metodes un rīka darbība ir publicēta rakstā (Kozmina et al., 2017). Šajā nodaļā aprakstītais rīks un metodes darbības paplašinājums ir turpinājums autora maģistra darbam.

2.1 Tradicionālas datu noliktavas izstrādes prasības

Pastāv būtiska atšķirība starp tradicionālu prasību inženieriju priekš operacionālām informācijas sistēmām un datu noliktavām. Datu noliktavas izstrādes prasībām tiek pievērsta īpaša uzmanība (Prakash & Prakash, 2019). Tradicionālas datu noliktavas izstrādes prasības nosaka (Bruckner, et al., 2001):

- Datu noliktavas darbības principus;
- Pieejamo informāciju;
- Datu transformācijas loģiku;
- Agregācijas loģiku;
- Aprēķinu loģiku.

Autori (Bruckner, et al., 2001) nosaka, ka galvenais avots datu noliktavas prasībām ir organizācijas biznesa prasības, kas nodrošina organizācijas augstākā līmeņa mērķu sasniegšanu, izmantojot datu noliktavu funkcionalitāti. Kā nākamais līmenis pēc biznesa prasībām, ir lietotāja prasības, kas ir pakārtotas biznesa prasībām, bet šajā līmenī organizācijas darbinieki var iesaistīties prasību definēšanā, lai sasniegtu mērķus, izpildītu uzticētos uzdevumus. No lietotāja prasībām tiek veidotas datu noliktavas prasības, kuras var iedalīt trīs lielās grupās:

- **Funkcionālās prasības**, apraksta datu noliktavas darbību, kas var tikt definēta kā servisi, uzdevumi vai kā nodrošināmā funkcionalitāte;
- **Informācijas prasības**, kas galvenokārt nosaka, kādi dati nepieciešami datu noliktavā – nepieciešamos atribūtus, mērījumu aprēķinu loģiku, datu avotus, informācijas kvalitātes kritērijus, datu granulitāti, drošības un pieejas prasības saistībā ar informāciju datu noliktavā;

- **Citas prasības**, kas nosaka saskarnes starp sistēmām, informācijas sistēmas platformas prasības.

2.2 Informācijas prasības

Nodaļā 2.1. tika aplūkotas tradicionālas datu noliktavas prasības. Kā viens no datu noliktavas prasību veidiem, ir informācijas prasības, kas definē nepieciešamo informāciju organizācijā. Neskatoties uz to, ka datu noliktavas kopumā organizācijām piedāvā virkni ieguvumu, datu noliktavas izstrāde var ciest neveiksmi, ja tā nesniedz ieinteresētajām personām nepieciešamo informāciju (Benkhaled & Berrabah, 2019). Piegādājot nepieciešamo informāciju, tiek palīdzēts organizācijām sasniegt izvirzītos mērķus un šī informācija var uzlabot ikdienas procesus organizācijā. Datu noliktavas informācijas prasības nosaka šādus aspektus par datu noliktavu (Schiefer et al., 2002):

- Datu noliktavu datu avoti:
    - Organizācijas iekšējie datu avoti;
    - Organizācijas ārējie datu avoti;
    - Informācijas/Datu piegādātāju avoti;
    - Datu kartēšanas noteikumi.
- Atskaišu un vaicājumu prasības:
    - Atskaišu prasības;
    - Datizraces risinājumu prasības;
    - Datu vizualizācija prasības;
    - Datu noliktavas lietotāju prasības.
- OLAP prasības
    - Organizācijas darbības rādītāji un mērījumi;
    - Analīzes dimensijas;
    - Laika dimensijas;
    - Datu kubi.

Datu noliktavas izstrādes izmaksas parasti ir augstas, kas saistīts ar augsti kvalificētu darbaspēka nepieciešamību. Kā vienu no iemesliem ar ko var saistīt datu noliktavas neveiksmes, ir organizācijas biznesa interesēm un vajadzībām neatbilstošs datu noliktavas datu modelis. Nākamajā nodaļā 2.3. tiek aplūkotas datu noliktavas prasību vākšanas metodes.

2.3 Prasību vākšanas metodes tradicionālām datu noliktavām

Šajā nodaļā tiek apskatītas pastāvošās metodes informācijas prasību iegūšanā tradicionālām datu noliktavām.

Metodē "Deriving the Conceptual Model of a Data Warehouse from Information Requirements" (Kozmina et al., 2013) galveno uzsvaru liek uz KPI formalizēšanu. Tiek noteikts, ka organizācijām, lai tās veiksmīgi spētu darboties, būtu jāveic darbības rezultātu uzraudzīšana (*monitoring*). Datu noliktavas tiek izmantotas kā rīks, kas nodrošina KPI uzraudzīšanas funkcionalitāti. Datu noliktava nodrošina šādas īpašības, lai analizētu organizācijas veiktspēju:

- Iespēja integrēt vairākus datu noliktavas datu avotus;
- Iespēja saglabāt datu noliktavā vēsturiskos datus;
- Datu noliktavai specifisks shēmas modelis, kas ir pielāgots tieši analīzes un mērījumu nodrošināšanai.

Metodē (Kozmina et al., 2013) tiek piedāvāts KPI formalizēt, jo to aprakstošajai valodai ir viena veida struktūra, tāpēc tos ir viegli formalizēt. Rakstā (Niedritis et al., 2011), kas ir šīs metodes vecākā versija, ir apskatīti 330 dažādi KPI piemēri kā tos formalizēt. Metodē tiek izmantots vienots formalizēto prasību repozitorijs, kur tiek apkopotas un saglabātas formalizētās prasības. Izmantojot metodē aprakstītu algoritmu, no šīm formālajām prasībām tiek iegūta datu noliktavas kandidātshēma vai vairākas kandidātshēmas. Šīs shēmas tiek apspriestas ar datu noliktavas ieinteresētajām personām, noskaidrojot nepieciešamos uzlabojumus un papildinājumus. Pēc šī metodes posma, kandidātshēmas elementi tiek pārveidoti par datu noliktavas konceptuālo modeli. Kandidātshēmas elementi tiek analizēti pēc atskaišu nepieciešamības un prioritātēm.

Metodes pirmais posms nosaka prasību formalizēšanu. Datu noliktavas informācijas prasības formalizēšanas piemērs - "attiecība procentos starp IT administratīvajām izmaksām un 17 kopējām izmaksām", ja tiek pārveidota šī prasība formālā datu noliktavas prasībā, tiek iegūts - "(sum(expense) where expense type = 'IT') / (sum (expense))". Metode paredz, ka formālās datu noliktavas prasības izteiksmes veidos datu noliktavas izstrādātāji, standarta gadījumā, tie būtu datu noliktavas sistēmas analītiķi. Izteiksmes struktūra un to sastāvdaļas ir nodefinētas UML[2] notācijā. UML metamodeli var apskatīt 1. pielikumā, kas apraksta likumus, pēc kuriem var sastādīt informācijasprasību formālās izteiksmes.

Metamodelis sastāv no vairākām UML klasēm. Metamodeļa galvenā klase ir "Prasība" ("*Requirement"*), kas sastāv no klasēm "Vienkārša prasība" ("*Simple Requirement*") un "Salikta prasība" ("*Complex Requirement*"). Klase "Vienkārša prasība" var sastāvēt no apakšklasēm "*Izteiksme*" ("*Expresion*"), "Objets" ("*Object*"), "Konstante" ("*Constant*"), "Kvantitatīvie dati" ("*Quantifying data*") un "Kvalitatīvie dati" ("*Qualifying data*").

[2] https://www.uml.org

Metamodelis paredz, ka prasības savā starpā var būt savienotas ar aritmētiskām operācijām. UML klasi "Salikta prasība" izvēlās gadījumos, kad prasība sastāv no vairākām vienkāršām prasībām. Divas vienkāršas prasības var būt savienotas ar aritmētiskajām operācijām. Piemēram, informācijas prasība "Starpība starp akadēmisko personālu un studentu skaitu". Šī informācijas prasība sastāv no divām vienkāršām prasībām - pirmā "darbinieks, kurš ir akadēmiskais personāls", otrā informācijas prasība - "studentu skaits". Tā kā informācijas prasībā nosaka "attiecību starp personālu un studentu skaitu", šīs vienkāršās prasības ir jādala ("/") sava starpā. Formālā prasība, atbilstoši metodē aprakstītajam metamodelim, izskatīsies: "(count(darbinieks) where amats = "akadēmiskais personāls")/(count(students))". Vienu prasību var formalizēt dažādi, metodē nav definētu stingru noteikumu, galvenais princips ir, ka formalizētajai prasībai jāatbilst metamodelim. Šo pašu informācijas prasību var formalizēt "(count(akadēmiskais personāls))/(count(students))". Šādi formalizējot informācijas prasību, vairs netiek lietots "*where*" nosacījums, jo tiek pieņemts, ka datu noliktavas datu avotā ir pieejama informācija un pastāv iespēja saskaitīt akadēmisko personālu.

Metodē "Monitoring Strategic Goals in Data Warehouses with Awareness Requirements" (Souza et al., 2012) tiek piedāvāts metamodelis "Awareness Requirements" (*AwReqs*), kurš nosaka saglabāt lietotāju prasības, lai varētu izsekot informācijas sistēmas stratēģiskos mērķus (*startegic-goals*). Informācijas prasības tiek saistītas ar datu noliktavas ieinteresēto personu mērķiem. Mērķi un informācijas prasības nosaka, ko ir iespējams sasniegt ar datu noliktavas palīdzību, vai arī iegūt mērījumu, lai noteiktu, vai lietotāja mērķis ir sasniegts. Organizācijas darbinieka lomai "Pārdošanas speciālists" mērķis ir, piemēram, "Piesaistīt gada laikā 50 jaunus klientus". Metamodeļa galvenais uzdevums ir modelēt datu noliktavas ieinteresēto personu mērķus. No šīs informācijas tiek iegūtas datu noliktavas izstrādes prasības. Metamodeļa "AwReqs" galvenā klase ir "Mērķis" ("*Goal"*), kas var tikt iedalīta trīs apakšklasēs - stratēģiskais (*strategic*), lēmumu (*decision*), informācijas (*infomation goal*) mērķis. Metamodelis paredz mērķus iedalīt:

- Stratēģiskajos, kas nosaka galvenos biznesa procesa mērķus, kā piemēram, palielināt pārdošanas apjomu vai palielināt klientu skaitu, metamodeļa klase "Stratēģiskie" ("*Startegic*");
- Lēmumu, kas nosaka, kādas darbības jāveic, lai sasniegtu stratēģiskos mērķus, piemēram, atvērt jaunu veikalu, metamodeļa klase "Lēmums" ("*Decision*");
- Informācijas mērķis, kas nosaka informāciju, kas nepieciešama, lai sasniegtu mērķi, piemēram, analizēt klientu pirkumus pa mēnešiem, metamodeļa klase "Informācija" ("*Information*").

Definējot datu noliktavas ieinteresēto personu mērķus, tiek noskaidroti jauni multidimensionālā modeļa elementi - fakti un dimensijas. Metodes metamodelis "AwReqs" sastāv no klases "Uzdevums" ("*Task"*), kam var būt apakšklase "Prasība" ("*Requirement"*). Metamodeļa "AwReqs" klase "Uzdevumi" var tikt iegūti, analizējot datu noliktavas informācijas mērķus. Metamodeļa klase "Resurss" ("*Resource"*) tiek izmantota vēlākos analīzes posmos. Klase "Resurss" apzīmē biznesa procesu, kam pieder šis mērķis. Klase "Mērījums" ("*Measure"*) un klase "Konteksts" ("*Context"*) apraksta mērījuma analīzes procesu. Klase "IActor" apraksta lomu organizācijā, kurai nepieciešams šis mērķis, kā piemēram, loma "Pārdošanas vadītājs". Metodes aprakstā (Souza et al., 2012) aprakstīts piemērs, "Palielināt pārdošanu par 10% katru gadu". Lai organizācijā sasniegtu šo mērķi, lēmumu pieņēmēji izvirza mērķi "Izmaksāt labas prēmijas produktīvākajiem darbiniekiem". Nākamais solis nosaka, lai sasniegtu lēmumu pieņemšanas mērķi, ir nepieciešams datu noliktavā nodrošināt datus, kas ļauj izsekot mērķa progresam, tāpēc tiek definēts jauns informācijas mērķis "Analizēt pārdošanu". Šis mērķis tiek sadalīts informācijas prasībās "Nodrošināt informācijas attiecību par pārdotajiem produktiem un prēmijām" un "Nodrošināt informāciju par pārdotajiem produktiem kopumā.". Metode nosaka, ka tiek iegūts stratēģiskais mērķis "Palielināt pārdošanu par 10%", kas ir savienots ar darījuma procesu "Riteņu pārdošana". No stratēģiskā mērķa tiek iegūts lēmumu mērķis – "nodrošināt atbilstošas prēmijas", kas nosaka definēt jaunu informācijas mērķi – "analizēt pārdošanu". Šis informācijas mērķis tiek sadalīts sīkākās informācijas prasībās. Šīs informācijas prasības ir uzdevumi – "nodrošināt informāciju par pārdošanu saistībā ar paaugstinājumiem" un "nodrošināt informāciju par produktu pārdošanu kopumā". Informācijas prasības jeb uzdevumi tiek veidoti par nepieciešamajiem mērījumiem un dimensijām.

Datu noliktavas izstrādes metode "A framework for multidimensional design of data warehouses from ontologies" (Romero & Abelló, 2010) paredz izstrādāt datu noliktavu, kas atbalsta organizācijas lēmumu pieņemšanas procesus. Metodes autori piedāvā pieeju, kas nosaka analizēt organizācijas pieejamos datu noliktavas datu avotus, lai veidotu multidimensonālo modeli, ņemot vērā lietotāju izvirzītās prasības. Metodi sastāv no trīs posmiem:

1. Metodes pirmais posms nosaka veikt datu avotu analīzi, šajā posmā netiek analizētas datu noliktavas lietotāju prasības;
2. Otrais solis paredz izmantot iegūto informāciju, lai atvieglotu prasību iegūšanas procesu. Šajā posmā tiek noskaidrotas lietotāju prasības;
3. Trešais metodes posms paredz automātisku mulitimesonālās shēmas ģenerāciju par pamatu izmantojot iegūtās prasības.

Metodes mērķis ir multidimesonālā modeļa elementus iegūt automātiskā veidā. Lai to panāktu, tiek iegūta ontoloģija no organizācijas datu noliktavas datu avotiem, ontoloģija var tikt automatizēti ģenerēta no ER modeļa. Metodes pieeja ir no iegūtajām ontoloģijām un ontoloģiju vārdnīcām iegūt potenciālās multidimesonālā modeļa dimensijas un faktus.

Šo prasību iegūšanas metodi var iedalīt pie daļēji lietotāju prasību orientētas un daļēji datu vadītas datu noliktavas izstrādes metodes. Metodes autori nosaka, jo vairāk datu noliktavas izstrādes procesi tiek automatizēti, jo vairāk datu noliktavas prasības tiek pazaudētas izstrādes posmā. Metodes pamatā tiek lietota ontoloģija, kas iegūta no datu noliktavas avota sistēmām, šī ontoloģija apraksta avota sistēmās pieejamos datus. Potenciālo multidimesonālā modeļa dimensiju konceptu iegūšana no ontoloģijas notiek pēc principiem:

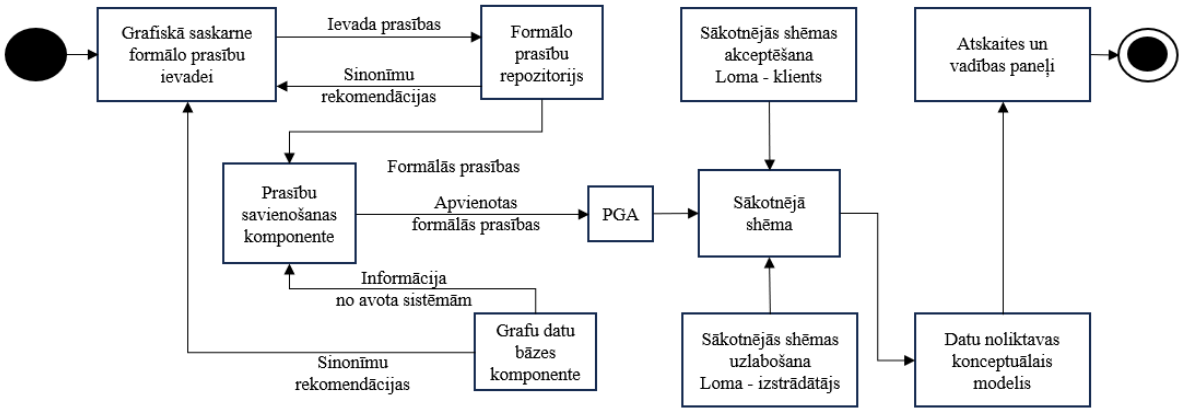
- Dimensiju koncepti veidojas no ontoloģijā esošajām klasēm un klašu atribūtiem un saitēm starp šīm klasēm. Katra klašu saite ontoloģijā veido jaunu multidimesonālā modeļa dimensijas konceptu. Ja divas klases ir saistītas ar divām saitēm, tās veido divus dažādus multidimesionālā modeļa dimensiju konceptus, jo katrai no saitēm var būt sava nozīme;
- Ja klasei {A, r}(A ir klase, bet r ir klases atribūts) pieder multidimesionālā modeļa dimensijas koncepts, B un {C, r1} ir dimensijas koncepts klasei A, tad {C,r1} ir arī B dimensijas koncepts. Mērījumu un faktu noteikšana no ontoloģijas notiek pēc principa:
    - Faktu klases var noteikt pēc saitēm ar citām klasēm, faktu klases ir saistītas ar "viens pret daudz" saiti multidimesionālajā modelī;
    - Faktu mērījumu var noteikt, nosakot atribūta datu tipu. Klases atribūtiem kam ir skaitlisks datu tips, to var uzskatīt par mērījumiem multidimesionālā modelī.

Faktu un dimensiju konceptu apvienošana notiek, ja iegūtajā ontoloģijā starp fakta klasi un dimensijas konceptu pastāv relācija. Lai datu noliktavas lietotājam būtu pieejamas agregācijas funkcijas, ir nepieciešams atribūtus grupēt hierarhijās. Kad ir noskaidroti visi potenciālie dimensiju koncepti un mērījumi, nepieciešams iegūtos rezultātus apspriest ar gala lietotāju. Šīs darbības rezultātā var tikt iegūti ļoti daudzi dimensiju koncepti. Lai datu noliktavas lietotājam attēlotu tikai viņam piemērotākos, var pielietot filtrācijas formulas, kas var atfiltrēt atbilstošos dimensiju konceptus. Hierarhijas tiek noteiktas pēc saitēm starp klasēm ontoloģijā, ja starp divām klasēm pastāv "viens pret daudz" relācija, tad tā tiek dēvēta arī kā augšup vērsta (*roll-up*). Kad no ontoloģijas ir iegūti visi potenciālie dimensiju koncepti, fakti ar mērījumiem un izveidotas hierarhijas, tad tiek noskaidrotas datu noliktavas lietotāja prasības un iegūtie

potenciālie datu noliktavas modeļi. Kad tiek iegūti beidzamie datu noliktavas modeļi, datu noliktavas izstrāde var sākties.

2.4 Paplašinātā metode datu noliktavas prasību formalizēšana

Kā viena no metodēm, kā iegūt datu noliktavas prasības, ir iepriekšējā nodaļā aplūkotā metode, kas aprakstīta rakstos (Kozmina & Niedrite, 2014) un (Kozmina et al., 2013). Promocijas darba ietvaros šī metode ir paplašināta ar jaunām komponentēm "Grafiskā saskarne formālo prasību ievadei", "Formālo prasību savienošanas komponente" un "Grafu datubāzes komponente" (Kozmina et al., 2017). Paplašinātā metode (Kozmina et al., 2017) nosaka datu noliktavas formalizēt pēc noteikta metamodeļa. Šo metamodeli autors ir paplašinājis ar jaunām klasēm, skatīt nodaļu 2.5., iegūstot "paplašināto metamodeli" datu noliktavas informācijas prasību formalizēšanai. Orģinālajā rakstā (Kozmina & Niedrite, 2014) tiek aprakstīta metode, bet nav piedāvāts rīks, kā metodi pielietot. Šajā nodaļā tiek aprakstīta paplašināta metode un praktisks rīks, kā no prasībām ģenerēt konceptuālo datu noliktavas datu modeli. Att. 2.1. var redzēt metodes darbības attēlojumu. Pirmajā metodes solī "Grafiskā saskarne formālo prasību ievadei" datu noliktavas prasības tiek ievadītas izmantojot speciāli izstrādātu rīku "iReq". Tiek paredzēts, ka prasības tiek ievadītas formālo prasību repozitorijā, atbilstoši datu noliktavas formālo prasību "paplašinātajam metamodelim". Ievadītās prasības tiek saglabātas "Formālo prasību repozitorijā". Prasību ievadīšana formālo prasību repozitorijā notiek, izmantojot eksistējošos terminus un to sinonīmus, kas ir iegūti no datu noliktavas avota sistēmu pieejamiem atribūtiem un iepriekš ievadītajiem terminiem. Šī informācija tiek saglabāta "Grafu datu bāzes komponentē". Nākamajā solī, izmantojot speciālu datu noliktavas shēmas ģenerēšanas algoritmu – "Sākotnējās shēmas ģenerācijas algoritms (PGA)", tiek iegūta vienkāršota datu noliktavas shēma – "Sākotnējā shēma". PGA algoritms izmanto informāciju no formālo prasību repozitorija. Datu noliktavas izstrādātāji apstrādā iegūtās vienkāršotās shēmas – noņemot dublikātus, veidojot hierarhijas. Vēlāk apstrādātās shēmas tiek pārrunātas ar ieinteresētajām personām un atbilstošākā shēma tiek izvēlēta datu noliktavas konceptuālā modeļa veidošanai. Pēdējais solis paredz veidot atskaites, balstoties pēc prasību prioritātēm.



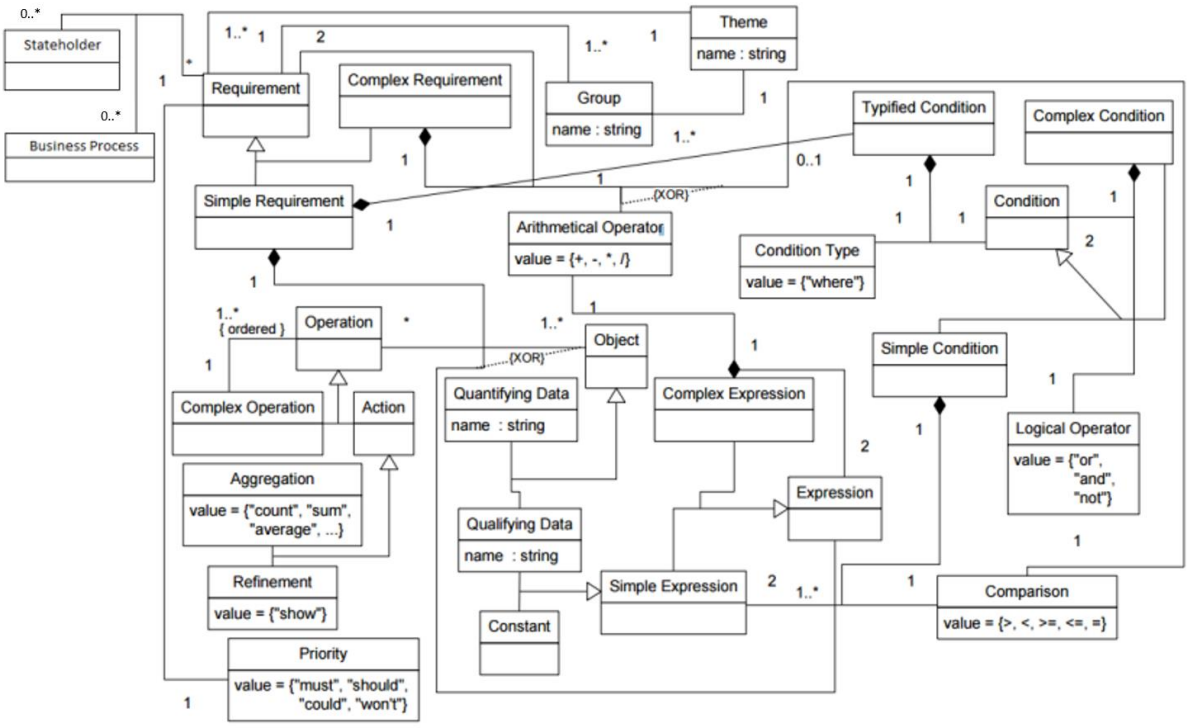
Att. 2.1. Metodes darbības diagramma

2.5 Paplašinātais metamodelis formalizētām datu noliktavas prasībām

Metamodelis, kas aprakstīts metodē (Kozmina & Niedrite, 2014) ir paredzēts, lai formalizētu datu noliktavas prasības. Šis metamodelis tiek paplašināts ar divām jaunām klasēm:

- **Ieinteresētā persona** (*Stakeholder*) ir paplašināta metamodeļa klase, kas saistīta ar klasi "Prasība" ("*Requirement*"). Šī klase ir pievienota pie klases "Prasība", lai būtu iespējams sasaistīt prasību ar ieinteresēto personu vai personām, kas ir ieinteresētas konkrētās prasības īstenošanā, piemēram, ja ar kādu prasību notiek izmaiņas vai datu noliktavas izstrādātāji paziņo par izmaiņām saistībā ar šo prasību, var ātri un viegli izsekot ieinteresētās personas. Grāmatā (Luján-Mora et al., 2006) tiek aprakstīta ieinteresēto personu nozīme informāciju sistēmu izstrādē, kas arī paredz atsevišķu testēšanas posmu, lai apmierinātu ieinteresēto personu izvirzītās prasības, šo testus sauc par akcepttestiem. Veicot akcepttestus, ir svarīgi zināt konkrētajās prasībās ieinteresētās personas. Pastāv metodes, kas nosaka saglabāt ieinteresētās personas katrai informācijas prasībai, piemēram, metode (Vicente-Chicote et al., 2007), kas paredz informācijas prasību formalizēšanu operacionālam informācijas sistēmām.
- **Darījuma process** (*Business process*) ir paplašināta metamodeļa klase, kas ir saistīta ar klasi "Prasība" ("*Requirement*"). Šī klase īpaši noderīga gadījumos, kad prasība apraksta kādu no KPI. Gadījumos, kad datu noliktava tiek izmantota KPI uzraudzībai, ir noderīgi, piemēram, nodrošināt meklēšanas funkcionalitāti pēc darījumu procesiem vai piešķirt tiesības apskatīt kāda konkrēta darījuma procesa mērījumus un atribūtus. Autora rekomendēto paplašināto metamodeli

var aplūkot att. 2.2. Pilnu orģinālā metamodeļa detalizētu aprakstu var atrast nodaļā 2.3



Att. 2.2. Autora paplašinātais metamodelis datu noliktavas prasību formalizēšanai

2.6 Autora piedāvātais rīks prasību vākšanas atbalstam tradicionālās datu noliktavās

Lai sniegtu iespēju formālās datu noliktavas prasības ievadīt apstrādei ar algoritmu "Sākotnējās shēmas ģenerācijas algoritms" - PGA algoritmu datu noliktavas prasības ir nepieciešams ievadīt formālo prasību repozitorijā (Kozmina et al., 2017). Promocijas darba autors ir izstrādājis speciālu rīku "iReq", kas sniedz iespēju lietotājam, izmantojot grafisku lietotāja saskarni, ievadīt datu noliktavas prasības, atbilstoši paplašinātajam metamodelim (Kozmina et al., 2017) un ģenerēt datu noliktavas kandidātshēmu, atbilstoši PGA.

iReq rīks ir izstrādāts, izmantojot tīmekļa tehnoloģijas - programmēšanas valodu PHP, ietvaru Laravel[3], JavaScript[4], CSS, jQuery [5] un ievaru Bootstrap[6]. Datu noliktavas formālās

[3] https://laravel.com
[4] https://www.javascript.com
[5] https://jquery.com
[6] https://getbootstrap.com

prasības tiek saglabātas datubāzes pārvaldības sistēmā MariaDB[7], bet informācija par datu noliktavas avota sistēmas atribūtiem un sinonīmiem grafu datubāzē Neo4j[8].

iReq rīks tika izstrādāts ar mērķi, lai nodrošinātu:

- Grafisku saskarni formālo prasību ievadīšanai, kas nodrošina formālo prasību ievadi atbilstoši formālo prasību metamodelim;
- Sastādīt un saglabāt formālās prasības;
- Ievadīt formālo prasības ar neierobežoti daudz apakšklasēm;
- Ģenerēt datu noliktavas kandidātshēmas modeļus.

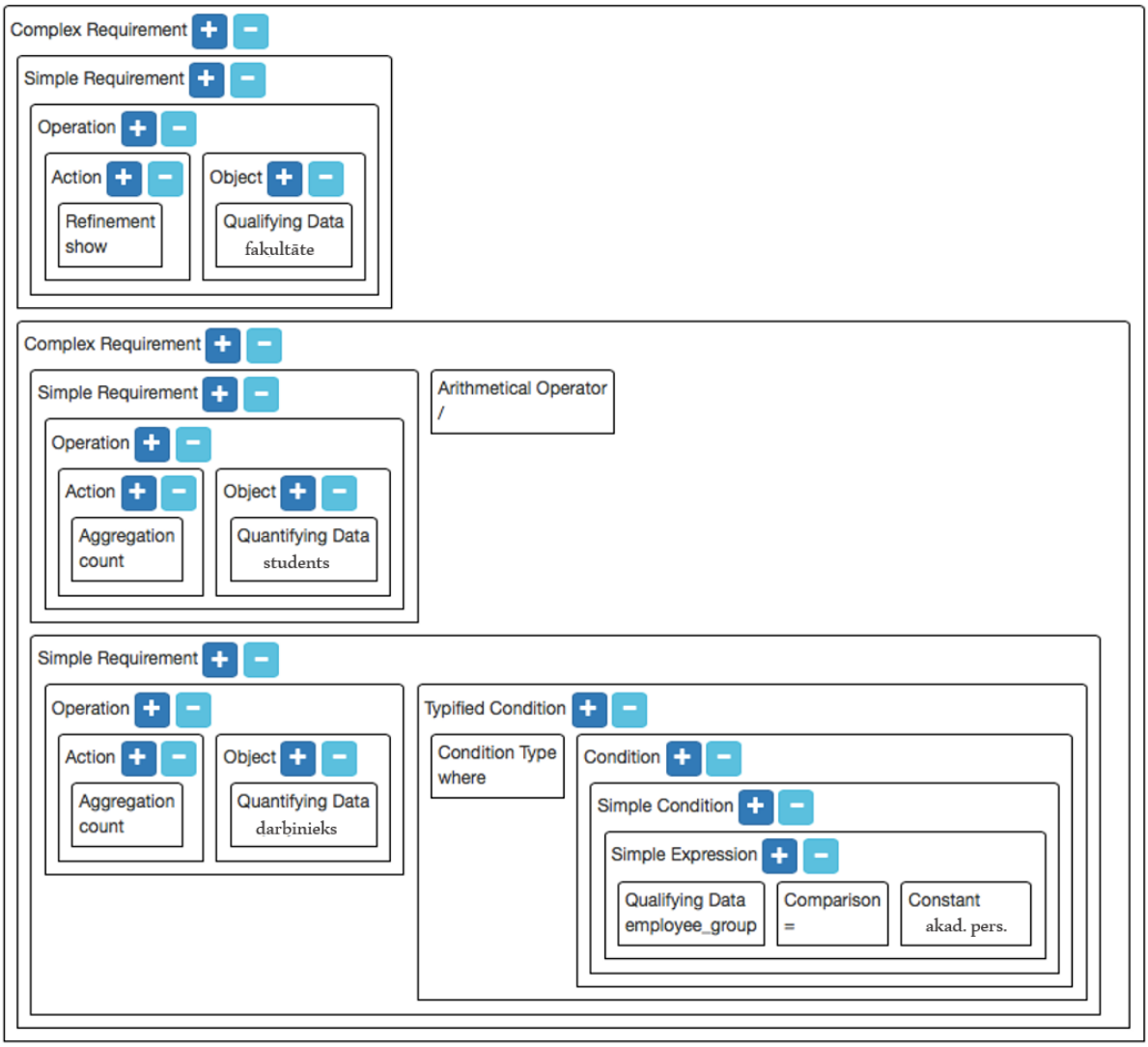
Rīks ir izstrādāts atbilstoši šīm prasībām. Pastāv ierobežojums, ka augot formālo prasību apakšklasēm, samazinās pārskatāmība uz datora ekrāna, šī situācija ir īpaši novērojama uz maziem ekrāniem.

2.6.1 Piemērs datu noliktavas prasības formalizēšanai

Kā piemērs tiek izvēlētas datu noliktavas prasības "*Attēlot informāciju par studentu un akadēmiskā personāla attiecību katrā fakultātē*", att. 2.3. un "*Attēlot informāciju par studentiem, kas ir no Rīgas un apmeklē lekcijas latviešu valodā*.", att. 2.4. Prasību "*Attēlot informāciju par studentu un akadēmiskā personāla attiecību katrā fakultātē*" var pārveidot formālā izteiksmē, atbilstoši metamodelim (Kozmina et al., 2017) - *show (fakultāte) count (students) / count (darbinieks) where employee_group = "akad. pers."*. "Dziļākie" klases elementi (att. 2.3) piemēram, "Bagātināt - attēlot" ("*Refinment – show*") reprezentē formālās izteiksmes elementus, bet to iekļaujošie – metamodeļa virsklases. Jebkuras prasības sākuma elements ir "Prasība" (*"Requirement"*). Klase "Prasība" var tikt saistīta ar klasi "Biznesa process" (*"Business process"*) un klasi "Ieinteresētā persona" (*"Stakeholder"*), skatīt piemērā "Studijas" un "Senāts". Prasība tiek iekļauta grupā vai vairākās grupās, metamodeļa klase "Grupa" (*"Group"*), attēlots aplūkotajā piemērā "Fakultātes plāns 2018". Klase "Tēma" ("Theme") ir augstāka līmeņa grupēšana, skatīt piemērā "Studijas". Prasības tiek prioritizētas pēc MoSCoW (Vestola, 2010) tehnikas. Prasība var būt "Vienkārša prasība" ("*Simple Requirement"*) vai "Salikta prasība" ("*Complex Requirement"*). "Salikta prasība" var sastāvēt no divām vai vairākām "Vienkāršām prasībām", kuras var būt saistītas ar aritmētisku operatoru ("*Arithmetic operator*"), piemēram "/" vai "Salīdzināšana" (*"Comparison"*). "Vienkāršā prasība" var saturēt tikai "Izteiksmi" ("*Expression*"). Piemērā, kas redzams att. 2.3. tiek pielietots nosacījums, lai salīdzinātu prasību ar iepriekš definētu konstanti.

