Jari Laurila

Proaktiivisten mittarien kehittäminen asuntovuokrausliiketoiminnan tarpeisiin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

10.4.2022

Tiivistelmä

Tekijä:	Jari Laurila
Otsikko:	Proaktiivisten mittarien kehittäminen asuntovuokrausliiketoiminnan tarpeisiin
Sivumäärä:	55 sivua
Aika:	10.4.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ohjaajat:	Talousjohtaja Mari Timonen Osaamisaluepäällikkö Janne Salonen

---

Insinöörityön aiheena ja tavoitteena oli toteuttaa proaktiivinen, eli ennakoiva mittari asuntojen taloudellisen käyttöasteen seurantaan. Ennakoivalla mittarilla tarkoitetaan tämän insinöörityön tapauksessa mittaria, joka näyttää tarkasteluhetken sopimustilanteen mukaista tulevien kuukausien käyttöastetta menneiden kuukausien käyttöasteen sijaan. Tavoitteena oli toteuttaa edellä mainittu mittari siten, että mittarin arvojen muutosta pystyy seuraamaan neljän edellisen viikon osalta vuorokauden tarkkuudella. Edellä mainitun mittarin lisäksi tavoitteena oli toteuttaa kaksi apumittaria asuntovuokrausliiketoiminnan toiminnan ohjaamisen tueksi ja toteuttaa päämittarin ja apumittarien avulla kerran vuorokaudessa päivittyvä Power BI -raportti.

Työn tuloksena tehty raportti toteutettiin toimeksiantajan Power BI -ympäristöön. Mittarien tietolähteinä käytettiin aiemmin muodostettua tietovarastoa sekä tämän työn yhteydessä ohjelmistorobotiikan avulla toimeksiantajan järjestelmästä haettavaa tietokokonaisuutta. Työn toteutus aloitettiin päämittarin tietotarpeiden analysoinnilla, päämittarin tiekokonaisuuden muodostamisella sekä päämittarin ja tähän liittyvien visualisointien muodostamisella. Tämän jälkeen siirryttiin analysoimaan apumittarien tietotarpeita. Tässä yhteydessä kävi ilmi, että apumittarien tiedot ovat saatavilla ainoastaan lähdejärjestelmän käyttöliittymän kautta, joten seuraavaksi muodostettiin ohjelmistorobotti tietojen hakuun. Tietotarpeet saatiin täytettyä ohjelmistorobotiikalla, jonka jälkeen muodostettiin apumittarien tietokokonaisuudet ja mittareihin liittyvät visualisoinnit. Lopuksi muodostetuista visualisoinneista koostettiin Power BI -raportti.

Tuloksena syntyneestä raportista näkee kuluvan kuukauden sekä kolmen seuraavan kuukauden käyttöaste-ennusteen tarkasteluhetken sopimuskannan mukaan laskettuna sekä jokaisen neljän kuukauden osalta käyttöaste-ennusteen kehityksen edellisen neljän viikon ajalta. Lisäksi raportilta näkee edellisen neljän viikon osalta irtisanomisten ja hakemusten käsittelyajan kehityksen. Raportista näkee hyvin visuaalisella tavalla esitettynä päivittäin tehdyn työn vaikutuksen tuleviin käyttöasteisiin. Lisäksi muodostetun raportin avulla osaston toiminnan ohjaamisen tueksi saadaan ennakoivaa tietoa ja tätä kautta mahdollistetaan ongelmatilanteisiin reagointivaiheessa, jossa mittarin lopullisiin arvoihin on vielä mahdollista vaikuttaa.

Avainsanat: Business Intelligence, tietovarastot, ohjelmistorobotiikka, mittaristot, raportointi, asuinhuoneiston vuokra

Abstract

Author:	Jari Laurila
Title:	Developing Proactive Indicators for Apartment Rental Business
Number of Pages:	55 pages
Date:	10 April 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Supervisors:	Group CFO Mari Timonen Head of School Janne Salonen

---

The subject and purpose of this thesis was to create proactive meters for reporting occupancy rate for apartments. In the context of this thesis proactive means displaying occupancy rates for future months instead of past months. The aim was to display history for development of the measure's values for past four weeks. The second purpose of the thesis was to create extra supporting measures for main measure and to create Power BI report for displaying the values of the created measures. The report was to be refreshed daily.

The resulting report was implemented in the client's Power BI environment. The data sources for the measures were previously established data warehouse, as well as the data set retrieved from the client's system user interface using robotic process automation. The implementation work started with the analysis of the information needs of the main meter. After the analysis the main meter was implemented and visualization for meter was created. The next part step was to analyse information needs for supporting measures. During the analysation the source data for supporting measures was found the to be available only through user interface of source system. Therefore, software robot for data retrieval was implemented to retrieve data from the user interface of source system. After all necessary information was available the supporting measures and visualisation were implemented after which, a Power BI report containing all created visualisations was created.

The resulting report shows forecast for current month's as well as next three month's occupancy rates. In addition, the development of values for the meters can be seen for previous four weeks. The report also shows past four weeks processing time for termination notices and applications. The implemented report gives a simple visual way the see the effect of the daily work done in apartment rental. The report also provides proactive information to support managing the operation of the department and enables reacting to problems in advance.

Keywords: Business Intelligence, data warehouses, RPA, robotic process automation, dashboards, reporting, apartment rental

Sisällys

Lyhenteet

ORM: *Object-relational mapping*. Ohjelmiston oliomallin mukaisen esityksen kuvaaminen relaatiotietokantamallin mukaiseksi esitykseksi ja kääntäen.

ETL: *Extract, Transform, Load*. Tietojen haku-, muokkaus- ja latausprosessi.

DDS: *Dimensional Data Store*. Ulottuvuusmuotoinen tietokantarakenne.

MDB: *Multidimensional Database*. Moniulotteinen tietokuutio.

BI: *Business Intelligence*. Liiketoimintatiedon hallinta.

CRM: *Customer Relationship Management.* Asiakkuudenhallintajärjestelmä.

RPA: *Robotic Process Automation.* Ohjelmistorobotiikka.

SQL: *Structured Query Language.* Kieli tietokantakyselyiden tekemiseen.

DAX: *Data Analysis Expressions.* Microsoftin kehittämä kieli, jota käytetään mm. PowerPivotissa Power BI:ssä ja SQL Server Analysis Services -järjestelmässä.

HTTP: *HyperText Transfer Protocol.* Hypertekstin siirtoprotokolla.

DQ: *Data Quality.* Laadunhallinta.

1 Johdanto

Monissa yrityksissä on käytössä erilaisia mittareita strategian toteutumisen seurantaan sekä päivittäisen toiminnan ohjaamisen tueksi. Usein mittarit ovat vähintään yrityksen johdon työkaluja, ja osassa yrityksistä mittarit tuodaan näkyville myös työntekijätasolla asti. Usein mittarit havainnollistavat jonkun tietyn yritykselle keskeisen suureen kautta yrityksen strategian toteutumista. (Ikäheimo ym. 2019: 149–150.) Monesti mittarit seuraavat jälkikäteen sitä, miten yritys on vaikkapa menneen kuukauden aikana suoriutunut mitattavan suureen suhteen tai miten tällä hetkellä suoriudutaan jonkun tietyn asian suhteen.

Toimeksiantajayrityksen liiketoiminnan yhtenä päämittarina on taloudellinen käyttöaste, joka kuvaa sitä, kuinka suuri prosentti sellaisen tilanteen vuokratuotosta, jossa kaikki asunnot ovat vuokrattuja, on saavutettu. Mittari lasketaan kuukausittain kullekin menneelle kuukaudelle. Lisäksi mittarille lasketaan liukuva keskiarvo 12 edellisen kuukauden osalta.

