VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

# Rodiklių duomenų apdorojimo sistemų generavimas iš duomenų modelių

## Generation of Indicator Data Processing Systems from Data Models

Bakalauro baigiamasis darbas

Atliko: Vytautas Žilinas (parašas)

Darbo vadovas: lekt. Andrius Adamonis (parašas)

Recenzentas: assoc. prof., dr. Karolis Petrauskas (parašas)

Vilnius – 2019

# Santrauka

Šį darbą sudaro teorinė ir eksperimentinė dalys. Teorinėje dalyje apibrėžiami rodiklių duomenys ir rodiklių duomenų struktūros pokyčiai. Apibrėžiamas srautinis apdorojimas ir, atlikus „Apache Storm“, „Apache Spark“, „Apache Flink“ ir „Heron“ srautinio apdorojimo sprendimų palyginimą, pasirenkamas srautinio apdorojimo sprendimas, kuris bus naudojamas eksperimentinėje dalyje. Remiantis padaryta analize ir apsibrėžtomis savybėmis nustatoma, kad šiam uždaviniui tinkamas „Heron“ srautinio apdorojimo sprendimas. Sudaryta srautinio apdorojimo sistemų sugeneruotu pagal apibrėžta rodiklių duomenų modelį architektūra. Sudarytai architektūrai patikrinti daromas sukurtas eksperimentinis sprendimas. Atliekant rodiklių duomenų pokyčių simuliaciją stebėjimais analizuojamas šios architektūros tinkamumas apibrėžtam uždaviniui spręsti. Gauti eksperimento rezultatai išnagrinėjami, pateikiamos išvados. Taip įrodoma, kad toks sprendimas gali būti įgyvendinamas ir, kad kodo generavimas ir srautinio apdorojimo sprendimas „Heron“ yra tinkami keičiamų rodiklių uždaviniui spręsti.

**Raktiniai žodžiai: srautinis apdorojimas, kodo generavimas, rodiklis, rodiklių duomenų pokyčiai**

# Summary

This work consists of a theoretical and an experimental parts. Theoretical part defines indicator data and changes of indicator data structure. Stream processing is defined and ”Apache Storm”, ”Apache Spark”, ”Apache Flink” and ”Heron” stream processing solutions are compared. Based on the results ”Heron” stream processing solution is chosen to solve this task. The architecture of the streaming systems generated according to the defined data model of data is created. A pilot version of the solution is created based on the proposed architecture. By doing the simulation of changes in indicator data structure, the suitability of this system for a defined task is tested. The results of this research are compared and conclusions are given. This demonstrates that such a solution can be implemented and that the code generation and stream processing solution ”Heron” is the right solution to deal with the challenge of change in indicator structure.

**Keywords: stream processing, code generation, indicator, indactor data change**

**TURINYS**

# Įvadas

Šiame darbe yra nagrinėjamas rodiklių duomenų apdorojimas ir kuriamas eksperimentinis sprendimas, galintis prisitaikyti, kai keičiama rodiklių duomenų struktūra. Rodiklių duomenimis vadinkime pasikartojančių įvykių parametrus aprašančius duomenis. Pavyzdžiui, įvairių matuoklių – temperatūros, resursų suvartojimo – fiksuojamus rodmenis, kasmėnesinius veiklos indikatorius, tokius kaip veiklos finansines ataskaitas ar veiklos procesų indikatorius. Taip pat rodiklių struktūra gali būti keičiama laikui bėgant: objektų atributų taksonomija (pvz., mirties priežasčių sąrašas, finansinių sąskaitų sąrašas) arba įrašo atributų sąrašai. Surenkamų rodiklių duomenų kiekis visada didėja, taip pat ir duomenų kiekis, kuriuos reikia apdoroti pagal rodiklius, auga, todėl standartiniai sprendimai, pavyzdžiui, reliacinės duomenų bazės, netinka dėl ilgos apdorojimo trukmės. Rodiklių duomenų bazės pasižymi tuo, kad duomenys į jas patenka iš daug skirtingų tiekėjų ir patekimo laikas tarp tiekėjų nėra sinchronizuojamas, o suagreguotą informaciją vartotojai gali užklausti bet kurio metu. Todėl šiame darbe bus nagrinėjamas srautinis duomenų apdorojimas, kuris patenkančius duomenis apdoros realiu laiku ir saugos jau apdorotus.

Realaus laiko duomenų apdorojimas (angl. Real-time data processing) yra jau senai nagrinėjamas kaip vienas iš būdų apdoroti didelių kiekių duomenis (angl. Big data). Vienas iš realaus laiko apdorojimo sprendimų yra srautinis duomenų apdorojimas. Srautinis duomenų apdorojimas (angl. stream processing) – lygiagrečių programų kūrimo modelis, pasireiškiantis sintaksiškai sujungiant nuoseklius skaičiavimo komponentus srautais, kad kiekvienas komponentas galėtų skaičiuoti savarankiškai [Bea15]. Darbe yra analizuojami jau egzistuojantys srautinio apdorojimo sprendimai pagal srautinio apdorojimo programų savybes aprašytas Michael Stonebraker, pasirenkamas vieną srautinio apdorojimo sprendimas ir su juo daromas eksperimentas.

Kadangi rodikliai laikui bėgant gali būti keičiami, reikia, kad sprendimas galėtų prisitaikyti prie pokyčių. Yra keli būdai kaip tokie sprendimai gali būti kuriami:

- Rankinio atnaujinimo sprendimas. Sukuriamas sprendimas pagal esamus reikalavimus, o atsiradus naujiems reikalavimams būtų kuriamos naujos arba keičiamos esamos apdorojimo sistemos.
- Universalus sprendimas. Sukuriamas universalus parametrizuojamas sprendimas ir pritaikomas užduotims nustatant parametrus.
- Kodo generavimo sprendimas. Sukuriamas sprendimas, kuris generuoja srautinio duomenų apdorojimo sistemas pagal iš anksto aprašytą struktūrą.

Jei nėra numatomas kitimas, pagal ką turi būti apdorojami duomenys, tai galima pasirinkti ir ranki-

nio apdorojimo sprendimą, kadangi nėra didelės tikimybės, kad teks keisti sprendimą. Toks sprendimas tiktų apdorojant išmaniųjų skaitiklių duomenis [Nev17]. Universalus sprendimas taip pat gali būti tinkamas, jei pirminiai rodiklių duomenis yra specifiški ir yra poreikis juos visus apdoroti. Toks sprendimas gali būti aktualus apdorojant duomenis iš sensorių, kurie matuoja namų būseną (temperatūrą, drėgmę ir t.t.) ir bet koks naujas sensorius taip pat turi būti prijungtas ir apdorotas [Yan17].

Pagal Jack Herrington 2003 metų knygą „Code Generation in Action“ kodo generavimas — tai kūrimas programinės įrangos, kuri kurs reikiamą programinę įrangą problemai spręsti. Tai daroma tokiais atvejais, kai sprendžiama problema reikalauja daug rankinio darbo, kurį įmanoma automatizuoti. Kuo didesnio sprendimo reikalauja uždavinys, tuo patraukliau tampa naudoti kodo generavimą sprendimo kūrimui. Kodo generavimas suteikia tokius privalumus:

- Architektūrinį nuoseklumą:
  - Verčia programuotojus labiau mąstyti apie architektūrą.
  - Jei sunku „priversti“ generatorių generuoti reikiamą kodą, problema gali būti architektūroje.
  - Geros dokumentacijos buvimas sumažina problemą, kai nariai palieka projektą.
- Abstrakciją:
  - Programuotojai galės kurti naujus šablonus, kurie leis esamą funkcionalumą pritaikyti kitomis kalbomis, sprendimams daug paprasčiau negu rankomis parašytą kodą.
  - Verslo analitikai galės apžvelgti ir patvirtinti sprendimo abstrakciją.
  - Abstrakcija padės paprasčiau paruošti dokumentaciją, testavimo atvejus, produkto palaikymo medžiagą ir t.t.
- Aukštą komandos moralę — rašomas kodas bus nuoseklus ir kokybiškas, todėl kels komandos pasitikėjimą savimi.
- Tinkamas sprendimas judriajam (angl. agile) programavimui, kadangi kodo generavimo kuriami sprendimai yra lankstesni, tai leidžia ateityje juos lengviau keisti ir atnaujinti.

Kodo generavimas tampa tikrai naudingas tada, kai jis naudojamas didelių kiekių rankiniam kodavimui pakeisti [Her03].

Darbe nagrinėjami architektūra ir eksperimentinis sprendimas, generuojantis srautinio apdorojimo sistemas pagal rodiklių duomenų modelius. Rodiklių duomenis apdorojantis eksperimentinis sprendimas turi pasižymėti tokiomis savybėmis:

- Srautinių apdorojimo sistemų generavimas pagal deklaratyvų aprašymą.
- Galimybė keisti esamas apdorojimo sistemas, kai pakeičiamas rodiklių duomenų modelis.

- Išvestinių rodiklių gavimas iš daugiau nei vieno rodiklio transformacijos.
- Išvestinių rodiklių apdorojimas pagal iš anksto apibrėžtas funkcijas.

Tikslas: Sukurti rodiklių duomenų srautinio apdorojimo sistemos, pritaikomos prie duomenų struktūrų naudojant kodo generavimą, architektūrą.

Uždaviniai:

1. Apibrėžti rodiklių duomenų modelį ir galimus rodiklių duomenų struktūros pokyčius.
2. Atliekant šaltinių analizę pasirinkti srautinio duomenų apdorojimo sprendimą.
3. Sudaryti srautinio apdorojimo sistemos architektūrą, pagal ją sukurti eksperimentinį sprendimą.
4. Nustatyti sukurto eksperimentinio sprendimo ir architektūros savybes atliekant bandymus su rodiklių duomenimis.