[7] https://mariadb.org
[8] https://neo4j.com

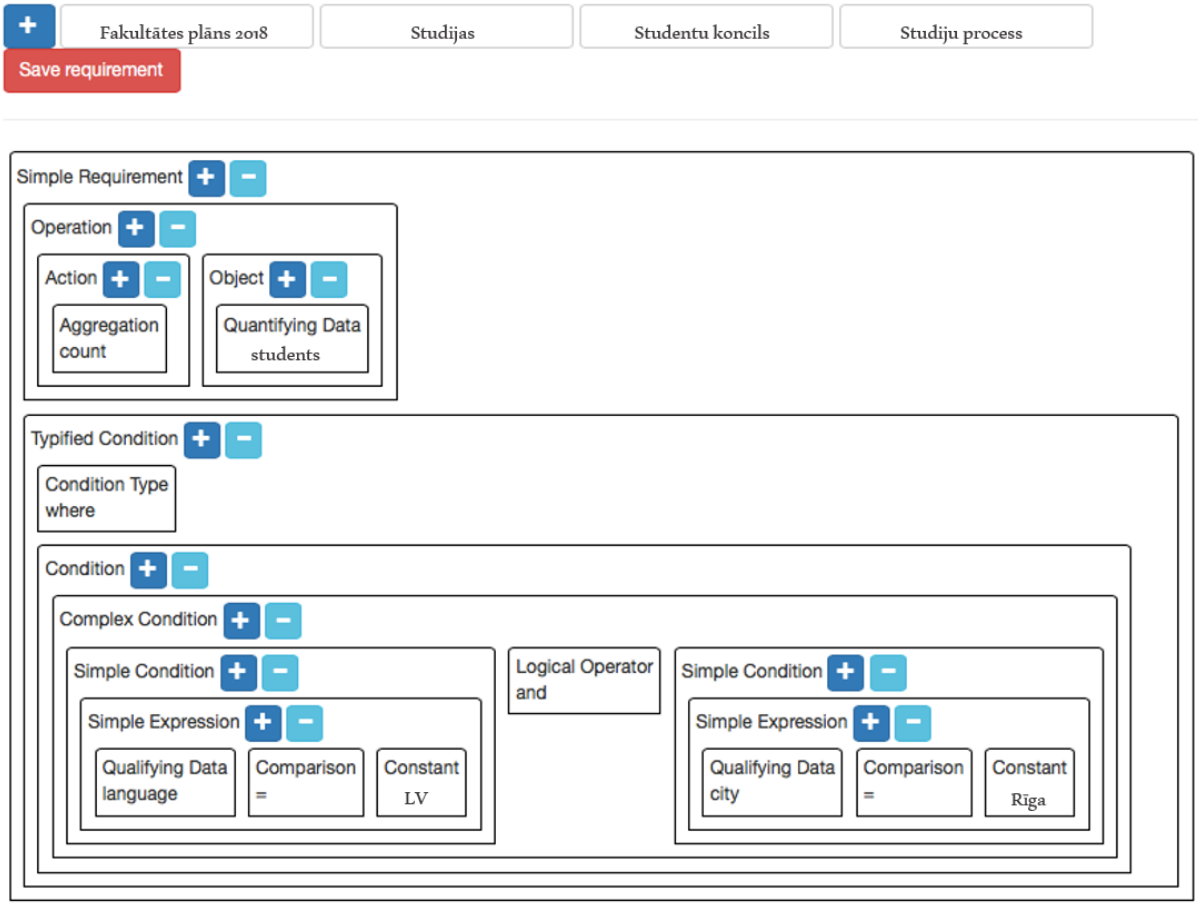
“Salikta prasība” sastāv no vienas vai vairākām “Izteiksmēm”, saistītām ar “Aritmētisku operatoru”. “Vienkārša izteiksme” var būt “Kvalificēti” dati (piemēram, - “darbinieku grupa”, “valoda” vai “pilsēta”) vai “Konstante” (piemēram – “akadēmiskais personāls”, “LV” vai “Riga”). “Vienkārša prasība” var saturēt “Operatoru” (*“Operator”*), kas nosaka komandu, kāda jāpielieto uz noteikto objektu. Papildus var tikt pievienots “Aprakstošais nosacījums” (“*Typified Condition*”). “Salikta prasība” sastāv no viena vai vairākiem “Notikumiem” (“*Action*”), kas var būt “Agregācija” (“Aggregation”), OLAP uz augšu ejošā operācija (*roll-up*) vai “Precizējošā” (“*Refinement*”) OLAP operācija (*drill-down*). “Objekts” (“*Object*”) ir instance kādam “Kvantitatīvam” (*“Quantitative”*) (piemēram, “studentu skaits”, “pasniedzēju skaits”) vai “Kvalificējošam” (*“Qualifying”*) (piemēram, atribūti vai īpašības). Lai sadalītu informāciju par klasi “Objekts, tiek pievienots ierobežojums, kas ir metamodeļa elements “Aprakstošais nosacījums”, piemērā att. 2.4. var redzēt “*where language = ‘LV’ and city = ‘Riga’* ”. “Salikta prasība” savieno vienu vai vairākus “Nosacījumus”, piemērā *“language = LV”* un *“city = Riga”* "Nosacījums" tiek savienots ar “Loģisko operatoru” (*“Logical operator”*) (piemēram – “vai”, “ne” vai “un”). “Vienkāršs nosacījums” (*“Simple condition”*) sastāv no “Salīdzināšanas” (“*Comparison*”) ar iespējamām vērtībām “>”, “<=”, “>=”, “=”.



Att. 2.3. Piemērs datu noliktavas prasībai "*Attēlot informāciju par studentu un akadēmiskā personāla attiecību katrā fakultātē* ", atbilstošā formālā izteiksme "*show (fakultāte) count (students) / count (darbinieks) where employee_group = 'akad. pers.'*"

Principi, kas pielietoti, lai formalizētu Att. 2.3 redzamo piemēru:

- Komponente, kura tiek paredzēta būt mērījums, tiek veidota kā agregācija, piemēram, "studenti" ir pārformulēta par "count(students)", kur agregācija ir vispiemērotākā funkcija;
- Gadījumos, kad prasība sastāda vārdus "%", "procenti" vai "attiecība", procenti tiek formalizēti kā vērtības dalīšana ar kopējo skaitu, bet attiecība – dalījums starp divām vērtībām. Piemēram, prasība "studentu un darbinieku attiecība" tiek pārformulēta kā "count(students)/count(darbinieks)".

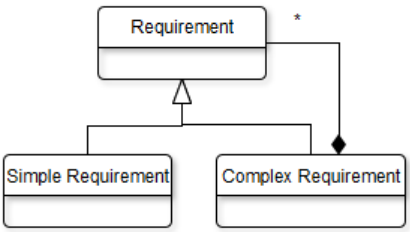


Att. 2.4. Piemērs datu noliktavas prasībai "*Attēlot informāciju par studentiem, kas ir no Rīgas un apmeklē lekcijas latviešu valodā*", atbilstošā formālā izteiksme "*count (students) where language = "LV" and city = "Rīga*"

2.6.2 Rīka iReq konfigurācija

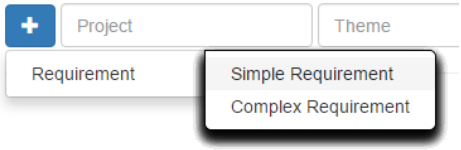
iReq rīka galvenais mērķis ir nodrošināt formālo prasību ievadīšanu un saglabāšanu formālo prasību repozitorijā, atbilstoši formālo prasību metamodelim (Kozmina et al., 2017). Formālo prasību metamodeļa klases ir aprakstītas iReq rīka JSON formāta konfigurācijas failā.

Att. 2.5. var redzēt fragmentu no formālu prasību metamodeļa, kurā modelētais elements "Requirement" sastāv no elementiem "Simple requirement" un "Complex requirement", kā arī "Complex requirement" var sastāvēt no vairākiem "Requirement" elementiem.



Att. 2.5. Formālo prasību metamodeļa fragments

Att. 2.6. var redzēt iReq rīka grafisko saskarni, ievadot formālās prasības. Lietotāja saskarne darbojās, atbilstoši konfigurācijas failam. Šajā gadījumā lietotājam tiek attēlots elements "Requirement", kuram ir divas apakšizvēlnes "Simple requirement" un "Complex requirement".



Att. 2.6. iReq rīka lietotāja saskarne pievienojot jaunu formālo prasību

iReq rīka konfigurācijas fails sastāv no JSON objektiem un to atribūtiem. Katrs JSON objekts atbilst kādam formālo prasību metamodeļa elementam. JSON objektam pastāv šādi elementi:

- Id – unikāls identifikators, piemēram, "arithOper";
- Name – nosaukums, kas tiks attēlots lietotāja grafiskajā saskarnē, piemēram, "Arithmetical Operator"
- Parent – citu elementu ID sarakts, kuri ir "vecāki" šim elementam. Var būt *NULL* vērtība. Lietotāja saskarnē šis elements tiks attēlots zem šiem "vecāku" elementiem;
- Action – darbība, kādu veiks iReq grafiskās saskarnes funkcionalitāte, iespējamā vērtība "dropdown". Gadījumos, kad "Action" atribūta vērtība ir "dropdown", lietotājam tiks attēlota izvēlne ar vērtībām no atribūta "values";
- Values – vērtības, ko attēlo lietotājam uzklikšķinot ar peli uz elementa, kuram ir atribūta "Action" vērtība "dropdown". Elementa "Arithmetical Operator" gadījuma vērtības ir "+", "-", "*", "/".

Att. 2.7. var redzēt fragmentu no konfigurācijas faila. Var redzēt, ka šis fragments sastāv no trīs objektiem ar ID "req", "simpleReq" un "complexReq". Starp objektiem ir nodefinētas "vecāks" un "bērns" attiecības, piemēram, elementam ar ID "simpleReq" ir vecāki "req" un "complexReq", tas nozīmē, ka elements "simpleReq" tiks attēlots rīka izvēlnē zem elementiem "req" un "complexReq". Esošais konfigurācijas fails nodrošina

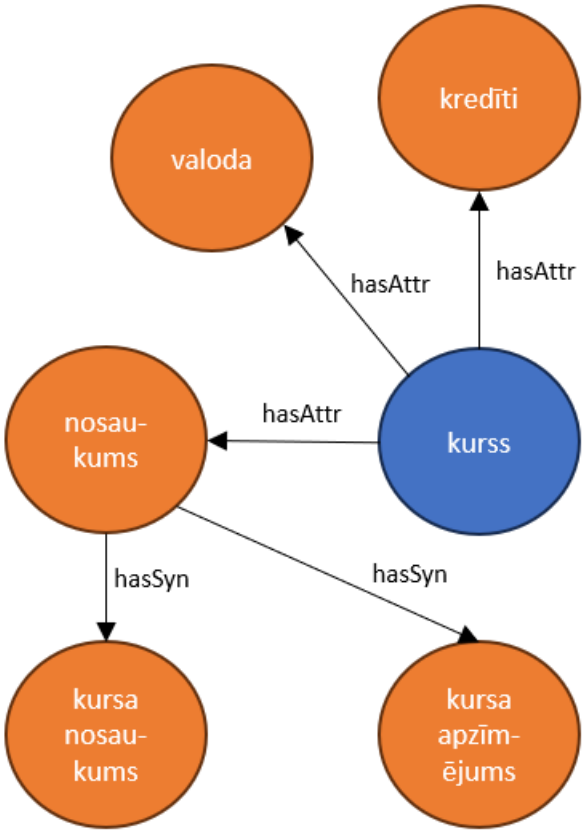
ievadīt prasības atbilstoši paplašinātajam formālajam prasību metamodelim, bet tas nenodrošina lietotājam ievadīt kļūdainus vai neatbilstošus rezultātus, piemēram, pastāv iespēja ievadīt divus "Arithmetical Operator" objektus viens otram blakus.

```
{
    id: 'req',
    name: 'Requirement',
    parent: null,
    values: null,
    action: null
},
{
    id: 'simpleReq',
    name: 'Simple Requirement',
    parent: ['req', 'complexReq'],
    values: null,
    action: null
},
{
    id: 'complexReq',
    name: 'Complex Requirement',
    parent: ['req', 'complexReq'],
    action: null,
    values: null
},
```

Att. 2.7. iReq rīka JSON konfigurācijas faila fragments

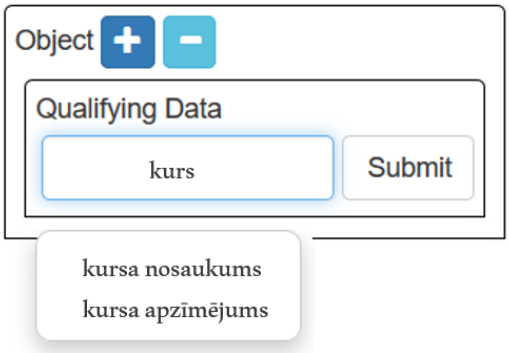
2.6.3 Atribūtu ievade iReq rīkā un jēdzienu vārdnīcas

iReq rīka funkcionalitāte tika papildināta ar jēdzienu vārdnīcu, kas ir ieviesta izmantojot grafu datu bāzi Noe4j, kur jēdzieni savā starpā ir saistīti ar dažāda veida saitēm. iReq jēdzienu vārdnīcu tiek paredzēts veidot par pamatu izmantojot datu noliktavas avota sistēmās pieejamos atribūtus un ar iespēju lietotājam papildināt šo vārdnīcu ar jēdzienu sinonīmiem un jauniem jēdzieniem. Grafu orientētajai datu bāzei Neo4j ir ērta lietotāja grafiskā saskarne, ar kuras palīdzību var viegli veikt jēdzienu vārdnīcas izmaiņas. Tiek rekomendēts šo jēdzienu vārdnīcu veidot tieši no datu noliktavas avota sistēmās eksistējošiem tabulu un atribūtu nosaukumiem. Vārdnīcu var veidot manuāli vai automātiskā veidā, eksportējot avota sistēmas shēmas struktūru CSV formātā un importējot to Noe4j datu bāzes vadības sistēmā. Kā piemēru var apskatīt, ja avota sistēmā pastāv tabula "kurss" ar kolonām "nosaukums", "valoda" un "kredīti", tad tiek noteikts veidot grafu, kas aplūkojams att. 2.8.



Att. 2.8. Grafa piemērs no grafu datubāzes komponentes – jēdzienu vārdnīca

Kolonnu un tabulu nosaukumi tiek saistīti ar saiti "hasAttr", visi šie atribūti tiek piedāvāti lietotājam ievadot prasības kvantificējošus un kvalificējošus atribūtus. Att. 2.8. var redzēt, ka jēdzienu vārdnīca ir papilināta ar jaunu atribūtu "kursa nosaukums", kas saistīta ar atribūtu "nosaukums" un saites tipu "hasSyn", kas norāda, ka klase "kursa nosaukums" ir sinonīms atribūtam "nosaukums". Lietotājam ievadot frāzi "kurs", tiks piedāvātā frāzes "kursa nosaukums" un "kursa apzīmējums", skatīt rīka iReq funkcionalitāti att. 2.9.



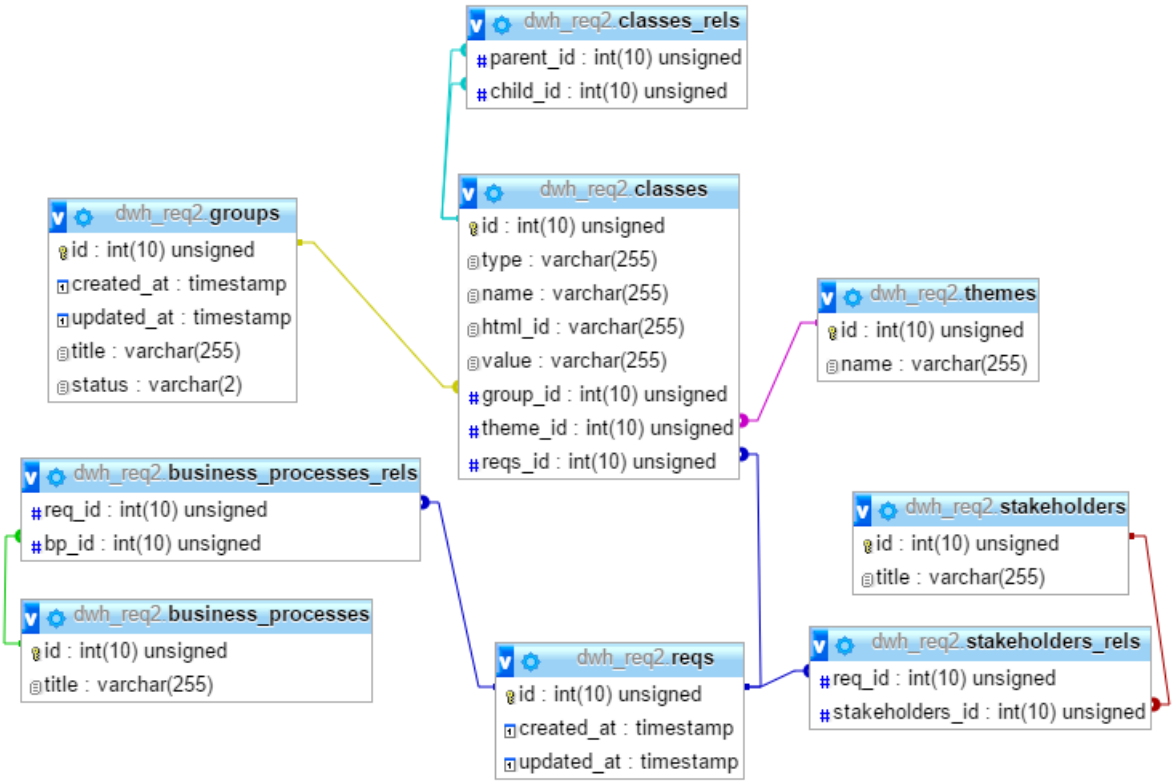
Att. 2.9. Rīka iReq formālo prasību ievadīšana, izmantojot jēdzienu vārdnīcas funkcionalitāti

Ievadot atribūtus, lietotājam tiek piedāvātas vērtības no jēdzienu vārdnīcas, bet joprojām tiek saglabāta iespēja neizvēlēties nevienu no piedāvātajām vērtībām, jo jēdzienu vārdnīca var būt nepilnīga un tādejādi lietotājam var tikt liegta iespēja ievadīt prasību. Lietotājam ievadot prasības ar jēdzienu vārdnīcā pieejamajiem jēdzieniem vajadzētu pietikt. Tiek rekomendēts lietot tieši piedāvātās vērtības, kas nākamajos datu noliktavas izstrādes posmos atvieglo datu noliktavas izstrādi un tiek ģenerēta precīzāka kandidātshēma.

Viens no iemesliem kādēļ lietotājam tiek atļauta iespēja ievadīt frāzes, ko rīks nepiedāvā ir, ka pastāv gadījumi, kad datu noliktavas prasības pieteicējs vēlas pieteikt informācijas prasību, kas nepastāv vēl avota sistēmā, attiecīgi, šis jēdziens nepastāvēs arī jēdzienu vārdnīcā. Šādas prasības var norādīt uz jaunu datu iegūšanu un saglabāšanu datu noliktavas avota sistēmā.

2.6.4 *iReq rīka formālo prasību repozitorijs*

Formālās prasības struktūra ir hierarhiska, kur katrai prasībai pastāv saknes elements un tās "bērnu" elements, kas ir saistīti ar "bērns - vecāks" saistībām, piemēram, "Salikta prasība" var sastāvēt no vienas vai vairākām "Vienkāršām prasībām". Teorētiski prasība var sastāvēt no neierobežoti daudz "bērnu" elementiem. iReq rīks saglabā katra konkrēta izvēlēta "bērna" visus "vecākus". Visi lietotāja grafiskajā saskarnē izvēlētie elementi ir apstrādāti no zemākā "bērna" elementa uz augšu, saglabājot tos rīka iReq datubāzē.



Att. 2.10. iReq rīka formālo prasību datu modelis

Att. 2.10. var redzēt rīka iReq formālo prasību repozitorija fizisko datu bāzes modeli. Tabulā "classes" tiek saglabāti elementi no kā sastāv formālā prasība piemēram, "Prasība", "Salikta prasība", ieskaitot katra elementa tipu, kas attiecīgi ir kāds formālā prasību metamodeļa elements, piemēram, "Kvalificējošie dati", "Notikums", "Vienkāršs nosacījums". Tabula "classes_rel" saglabā attiecību starp formālās prasības elementiem. Informācija par prasības ieinteresētajām personām un saistītajiem biznesa procesiem ir saglabāta tabulās "buniness_processes", "business_processes_rels" un "stakeholders", "stakeholders_rels". Katra prasība tiek piesaistīta kādai grupai (tabula "groups") un tēmai (tabula "theme"). Katru reizi veiksmīgi saglabājot vai atjaunojot prasību, tiek uzģenerēts jauns ID ieraksts tabulā "reqs" un visās saistītajās tabulās.

Tabula "reqs" reprezentē vienu formālu datu noliktavas prasību, kura fiziski var tikt saglabāta vairākas reizes. Formālo prasību metamodeļa elementi "Vienkārša prasība" un "Salikta prasība" tiek saglabāti, tāpat kā citi, metamodeļa tabulā "classes", kur tos atšķir ar atribūtu "type". Iespējamie tipi, to "vecāka-bērna" attiecības ir definētas iReq konfigurācijas failā, nodaļā 2.6.2. Mainoties metamodelim un konfigurācijas failam, esošo datu modeli nav nepieciešams mainīt.

2.7 Secinājumi

Šajā nodaļā tiek aplūkotas tradicionālas datu noliktavas izstrādes prasības un informācijas prasību jēdziens. Datu noliktavas prasības var iedalīt trīs grupās - funkcionālās, informācijas un citas prasības. Datu noliktavas kontekstā īpaša uzmanība ir jāpievērš informācijas prasībām, jo tās nosaka, kādi dati tiek saglabāti datu noliktavā.

Tiek aplūkotas pastāvošās prasību vākšanas metodes tradicionālām datu noliktavām, viena no aplūkotajām metodēm (Kozmina & Niedrite, 2014) un (Kozmina et al., 2013) nosaka informācijas prasības formalizēt, izmantojot īpašu metamodeli. Šī metode ir papildināta ar jaunām komponentēm "Grafiskā saskarne formālo prasību ievadei", "Formālo prasību savienošanas komponente" un "Grafu datubāzes komponente", kā arī metodē izmantotais metamodelis tiek papildināts ar jaunām klasēm "Darījuma process" un "Ieinteresēta persona". Nodaļā tiek aprakstīts rīks "iReq", paplašinātā metode un paplašinātais metamodelis. Šis rīks nodrošina daļēji automātiskā veidā ģenerēt datu noliktavas kandidātshēmas modeļus. "iReq" rīks nodrošina grafisku saskarni, kas ļauj, atbilstoši paplašinātajam metamodelim, ievadīt formālās datu noliktavas prasības. Lai uzlabotu formālo prasību ievadīšanu un līdzīgo prasību apvienošanu, tiek izmantota ontoloģija, kura tiek veidota par pamatu izmantojot datu noliktavas avotu sistēmu shēmas struktūru. Ievadītās prasības tiek saglabātas formālo prasību

repozitorijā. Izmantojot speciālu “PGA” algoritmu, tiek ģenerēti datu noliktavas kanditātshēmu modeļi. Rīks “iReq” nodrošina apstrādāt formālas datu noliktavas prasības, lai daļēji automātiskā veidā iegūtu datu noliktavas kandidātshēmas.

Šajā nodaļā aprakstītā un uzlabotā metode, lai ģenerētu datu noliktavas kandidātshēmas, izmanto formālas datu noliktavas prasības. Formālajām datu noliktavas prasībām ir noteikta struktūra (atbilstoša papildinātajam metamodelim), taču, pieaugot lielo datu apjomiem un nestrukturētajiem datiem, rodas jautājums: “Vai ir iespējams ģenerēt datu noliktavas datu modeļus, izmantojot liela apjoma un nestrukturētus datus?”. Lai atbildētu uz šo jautājumu, nākamajā nodaļā ir veikts literatūras pārskats par iespējām veikt prasību vākšanu, datu noliktavām pielietojot lielo datu tehnoloģijas.

3 PRASĪBU VĀKŠANA DATU NOLIKTAVĀM LIELO DATU LAIKMETĀ

Šajā nodaļā apskatītas prasību inženierijas aktivitātes lielo datu kontekstā. Tiek aplūkots lielo datu jēdziens, prasību iegūšanas paņēmieni lielo datu projektos un lielo datu apstrādes metodes. Nodaļā tiek analizēti pētījumi, kas apraksta lielo datu apstrādes metodes. Šī nodaļa balstās uz publikācijām (Kozmina et al., 2018) un (Kozmina et al., 2019).

3.1 Lielie dati

Lielo datu termins pirmās reizes ir parādījies akadēmiskajā literatūrā 2000. gadu sākumā statistikas un ekonometrijas nozarēs. Lielo datu termins tika izmantots, lai aprakstītu liela apjoma pieejamos, neapstrādātos datus un to ielasīšanas un saglabāšanas tehnoloģijas (Diebold, 2010). Lielo datu tehnoloģijas tiek raksturotas ar "trīs V īpašībām" (*3Vs*) (Laney, 2001):

- Liela apjoma (*volume*);
- Ātra apstrāde (*velocity*);
- Dažāda veida (*variety*).

Vēlākajos gados lielo datu tehnoloģijām tika pievienotas citas raksturojošas īpašības – patiesums (*veracity*) (Ward & Barker, 2013), vērtība (*value*) (Favaretto et al., 2020), mainīgums (*variability*) (Fan & Bifet, 2013), sarežģītība (*complexity*) (Perry, 2017) u.c.

Autori 2011 gadā (Brown et al., 2011) apraksta *lielo datu laikmetu* un ar to saistītos jautājumus. *Lielo datu laikmeta* iestāšanās tiek skaidrota ar organizāciju vēlmi strauji investēt datos, tas ir, lai iegūtu, ieintegrētu datus un izmantotu tos eksperimentos, kā arī organizācijas ikdienas darbībā.

Attīstoties tādām tehnoloģijām kā mākoņskaitļošanas tehnoloģijai, 5G[9] interneta pieejamībai, sociālajām platformām, lietu internetam (*Internet of Things*), strauji pieaug datu apjomi un datu veidi organizācijās (Wang et al., 2022). Lielo datu tehnoloģijas paver iespēju šādu datu apstrādē un analīzē (Wang et al., 2022). Laika posmā no 2011.gada līdz 2022.gadam, populārākās nozares, kurās tiek pielietota lielo datu analītika, ir inženierzinātnes, datorzinātnes, automatizācijas kontroles, ekonomikas un telekomunikācijas (Wang et al., 2022).

Lielo datu avotu kopas parasti ir dažāda formāta, neviendabīgas. Autori (Ikegwu et al., 2022) lielo datu avotus iedala sešās kategorijās:

- Izglītības un zinātnes dati;

[9] https://www.cisco.com/c/en/us/solutions/what-is-5g.html

- Lietu interneta un sensoru dati;
- Sociālo platformu dati;
- Datizraces dati;
- Multivides dati;
- Veselības aprūpes dati.

Izglītības un zinātnes dati tiek galvenokārt iegūti no augstākās izglītības studentu datiem, piemēram, lekciju apmeklētība, atzīmes, sporta aktivitātes, absolventi, u.c. (Khan, 2018). Pētījumā (Attaran et al., 2018) tiek izmantoti izglītības dati, lai ar lielo datu analīzes iespējām uzlabotu augstskolu un studentu sasniegumus. Augstskolas var pilnveidot savu darbību, izmantojot lielo datu analīzes iespējas, lai uzlabotu, piemēram, uzņemšanas kārtību, studentu pabalstu piešķiršanas kārtību, augstskolas finansēšanas pārvaldību u.c.

Lietu internets un sensoru dati tiek iegūti no savstarpēji savienotām ierīcēm, piemēram, kameras, mikrofoni, viedpulksteņi, viedtālruņi, lietus sensori, mitruma sensori, u.c. Tāda veida sensori, kā pirkstu nospiedumu lasītājs, adreses identifikatora noteicējs u.c., ir izcili piemēroti, lai nodrošinātu datu vadītu (*data-driven*) vidi (Tortonesi et al., 2019).

Sociālo platformu dati tiek ģenerēti sociālajos tīklos lietotājiem publicējot ierakstus, daloties ar failiem, straumējot datus u.c. Pastāv dažādas sociālās platformas, piemēram, Instagram, YouTube, Facebook, Twitter, Yahoo Messenger, Skype u.c. Pētījumā (Kim, 2019) tiek izmantoti sociālo platformu dati, lai uzlabotu tūrisma jomu aizsargājamās teritorijās.

Datizraces dati ir liela apjoma un nestrukturēti. Šādi dati var tikt iegūti no tērzēšanas (*chatting*) platformām, klientu aprūpes pakalpojumiem, darījumu inteliģences, e-pasta vēstulēm, reklāmām u.c. (Wu & Guan, 2021). Lai pārstrādātu šādus datus, tiek ieviestas jaunas metodes. Metodē (Tamrakar et al., 2020) tiek aprakstīts jauns neironu tīkls (*FP-ANN*), lai apstrādātu nestrukturētu tekstu ar teksta datizraces tehnikām. Metode (Tamrakar et al., 2020) apstrādā datus ātrāk un patērē mazāk servera resursus.

Multivides dati ir dati video, attēla, skaņas, animācijas formātos (Rousseeuw et al., 2018). Pētījumā (Hu et al., 2019) tiek aprakstīta atvērtā pirmkoda platforma, kas tiek izmantota bilžu analīzei. Tiek izmantota Python bilžu programmatūras pakotne - Image.Py[10]. Šī pakotne ir darbināma uz jebkura mūsdienu datora, lai apstrādātu sarežģītas bildes no dažādām jomām, kā piemēram, medicīnas.

Veselības aprūpes dati ir dažāda veida un bieži tie tiek ģenerēti veselības pakalpojumu sniedzēju iestādēs. Medicīnas dati iekļauj klīnisko izmeklējumu datus, farmaceitisko līdzekļu aprakstus, sensoru ierīču datus, ārstniecības personāla datus, medicīnisko pakalpojumu

[10] https://pypi.org/project/image

izmaksas (Vidhya & Shanmugalakshmi, 2020). Veselības aprūpes dati tiek ģenerēti dažādās nozarēs. Veicot veselības aprūpes datu analīzi, ir iespējams uzlabot organizācijas pārvaldību un lēmumu pieņemšanas procesu. Metodē (Vidhya & Shanmugalakshmi, 2020) tiek izstrādāta tehnoloģija, kā prognozēt cilvēka veselības stāvokli, kā datu avotu izmantojot Twitter datus. Tehnoloģija darbojās reālā laikā, izmantojot Apache Spark un mašīnmācīšanās lēmumu pieņemšanas shēmu.

3.2 Prasību inženierija lielo datu projektos

Lielo datu tehnoloģijas tiek plaši pielietotas dažādās jomās, lielo datu tehnoloģijas bieži tiek saistītas ar tādiem terminiem, kā lietu internets, sociālie tīkli, lielo datu analīze, mākoņskaitļošana, NoSQL datubāzes u.c. Pastāv dažādi veidi, kā tiek ieviesti lielo datu risinājumi organizācijās, bet ne vienmēr projekti ir veiksmīgi. Neveiksmes lielo datu izstrādes projektos var saistīt ar dažādām problēmām (Kart et al., 2013):

- Nespēju iegūt noderīgu informāciju no datiem;
- Integrācijas problēmām starp lielo datu avotiem;
- Datu kvalitātes problēmām.

Pastāv arī ar tehnoloģiju saistītas problēmas (Katal et al., 2013):

- Atmiņas ierobežojumi;
- Apstrādes problēmas;
- Analīzes un tehniskie izaicinājumi.

Autori apgalvo (Tardio et al., 2015), ka problēmas lielo datu projektos var tikt novērstas izvēloties un pielietojot pareizos rīkus un izstrādes metodes, piemēram, pareizos automatizācijas rīkus vai projekta izstrādes metodoloģijas (Di Tria et al., 2014). Pastāv vairāki piedāvājumi, kā atrisināt konkrētu problēmu lielo datu izstrādes procesā, bet nepastāv vienotas nostājas par šādiem jautājumiem:

- Kādu projekta izstrādes metodoloģiju izvēlēties, izstrādājot lielo datu risinājumu, par pamatu izmantojot biznesa procesus un prasības?
- Kāda informācija ir jānoskaidro, lai izstrādātu projektu atbilstoši informācijas prasībām?

Analizējot literatūras avotus, tika aplūkotas pastāvošās metodes par prasību analīzi lielo datu kontekstā, vērtējot sekojošus aspektus:

- Metodes darbību kopumā;
- Prasību analīzes saistītās īpašības;
- Lielo datu tehnoloģiju saistītās īpašības.