Vuokra-asunnoista tiedetään irtisanotut asunnot yhtä kalenterikuukautta ennen vuokrasopimuksen päättymistä ja asumisoikeusasunnoista kolmea kuukautta ennen sopimuksen päättymistä (Laki asuinhuoneiston vuokrauksesta. 1995. 52§; Laki asumisoikeusasunnoista. 2021. 57§). Myös uudet alkavat sopimukset tiedetään sekä vuokra- että asumisoikeusasuntojen osalta monesti hyvissä ajoin ennen sopimuksien alkua. Täten käyttöasteelle olisi mahdollista laskea ennustetta useampi kuukausi tulevaisuuteen. Jos eletään vaikkapa tammikuuta, olisi mahdollista laskea käyttöaste-ennuste nykyisen sopimuskannan perusteella esimerkiksi maaliskuun lopun tilanteelle.

Tämän insinöörityön tavoitteena on luoda proaktiivinen, eli ennakoiva mittari taloudellisen käyttöasteen seurantaan. Lisäksi tavoitteena on muodostaa kaksi uutta apumittaria toiminnan ohjauksen tueksi. Ensimmäinen apumittari mittaa irtisanomisten ja toinen hakemusten käsittelyaikaa. Toimeksiantajayrityksellä ei ole tähän asti ollut käytössään ennakoivaa käyttöastemittaria tai irtisanomisten

käsittelyaikaa kuvaavaa mittaria. Hakemusten käsittelyaikaa kuvaava mittari on laadittu mittari jo aiemmin, mutta mittari muodostetaan tällä hetkellä karkeammalla tarkkuudella kuin tässä työssä esitettävä mittari. Myös muut aiemmin luodut vastaavien suureiden mittarit on laskettu kuukausitasolle eikä päivätasolle, kuten tässä työssä laadittavat mittarit, joten tämän insinöörityön kautta saadaan uusia työkaluja päivittäisen toiminnan johtamiseen ja tuodaan aiemmin puuttunutta näkyvyyttä päivätasolla tehtyyn työhön.

Insinöörityön toimeksiantaja YH Kodit Oy on Länsi-Suomen alueella toimiva yritys, joka omistaa noin 8000 vuokra- ja asumisoikeusasuntoa pääosin Tampereen ja Turun seuduilta ja hoitaa sekä asuntojen vuokrauksen että isännöinnin. Vuokra-asuntokanta pitää sisällään sekä valtion tukemaa asuntotuotantoa, jonka asukasvalintaan kohdistuu kohteesta riippuen tiettyjä rajoituksia, että nk. vapaarahoitteista tuotantoa, joissa em. rajoituksia ei ole. Lisäksi yritys rakennuttaa asuntoja sekä omille yhtiöilleen että ulkopuolisille tahoille. YH Kodit Oy:llä on toimipisteet Tampereella ja Turussa, ja yritys työllistää n. 50 henkilöä. (Vastuullisuuskertomus 2020 2021. 4, 16, 17.)

Tämä työ keskittyy asuntovuokrauksen osa-alueeseen, joka on osa manageerausliiketoimintaa. Manageerauksen päätyöväline on Visma Tampuuri -kiinteistöhallintajärjestelmä, jossa hoidetaan mm. asuntojen vuokraus, kiinteistöjen ylläpidon seuranta sekä vuokrareskontra. Tässä työssä käytetyt tiedot perustuvat Tampuurista haettuihin tietoihin. Suurin osa tiedoista haetaan Tampuurin ohjelmistotoimittajan muodostaman tietovaraston tietojen pohjalta muodostettuun YH Kotien omaan SQL Server -alustalla toimivaan tietovarastoon. Lisäksi em. tietoja täydennetään muuta kautta haetulla tiedolla niiltä osin, kun tieto ei ole saatavilla Tampuurin tietovaraston kautta. Työn tuloksena laadittava raportti tehdään Microsoft Power BI -alustalle.

2 Suorituskyvyn mittaus, mittarit ja raportointi

Useimmiten strategian mielletään olevan suunnitelma, jossa määritellään se, mihin yrityksen liiketoiminta suuntautuu nyt ja tulevaisuudessa ja mikä liiketoiminnan laajuuden suunnitelma on pidemmällä aikavälillä ja millä keinoilla tavoitteisiin pyritään. Monesti strategia pitää sisällään myös asemoitumisen liiketoimintaympäristöön siten, että yrityksen toiminta on mahdollisimman kannattavaa. Nykyään strategia-sanaa tosin käytetään myös muissa asiayhteyksissä, kuten kuvaamaan tapaa yksittäisen asian läpivientiin tai kuvaamaan pienempää yritykselle tärkeää asiaa tai tavoitetta. Tämä ei kuitenkaan ole termin alkuperäinen merkitys. (Hämäläinen ym. 2016: 3. luku.)

Koska strategia kuvaa yrityksen pitkän aikavälin tavoitteita, voidaan strategiatyön tuloksena laatia esimerkiksi tiekartta, jossa määritellään toteutuksen aikataulu ja järjestys sekä laaditaan ylätason tavoitteet kunkin tiekartan kohdan toteutumisen seurannan tueksi. Toisin kuin aiemmin, nykyään strategia nähdään myös useimmiten mukautuvana työkaluna, joka elää kulloinkin vallitsevan tilanteen mukaan ja mukautuu toimintaympäristön muuttuessa. Vaikka strategian laadintaan osallistetaan nykyään usein myös henkilöstöä, ymmärrys strategiasta jää monesti hyvin vajavaiseksi ylimmän johdon ulkopuolella. (Hämäläinen ym. 2016: 3. luku.)

Jotta strategian toteutuminen saadaan tuotua näkyväksi, laaditaan monesti strategiasta johdettua suorituskykymittareita, joilla seurataan valittujen suureiden kautta yrityksen onnistumista strategian suhteen. Mittari voi olla esimerkiksi talouteen tai operatiiviseen toimintaan liittyvä. Suorituskykymittareille on myös muita määrityksiä, mutta tämä on yksi yleisesti käytetyistä. Keskeisimmistä mittareista käytetään usein nimitystä KPI-mittari, joka tulee englannin kielen sanoista Key Performance Indicator, eli vapaasti suomennettuna avainsuorituskykymittari. Kaikki mittarit eivät välttämättä ole sellaisia, joihin yrityksen työntekijät voivat suoraan vaikuttaa, mutta useimmiten ainakin suurimman osan mittareista on hyvä olla sellaisia, joihin työntekijät voivat vaikuttaa ainakin välillisesti omalla

työllään ja jotka eivät sisällä juurikaan sellaisia tekijöitä, joihin henkilöstö ei pysty itse millään tavalla vaikuttamaan. (Ikäheimo ym. 2019: 149–151.)

Tässä insinöörityössä käsitellään yhtä toimeksiantajayrityksen strategian pohjalta muodostetuista päämittareista, projisoidaan mittarin luvut tulevaan aikajaksoon ja tuodaan seurannan taso kuukausitasolta päivätasolle. Tällä pyritään liittämään strategia osaksi henkilöstön päivittäistä työntekoa tuomalla päivittäin tehty työ näkyväksi strategisen mittarin kontekstissa ja luomalla uusia päämittarin muutoksen syitä ainakin osittain selittäviä apumittareita. Myös apumittarien osalta mittarit tehdään päivätasolle, jotta tehdyn työn vaikutus saadaan näkyviin päivittäin ja yhteys päämittarin lukujen päivittäisiin muutoksiin säilyy. Apumittareiden mitattavat suureet on valittu siten, että henkilöstöllä on suora vaikutusmahdollisuus mittarin arvoihin.