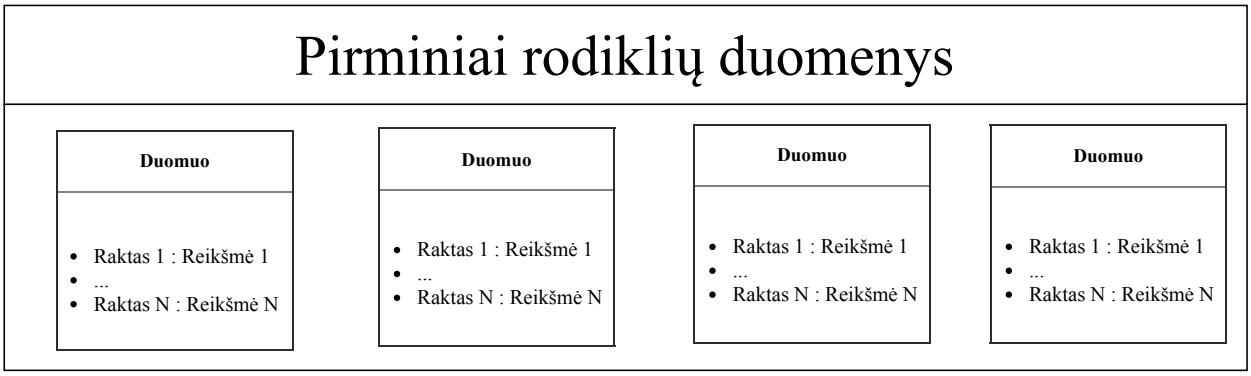
# 1. Rodiklių duomenys

Rodiklių duomenys — pasikartojančių įvykių parametrus aprašantys duomenys. Rodiklių duomenys pasižymi tuo, jog tik surinkti ir apdoroti tampa aktualūs. Rodikliai yra nepastovūs — jų struktūra gali būti keičiama gali atsirasti naujų savybių, kurias reikia surinkti, arba yra pašalinamos jau esamos savybės. Rodiklių duomenys yra renkami ir analizuojami labai skirtingiems poreikiams — visur kur reikalinga statistika ir apdorojami dideli kiekiai duomenų yra naudojami rodikliai: tai gali būti įmonės vedama „Microsoft Excel“ skaičiuoklė, kurioje užrašyta kiek buvo mokėta už elektrą kiekvienais metais per paskutinius 10 metų arba juodosios skylės nuotrauka, padaryta koreliuojant didelį duomenų kiekį iš 8 skirtingų teleskopų, ir gauti rodikliai rodo taškus ant paveikslėlio, kurio dydis yra $10^{12}$ kartų mažesnis nei pradiniai duomenys [AAA+19].

## 1.1. Pirminiai rodiklių duomenys

Rodiklių duomenys yra gaunami apdorojus pirminius rodiklių duomenis, kurie susidaro iš raktų ir reikšmių žodyno (1 pav.). Pirminių rodiklių duomenų raktai ir raktų kiekis gali nesutapti — žodynas gali turėti daugiau raktų nei reikia arba žodyno reikšmės gali būti nereikalingos einamojo rodiklio apskaičiavimui.

Kadangi pirminiai rodiklių duomenys gali būti gaunami iš skirtingų šaltinių, skirtingais laikais, todėl kuriamas sprendimas turi galėti priimti duomenis asinchroniškai ir iš skirtingų šaltinių, o sprendimas realiu laiku turi sugebėti apdoroti reikiamus duomenis, o nereikalingus atmesti.

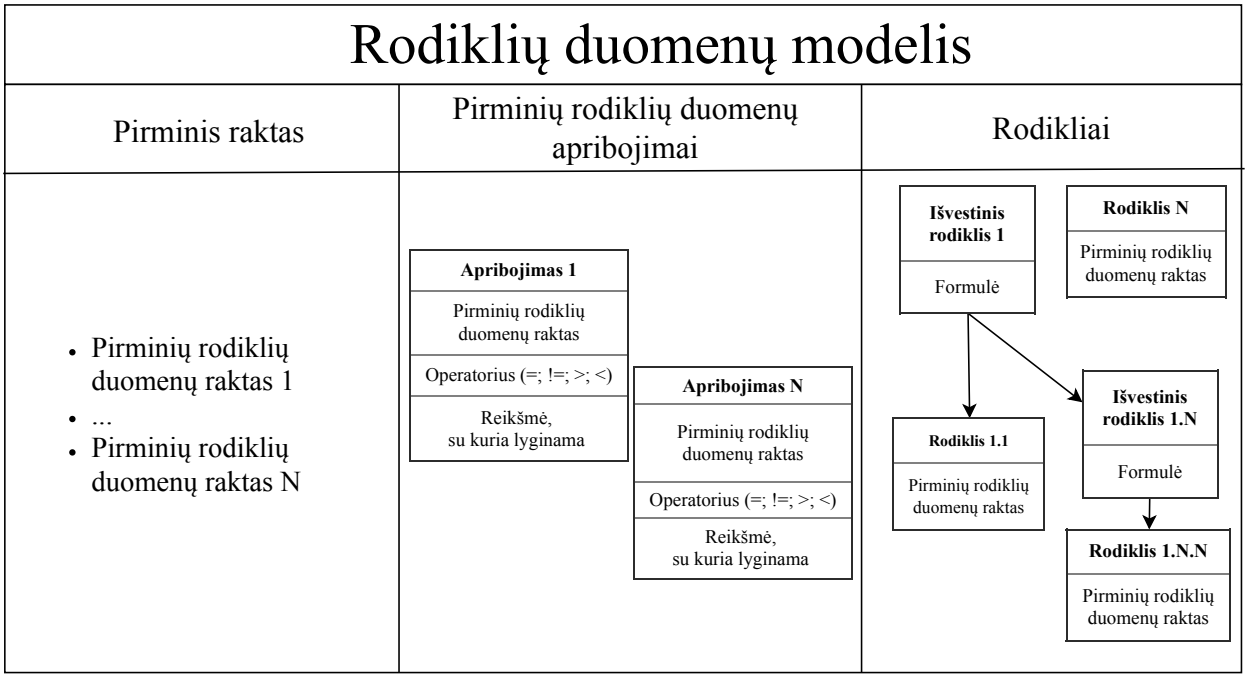



1 pav. Pirminiai rodiklių duomenys

## 1.2. Rodiklių duomenų modelis

Apibrėžiamas bendras rodiklių duomenų modelis (2 pav.), pagal kurį bus generuojamos srautinio duomenų apdorojimo sistemos. Siūlomas apibrėžimas susidaro iš trijų dalių:

- Rodiklių duomenų pirminis raktas, sudarytas iš pirminių rodiklių duomenų raktų rinkinio.
- Pirminių rodiklių duomenų reikšmių srities apribojimai.
- Rodikliai.




2 pav. Rodiklių duomenų modelis

### 1.2.1. Pirminis raktas

Rodiklio apdorojimo apibrėžimo dalis nusako pirminių rodiklių raktų rinkinį — reikšmių, pagal kurias grupuojami apdoroti duomenys. Pavyzdžiui:

- Statistikos srityje: norint surinkti duomenis, aprašančius **skirtingų savivaldybių** nekilnojamo turto kainas pagal **metus**, raktų rinkinys atrodys taip:

$$\{\text{„Savivaldybė“, „Metai“}\}$$

- Sensoriniai duomenys: norint surinkti vienos dienos duomenis, aprašančius **skirtingų sensorių** parodymus **kiekvieną valandą**, **kiekvienam ofiso kabinetui**, raktų rinkinys atrodys taip:

$$\{\text{„Sensoriaus tipas“, „Valanda“, Kabineto Numeris}\}$$

- Buhalterijoje: norint apdoroti duomenis, parodančius **skyrių** patirtas išlaidas **kiekvieną mėnesį**, raktų rinkinys atrodys taip:

$$\{\text{„Skyriaus pavadinimas“, „Mėnuo“}\}$$

Raktų kiekis turi būti neribojamas, kadangi realiame pasaulyje yra neribotas kiekis duomenų, pagal kuriuos galima grupuoti. Visų raktų reikšmių kiekių sandauga — apdorotų rodiklių duomenų pirminių raktų aibė.

### 1.2.2. Pirminių rodiklių duomenų apribojimai

Rodiklio duomenų modelio dalis yra sudaryta iš rinkinio apribojimų, kurie aprašo kokie, duomenys turi būti apdorojami. Apribojimai yra aprašomi predikatais. Pavyzdžiui:

- Statistikos srityje: norint surinkti žmonių, **vyresnių nei 60 metų**, atlyginimus pagal kalendorinius metus, apribojimų sąrašas atrodys taip:

$$\text{„Amžius“} > 59$$

- Sensoriniai duomenys: norint surinkti **tik savaitgalio** sensorių rodiklius, apribojimų sąrašas atrodys taip:

$$\text{„Savaitės Diena“} > 5, \text{„Savaitės Diena“} < 8$$

- Buhalterijoje: norint apdoroti tam tikrus duomenis **tik „Marketingas“ skyriaus** apribojimų sąrašas atrodys taip:

$$\text{„Skyriaus pavadinimas“} == \text{„Marketingas“}$$

Apribojimų skaičius gali būti bet koks (nuo 0 iki begalybės). Pirminiai rodiklių duomenys yra tinkami rodiklio apskaičiavimui, jei jų reikšmės priklauso apribojimų aibių sankirtai.

### 1.2.3. Rodikliai

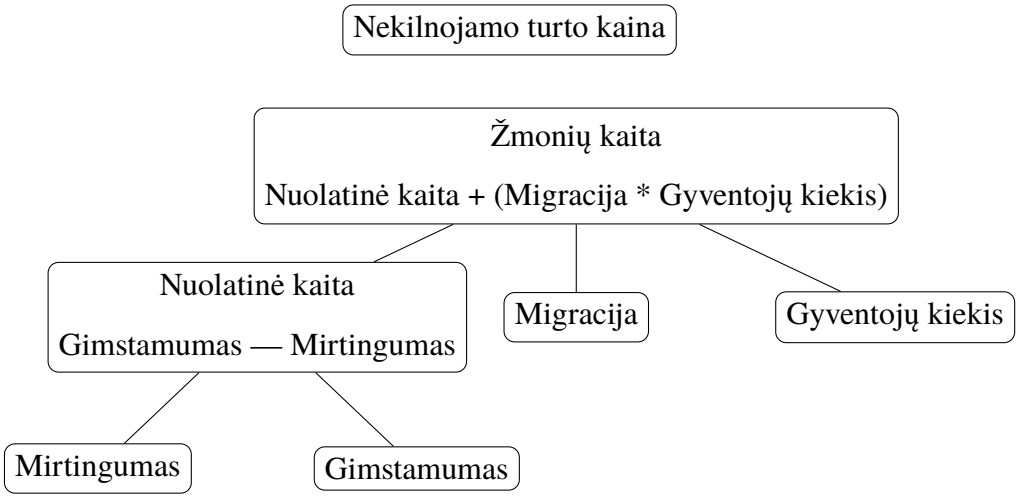
Rodiklių duomenų modelio dalis sudaryta iš dviejų skirtingų tipų komponentų:

- Rodiklis — parametrai gaunami iš pirminių rodiklių duomenų. Jis yra sudarytas iš pavadinimo ir pirminių rodiklių duomenų rakto.
- Išvestinis rodiklis — parametrai apskaičiuojami iš vieno ar daugiau rodiklių. Jis yra sudary-

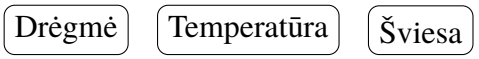
tas iš pavadinimo ir transformacijos funkcijos, naudojančios rodiklius, nuo kurių priklauso išvestinis rodiklis.