Viens no lielo datu izstrādes projektu neveiksmju iemesliem ir tas, ka sākotnēji netiek noskaidrotas visas nepieciešamās informācijas prasības vai arī, lietotāja sākotnējās prasības ir atšķirīgas no izstrādātā produkta, piemēram, neatbilstoša datu kvalitāte vai neatbilstoša lietotāju tiesību pārvaldība. Dažos gadījumos projekta neveiksmi var izskaidrot ar to, ka netika veiktas prasību analīzes aktivitātes pirms lielo datu projekta izstrādes. Var secināt, ka prasību inženierijai lielo datu izstrādes projektos ir būtiska nozīme. Prasību un datu analīze būtu jāintegrē divos soļos:

- Pirms projekta izstrādes noskaidrot lietotāja prasības un salīdzināt šīs prasības ar pieejamajiem datiem avota sistēmās, datu kvalitāti, granulitāti, projekta pieejamajiem resursiem, izstrādātāju zināšanām, budžetu un laika termiņiem, (Tardio et al., 2015), (Arruda & Madhavji, 2017), (Eridaputra et al., 2014) un (Mallek et al., 2017);
- Posmā, kad dati jau ir ielasīti no avota sistēmām. Pastāv iespēja, ka lietotājs nav nodefinējis visas informācijas prasības. Veicot reālo datu analīzi, var atklāties jaunas saistības starp datiem un rasties jaunas informācijas prasības, (Santos & Costa, 2016), (Liu et al., 2015) un (Cheptsov et al., 2014);

Autori (Kotonya & Sommerville, 2011) nosaka prasību inženieriju iedalīt četros posmos:

- Prasību iegūšana – šajā posmā tiek iegūtas prasības no projekta ieinteresētajām personām;
- Prasību analīze – šajā posmā tiek iegūta jauna informācija. Nepieciešamības gadījumā precizētas ar ieinteresētajām personām;
- Prasību specificēšana – prasības tiek specificētas, strukturētas, lai tās izmantotu programmatūras izstrādē. Nepieciešamības gadījumā prasības tiek precizētas ar ieinteresētajām personām;
- Prasību validēšana – tiek apstiprināts, ka prasības atbilst ieinteresēto personu vajadzībām. Nepieciešamības gadījumā prasības tiek precizētas ar ieinteresētajām personām.

Nākamajās nodaļās tiek aplūkotas lielo datu apstrādes metodes, kuru aktivitātes ir attiecināmas uz kādu no augstāk minētajiem prasību inženierijas posmiem. Lai noskaidrotu lielo datu prasību inženierijas īpašības, tika veikts literatūras pārskats, kas publicēts rakstos (Kozmina et al., 2018) un (Kozmina et al., 2019). Šis literatūras pārskats tika veikts atbilstoši Kitchenham and Charters vadlīnijām (Keele, 2007). Literatūras pārskata mērķis ir izpētīt informācijas prasību aspektus lielo datu kontekstā. Lai sasniegtu mērķi, tika izvirzīti šādi literatūras pārskata jautājumi:

- Kā prasību analīze tiek veikta lielo datu projektos?
- Kādas metodes ir pielietotas prasību analīzei lielo datu kontekstā?
- Vai var ģenerēt informācijas prasības lielo datu projektos, automātiskā vai daļēji automātiskā veidā apstrādājot datus?

Literatūras pārskatā tika iekļauti raksti pēc šādiem kritērijiem:

- Pētījumi, kas publicēti laika periodā no 2014. līdz 2017. gadam;
- Pētījumi, kas ir indeksēti kādā no publikāciju datubāzēm – ACM ("*Association for Computing Machinery*"), Google Scholar, SpringerLink, Scopus, Web of Science, IEEE ("*Institute of Electrical and Electronics Engineers*");
- Pētījumi, kas ir aprakstīti angļu valodā;
- Nosaukums, atslēgas vārdi, anotācija ir saistīti ar literatūras pārskata jautājumiem;
- Pētījuma saturs (virsraksti, attēli, tabulas, ievads un secinājumi) ir saistīti ar literatūras pārskata jautājumiem.

No literatūras pārskata tika izkļauti raksti atbilstoši šādiem kritērijiem:

- Dublikāti;
- Viena pētījuma dublicējošās versijas (pēdējā vai detalizētāk aprakstītā tika atstāta);
- Disertācijas, maģistra darbi un bakalaura darbi;
- Atslēgas vārdi nav atrodami pētījuma aprakstā, bet izmantotās literatūras sarakstā;
- Pētījums ir pārāk detalizēts konkrētam domēnam un to nevar vispārināt citiem domēniem.

Meklējot publikāciju datubāzēs, izmantojot atslēgas vārdus "lielie dati" ("*big data*"), "prasību inženierija" ("*requirement engineering*"), sākotnēji tika atrasti 242 pētījumi, bet pielietojot augstāk minētos kritērijus, detalizētai analīzei tika izvelēti 22 pētījumi, uz kuru pamata tika veikts literatūras pārskats.

Literatūras pārskatā tiek aplūkoti raksti, kas saistīti ar prasību inženieriju un lielajiem datiem. Metodes tika aplūkotas pēc vienotiem principiem:

- Kuri prasību inženierijas posmi ("prasību iegūšana", "prasību analīze", "prasību specifikācija" un "prasību validācija") ir attiecināmi uz metodes darbību;
- Kādi prasību noteikšanas paņēmieni tiek pielietoti ("scenāriji", "mērķu noteikšana", "metodes specifisks paņēmiens", u.c.);
- Metodes piemērotība, lai veiktu kādu no prasību inženierijas posmiem.

3.3 Lielo datu apstrādes metodes, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmiem “prasību iegūšana”, “prasību analīze”, “prasību specifikācija” un “prasību validācija”

Šajā nodaļā tiek aplūkotas lielo datu apstrādes metodes, kuru darbība ir attiecināma uz prasību inženierijas posmiem “prasību iegūšana”, “prasību analīze”, “prasību specifikācija” un “prasību validācija”.

Metodē “Towards a comprehensive data lifecycle model for big data environments” (Sinaeepourfard et al., 2016) tiek izmantots “Data LifeCycle (DLC)” modelis. Šī modeļa mērķis ir nodrošināt organizācijas datu dzīvescikla posmus, katra posma pārvaldības principus un posmu savstarpējo saistību. Tiek aprakstīts speciāls scenārijs “Comprehensive Scenario Agnostic Data LifeCycle”, kas sastāv no trīs savstarpēji saistītiem blokiem:

- Datu iegūšanas bloks, kas sastāv no četrām fāzēm:
  - Datu iegūšanas – datu iegūšana no avota sistēmām un ierīcēm, atbilstoši biznesa prasībām;
  - Datu filtrēšanas – vienkāršas datu transformācijas ar mērķi samazināt apstrādājamo datu apjomu nākamajās fāzēs;
  - Datu kvalitātes – atmest zemas kvalitātes datus;
  - Datu aprakstīšanas slānis.
- Datu apstrādes bloks:
  - Datu apstrādes – neapstrādātu datu apstrāde, lai iegūtu izsmalcinātākus datus atbilstoši biznesa prasībām;
  - Datu kvalitātes – datu kvalitātes pārbaudes solis;
  - Datu analīzes;
- Datu saglabāšanas – nodrošina datu saglabāšanu, klasificēšanu un padara tos pieejamus apstrādei nākotnes vajadzībām.

Šī metode apraksta aktivitātes, kas būtu jāveic katrā blokā, neiedziļinoties detaļās, kā to veikt. Metode paredz noskaidrot prasības aprakstot scenārijus. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metode ir pielāgojama dažāda veida scenārijiem un prasībām, saglabājot augstu datu kvalitāti.

Metodē “An iterative methodology for big data management, analysis and visualization” (Tardio et al., 2015) tiek prezentēta iteratīva metodoloģija, kā uzlabot lielo datu risinājumu šādos aspektos – pārvaldība, analīze un vizualizācija. Metodoloģijai ir piecas fāzes:

1. Datu stāvokļu definēšana;
2. Datu avotu noskaidrošana un pārvaldība;
3. Informācijas iegūšana no datiem;

4. Datu izvēle un datu noliktavas izstrāde;
5. Lielo datu tehnoloģiju vizualizācijas izstrāde.

Pirmajā posmā tiek noteiktas informācijas un nefunkcionālās prasības katram analīzes uzdevumam izstrādes projektā. Informācijas prasības kalpo, lai noteiktu datu struktūru, bet nefunkcionālās prasības (datu kvalitātes prasības, laika ierobežojumi, vaicājumu ātrdarbība) nosaka atbilstošākos rīkus un modeļus katram analīzes uzdevumam. Autori pielieto multidimensonālas modelēšanas tehniku, jo tā ir vienkārša un efektīva. Trešais posms nosaka avota sistēmu datu izpēti. Katrai informācijas prasībai tiek pārbaudīti pieejamie dati avota sistēmās un to noderīgums. Ceturtais solis nosaka datu pārveidi faktos un dimensijās. Multidimensonālie modeļi katrai avota sistēmai iteratīvi tiek integrēti vienā kopīgā modelī, salīdzinot līdzīgos faktus. Piektajā posmā tiek implementēta organizācijas datu noliktava. Tādi rīki kā Apache Pig[11] vai Hive[12] var tikt izmantoti, lai atlasītu neapstrādātus datus no multidimensionālās datu noliktavas shēmas.

Metode paredz noskaidrot prasības, aprakstot izvirzot mērķus. Lielo datu vaicājumi tiek veidoti atbilstoši informācijas prasībām, ņemot vērā avota sistēmā pieejamos datus. Neapstrādātie dati tiek transformēti faktos un dimensijās. Katras avota sistēmas multidimesionālas modelis tiek integrēts vienotā modelī. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā.

Metode "Big pic-ture of big data software engineering: with example research challenges" (Madhavji et al., 2015). Tiek veidots lielo datu programmatūras inženierijas konteksta modelis. Šajā rakstā par prasībām tiek runāts kā vienu no lielo datu prasību inženierijas izaicinājumiem. Šī metode paredz veidot speciālus domēna modeļus, iegūt lietojuma piemērus un prasības no ieinteresētajām personām, izstrādāt funkcionālos un darbības modeļus, veikt analīzi, prioritizēšanu un validēšanu. Veidojot projektējumu, ir jāiekļauj tādas lielo datu specifiskās īpašības, kā datu apjoms, ātrdarbība, datu dažādība, kā arī domēna specifiskās prasības, piemēram, drošības jautājumi, privātums, rezultātu ticamība, lietošanas ērtums un personalizācijas iespējas.

Metode paredz noskaidrot prasības, aprakstot scenārijus. Scenāriji tiek iegūti, analizējot sistēmu rezultātus un darbību. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā.

Metode "Evolving mashup interfaces using a distributed machine learning and model transformation methodology" (Fernandez-Garcia et al., 2015) nosaka, ka saskarnes, kas ir uz komponentēm bāzētas, ir regulāri jāatjaunina par pamatu izmantojot lietotāju prasības un

[11] https://pig.apache.org
[12] https://hive.apache.org

prasību izmaiņas. Prasības var mainīties dažādās vidēs, laikam ejot. Metode sastāv no pieciem soļiem:

1. Pirmajā solī notiek informācijas ievākšana par lietotāju darbībām grafiskajā saskarnē, kā arī tiek ievākta informācija par lietotāja lokāciju un laiku. Šī informācija parasti ir liela apjoma un dažādas struktūras, tāpēc tā tiek apstrādāta ar lielo datu tehnoloģijām;
2. Otrajā solī tiek definēti skati par datiem, kas ievākti pirmajā solī - mērķis ir atlasīt atbilstošus datu elementus atšķirīgiem mašīnmācīšanās algoritmiem;
3. Trešais solis ir paredzēts mašīnmācīšanās algoritmu pielietošanai, lai pārbaudītu un novērtētu pareizo algoritmu un iegūtu noderīgu informāciju. Šī informācija ļauj ģenerēt jaunus grafiskās saskarnes noteikumus un uzlabojums;
4. Ceturtajā solī tiek izmantotas modeļu transformācijas ar ML algoritmiem, lai pārveidotu iegūtos datus. Rezultātā tiek iegūti jauni "pārveides noteikumi";
5. Piektajā solī ir jāizstrādā izvērtēšanas noteikumi, kas tiek iegūti no "pārveides noteikumiem".

Metode paredz noskaidrot prasības izmantojot īpašu, metodei specifisku paņēmienu. Metode nosaka iegūt un saglabāt lietotāju darbības. Lai izanalizētu šos datus, tos ir paredzēts apstrādāt ar mašīnmācīšanās algoritmiem. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metode paredz apstrādāt lietotāju darbību datus ar lielo datu tehnoloģijām, lai uzlabotu lietotāju saskarni.

Metode "An analysis pattern driven requirements modeling method" (Ji & Peng, 2016). Šī ir prasību analīzes modelēšanas metode, kas balstīta uz analīzes paraugiem, iepriekšēju pieredzi prasību modelēšanā, problēmu, mērķu un modeļu analīzē. Analīzes piemēra apraksts ir balstīts uz iepriekš definētu paraugu. Šī metode sastāv no pieciem soļiem, lai iegūtu prasības:

1. Izmantojot iepriekš definētu speciālu diagrammu - "sākotnējo lietotāju problēmu diagrammu" (*IUPD – "Initial User Problem Diagram"*), tiek iegūti prasību analīzes elementi un saistītie mērķi. Šo diagrammu ir paredzēts iegūt automātiskā veidā;
2. Atbilstošais analīzes paraugs tiek izvēlēts no paraugu repozitorija un pielietots speciālai diagrammai "IUPD" kā arī bagātināts ar sākotnējo diagrammu;
3. Tie izvēlēts labākais analīzes modelis no modeļu repozitorija;
4. Analīzes modelis tiek konfigurēts, lai iegūtu analīzes darbību (*AM - analysis machine)*. Šī analīzes darbība tiek integrēta ar speciālo diagrammu "IUPD".

Šī metode izmanto ne tikai analītiķu pieredzi, bet arī pieredzi, kas ir iegūstama no informācijas, kas pieejama paraugu repozitorijos.

Metode paredz noskaidrot prasības, izvirzot mērķus un veidojot analīzes šablonus. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā.

3.4 Lielo datu apstrādes metodes, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmiem "prasību iegūšana" un "prasību analīze"

Šajā nodaļā tiek aplūkotas lielo datu apstrādes metodes, kuru darbība ir attiecināma uz prasību inženierijas posmiem "prasību iegūšana" un "prasību analīze".

Metodes "Data warehousing in big data: from multidimensional to tabular data models" (Santos & Costa, 2016) priekšrocība tiek norādīta, ka nav nepieciešams definēt informācijas prasības pēc tam, kad ir definēts multidimensonālais modelis. Metode piedāvā likumu kopu, kā automātiski transformēt multidimensionālo datu modeli shēmā, kas sastāv no tabulām. Šī shēma var tik izstrādāta un implementēta, piemēram, izmantojot Hive un HiveQL[13] tehnoloģiju. Zvaigznes vai zvaigznāja shēma tiek ieviesta kā tabulas, kuras veido dimensijas, agregācijas, kas kalpo par pamatu analītiskajām vajadzībām.

Datu analīzes un vizualizācijas rīki (piemēram *Tableau*[14]) var tikt pielietoti, lai lietotāji varētu veikt nepieciešamo analīzi. Metode ir pielietota nelielam pētījumam, izmantojot Hive tehnoloģiju ar divām fakta tabulām un četrām dimensijas tabulām.

Metode izmanto īpašu veidu, kā noskaidrot prasības, proti, modelējot multidimensionālus modeļus, atbilstoši lietotāju prasībām. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metode aptver dažādus integrācijas aspektus, lai integrētu datu avotus, kuri ir modelēti pēc daudzdimensionāla principa lielo datu apstrādes dzīves ciklā.

Metodē "What Are My Users Looking for When Preparing a Big Data Campaign" (Ardagna et al., 2017) tiek pielietots TOREADOR ietvars, kā arī tiek aktīvi iesaistīti lietotāji un ieinteresētās personas prasību iegūšanā, lai bagātinātu informācijas prasības projekta sākotnējos posmos. Tas tiek darīts, jo tiek apgalvots, ka tas ir būtisks aspekts lielo datu projektos. Sākotnējais prasību saraksts tiek veidots balstoties uz ieinteresēto personu iesaisti. Visi iesaistītie izsaka prasības, izmantojot vienotu formātu "Vārds + Īpašības + Pamatojums + Tvērums + Prioritāte + Atkarības". Prasības tiek grupētas 6 grupās – priekšnosacījumi, infrastruktūra, servisa līmeņa vienošanās, juridiskās, eksperimentālās. Pēc prasību iegūšanas posma, prasībām ir jābūt nemainīgām, labi strukturētām. Nākamais solis nosaka veikt

[13] https://learn.microsoft.com/en-us/azure/hdinsight/hadoop/hdinsight-use-hive
[14] https://www.tableau.com

pētījumu, kurā tiek veikta Kano[15] aptauja, iegūtie dati tiek apstrādāti un agregēti, kas rezultējās Kano modelī. Kano modelis klasificē prasības, balstoties uz divām dimensijām:

- Ieinteresēto personu apmierinātība;
- Funkcionalitātes atbilstība.

Kano aptauja sastāv no jautājumu pāriem par katru produktu:

- Funkcionāla jautājuma, piemēram, kā būs, ja prasība tiks īstenota veiksmīgi?
- Pretēja jautājuma - kas notiks ar saistīto produktu, ja prasība netiks veiksmīgi īstenota?

Metode paredz noskaidrot prasības izvirzot mērķus. Prasības tiek iegūtas par pamatu izmantojot Kano aptaujas un analīzes paņēmienus. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metodes ietvars iekļauj metodei specifiskus modeļus lielo datu analīzes procesam.

Metodē "Big data services requirements analysis" (Yasin et al., 2018) galvenā uzmanība tiek pievērsta prasību apstrādei lielo datu lietojumprogrammu pakalpojumiem, nodrošinot uz prasībām balstītu pieeju pakalpojumu izstrādei un pakalpojumu izpildes vides atbalstam. Servisu apstrāde lielo datu kontekstā nozīmē servisu konveijeru izstrādi, lai atbalstītu lielo datu apstrādi. Tiek aptverts viss lielo datu dzīves cikls, tostarp, datu vākšana, glabāšana un analīze, kā arī pakalpojumu izstrādes process, tostarp, īstenošana, uzlabošana un optimizācija. Nepieciešamības gadījumā, pieeja iesaka izmantot esošos pakalpojumus, kas atbilst biznesa vajadzībām. Autori sniedz prasību klasifikācijas parametrus, kurus viņi izmanto kā pamatu lielo datu pakalpojumu prasību katalogam, kas aptver šādus aspektus: prasību tips, datu tips, biznesa process un veiktspējas indekss. Autori sniedz prasību piemērus, kas saistīti ar šādu komponentu izvēli:

- Glabāšanas veids;
- Datu konsekvences pārvaldības stratēģija;
- Datu pienākšanas režīms.

Metode nosaka, kā izveidot lielu datu apstrādes sistēmu, kas ir saskaņota ar biznesa vajadzībām un sastāv no atbilstošām sastāvdaļām un pakalpojumiem. Izstrādātā lielu datu apstrādes sistēma tiek iteratīvi uzlabota saskaņā ar jaunām prasībām, kas iegūtas no žurnāla datiem. Metode tiek pielietota veselības aprūpes jomā.

Metode paredz noskaidrot prasības, izmantojot īpašu, metodei specifisku paņēmienu. Prasības tiek definētas, balstoties uz pieredzi un iteratīvu izstrādi, atbilstoši lietotāju un ieinteresēto personu vajadzībām. Metode ir piemērota, lai attiecinātu prasību inženierijas

[15] https://miro.com/templates/kano-model

posmus lielo datu kontekstā. Metode paredz lietotāju iesaistīšanos datu apstrādē ar lielo datu tehnoloģijām, lai iegūtu jaunas prasības.

3.5 Lielo datu apstrādes metodes, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmiem "prasību iegūšana" un "prasību specificēšana"

Šajā nodaļā tiek aplūkotas lielo datu apstrādes metodes, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmiem "prasību iegūšana" un "prasību specificēšana".

Metodē "Towards a requirements engineering artefact model in the context of big data software development projects: Research in progress" (Arruda & Madhavji, 2017) tiek apgalvots, ka pastāv prasību inženierijas modeļu trūkums, kas atvieglotu prasību inženierijas procesu un veicinātu vienotu izpratni par lielo datu programmatūras projektiem. Rakstā (Arruda & Madhavji, 2017) tiek aplūkots prasību iegūšanas un prasību specificēšanas posms. Tiek piedāvāts speciāls prasību inženierijas modelis un metode (*BD-REAM*). Prasību inženierijas modeļa izveide sastāv no četriem soļiem:

- Elementu un konceptu noteikšana;
- Savstarpējo saistību noteikšana;
- Modeļa kardinalitātes noteikšana;
- Kopējā modeļa komplektācija.

Iegūtais modelis sastāv no 22 objektiem un saistībām starp šiem objektiem. Objekti sastāv ar prasību inženieriju saistītiem objektiem (piemēram, datu avots, datu transformācijas, datu kvalitātes prasības), kā arī no objektiem, kas ir saistīti ar lielo datu tehnoloģijas un scenāriju prasībām. Šobrīd nepastāv automatizēts paņēmiens, kā lielo datu scenārijus pārveidot formālās prasībās, tas nozīmē, ka analītiķiem tas ir jādara manuāli. Metodes piemērs aplūko gadījumu par projektu, kas saistīts ar finanšu jomu.

Metode paredz noskaidrot prasības, izvirzot mērķus un aprakstot scenārijus. Prasības, ierobežojumi un scenāriji tiek noskaidroti intervijās. Tiek veidots speciāls prasību modelis. Prasības un scenāriji tiek grupēti atbilstoši klasēm speciālajā prasību modelī. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā.

Metodē "Modeling the requirements for big data application using goal oriented approach" (Eridaputra et al., 2014) tiek piedāvāts izmantot vispārīgu prasību modeli, lai modelētu prasības lielo datu projektiem. Metode par pamatu izmanto pastāvošos "*i* Framework*" un "*Knowledge Acquisition autOmated Specification (KAOS)*" modeļus. Modeļi tiek papildināti ar lielo datu tehnoloģiju specifiskām klasēm. i* modelis palīdz organizēt atkarības starp lomām. KAOS modelis atbalsta vairākas paradigmas, tas atļauj kombinēt,

piemēram, daļēji strukturētus un strukturētus mērķus. KAOS modelis sastāv no četriem modeļiem – mērķu, atbildības, objektu un operacionālā. Kopējās lielo datu programmatūras prasības ir definētas, balstoties uz lielo datu 4Vs īpašībām un izaicinājumiem:

- Liels datu apjoms;
- Apmierinoša datubāzes ātrdarbība;
- Datu kvalitāte un struktūra;
- Datu privātums un drošība.

Tiek paredzēta lielo datu programmatūras izstrāde, izmantojot i* modelēšanu, iesaistot trīs lomas:

- Lietotājs;
- Programmatūra;
- Programmatūras administrators.

Šīs lomas tiek attēlotas divos modeļos:

- Stratēģisko atkarību modelī;
- Stratēģisko motīvu modelī.

Stratēģiskais motīvu modelis ir detalizētāka versija par stratēģisko atkarību modeli, iekļaujot konkrētus uzdevumus. Modelējot lielo datu programmatūru, izmantojot KAOS modeļus, tiek rekomendēts katru mērķi iegūt, uzdodot jautājumu "kāpēc?" un "kā?". Metode ir pielieta lielo datu izstrādes projektā saistībā ar valsts pārvaldi, kā rezultātā tika iegūtas 26 funkcionālās prasības un 10 nefunkcionālās prasības.

Metode paredz noskaidrot prasības, izvirzot mērķus. Prasības tiek iegūtas, veidojot speciālus mērķu modeļus (*i** un *KAOS*). Metode ir daļēji piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Prasības priekš lielo datu programmatūras tiek modelētās kā vāji mērķi (*soft goals*).

Metodē "Data collect requirements model" (Tikito & Souissi, 2017) tiek apgalvots, ka lielo datu tehnoloģijas nevar tikt interpretētas kā tradicionālas informācijas sistēmas. Lielo datu programmatūra būtu jāuztver kā sistēma no sistēmām. Metode nosaka, veidot lielo datu modeli no iegūtajiem datiem trīs posmos:

- Iegūt lietotāja prasības un prasības, kas saistītas ar lielo datu tehnoloģijas ierobežojumiem;
- Iegūt kritērijus katrai prasībai;
- Modelēt prasības, veidojot detalizētu aprakstu par to, kā prasības ir saistītas ar lielo datu definīciju.

Metodes(Tikito & Souissi, 2017) aprakstā pastāv neliels apraksts par metodes praktisku pielietojumu. Tiek minēts, ka metode ir izmantota apstrādājot mikro čipu datus. Metodes(Tikito & Souissi, 2017) autori apgalvo, ka uz lielo datu risinājumu nevar lūkoties kā uz tipisku sistēmu, tā vietā uz to būtu jāraugās kā uz sistēmu no sistēmām (*SoS - System of Systems*). Tiek noteikts SoS tehnoloģiju pilnveidot atbilstoši lielo datu specifikai. Metodē tiek paredzēts veidot modeli no visiem saņemtajiem datiem, šo uzdevumu tiek paredzēts veikt trīs posmos:

- Prasību identificēšana no diviem aspektiem, proti, lietotāja prasības un prasības, kas saistītas ar lielo datu ierobežojumiem;
- Definēt katrai prasībai kritērijus, kam par pamatu kalpo pirmajā posmā iegūtā informācija;
- Prasību modelēšana - tiek definēts, kā prasības mijiedarbojas lielo datu kontekstā.

Metode paredz noskaidrot prasības, veidojot prasību modeļus. Prasības tiek modelētas, par pamatu izmantojot lietotāju prasības un lielo datu īpašības "*7V*". Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metodē tiek aprakstītas lielajiem datiem raksturīgas prasības.

3.6 Lielo datu apstrādes metode, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmu "prasību specificēšana"

Metode "A framework for data-driven automata design" (Zhang et al., 2015) paredz projektēt programmatūru, balstoties uz prasībām, kas iegūtas no lietotājiem un apraksta sistēmas darbību. Prasības tiek analizētas ar mērķi noskaidrot sistēmas specifikāciju. Autori apskata tikai tāda veida datus, kas ir nepieciešami sistēmas funkciju izpildei un to modeļus. Metode ir pielietota, lai izveidotu programmatūru telpiskai sistēmai.

Metode paredz noskaidrot prasības, izmantojot īpašu, metodei specifisku paņēmienu.

3.7 Lielo datu apstrādes metode, kuru aktivitātes ir attiecināmas uz prasību inženierijas posmu "prasību validēšana"

Metode "BUDGET: A tool for supporting software architecture traceability research" (Santos et al., 2016) apraksta tehnoloģijas, kuras balstās uz mašīnmācīšanās algoritmiem. Šie algoritmi atrod saistības starp prasībām, arhitektūru un pirmkodu. Autori nepiedāvā jaunu metodi, bet piedāvā risinājumu, kā ar rīkiem automatizēti var atrisināt datu kopas ģenerāciju ar iespēju izsekot arhitektūras pieeju. Šis risinājums darbojās par pamatu izmantojot tīmekļa

datus un lielo datu tehnoloģijas. Šis risinājums ir ieviests speciālā rīkā "BUDGET". Šis rīks atbalsta programmatūrā pielietotās arhitektūras un prasību inženierijās izpēti.

Metode paredz noskaidrot prasības, izmantojot īpašu, metodei specifisku paņēmienu. Lai iegūtu prasības, tiek izmantotas tīmekļa, datizraces metodes un lielo datu analīzes tehnoloģijas. Metode ir piemērota, lai attiecinātu prasību inženierijas posmus lielo datu kontekstā. Metode nosaka izmantot dokumentu indeksēšanas tehnoloģijas.

3.8 Secinājumi

Šajā nodaļā tika aplūkots lielo datu jēdziens, lielo datu avoti un pastāvošās lielo datu apstrādes metodes. Metodes tiek aplūkotas pēc vienotiem aspektiem saistībā ar prasību inženierijas iespējām datu noliktavām. Var secināt, ka lielo datu apstrādes metodes var tikt attiecinātas kādam no datu noliktavas prasību inženierijas posmiem, jo no 22 aplūkotajām metodēm, pētījumos (Kozmina et al., 2018) un (Kozmina et al., 2019) tika noskaidrots, ka 53% gadījumos pastāv "augsta" iespēja piemērot kādu no prasību inženierijas posmiem, 14% gadījumos pastāv "vidēja" iespēja, bet 33% gadījumos no metodes apraksta šādu informāciju nevarēja izsecināt. Šos vērtējumus izanalizēja pētījuma (Kozmina et al., 2018) un (Kozmina et al., 2019) autori, vērtējums "augsts" tika ielikts gadījumos, ja bija saskatāmas tiešas iespējas piemērot kādu no datu noliktavas prasību inženierijas posmiem, "vidējs" gadījumos, kad pastāv kāds šķērslis vai tas nav tieši izdarāms pielietojot attiecīgo metodi, bet "zems" vērtējums tika ielikts gadījumos, kad nevar saskatīt iespēju piemērot kādu no datu noliktavas prasību inženierijas posmiem.

No aplūkotajām lielo datu apstrādes metodēm, 16% gadījumos var attiecināt vienu no četriem prasību inženierijas posmiem ("prasību iegūšana", "prasību analīze", "prasību specifikācija" un "prasību validācija"), 46 % gadījumos - divus, bet visus četrus prasību inženierijas posmus - 38% gadījumos.

Aplūkojot, kuri prasību inženierijas posmi var tikt visvairāk pielietoti, tiek iegūti rezultāti, ka 30% gadījumos no aplūkotajām metodēm var attiecināt uz "prasību iegūšanas" vai "prasību specifikācijas" posmu, bet 20% procentos gadījumos var attiecināt uz "prasību analīzes" vai "prasību validācijas" soli.

Lielo datu apstrādes metodes izmanto lielo datu tehnoloģijas dažādiem mērķiem lai analizētu sistēmas lietotāju darbības (Fernandez-Garcia et al., 2015), brīvā teksta analīzei (Cheptsov et al., 2014) vai, lai uzlabotu lielo datu pārvaldību, datu analīzi un vizualizācijas risinājumu (Tardio et al., 2015), bet no aplūkotajām metodēm neviena metode nenosaka kā analizēt datus, lai definētu jaunas informācijas prasības.

Aplūkojot metodes, var secināt, ka ir liela nozīme prasību definēšanai, izstrādājot lielo datu risinājumu. Viens no iemesliem lielo datu projektu neveiksmēm ir, ka projekta sākumā nav pieejamas visas informācijas prasības vai lietotāju vēlmes ir atšķirīgas attiecībā pret datu kvalitāti, datu pieeju, u.c. Dažos gadījumos tas ir pretēji - projekta neveiksme var tikt saistīta ar to, ka netika veikts prasību analīzes solis pirms izstrādes.

No aplūkotajām metodēm, prasību analīze tiek veikta pirms izstrādes fāzes (Tardio et al., 2015), (Arruda & Madhavji, 2017), (Eridaputra et al., 2014), (Tikito & Souissi, 2017) vai pēc datu ielasīšanas no avota sistēmas (Santos & Costa, 2016), (Liu et al., 2015) un (Cheptsov et al., 2014), bet neviena no metodēm nenosaka veikt prasību analīzi projekta sākumā un pēc datu ielasīšanas no avota.

Var secināt, ka lielo datu apstrādes metodes var nebūt saistītas ar prasību inženieriju, bet, izmantojot lielo datu apstrādes tehnoloģijas, metodes darbības var tikt attiecinātas uz kādu no datu noliktavas prasību inženierijas posmu vai posmiem. Metodēs pielietotās tehnoloģijas var tikt izmantotas automātiskai vai daļēji automātiskai informācijas prasību iegūšanai, bet šobrīd neviena no pētījumos aplūkotajām metodēm nenodrošina šādu iespēju. Nākamajā nodaļā tiek analizēta iespēja izmantot nestrukturētus datus kā informācijas avotu un informācijas prasības, jo kā viens no lielo datu avotiem, ir nestrukturēti dati.

4 BRĪVAIS TEKSTS KĀ DATU AVOTS UN KPI KĀ INFORMĀCIJAS PRASĪBU VEIDS

Šajā nodaļā tiek aplūkoti temati par brīvo tekstu, brīvā teksta apstrādes metodēm un tipiskākajām problēmām, apstrādājot brīvo tekstu. Tiek aplūkotas iespējas kā brīvu tekstu izmantot par datu avotu, lai iegūtu datu noliktavas informācijas prasības. Otrs temats, kas tiek aplūkots šajā nodaļā, ir organizācijas KPI, kas parasti tiek formulēts teksta formātā. Tiek aplūkots KPI un potenciāls KPI izmatošanai, kā informācijas prasības. Šīs nodaļas rezultāti publicēti rakstā (Zemnickis et al., 2020).

4.1 Teksts kā informācijas avots

Sociālo platformu attīstība ir mainījusi brīvā teksta apstrādei pieejamos datu apjomus un datu tipus. Pieejamie dati no tādiem avotiem kā Twitter, Facebook, YouTube, tīmekļa forumiem, tīmekļa žurnāliem (*blog*), dod iespēju izpētīt saistības starp demogrāfisko informāciju, valodas lietojumu un sociālo mijiedarbību (Russell, 2013). Lai iegūtu datus, tiek izmantota tīmekļa skenēšanas (*WEB-scraping*) tehnika. Šādā veidā tiek iegūti liela apjoma nestrukturēti dati. Izmantojot statistikas un mašīnmācīšanās tehnoloģijas, no teksta var tikt iegūta demogrāfiskā informācija, piemēram, vecums un dzimums, iespējams veikt noskaņojuma analīzi, izsekot aktuālajām tēmām, noskaidrot lietotāju viedokli, uzskatus par produktiem, politiķiem, prognozēt slimību izplatību (Elhadad et al., 2014).

Laikmetā, kad lielo datu tehnoloģijas ir plaši izmantotas un ir plaši pieejami sociālo platformu dati, ir mainījies veids, kā reklāmdevēji, žurnālisti, uzņēmumi, politiķi un mediķi iegūst un praktiski pielieto datus (Hirschberg & Manning, 2015). Sociālo platformu dati un tīmekļa forumi satur sarunvalodu – brīvo tekstu. Piemēram, izmantojot datus no politiskajām debatēm, ir iespējams prognozēt vēlēšanu rezultātus un apelācijas. Sociālo platformu dati var tikt izmantoti, lai noskaidrotu indikatorus, kas ietekmē un provocē noteiktas sabiedrības grupas. Publiski pieejamie medicīnas forumu dati var tikt analizēti, lai noskaidrotu biežāk uzdotos jautājumus, noskaidrotu maldīgos priekštatus un diagnosticētu slimības.