3 Tietojen haku ja tietovarastot

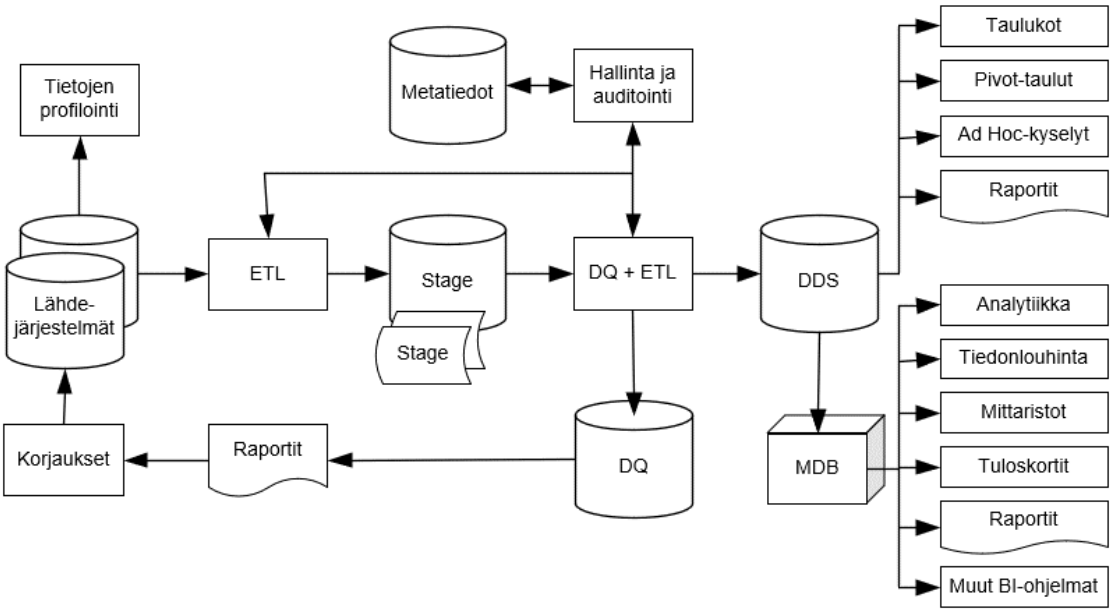
Mittarien ja raporttien muodostamista varten tarvittavat tiedot sijaitsevat yleensä pääasiassa organisaation operatiivisissa järjestelmissä. Organisaatioilla on monesti useita erillisiä operatiivisia järjestelmiä, joita käytetään eri toiminnoissa tai eri toiminnon osissa. Usein järjestelmien tuottamat tiedot ovat myös sellaisessa muodossa, joka ei ole paras mahdollinen raportointia ajatellen. Jotta eri järjestelmien tiedot saadaan yhdistettyä yhtenäiseksi kokonaisuudeksi ja muokattua raportointiin soveltuvaan muotoon, voidaan valita useita eri lähestymistapoja ongelman ratkaisemiseksi. Lähdetiedoista voidaan muodostaa esimerkiksi tietovarasto ja tehdä raportit tämän tietovaraston päälle. Vaihtoehtoisesti raportointia voidaan tehdä myös suoraan lähdetietojärjestelmien tiedoista. Myös näiden yhdistelmä on mahdollinen. Se, mitä lähestymistapaa käytetään, riippuu täysin tarpeesta ja käytettävistä resursseista. Seuraavassa luvussa on kuvattu lyhyesti tietovarastojen rakenne sekä tietovarastoihin liittyvät prosessit.

3.1 Tietovarastojärjestelmät

Tietovarastojärjestelmä koostuu useasta osajärjestelmästä, joista kaikkia ei välttämättä löydy jokaisesta tietovarastojärjestelmästä. Kuvassa 1 on esitelty osajärjestelmät.

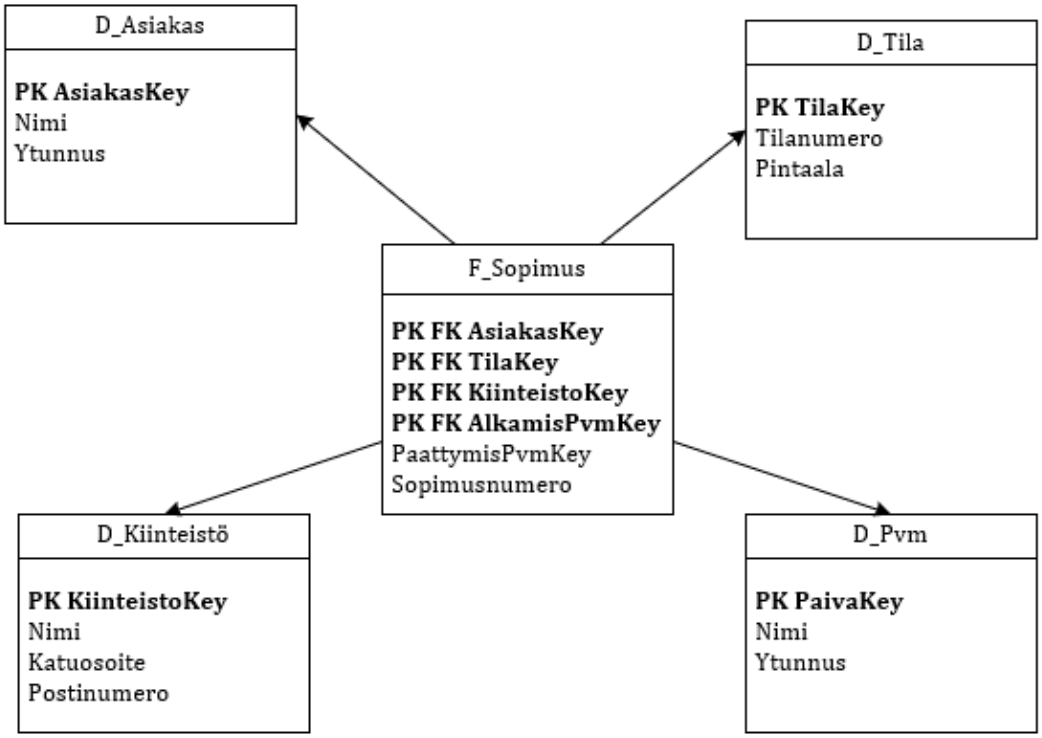
Ennen varsinaista tietojen tuontia lähdetietojärjestelmien tietoja voidaan tarkastella tietojen profilointiprosessin avulla, joka kerää tietoja mm. siitä, kuinka suuria tietokokonaisuudet ovat ja onko tiedoissa esimerkiksi paljon tyhjiä arvoja. (Rainardi 2008: 2.)

Itse tietojen tuonti lähtee liikkeelle lähdejärjestelmistä, joita ei kuitenkaan varsinaisesti lasketa osaksi tietovarastoa. Tiedot tuodaan ETL-prosessilla, jonka voi suomentaa tietojen haku-, muokkaus- ja latausprosessiksi. Ensimmäisessä vaiheessa tietoa ei juurikaan muokata, vaan tiedot tuodaan lähes alkuperäisenkaltaisina Stage-tietokantaa, joka toimii eräänlaisena välitietokantana jatkoprosessointia varten. Stage-tietokantaan tuodaan myös usein tietoja useammasta lähteestä, ja osa kyseiseen tietokantaan tuoduista tiedoista voi olla myös pysyvämmin välitietokantaan tallennettua tietoa. Tämän jälkeen tietojen laatu tarkastetaan DQ-prosessissa (Data Quality) ja virheelliset tiedot siirretään DQ-tietokantaan ja raportoidaan eteenpäin korjauksia varten. Samassa yhteydessä tietojen laaduntarkastuksen kanssa tiedot yhdistellään ja prosessoidaan lopulliseen DDS-, eli dimensional data storage-tietokannan esitysmuotoon ja tehdään varsinaisen tietojen vienti. (Rainardi 2008: 2.)