Rodiklių duomenų modelyje komponentų priklausomybės sudaro priklausomybių medį, kurio viršūnės — rodikliai, o briaunos — priklausomybės tarp rodiklių. Priklausomybių medyje kabančios (galinės) viršūnės visada bus rodiklio tipo, o visos kitos — išvestinio rodiklio tipo. Pavyzdžiui:

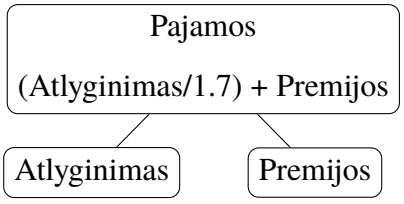
- Statistikos srityje: norint surinkti rodiklių duomenis aprašančius nekilnojamo turto kainas ir žmonių kaitą (*Žmonių kaita* $=$ (*Gimstamumas — Mirtingumas*) $+$ (*Migracija* $*$ *Gyventojų kiekis*), renkamų rodiklių medžiai apsirašytų taip:




- Sensoriniai duomenys: norint surinkti drėgmės, temperatūros ir šviesos sensorių rodiklius, rodiklių medžiai atrodys taip:




- Buhalterijoje: norint suskaičiuoti, kiek įmonės darbuotojai uždirba pajamų (*Pajamos* $=$ (*Atlyginimas*$/1.7$) $+$ *Premijos*), rodiklių medis bus toks:




## 1.3. Rezultatų struktūra

Apdorotų rodiklių rezultatai, gauti iš pirminių rodiklių duomenų, gali būti pavaizduoti lentelę. Lentelės duomenų pirminis raktas yra sudarytas iš pirminių rodiklių duomenų reikšmių kombina-

cijos, gautos pagal rodiklio modelio pirminį raktą. Rezultatų lentelės stulpeliuose aprašyti pirminis raktas ir išvestiniai rodikliai. Rezultatų lentelės kolonėlės susidaro iš apdorotų rodiklių duomenų.

| Rezultatai | | | |
|---|---|---|---|
| Pirminis raktas | Rodiklis 1 | ... | Rodiklis N |
| [Duomuo[Raktas1], ..., Duomuo[RaktasN]] | Apdoroti duomenys 1 | ... | Apdoroti duomenys N |
| ... | ... | ⋱ | ... |
| [Duomuo[Raktas1], ..., Duomuo[RaktasN]] | Apdoroti duomenys N | ... | Apdoroti duomenys N |

3 pav. Rezultatų struktūra

## 1.4. Rodiklių duomenų struktūros kitimas

Laikui bėgant rodiklių duomenų struktūra gali būti keičiama. Buhalterijos srityje tai gali įvykti dėl naujo įstatymo, kuris pakeičia tam tikrų rodiklių apskaičiavimą. Išmaniuosiuose namuose tai gali būti naujas išmaniųjų namų sensorius. Statistikos srityje tai gali būti poreikis išskaidyti tam tikrus rodiklius į smulkesnius rodiklius — rinkti ne migracijos rodiklį kaip vieną, o emigracijos ir imigracijos rodiklius atskirai.

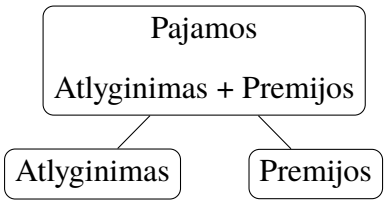
Šiame darbe rodiklių duomenų struktūrų pokyčiai apribojami taip:

- Pirminis raktas negali būti keičiami, nes naujos reikšmės tampa nepalyginamos su anksčiau surinktomis. Pavyzdžiui, buvo renkami metinių duomenų rodikliai, o pagal poreikius reikia rinkti mėnesinius duomenis, ir lyginimas su anksčiau surinktais tampa neįmanomas. Todėl bet kokie pirminio rakto pokyčiai turi būti laikomi nauju rodikliu.
- Apribojimai gali būti keičiami, tačiau šie pokyčiai turi būti naudojami tik tam, kad būtų išlaikytos pradinės apdorojamų pirminių rodiklių duomenų aibės. Pavyzdžiui, egzistuoja poreikis apdoroti duomenis iš visų skyrių, išskyrus administraciją, ir tokiam uždaviniui buvo sukurtas apribojimas — *„Skyriaus pavadinimas“ != „Administracija“*. Bet po tam tikro laiko dalis administracijos skyriaus tapo buhalterijos skyriumi. Siekiant, kad būtų išlaikyta apdorotų rodiklių lyginimo prasmė, nauji apribojimai atrodys taip: *„Skyriaus pavadinimas“!=„Administracija“*, *„Skyriaus pavadinimas“!=„Buhalterijos“*.
- Rodikliai ir išvestiniai rodikliai gali būti keičiami — nauji rodikliai pridedami, o seni pašalinami. Šie pokyčiai gali būti daromi bet kurioje rodiklių tarpusavio priklausomybės medžio vietoje. Rodiklių pokyčiai neturi daryti įtakos kitų rodiklių skaičiavimams.

Pavyzdžiui, pajamos skaičiuojamos naudojant formulę:

$$Pajamos = Atlyginimas + Premijos$$

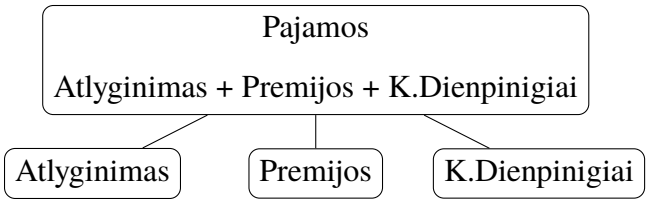
Pagal tai sugeneruotas toks medis:




Po tam tikro laiko atsirado poreikis pajamas skaičiuoti pagal formulę:

$$Pajamos = Atlyginimas + Premijos + Komandiruočių\ dienpinigiai$$

Naujas rodiklių medis atrodys taip:



# 2. Srautinio duomenų apdorojimo sprendimų analizė

## 2.1. Srautinis duomenų apdorojimas

Siekiant apžvelgti modernius srautinio duomenų apdorojimo sprendimus šiame skyriuje apsibrėžiamos jų savybės. Michael Stonebraker 2005 metais apibrėžė 8 taisykles realaus laiko (angl. real-time) srautinio duomenų apdorojimo architektūrai [SÇZ05]:

1 taisyklė: Duomenys turi judėti. Žemo uždelstumo užtikrinimui sistema turi apdoroti duomenis nenaudojant duomenų saugojimo operacijų. Taip pat sistema turi ne pati užklausti duomenų, o gauti juos iš kito šaltinio asinchroniškai.

2 taisyklė: Duomenų transformacijos turi būti vykdomas SQL pobūdžio užklausomis. Žemo abstrakcijos lygmens srautinio apdorojimo sistemos reikalauja ilgesnio programavimo laiko ir brangesnio palaikymo. Tuo tarpu aukšto abstrakcijos lygmens sistema, naudojanti SQL užklausas, kurias žino dauguma programuotojų ir yra naudojamos daugelyje skirtingų sistemų, leidžia efektyviau kurti srautinio apdorojimo sprendimus.

3 taisyklė: Architektūra turi susidoroti su duomenų netobulumais. Architektūra turi palaikyti galimybę nutraukti individualius skaičiavimus tam, kad neatsirastų blokuojančių operacijų, kurios sustabdo vieno modulio veikimą ir tuo pačiu visos architektūros veikimą. Taip pat ši architektūra turi sugebėti susidoroti su vėluojančiomis žinutėmis, pratęsiant laiko tarpą, per kurį ta žinutė turi ateiti.

4 taisyklė: Architektūra turi būti deterministinė. Kiekvieną kartą apdorojant tuos pačius duomenis rezultatai turi būti gaunami tokie patys.

5 taisyklė: Architektūra turi gebėti apdoroti išsaugotus duomenis ir realiu laiku gaunamus duomenis. Sistema, parašyta su tokia architektūra, turi galėti apdoroti jau esančius duomenis taip pat, kaip ir naujai ateinančius. Toks reikalavimas buvo aprašytas, nes atsirado poreikis nepastebimai perjungti apdorojimą iš istorinių duomenų į realiu laiku ateinančius duomenis automatiškai.

6 taisyklė: Architektūra turi užtikrinti duomenų saugumą ir apdorojimo prieinamumą. Kadangi sistema turi apdoroti didelius kiekius duomenų, architektūra klaidos atveju turi sugebėti persijungti į atsarginę sistemą ir tęsti darbą toliau. Taip pat tokios klaidos atveju atsarginė sistema turi būti apdorojusi visus duomenis ir sugebėti iš karto priimti naujus duomenis, o ne apdoroti duomenis iš pradžių.

7 taisyklė: Architektūra turi užtikrinti sugebėjimą paskirstyti sistemos darbus automatiškai.

Srautinio apdorojimo sistemos turi palaikyti kelių procesoriaus gijų operacijas. Taip pat sistema turi galėti veikti ant kelių kompiuterių vienu metu ir prireikus paskirstyti resursus pagal galimybes.

8 taisyklė: Architektūra turi apdoroti ir atsakyti akimirksniu. Anksčiau minėtos taisyklės nėra svarbios, jeigu sistema nesugeba greitai susidoroti su dideliu kiekiu naujų duomenų. Todėl turi būti naudojamas ne tik teisingas ir greitas srautinio apdorojimo sprendimas, bet ir gerai optimizuota sistema.

Todėl norint sužinoti tam tikro srautinio duomenų apdorojimo sprendimo tinkamumą uždaviniui reikia išanalizuoti jų savybes.

## 2.2. Srautinio duomenų apdorojimo sprendimai

Norime nustatyti srautinio apdorojimo sprendimų savybes pagal:

- Pristatymo semantiką (angl. delivery semantics) — apibrėžia, pagal kokį modelį bus pristatyti duomenys. Egzistuoja trys semantikos [Kah18]:
  - Bent vieną kartą (angl. at-least-once) — užtikrina, kad duomenys bus apdoroti bent kartą, bet gali atsirasti dublikatų.
  - Ne daugiau vieno karto (angl. at-most-once) — užtikrina, kad duomenys bus apdoroti daugiausia tik vieną kartą, bet gali atsirasti praradimų.
  - Tiksliai vieną kartą (angl. exactly-once) — užtikrina, kad duomenys bus apdoroti tik vieną kartą, ir klaidos bus suvaldytos.
- Uždelstumas (angl. latency) — apibrėžia laikų sumą — kiek laiko truko viena operacija ir kiek laiko ši operacija turėjo laukti eilėje kol bus pabaigtos kitos operacijos [KRK+18].
- Pralaidumas (angl. throughput) — apibrėžia, kiek pavyks įvykdyti operacijų per tam tikrą laiko tarpą.
- Abstrakcijos lygmuo (angl. abstraction) — apibrėžia kokio lygmens programavimo sąsają pateikia sprendimas.