4.1.1 Dabīgās valodas apstrāde (NLP)

Pēdējo 50 gadu laikā ir bijuši daudzi mēģinājumi analizēt dabīgo valodu ar datora palīdzību, bet senāk runas un valodas izpratnes programmatūra nedarbojās pietiekami labi, lai to lietotu lietojumprogrammās. Situācija ir strauji mainījusies kopš 2010. gada, kad parādījās ievērojami uzlabojumi runas atpazīšanā, izmantojot datus no mobilajiem tālruņiem (Hirschberg & Manning, 2015). NLP ir zinātnes un programmatūras joma, kas pēta, kā datoram saprast dabiskās valodas tekstu vai runu, lai to varētu lietderīgi izmantot (Chowdhary

& Chowdhary, 2020). NLP pēta vairākās nozarēs, piemēram, datorzinātnēs, valodniecībā, matemātikā, elektronikas inženierijas, mākslīgā intelekta, robotikas, psiholoģijas un citās nozarēs. NLP programmatūra iekauj dažādas tehnoloģijas un risinājumus - mašīntulkošanas (*machine translation*), lietotāja saskarnes, daudzvalodīgums (*multilingual*) un starpvalodas informācijas ieguvi (*CLIR - multilingual and cross language information retrieval*), runas atpazīšanas, mākslīgā intelekta un citas tehnoloģijas. Lai izstrādātu datorprogrammu, kas saprot dabīgo valodu, nākas saskarties ar trīs galvenajām problēmām (Chowdhary & Chowdhary, 2020):

- Izprast domāšanas procesu (saprast domu);
- Valodas reprezentācijas un valodas nozīme (saprast teikumu);
- Vārdu nozīme.

Autori (Liddy, 1998) un (Feldman, 1999) piedāvā septiņus neatkarīgus līmeņus, lai saprastu dabīgo valodu:

- Fonētiskais vai fonoloģiskais līmenis, kas attiecas uz izrunu;
- Morfoloģiskais līmenis, kas attiecas uz mazākajām vārdu daļām - sufiksi un prefiksi;
- Leksiskais līmenis, kas atpazīst ar vārdu un vārdšķiras nozīmi;
- Sintaktiskais līmenis, kas atpazīst ar teikumu gramatiku un struktūru;
- Semantiskais līmenis, kas atpazīst ar vārdu un teikumu nozīmi;
- Diskursa līmenis, kas aplūko dažāda veida teksta struktūras, izmantojot dokumentu struktūru;
- Pragmatisks līmenis (*pragmatic level*), kas veido zināšanas un informāciju, kas iegūta no ārpus analizējamā dokumenta satura.

4.1.2 Dabīgās valodas apstrādes (NLP) sistēmas

NLP sistēmas parasti sāk apstrādi ar vārdu nozīmi – morfoloģiskās struktūras noteikšanu, vārdšķiru, nozīmi, u.c. Turpinot ar teikuma nozīmi – nosakot vārdu secību, gramatiku un teikuma nozīmi u.c. Visbeidzot, nosakot kopējo nozīmi visam domēnam. Vārdam vai teikumam var būt specifiska nozīme un tas var būt saistīts ar citiem vārdiem teikumā vai domēnā. NLP sistēmas nosaka veikt manipulācijas ar tekstu, lai iegūtu zināšanas, veiktu automātisku indeksēšanu, sagatavotu tekstu vēlamajā formātā. Šīs manipulācijas ar tekstu ir nozīmīga NLP pētniecības joma. NLP sistēmas nosaka apstrādāt liela apjoma tekstus, lai iegūtu noteiktu informāciju noteiktā struktūrā, ko var izmantot tālākiem mērķiem. Automātiskās teksta apstrādes sistēmas parasti izmanto noteiktu teksta ievades formu, bet

atgriež informāciju citā formā. Galvenais NLP sistēmu uzdevums ir izskaidrot potenciāli neskaidro dabīgās valodas tekstu tādā formātā, ko var viegli un nepārprotami saprast (Liddy, 1998). NLP sistēmas bieži sākās ar morfoloģisko analīzi. Pēc tam seko dokumenta vārdu un terminu atdalīšana, lai iegūtu visu iesaistīto vārdu morfoloģiskos variantus. Leksikas un sintaktiskā apstrāde ietver leksikonu izmantošanu vārda īpašību noteikšanai, vārdšķiras, vārdu un frāžu noteikšanai, kā arī, lai noteiktu vārdu saistību teikumā.

Pastāv NLP sistēmas, kas ir būvētas, lai apstrādātu tekstu, izmantojot domēnam specifisku valodu (*sublanguages*). Tas tiek darīts, lai samazinātu darbību skaitu un sistēmas sarežģītību (Chowdhary & Chowdhary, 2020). Šie domēna specifiskie pētījumi ir pazīstami kā "domēnam specifiskas valodas analīze" ("*sublanguage analyses"*) (Grishman & Kittredge, 2014). Šie pētījumi parasti ir ierobežoti konkrētā jomā, piemēram, medicīnas jomā vai, piemēram, viena veida dokumentiem.

4.1.3 Nestrukturēto datu potenciāls

Dabīgā valoda ir viens no nestrukturētu datu veidiem. Šajā nodaļā tiek aplūkots nestrukturētu datu potenciāls. Nestrukturētu datu analīze ir sarežģītāka salīdzinot ar strukturētu datu analīzi (Howatson, 2016). Nestrukturētu datu analīzē ir vēl daudz neatklātas iespējas (Bach et al., 2019), kā arī ir analīzes un datu apstrādes posmi, ko ir iespējams automatizēt. Pētījumi un prakse rāda, ka informācija no nestrukturētiem datiem var būt noderīga, lai pieņemtu lēmumus (Bach et al., 2019).

Autori (Danaher et al., 2015) apgalvo, ka organizācijas analizē tikai 12% no pieejamajiem nestrukturētajiem datiem. Pētījums (Rizkallah, 2017) rāda, ka 80% dati organizācijās ir nestrukturēti. 87% no biznesa pārstāvjiem saprot, ka cieš zaudējumus, jo nespēj pilnībā izmantot vērtību no nestrukturētajiem datiem (Howatson, 2016). Problēmas ar ko saskarās organizācijas, lai iegūtu vērtību no lielajiem datiem, ir saistītas ar regulāriem lieliem ieguldījumiem un datu analīzes problēmām (WEB, a). Pētījums (Rizkallah, 2017) rāda, ka 80% no organizācijām nav zināms vai ir nelielas zināšanas par tās nestrukturētajiem datiem. 95% no organizācijām sagādā grūtības efektīvi analizēt nestrukturētus datus (Baviskar et al., 2021).

Pastāv daudz neatklātas iespējas nestrukturētu datu analīzē (Bach et al., 2019). Lielākā daļa nestrukturētu un daļēji strukturētu datu potenciāls nav izmantots organizācijās (Hai, 2021). Organizācijas sāk orientēt savu darbību vērstu uz datiem (*data-driven*) (Bode et al., 2023). No tā var secināt, ka organizācijas apzinās, ka pastāv daudz neapstrādātu nestrukturētu datu un tie var sniegt vērtīgu ieguvumu organizācijas attīstībai. Tai pašā laikā,

organizācijās pietrūkst zināšanas un kompetence, lai apstrādātu nestrukturētus datus. Projekti tiek izstrādāti, nesasniedzot mērķus vai pārsniedzot projekta izmaksas.

4.2 KPI kā informācijas prasību veids

KPI parasti tiek formulēti brīvā tekstā. KPI nosaka, kādi uzdevumi ir jāveic, lai ievērojami palielinātu organizācijas veiktspēju (Parmenter, 2015). Lai uzraudzītu organizācijas stāvokli un atbilstību izvirzītajiem KPI, organizācijām ir jābūt pieejamai informācijai par pastāvošo stāvokli organizācijā. Šajā nodaļā tiek aplūkoti KPI, to īpašības un to potenciāls kā informācijas prasības datu noliktavas kontekstā.

4.2.1 KPI - galvenie izpildes pamatrādītāji

Autors (Parmenter, 2007) apraksta, ka *veiksmes faktori* (*success factors*) ir industrijā definēti aspekti, kas nosaka organizācijas veiktspēju, bet *kritiskie veiksmes faktori* (*critical success factors*) ir tie aspekti vai problēmas, kas raksturo organizācijas pamatstāvokli, respektīvi pašus svarīgākos organizācijas darbības faktorus. Ar *veiktspējas mērījumiem* (*performance measures*) tiek saprasti mērījumi, kas raksturo veiktspēju organizācijā. *Veiktspējas mērījumi* organizācijā tiek noteikti kā *rezultējošie indikatori* (*RI - result indicators*) vai *veiktspējas indikatori* (*PI – performance indicators*). Apzīmējums "*KPI*" ir cēlies no PI, tikai papildus ir pievienots vārds "Key". Galvenais izpildes pamatrādītājs (*KPI* – "*key performance indicator"*) nosaka, kādi uzdevumi ir jāveic, lai ievērojami palielinātu veiktspēju (Parmenter, 2015). KPI koncentrējas uz svarīgākajiem aspektiem, kas var uzlabot organizācijas šī brīža vai nākotnes veiktspēju. Viena no KPI īpašībām ir tā noteiktība (*hardness*). KPI var būt noteikts vai nenoteikts (*hard/soft*) (Domínguez et al., 2019). Ja KPI ir nenoteikts, tad tas nav tieši mērāms, tas ir kvalitatīvs (*qualitative*), piemēram, organizācijas vai darbinieka reputācija. Gadījumos, kad KPI ir noteikts tas var tikt nomērīts, šādi KPI ir kvantitatīvi (*quantitative*), piemēram, pārdoto preču skaits.

4.2.2 KPI kā informācijas prasību veids

Iepriekšējā nodaļā 4.2.1. tika noskaidrots, ka KPI nosaka darbības, kuru rezultātā ievērojami var tikt paaugstināta organizācijas veiktspēja. No tā izriet, ka organizācijām ir svarīgi veikt galveno izpildes rādītāju novērošanu. Lai būtu iespējams veikt KPI novērošanu ir:

- Jādefinē organizācijas darbībai atbilstoši KPI un to mērķu vērtības;

- Jābūt pieejamiem datiem par organizācijas veiktspēju, piemēram, "pārdoto preču skaits pa mēnešiem";
- Programmatūra, kas nodrošina KPI novērošanu un salīdzināšanu pret mērķa vērtībām.

Organizācijas veiktspējas rādītāji ir dati par organizācijā notiekošajiem procesiem un šie dati būtu jāievāc par ilgāku periodu, lai varētu novērot tendenci, kā organizācijas darbība attīstās vai, tieši pretēji, pasliktinās. Šos datus ir nepieciešams ievākt un saglabāt kādā informācijas sistēmā, piemēram, datu noliktavā. Var secināt, ka KPI novērošanai nepieciešamie dati var kalpot kā informācijas prasības informācijas sistēmai, kurā šie dati, kas nepieciešami KPI aprēķināšanai un analīzei, tiks glabāti. Līdzīgi kā pētījumā (Niedritis et al., 2011), KPI tiek interpretēts kā informācijas prasība. Pētījumā (Niedritis et al., 2011) tiek aplūkoti tipiski KPI un, balstoties uz KPI formulējumiem, predefinētā veidā tiek veidots metamodelis informācijas prasībām.

4.2.3 KPI definēšanas aspekti apstrādājot nestrukturētu datus

Rakstā (Domínguez et al., 2019) tiek aplūkotas KPI īpašības no pieciem aspektiem – "Kas tiek mērīts?", "Kādas KPI īpašības un specifika tiek sagaidīta?", "Kādam nolūkam KPI tiek mērīts?", "Kādi artefakti tiek izmantoti, lai izstrādātu un specificētu KPI?" un "Kādas ir KPI pārvaldības īpašības?". Divi aspekti var palīdzēt iegūt KPI no nestrukturēta teksta, tie ir:

- "Kas tiek mērīts?", šis aspekts nosaka, kas tiek mērīts ar KPI. Tiek paredzēts noskaidrot KPI īpašības, kā "*veiktspējas mērījumu iespējas*" ("*Performance measurement perspectives*") un darbības *jomu*. Zināmākais ietvars, kas nosaka veiktspējas mērījumus ir "Līdzsvarotā vadības karte" ("*Balanced Scorecards"*) (Kaplan, 1992). Šis ietvars nosaka četras mērījuma iespējas: finanšu, lietotāju, iekšējo procesu, izaugsmes un mācības. KPI *jomas* aspekts var būt ļoti atšķirīgs, piemēram, "restorānu bizness", "banku sektors", "tūrisms" u.c.;
- "Kādas KPI īpašības un specifika tiek sagaidīta?", šis aspekts nosaka KPI specifiku un aprēķinu īpašības. Aprēķinu īpašības nosaka, kā pareizi varētu noteikt KPI mērķa vērtības. Piemēram, KPI noteiktība (*hardness*) ir atkarīga no KPI būtības, tas ir, mērķi (skaitlis, vērtība) ir tieši mērāmi, bet objekti (raksturojums par objektu) nav. *Aprēķinu noteikumi* (*calculation rule*) definē formulu, bet *vērtības tips* nosaka datu tipu KPI mērķa vērtībām un mērvienību. *Filtrs* nosaka ierobežojumus KPI aprēķinam, piemēram, laika periods. *Biežums* nosaka intervālu starp aprēķiniem. *Mērķis* nosaka sasniedzamo mērķa vērtību. *Avots* savieno ar nepieciešamajiem datiem, lai aprēķinātu KPI.

4.2.4 KPI definēšana restorānu nozarē

Šajā nodaļā tiek aplūkoti KPI specifiska konkrētai nozarei – restorāni. Avotā (WEB, f) KPI tiek grupēti sešās lielās grupās:

- Ieņēmumu - mērāms lielums naudas izteiksmē, piemēram, ieņēmumi par galdiņu;
- Noslogojums – tiek noteikts procentos vai kā skaitlis, piemēram, atcelto galdiņu skaits;
- Lietotāju atsauksmes – tiek noteikts procentos, piemēram pozitīvo atsauksmju īpatsvars;
- Apkalpošana – tiek noteikts procentos, piemēram, nepieejamo ēdienu skaits no ēdienkartes;
- Kvalitātes atbilstība – tiek noteikts procentos, piemēram ēdienu kvalitāte;
- Izmaksu pārvaldība – tiek noteikts procentos, pārtikas vai dzērienu zudums.

Pētījumā (Kim & Chungm 2011) tiek veikta analīze par kritērijiem, kas piesaista un palīdz izvēlēties restorānu. Tiek aplūkoti kritēriji, kā saglabāt esošos klientus. Par pamatu šai analīzei tika izmantotas "BIGresearch's CIA"[16] (*Consumer Intentions and Actions*) aptaujas no 8127 klientiem no dažādām ekonomiskajām grupām. Aplūkojot klientu atbildes, kas motivē klientus apmeklēt konkrētu restorānu, tiek noteikti šādi kritēriji – "reklāma", "tīrība", "ātrs serviss", "veselīga ēdiena izvēles iespēja", "izvēle bērniem", "atrašanās vieta", "ēdienkartes izvēles iespējas", "vēls darba laiks", "porcijas lielums", "cena", "ēdiena kvalitāte", "izdevīgi piedāvājumi", "reputācija", "apkalpošana", "draudzīgs apkalpojošais personāls", "ēdiena līdzi paņemšanas iespēja" un "ēdienkartes īpašie piedāvājumi/atlaides". Šie ir kritēriji, kas ir savākti, lai klients izvēlētos konkrētu restorānu. No šiem kritērijiem ir jādefinē jauni KPI. Piemēram, no kritērija "ātrs serviss", tiek definēts jauns kvantitatīvais KPI "Apkalpošanas ilgums minūtēs pīķa stundās" vai no kritērija "tīrība" tiek definēts jauns kvalitatīvais KPI "Galdu/grīdas/tualetes tīrība."

4.3 Metodes, kas saitītas ar KPI noteikšanu vai informācijas prasību ieguvi datu noliktavai no brīvā teksta

Metodē "Exploring tourist dining preferences based on restaurant reviews" (Vu et al., 2019) kā datu avotu izmanto interneta vietnē TripAdvisor pieejamos datus par restorānu

[16] https://www.hometextilestoday.com/industry-news/bigresearch-consumer-confidence-flagging

apmeklētāju atsauksmēm. Autori pielieto teksta ieguves (*text mining*) tehnoloģijas, lai apstrādātu klientu atsauksmes. Pēc priekšapstrādes ar tādiem paņēmieniem, kā teksta dalīšana tekstvienībās (*tokenization*), atdalīšana (*stemming*), vārdšķiras (*Part of Speech*) noteikšana, tiek saskaitīti un iegūti populārākie vārdi. Autori (Vu et al., 2019) veica:

- Datu izpēti, lai noteiktu ēdienu popularitāti starp dažādām vecuma grupām;
- Analīzi par ēdienu izvēlēm, restorāna kopējām funkcijām, piemēram, pakalpojumu kvalitāti un apmeklētāju noskaņojumu.

Metodē "A cloud-based tool for sentiment analysis in reviews about restaurants on TripAdvisor" (Agüero-Torales et al., 2019) tiek izstrādāts speciāls rīks, lai apstrādātu klientu atsauksmes par restorānu jomu. Rīks izmato tādas teksta apstrādes tehnaloģijas kā teksta ieguve, noskaņojuma analīze (*sentiment analysis*), teksta parsēšanu (*parsing*) un teksta dalīšanu tekstvienībās. Rīks nodrošina atsauksmju analīzi no interneta vietnes TripAdvisor un ir paredzēts to galvenokārt izmantot restorānu jomā. Metodē lietotāju novērtējumi un atsauksmes tiek izmantotas kā teksta avots. Noskaņojuma analīze tiek veikta, izmantojot vārdu leksiku un emocijas. Gala novērtējums ir kombinēts no noskaņojuma analīzes un lietotāja atsauksmēm.

Metodē "A hybrid AI tool to extract key performance indicators from financial reports for benchmarking" (Brito et al., 2019) tiek prezentēts rīks automātiskai KPI ieguvei no publiski pieejamām finanšu atskaitēm. Rīks nodrošina uzraudzību par organizācijas tīmekļa vietnēm un finanšu atskaitēm. Pēc finanšu atskaišu teksta apstrādes tiek pielietoti konvolūciju neironu tīkli (*convolutional neural networks*), lai noteiktu KPI. Iegūtie KPI un to mērķa vērtības tiek saglabātas datubāzē vēlākai izmantošanai. Apstrādes rezultātā no atskaitēm tiek iegūti KPI un to mērķa vērtības. Tomēr šīs analīzes rezultātā tiek iegūti KPI, kas aptver tikai daļu no iespējamajiem KPI.

Metode "Cultural heritage and social pulse: a semantic approach for CH sensitivity discovery in social media data" (Chianese et al. 2015) par kultūras mantojumu nosaka pielietot datu vadītu pieeju, kas ir integrēta ar ontoloģiju pielietojumu. Sākumā tika iegūti Twitter dati no četrām Indijas pilsētām, kas satur tēmturi "*#culturalheritage*". Twitter teksts tiek apstrādāts ar noskaņojuma analīzes tehniku. Iegūtie termini un jēdzieni no Twitter teksta tiek sasaistīti ar iepriekš definētām kategorijām, piemēram, "skulptūras" vai "muzeji". Nākamais solis paredz saskatīt kategoriju biežumu tvītos. Noslēdzošais solis paredz izmantot tvīta ģeogrāfiskos datus un izveidošanas datumu, lai izveidotu korelāciju starp konkrētiem kultūras pasākumiem, to norises ilgumu un kultūras pasākuma izraisīto emocionalitāti. Galvenais mērķis ir nodrošināt biznesa analīzes servisiem datus par klientu apmierinātību, izmantojot sociālo tīklu datus par konkrētu notikumu vai apskates objektu. Šos datus vēlāk izmanto KPI mērķa vērtību

novērtēšanā. Atšķirība starp autoru (Zemnickis et al., 2020) un (Chianese et al. 2015) metodēm ir, ka metodē (Chianese et al. 2015) KPI ir definēti iepriekš un iegūtie dati tiek izmantoti, lai noteiktu vai sasniegta KPI mērķa vērtība, bet promocijas darbā piedāvātā metode ļauj meklēt jaunus KPI, izmantojot klientu atsauksmes.

Pastāv metodes, kuras paredz analizēt tekstu un sociālo mēdiju datus, bet ne visas metodes paredz izmantot NLP rīkus un tehnoloģijas. Pārskata rakstā (Farzindar et al., 2015) apskata eksistējošas metodes, kas izmanto NLP rīkus, lai analizētu pieejamo informāciju no sociālajiem mēdijiem. Tiek konstatēts, ka metodēm pietrūkst eksperimentālā daļa. Pētījumā (Pinto et al., 2016) tiek aplūkotas tehnikas, lai analizētu Twitter datus, izmantojot NLP rīkus. Pielietoto tehniku rezultāti tiek apspriesti un salīdzināti, bet pietrūkst paskaidrojums, kā dati varētu tikt izmantoti organizācijas analīzes vajadzībām, lai uzlabotu organizācijas veiktspēju. Pētījumā (Denecke, 2014) tiek aplūkoti medicīnas nozarei specifiski datu analīzes rīki, lai analizētu medicīnas datus, bet nav skaidrs, kā šie rīki darbotos citās nozarēs.

4.4 Secinājumi

Šajā nodaļā tika aplūkots brīvā teksta jēdziens, brīvā teksta apstrādes tehnikas, tipiskās problēmas un teksta potenciāls kā informācijas avots, kā arī KPI un to īpašības, potenciāls izmantošanā kā informācijas prasības.

Aplūkojot literatūru par NLP, var novērot, ka brīvais teksts var saturēt bagātīgu informāciju (Bach et al., 2019), piemēram, analizējot sociālo platformu datus vai tīmekļa forumu datus. Analizējot brīvo tekstu, ir iespējams prognozēt, piemēram, vēlēšanu rezultātus vai noskaidrot dažādu tematu ietekmi uz dažādām sabiedrības grupām (Hirschberg & Manning, 2015). Organizācijas sāk orientēt savu darbību vērstu uz datiem (*data-driven*) (Bode et al., 2023). Var secināt, ka brīvā valoda var tikt izmantota kā informācijas prasības, kas nosaka kādi dati ir jāiegūst no brīvā teksta.

Līdzīgi brīvais teksts arī KPI var tikt izmantots kā informācijas prasības, jo KPI nosaka darbības, kuru rezultātā ievērojami var tikt paaugstināta organizācijas veiktspēja, bet, lai būtu iespējams veikt KPI novērošanu, kā viens no priekšnosacījumiem, ir jābūt pieejamiem datiem par organizācijas veiktspēju, piemēram, "pārdoto preču skaits pa mēnešiem", tas nozīmē, ka šādai informācijai jābūt pieejamai organizācijā, piemēram, datu noliktavā. Līdzīgi kā pētījumā (Niedritis et al., 2011), KPI var tikt interpretēts kā informācijas prasība. Var secināt, ka KPI novērošanai nepieciešamie dati var kalpot kā informācijas prasības organizācijas informācijas sistēmām, tostarp, datu noliktavām.

Aplūkojot pastāvošās metodes, tiek noskaidrots, ka pastāv metodes, kas paredz analizēt brīvo tekstu, piemēram, lai iegūtu informāciju par restorānu apmeklētāju atsauksmēm, lai

noskaidrotu klientu vēlmes un apmierinātību (Vu et al., 2019) un (Agüero-Torales et al., 2019). Pastāv metodes, kas paredz analizēt datus, lai definētu KPI vai lai noskaidrotu KPI vērtības. Metode (Brito et al., 2019) nosaka kā analizēt finanšu atskaites (daļēji strukturētus datus), lai noteiktu kvantitatīvu KPI mērķa vērtības, vai metode (Chianese et al. 2015), kas paredz analizēt brīvo tekstu saistībā ar ģeogrāfiskajiem datiem, lai noskaidrotu vai ir sasniegtas KPI mērķa vērtības.

Var secināt, ka no aplūkotajām metodēm neviena metode nepiedāvā analizēt brīvu tekstu, lai definētu jaunus KPI, kuru atribūti tiek izmantoti kā informācijas prasības organizācijas informācijas sistēmai, piemēram, datu noliktavai. Nākamajā nodaļā tiek aprakstītas jaunas metodes, kas, izmantojot tekstu kā informācijas avotu, ļauj specificēt jaunus KPI un paplašināt datu noliktavas datu modeļus.

5 PIEDĀVĀTĀS METODES

Šajā nodaļā autors apraksta divas metodes, kas ļauj papildināt datu noliktavas datu modeli un noskaidrot jaunus KPI, analizējot nestrukturētu tekstu. Nodaļā aprakstītie rezultāti publicēti rakstos (Zemnickis et al., 2020) un (Zemnickis, 2023). Pirmā metode, kas aprakstīta nodaļā 5.1., nodrošina organizācijas lietotājam specificēt jaunus KPI un to mērķu vērtības. Lai specificētu KPI un to mērķu vērtības, tiek paredzēts izmantot vārdu saistību grafu, kas attēlo saistības starp vārdiem un to lietošanas biežumu. Otrā metode, kas aprakstīta nodaļā 5.2., nodrošina paplašināt organizācijā pastāvošas datu noliktavas datu modeli, kā arī daļēji automātiskā veidā ģenerēt kvantitatīvos KPI un to mērķu vērtības.

5.1 Metode KPI noteikšana izmantojot nestrukturētu tekstu

Šajā nodaļā tiek aprakstīta metode, kas nodrošina noskaidrot jaunus KPI, analizējot nestrukturētus datus – brīvo tekstu. Metode paredz apstrādāt brīvo tekstu, izmantojot kādu no dabīgās valodas apstrādes bibliotēkām, lai noskaidrotu vārdu vārdšķiras un vārdu saistības teikumā.

5.1.1 Dabīgās valodas apstrāde (NLP) izmantojot Stanford CoreNLP Toolkit

Rīkkopa Stanford CoreNLP (Manning et al., 2014) nodrošina izplatītākos NLP soļus, sākot no dalīšanas tekstvienībās līdz koreferenču noteikšanai. Rīkkopa ir izstrādāta, izmantojot programmēšanas valodu Java, kas to padara viegli integrējamu, ieskaitot tīmekļa bāzētās programmatūrās. Rīkkopa Stanford CoreNLP nodrošina teksta apstrādi dažādās valodās. Pēc noklusējuma rīkkopa ir nokonfigurēta apstrādāt angļu valodu. Promocijas darbā piedāvātā metode izmanto rīkkopu Stanford CoreNLP, izsaucot .jar failus brīdī, kad tiek izpildīts PHP kods.

Metodē tiek izmantots "Enhanced English Universal Dependencies" parsētājs (*parser*) (Schuster & Manning, 2016). Stanford modelis "depparse.model" nosaka teikuma vārdu saistības (WEB, k). Piemēram, teikums "Waited 30 min no service.", izmantojot tiešsaistē pieejamo rīku (WEB, c), tiek sadalīts vārdšķirās un iegūtas saistības starp vārdiem, kas redzams att. 5.1. Piemēram, vārdam "minutes" ir noteikta vārdšķira "NNS", bet piemēram, vārdi "waited" un "minutes" ir saistīti ar attiecību "dobj".

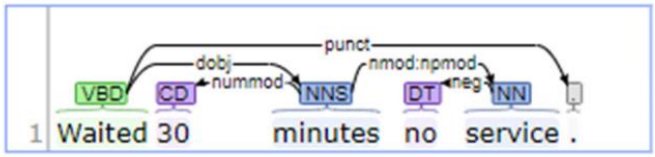
Vārdšķiru apzīmējumi ir aprakstīti (Toutanova et al., 2003). Att. 5.1 redzamie vārdu apzīmējumi nozīmē:

- NNS – Lietvārds (daudzskaitlī);
- VBD – Darbības vārds (pagātnē);
- NN – Lietvārds;
- DT – Noteicējs;

- CD - Skaitļa vārds.

Avotā (De Marnee & Manning, 2010) ir detalizēti aprakstīts Stanford modelis "depparse.model". Att. 5.1. redzamo vārdu saistību apzīmējumi nozīmē:

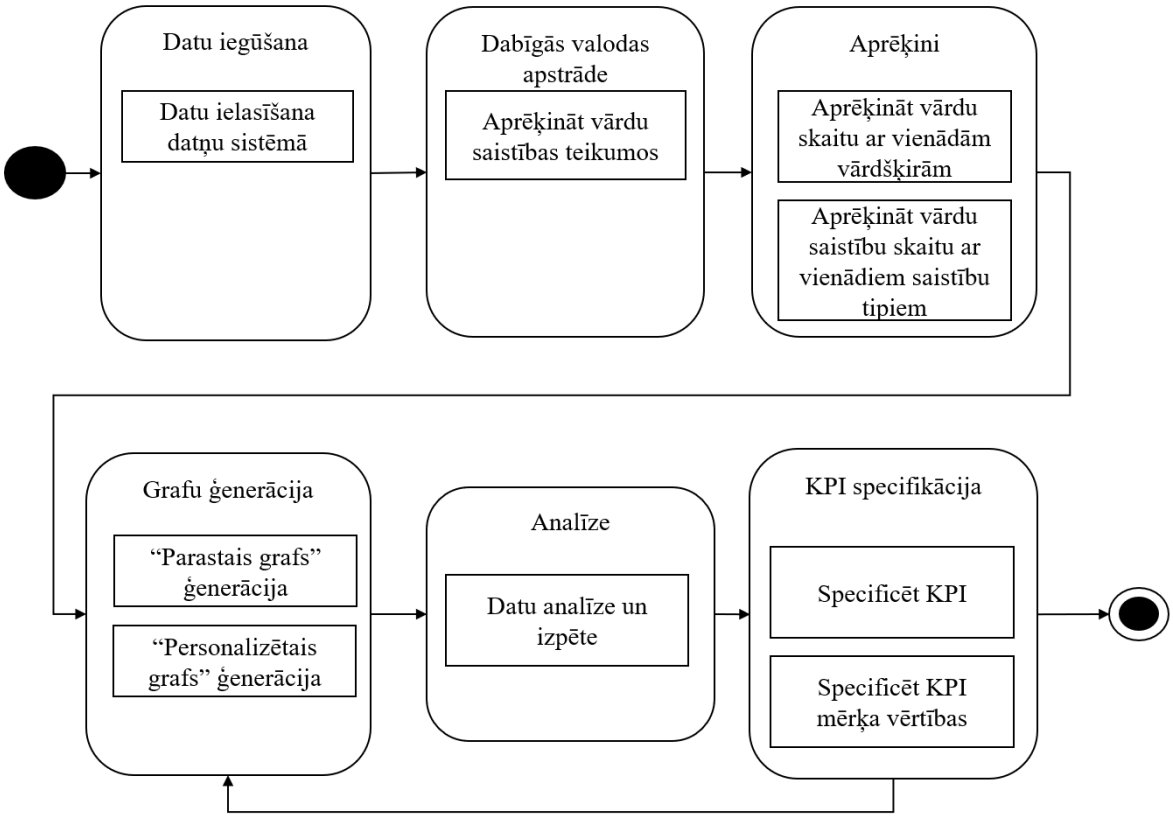
- nummod – Lietvārda skaitliskais pārveidotājs, piemēram, jebkurš vārds, kas kalpo lai piešķirtu skaitlisku nozīmi;
- dobj - darbības vārda tiešais objekts;
- neg – Negatīvais pārveidotājs.



Att. 5.1. Piemērs teikuma vārdu saistību noteikšanai izmantojot Stanford CoreNLP

5.1.2 Metodes apraksts un rīka arhitektūras risinājums

Metodes darbībai tiek izmantota viena no lielākajām publiski pieejamajām datubāzēm par lietotāju atsauksmēm, tā ir sociālā platforma Twitter. Att. 5.2, redzama metodes darbības diagramma.



Att. 5.2. Rīka galvenās komponentes

Dati no Twitter tika ielasīti, izmantojot tiešsaistes straumēšanas programmatūru Twitter API[17], solī "Datu iegūšana". Tiek ielasīti tvīti, kas satur atslēgas vārdus "restaurant", "food" vai "sushi". Šādi atslēgas vārdi tika izvēlēti, jo metodes darbība tika īstenota organizācijā, kura darbojās ēdināšanas jomā. Dati no Twitter tiek ielasīti datubāzē bez priekšapstrādes. Kopā no Twitter tika ielasīti 129046 tvītu laika periodā no 2019.12.19. līdz 2020.01.14. Šajā eksperimentā tiek saglabāta šāda informācija par konkrētu tvītu:

- Tweet_id – Twitter iekšēja unikāla suragātatslēga katram tvītam;
- User_id – Twitter lietotāja iekšēja unikāla suragātatslēga;
- Text – tvīta teksts;
- Created_at – Tvīta izveidošanas datums;
- Name – Twitter lietotāja lietotājvārds.

Nākamais solis "Dabīgās valodas apstrāde" nosaka tvīta tekstu apstrādāt ar bibliotēku "Stanford CoreNLP", kur tiek iegūtas vārdu saistības teikumos un vārdu vārdšķiras. Pēc apstrādes ar NLP tiek iegūti 2127989 vārdi. Katram teikuma vārdam tiek noskaidrota tā vārdšķira, tā saistītais vārds teikumā un šis saistības tips. Metodē tiek aprēķinātas agregētas vērtības, lai uzlabotu metodes nākamo soļu veiktspēju un samazinātu algoritma sarežģītību. Šīs vērtības tiek izmantotas, lai ģenerētu vārdu saistību grafus. Agregētas vērtības tiek aprēķinātas solī "Aprēķini". Šajā solī tiek veikti šādi aprēķini:

- Vārda un vārda tipa kopējais skaits, piemēram, "waiter NN 269", NN – lietvārds, skaitlis 269 norāda vārda "waiter" lietošanas biežumu tvītos;
- Vārda un vārda tipa, saistītā vārda un tipa un abu vārdu saistības kopējais skaits, piemēram, "*waiter* NN un *bringing* VBG 45 ACL", tas nozīmē, ka vārdu waiter ar tipu NN un vārdu bringing ar tipu VBG (darbības vārds vai tagadnes divdabis) un relācijas ar tipu ACL (lietvārda teikuma modifikators) kopējais skaits ir 45.