Kuva 1. Esimerkki tietovarastojärjestelmästä (Muokattu lähteestä Rainardi 2008: 2)

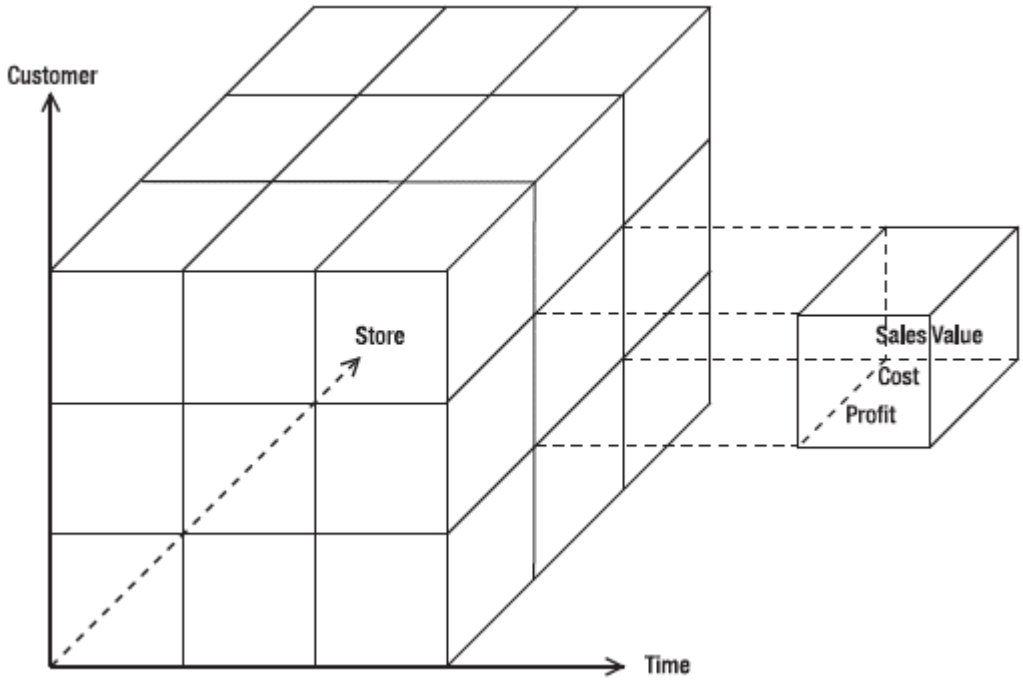
DDS-tietokannan eli ulottuvuusmuotoisen tietokannan muoto riippuu siitä, mitä tarkoitusta varten tietokanta muodostetaan. Usein raportointitarkoituksiin käytettävä tietokanta on esimerkiksi tähti- tai lumihiutalemallinen. Tähti ja lumihiutale kuvaavat sitä, minkälaisiin tietokantatauluihin tiedot on järjestetty. Tähti-muotoisessa tietokannassa tiedot ovat järjestetty esimerkiksi siten, että faktataulut, eli tapahtumataulut on ympäröity yhdellä tasolla dimensio, eli tietotauluja. Kuvassa 2 on esimerkki tähtimuotoon järjestetystä tietokannasta.



Kuva 2. Tähtimuotoon järjestetty tietovarastotietokanta. F_Sopimus on fakta, eli tapahtumataulu ja D-alkuiset taulut ovat dimension eli tietotauluja.

DDS-tiekannoista voidaan muodostaa myös MDB-tietokanta, eli multidimensional database, joita kutsutaan monesti myös nimellä kuutio. MDB-tietokantojen muoto on hyvin erilainen verrattuna perinteisiin tietokantoihin, ja läheskään kaikki raportointiin käytettävät työkalut eivät osaa käyttää MDB-tietokantojen tietoja. Kuutioiden etuina voidaan pitää mm. sitä, että raportointiin käytettyjä tietoja lasketaan kuutioissa valmiiksi ja raportointia voi tehdä helposti ja nopeasti eri kuution ulottuvuuksien suhteen. Toisaalta koska kuution tiedot lasketaan valmiiksi, tietokanta vaatii esiprosessoinnin, jossa luvut muodostetaan ja joka voi kestää suhteellisen pitkään. Kuutio voi olla muodostettu vaikkapa akseleille asiakas, myymälä ja aika siten, että kuution jokaisessa solussa on kolme arvoa: myyntien summa, kustannukset ja kate (kuva 3). Tällöin raportoinnissa voidaan helposti analysoida myyntiä eri akselien suhteen. Toisaalta rakenteen suhteen tehdyt valinnat myös rajoittavat sitä, minkälaisia raportteja tiedoista voidaan

muodostaa. MDB-tietokantoja käytetään pääasiassa analytiikkaan ja tiedonlouhintaan. (Rainardi 2008: 3, 377-378.)



Kuva 3. MDB-tietokanta, eli kuutio, jossa katetta (profit), myyntien summaa (sales value) ja kustannuksia (cost) voidaan tarkastella asiakkaiden (customer), myymä-län (store) ja ajan (time) suhteen. (Rainardi 2008: 3.)

Järjestelmään voi liittyä myös hallinnointi ja auditointi -järjestelmä, joka kirjaa lokitietoa järjestelmän toiminnasta metatietokantaan. Metatietokanta pitää sisällään myös määrityksiä tiedon rakenteesta ja muista lähdetietojen prosessointiin tarvittavista tiedoista ja määrityksistä. (Rainardi 2008: 3.)

Kuten aiemmin mainitsin, kaikissa tietovarastojärjestelmissä ei ole kaikkia tässä esitettyjä osajärjestelmiä. Jotta järjestelmää voidaan kutsua tietovarastoksi, tulee järjestelmästä kuitenkin löytyä vähintään ETL-prosessi sekä DDS-tietokanta ja tämän kokonaisuuden tulee liittyä ainakin yhteen lähdejärjestelmään, jota ei kuitenkaan lasketa osaksi tietovarastojärjestelmää.

Yhteenvetona voidaan todeta, että tietovarasto hakee useimmiten tietoja säännöllisesti lähdejärjestelmistä ja yhdistelee-, sekä normalisoi tietoja, jolloin tietokanta saadaan muodostettua siten, että raporttien luonti on helppoa ja onnistuu järjestelmärajojen yli. Lisäksi tietovarastoilla ja ohjelmistojen omilla tietokannoilla on yleensä yksi oleellinen ero, joka on se, että ohjelmistojen omat tietokannat säilyttävät usein historiaa hyvin rajatulta ajalta, välillä jopa ainoastaan jonkin kohteen viimeisintä versiota, kun taas tietovarastoissa säilytetään useimmiten historiatietoja hyvinkin pitkältä ajalta. Tietovaraston tietoja käytetään ohjelmistojen omista tietokannoista poiketen myös pääasiassa muiden tietolähteiden tietojen tallennukseen, eli tietovarasto ei ole suoraan minkään tuotantojärjestelmän päätietokanta. (Rainardi 2008: 4)

3.2 Raportointijärjestelmät

Jotta tietovarastojen tietoja voidaan hyödyntää raportointiin, täytyy tietovaraston tietojen päälle muodostaa raportteja jollain raportointityökalulla. Toisin sanoen tietovarasto itsessään ei pidä sisällään raportoinnin osuutta. Sekä DDS-, että MDB-tietokantoihin liitettäviä raportointityökaluja löytyy todella paljon ja se, mikä työkalu valitaan raportointiin, riippuu raporteille asetetuista vaatimuksista, sekä mm. lisensointikustannuksista. Kaksi yleisintä käyttötapausta tietovarastojen tiedoille ovat BI-raportit, eli liiketoimintatiedonhallinnan raportit, sekä CRM, eli asiakkuudenhallinta, jotka molemmat vaativat omanlaisen lähestymistavan raporttien muodostamisen näkökulmasta. (Rainardi 2008: 329)