Tuo tikslu išnagrinėsime ir palyginsime, kaip šitas savybes atitinka keturi egzistuojantys srautinio apdorojimo sprendimai — „Apache Storm“, „Apache Spark“, „Apache Flink“ ir „Heron“.

## 2.3. Pristatymo semantika

„Apache Spark“ ir „Apache Flink“ sprendimų pristatymo semantika yra tiksliai vieną kartą (angl. exactly-once), tai reiškia, kad visi duomenys bus apdoroti tik vieną kartą. Tačiau tam,

kad būtų užtikrinta ši semantika, sprendimas sunaudoja daug resursų, kadangi reikia užtikrinti, kad operacija bus vykdoma būtent vieną kartą kiekviename srautinio apdorojimo žingsnyje: duomenų gavime, kuris stipriai priklauso nuo duomenų šaltinio, duomenų transformacijoje, kurią turi įvykdyti srautinio apdorojimo sprendimas, ir duomenų saugojime; tai turi būti užtikrinta sprendimo ir naudojamos saugyklos [ZHA17].

„Apache Storm“ pristatymo semantika yra bent vieną kartą (angl. at-least-once), tai reiškia, kad į šį sprendimą siunčiami duomenys bus visada apdoroti, tačiau kartais gali būti apdoroti kelis kartus [DAM16]. Jei uždavinys nereikalauja tiksliai vieno karto apdorojimo, tai geriau rinktis bent vieną kartą ar ne daugiau vieno karto semantikas, kadangi jos neturi papildomų apsaugų, kurios reikalingos tiksliai vieno karto apdorojimui, ir todėl veikia greičiau [ZHA17].

„Heron“ sprendimų pristatymo semantika gali būti keičiama, programuotojas, kuriantis srautinio apdorojimo sistemą, šiam sprendimui aprašo, kokio tipo pristatymo semantikos kuriama sistema reikalauja [Ram17].

## 2.4. Uždelstumas

Uždelstumas — laiko trukmė, kuri parodo kaip greitai sprendimas įvykdo vieną operaciją, nuo jos patekimo į eilę iki šios operacijos apdorojimo pabaigos. Pagal [LLD16] aprašytus Martin Andreoni Lopez „Apache Storm“, „Apache Spark“ ir „Apache Flink“ bandymus galima matyti, kad būtent „Apache Storm“ turi mažiausią uždelstumą. Parinkus tinkamą paralelizmo parametrą šis sprendimas su užduotimi susidorojo net iki 15 kartų greičiau. Antroje vietoje liko „Apache Flink“, o po jos — „Apache Spark“.

„Heron“ sprendimas yra sukurtas siekiant pagerinti „Apache Storm“ sprendimo greitaveiką, todėl jo uždelstumas yra dar mažesnis nei „Apache Storm“ [KBF+15].

## 2.5. Pralaidumas

Pralaidumas apibrėžia kokį kiekį procesų sprendimas gali įvykdyti per tam tikrą laiko tarpą. Sanket Chintapalli 2016 metais išmatavo „Apache Storm“, „Apache Spark“ ir „Apache Flink“ sprendimų pralaidumą ir uždelstumą bei palygino rezultatus. Kaip ir anksčiau manyta, „Apache Spark“ turėjo aukščiausią pralaidumą iš visų, kadangi jis vienintelis duomenis apdoroja mikropaketais[CDE+16]. Antroje vietoje liko „Apache Flink“, kuris yra subalansuotas pralaidumo atžvilgiu, ir paskutinis liko „Apache Storm“, kuris turi žemą uždelstumą, todėl nukenčia pralaidumas. „Heron“ tuo tarpu turi aukštesnį pralaidumą ir žemesnį uždelstumą nei „Apache Storm“ [Ram15].

## 2.6. Abstrakcijos lygmuo

„Apache Storm“ parašytos sistemos yra žemo abstrakcijos lygmens, tai reiškia, kad turi būti aprašyti visi srautinio apdorojimo moduliai: setSpouts(..), kuriame nustatoma duomenų įeiga ir koks bus paralelizmo lygis, setBolt(..), kuriame nustatomi apdorojimo moduliai, kokius duomenis gaus iš prieš tai buvusio modulio ir paralelizmo lygis. Kiekvieno modulio execute() metodas aprašo, kaip šis modulis turi apdoroti duomenis [tut18]. Šio sprendimo programų kūrimo laikas užtruks ilgiau negu kitų sprendimų su aukštu abstrakcijos lygmeniu, tačiau žemas abstrakcijos leidžia rašyti daug greičiau veikiančias sistemas, kadangi programuotojas turi pilną kontrolę.

„Apache Spark“ parašytos programos yra aukšto abstrakcijos lygmens. Sistemos aprašomos funkciškai, todėl kodo rašymas trunka daug trumpiau ir jį yra daug lengviau skaityti. Tačiau prarandama galimybė optimizuoti ir paralelizmo klausimas paliekamas sprendimui. Kadangi „Apache Spark“ yra ne pilnai srautinis, o mikro-paketinis (angl. micro-batching) sprendimas, todėl vartotojas turi apsirašyti, kokio dydžio paketais bus renkami duomenys [SS15].

„Apache Flink“ parašytos programos yra aukšto abstrakcijos lygmens. „Apache Flink“ sprendimas pats užsiima resursų distribucija, todėl programuotojui lieka tik parašyti veikianti kodą, o sprendimas pats susitvarkys su paralelizmu [Doc18]. Tačiau tai reiškia, kad su šiuo sprendimu parašytos programos nepavyks optimizuoti taip pat gerai kaip žemo abstrakcijos lygmens sistemos.

„Heron“ sprendimas turi skirtingus API, kurie naudoja skirtingus abstrakcijos lygius, kuriuos galima rinktis pagal tinkamumą sprendžiamai problemai. Darbo rašymo metu „Heron“ turi 4 skirtingus API:

- „Heron Streamlet API“ — aukšto abstrakcijos lygmens API, rašomas su „Java“ programavimo kalba. Panaši sintaksė į „Apache Flink“ rašomų sprendimų.
- „Heron ECO API“ — eksperimentinis aukšto abstrakcijos lygmens API, rašomas su „Java“ programavimo kalba. Skiriasi nuo „Heron Streamlet API“, nes modulių apdorojimo eiliškumas apsirašo YAML formatu, kas leidžia keisti sukurtos srautinio duomenų apdorojimo programos struktūrą nekeičiant kodo.
- „Heron Topology API for Java“ — žemo abstrakcijos lygmens API, rašomas su „Java“ programavimo kalba. Rašomas identiškas kodas, kaip ir „Apache Storm“, kadangi „Heron“ sprendimas buvo sukurtas siekiant pagerinti „Apache Storm“.
- „Heron Topology API for Python“ — žemo abstrakcijos lygmens API, rašomas su „Python“ programavimo kalba. Su šiuo API kuriamos srautinio apdorojimo sistemos yra panašios į „Apache Storm“ sistemas, tik su „Python“ programavimo kalbos privalumais.

## 2.7. Apibendrinimas

Pagal atlikta analizę sudaryta 1 lentelė. Iš šių keturių sprendimų pasirinktas vienas, kuris labiausiai tinka rodiklių duomenų srautinio apdorojimo sistemų generavimui pagal aprašytas savybes. Tinkamas sprendimas turi pasižymėti:

- Apdorojimo greičiu, svarbu, kad duomenys būtų apdorojami kai tik patenka į sistemą, todėl turi būti žemas uždelstumas.
- Tiksliai vieną kartą apdorojimu, kadangi norint gauti tikslius rodiklių duomenis turi neišsikreipti pirminių rodiklių duomenų aibė apdorojimo metu.
- Žemu abstrakcijos lygiu, nes tai leidžia generuoti iš anksto optimizuotas srautinio apdorojimo sistemas.

1 lentelė. Srautinių duomenų apdorojimo sprendimų palyginimas

| Charakteristika | „Apache Storm“ | „Apache Spark“ | „Apache Flink“ | „Heron“ |
|---|---|---|---|---|
| Pristatymas | Bent vieną kartą | Tiksliai vieną kartą | Tiksliai vieną kartą | Pasirenkamas |
| Uždelstumas | Žemas | Aukštas | Vidutinis | Žemas |
| Pralaidumas | Žemas | Aukštas | Vidutinis | Vidutinis |
| Abstrakcijos lygis | Žemas | Aukštas | Aukštas | Pasirenkamas |

Pagal analizę (1 lentelė.) ir apsibrėžtus reikalavimus darbe sprendžiamam uždaviniui tinkamiausias srautinio apdorojimo sprendimas yra „Heron“.

# 3. Srautinio duomenų apdorojimo sistemų architektūra

Šiame skyriuje apžvelgiamas srautinio apdorojimo sistemų rašymas „Heron“ sprendimui ir srautinio apdorojimo sistemų generavimas pagal deklaratyviai aprašytą rodiklių duomenų modelį.

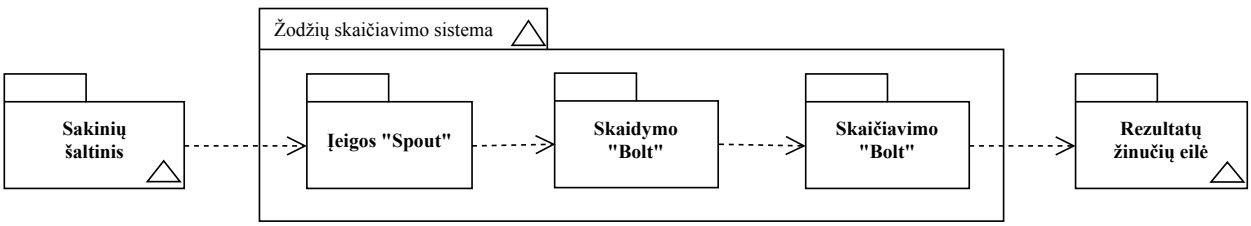
## 3.1. Srautinio apdorojimo sistemos komponentai

„Heron“ sprendimui parašytos srautinio apdorojimo sistemos susideda iš pateiktų komponentų tipų:

- „Spout“ — duomenų įeigos komponentai, kurių gali būti daugiau nei vienas, jeigu yra daugiau duomenų šaltinių. Pavyzdžiui, žinučių eilės „Spout“ komponentas ir duomenų bazės „Spout“ komponentas.
- „Bolt“ — duomenų apdorojimo komponentai, kurių gali būti daugiau nei vienas. Šie komponentai gali gauti duomenis iš „Spout“ komponentų ir iš kitų tokio pačio tipo komponentų.
- „Topology“ — srautinio apdorojimo sistemos konfigūracijos komponentas, kuris visada yra vienas. „Topology“ komponentas skirtas nustatyti srautinio apdorojimo sistemos pavadinimą ir sistemos konfigūracijas (pristatymo semantika, kontrolinių saugojimo taškų intervalas ir t.t.) bei specifikuoti, kaip tarpusavyje yra sujungti „Spout“ ir „Bolt“ komponentai.