Izmantojot aprēķinātās vērtības, rīka lietotājs var būvēt sev interesējušo vārdu saistību grafu, turpmāk - "Parastais grafs". Lietotājam vispirms ir jāizvēlas konkrēti atslēgas vārdi. Piemēram, rīka lietotājam interesē, ko cilvēki izsakās par ēdienu "spaghetti". Šādā gadījumā lietotājam ir jāievada rīka izvēlnē atslēgas vārdu "spaghetti". Rīks grafā attēlo, ka pastāv saistība starp vārdiem "spaghetti" ar tipu "NNS", kas tvītos minēts 97 reizes un ar vārdu "real" ar tipu "JJ" (īpašības vārds), kas tvītos minēts 628 reizes, skatīt att. 5.3.

[17] https://developer.twitter.com/en/docs/twitter-api

Att. 5.3. Piemērs no "parastā grafa", kur redzama divu vārdu saistība

Uzklikšķinot uz relācijas tiek atvērti tvīti, kur pastāv šo vārdu relācija. Piemēram, šī relācija pastāvēja tvītā "Give me the real spaghetti please cause I ain't eating that dog food over there".

Grafā virsotnēs tiek attēloti vārdi tvītos un to vārdšķiras, bet grafa šķautnes ir vārdu savstarpējās saistības tvītos. Metodes solī "Analīze" lietotājam ir iespēja apskatīties kādu vārdu detalizēti – tas ir visus tvītus, kur ir minēts šis vārds ar šādu vārdšķiru. Lietotājs var atvērt šo detalizēto skatu uzklikšķinot uz kādas no virsotnēm.

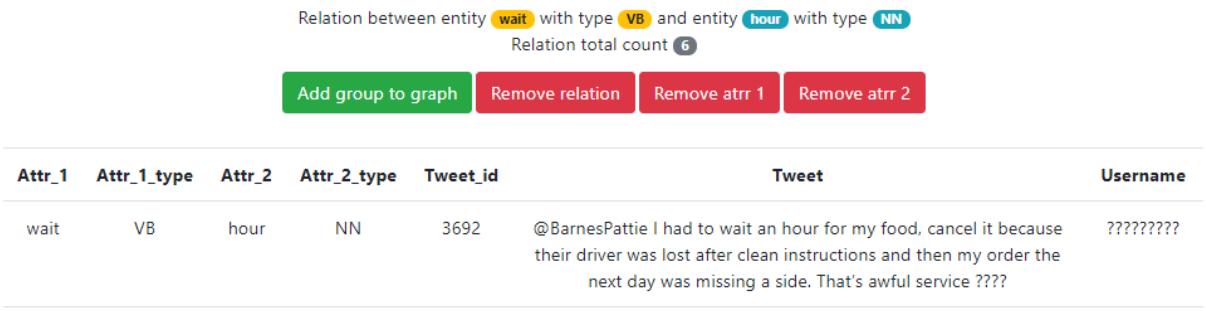
Entity long with type JJ Total count 372

Add to graph | Remove form graph

attr_1	attr_1_type	attr_2	attr_2_type	tweet_id	tweet	username
long	JJ	for	IN	113	RT @NzingaQ: I burnt my toddler's supper once last week, now every time I prepare food she keeps reminding me not to leave it for too long....	Zawadi

Att. 5.4. Piemērs vārda "*long*" detalizētajam skatam

Att. 5.4. redzams tvīts, kur ir minēts vārds "*long*". Lietotājam šo vārdu ir iespējams pievienot vai noņemt no "Personalizētais grafs". Tas ir izdarāms uzklikšķinot uz pogas "Add to graph" vai noņemt - uzklikšķinot uz pogas "Remove from graph". "Personalizētais grafs" ir vizuāli tāds pats grafs kā "Parastais grafs", bet šajā grafā tiek attēloti tikai tie elementi, ko lietotājs ir atzīmējis, piemēram, lai novērotu konkrētu elementu saistību grafā bez citiem elementiem. Līdzīgi kā konkrētu vārdu var atvērt aplūkošanai detalizētajā sarakstā, arī relāciju starp diviem vārdiem var šādi aplūkot. Tas ir izdarāms uzklikšķinot grafā uz šķautnes starp divām virsotnēm. Atveras relācijas detalizētais skats, skatīt att. 5.5.



Att. 5.5. Piemērs saistības starp vārdiem "*wait*" un "*hour*" detalizētajam skatam

Relācijas, tāpat kā vārdus, ir iespējams pievienot "Personalizētajam grafam". Piemēram, ja tiktu pievienots vārds "long" un relācija "wait" ar "hour" personalizētajam grafam, tad šis personalizētais grafs sastāvētu tikai no šiem trīs vārdiem un vārdi "wait" un "hour" būtu savienoti.

Metodes beidzamais solis ir "KPI specifikācija". Šajā solī tiek noteikts, ka metodes lietotājs, izmantojot analīzes posmā iegūto informāciju specificē jaunus KPI un to mērķu vērtības. KPI specificēšana var notikt iteratīvi, par metodes lietotāja interesējošajiem atslēgas vārdiem.

5.1.3 Implementācijas tehniskais risinājums

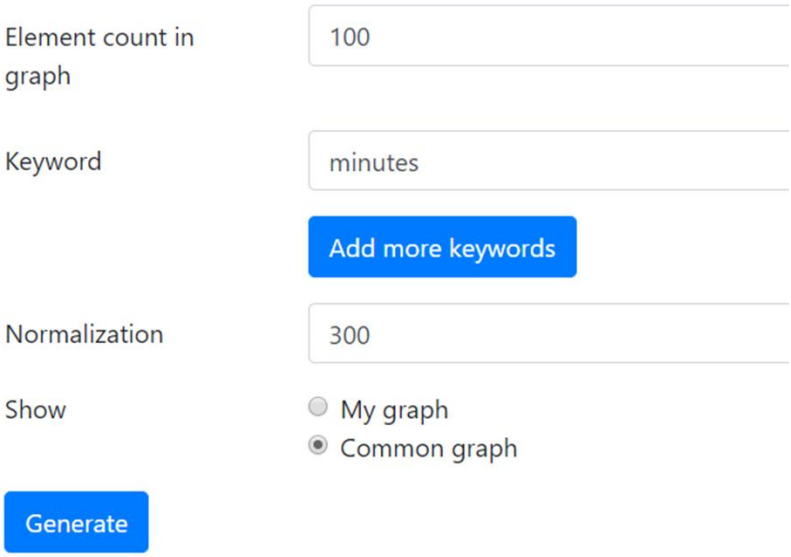
Promocijas darba ietvaros izstrādāts rīks, kas nodrošina funkcionalitāti, lai analizētu nestrukturētus datus, dodot iespēju definēt jaunus KPI. Rīks ir izstrādāts izmantojot tīmekļa tehnoloģijas. Rīka funkcionalitāte ir izstrādāta izmantojot programmēšanas valodu PHP versiju 7.3.1, MVC ietvaru CodeIgniter[18] 3.1.11. Kā datubāze tiek izmantota datubāzes pārvaldības sistēma MariaDB[19] 10.1.37. Grafiskās saskarnes funkcionalitāte ir izstrādāta ar ietvariem jQuery[20] v3.4.1, CSS, HTML un bibliotēku Bootstrap[21] versiju 4.3.1. Rīka lietotājam attēlotais grafs tiek implementēts ar bibliotēkām D3.js[22] ar versiju 5.15.0 bibliotēku un "Pulsa para acceder a Language Network" (WEB, g) piemēru. NLP apstrāde tiek veikta ar Stanford NLP[23] (stanford-parser-3.9.2).

[18] https://codeigniter.com
[19] https://mariadb.org
[20] https://jquery.com
[21] https://getbootstrap.com
[22] https://d3js.org
[23] https://nlp.stanford.edu/software

5.1.4 Aprobācija kvantitatīva KPI specificēšanai

Literatūrā pastāv KPI izvērtēšanas kritēriji, piemēram, (Kim & Chungm 2011) aprakstīts nodaļā 4.2.4, taču ne vienmēr restorāna vadībai ir skaidri zināms, kura vērtība ir optimālā. Apkalpot klientus ātri pīķa stundās var prasīt zināmas investīcijas no restorāna puses, piemēram, papildus darba spēka nolīgšanu, kuri varētu būt mazāk noslogoti citās darba dienas daļās, vai jaunu iekārtu iegāde. Metodē aprakstītais rīks piedāvā iespēju tvītos atrast lietotāju atsauksmes, ko var izmantot kā KPI vērtības.

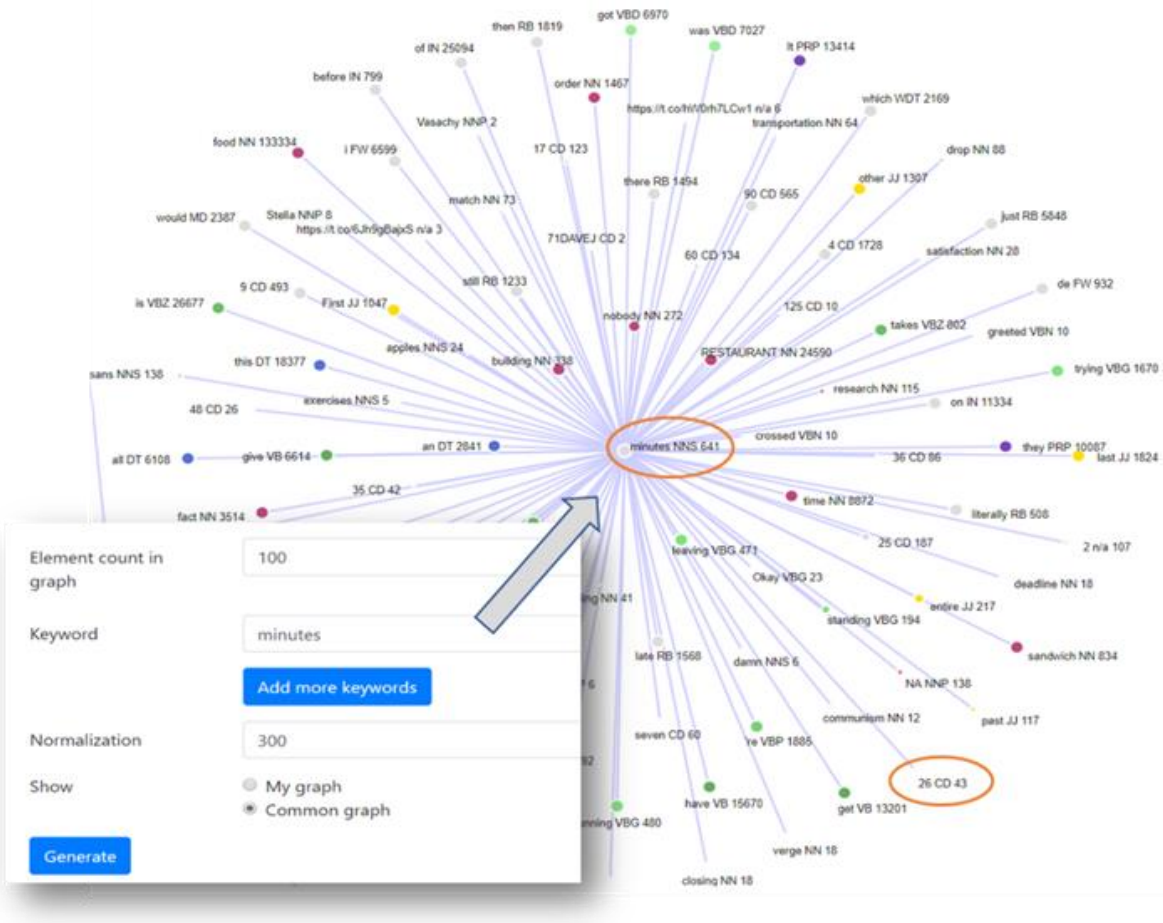
Vispirms rīka lietotājam būtu jāizvēlas veidot jaunu vārdu saistību grafu – parasto grafu. Lietotājam tiek attēlota att. 5.6. redzamā forma.



Att. 5.6. Ievades forma veidojot jaunu grafu

Veidojot jaunu vārdu saistību grafu, lietotājam ir jāizvēlas cik elementus attēlot grafā – parametrs "*Element count in graph*", šis filtrs nepieciešams, lai noskaidrotu biežāk lietotos vārdus tvītos un, lai uzlabotu pārredzamību, maza monitora gadījumā šī vērtība varētu būt mazāka. Parametrs "*Keyword*" ir svarīgākais parametrs, jo rīks veidos grafu par pamatu ņemot šo parametru. Grafu var veidot no teorētiski neierobežoti daudziem atslēgas vārdiem. Šajā gadījumā grafs tiks veidots no vārdiem, kas saistīti ar vārdu "*minutes*". Dati grafā tiks atlasīti saistībā ar restorāniem un ēdieniem, jo datu kopa sastāv tikai no datiem par "*restaurant*", "*food*", "*sushi*". Parametrs "*Normalization*" nosaka grafā maksimālo grafa elementa izmēru. Šis parametrs palīdz uzlabot redzamību gadījumos, ja kāds no grafa elementiem ir pārlieku liels (elementa izmēru nosaka vārda biežums tvītā), piemēram, tas var aizņemt visu ekrānu. Izvēle "*Show*" nosaka grafa tipu, kurā meklēt atslēgas vārdus. Izvēles "*My graph*" gadījumā atslēgas vārds tiks meklēts tikai tajos vārdos, kurus lietotājs ir

pievienojis personalizētajam grafam. Izvēles “Common graph” gadījumā, atslēgas vārds tiks meklēts visos aprēķinātajos datos rīka datubāzē, šī izvēle ir piemērotāka priekš sākotnējās datu analīzes. Nospiežot pogu “*Generate*”, tiek uzģenerēts grafs skatīt att. 5.7. Grafa virsotnes atspoguļo vārdus tvītā, šķautnes starp virsotnēm attēlo relāciju starp diviem vārdiem tvītā.



Att. 5.7. Grafa (“*Parastais grafs*”) piemērs pielietojot atslēgas vārdu "*minutes*"

Šajā gadījumā rīka lietotājs interesējās par skaitliskām vērtībām, jo mērķis ir noteikt KPI vērtību pie kuras lietotāji kļūst neapmierināti. Aplūkojot vārdu saistību grafu, lietotājs redz, ka pastāv attiecība starp vārdu “*minutes*” ar tipu “*NNS*” - lietvārds daudzskaitļa formā un vārdu “26” ar tipu “*CD*” – skaitļa vārds. Rīka lietotājam uzklikšķinot uz šķautnes atveras detalizēts saraksts skatīt att. 5.8. Šajā skatā lietotājam ir iespēja aplūkot visas saistības starp šiem vārdiem un to tipiem, kā arī izlasīt tvītu, kur šis vārdu salikums ir lietots. Konkrētajā gadījumā skaitlis “26” ir lietots, lai aprakstītu laiku, ko restorāna klients ir pavadījis gaidot pasūtīto ēdienu un, ka restorāna klients ir neapmierināts par to, ka tik ilgi ir jāgaida ēdiens.

Analītiķis šajā gadījumā varētu izdarīt secinājumu, ka gaidīšanas laiks, kas ir ilgāks par 26 minūtēm var izraisīt negatīvas emocijas.

Attr_1	Attr_1_type	Attr_2	Attr_2_type	Tweet_id	Tweet	Username
minutes	NNS	26	CD	91631	ordered my food 26 minutes ago and still waiting. @tacobell staff needs to get their shit together.????	bre

Att. 5.8. Vārdu "*minutes*" un "*26*" detalizēts skats

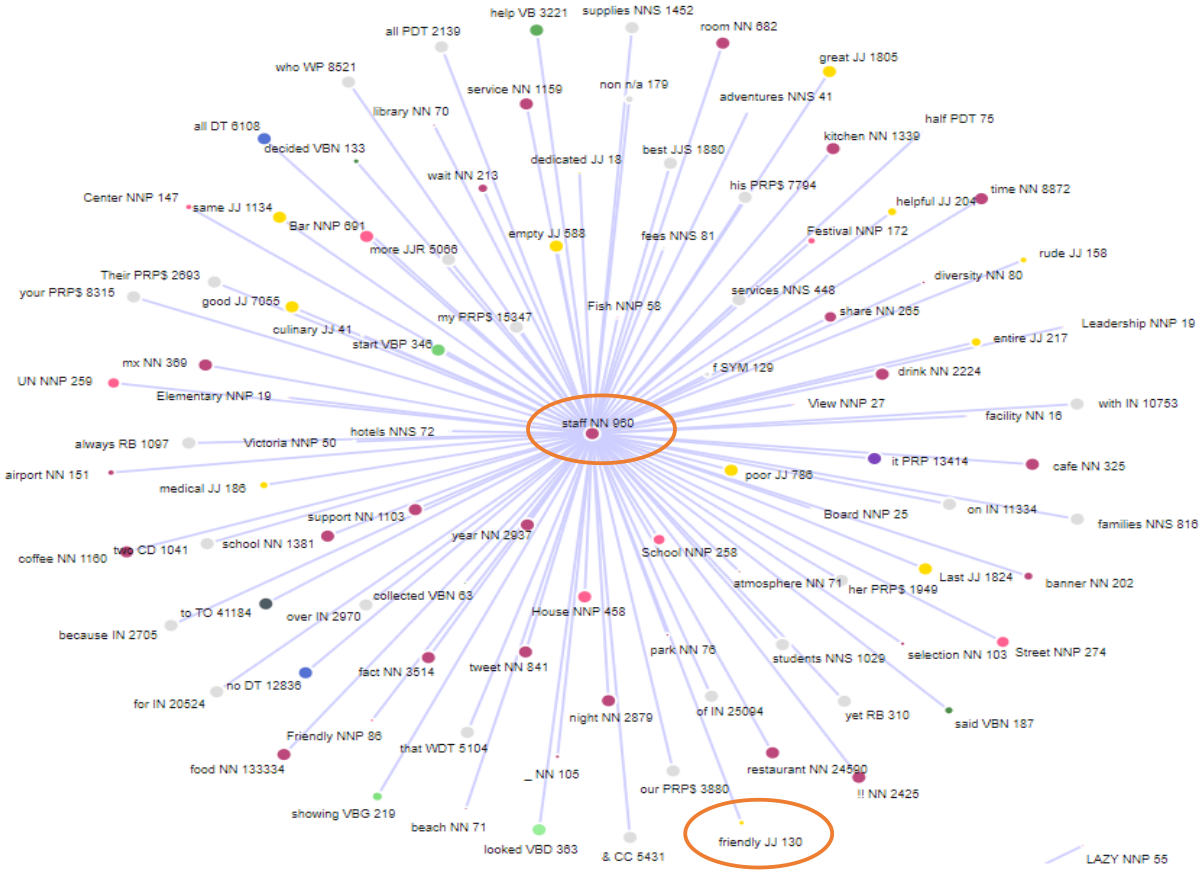
Metode paredz analizēt lietotāju atsauksmes par metodes lietotāja interesējošo tematu. Tabula 5.1. ir apkopotas atsauksmes par vārdu "minutes". Tabulā 5.1. var redzēt, ka lietotāju atsauksmes ir negatīvas, ja gaidīšanas laika ir ilgāks kā 30 minūtes, bet, ja gaidīšanas laiks ir 4 vai 5 minūtes klienti sniedz pozitīvu atsauksmi. Tādejādi analītiķis izvirza šādu KPI: "Gaidīšanas laikam ēdiena saņemšanai ir jābūt mazākam kā 25 minūtes.", kas nedaudz zem 30 minūtēm, kad klienti kļūst neapmierināti.

tabula 5.1.

Saistītā atribūta vērtība	**Tvīts**
30	30 minutes for a sandwich At a FAST FOOD RESTAURANT
30	Wings to go took 30 minutes for 2 orders of wings and a kids quesadilla just took forget our sides. It takes a lot for me to get irritated with a food establishment but I'm annoyed to say the least
30	Waited 30 minutes for food I've lost full..
36	It's been 36 minutes and I finally got my https://t.co/vnWU2ilDM7 PHONE IS ON 15 PERCENT.@PopeyesChicken thanks and no thanks?
35	@PopeyesChicken been waiting 35 minutes for my food. So much for fast food. Disappointing.
45	RT @BostonJerry: There's no fast food item that's worth 45 minutes waiting in line.
4	RT @DailyGuineaLips: It has been a whole 4 minutes since my face has been greeted with food... TOO LONG. Pic sent by @LeonardREW
5	Gluten-Free Vegan No-Cook Raw Cranberry Sauce...made with only 6 clean, real food ingredients and is ready in 5 minutes!

5.1.5 Aprobācija kvalitatīva KPI specificēšanai

Nodaļā 4.2.1. tika noskaidroti, ka pastāv divu veidu KPI - kvantitatīvi un kvalitatīvi. Iepriekšējā nodaļā 5.1.4. tika definēts kvantitatīvs KPI. Šajā nodaļā tiks definēts kvalitatīvs KPI. Rakstā (Kim & Chungm 2011), kas aplūkots nodaļā 4.2.4., viens no restorānu vērtēšanas kritērijiem ir "draudzīgs personāls" ("*Friendly staff*"). Šajā gadījumā tas tiks izteikts kā KPI kvalitatīvs kritērijs, kas ir kāda vērtība, kas nav skaitliska. Rīka lietotājam jāizvēlas veidot jaunu vārdu saistību grafu un kā atslēgas vārds jāizvēlas "*staff*". Pārējie parametri tiek lietoti tādi paši kā iepriekšējā piemērā. Lietotājam tiek uzģenerēts grafs, ko var aplūkot att. 5.9., var redzēt, ka tiek iegūti vārda "*staff*" saistītie vārdi "*great*", "*friendly*", "*poor*" un citi.



Att. 5.9. Grafs lietojot atslēgas vārdu "*staff*"

Līdzīgi kā pirmajā piemērā, rīka lietotājam ir iespēja izpētīt detalizēti vārdu relācijas, uzspiežot uz saites starp vārdiem, dažus piemērus var aplūkot tabula 5.2.

tabula 5.2.

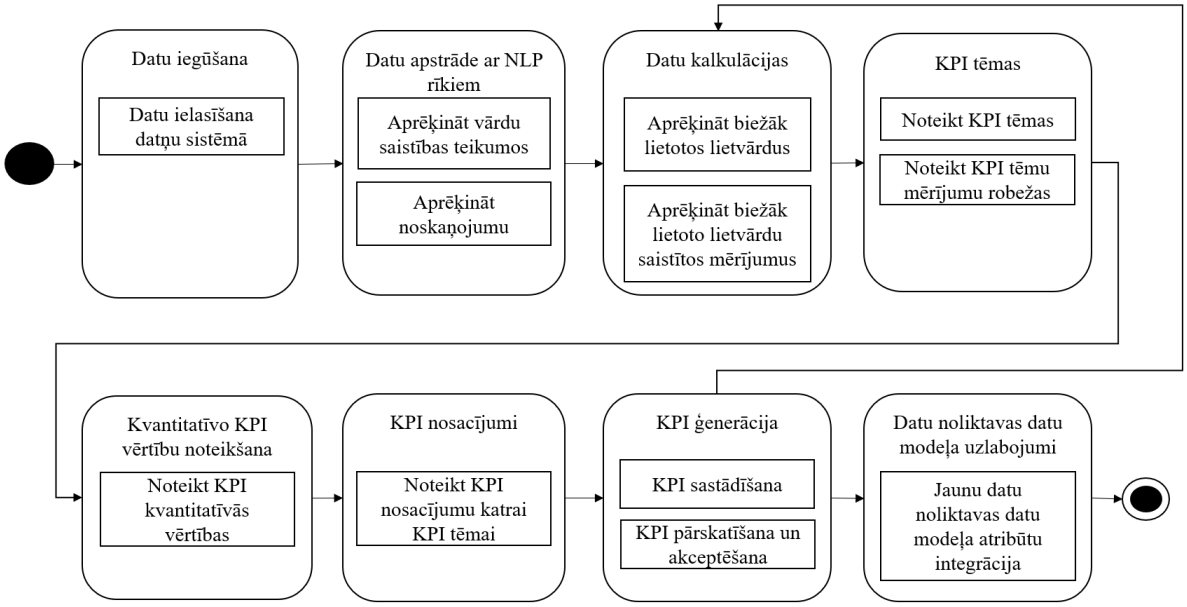
Saistītā atribūta vērtība	**Tweet**
friendly	Great little spot, friendly staff with delicious food and decent coffee, bit out of the way, but worth it
friendly	Great Food!!! Large portions Friendly staff
great	RT @johnbrewer168: @KitchenClonc @bigmoosecoffee for me, great staff, coffee and food always.
great	Great party, great food, great people, great Chef, great staff, great Addo. Looking forward for a wonderful year of new experiences at Addo
poor	The food was really good, but the poor staff is way over worked. Those awesome waitresses need at least one more on deck.
rude	@busabaeathai Westfield horrible food, extremely rude staff “Laura“, use to be an amazing place to eat not anymore sadly.. shocked at the quality of food being served.
friendly helpful	Jackson's have amazing food served by really helpful and friendly staff.

Analītiķis var novērot, ka Twitter lietotāji izsaka atsauksmes par apkalpojošo personālu saistībā ar ēdināšanas jomu. Tas liecina, ka restorāna klienti pievērš uzmanību un vērtē personālu. Lietotāji negatīvi izsakās par personālu gadījumos, kad personāls ir bijis rupjš vai “vājš”. Bet atsauksmes bija pozitīvas gadījumos, kad personāla sniegums tika vērtēts kā draudzīgs un izpalīdzīgs. Pēc šādiem novērojumiem analītiķis varētu izvirzīt KPI: “Restorāna klientu apmierinātība par apkalpojošo personālu ir jābūt augstai.”

5.2 Metode datu noliktavas datu modeļa uzlabojumiem no klientu atsauksmēm

Šajā nodaļā tiek aprakstīta metodes darbība un aprakstīti piemēri no metodes aprobācijas uzņēmumā. Att. 5.10. var aplūkot metodes darbības diagrammu. Metodi var iedalīt astoņos soļos. Metodes pirmais solis ir “Datu iegūšana”, kas nosaka datu iegūšanu un saglabāšanu uz servera, lai tos var izmantot nākamajos metodes soļos. Nākamais solis ir “Datu apstrāde ar NLP rīkiem”, šajā solī ielasītie dati tiek apstrādāti ar metodē izmantotajiem NLP rīkiem, lai iegūtu vārdu savstarpējās saistības tekstā un veiktu noskaņojuma analīzi, rezultāti tiek

saglabāti uz servera. Solī “Datu kalkulācijas” tiek izrēķināti biežāk lietotie vārdi un šo vārdu saistības ar mērījumiem. Solī “KPI tēmas” tiek noskaidrotas KPI tēmas un KPI tēmu saistīto mērījumu robežas. KPI tēmas vēlākajos metodes posmos tiks izmantotas KPI sastādīšanā, bet KPI tēmu saistīto mērījumu robežas nosaka, kādā skaitliskā intervālā tiks meklētas lietotāju atsauksmes. Solī “Kvantitatīvās KPI vērtības” tiek izrēķinātas KPI kvantitatīvās vērtības. Solī “KPI nosacījumi” tiek manuāli noteikti KPI nosacījumi katrai KPI tēmai. Solī “KPI ģenerēšana” tiek sastādīti KPI no iepriekšējos soļos iegūtās informācijas un apspriesti ar ieinteresētajām personām. KPI sastādīšana var notikt iteratīvi par interesējošajiem biežāk lietotajiem lietvārdiem. Metodes noslēdzošais solis ir “Datu noliktavas datu modeļa uzlabojumi”, kas paredz iegūtos atribūtus integrēt datu noliktavas datu modelī.



Att. 5.10. Metodes darbības diagramma

5.2.1 Datu iegūšanas posms

Metode paredz izmantot lietotāju komentārus un atsauksmes kā datu avotu. Lietotāju komentāri un atsauksmes parasti ir nestrukturēti dati. Analizējot šos datus, ir iespējams iegūt vērtīgu informāciju organizācijas attīstībai.

Pastāv vairākas iespējas, ko organizācijā var izmantot kā datu avotu, lai iegūtu atsauksmes un komentārus: a)klientu anketēšana, b)sūdzību/ierosinājumu uzklausīšana, c)atsauksmes par organizāciju vai nozari ārpus konkrētās organizācijas, piemēram, Twitter dati. Pastāv iespēja šos datu avotus kombinēt, lai iegūtu rezultātus, kas balstīti uz plašāku datu apjomu. Nestrukturēti dati datu avotā var būt par jebkuru tematu, tie var būt nesaistīti ar konkrēto nozari par ko tiek veikta analīze. Metode paredz, ka ar nozari nesaistītu datu analīze var nepareizi ietekmēt rezultātus, tāpēc nestrukturētos datus nepieciešams filtrēt un izmantot

tikai konkrētajai nozarei atbilstošos. Lai atfiltrētu konkrētajai nozarei atbilstošos datus, ir nepieciešams pielietot filtrus.

Metodes soļa implementācija.

Kā datu avots klientu atsauksmēm, ir izmantoti publiski pieejamie Twitter dati. Metodē tiek paredzēts, ka dati tiek ievākti par konkrētu jomu, konkrētajā piemērā par ēdināšanu, tas ir, tvītam jābūt saistītam ar vārdiem "*food*", "*restaurant*" vai "*sushi*". Konkrētā joma un atslēgas vārdi ir izvēlēti, jo aprobācijai izvēlētais uzņēmums darbojās ēdināšanas jomā, sniedzot ēdināšanas pakalpojumus uz vietas – restorānā un apkalpo pasūtījumus ar piegādi ārpus restorāna. Restorāna galvenais fokuss ir Japāņu virtuves ēdiens – suši. Katrā organizācijā un atbilstošajai jomai šie vārdi var būt atšķirīgi, pēc jomas specifikas jāizvēlas aizbilstošākie atslēgas vārdi. Twitter tvīts tiek interpretēts kā atsauksme vai komentārs par kādu jomu, bet var izmantot arī citus datu avotus metodes darbībā, piemēram, Youtube, Facebook komentārus vai organizācijas klientu aptaujas.

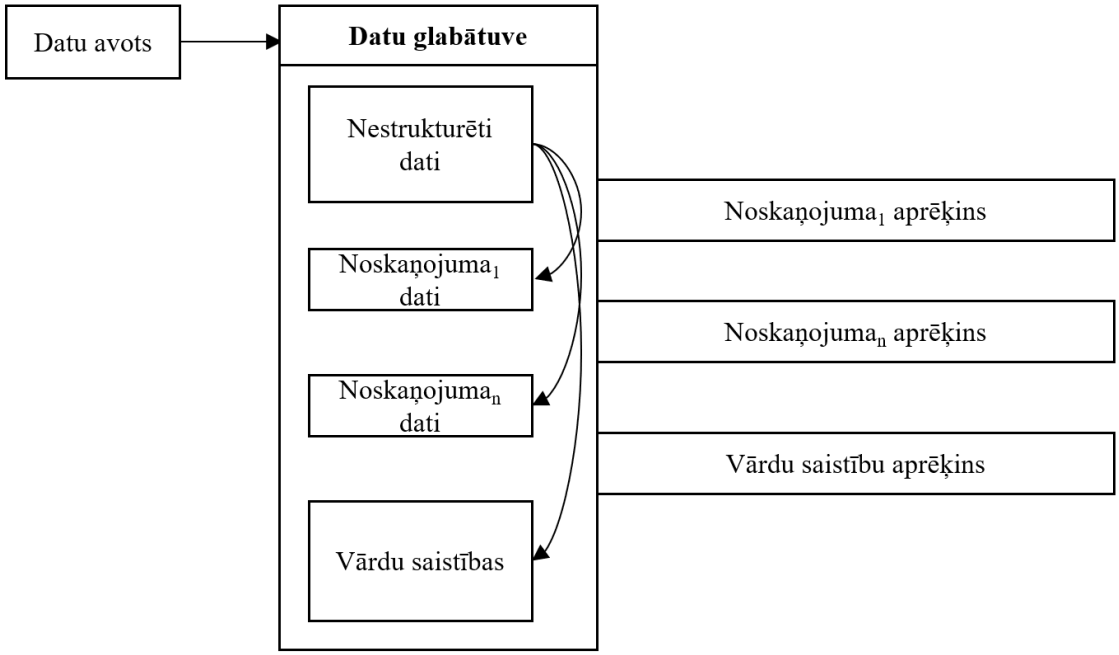
5.2.2 Datu apstrāde ar NLP rīkiem

Metode paredz izmantot kādu no NLP bibliotēkām (StandfordCoreNLP, NLTK, SparkNLP, u.c), kas ļauj noteikt tekstam noskaņojuma koeficientus un teikumā vārdu savstarpējās saistības. Att. 5.11. var redzēt metodes soļa "Datu apstrāde ar NLP rīkiem" vispārīgu atainojumu. Tiek paredzēts apstrādāt nestrukturētus datus, kas ir ielasīti "Datu glabātuvē". Kā datu glabātuvi var izmantot dažādas tehnoloģijas (HDFS, Google Cloud BigQuery[24], Databricks Lakehouse Platform[25]). Att. 5.11. elementi "Vārdu saistību aprēķins*"* un "Vārdu saistības" attēlo teikuma vārdu savstarpējo saistību aprēķinu. Elementi "Noskaņojuma$_1$ aprēķins" un "Noskaņojuma$_n$ aprēķins" attēlo vairāku noskaņojuma koeficientu aprēķinu. Šī informācija tiek saglabāta datu glabātuvē, lai to varētu izmantot tālākajos metodes posmos, skatīt elementus "Noskaņojuma$_1$ dati" un "Noskaņojuma$_n$ dati". Metodes pielietojumā teksts tiek analizēts pēc trīs noskaņojuma tipiem:

- Noskaņojums, kas iekļauj "pārsteigums", "skumjas", "prieks", "bailes";
- Pozitīvais noskaņojums, kas nosaka vai teksts ir ar pozitīvu vai negatīvu "nokrāsu";
- Sarkasms, kas nosaka vai noskaņojums ir ar sarkasma nozīmi.

Atkarībā no pielietotajām NLP bibliotēkām, tad var pievienot papildus noskaņojuma tipus pēc kuriem analizēt tekstu. Vēlākajos metodes darbības posmos katrs noskaņojuma tips var sniegt papildus informāciju.