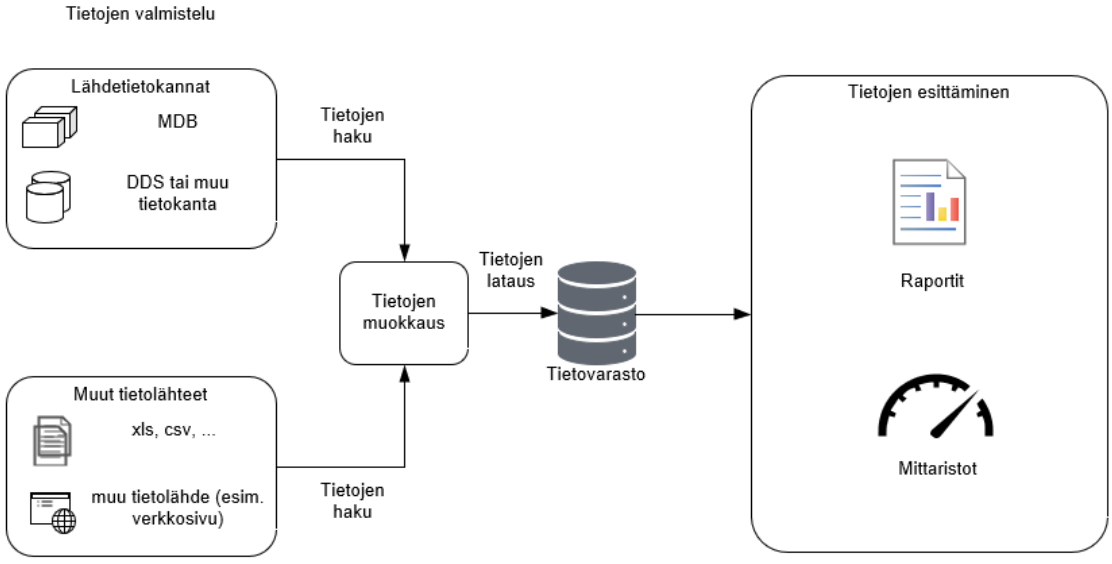
3.3 Power BI

Power BI on Microsoftin kehittämä järjestelmä, jolla voidaan yhdistellä tietoa useista eri lähteistä ja visualisoida nämä tiedot raporteiksi ja koontinäytöiksi, sekä jakaa tiedot halutulla laajuudella. (Mikä Power BI on? 2022)

Power BI-nimi pitää itse asiassa sisällään useita eri ohjelmistoja ja palveluita. Näistä käytetyimmät ovat ilmainen Power BI Desktop, jolla voidaan tehdä tiedonmallinnusta ja raportointia, sekä Power BI -palvelu, jonka kautta voidaan

esimerkiksi jakaa raportteja sekä ajastaa raporttien tietolähteiden päivitys. Power BI -palvelussa käytettävissä oleva toiminnallisuus riippuu käytössä olevasta lisenssistä. Pienille organisaatioille (alle 500 käyttäjää) sopivin on useimmiten Power BI Pro tai Power BI Premium Per User -palvelu, jossa jokaisella raportin muokkaajalla ja katselijalle tulee olla käytössä vähintään Power BI Pro -lisenssi. (Enho 2021.)

Power BI:n arkkitehtuuri pitää sisällään raportoinnin lisäksi tietojen haun, muokkauksen sekä latauksen, eli ETL-prosessin osat, kuten kuvasta 4 käy ilmi. Power BI -järjestelmällä voidaan täten hoitaa niin tietovaraston muodostus että itse raportointi. Lähteenä voi kuitenkin olla myös aiemmin muodostetut tietovarastot ja useimmiten näitä tietolähteitä käytetäänkin raportoinnin pohjana, koska tietovarastoihin tuotu data on yleensä valmiiksi siivottua ja sopivaa muotoon muunnettua, jolloin oikeellisten raporttien muodostaminen tietojen pohjalta on huomattavasti helpompaa kuin lähdejärjestelmien tietokannan käyttäminen suoraan. (Verbeeck 2020.)



Kuva 4. Power BI:n arkkitehtuuri (Muokattu lähteestä Power BI Tutorial – A Complete Guide on Introduction to Power BI)

Vaikka raportoinnin pohjana toimisikin aiemmin muodostettu tietokanta, voidaan tietoihin yhdistää tietoja myös muista lähteistä. Usein analysoinnin tueksi tuodaan esimerkiksi julkisten tietovarantojen tietoja. Toinen syy yhdistellä tietoja voi olla myös se, että tietovaraston tietosisältöä ei haluta laajentaa käsittämään kaikkea sitä tietoa, jota raportointiin tarvitaan, oli tietotarve ulkoinen tai yrityksen omiin tietolähteisiin tukeutuva. Syynä sille, että tietovaraston sisältöä ei haluta laajentaa, voi olla mm. se, että yrityksen varsinaisen tietovarastojen muokkaaminen on liian raskas tai hintava prosessi tästä saatavaan hyötyyn nähtynä, koska tietoa ei tarvita muissa raporteissa. Tällöin lähestymistapana voidaan käyttää tietojen tuontia suoraan Power BI -järjestelmän tietomalliin. (Verbeeck 2020.)

Power BI tukee huomattavan laajaa joukkoa erilaisia tietolähteitä, oli kyse strukturoidusta, osittain strukturoidusta tai strukturoimattomasta tiedosta. Tyypillisiä strukturoituja tietolähteitä ovat esimerkiksi erilaiset tietokannat. Osittain strukturoituja tietolähteitä ovat mm. json- tai xml-muotoiset tielähteet, joissa tiedot on jaettu jonkinlaiseen rakenteeseen. Tämä pitää sisällään esimerkiksi tietokenttiä nimi- arvopareina. Strukturoimattomina lähteinä voidaan pitää vaikkapa teksti-, kuva- tai äänitiedostoja. (Gramlich 2020.)

Jossain tapauksissa Power BI:n tiedontuontimahdollisuudet eivät kuitenkaan laajasta tietolähdetuesta huolimatta riitä. Näin voi käydä esimerkiksi tilanteessa, jossa tietoja täytyy tuoda monimutkaisen verkkopalvelun käyttöliittymästä. Tällöin avuksi voi ottaa ohjelmistorobotiikan, jolla tehdään tietojen haku ja muokkaus Power BI:lle sopivampaan muotoon.

Tässä insinöörityössä raportin muodostamiseen käytetään Power BI -järjestelmää. Ohjelma oli jo ennestään käytössä toimeksiantajayrityksellä, joten raportin muodostamista varten ei tarvinnut hankkia uusia lisenssejä. Näin työn tekeminen ei aiheuttanut lisäkustannuksia toimeksiantajalle, ja raportin muodostamisen aiheuttamat osaamistarpeet olivat huomattavasti kevyemmät kokonaan uuteen raportointijärjestelmään siirtymiseen verrattuna.