Konkretus pavyzdys srautinio apdorojimo sistemos atrodo taip: reikalinga srautinio apdorojimo sistema, kuriai bus siunčiami sakiniai, o ji grąžins žodyną su žodžių dažniu. Vienas iš būdų įgyvendinti tokią sistemą (4 pav.) yra:

- Įeigos „Spout“ komponentas, gaunantis sakinius iš žinučių eilės ir perduodantis (angl. emit) sakinius toliau.
- Skaidymo „Bolt“ komponentas, gaunamus sakinius suskaidantis į žodžius ir perduodantis juos toliau.
- Skaičiavimo „Bolt“ komponentas, saugantis pasikartojančių žodžių žodyną, kur raktas yra žodis, o reikšmė — kiek kartų pasikartojo. Gavęs žodį, jis padidina jo skaitiklį žodyne ir perduoda žodyną į „Rezultatų“ žinučių eilę.



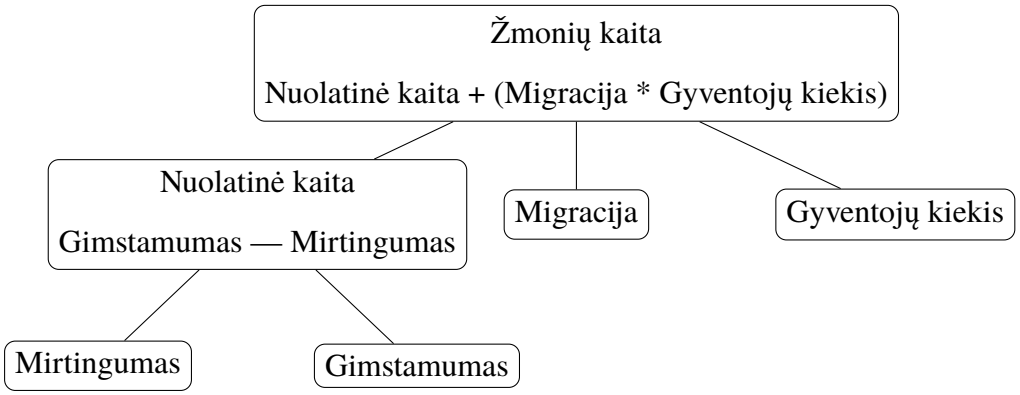

4 pav. Srautinio apdorojimo sistemos pavyzdys

## 3.2. Srautinio apdorojimo sistemų sudarymas pagal rodiklių duomenų modelį

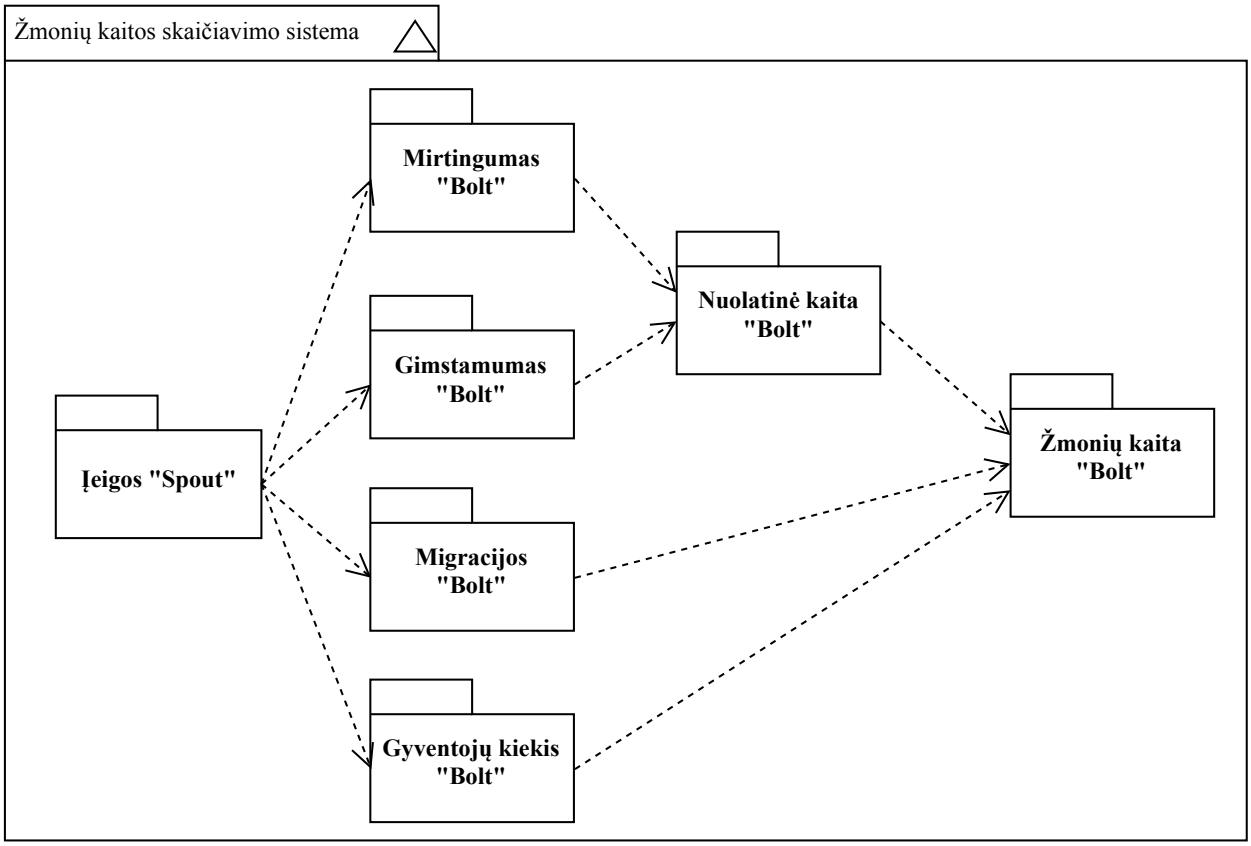
Srautinio apdorojimo sistemos, sudarytos pagal deklaratyvų rodiklių duomenų modelį atrodo taip:

1. „Spout“ tipo įeigos komponentas. Šis komponentas aprašo priminių rodiklių duomenų filtravimo predikatus ir pirminį raktą.
2. „Bolt“ tipo komponentai, kurie sudaryti pagal rodiklių duomenų modelio rodiklius. Kiekvienam rodikliui turi būti vienas „Bolt“ tipo komponentas. „Bolt“ komponentai gali būti dviejų tipų:
   - Paprastų rodiklių „Bolt“ komponentai, kurie atsakingi už pirminių rodiklių duomenų apdorojimą.
   - Išvestiniai rodiklių „Bolt“ komponentai, kurie atsakingi už duomenų, gautų iš paprastų arba išvestinių rodiklių „Bolt“ komponentų, transformaciją. Transformacijos — bet kokie iš anksto aprašyti nuoseklūs veiksmai su duomenų rinkiniu, gražinantys vieną rezultatą. Pavyzdžiui: norint sudaryti išvestinį rodiklį, kuris skaičiuoja dviejų pirminių rodiklių sumą, turi būti:
     - Iš anksto aprašyta funkcija „suma“, kuri priima du skaičius ir grąžina vieną.
     - Rodiklio modelyje užrašyta transformacija — suma(rodiklis1, rodiklis2).
3. „Topology“ aprašas, kuriame specifikuojamos „Spout“ ir „Bolt“ komponentų sąjungos pagal priklausomybių medį gautą iš rodiklių duomenų modelio.

Pavyzdžiui, jei rodiklis atrodo taip:



Tokį rodiklį apdorojančios sistemos architektūra „Heron“ sprendime turėtų būti tokia, kaip pavaizduota 5-ame pav:




5 pav. Srautinio apdorojimo sistemos pavyzdys

# 4. Eksperimentas

## 4.1. Eksperimento tikslas

Eksperimento tikslas — sukurti eksperimentinį sprendimą, kurio pagalba patikrinti, ar sudaryta architektūra atitinka šias savybes:

- Srautinių apdorojimo sistemų generavimas pagal deklaratyvų aprašymą.
- Galimybė keisti esamas apdorojimo sistemas, kai pakeičiamas rodiklių duomenų modelis.
- Išvestinių rodiklių gavimas iš daugiau nei vieno rodiklio transformacijos.
- Išvestinių rodiklių apdorojimas pagal iš anksto apibrėžtas funkcijas.

Siekiant patikrinti sudarytos architektūros savybes, buvo atlikti šie bandymai:

1. Sukurtas pavyzdinis rodiklių duomenų modelis, aprašytas deklaratyviai ir pateiktas į eksperimentinį sprendimą.
2. Atnaujintas rodiklių duomenų modelis (pridėtas naujas išvestinis rodiklis) ir pateiktas į eksperimentinį sprendimą.

## 4.2. Eksperimento vykdymo aplinka

Kompiuterio, kuriame buvo vykdomas eksperimentas, specifikacijos:

- Procesorius — Intel Core i7—7700k 4,50GHz.
- Sisteminis diskas — 512 GB SSD.
- Operatyvioji atmintis — 16 GB.
- Operacinė sistema — Ubuntu 18.04 versija.