[24] https://cloud.google.com/bigquery
[25] https://docs.databricks.com/lakehouse/index.html



Att. 5.11. NLP apstrādes posma attēlojums diagrammā

Pēc šī metodes soļa ir pieejama informācija par:

- Teksta noskaņojumu piemērā "Noskaņojuma dati" - teksta pozitīvu vai negatīvu nokrāsu vienā vai vairākos noskaņojuma tipos, šī informācija tiek aprēķināta ar darbību "Noskaņojuma aprēķins";
- Vārdu vārdšķiras piemērā "Vārdu saistības" - vārda un tā vārdšķira teikumā, piemēram, vārds "*potatoes*" un tips "*NNS*" – lietvārds daudzskaitļa formā, šī informācija tiek aprēķināta ar darbību "Vārdu saistību aprēķins";
- Vārdu savstarpējās saistības teikumā piemērā "Vārdu saistības", piemēram, ļoti vienkāršs teikums "*Warm potatoes*", kurā vārdi "*warm*" un "*potatoes*" ir saistīti ar saiti "*amod"* – īpašības vārda noteicējs, šī informācija tiek aprēķināta ar darbību "Vārdu saistību aprēķins".

Metodes soļa implementācija.

Metodes soļa implementācijā tiek izmantots vairāku tehnoloģiju kopumus. Lai straumētu datus no Twitter, tiek izmantota Pythoon [26] bibliotēka Tweepy[27]. Neapstrādāti tvīti tiek saglabāti uz servera JSON failu formāta. Nākamais solis paredz šos nepastrādātos tvītus ielasīt HDFS, jo metode nosaka, ka dati var būt liela apjoma un tie var būt nestrukturēti. Lai datus atlasītu no HDFS, tiek izmantots dzinējs Apache Spark[28]. Metodes soļa implementācijā

[26] https://www.python.org
[27] https://www.tweepy.org
[28] https://spark.apache.org

izmanto Jupyter[29] projektu, lai būtu pieejams apstrādes rīks Jupyter Notebook[30], kas ļauj izmantot grafisko saskarni, izstrādājot Apache Spark uzdevumus (*jobs*). Tiek izmantots Apache Spark modulis pyspark.sql[31], lai varētu Apache Spark uzdevumus izstrādāt daudziem IT speciālistiem zināmā sintaksē - pazīstamajā SQL sintaksē. Att. 5.12. var redzēt piemēru, kā no Jupyter Notebook tiek veidoti Apache Spark uzdevumi. Pirmajā Apache Spark uzdevumā ar 9 numuru tiek ielasīti visi faili no HDFS ar datu tipu JSON un ielasīts mainīgajā "df", kam ir speciāls Apache Spark datu tips DataFrame[32]. DataFrame pēc struktūras ir līdzīgs divdimesiju masīvam. Komanda *df.count()* saskaita un attēlo ielasīto rindiņu skaitu. Otrajā Apache Spark uzdevumā ar numuru 10 tiek izveidots Apache Spark pagaidu skats no iegūtā "*df*" ar nosaukumu "*data*". Vēlāk šo skatu var izmantot rakstot SQL vaicājumus. Piemērā var redzēt, ka tiek izmantota SQL komanda, lai noskaidrotu ID dublikātus.

```
In [9]: df = spark.read.json("hdfs://localhost:9820/shopping/44/*.json")

        df.count()

Out[9]: 35574

In [10]: df.createOrReplaceTempView("data")

         aa = spark.sql("SELECT id, count(1) aa from data "
                        " group by id "
                        " having aa >1  "
                        " order by aa desc ")
         aa.count()

Out[10]: 3
```

Att. 5.12. Piemērs Apache Spark uzdevumam SQL sintaksē

Lai apstrādātu tvīta tekstu, tiek izmantota SparkNLP [33] atvērtā pirmkoda NLP bibliotēka. SparkNLP tiek izmantots, jo tam ir plaša atvērtā pirmkoda modeļu pieejamība, tas ir viegli instalējams un tas ir plaši izmantots organizācijās. SparkNLP ir izstrādāts pateicoties John Snow Labs [34] atbalstam. SparkNLP ir uzbūvēts par pamatu izmantojot Spark ML[35]. SparkNLP anotācijas izmanto noteikumos balstītus algoritmus Tensorflow[36], mašīnmācīšanās (ML - *machine learning*) tehnikas, lai īstenotu dziļo mašīnmācīšanos (*deep learning*). Apache Spark ML ļauj viegli tehnisko izstrādes projektu mērogot bez izmaiņām kodā. Tensorflow ļauj

[29] https://jupyter.org
[30] https://jupyter.org
[31] https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html
[32] https://spark.apache.org/docs/latest/sql-programming-guide.html
[33] https://sparknlp.org
[34] https://www.johnsnowlabs.com
[35] https://spark.apache.org/docs/latest/ml-guide.html
[36] https://www.tensorflow.org

darbināt dziļās mašīnmācīšanās algoritmus, izmantojot GPU tādas tehniskas platformas kā nVidia's DGX-1[37] un Intel's Cascade Lake processors[38]. SparkNLP nodrošina pazīstamākos NLP uzdevumus - teksta dalīšana tekstvienībās, lemmatizācija, vārdšķiras noteikšana, noskaņojuma analīze, pareizrakstības pārbaude nosaukto entitāšu atpazīšana (*named entity recognition*) u.c. Spark NLP nodrošina iespēju izmantot publiski pieejamus atvērtā pirmkoda modeļus, kurus var trenēt uz saviem datiem, kā arī jau iepriekš uztrenētām saistītām funkcijām (*pipelines*) un modeļiem. Metodes implementācijā tiek izmantoti iepriekš uztrenēti modeļi, ar kuru palīdzību tiek noskaidrotas vārdu saistības teikumos un teksta noskaņojuma analīze att. 5.11. Iepriekš uztrenētie modeļi var tikt lejupielādēti to izmantošanas brīdī, norādot nosaukumu un konfigurācijas parametrus. Ar Spark NLP iepriekš uztrenētajiem modeļiem tiek apstrādāts tvīta brīvais teksts. Tiek izmantots tvīts "*Had lunch over zoom some days ago. I guess I may open a restaurant at long last.*", kā piemērs, lai aprakstītu izmantotos SparkNLP modeļus.

SparkNLP iepriekš uztrenētā modeļa classifierdl_use_emotion izmantojums. Modelis classifierdl_use_emotion (WEB, d) nodrošina funkcionalitāti noteikt teikumam noskaņojumu – pārsteigums, skumjas, sajūsma vai bailes. Šis SparkNLP modelis ir trenēts izmantojot vairākas datu kopas, kā piemēram, YouTube komentārus, Twitter un ISEAR[39] (*International Survey on Emotion Antecedents and Reactions*) datu kopu. Iegūtie noskaņojumu koeficienti tiek izmantoti analizējot lietotāju atsauksmes. Par katru noskaņojuma tipu ir pieejams koeficients, skatīt tabula 5.3., kur tvīts ir apstrādāts ar classifierdl_use_emotion modeli. Var redzēt, ka SparkNLP classifierdl_use_emotion modelis ir noteicis, ka visvairāk šis tvīts atbilst noskaņojumam "sajūsma" ar koeficientu 0.9977035.

tabula 5.3.

Tvīts	**Noskaņojums**	**Koeficients**
Had lunch over zoom some days ago. I guess I may open a restaurant at long last.	Skumjas	0.0006125135
	Pārsteigums	0.0016160638
	Sajūsma	**0.9977035**
	Bailes	0.0000679279

SparkNLP iepriekš uztrenēta modeļa SENTIMENTDL_USE_TWITTER izmantojums. Modelis SENTIMENTDL_USE_TWITTER (WEB, h) par pamatu izmanto

[37] https://www.nvidia.com/en-gb/data-center/dgx-systems/dgx-1
[38] https://www.intel.com/content/www/us/en/products/platforms/details/cascade-lake.html
[39] https://paperswithcode.com/dataset/isear

modeli “Universal Sentence Encoder” (Cer et al., 2018). Šis modelis ir trenēts uz Twitter datu kopu, kas sastāda 1,6 miljonus tvītu. Apstrādājot tekstu ar šo modeli, tiek iegūti koeficienti, vai šis tvīts ir pozitīvs vai negatīvs. Piemēru skatīt tabula 5.4. Piemērā var redzēt, ka tvīts ir ar gandrīz 100% pozitīvu noskaņu.

tabula 5.4

Tvīts	**Noskaņojums**	**Koeficients**
Had lunch over zoom some days ago. I guess I may open a restaurant at long last.	Negatīvs	0.0003254917
	Pozitīvs	**0.99967456**

SparkNLP iepriekš uztrenēta modeļa classifierdl_use_sarcasm izmantojums. Modelis classifierdl_use_sarcasm (WEB, i) ir trenēts izmantojot datu kopu “Sarcasm-Detection” (WEB, j). Šis modelis nosaka, vai teksts ir lietots kā sarkasms. Piemērā, kas redzams tabula 5.5, var novērot, ka tas nav sarkastiskā nozīmē, jo gandrīz 100% tas ir normāls.

tabula 5.5

Tvīts	**Sarkasms**	**Koeficients**
Had lunch over zoom some days ago. I guess I may open a restaurant at long last.	Sarkasms	0.0015251364
	Normāls	**0.9984749**

SparkNLP iepriekš uztrenēta modeļa dependency_typed_conllu izmantojums. Modelis dependency_typed_conllu (WEB, m) sadala tekstu vārdos un atgriež katra vārda vārdšķiras tipu, saistīto vārdu un šīs saistības tipu. Šis modelis ir trenēts izmantojot “CONLL” datu kopu (WEB, b). Apstrādājot piemērā izmantoto tvītu “Had lunch over zoom some days ago. I guess I may open a restaurant at long last.”, tiek iegūts rezultāts, ko var aplūkot att. 5.13., var redzēt, ka tiek iegūta informācija par katru vārdu teikumā. Vārdi no piemēra teikuma ir redzami kolonnā “chunk”, kolonnās “begin” un “end” ir redzamas šo vārdu sākuma un beigu pozīcijas teikumā. Kolonnā “pos” ir redzamas vārdu vārdšķiras. Kolonnā “dependency” ir redzami teikumā saistītie vārdi, bet kolonnā “dependency_type” ir redzams šīs saistības tips.

Piemēram, vārds “restaurant” ir teikumā sākot no 56 līdz 65 pozīcijai, tā vārdšķira ir “NN” – lietvārds. Šis vārds ir saistīts ar vārdu “open” ar relācijas tipu “nsubj” – nominālais priekšmets.

chunk	begin	end	pos	dependency	dependency_type
Had	0	2	VBD	lunch	parataxis
lunch	4	8	NN	ROOT	root
over	10	13	IN	zoom	det
zoom	15	18	NN	lunch	flat
some	20	23	DT	days	nsubj
days	25	28	NNS	ago	nsubj
ago	30	32	RB	lunch	amod
.	33	33	.	lunch	punct
I	35	35	PRP	guess	nsubj
guess	37	41	VBP	lunch	parataxis
I	43	43	PRP	open	nsubj
may	45	47	MD	open	appos
open	49	52	VB	guess	parataxis
a	54	54	DT	restaurant	nsubj
restaurant	56	65	NN	open	nsubj
at	67	68	IN	long	case
long	70	73	JJ	restaurant	amod
last	75	78	NN	long	nsubj
.	79	79	.	lunch	punct

Att. 5.13. Piemērs modeļa “dependency_typed_conllu” darbībai

5.2.3 Datu kalkulācijas posms

Datu kalkulācija notiek divos posmos. Pirmais posms nosaka aprēķināt **“biežāk izmantotie vārdi ar vārdšķiru “lietvārds””** par konkrētu noskaņojumu, piemēram, negatīvu noskaņojumu. Otrais posms nosaka aprēķināt **“biežāk lietoto lietvārdu vārdu saistību ar kādu mērījumu”**. Tiek noteikts, ka mērījumi ir tie vārdi tekstā, kuru vārdšķira ir skaitlis un skaitļa saistītais vārds ir mērījuma mērvienība, piemēram, ja saistītais vārds ir “minūtes”, tad var attiecināt, ka mērījums ir par laiku, bet ja skaitļa saistītais vārds ir “EUR”, tad var attiecināt, ka mērījums ir par izmaksām.

Lai izrēķinātu “biežāk izmantotos vārdus ar vārdšķiras tipu “lietvārds””, nepieciešams:

- Saskaitīt vārdus, kuriem vārdšķira ir lietvārds un kur, piemēram, negatīvais noskaņojuma koeficients ir lielāks par citiem koeficientiem;
- Izslēgt analīzei nederīgos vārdus, kā piemēram - tehniskas frāzes (“https”, “@”), vārdus, kuri tika izmantoti, lai atfiltrētu atsauksmes un komentārus par konkrētu jomu. Tas nepieciešams, jo vārdi kuri tika izmantoti kā filtri datu ieguvei, lai atlasītu ar jomu saistītās atsauksmes, visbiežāk parādīsies kā biežāk lietotie vārdi.

Metodes soļa implementācija.

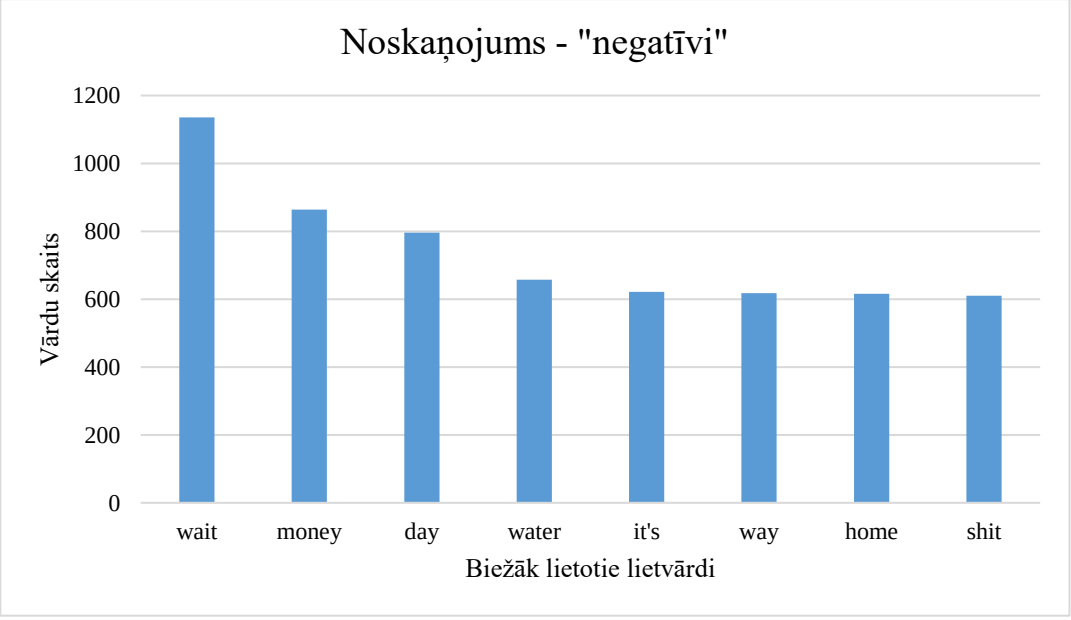
Att. 5.14. var redzēt Spark SQL vaicājumu, kā tiek aprēķināti "**biežāk izmantotie vārdi ar vārdšķiru "lietvārds""**

```
In [6]: df.createOrReplaceTempView("data")

sqlDF = spark.sql("SELECT  chunk , "
                  " count(1) negative"
                  " FROM data "
                   "join tmp on tmp.id = data.id"
                  " where tmp.pos = 'NN' "
                  " and cast(concat_ws(',',sentiment_2.metadata.negative) as double)  > 0.8"
                  " and chunk not in ('food', 'restaurant', 'sushi') "
                  " and chunk not like '@%' "
                  " and chunk not like 'https%' "
                  " and chunk not like '&amp%'"
                  " group by chunk "
                  " order by negative desc limit 10" )
sqlDF.show()
```

Att. 5.14. Biežā lietoto lietvārdu kalkulācijas SQL

Att. 5.15. var redzēt iegūtos rezultātus. Var redzēt, ka lietotāji izsakās negatīvi, ja tvīts satur kādu no vārdiem "time", "money", "day", "shit". Šie vārdi var liecināt par ko cilvēki visbiežāk ir neapmierināti saistībā ar ēdināšanas jomu. Piemēram, vārds "time" varētu nozīmēt to, ka cilvēki ir neapmierināti gadījumos, kad ir ilgi jāgaida, lai saņemtu ēdienu. Vārds "money" varētu liecināt, ka ēdiens bija dārgs vai arī ēdiens vai serviss bija cenai neatbilstošas kvalitātes. Vārdu "day" varētu sasaistīt ar teicienu "waiting all day long", kas arī varētu liecināt par to, ka cilvēki ir neapmierināti, ja ir ilgi jāgaida, lai saņemtu ēdienu.



Att. 5.15. Biežāk lietotie lietvārdi

Lai izrēķinātu **“biežāk lietoto lietvārdu vārdu saistību ar mērījumu”** nepieciešams:

- Meklēt informāciju par kādu no biežāk lietotajiem lietvārdiem;
- Pievienot nosacījumu, ka tekstā jāpastāv kādam mērījumam, tas ir, vārdam ar vārdšķiru “skaitlis” (*CD*), kā arī jāpievieno nosacījums, kas nosaka mērvienību, piemēram, “minutes”, tas nosaka, ka mērījumi ir vienā mērvienībā un tos var savā starpā salīdzināt;
- Konkrētam mērījumam (skaitlim) izrēķināt vidējās vērtības par katru no noskaņojumiem.

Pēc šo operāciju veikšanas tiek iegūta saistība starp mērījumu un noskaņojumu vidējām vērtībām skatīt tabula 5.6. Kolona M apzīmē iegūtos mērījums - skaitļus $M_1..M_n$, bet kolonas $E_1..E_n$ apzīmē konkrēto noskaņojumu, piemēram, sarkasms, par kuriem tiek rēķinātas vidējās vērības – koeficientus attiecīgie tabulas C_{EM} elementi.

tabula 5.6.

M	**E_1**	**E_n**
M_1	C_{E1M1}	C_{EnM1}
M_n	C_{E1Mn}	C_{EnMn}

Att. 5.16. var redzēt Spark SQL vaicājumu, lai noskaidrotu biežāk lietoto lietvārdu saistību ar mērījumiem. Tiek izmantoti iepriekš izrēķinātie dati, kas iegūti metodes solī “NLP processing” un saglabāti HDFS. Konkrētajā gadījumā tiek meklēti mērījumi ar mērvienību “minutes” par lietvārdu “wait”. SQL vaicājumā 16 rindiņā var redzēt, ka tiek meklēti vārdi, kuru tipi ir “CD” – skaitliska vērtība un ar šo vārdu saistītajam vārdam jābūt “minutes” - 14 rindiņa, kā arī tvītā, kurā ir šis vārdu salikums, ir jābūt vārdam “wait” - 15. rindiņa.

```
SELECT
  cast(tmp.chunk as int) as minutes,
  avg(
    cast(
      concat_ws(
        ',', sentiment.metadata.negative
      ) as decimal(38, 8)
    )
  ) negative
FROM
  tmp
  join data on tmp.id = data.id
where
  tmp.dependency = 'minutes'
  and tmp.text like '%wait%'
  and tmp.pos = 'CD'
  AND cast(tmp.chunk as int) is not null
group by   minutes

```

Att. 5.16. SQL kalkulācijas solis lai aprēķinātu "biežāk lietoto lietvārdu vārdu saistību ar mērījumu"

Tabula 5.7. var redzēt saistību starp cilvēku negatīvo noskaņojumu un laiku. Tabula 5.7. var novērot, ka laikam ejot pieaug cilvēku negatīvais noskaņojums. Tas var liecināt, ka klienti izsakās negatīvi, ja ilgāku laiku pavada, piemēram, rindā.

tabula 5.7.

Laiks (m)	**Negatīvā noskaņojuma koeficients**
4	0.15819366
5	0.00E+00
15	0.71939789
20	0.99982475
24	0.00004389
25	0.9999994
30	0.78878643
35	1
40	1
45	0.49657143
50	1
90	0.98592675

5.2.4 KPI tēmu noteikšana

Klientu atsauksmes un komentāri par konkrētu tematu, kuri ir izteikti pozitīvi vai negatīvi, ir svarīgas detaļas, jo tās ir raisījušas organizācijas klientu emocijas – pozitīvas vai negatīvas, un tieši šīs atsauksmes ir tās, par kurām ir jāveido jauni KPI organizācijas attīstībai. Tas nozīmē, ka biežāk lietotie lietvārdi lietotāju atsauksmēs un komentāros nosaka tematus, par kuriem ir nepieciešams veidot jaunus KPI. Biežāk lietotie lietvārdi tika noskaidroti metodes solī "Datu kalkulācijas" aprēķinā "biežāk izmantotie vārdi ar vārdšķiru "lietvārds"". Iegūtie lietvārdi būtu jāizvērtē un jāpārrunā ar ieinteresētajām personām, iespējams kāds lietvārds nav derīgs lai par to sastādītu KPI.

Otra darbība, kas jāveic šajā solī, ir noteikt katras tēmas mērījumiem maksimālās vērtības, piemēram, mērījumu maksimālās vērtības cenai ēdināšanas jomā un apdrošināšanas atlīdzībai atšķirties.

Metodes soļa implementācija.

Aprēķinā "biežāk izmantotie vārdi ar vārdšķiru "lietvārds"" tika noskaidrots, ka biežāk lietotie lietvārdi ar negatīvu koeficientu ir "wait", "money", "day", "water", "it's", "way", "home", "shit". Ne visi no šiem lietvārdiem ir pielietojami kā KPI tēmas, šajā brīdī manuāla analītiķa iesaistīšanās būtu nepieciešama, lai iekļautu tos lietvārdus, kuri ir piemēroti priekš KPI tēmas. Piemēram, dažos gadījumos piemērotāk būtu lietot vārda kādu sinonīmu. Konkrētajā gadījumā lietvārdi, kuri būtu piemēroti KPI tēmām ir "wait", "money". KPI tēmai "wait" maksimālā mērījuma vērtība tiek noteikta 95 minūtes, bet tēmai "money" 20 EUR.

5.2.5 Kvantitatīvo KPI vērtību noteikšana

Šajā metodes solī tiek paredzēts noteikt KPI kvantitatīvās vērtības. Lai noteiktu KPI kvantitatīvās vērtības, ir jāizmanto iepriekšējos posmos izrēķināto informāciju - "biežāk lietotajiem lietvārdiem saistībā ar mērījumu".

Princips, kā noteikt automātiskā veidā KPI kvantitatīvo vērību, ir noteikt mērķi, ka KPI vērtībai ir jābūt ne lielākai kā vidējai vērtībai pret kopējo *trendu* no lietotāju atsauksmēs iegūtajām koeficientu vērtībām par konkrētu tematu. Piemēram, temats var būt "gaidīšanas laiks". Autors (Hill, 1978) *trendu* definē kā vērtību tendenci noteiktā laikā ar vienmērīgu regularitāti palielināties vai samazināties. Tendenču līnija (*trendline*) tiek izmantota, lai attēlotu lietotāju atsauksmju tendenci pēc noteikta principa. Tendenču līnijas tiek pielietotas dažādās jomās, kā piemēram, meteoroloģijā (Jain & Kumar, 2012), medicīnā (Birnbaum et al., 1997) un (Liu et al., 2022) u.c.

Noskaņojuma koeficienta vidējo vērtību var aprēķināt izmantojot formulu $A = \frac{1}{n}\sum_{i=1}^{n} e_i = \frac{a_1+a_2+a_n}{n}$, kur $a_1..a_n$ ir noskaņojuma tipa koeficientu vidējās vērtības un n ir

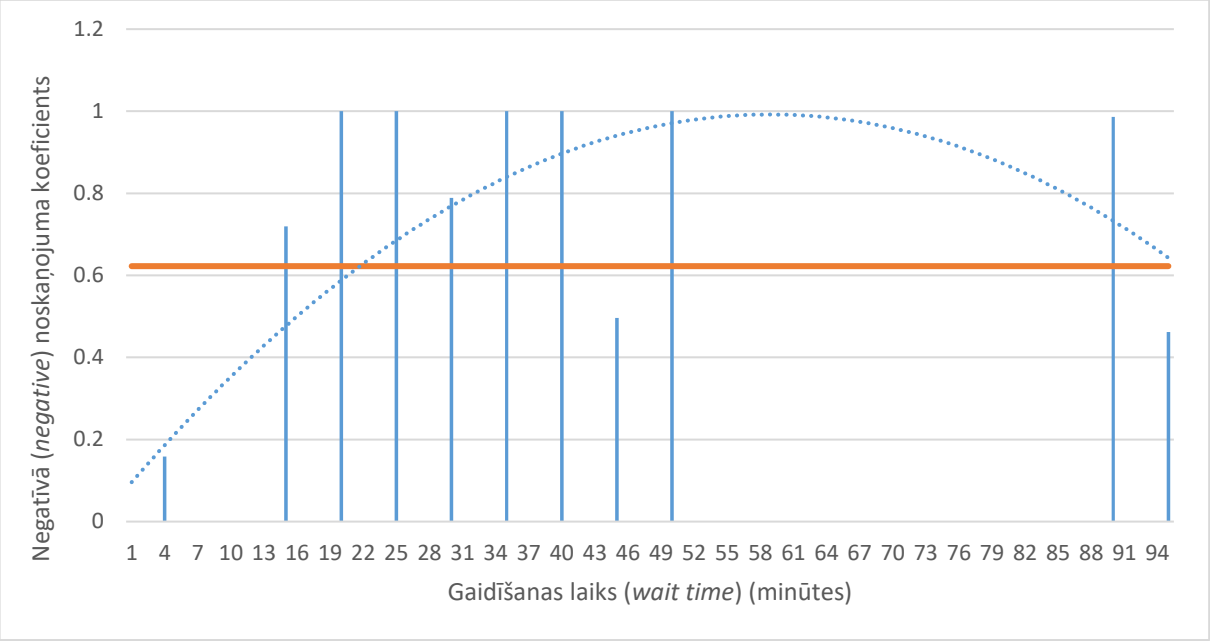
koeficientu skaits. Iegūtā vidējā vērtība grafikā reprezentē taisni $Y=A$. Tendenču līnija grafikā attēlo tendenci kā mainās lietotāju noskaņojums, izmantojot pieejamās vērtības. Tendenču līnijas noteikšana ir nepieciešama, jo kā att. 5.17. var redzēt, pie koeficientu vērtības tendenču līnija strauji pieaug, bet pie 24 minūšu atzīmes tā varētu nesniegt kopējai tendencei raksturojošu mērījumu, jo tur vērtība ir 0. Tendenču līnija tiek izrēķināta, izmantojot noskaņojuma koeficientu vērtības un parabolas funkciju $ax^2 + bx + c = y$.

KPI kvantitatīvā vērība ir taisnes, kas grafikā atspoguļo vidējo vērtību no lietotāju atsauksmju koeficientu vērtībām un lietotāju atsauksmju koeficientu tendenču līnijas krustpunkts.

Metodes soļa implementācija.

Lietotāju komentāros par gaidīšanas laiku vidējā vērtība no negatīvo noskaņojumu koeficientiem ir 0.622386, tātad $Y = 0.622386$, bet tendenču līnijas (parabolas funkcija) ir $y = -0.00027x^2 + 0.0314991x + 0.064778$. Tendenču līnija (parabolas funkcija) tiek iegūta no zināmajām X (laiks) un Y (noskaņojumu koeficientu) vērtībām. Grafikā (att. 5.17) y ass apzīmē negatīva noskaņojuma koeficientu, bet x ass - laika (minūtes) vērtības. Grafikā var redzēt, ka negatīvās emocijas koeficients pie 4 minūšu atzīmes ir zems, tas ir mazāks kā 0.2, bet sasniedzot 15 minūšu atzīmi tas strauji ceļas, pie 20 minūšu atzīmes tas jau ir sasniedzis maksimālo vērtību koeficientu 1. Pie 30 minūšu atzīmes tas nedaudz nokritās, bet joprojām ir izteikti negatīvs - ap 0.79. Grafikā var redzēt, ka pie 45 minūšu atzīmes negatīvais koeficients samazinās, to, iespējams, var skaidrot ar piegādēm uz mājām, kas šādos gadījumos būtu pieņemama vērtība. Grafikā horizontālā taisne ir vidējā vērtība no negatīvo noskaņojumu koeficientu vērtībām, kas ir 0.622386, bet pārtrauktā līnija ir tendenču līnija (parabolas funkcija no x un y vērtībām).

Metode nosaka, ka KPI vērtībai jābūt mazākai kā vidējai vērtībai no lietotāju atsauksmēm par konkrēto tematu. Tas nozīmē, ka tendenču līnijas un taisnes krustpunkts norāda uz KPI kvantitatīvo vērtību. Šajā gadījumā tās ir 22 minūtes.



Att. 5.17. Tendenču līnijas un vidējās vērtības krustpunkts

5.2.6 KPI nosacījumu noteikšana

Lai veidotu kvantitatīvus KPI, ir nepieciešams nosacījums par konkrētu mērījumu, metode nosaka pielīdzināt katru KPI tēmu kā pozitīvu vai negatīvu, skatīt tabula 5.8., ET_1 apzīmē konkrētu KPI tēmu, kurš var būt pielīdzināts kā pozitīvs vai kā negatīvs, tabulas kolonna "Nosacījums". Šī darbība katrai KPI tēmai ir jāveic manuāli. Klienti parasti sagaida, ka KPI tēmas kā "cena", "gaidīšanas laiks" ir mazāka vērtība, tāpēc tiek piemērots nosacījums "mazāk (<)", piemērām "cena < 12 EUR". Bet tādām KPI tēmām kā "apdrošināšanas atlīdzības apjoms", vai "brīvās vietas restorānā" ir vēlams būt augstākai vērtībai, tāpēc ir jāpielieto nosacījums "lielāks (>)", piemēram, "apdrošināšanas atlīdzības apjoms > 600 EUR".

tabula 5.8.

KPI tēma	**Nosacījums**
ET_1	Lielāks vai mazāks (> vai <)
ET_n	Lielāks vai mazāks (> vai <)

Metodes soļa implementācija.

Tabula 5.9. var redzēt piemēru par pielietoto KPI tematu attiecinājumu kā "lielāks" vai "mazāks" nosacījumam.

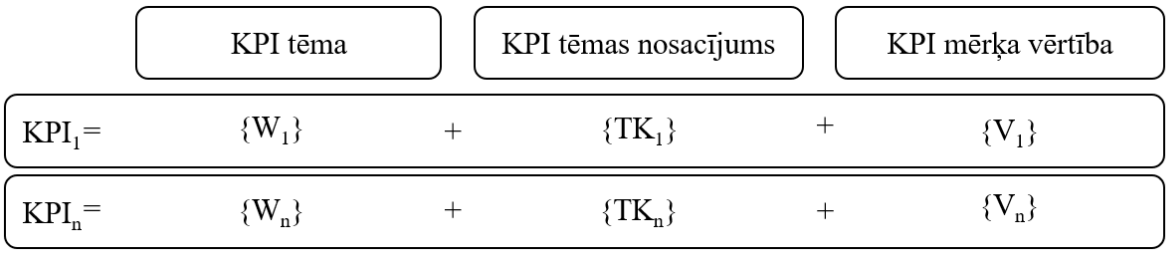
tabula 5.9.

KPI tēma	**Nosacījums**
Price	< (negatīvs)
Wait time	< (negatīvs)

5.2.7 KPI ģenerācija

Vienkāršs kvantitatīvs KPI sastāv no lietvārda, nosacījuma, KPI mērķa vērtības un mērvienības. Metode nosaka, ka kā lietvārdu jāizmanto iepriekš noskaidrotie KPI temati - biežāk izmantotos vārdus ar vārdšķiras tipu "lietvārds", att. 5.15., nosacījums tiek pielietots atbilstoši definētajai katrai KPI tēmai, skatīt tabula 5.9., kvantitatīvā KPI mērķa vērtības tiek pielietotas no posma "Kvantitatīvo KPI vērtību noteikšana", skatīt Att. 5.17.

Att. 5.18. var aplūkot vispārīgā formā ilustrētu paņēmienu kā sastādīt kvantitatīvus KPI. "KPI tēma" ir biezāk lietotie vārdi, "KPI tēmas nosacījums" - pozitīvs vai negatīvs, bet "KPI mērķa vērtības" ir izrēķinātās mērķa vērtības konkrētam KPI un vērtības attiecīgā mērvienība. Pēc KPI automātiskas sastādīšanas iegūtos KPI ir nepieciešams pārskatīt un pārrunāt ar ieinteresētajām personām.



Att. 5.18. Kvantitatīvs KPI vispārīgā formā

Metodes soļa implementācija.

Aplūkotajā piemērā tika noskaidrotas KPI tēmas – "wait time", "money", "quality". Tika izvēlēts vārds "wait time", par kuru nepieciešams noskaidrot (kalkulēt) mērījumus. Piemērā tika noskaidroti mērījumi saistībā ar negatīvo noskaņojumu. KPI tēma "wait time" ir pielīdzināta nosacījumam < (mazāks). Tātad tika iegūts lietvārds "wait time", nosacījums "<", mērījuma vērtība "24" un mērījuma mērvienība "minutes". Pēc metodes apraksta tiek izveidots vienkāršots KPI – "wait < 24 minutes", ko var pārformulēt cilvēciskākā formātā "Wait time should be less than 24 minutes."