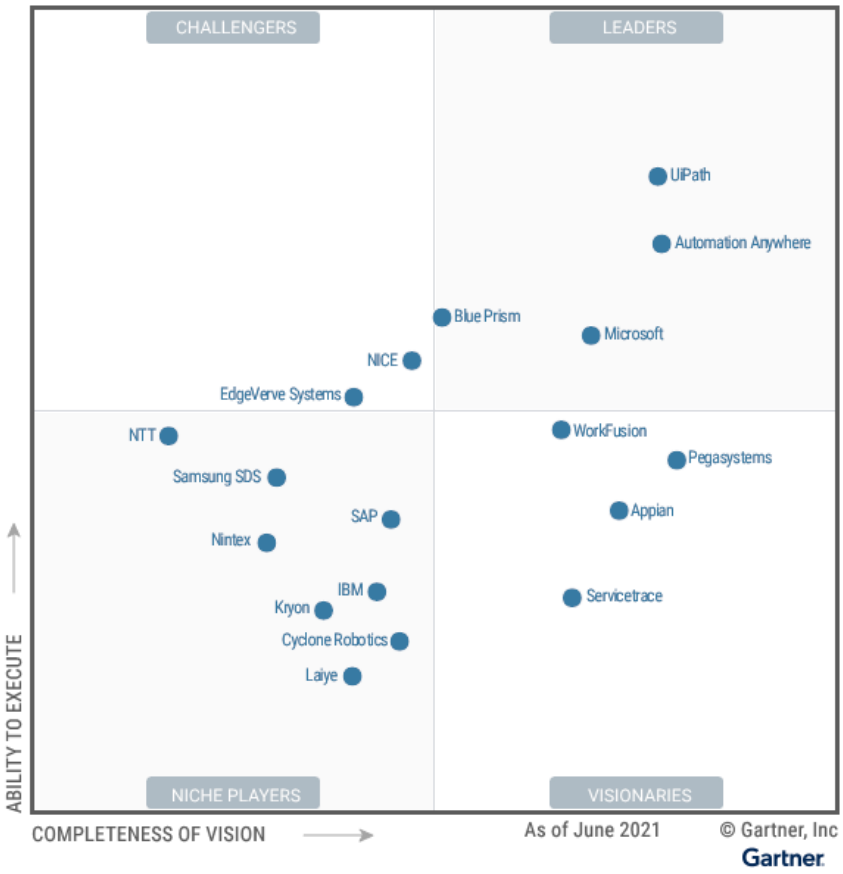
3.4 RPA eli ohjelmistorobotiikka raportoinnissa

Ohjelmistorobotiikalla tarkoitetaan tietyn tehtävän tekemistä ohjelmistorobotilla, eli ohjelmistolla, joka osaa tarpeesta riippuen käyttää esimerkiksi www-selainta ihmiskäyttäjän tavoin ja joka suorittaa tehtävän vaatimat vaiheet yleensä ilman ihmisen ohjausta. Monesti ohjelmistorobottia käytetään suorittamaan usein toistuvia rutiininomaisia tehtäviä ihmisen sijaan silloin, kun tehtävien automatisointiin ei ole muita keinoja, kuten ohjelmiston sisään rakennetut automaatiot tai rajapintojen kautta suoritettavat automaatioprosessit. Usein ohjelmistorobotit voivat myös suorittaa tehtävään liittyviä vaiheita useissa eri ohjelmistoissa ja järjestelmissä. (Taulli 2020: 3-4.)

Ohjelmistorobotiikkaa voi käyttää myös raportointiin vaadittavien tietojen hakuun lähdejärjestelmistä. Näin toimitaankin monesti erityisesti tilanteissa, joissa lähdejärjestelmä ei mahdollista tietojen automaattista vientiä järjestelmästä muilla keinoilla tai järjestelmän raportointia varten muodostama tieto on puutteellista raportoinnin vaatimuksiin nähtynä. Ensisijaisena keinona hakea tiedot raportointia varten on kuitenkin useimmiten lähdejärjestelmän itse muodostama aineisto tai lähdejärjestelmän tietokanta.

Ohjelmistorobotit ovat monesti suhteellisen nopeita ja kevyitä toteuttaa verrattuna perinteisen ohjelmistokehityksen kautta tehtyyn kehitykseen. Ohjelmistorobotiikalla on kuitenkin omat haasteensa, joista yksi on se, että muutokset lähdejärjestelmissä aiheuttavat monesti muutostarpeita myös ohjelmistorobotteihin. Näin voi käydä vaikkapa ohjelmiston käyttöliittymän muutoksen yhteydessä. Toisin kuin ihminen, ohjelmistorobotti ei osaa tulkita esimerkiksi paikkaansa muuttanutta painiketta samaksi painikkeeksi. Mikäli edellä mainitut tilanteet otetaan kuitenkin oikein huomioon robottia suunnitellessa, ongelmatilanteet ovat kuitenkin helppoja huomata, jonka jälkeen robotin voi helposti mukauttaa toimimaan myös muutosten jälkeen. Ohjelmistorobotin käyttäminen toistuvien tehtävien suorittamiseen automaatioon sopivissa kohteissa on useimmiten joka tapauksessa huomattavasti halvempaa, nopeampaa ja varmempaa kuin se, että ihminen tekisi vastaavat tehtävät. (Taulli 2020: 10-14; Tanskanen 2018.)

Markkinoilla on suuri määrä eri tasoisia ja eri käyttötarkoituksiin soveltuvia RPA-ohjelmistoja. Suuri osa ohjelmistoista on kaupallisia, mutta myös muutama ilmainen avoimen lähdekoodin ohjelmisto löytyy. Kuten kuvasta 5 voi tulkita, kaupallisista ohjelmistoista tällä hetkellä johtavana voidaan pitää UI.Path-nimistä ohjelmistoa, joka on selkeä johtaja toteuttamiskyvyn (ability to execute) ja lähes kärjessä myös vision täydellisyyden (completeness of vision) akselilla. (Ray ym. 2021). Avoimen lähdekoodin ilmaisia ohjelmistoja ovat mm. Robot Framework ja OpenRPA. Lisäksi löytyy avoimen lähdekoodin ohjelmistoja, jotka markkinoivat itseään ensisijaisesti ohjelmistotestaukseen suunniteltuina ohjelmistoina, mutta joita käytetään yleisesti myös ohjelmistorobotiikkaan. Tällaisia ohjelmistoja ovat mm. pitkään markkinoilla ollut Selenium, jolla voi automatisoida selainta testaustarkoitusta varten sekä uudempi Microsoftin kehittämä Playwright, joka on suunniteltu testaukseen selainautomaation avulla samoin kuin Selenium.



Kuva 5. Kaupallisten RPA-ohjelmistojen asema markkinoilla. (Ray. 2021)

3.5 Playwright

Kuten aiemmin mainitsin Playwright, joka on oikeastaan ohjelmistokirjasto, eikä varsinainen ohjelmisto, on suhteellisen uusi tulokas selainten automatisoinnin kentällä. Ohjelmistokirjastoa käytetään yleisesti myös selainkäyttöisten sovellusten ohjelmistorobotiikkaan. Kirjasto mainitsee vahvuuksikseen mm. sen, että kirjastolla saa tehtyä luotettavia automaatioita, koska kirjaston suunnittelussa on otettu alusta asti huomioon tietyt selainkäyttöisiin ohjelmistoihin liittyvät ominaispiirteet. Lisäksi vahvuuksiksi mainitaan laaja selaintuki, sekä syvä integraatio selainten toimintoihin. (Tapper 2021.)

Toisin kuin monissa varsinaisissa RPA-ohjelmistoissa, Playwrightissa ei ole varsinaista käyttöliittymää, vaan kaikki toiminnot tulee toteuttaa itse ohjelmakoodilla. Playwright-kirjastoa voi käyttää Python- ja Javascript-, sekä monen muun ohjelmointikielen kautta. Kaupalliset RPA-ohjelmistot ovat usein nk. Low-Code-ohjelmistoja, eli ohjelmistoja, joissa toiminnot saa tehtyä pienellä määrällä yksinkertaista koodia ja itse koodaus tehdään useimmiten ohjelmiston käyttöliittymän kautta. Lisäksi RPA-ohjelmistoissa on usein valmiiksi toteutettuna tarvittavat toiminnot esimerkiksi virheenhallintaan tai suuren robottimäärän orkestrointiin sekä monia robottien ohjelmointityötä helpottavia ominaisuuksia.