## 4.3. Eksperimentinis sprendimas

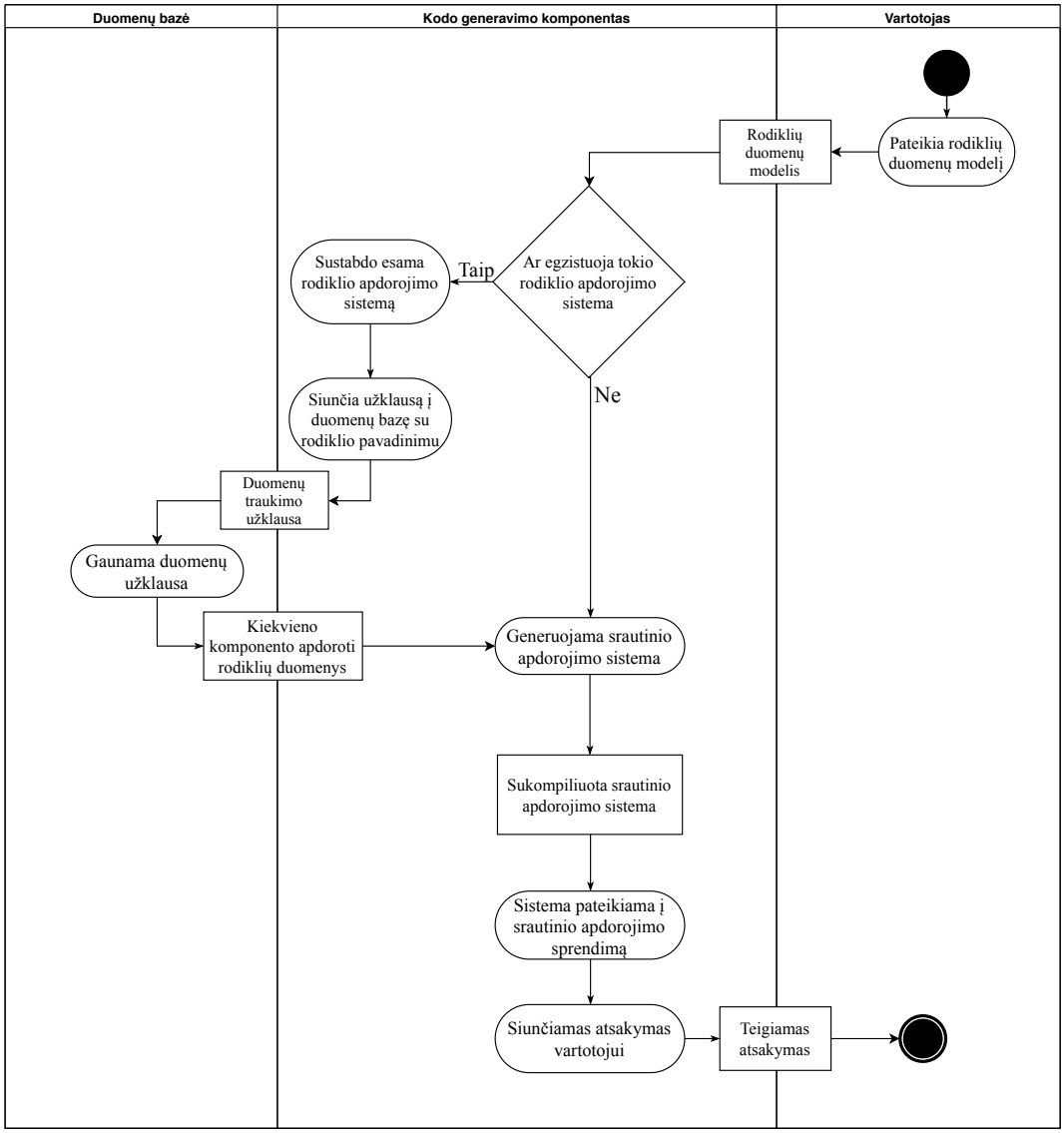
Siekiant nustatyti sudarytos architektūros savybes sukurtas eksperimentinis sprendimas. Jis sudarytas iš šių pagrindinių komponentų:

- Sistemų generavimo komponentas — programa, generuojanti srautinio apdorojimo sistemas.
- Duomenų bazės komponentas — duomenų bazė, sauganti rodiklių duomenų apdorojimo rezultatus.
- Srautinio apdorojimo sprendimas — programinė įranga, kurioje veikia sugeneruotos srautinio apdorojimo sistemos.

### 4.3.1. Sistemų generavimo komponentas

Sistemų generavimo komponentas — programa sukurta su „.NET core“ biblioteka, nes reikalingas taškas (angl. endpoint) deklaratyviai aprašyto rodiklio modelio pateikimui. Srautinio apdorojimo sistemų kodo generavimas yra vykdomas pagal šablonus, kurie sukurti kiekvienam srautinio apdorojimo sprendimo komponento tipui. Šablono laukai, kuriuos reikia pakeisti, užrašyti specialiais identifikuojančiais elementais. Generavimo metu šie elementai keičiami reikiamu kodu užduočiai spręsti. Šiame komponente kodo generavimas veikia taip (6 pav.):

1. Gaunamas rodiklių duomenų modelis.
2. Tikrinama, ar jau egzistuoja tokia srautinio apdorojimo sistema. Tai daroma kreipiantis į duomenų bazę ir tikrinant, ar egzistuoja įrašų, kurių raktai sutampa su generuojamos sistemos pavadinimu.
3. Jei egzistuoja:
   (a) Sustabdoma veikianti sistema. Tai vykdoma kviečiant stabdymo komandą per Ubuntu operacinės sistemos terminalą.
   (b) Iš duomenų bazės ištraukiami jau apdoroti rodiklių duomenys ir perduodami į generavimo dalį.
4. Generuojamas srautinio apdorojimo sistemos įeigos „Spout“ tipo komponentas. Jame įrašomas žinučių eilės grupės pavadinimas, sąlygos pirminių rodiklių duomenims pagal apribojimus ir pirminių rodiklių duomeniui sugeneruojamas unikalus identifikatorius, kurio pagalba išvestinių rodiklių apdorojimo komponentai vykdo transformacijas teisinga tvarka.
5. Pagal rodiklius generuojami duomenų apdorojimo „Bolt“ tipo komponentai. Jeigu generuojamas komponentas jau egzistuoja, sugeneruotas rodiklio apdorojimo „Bolt“ tipo komponento kodas užpildomas rezultatais gautais iš duomenų bazės.
6. Generuojamas sistemos „Topology“ tipo aprašas, kuriame yra specifikuojamas komponentų tarpusavio jungimasis.
7. Visas sugeneruotas kodas talpinamas į vieną aplanką ir sukompiliuojamas.
8. Sukompiliuota sistema pateikiama į srautinio apdorojimo programą.




6 pav. Srautinio apdorojimo sistemų generavimo komponento veikimas

### 4.3.2. Apdorotų rodiklių duomenų bazė

Duomenų bazė eksperimentiniame sprendime saugo apdorotus rodiklių duomenis. Norint saugoti apdorotus išvestinių ir pagrindinių rodiklių duomenis, visi srautinio apdorojimo sistemų komponentai yra prijungti prie duomenų bazės ir apdoroję duomenis talpina juos į duomenų bazę perrašydami ten jau esamus. Duomenys iš duomenų bazės naudojami dviems poreikiams:

1. Rezultatų gavimui — siunčiant užklausą su srautinio apdorojimo sistemos pavadinimu į eksperimentiniame sprendime sukurtą tašką.
2. Nepakitusių rodiklių „Bolt“ komponentų generavimui. Kadangi po rodiklių pridėjimo arba pašalinimo apdorojimas turi vykti toliau, kodo generavimo komponentas naudoja esamus rezultatus nepakitusiu komponentų generavimui.

Saugoti apdorotus duomenis buvo pasirinkta „Redis“ duomenų bazė. Ji pasirinkta, kadangi

apdoroti rodiklių duomenys užima nedaug vietos, o labai svarbu greitai pasiekti ir įdėti duomenis. „Redis“ duomenų bazių valdymo sistema tinka, nes duomenys saugomi operatyvioje atmintyje, o tai leidžia duomenis ištraukti ir įdėti greičiau nei tradicinėse duomenų bazėse [Car13].

Apdoroti rodikliai saugomi unikalioje eilėje pagal srautinės apdorojimo sistemos ir rodiklio pavadinimų kombinaciją. Pavyzdžiui: renkant darbuotojų atlyginimus per metus pagal uždarbio ir premijos kombinaciją, apdoroti rodiklių duomenys būtų saugomi pagal raktus: „darbuotojų—atlyginimas:uždarbis“, „darbuotojų—atlyginimas:premijos“ ir „darbuotojų—atlyginimas:atlyginimas“.

### 4.3.3. Papildoma programinė įranga

Eksperimentinio sprendimo veikimui reikia papildomos programinės įrangos:

- Žinučių eilės pagrindinis uždavinys — perduoti duomenis į srautinio apdorojimo sistemą. Žinučių eilės naudoja publikavimo—prenumeratos modelį (angl. publish—subscribe pattern), komponentai, norintys gauti duomenis, užsiregistruoja žinučių eilėje, o siunčiantys komponentai tiesiog perduoda juos į žinučių eilę. Taip yra įgyvendinamas asinchroninis bendravimas, komponentai perduoda duomenis į žinučių eilę, o komponentas, norintis gauti duomenis, gali juos pasiimti bet kuriuo momentu iš eilės. Eksperimentinio sprendimo įgyvendinimui pasirinkta Apache Kafka žinučių eilė.
- Sugeneruotos sistemos kompiliavimui naudojama „Pants“ kompiliavimo sistema.

Eksperimentų atlikimui yra reikalinga tokia programinė įranga:

- Pirminių rodiklių duomenų siuntimo komponentas. Šis Python kalba parašytas komponentas į žinučių eilę reguliaru intervalu siunčia sugeneruotus duomenis.
- Postman programinė įranga, kuri naudojama išsiųsti rodiklių duomenų modelį į eksperimentinio sprendimo generavimo tašką.
- Naršyklė, kuri naudojama stebėti sugeneruotas ir pateiktas srautinio apdorojimo sistemas per „Heron“ sąsają ir gauti duomenis iš apdorotų rodiklių duomenų bazės.

## 4.4. Eksperimentas

Eksperimento vykdymui sudarytas rodiklių duomenų uždavinys, kuriame apdorojami išmanaus ofiso sensorių — drėgmės, temperatūros, šviesos — rodikliai. Renkami kiekvienam dienos metui (rytas, diena, vakaras) ir kiekvienam kabinetui išskyrus kabinetą, kurio numeris 100. Taip

pat reikia apskaičiuoti išvestinį juntamosios temperatūros rodiklį pagal formulę [ABP13]:

$$\textit{Juntamoji temperatūra} = \textit{Temperatūra}–(0.55 * (1– \frac{\textit{Drėgmė}}{1000}) * (\textit{Temperatūra}–14.5))$$

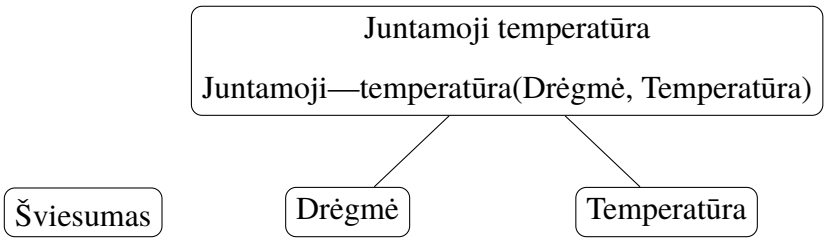
Po tam tikro laiko pridėtas naujas bendro pojūčio išvestinį rodiklį, kuris apskaičiuojamas taip:

$$\textit{Bendras pojūtis} = \frac{\textit{Šviesumas}}{75} * \frac{\textit{Juntamoji temperatūra}}{25}$$