5.2.8 Datu noliktavas datu modeļa uzlabojumi

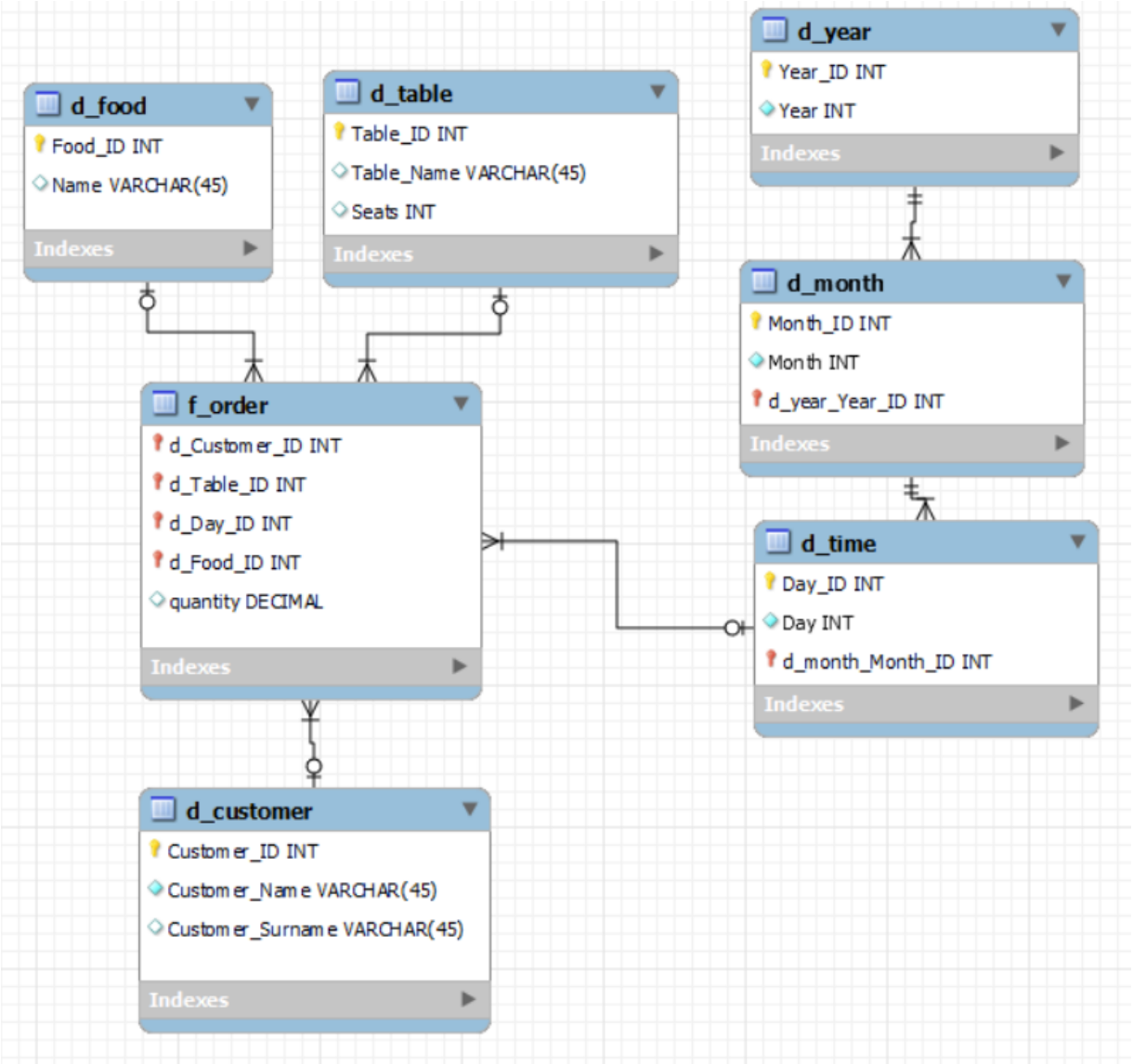
Metode paredz, ka tie lietvārdi, par kuriem tiek veikti mērījumi, ir jāiekļauj datu noliktavas datu modelī kā atribūti. Šāda darbība jāveic, lai organizācijas varētu veikt mērījumus par jauniegūtumiem KPI. Šobrīd atribūtu integrēšana datu modelī tiek paredzēta manuāli, bet šo manuālo darbību ir paredzēts automatizēt.

5.2.9 Metodes aprobācija uzņēmumā

Lai pārliecinātos kā metode darbojās, metode tika pielietota organizācijā, kas darbojās ēdināšanas jomā. Aprobācijas soļu darbība tikai aprakstīta no nodaļas 5.2.1. līdz 5.2.8. nodaļai, rindkopās "Metodes soļa implementācija". Šajā nodaļā tiek aprakstīts organizācijas pašreizējais stāvoklis, metodes darbības rezultāti, izmantotās tehnoloģijas, datu iegūšana un pilnveidots organizācijas datu noliktavas datu modelis.

Pastāvoša situācija organizācijā.

Pastāv organizācija, kas darbojās ēdināšanas jomā. Organizācijai ir sava informācijas sistēma, kas ļauj saglabāt informāciju par klienta veiktajām galdiņu rezervācijām, pasūtījumiem, personālu, pasūtītajiem ēdieniem, ēdienkarti, ēdienkartes ēdienu alergēniem un alergēnu tipiem. Organizācija pastāv datuve, skatīt att. 5.19. Datuvē ir viena fakta tabula, kurā tiek saglabāts fakts par pasūtījumu, un dimensiju tabulas d_table, d_custumer, d_food un laika dimensijas d_time, d_month un d_year.



Att. 5.19. Pastāvošais organizācijas datuves datu modelis

Datu iegūšana un izmantotās tehnoloģijas.

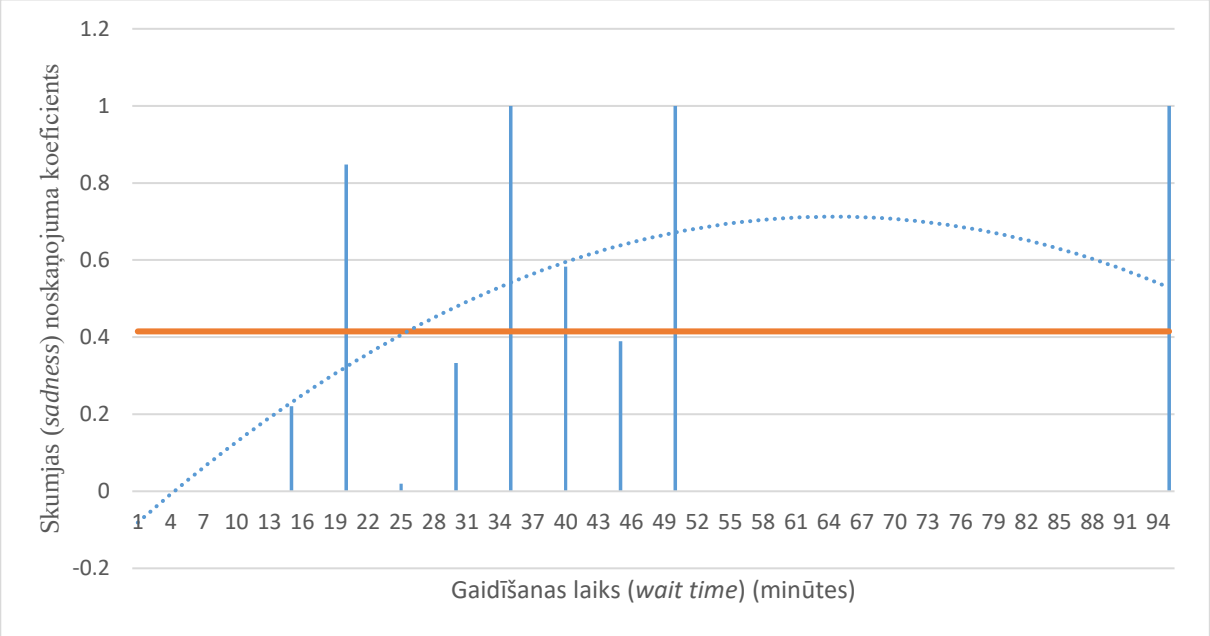
Metodes darbībai tika straumēti lietotāju tvīti no sociālās platformas Twitter. Tvīts ir lietotāja brīvi ierakstīts teksts, kas var būt līdz 280 simboliem garš. Datu iegūšana no Twitter notiek ar Python 3.8.13 bibliotēku Tweepy 3.10.0. Metodes aprobācijā tika ielasīti 1460364 tvīti, no tiem 114006 tvīti saturēja skaitliskas vērtības. Neapstrādāti tvīti tiek saglabāti HDFS 2.7.7. Tvīta teksts tiek apstrādāts Apache Spark 3.0.3, izmantojot[40] 2.12.10. SparkNLP 4.0.2 modeļi nodrošina noskaņojuma analīzi. Iegūtā informācija tiek saglabāta HDFS. Datu kalkulācijas notiek izmantojot Apache Spark, iegūtie starprezultāti un rezultāti tiek saglabāti HDFS.

[40] https://www.scala-lang.org

Metodes darbības rezultāti.

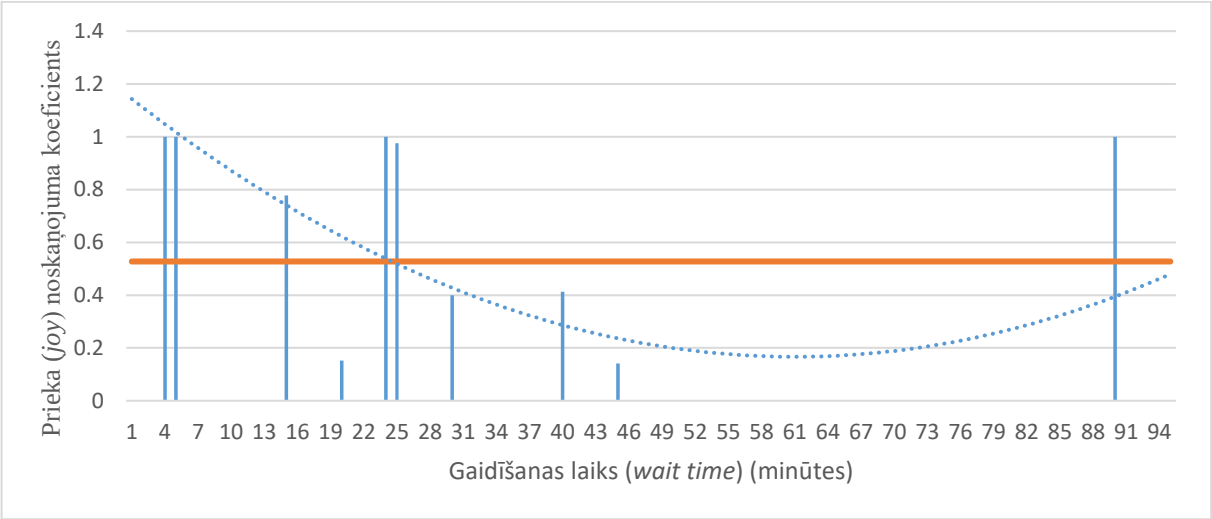
Iepriekšējās nodaļās tika aprakstītas metodes aprobācija uzņēmumā, kur tika aplūkots viens no biežāk lietotajiem vārdiem klientu atsauksmēs "wait time". Saistībā ar negatīvo noskaņojumu lietotāju atsauksmēs, tika iegūts jauns KPI un datu noliktavas datu modeļa atribūts – "wait time". Metode paredz, ka šādu saistību var veikt par citiem biežāk lietotajiem vārdiem klientu atsauksmēs. Šajā nodaļā tiek aplūkoti metodes darbības iegūtie rezultāti pēc tāda paša principa, kā iepriekšējās nodaļās aprakstītā.

Rezultāts "gaidīšanas laiks - skumjas" ("*wait time* - *sadness*") attiecībai. Att. 5.20. var redzēt vārda "wait time" un noskaņojuma "skumjas" saistību. Grafikā var redzēt, ka laikam ejot, noskaņojuma "skumjas" koeficients palielinās. Vidējā vērtība noskaņojumam "skumjas" lietotāju atsauksmēs ir apmēram 0.4, no zināmajām mērījumu vērtībām "biežāk lietoto lietvārdu vārdu saistību ar kādu mērījumu" tiek izrēķināta tendenču līnija. Taisnes un parabolas krustpunktā x ass vērtība ir 25. Pēc metodes posma "KPI nosacījumu noteikšana" noteiktā paņēmiena, noskaņojums "skumjas" ir pielīdzināms kā negatīvs noskaņojums, tas nozīmē, ka KPI nosacījumam ir jābūt mazākam (<). Šajā gadījumā lietvārds ir "wait time". Pēc metodē noteiktā, tiek iegūts vienkāršots KPI "wait time < 25 minutes".



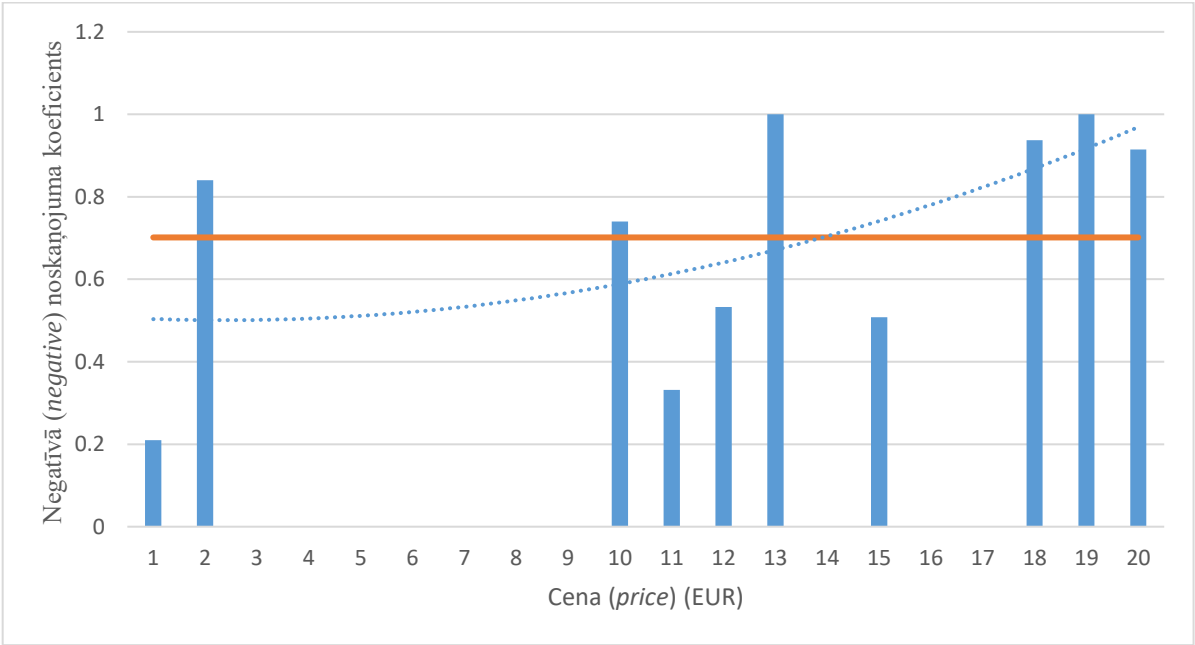
Att. 5.20. Noskaņojuma "Skumjas" koeficienta tendenču līnija un vidējās vērtības laikā

Rezultāts "gaidīšanas laiks - prieks" ("*wait time* - *joy*") attiecībai. Att. 5.21. var redzēt laika un noskaņojuma – "prieks" koeficientu saistību. Pēc metodē noteiktā paņēmiena tiek iegūts vienkāršots KPI "wait time < 25 minutes".



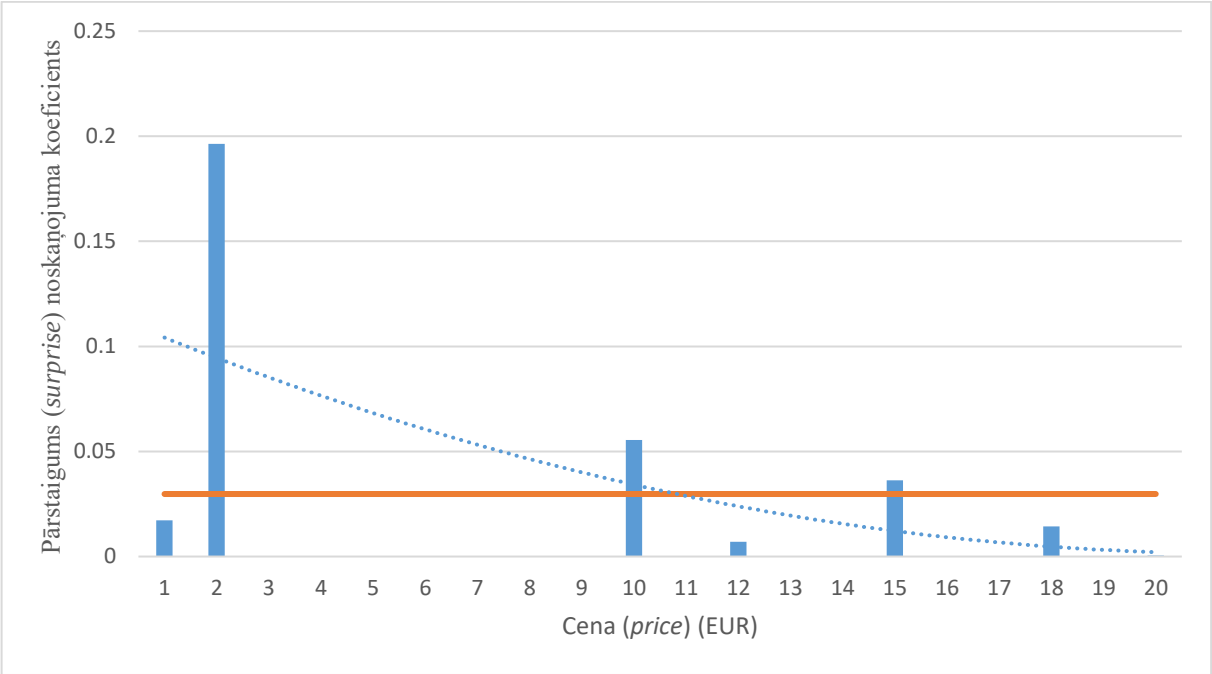
Att. 5.21. Noskaņojuma "Prieks" koeficienta tendenču līnija un vidējās vērtības laikā

Rezultāts "cena - negatīvs" ("*price - negative*") attiecībai. Att. 5.22**.** Var redzēt noskaņojuma –"negatīvs" koeficientu un cenas saistību. Pēc metodē noteiktā paņēmiena tiek iegūts vienkāršots KPI "price < 14 EUR".



Att. 5.22. Noskaņojuma "Negatīvs" koeficienta tendenču līnija un vidējās vērtības attiecībā pret cenu

Rezultāts "cena - pārsteigums" (*"price - surprise"*) attiecībai. Att. 5.23. var redzēt laika un noskaņojuma – "pārsteigums" koeficientu. Pēc metodē noteiktā paņēmiena tiek iegūts vienkāršots KPI "price < 11 EUR".

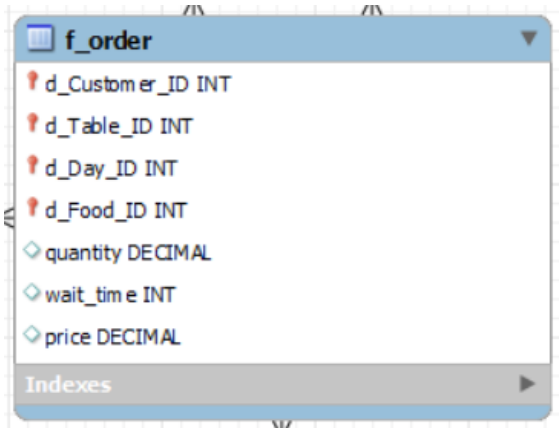


Att. 5.23. Noskaņojuma "Pārsteigums" koeficienta tendenču līnija un vidējās vērtības attiecībā pret cenu

Informācijas prasību un datu noliktavas modeļa pilnveidošana.

No iegūtajiem KPI pēc metodē noteiktajiem paņēmieniem tiek iegūtas informācijas prasības datu noliktavas datu modelim. No KPI "wait time < 25 minutes" un "wait time < 25 minutes" tiek iegūta vienkārša datu noliktavas datu modeļa prasība attēlot atribūtu "wait time", bet no KPI "price < 14 EUR" un "price < 11 EUR" tiek iegūta vienkārša datu noliktavas datu modeļa prasība attēlot atribūtu "price".

Att. 5.24. var redzēt, kā datu noliktavas datu modeļa faktu tabula ir papildināta ar diviem atribūtiem "wait_time" un "price". Konkrētajā gadījumā tika atrasti divi atribūti, bet ielasot papildus informāciju, var atrast jaunus atribūtus.



Att. 5.24. Organizācijas fakta tabula f_order papildināta ar diviem atribūtiem

5.3 Secinājumi

Šajā nodaļā tika aprakstītas divas jaunas metodes. Metode "KPI noteikšana izmantojot nestrukturētu tekstu", kas aprakstīta nodaļā 5.1. nodrošina funkcionalitāti specificēt jaunus organizācijas KPI, izmantojot klientu atsauksmes – nestrukturētu tekstu. Metode paredz apstrādāt klientu atsauksmes ar NLP rīku. No iegūtās informācijas tiek veikti speciāli aprēķini, kuru rezultāti tiek izmantoti, lai ģenerētu vārdu saistību grafus. Šie grafi tiek ģenerēti atbilstoši lietotāja ievadītajiem parametriem. Informāciju, kas attēlota grafā lietotājam ir iespējams apskatīties detalizēti, apskatot detalizētus tvītus, kuros pastāvēja vārdu saistība. Metodes lietotājam dod iespēju veidot jaunus grafus, kuros ir iespējams pievienot tikai lietotāja izvēlētus vārdus un vārdu relācijas, ļaujot novērot interesējošas saistības. Izmantojot grafa funkcionalitāti, organizācijas darbinieks var specificēt jaunus KPI un to mērķu vērtības. Metodes aprobācijā ir aprakstīts, kā specificēt jaunu kvantitatīvu un kvalitatīvu KPI un to mērķu vērtības.

Nodaļā 5.2. tika aprakstīta jauna metode "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm". Šī metode izmanto atsevišķus soļus no metodes "KPI noteikšana izmantojot nestrukturētu tekstu". Šī metode nodrošina:

- Klientu atsauksmju datu analīzes iespējas;
- Daļēji automātisku kvantitatīvu KPI specificēšanu;
- Daļēji automātisku KPI mērķu vērtību noteikšanu;
- Principus datu noliktavas datu modeļa papildināšanai.

Šīs metodes mērķis ir nodrošināt funkcionalitāti datu noliktavas datu modeļa uzlabošanai, izmantojot klientu atsauksmes – nestrukturētu tekstu. Metode nosaka, ka lietotāju atsauksmes tiek apstrādātas ar NLP rīku. No klientu atsauksmēm tiek iegūta informācija par vārdu saistībām teikumā un vārdu noskaņojumiem. Metode nosaka veikt speciālas kalkulācijas, lai ģenerētu kvantitatīvos KPI un to mērķu vērtības. Iegūtie atribūti no ģenerētajiem KPI ir jāiekļauj datu noliktavas datu modelī, lai organizācijas varētu uzraudzīt organizācijas veiktspēju atbilstoši KPI mērķa vērtībām. Saglabājot šos atribūtus datu noliktavā, organizācijas varēs sekot līdzi organizācijas attīstībai un uzlabot tās darbību. Šī metode nenodrošina ģenerēt datu noliktavas modeli no pirmsākumiem, bet metode sniedz iespēju paplašināt esošo datu modeli ar jauniem atribūtiem.

Lai organizācija pielietotu metodi, tai jābūt pieejamiem konkrētās vai citas organizācijas klientu atsauksmju datiem par jomu, kurā tā darbojās. Var izmantot arī publisku datu avotu, kā tas ir darīts šis metodes aprobācijā, kur tika izmantoti Twitter platformas dati. Lai no klientu atsauksmēm varētu ģenerēt kvantitatīvus KPI, klientu datiem ir jāsatur

skaitliskas vērtības, jo šis skaitliskās vērtības tiek izmantotas KPI mērķa vērtību noskaidrošanā.

Metodes aprobācijā aprakstīta organizācija, kas darbojās ēdināšanas jomā. Aprobācijā tiek lietoti Twitter dati kā datu avots. Aprobācijas rezultātā tika noskaidroti divi jauni datu noliktavas datu modeļa atribūti, pieci kvantitatīvi KPI un to mērķa vērtības.

Šī ir metodes pirmā versija, to ir iespējams attīstīt vairākos aspektos:

- Šobrīd metodes posmi ir jāizpilda manuāli, līdzīgi kā tas tika veikts metodes aprobācijā. Metodi varētu attīstīt automatizējot metodes posmu izpildi;
- Papildināt metodi nodrošinot kvalitatīvo KPI specificēšanu, šobrīd metode nodrošina tikai kvantitatīvo KPI sastādīšanu, bet no klientu atsauksmēm ir iespējams iegūt kvalitatīvos KPI;
- Attīstīt metodes posmu "Datu noliktavas datu modeļa uzlabojumi", šobrīd jauno atribūtu integrācija eksistējošajā datu modelī notiek manuāli, bet šo integrāciju varētu automatizēt;
- Attīstīt metodes kalkulācijas posmus, lai aprēķinos iekļautu arī neitrālus noskaņojumus un atsauksmes izveidošanas lokāciju, piemēram ņemt vērā tās atsauksmes kuras veiktas reģionā kurā darbojās organizācija;
- Attīstīt metodes kalkulācijas posmus, lai sinonīmi tiktu aprēķināti kopā, šobrīd sinonīmi tiek aprēķināti kā atsevišķi vārdi;
- Lai iegūtu precīzāku rezultātu, iespējams, būtu noderīgi izkļaut "saikļus" un konkrētās nozares biežāk lietotos vārdus.

REZULTĀTI

Promocijas darbā izvirzītās tēzes un mērķis ir sasniegts, jo autors ir izstrādājis divas jaunas metodes KPI specificēšanai un datu noliktavas datu modeļa paplašināšanai, kā arī ir paplašinājis eksistējošu datu noliktavas izstrādes metodi un izstrādājis rīku, kas nodrošina ģenerēt datu noliktavas kaditātshēmas, izmantojot formālās prasības.

Promocijas darba otrajā nodaļā ir aprakstīta paplašinātā metode un paplašināts metamodelis datu noliktavas prasību formalizēšanai. Metode tika paplašināta ar jaunām komponentēm - "Grafiskā saskarne formālo prasību ievadei", "Formālo prasību savienošanas komponente" un "Grafu datubāzes komponente", bet formālais prasību metamodelis ar jaunām klasēm "Ieinteresētā persona" un "Biznesa process". Lai īstenotu metodes darbību un ļautu ģenerēt datu noliktavas kandidātshēmas, tika izstrādāts iReq rīks, kas nodrošina:

- Ievadīt un saglabāt formālās prasības, atbilstoši paplašinātajam datu noliktavas formālo prasību metamodelim formālo prasību repozitorijā;
- Ģenerēt datu noliktavas kanditātshēmas, izmantojot datus no formālo prasību repozitorija.

Autors 5. nodaļā piedāvā divas jaunas metodes, kas, apstrādājot nestrukturētus datus, ļauj paplašināt organizācijā eksistējošas datu noliktavas datu modeli un specificēt jaunus organizācijas KPI. Metode "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm", kas aprakstīta nodaļā 5.2., izmanto atsevišķus soļus no metodes "KPI noteikšana izmantojot nestrukturētu tekstu", kas aprakstīta nodaļā 5.1.

` Nodaļā 5.1 aprakstītā metode paredz:

- Analizēt lietotāju atsauksmes par jomu, kurā darbojās organizācija. Lietotāju atsauksmes parasti ir brīva teksta formātā. Pastāv vairākas iespējas, ko organizācija var izmantot kā datu avotu, lai iegūtu atsauksmes un komentārus: a) klientu anketēšana, b) sūdzību/ierosinājumu uzklausīšana, c) atsauksmes par organizāciju vai nozari ārpus konkrētās organizācijas, piemēram, Twitter dati;
- Apstrādāt lietoju atsauksmes ar NLP rīkiem. Metode paredz izmantot kādu no NLP bibliotēkām (StandfordCoreNLP, NLTK, SparkNLP, u.c), lai iegūtu vārdu saistības teikumā;
- Veikt speciālas kalkulācijas ar datiem, lai nodrošinātu iespēju ģenerēt vārdu saistību grafu "*Parastais grafs*", uzlabotu ātrdarbību un samazinātu algoritmu sarežģītību nākamajos metodes soļos;
- Veidot jaunus "*Personalizētos grafus*", kas satur tikai lietotāja atlasītus elementus;

- Izmantot vārdu saistību grafus "*Parastais grafs*" un "*Personalizētos grafus*", lai specificētu jaunus organizācijas KPI.

Nodaļā 5.2 aprakstītā metode paredz:

- Analizēt lietotāju atsauksmes par jomu, kurā darbojās organizācija;
- Apstrādāt lietotāju atsauksmes ar NLP rīkiem, lai iegūtu tekstam noskaņojumu koeficientus un teikuma vārdu savstarpējās saistības;
- Veikt speciālas kalkulācijas ar datiem, kas iegūti pēc lietotāju atsauksmju apstrādes ar NLP rīkiem. Datu kalkulāciju notiek divos posmos. Pirmais posms nosaka aprēķināt "biežāk izmantotos vārdus ar vārdšķiru "lietvārds"" par konkrētu noskaņojumu, piemēram, negatīvu noskaņojumu. Otrais posms nosaka aprēķināt "biežāk lietoto lietvārdu vārdu saistību ar kādu mērījumu". Šī informācija tiek izmantota vēlākajos metodes posmos;
- Noteikt KPI tēmas. Klientu atsauksmēs un komentāros par konkrētu tematu, kuri ir izteikti pozitīvi vai negatīvi, ir svarīgas detaļas (informācija), jo tās ir raisījušas organizācijas klientu emocijas – pozitīvas vai negatīvas, un tieši šīs atsauksmes ir tās, par kurām ir jāveido jauni KPI organizācijas attīstībai. Tas nozīmē, ka biežāk lietotie lietvārdi lietotāju atsauksmēs un komentāros nosaka tematus, par kuriem ir nepieciešams veidot jaunus KPI;
- Aprēķināt KPI kvantitatīvās vērtības. Princips, kā noteikt automātiskā veidā KPI kvantitatīvo vērību, ir noteikt mērķi, ka KPI vērtībai ir jābūt ne lielākai kā vidējai vērtībai pret kopējo trendu no lietotāju atsauksmēs iegūtajām koeficientu vērtībām par konkrētu tematu. Lai noskaidrotu šo vērtību ir jāaprēķina vidējā vērtība un tendenču līnija, šo līniju krustpunkts norāda KPI mērķa vērtību;
- Definēt jaunus KPI un KPI mērķa vērtības;
- Paplašināt datu noliktavas datu modeli, izmantojot atribūtus no KPI.

Izstrādātā metode "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm" ir piemērota organizācijām, kurām ir pieejami klientu atsauksmju dati, pastāv datu noliktava un organizācijā tiek veikta KPI novērošana. Metode nenodrošina ģenerēt datu noliktavas modeli no pirmsākumiem, bet metode sniedz iespēju paplašināt esošo datu modeli ar jauniem atribūtiem. Lai organizācijā pielietotu metodi, tai jābūt pieejamiem konkrētās vai citas organizācijas klientu atsauksmju datiem par jomu, kurā tā darbojās, var izmantot arī publisku datu avotu, kā tas ir darīts šīs metodes aprobācijā, kur tika izmantoti Twitter dati. Lai no klientu atsauksmēm varētu ģenerēt kvantitatīvus KPI, klientu atsauksmju datiem ir jāsatur skaitliskas

vērtības. Metodes aprobācijā ir aprakstīts par organizāciju, kas darbojās ēdināšanas jomā. Aprobācijas rezultātā tika noskaidroti divi jauni datu noliktavas datu modeļa atribūti, pieci kvantitatīvi KPI un to mērķa vērtības.

SECINĀJUMI

Autors izpētīja literatūru par datu noliktavu izstrādes metodēm, informācijas prasībām un prasībām lielo datu laikmetā, aplūkoja tematu par tekstu kā informācijas avotu un KPI kā informācijas prasību veidu.

Attīstoties lielo datu tehnoloģijām un biznesa vajadzībām, ir mainījušās datu noliktavas un to arhitektūra, salīdzinot ar tradicionālām trīs līmeņu datu noliktavām. Attīstoties tādām datubāzes pārvaldības sistēmām kā NoSQL un grafu datubāzes, datu noliktavas modelēšanas metodes liek akcentu uz datiem un avotu shēmām. Tomēr literatūras apskats liecina, ka joprojām ir svarīgi iegūt datu noliktavu ieinteresēto personu informācijas prasības, jo datu noliktavu biežākais neveiksmes iemesls ir informācijas prasībām neatbilstošas datu noliktavas.

Promocijas darba tika izvirzītas sekojošas tēzes:

- Datu noliktavas prasības ir iespējams formalizēt, un no formālajām prasībām ir iespējams ģenerēt datu noliktavas kandidātshēmas;
- Brīvais teksts var tikt izmantots kā datu avots, lai specificētu jaunus organizācijas KPI, kas var kalpot kā informācijas prasības, lai paplašinātu datu noliktavas datu modeli.

Promocijas darbā izvirzītās tēzes un mērķis "Izstrādāt metodes, kas atbalsta datu noliktavas prasību iegūšanas posmu un paplašina datu noliktavas datu modeli, izmantojot nestrukturētus datus un valodas apstrādes tehnoloģijas" ir sasniegts, jo promocijas darba ietvaros ir:

- Paplašināta metode, kas nodrošina ģenerēt datu noliktavas kandidātshēmas, izmantojot formālās datu noliktavas prasības;
- Izstrādātas divas jaunas metodes, kas nodrošina specificēt KPI un paplašināt datu noliktavas datu modeli, izmantojot klientu atsauksmes.

Tradicionālas datu noliktavas kontekstā īpaša nozīme tiek pievērsta informācijas prasībām, jo tās nosaka, kādi dati ir jāielasa datu noliktavā. Dati, kas ir ielasīti datu noliktavā nodrošina organizācijas darbiniekiem iegūt vērtīgu informāciju, lai pieņemtu svarīgus lēmumus organizācijas attīstībai. Tiek aplūkotas pastāvošās datu noliktavas prasību vākšanas metodes. Viena no metodēm, kas nosaka datu noliktavas prasības formalizēt, tiek paplašināta ar jaunām komponentēm un formālo prasību metamodeļa klasēm. Autors ir izstrādājis praktisku rīku "iReq", kas nodrošina ģenerēt datu noliktavas kadidātshēmas modeļus daļēji automātiskā veidā. Izmantojot paplašināto metodi, tiek apstrādāti tikai strukturēti dati (atbilstoši formālo prasību metamodelim). Autors apskata iespējas līdzīgu funkcionalitāti nodrošināt, izmantojot nestrukturētus datus – brīvo tekstu.