Playwrightin valinta voi olla perusteltu RPA-projektiin, mikäli tarvittava ohjelmointiosaaminen löytyy jo valmiiksi ja automaatiosta saatu hyöty ei ylitä kaupallisten ohjelmistojen hankintakustannuksia. Kaupallisten ohjelmistojen hankintakustannukset ovatkin yleensä varsinkin pienemmille yritykselle merkittävät, joten kovin pieneen tarpeeseen ohjelmistoa ei kannata yleensä hankkia. Tietyt toimijat ovat alkaneet myös vuokraamaan ohjelmistorobotteja siten, että korvausta vastaan saa vuokrattua jokaista vuorokautta kohden tietyn minuuttimäärän robotin ajoaikaa. Tällöin suurilta hankintakustannuksilta vältytään, mutta toisaalta robottia ei rajoitetun ajoajan takia pysty välttämättä käyttämään esimerkiksi useita kertoja vuorokauden sisällä tehtävän prosessin automatisointiin ja robotille ei välttämättä pääse muodostamaan itse ajettavia tehtäviä.

Tässä insinöörityössä päädyttiin käyttämään raporttien tietojen haun automatisointiin Playwright-ohjelmistokirjastoa. Valintaan päädyttiin muun muassa, koska kirjasto oli jo tuttu aiemman käytön perusteella ja kirjasto käyttäminen ei edellyttänyt uuden ohjelmiston hankkimista. Avoimen lähdekoodin tuotteiden joukosta Playwrightin valintaan päädyttiin, koska Playwright oli uusimmasta päästä ja verrattuna esimerkiksi Selenium-kirjastoon, Playwrightilla pystyi tekemään aiemman kokemuksen perusteella pienemmällä vaivalla varmemmin toimivia ohjelmistorobotteja. Lisäksi kirjastoa pystyi käyttämään Python-ohjelmointikielellä, joka oli jo entuudestaan hyvin tuttu.

4 Mittarien valinta, suunnittelu ja toteuttaminen

Mittareita valitessa on hyvä tiedostaa jo valintahetkellä se, mistä tarvittavat tiedot mittarien toteuttamiseen löytyvät. Jos mittarin laatimiseen vaaditaan esimerkiksi suuri määrä manuaalista työtä tai mittarissa käytettävä tietolähde ei ole riittävän luotettava, ei mittaria kannata välttämättä muodostaa ollenkaan. Tällöin voi olla kannattavampaa valita joku muu mittari. Lisäksi mittarin on hyvä olla sellainen, johon työntekijät pystyvät vaikuttamaan työllänsä ja jossa viive tehdyn työn ja mittarin arvon muuttumisen välillä ei ole kohtuuttoman pitkä. Seuraavaksi tarkastellaan valittuja mittareita, mittareiden valintaperusteita ja tietolähteitä.

4.1 Taloudellisen käyttöasteen mittari

Yksi toimeksiantajayrityksellä tällä hetkellä käytössä olevista, yrityksen strategiatyön pohjalta muodostetuista mittareista, on taloudellisen käyttöasteen mittari. Mittari kuvaa sitä, kuinka suuri prosentuaalinen osuus vuokratuotoista on saavutettu tarkasteluajanjaksona verrattuna tilanteeseen, jossa kaikki tilat ovat vuokrattuja. Mittaria tarkastellaan tällä hetkellä jälkikäteen kuukausittain.

Nykyisen käyttöastemittarin laskemiseen käytetään kohdekuukauden vuokratavoitetta, eli sitä summaa, jolla tila vuokrataan, ja vuokratoteumaa, eli summaa,

joka asiakkaalta on kohdekuukautena laskutettu. Mittarien tarvitsemat tiedot siirretään kerran vuorokaudessa ohjelmistotoimittajan raportointia varten muodostamaan tietokantaan, josta tiedot siirretään eteenpäin yrityksen omiin järjestelmiin. Tietojen pohjalta muodostetaan yrityksen oma DDS-muotoinen tietovarasto sekä MDB-tietokanta, joita molempia käytetään yrityksen raporttien ja mittarien tietolähteinä.

Ajatus tämän insinöörityön aiheesta lähti liikkeelle siitä huomiosta, että kaikki taloudellisen käyttöasteen mittarin muodostamiseen tarvittavat tiedot ovat tiedossa riittävälle tarkkuudelle myös lähitulevaisuuden osalta. Vaikka ohjelmistotoimittajan raportointitietokanta pitää sisällään vuokratoteumien osalta ainoastaan viimeisintä päättynyttä kuukautta koskevia ja tätä aiempia tietoja, sopimuksien alkamis- ja päättymisajankohdat ovat tiedossa jo aiemmin ja kyseiset tiedot tuodaan raportointitietokantaan, kuten myös asiakkaan maksulajit, eli vuokralaskulla kuukausittain veloitettavat hinnat. Näiden tietojen perusteella on täten mahdollista muodostaa käyttöastemittari, joka kuvaa senhetkisen sopimustilanteen mukaista parasta tietoa tulevaisuuden käyttöasteesta. Lisäksi tälle mittarille voidaan näyttää päiväkohtaista historiatietoa, koska sopimuksilta on tiedossa myös allekirjoitus- ja irtisanomispäivät, joita voidaan käyttää rajaamaan tarkastelupäivänä tiedossa olleet muutokset sopimusten alkamis- ja päättymispäiviin.

Aiemman käyttöastemittarin ongelmana voidaan pitää historiaan katsomisen lisäksi sitä, että mittari päivittyy ainoastaan kerran kuukaudessa. Tällöin yhteys päivittäin tehtyyn työhön jää helposti etäiseksi. Nämä kaksi uutta asiaa, eli historian sijasta tulevaisuuteen katsominen, sekä tarkastelujakson tuominen kuukausitasolta päivätasolle, lisäävät varmasti mittarin kiinnostavuutta. Käyttöastemittari on mukana myös henkilöstön tulospalkkioperusteissa, joka entisestään lisää mielenkiintoa mittaria kohtaan, etenkin koska uuden raportin myötä mittarin lopulliseen arvoon pystyy vielä vaikuttamaan.

4.1.1 Tietolähteet, tietojen haku ja muokkaus

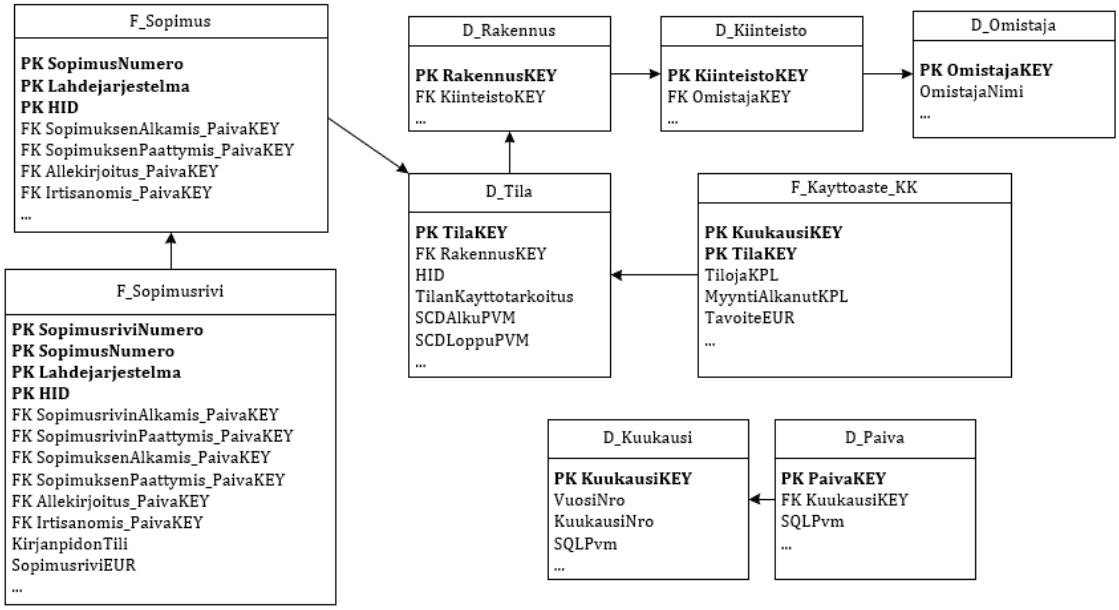
Mittarin muodostamisen tarvitaan sopimuksiin liittyvien asuntojen perustiedot. Jotta koko kiinteistömassasta pystytään rajaamaan haluttujen kiinteistöjen asunnot, tarvitaan lisäksi tieto siitä, missä rakennuksessa asunto sijaitsee, sekä tieto siitä, mihin kiinteistöön rakennus kuuluu. Koska kiinteistöjä halutaan rajata omistajittain, tarvitaan lisäksi tieto siitä, kuka kiinteistön omistaa.