### 4.4.1. Pradiniai duomenys

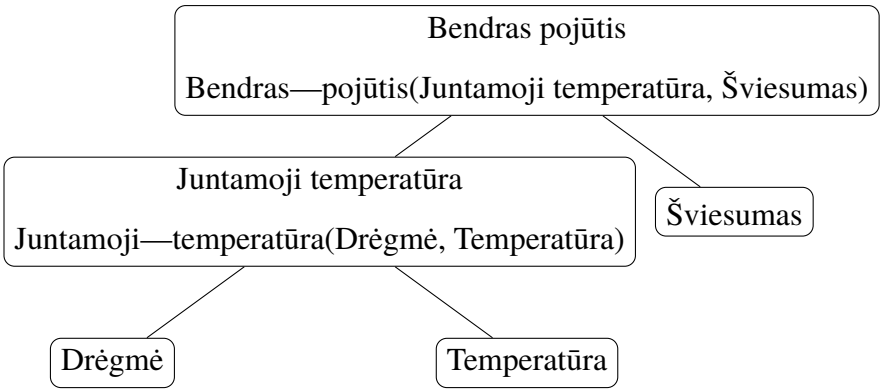
Pradinis rodiklių duomenų modelis pagal uždavinį atrodo taip:

- Pirminis raktas: *„DienosMetas“, „Kabinetas“*
- Pirminių rodiklių duomenų apribojimai: *„Kabinetas“ != „100“*
- Rodikliai:




Atnaujintas rodiklių duomenų modelis atrodo taip:

- Pirminis raktas: *„DienosMetas“, „Kabinetas“*
- Pirminių rodiklių duomenų apribojimai: *„Kabinetas“ != „100“*
- Rodikliai:



Taip pat sugeneruoti pirminiai rodiklių duomenys naudojant `https://mockaroo.com/` įrankį (Priedas nr. 1). Šių duomenų raktai atrodo taip:

- Kabineto numeris — reikšmės iš aibės — 100, 101, 102, 103.
- Dienos Metas — reikšmės iš aibės — RYTAS, DIENA, VAKARAS.
- Drėgmė — reikšmes nuo 0 iki 100.
- Temperatūra — reikšmės nuo 10 iki 30.
- Šviesumas — reikšmės nuo 0 iki 100.

### 4.4.2. Eksperimento eiga

Eksperimentas buvo vykdomas tokia eiga:

1. Pradinių rodiklių duomenų modelis užrašomas JSON formatu (Priedas nr. 2) ir naudojant Postman programą siunčiantis į sprendimą.
2. Atsidarius „Heron“ sąsają naršyklėje matoma sugeneruotos srautinio apdorojimo sistemos diagrama (Priedas nr. 3).
3. Su pirminių rodiklių duomenų siuntimo programa duomenys siunčiami į žinučių eilę.
4. Naršyklėje išsiunčiama užklausa į sprendimo duomenų prieigos tašką ir gaunami apdoroti rodiklių duomenys (Priedas nr. 4).
5. Atnaujintas rodiklių duomenų modelis užrašomas JSON formatu (Priedas nr. 5) ir naudojant Postman programą paduodamas į sprendimą.
6. Atnaujinus „Heron“ sąsają naršyklėje matoma, jog atsinaujino srautinio apdorojimo sistemos diagrama (Priedas nr. 6).
7. Naršyklėje siunčiama užklausa į sprendimo tašką ir gaunami apdoroti rodiklių duomenys, kuriuose jau galima matyti naujo rodiklio rezultatus (Priedas nr. 7).

### 4.4.3. Eksperimento rezultatai

Stebint eksperimentą gauti rezultatai:

- Sėkmingai sugeneruota srautinio apdorojimo sistema iš rodiklio duomenų modelio.
- Srautinio apdorojimo sistema sėkmingai apdorojo pirminius rodiklių duomenis pagal rodiklių duomenų modelį.
- Išvestinių rodiklių „Bolt“ komponentai teisingai apskaičiavo rezultatus pagal iš anksto aprašytas funkcijas naudojant duomenis iš daugiau nei vieno „Bolt“ komponento.
- Sėkmingai atnaujinta srautinio apdorojimo sistema pakeitus rodiklių duomenų modelį.

- Atnaujinus srautinio apdorojimo sistemą nepakeisti „Bolt“ tipo komponentai toliau pratęsė rodiklių apdorojimą.

Taip pat atliekant eksperimentą buvo nustatyta, jog architektūra palaiko paralelinį skaičiavimą. Sustabdžius duomenų siuntimą į srautinio apdorojimo sprendimą „Bolt“ komponentai skaičiavimus baigia skirtingais laikais. Tai parodo, jog sprendimas, kuris naudoja sudarytą srautinio apdorojimo sistemos architektūrą, nepriklauso nuo rodiklių transformacijos sudėtingumo.

# Rezultatai

1. Apibrėžtas rodiklių duomenų modelis ir galimi rodiklių duomenų struktūros pokyčiai.
2. Atliktas srautinio apdorojimo sprendimų — „Apache Storm“, „Apache Spark“, „Apache Flink“ ir „Heron“ — palyginimas, ir pateikta analizės suvestinė.
3. Sudaryta srautinio apdorojimo sistemos architektūra, kuri yra tinkama spręsti rodiklių apdorojimo uždavinį su „Heron“ srautinio apdorojimo sprendimu.
4. Sukurtas eksperimentinis sprendimas (`https://git.mif.vu.lt/vyzi2849/stream-processing-generator`) ir jo pagalba nustatyta, jog srautinio apdorojimo sistemų architektūra atitinka apsibrėžtas savybes:
    - Srautinių apdorojimo sistemų generavimas pagal deklaratyvų aprašymą.
    - Galimybė keisti esamas apdorojimo sistemas, kai pakeičiamas rodiklių duomenų modelis.
    - Išvestinių rodiklių gavimas iš daugiau nei vieno rodiklio transformacijos.
    - Išvestinių rodiklių apdorojimas pagal iš anksto apibrėžtas funkcijas.

# Išvados

1. Pagal padarytą šaltinių analizę rodiklių duomenis apdorojančioms srautinio apdorojimo sistemoms generuoti tinka „Heron“ srautinio apdorojimo sprendimas, nes rodiklių duomenų apdorojimui svarbus: žemas uždelstumas, kad būtų užtikrintas apdorojimo greitis; tiksliai vieną kartą pristatymo semantika, kad nebūtų prarandami duomenys apdorojimo metu; žemas abstrakcijos lygis, kad būtų galima generuoti optimizuotas srautinio apdorojimo sistemas.
2. Eksperimento būdų įrodyta, jog siūlomas srautinio apdorojimo sistemų generavimas pagal deklaratyvų rodiklių duomenų modelį gali būti įgyvendinamas. Sugeneruotos srautinio apdorojimo sistemos atitinka apsibrėžtas savybes, kurių reikia rodiklių duomenų apdorojimui. Įrodyta, jog srautinių apdorojimo sistemų generavimas „Heron“ srautinio apdorojimo sprendimui yra tinkamas rodiklių duomenų apdorojimo uždaviniui spręsti.
3. Eksperimento metu nustatyta, jog sudaryta srautinio apdorojimo sistemos architektūra palaiko paralelinius skaičiavimus ir todėl sprendimas nepriklauso nuo „Bolt“ skaičiavimo sudėtingumo.

# Literatūra


[AAA+19] Kazunori Akiyama, Antxon Alberdi, Walter Alef, Keiichi Asada ir k.t. First m87 event horizon telescope results. iii. data processing and calibration. *The Astrophysical Journal Letters*, 875(1):L3, 2019.

[ABP13] G Brooke Anderson, Michelle L Bell ir Roger D Peng. Methods to calculate the heat index as an exposure metric in environmental health research. *Environmental health perspectives*, 121(10):1111–1119, 2013.

[Bea15] Jonathan Beard. A short intro to stream processing. `http://www.jonathanbeard.io/blog/2015/09/19/streaming-and-dataflow.html`, 2015-09.

[Car13] Josiah L Carlson. *Redis in action*. Manning Publications Co., 2013.

[CDE+16] Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar ir k.t. Benchmarking streaming computation engines: storm, flink and spark streaming. *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*, p. 1789–1792. IEEE, 2016.

[DAM16] PRITHIVIRAJ DAMODARAN. "exactly-once" with a kafka-storm integration. `http://bytecontinnum.com/2016/06/exactly-kafka-storm-integration/`, 2016.

[Doc18] Flink Documentation. Flink datastream api programming guide. `https://ci.apache.org/projects/flink/flink-docs-release-1.5/dev/datastream_api.html`, 2018.

[Her03] Jack Herrington. *Code generation in action*. Manning Publications Co., 2003.

[Yan17] Shusen Yang. Iot stream processing and analytics in the fog. *IEEE Communications Magazine*, 55(8):21–27, 2017.

[Kah18] Ensar Basri Kahveci. Processing guarantees in hazelcast jet. `https://blog.hazelcast.com/processing-guarantees-hazelcast-jet/`, 2018.

[KBF+15] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy ir Siddarth Taneja. Twitter heron: stream processing at scale. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, p. 239–250, Melbourne,

Victoria, Australia. ACM, 2015. ISBN: 978-1-4503-2758-9. DOI: 10.1145/2723372.2742788. URL: http://doi.acm.org/10.1145/2723372.2742788.

[KRK+18] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen ir Volker Markl. Benchmarking distributed stream processing engines. *arXiv preprint arXiv:1802.08496*, 2018.

[LLD16] Martin Andreoni Lopez, Antonio Gonzalez Pastana Lobato ir Otto Carlos Muniz Bandeira Duarte. A performance comparison of open-source stream processing platforms. *2016 IEEE Global Communications Conference (GLOBECOM)*:1–6, 2016.

[Nev17] Mantas Neviera. Išmaniųjų apskaitų didelių duomenų kiekių apdorojimas modernioje duomenų apdorojimo architektūroje, 2017.

[Ram15] Karthik Ramasamy. Flying faster with twitter heron. https://blog.twitter.com/engineering/en_us/a/2015/flying-faster-with-twitter-heron.html, 2015-06.

[Ram17] Karthik Ramasamy. Why heron? part 2. https://streaml.io/blog/why-heron-part-2, 2017-09.

[SÇZ05] Michael Stonebraker, Uğur Çetintemel ir Stan Zdonik. The 8 requirements of real-time stream processing. *ACM Sigmod Record*, 34(4):42–47, 2005.

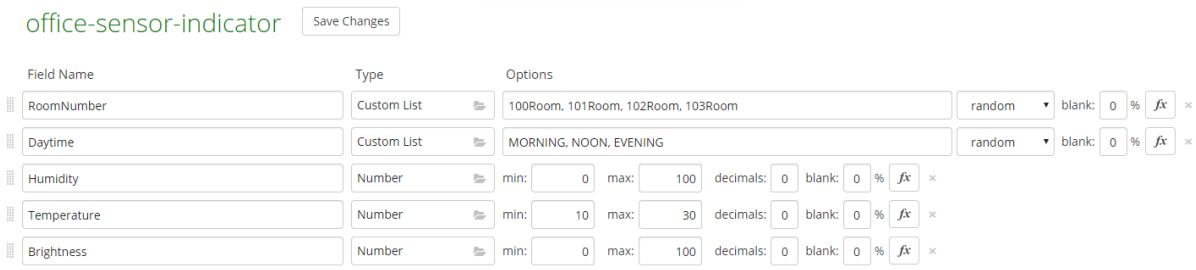
[SS15] Abdul Ghaffar Shoro ir Tariq Rahim Soomro. Big data analysis: apache spark perspective. *Global Journal of Computer Science and Technology*, 2015.