Brīvais teksts ir viens no nestrukturēto datu veidiem, kas var tikt apstrādāts ar lielo datu tehnoloģijām. Apkopojot pētījumus par lielo datu un prasību inženieriju, var secināt, ka lielo datu apstrādes metodes var tikt attiecinātas kādam no prasību inženierijas posmiem. Lielo datu apstrādes metodes izmanto lielo datu tehnoloģijas dažādiem mērķiem, bet šobrīd nepastāv metodes, kas nosaka kā analizēt datus, lai definētu jaunas datu noliktavas informācijas prasības. Lielo datu apstrādes metodes var nebūt saistītas ar prasību inženieriju, bet, izmantojot lielo datu apstrādes metodēs aprakstītas tehnoloģijas, to darbības var tikt attiecinātas uz kādu no prasību inženierijas posmiem. Metodēs pielietotās tehnoloģijas var tikt izmantotas automātiskai vai daļēji automātiskai informācijas prasību iegūšanai, bet šobrīd neviena no metodēm nenodrošina šādu iespēju.

KPI nosaka darbības, kas var būtiski uzlabot organizācijas darbību. Lai veiktu KPI novērošanu, ir nepieciešami dati, kas nosaka rādītāja stāvokli, no tā var secināt, ka nepieciešamie dati KPI novērošanai var kalpot kā organizācijas informācijas prasības.

Autors ir izstrādājis divas jaunas metodes "KPI noteikšana izmantojot nestrukturētu tekstu" un "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm", kas paredz analizēt brīvo tekstu, lai nodrošinātu iespēju specificēt jaunus KPI un paplašināt organizācijā eksistējošas datu noliktavas datu modeli.

Metode "Datu noliktavas datu modeļa uzlabojumi no klientu atsauksmēm" nodrošina funkcionalitāti datu noliktavas datu modeļa uzlabošanai, izmantojot klientu atsauksmes – nestrukturētu tekstu. Metode nosaka, ka lietotāju atsauksmes tiek apstrādātas ar NLP tehnoloģijām. No klientu atsauksmēm tiek iegūta informācija par vārdu saistībām teikumā un klientu atsauksmju noskaņojumiem. Metode nosaka veikt speciālas kalkulācijas, lai ģenerētu KPI un to kvantitatīvās vērtības. Iegūtie atribūti no ģenerētajiem KPI ir jāiekļauj datu noliktavas datu modelī, lai organizācijas varētu uzraudzīt veiktspēju atbilstoši KPI mērķa vērtībām. Saglabājot šos atribūtus datu noliktavā, organizācijas varēs sekot līdzi organizācijas attīstībai un uzlabot tās darbību. Lai organizācija pielietotu metodi, tai jābūt pieejamiem konkrētās vai citas organizācijas klientu atsauksmju datiem par jomu kurā tā darbojās, var izmantot arī publisku datu avotu, kā tas ir darīts šis metodes aprobācijā, kur tika izmantoti Twitter dati. Promocijas darbā izstrādātā metode var tikt uzlabota vairākos virzienos, piemēram, attīstot NLP tehnoloģiju lietojumu. Lai no klientu atsauksmēm varētu ģenerēt kvantitatīvus KPI, klientu atsauksmēm ir jāsatur skaitliskas vērtības.

IZMANTOTĀ LITERATŪRA

Abdelhedi, F., Jemmali, R., & Zurfluh, G. (2022). Relational Databases Ingestion into a NoSQL Data Warehouse. arXiv preprint arXiv:2203.06949.

Agüero-Torales, M. M., Cobo, M. J., Herrera–Viedma, E., & López-Herrera, A. G. (2019). A cloud-based tool for sentiment analysis in reviews about restaurants on TripAdvisor. Procedia Computer Science, 162, 392–399.

Akid, H., Frey, G., Ayed, M. B., & Lachiche, N. (2022). Performance of NoSQL Graph Implementations of Star vs. Snowflake Schemas. IEEE Access, 10, 48603-48614.

Al-Okaily, A., Al-Okaily, M., Teoh, A. P., & Al-Debei, M. M. (2022). An empirical study on data warehouse systems effectiveness: the case of Jordanian banks in the business intelligence era. EuroMed Journal of Business.

Ardagna, C., Ceravolo, P., Cota, G. L., Kiani, M. M., & Damiani, E. (2017, June). What are my users looking for when preparing a big data campaign. In 2017 IEEE International Congress on Big Data (BigData Congress) (pp. 201-208). IEEE.

Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021, January). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR (Vol. 8).

Arruda, D., & Madhavji, N. H. (2017, December). Towards a requirements engineering artefact model in the context of big data software development projects: Research in progress. In 2017 IEEE international conference on big data (big data) (pp. 2314-2319). IEEE.

Attaran, M., Stark, J., & Stotler, D. (2018). Opportunities and challenges for big data analytics in US higher education: A conceptual model for implementation. Industry and Higher Education, 32(3), 169-182.

Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies?. *Communications of the ACM*, *46*(1), 78-82.

Baralis, E., Dalla Valle, A., Garza, P., Rossi, C., & Scullino, F. (2017, December). SQL versus NoSQL databases for geospatial applications. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 3388-3397). IEEE.

Baviskar, D., Ahirrao, S., & Kotecha, K. (2021). Multi-layout unstructured invoice documents dataset: A dataset for template-free invoice processing and its evaluation using AI approaches. IEEE Access, 9, 101494-101512.

Benkhaled, H. N., & Berrabah, D. (2019). Data Quality Management For Data Warehouse Systems: State Of The Art. JERI.

Bimonte, S., Antonelli, L., & Rizzi, S. (2021). Requirements-driven data warehouse design based on enhanced pivot tables. Requirements Engineering, 26, 43-65..

Bimonte, S., Boulil, K., Pinet, F., & Kang, M. A. (2013, July). Design of complex spatio-multidimensional models with the ICSOLAP UML profile-an implementation in MagicDraw. In International Conference on Enterprise Information Systems (Vol. 2, pp. 310-315). SCITEPRESS.

Birnbaum, D., Ely, J. W., Dawson, J. D., Lemke, J. H., & Rosenberg, J. (1997). An introduction to time-trend analysis. Infection Control & Hospital Epidemiology, 18(4), 267-274.

Bode, J., Kühl, N., Kreuzberger, D., & Hirschl, S. (2023). Data Mesh: Motivational Factors, Challenges, and Best Practices. arXiv preprint arXiv:2302.01713.

Bouaziz, S., Nabli, A., & Gargouri, F. (2019). Design a data warehouse schema from document-oriented database. Procedia Computer Science, 159, 221-230.

Brito, E., Sifa, R., Bauckhage, C., Loitz, R., Lohmeier, U., & Pünt, C. (2019, September). A hybrid AI tool to extract key performance indicators from financial reports for benchmarking. In Proceedings of the ACM Symposium on Document Engineering 2019 (pp. 1-4).

Brown, B., Chui, M., & Manyika, J. (2011). Are you ready for the era of 'big data'. McKinsey Quarterly, 4(1), 24-35.

Bruckner, R., List, B., & Scheifer, J. (2001). Developing requirements for data warehouse systems with use cases. AMCIS 2001 Proceedings, 66.

Castellanos, M., Simitsis, A., Wilkinson, K., & Dayal, U. (2009, March). Automating the loading of business process data warehouses. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (pp. 612-623).

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Kurzweil, R. (2018). Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Chebotko, A., Kashlev, A., & Lu, S. (2015, June). A big data modeling methodology for Apache Cassandra. In 2015 IEEE International Congress on Big Data (pp. 238-245). IEEE.

Cheptsov, A., Tenschert, A., Schmidt, P., Glimm, B., Matthesius, M., & Liebig, T. (2014). Introducing a new scalable data-as-a-service cloud platform for enriching traditional text mining techniques by integrating ontology modelling and natural language processing. In Web Information Systems Engineering–WISE 2013 Workshops: WISE 2013 International Workshops BigWebData, MBC, PCS, STeH, QUAT, SCEH, and STSC 2013, Nanjing, China, October 13-15, 2013, Revised Selected Papers 14 (pp. 62-74). Springer Berlin Heidelberg..

Cherradi, M., & EL Haddadi, A. (2021, November). Data lakes: A survey paper. In The Proceedings of the International Conference on Smart City Applications (pp. 823-835). Cham: Springer International Publishing..

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., & Tournier, R. (2015). Implementing multidimensional data warehouses into NoSQL..

Chianese, A., Marulli, F., & Piccialli, F. (2016, February). Cultural heritage and social pulse: a semantic approach for CH sensitivity discovery in social media data. In 2016 IEEE Tenth International Conference on Semantic Computing (ICSC) (pp. 459-464). IEEE..

Chowdhary, K., & Chowdhary, K. R. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649..

Danaher, P. J., Smith, M. S., Ranasinghe, K., & Danaher, T. S. (2015). Where, when, and how long: Factors that influence the redemption of mobile phone coupons. Journal of Marketing Research, 52(5), 710-725.

De Marnee, M. C., & Manning, C. D. (2008). Stanford typed dependencies manual (pp. 338-345). Technical report, Stanford University.

Dehghani, Z. (2022). Data Mesh. " O'Reilly Media, Inc.".

Denecke, K. (2014). Extracting medical concepts from medical social media with clinical NLP tools: a qualitative study. In Proceedings of the fourth workshop on building and evaluation resources for health and biomedical text processing (pp. 54-60).

Devlin, B. (1996). Data warehouse: from architecture to implementation. Addison-Wesley Longman Publishing Co., Inc..

Di Tria, F., Lefons, E., & Tangorra, F. (2014, October). Design process for big data warehouses. In 2014 International Conference on Data Science and Advanced Analytics (DSAA) (pp. 512-518). IEEE.

Diebold, F. X. (2010). “Big Data” Dynamic Factor Models for Macroeconomic Measurement and Forecasting: A discussion of the papers by Lucrezia Reichlin and by Mark W. Watson. In Cambridge University Press eBooks (pp. 115–122). https://doi.org/10.1017/cbo9780511610264.005

Domínguez, E., Pérez, B., Rubio, Á. L., & Zapata, M. A. (2019). A taxonomy for key performance indicators management. Computer Standards & Interfaces, 64, 24-40.

Eggert, M., & Alberts, J. (2020). Frontiers of business intelligence and analytics 3.0: a taxonomy-based literature review and research agenda. Business Research, 13(2), 685-739.

Elhadad, N., Gravano, L., Hsu, D., Balter, S., Reddy, V., & Waechter, H. (2014). Information extraction from social media for public health. In KDD at Bloomberg Workshop, Data Frameworks Track (KDD 2014).

Eridaputra, H., Hendradjaya, B., & Sunindyo, W. D. (2014, November). Modeling the requirements for big data application using goal oriented approach. In 2014 international conference on data and software engineering (ICODSE) (pp. 1-6). IEEE.

Fan, W., & Bifet, A. (2013). Mining big data: current status, and forecast to the future. ACM SIGKDD explorations newsletter, 14(2), 1-5.

Farzindar, A., Inkpen, D., & Hirst, G. (2015). Natural language processing for social media. San Rafael: Morgan & Claypool.

Favaretto, M., De Clercq, E., Schneble, C. O., & Elger, B. S. (2020). What is your definition of Big Data? Researchers' understanding of the phenomenon of the decade. PloS one, 15(2), e0228987.

Feldman, S. (1999). NLP meets the Jabberwocky: Natural language processing in information retrieval. ONLINE-WESTON THEN WILTON-, 23, 62-73.

Fernandez-Garcia, A. J., Iribarne, L., Corral, A., & Wang, J. Z. (2015). Evolving mashup interfaces using a distributed machine learning and model transformation methodology. In On the Move to Meaningful Internet Systems: OTM 2015 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, EI2N, FBM, INBAST, ISDE, META4eS, and MSC 2015, Rhodes, Greece, October 26-30, 2015. Proceedings (pp. 401-410). Springer International Publishing.

Ferreira, L. M., Alves-Souza, S. N., & da Silva, L. M. (2022). Startable: Multidimensional Modelling for Column-Oriented NoSQL.

Giorgini, P., Rizzi, S., & Garzetti, M. (2008). GRAnD: A goal-oriented approach to requirement analysis in data warehouses. Decision Support Systems, 45(1), 4-21.

Giovinazzo, W. A. (2000). Object-oriented data warehouse design: building a star schema. Prentice Hall PTR.

Golfarelli, M., & Rizzi, S. (2009). Data warehouse design: Modern principles and methodologies. McGraw-Hill, Inc..

Golfarelli, M., Maio, D., & Rizzi, S. (1998). The dimensional fact model: A conceptual model for data warehouses. International Journal of Cooperative Information Systems, 7(02n03), 215-247.

Grishman, R., & Kittredge, R. (Eds.). (2014). Analyzing language in restricted domains: sublanguage description and processing. Psychology Press.

Guide, P. M. B. O. K. (2008). A guide to the project management body of knowledge.

Gupta, S., & Giri, V. (2018). Practical enterprise data lake insights.

Hai, R., Quix, C., & Jarke, M. (2021). Data lake concept and systems: a survey. arXiv preprint arXiv:2106.09592.

Halim, S., Mubarokah, I., & Hidayanto, A. N. (2020, November). Rank critical success factors (CSFs) of data warehouse and business intelligence (DW/BI) implementation in banking sector using analytical hierarchy process (AHP). In 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) (pp. 313-318). IEEE.

Hill, K. Q. (1978). Trend extrapolation. Handbook of Futures Research, 249.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. Science, 349(6245), 261-266.

Howatson, A. (2016). How to unlock the power of unstructured data. Marketing Tech News.

Hu, X., Yi, W., Jiang, L., Wu, S., Zhang, Y., Du, J., Ma, T., Wang, T., & Wu, X. (2019). Classification of metaphase chromosomes using deep convolutional neural network. Journal of Computational Biology, 26(5), 473–484.

Ikegwu, A. C., Nweke, H. F., Anikwe, C. V., Alo, U. R., & Okonkwo, O. R. (2022). Big data analytics for data-driven industry: a review of data sources, tools, challenges, solutions, and research directions. Cluster Computing, 25(5), 3343-3387.

Inmon, B. (2016). Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump. Technics publications.

INMON, W. (1996). Building the Warehouse, 2nd.

Yang, Q., Ge, M., & Helfert, M. (2019). Analysis of Data Warehouse Architectures: Modeling and Classification.

Yasin, A., Liu, L., Cao, Z., Wang, J., Liu, Y., & Ling, T. S. (2018). Big data services requirements analysis. In Requirements Engineering for Internet of Things: 4th Asia-Pacific Symposium, APRES 2017, Melaka, Malaysia, November 9–10, 2017, Proceedings 4 (pp. 3-14). Springer Singapore.

Yessad, L., & Labiod, A. (2016, November). Comparative study of data warehouses modeling approaches: Inmon, Kimball and Data Vault. In 2016 International Conference on System Reliability and Science (ICSRS) (pp. 95-99). IEEE.

Jain, S. K., & Kumar, V. (2012). Trend analysis of rainfall and temperature data for India. Current Science, 37-49.

Jameel, K., Adil, A., & Bahjat, M. (2022). Analyses the Performance of Data Warehouse Architecture Types. Journal of Soft Computing and Data Mining, 3(1), 45-57.

Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2002). Fundamentals of data warehouses. Springer Science & Business Media.

Ji, J., & Peng, R. (2016, September). An analysis pattern driven requirements modeling method. In 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW) (pp. 316-319). IEEE.

Jukic, N., & Nicholas, J. (2010, June). A framework for requirement collection and definition process for data warehousing projects. In Proceedings of the ITI 2010, 32nd International Conference on Information Technology Interfaces (pp. 187-192). IEEE.

Kaplan, R. S. (1992). The balanced scorecard measures that drive performance. Harvard business review.

Kart, L., Heudecker, N., & Buytendijk, F. (2013). Survey analysis: big data adoption in 2013 shows substance behind the hype. Gartner Report GG0255160, 13.

Katal, A., Wazid, M., & Goudar, R. H. (2013, August). Big data: issues, challenges, tools and good practices. In 2013 Sixth international conference on contemporary computing (IC3) (pp. 404-409). IEEE.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Khan, N., Alsaqer, M., Shah, H., Badsha, G., Abbasi, A. A., & Salehian, S. (2018, March). The 10 Vs, issues and challenges of big data. In Proceedings of the 2018 international conference on big data and education (pp. 52-56).

Kim, Y., Kim, C. K., Lee, D. K., Lee, H. W., & Andrada, R. I. T. (2019). Quantifying nature-based tourism in protected areas in developing countries by using social big data. Tourism Management, 72, 249-256..

Kim, S., & Chung, J. E. (2011). Restaurant selection criteria: Understanding the roles of restaurant type and customers' sociodemographic characteristics.

Kimball, R., & Ross, M. (1996). The Data Warehouse Toolkit: the complete guide to dimensional modeling

Kimball, R., & Ross, M. (2011). The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons.

Kotonya, G., & Sommerville, I. (1998). Requirements engineering: processes and techniques. Wiley Publishing.

Kozmina, N., & Niedrite, L. (2014). Extending a metamodel for formalization of data warehouse requirements. In Perspectives in Business Informatics Research: 13th International Conference, BIR 2014, Lund, Sweden, September 22-24, 2014. Proceedings 13 (pp. 362-374). Springer International Publishing.

Kozmina, N., Niedrite, L., & Golubs, M. (2013, July). Deriving the Conceptual Model of a Data Warehouse from Information Requirements. In International Conference on Enterprise Information Systems (Vol. 2, pp. 136-144). SCITEPRESS.

Kozmina, N., Niedrite, L., & Zemnickis, J. (2017, April). Gathering Formalized Information Requirements of a Data Warehouse. In International Conference on Enterprise Information Systems (Vol. 2, pp. 217-224). SCITEPRESS.

Kozmina, N., Niedrite, L., & Zemnickis, J. (2018). Information requirements for big data projects: A review of state-of-the-art approaches. In Databases and Information Systems: 13th International Baltic Conference, DB&IS 2018, Trakai, Lithuania, July 1-4, 2018, Proceedings 13 (pp. 73-89). Springer International Publishing.

Kozmina, N., Niedrite, L., & Zemnickis, J. (2019). Perspectives of Information Requirements Analysis in Big Data Projects. In Databases and Information Systems X (pp. 109-124). IOS Press.

Krstev, S., & Krneta, D. (2022, March). Data Warehouse to Support Creation of Key Performance Indicators (KPIs) in Electricity Supply. In 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH) (pp. 1-4). IEEE.

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. META group research note, 6(70), 1.

Liddy, E. D. (1998). Enhanced text retrieval using natural language processing. Bulletin of the American Society for Information Science and Technology, 24(4), 14-16.

Liu, C., Yu, R., Zhang, J., Wei, S., Xue, F., Guo, Y., ... & Dong, W. (2022). Research hotspot and trend analysis in the diagnosis of inflammatory bowel disease: A machine learning bibliometric analysis from 2012 to 2021. Frontiers in Immunology, 13, 972079.

Liu, J., Shang, J., Wang, C., Ren, X., & Han, J. (2015, May). Mining quality phrases from massive text corpora. In Proceedings of the 2015 ACM SIGMOD international conference on management of data (pp. 1729-1744)..

Llave, M. R. (2018). Data lakes in business intelligence: reporting from the trenches. Procedia computer science, 138, 516-524.

Luján-Mora, S., Trujillo, J., & Song, I. Y. (2006). A UML profile for multidimensional modeling in data warehouses. Data & knowledge engineering, 59(3), 725-769..

Machado, I. A., Costa, C., & Santos, M. Y. (2022). Data mesh: concepts and principles of a paradigm shift in data architectures. Procedia Computer Science, 196, 263-271.

Madera, C., & Laurent, A. (2016, November). The next information architecture evolution: the data lake wave. In Proceedings of the 8th international conference on management of digital ecosystems (pp. 174-180).

Madhavji, N. H., Miranskyy, A., & Kontogiannis, K. (2015, May). Big picture of big data software engineering: with example research challenges. In 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering (pp. 11-14). IEEE.

Mallek, H., Ghozzi, F., Teste, O., & Gargouri, F. (2017). BigDimETL: ETL for multidimensional big data. In Intelligent Systems Design and Applications: 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016) held in Porto, Portugal, December 16-18, 2016 (pp. 935-944). Springer International Publishing.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

Michael, A. V., & Ahirao, P. (2020, April). Improved use of ETL tool for updation and creation of data warehouse from different RDBMS. In Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST).

Nambiar, A., & Mundra, D. (2022). An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management. Big Data and Cognitive Computing, 6(4), 132.

Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., & Arocena, P. C. (2019). Data lake management: challenges and opportunities. Proceedings of the VLDB Endowment, 12(12), 1986-1989.

Nasiri, A., Wrembel, R., & Zimányi, E. (2015). Model-based requirements engineering for data warehouses: from multidimensional modelling to KPI monitoring. In Advances in Conceptual Modeling: ER 2015 Workshops AHA, CMS, EMoV, MoBID, MORE-BI, MReBA, QMMQ, and SCME, Stockholm, Sweden, October 19-22, 2015, Proceedings 34 (pp. 198-209). Springer International Publishing.

Niedritis, A., Niedrite, L., & Kozmina, N. (2011). Performance measurement framework with formal indicator definitions. In Perspectives in Business Informatics Research: 10th International Conference, BIR 2011, Riga, Latvia, October 6-8, 2011. Proceedings 10 (pp. 44-58). Springer Berlin Heidelberg.

Nordeen, A. (2020). Learn Data Warehousing in 24 Hours. Guru99.

Parmenter, D. (2007). Key performance Indicators: Developing, implementing, and using winning KPIs.

Parmenter, D. (2015). Key performance indicators: developing, implementing, and using winning KPIs. John Wiley & Sons.

Pejić Bach, M., Krstić, Ž., Seljan, S., & Turulja, L. (2019). Text mining for big data analysis in financial sector: A literature review. Sustainability, 11(5), 1277.

Perry, J. S. (2017). What is big data? More than volume, velocity and variety. Retrieved April, 24, 2019.

Phipps, C., & Davis, K. C. (2002, May). Automating data warehouse conceptual schema design and evaluation. In DMDW (Vol. 2, pp. 23-32).

Pilipenko, O. V., Provotorova, E. N., Sergeev, S. M., & Rodionov, O. V. (2019, December). Automation engineering of adaptive industrial warehouse. In Journal of Physics: Conference Series (Vol. 1399, No. 4, p. 044045). IOP Publishing.

Pinto, A., Gonçalo Oliveira, H., & Oliveira Alves, A. (2016). Comparing the performance of different NLP toolkits in formal and social media text. In 5th Symposium on Languages, Applications and Technologies (SLATE'16). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Prakash, D., & Prakash, N. (2019). A multifactor approach for elicitation of Information requirements of data warehouses. Requirements Engineering, 24, 103-117.

Ravat, F., & Zhao, Y. (2019). Data lakes: Trends and perspectives. In Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30 (pp. 304-313). Springer International Publishing.

Rizkallah, J. (2017). The big (unstructured) data problem. Forbes. Retrieved on September, 5, 2017.

Romero, O., & Abelló, A. (2010). A framework for multidimensional design of data warehouses from ontologies. Data & Knowledge Engineering, 69(11), 1138-1157.

Rousseeuw, P. J., Raymaekers, J., & Hubert, M. (2018). A measure of directional outlyingness with applications to image data and video. Journal of Computational and Graphical Statistics, 27(2), 345-359.

Russell, M. A. (2013). Mining the social web: data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more. " O'Reilly Media, Inc.".

Santos, J. C., Mirakhorli, M., Mujhid, I., & Zogaan, W. (2016, April). BUDGET: A tool for supporting software architecture traceability research. In 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA) (pp. 303-306). IEEE.

Santos, M. Y., & Costa, C. (2016, July). Data warehousing in big data: from multidimensional to tabular data models. In Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering (pp. 51-60).

Schiefer, J., List, B., & Bruckner, R. (2002). A holistic approach for managing requirements of data warehouse systems. AMCIS 2002 Proceedings, 13.

Schuster, S., & Manning, C. D. (2016, May). Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 2371-2378).

Sellami, A., Nabli, A., & Gargouri, F. (2020, November). Graph NoSQL data warehouse creation. In Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services (pp. 34-38).

Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., & Marín-Torder, E. (2016, December). Towards a comprehensive data lifecycle model for big data environments. In Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (pp. 100-106).

Souibgui, M., Atigui, F., Zammali, S., Cherfi, S., & Yahia, S. B. (2019). Data quality in ETL process: A preliminary study. Procedia Computer Science, 159, 676-687.

Souza, V. E. S., Mazón, J., Garrigós, I., Trujillo, J., & Mylopoulos, J. (2012). Monitoring strategic goals in data warehouses with awareness requirements.

Tamrakar, A., Mewada, P., Gubrele, P., Prasad, R., & Saurabh, P. (2020). An ANN-based text mining approach over hash tag and blogging text data. In Soft Computing for Problem Solving: SocProS 2018, Volume 2 (pp. 399-408). Springer Singapore.

Tardio, R., Mate, A., & Trujillo, J. (2015, October). An iterative methodology for big data management, analysis and visualization. In 2015 IEEE International Conference on Big Data (Big Data) (pp. 545-550). IEEE.

Tikito, I., & Souissi, N. (2017, March). Data collect requirements model. In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications (pp. 1-7).

Tortonesi, M., Govoni, M., Morelli, A., Riberto, G., Stefanelli, C., & Suri, N. (2019). Taming the IoT data deluge: An innovative information-centric service model for fog computing applications. Future Generation Computer Systems, 93, 888-902.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In Proceedings of the 2003 human language technology conference of the north american chapter of the association for computational linguistics (pp. 252-259).

Vestola, M. (2010). A comparison of nine basic techniques for requirements prioritization. Helsinki University of Technology, 1-8.

Vicente Chicote, C., Moros Valle, B., & Toval Álvarez, A. (2007). REMM-Studio: an integrated model-driven environment for requirements specification, validation and formatting.

Vidhya, K., & Shanmugalakshmi, R. (2020). Modified adaptive neuro-fuzzy inference system (M-ANFIS) based multi-disease analysis of healthcare Big Data. The Journal of Supercomputing, 76(11), 8657-8678.

Vu, H. Q., Li, G., Law, R., & Zhang, Y. (2019). Exploring tourist dining preferences based on restaurant reviews. Journal of Travel Research, 58(1), 149-167.

Wang, J., Xu, C., Zhang, J., & Zhong, R. (2022). Big data analytics for intelligent manufacturing systems: A review. Journal of Manufacturing Systems, 62, 738-752.

Ward, J. S., & Barker, A. (2013). Undefined by data: a survey of big data definitions. arXiv preprint arXiv:1309.5821.

WEB (a). Tech Trends 2017: An Overview. https://deloitte.wsj.com/articles/tech-trends-2017-an-overview-1486530247, atsauce <2023.08.29>

WEB (b). CoNLL-2003. https://paperswithcode.com/dataset/conll-2003, atsauce <2023.08.29>

WEB (c). Stanford CoreNLP Online Tool. https://corenlp.run, atsauce <2023.08.29>

WEB (d). Detect emotions in tweets. https://demo.johnsnowlabs.com/public/SENTIMENT_EN_EMOTION, atsauce <2023.08.29>

WEB (e). NetApp's new CEO pushes ahead with data fabric strategy. https://www.zdnet.com/article/netapps-new-ceo-pushes-ahead-with-data-fabric-strategy, atsauce <2023.08.23>

WEB (f). 25. Most Important KPIs and Metrics for All Restaurant Businesses. https://www.poshighway.com/blog/25-most-important-KPIs-metrics-for-restaurant-businesses, atsauce <2023.08.29>

WEB (g). Language Network. https://www.cotrino.com/2012/11/language-network, atsauce <2023.08.29>

WEB (h). Sentiment Analysis of Tweets. https://nlp.johnsnowlabs.com/2021/01/18/sentimentdl_use_twitter_en.html, atsauce <2023.08.10>

WEB (i). Sarcasm Classifier. https://nlp.johnsnowlabs.com/2020/07/03/classifierdl_use_sarcasm_en.html, atsauce <2023.08.10>

WEB (j). Sarcasm-Detectio. https://github.com/MirunaPislar/Sarcasm-Detection/tree/master/res/datasets/riloff, atsauce <2023.08.10>

WEB (k). Stanford Dependencies. https://nlp.stanford.edu/software/stanford-dependencies.html, atsauce <2023.08.29>

WEB (l). Data Mesh in Practice: How Europe's Leading Online Platform for Fashion Goes Beyond the Data Lake. https://www.slideshare.net/databricks/data-mesh-in-practice-how-europes-leading-online-platform-for-fashion-goes-beyond-the-data-lake-236728093, atsauce <2023.08.23>

WEB (m). Typed Dependency Parsing for English. https://nlp.johnsnowlabs.com/2021/03/27/Typed_Dependency_Parsing_en.html, atsauce <2023.08.10>

Wu, D., & Guan, Y. (2021). Artificial intelligence retrieval algorithm for text data from multiple data sources. International Journal of Computers and Applications, 43(7), 715-719.

Zagan, E., & Danubianu, M. (2020, May). Data lake approaches: A survey. In 2020 International Conference on Development and Application Systems (DAS) (pp. 189-193). IEEE.

Zaidi, E., Thoo, E., De Simoni, G., & Beyer, M. (2019). Data Fabrics Add Augmented Intelligence to Modernize Your Data Integration. With Eric Thoo. Gartner Grou, 17.

Zemnickis, J. (2023). Data Warehouse Data Model Improvements from Customer Feedback. Baltic Journal of Modern Computing, 11(3).

Zemnickis, J., Niedrite, L., & Kozmina, N. (2020). A Little Bird Told Me: Discovering KPIs from Twitter Data. In Databases and Information Systems: 14th International Baltic Conference, DB&IS 2020, Tallinn, Estonia, June 16–19, 2020, Proceedings 14 (pp. 161-175). Springer International Publishing.

Zhang, Y., Chen, Y., & Ma, Y. (2015). A framework for data-driven automata design. In Requirements Engineering in the Big Data Era: Second Asia Pacific Symposium,

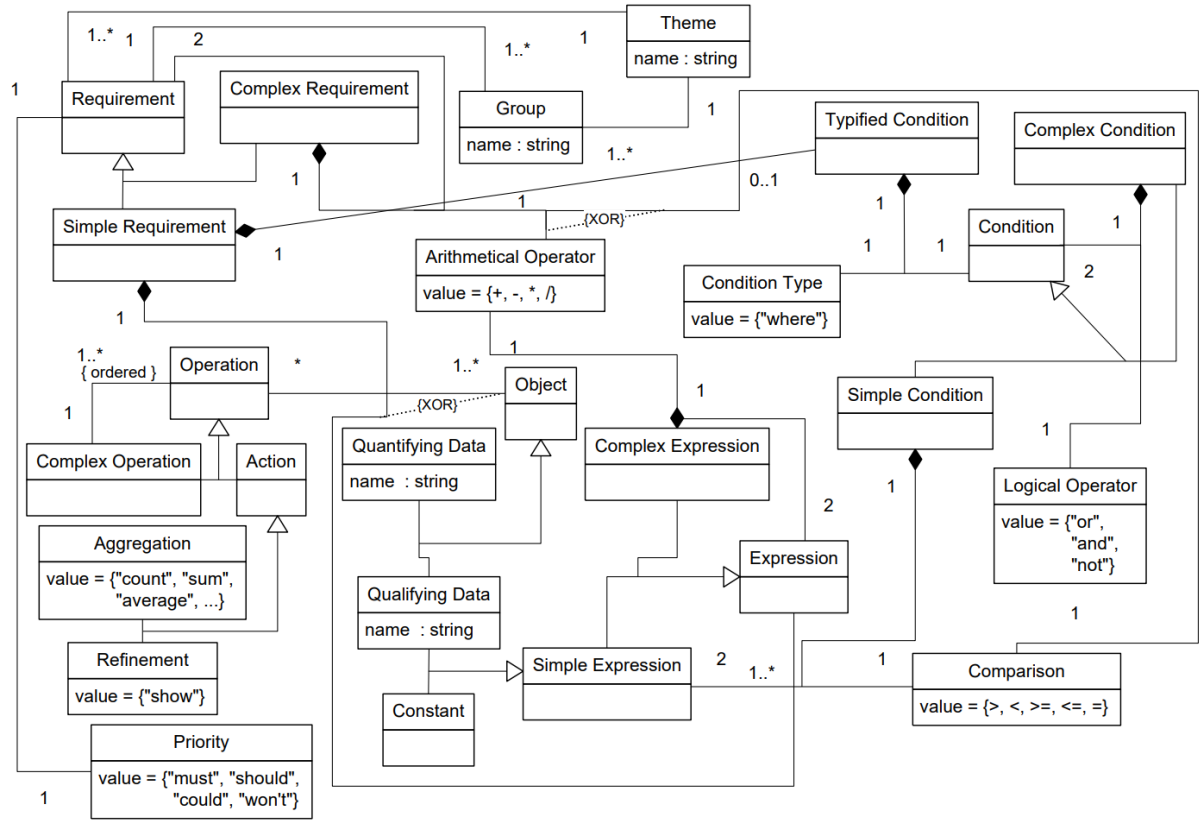
APRES 2015, Wuhan, China, October 18–20, 2015, Proceedings (pp. 33-47). Springer Berlin Heidelberg.

Zhu, Q., & Xiang, H. (2016). Differences of Pareto principle performance in e-resource download distribution: An empirical study. The electronic library, 34(5), 846-855.

PIELIKUMI

1. Pielikums

Pielikumā redzams orģinālais (nepaplašinātais) autoru (Kozmina et al., 2013) metamodelis, lai formalizētu datu noliktavas informācijas prasības. Promocijas darba ietvaros šis metamodelis tika paplašināts.



Att. 1.1 Autoru (Kozmina et al., 2013) piedāvātais metamodelis datu noliktavas prasību formalizēšanai