[tut18] tutorialspoint.com. Apache storm - core concepts. https://www.tutorialspoint.com/apache_storm/apache_storm_core_concepts.htm, 2018.

[ZHA17] Ji ZHANG. How to achieve exactly-once semantics in spark streaming. http://shzhangji.com/blog/2017/07/31/how-to-achieve-exactly-once-semantics-in-spark-streaming/, 2017.

# Priedas nr. 1

# Pirminių rodiklių duomenų generavimo įrankis



## Priedas nr. 2

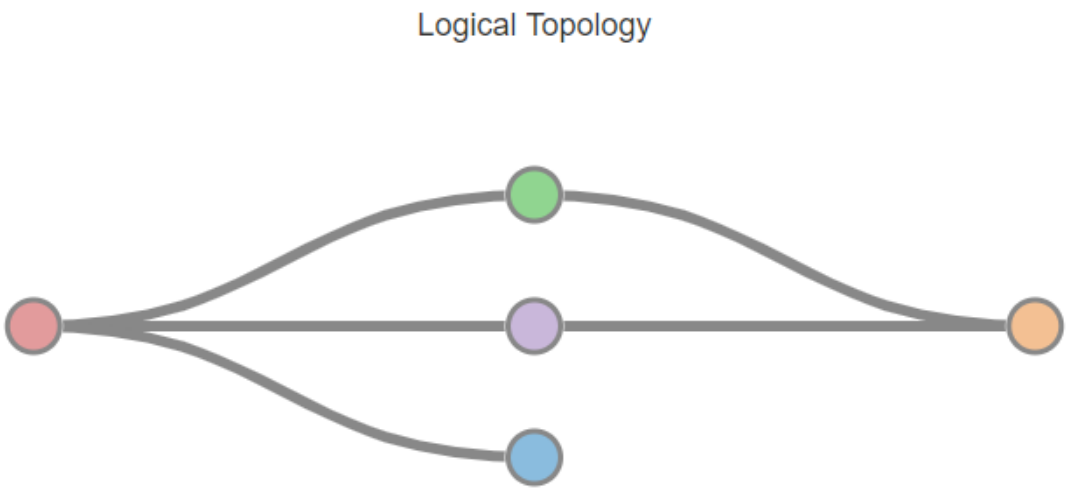
## Rodiklių duomenų modelis JSON formatu


```
{
    "IndicatorId": "32c24420-afbd-44fb-b045-ef72c48cb04e",
    "Name": "office-sensors-indicator",
    "VersionId": "1-416f51b5-2b8a-4cb0-9978-f713d5990c52",
    "PrimaryKey": [
        "RoomNumber",
        "Daytime"
    ],
    "Filters": [
        {
            "FieldName": "RoomNumber",
            "Value": "100Room",
            "Operator": "NEQU"
        }
    ],
    "Values": [
        {
            "Id": "5dbfcf52-bbce-4ec6-ad31-1cf0812b3106",
            "FieldName": "HeatIndex",
            "Formula": "%65c89c44-f5c5-48d4-ba31-01e3d4b69e42% - (0.55 * (1 -
                (%117af450-3086-412f-90f5-ad63e1989d04%/1000)) * (%65c89c44-
                f5c5-48d4-ba31-01e3d4b69e42% - 14.5))",
            "NextValues": [
                {
                    "Id": "65c89c44-f5c5-48d4-ba31-01e3d4b69e42",
                    "FieldName": "Temperature"
                },
                {
                    "Id": "117af450-3086-412f-90f5-ad63e1989d04",
                    "FieldName": "Humidity"
                }
            ]
        },
        {
```

```
            "Id": "252606d7-9b72-42d5-8923-ac8bce4f60e5",
            "FieldName": "Brightness"
        }
    ]
}
```

**Priedas nr. 3**

**Sugeneruota srautinio apdorojimo sistema „Heron“ sąsajoje**



## Priedas nr. 4

## Sugeneruotos srautinio apdorojimo sistemos rezultatai

| Raktas | Šviesumas | Drėgmė | Temperatūra | Juntamoji temperatūra |
|---|---|---|---|---|
| 101RYTAS | Suma:119666<br>Kiekis:2263 | Suma:114833<br>Kiekis:2435 | Suma:47115<br>Kiekis:2371 | Suma:77101.1<br>Kiekis:2290 |
| 101DIENA | Suma:103408<br>Kiekis:1983 | Suma:107138<br>Kiekis:2050 | Suma:41534<br>Kiekis:2080 | Suma:68657.23<br>Kiekis:2010 |
| 101VAKARAS | Suma:105595<br>Kiekis:2091 | Suma:98530<br>Kiekis:2109 | Suma:44271<br>Kiekis:2177 | Suma:74529.75<br>Kiekis:2175 |
| 102RYTAS | Suma:111895<br>Kiekis:2128 | Suma:99639<br>Kiekis:2016 | Suma:39654<br>Kiekis:2004 | Suma:69890<br>Kiekis:2077 |
| 102DIENA | Suma:117943<br>Kiekis:2302 | Suma:124621<br>Kiekis:2261 | Suma:46639<br>Kiekis:2276 | Suma:81079.7<br>Kiekis:2365 |
| 102VAKARAS | Suma:90282<br>Kiekis:1731 | Suma:94919<br>Kiekis:1819 | Suma:35410<br>Kiekis:1756 | Suma:61869.25<br>Kiekis:1814 |
| 103RYTAS | Suma:97362<br>Kiekis:2164 | Suma:103507<br>Kiekis:2064 | Suma:40542<br>Kiekis:2048 | Suma:70168.2<br>Kiekis:2073 |
| 103DIENA | Suma:92378<br>Kiekis:1879 | Suma:94471<br>Kiekis:1847 | Suma:38121<br>Kiekis:1919 | Suma:65073.73<br>Kiekis:1916 |
| 103VAKARAS | Suma:105595<br>Kiekis:2091 | Suma:101889<br>Kiekis:2103 | Suma:42007<br>Kiekis:2076 | Suma:70967.3<br>Kiekis:2068 |

## Priedas nr. 5

## Atnaujintas rodiklių duomenų modelis JSON formatu


```
{
    "IndicatorId": "32c24420-afbd-44fb-b045-ef72c48cb04e",
    "Name": "office-sensors-indicator",
    "VersionId": "2-a4c42274-65e9-4e9d-863b-671d51d702cf",
    "PrimaryKey": [
        "RoomNumber",
        "Daytime"
    ],
    "Filters": [
        {
            "FieldName": "RoomNumber",
            "Value": "100Room",
            "Operator": "NEQU"
        }
    ],
    "Values": [
        {
            "Id": "5dbfcf52-bbce-4ec6-ad31-1cf0812b3106",
            "FieldName": "GeneralIndex",
            "Formula": "(%252606d7-9b72-42d5-8923-ac8bce4f60e5% / 75) * (%5
                dbfcf52-bbce-4ec6-ad31-1cf0812b3106% / 25)",
            "NextValues": [
                {
                    "Id": "5dbfcf52-bbce-4ec6-ad31-1cf0812b3106",
                    "FieldName": "HeatIndex",
                    "Formula": "%65c89c44-f5c5-48d4-ba31-01e3d4b69e42% - (0.55
                        * (1 - (%117af450-3086-412f-90f5-ad63e1989d04%/1000))
                        * (%65c89c44-f5c5-48d4-ba31-01e3d4b69e42% - 14.5))",
                    "NextValues": [
                        {
                            "Id": "65c89c44-f5c5-48d4-ba31-01e3d4b69e42",
                            "FieldName": "Temperature"
                        },
                        {
```

```
                    "Id": "117af450-3086-412f-90f5-ad63e1989d04",
                    "FieldName": "Humidity"
                }
            ]
        },
        {
            "Id": "252606d7-9b72-42d5-8923-ac8bce4f60e5",
            "FieldName": "Brightness"
        }
    ]
  }
]
}
```

**Priedas nr. 6**

**Atnaujintos srautinio apdorojimo sistemos „Heron“ sąsajoje**

## Priedas nr. 7

## Atnaujintos srautinio apdorojimo sistemos rezultatai

| Raktas | Šviesumas | Drėgmė | Temperatūra | Juntamoji temperatūra | Bendra pojūtis |
|---|---|---|---|---|---|
| 101RYTAS | Suma:120834<br>Kiekis:2282 | Suma:115954<br>Kiekis:2459 | Suma:47426<br>Kiekis:2388 | Suma:77766.15<br>Kiekis:2296 | Suma:5.65<br>Kiekis:40 |
| 101DIENA | Suma:104169<br>Kiekis:1997 | Suma:108007<br>Kiekis:2067 | Suma:41923<br>Kiekis:2098 | Suma:69196.5<br>Kiekis:2025 | Suma:6.6<br>Kiekis:31 |
| 101VAKARAS | Suma:115994<br>Kiekis:2182 | Suma:99423<br>Kiekis:2125 | Suma:44762<br>Kiekis:2201 | Suma:75197.95<br>Kiekis:2145 | Suma:6.65<br>Kiekis:40 |
| 102RYTAS | Suma:112858<br>Kiekis:2146 | Suma:100588<br>Kiekis:2039 | Suma:40034<br>Kiekis:2023 | Suma:70383.5<br>Kiekis:2083 | Suma:4.483<br>Kiekis:30 |
| 102DIENA | Suma:118956<br>Kiekis:2321 | Suma:125430<br>Kiekis:2277 | Suma:46972<br>Kiekis:2292 | Suma:82100.9<br>Kiekis:2370 | Suma:14.09<br>Kiekis:60 |
| 102VAKARAS | Suma:91238<br>Kiekis:1749 | Suma:95147<br>Kiekis:1827 | Suma:35833<br>Kiekis:1777 | Suma:62468.9<br>Kiekis:1821 | Suma:4.48<br>Kiekis:34 |
| 103RYTAS | Suma:98434<br>Kiekis:2189 | Suma:104604<br>Kiekis:2087 | Suma:40912<br>Kiekis:2065 | Suma:70852.6<br>Kiekis:2082 | Suma:4.06<br>Kiekis:40 |
| 103DIENA | Suma:93000<br>Kiekis:1893 | Suma:95544<br>Kiekis:1867 | Suma:38529<br>Kiekis:1938 | Suma:65618.15<br>Kiekis:1933 | Suma:4.83<br>Kiekis:33 |
| 103VAKARAS | Suma:106695<br>Kiekis:2115 | Suma:102732<br>Kiekis:2121 | Suma:42451<br>Kiekis:2100 | Suma:71619.88<br>Kiekis:2087 | Suma:5.72<br>Kiekis:39 |