Itse käyttöasteen muodostamista varten tarvitaan jo aiemmin mainitut sopimuksen perustiedot ja -hintatiedot sekä tieto kuukauden tavoitteista tiloittain tarkasteluajanjakson ajalta. Sopimuksen perustiedoista oleellisia ovat tieto siitä, mitä asuntoa sopimus koskee, sopimuksen alkamis- ja loppumisajankohdat, sekä sopimuksen allekirjoitus- ja päättämisajankohdat.

Aiemmat käyttöasteraportit käyttävät tietolähteenään YH Kotien tietovarastoa, joten tämänkin mittarin osalta on luonnollista käyttää samaa tietolähdettä, jotta raporttien tiedoista saadaan mahdollisimman yhdenmukaisia. Tietovaraston valintaa puoltaa myös se, että tietovaraston tiedot ovat muokattu myös valmiiksi helpommin raportoitavaan muotoon ja perustiedot on versioitu, jolloin raportoinnissa pystyään ottamaan huomioon historia perustietojen osalta. Tietosisällön osalta tietovarasto piti sisällään kaikki vaaditut tiedot. Tosin sopimuksen allekirjoituspäivämäärän osalta kävi ilmi, että tietovaraston kentässä käytettiin arvona sopimuksen alkamispäivämäärä, koska allekirjoituspäivämäärää ei ollut aiemmin saatavilla alkuperäisessä tietolähteessä. Tämä puute saatiin kuitenkin korjattua helposti vaihtamalla kyseisen kentän lähde oikeaan alkuperäisen tietolähteen kenttään.

Kuvassa 6 on esitelty tietovaraston rakenne mittariin tarvittavien tietojen osalta. Kuvassa näkyvä D_Tila-dimensiotaulu pitää sisällään mm. raportoinnissa tarvittavat asunnot. Taulu on historioitu dimensiotaulu, eli on voimassa aikavälillä SCDAlkuPVM ja SCDLoppuPVM. Itse asiassa myös muut kiinteistöihin liittyvät dimensiotaulut ovat historioituja, mutta koska ylempiin dimensioihin viittaavat avaimet viittaavat kyseisen dimension tiettyyn versioon, asiaa ei tarvitse ottaa

raportoinnissa huomioon. Faktataulut sen sijaan eivät viittaa suoraan tilan tiettyyn versioon F_Kayttoaste_KK-faktataulua lukuun ottamatta, koska faktarivien voimassaolo voi kattaa useammankin ylemmän objektin version. Sopimuksen hinnat täytyy kuitenkin saada raportointia varten viittaamaan yksiselitteisesti tiettyyn tilan versioon. F_Kayttoaste_KK-faktataulua käyttävä aiempi käyttöastemittari käyttää aina sitä tilan versiota, joka on voimassa kuukauden ensimmäisenä päivänä, joten myös muiden faktojen osalta toimitaan samoin, jotta raporteista saadaan mahdollisimman yhdenmukaiset aiempiin raportteihin verrattuna.



Kuva 6. Tietovaraston rakenne mittariin tarvittavien tietojen osalta

Tietojen lataus suoritetaan tietovarastosta muodostamalla T-SQL-kielellä kyselyt, joilla saadaan haettua raporttia varten tarvittavat tiedot ja rajattua tietojoukko sopivaksi raportointia varten. Seuraavaksi kuvataan tietojen tuonti sekä tilan tavoitteiden että sopimuksen toteumaennusteen osalta. Tavoitteella tarkoitetaan sitä euromääräistä summaa, joka olisi saatu, mikäli tila olisi ollut vuokrattuna koko kuukauden. Toteuma tarkoittaa toteutunutta euromääräistä tuottoa.

Tilojen kuukausitavoitteet muodostetaan jo tälläkin hetkellä tilan hinnaston ja sopimusten perusteella tulevaisuuteen, joten toteumien osalta voidaan käyttää

suoraan valmiiksi muodostettuja kuukausitavoitteita. Kuvassa 7 on kuvattu toteumien tuontiin käytetty ohjelmakoodi. Ainoa uusi tieto, joka täytyy muodostaa, on tavoitteen yksilöivä avain TavoiteKEY. Avain muodostetaan yhdistämällä tavoitteen KuukausiKEY ja TilaKEY. Avain täytyy muodostaa, jotta Power BI:n puolelle saadaan muodostettua yksilöivä avain, jolla saadaan kohdistettua toteumat tiettyyn tavoiteriviin. Lisäksi tuotavat tiedot rajataan koskemaan ainoastaan raportin kannalta oleellisia tietoja.

```
DECLARE @LoppuPvm date
SET @LoppuPvm = EOMONTH(GETDATE(), 3)

SELECT
    CAST(ka.KuukausiKEY AS varchar) + CAST(ka.TilaKEY AS varchar) TavoiteKEY,
    ka.KuukausiKEY,
    ka.TilaKEY,
    ka.TavoiteEUR
FROM F_Kayttoaste_KK ka
INNER JOIN D_Tila ti
ON ka.TilaKEY = ti.TilaKEY
WHERE ka.KuukausiKEY BETWEEN
    YEAR(GETDATE()) * 100 + MONTH(GETDATE()) AND
    YEAR(@LoppuPvm) * 100 + MONTH(@LoppuPvm)
AND ka.TilojaKPL = 1
AND ka.MyyntiAlkanutKPL = 0
AND ti.TilanKayttotarkoitus = 'Asuinhuoneisto'
AND ti.PintaAlaM2 IS NOT NULL
AND ti.PintaAlaM2 > 0
ORDER BY 3, 2
```

Kuva 7. Kuukausikohtaisen tavoitteen haku tiloille

Koska sopimukselle muodostetut laskut eivät ole käytettävissä tulevaisuuden osalta ja täten nykyiseen käyttöasteen faktatauluun ei muodostu toteumaa tulevaisuuden osalta, täytyy toteumaennusteet muodostaa sopimusten hinnastorivien kautta. Toinen asioita mutkistava seikka on se, että käyttöasteen kehitystä halutaan tarkastella edellisenä 28 päivänä tehtyjen irtisanomisten ja allekirjoitettujen sopimusten suhteen, jolloin tarkastelupäivästä riippuen täytyy muodostaa toteumia myös päättyneille sopimuksille päättymispäivän jälkeiselle ajalle. Toteumaennusteen muodostamiseen käytetty koodi on edellä mainituista syistä johtuen hieman monimutkaisempi, joten pilkoin tarkasteltavan koodin useampaan kokonaisuuteen, joita tarkastellaan alla erikseen.

Kuvassa 8 on kuvattu koodi, jolla yhdistellään tarvittavat taulut ja tehdään tarvittavat rajaukset. Aluksi haetaan D_Kuukausi-taulusta kaikki tarkasteluaikavälin

kuukaudet. Jokaiselle kuukaudella tarvitaan lisäksi alkamis- ja loppumispäivät, jotka haetaan D_Paiva-taulun kautta aliaksille pva ja pvl. Tilan osalta haetaan jokaisen tarkastelukuukauden ensimmäisenä päivänä voimassa oleva tila vastaavasti kuin tavoitteiden osalta. Sopimukset ja sopimusrivit (F_Sopimusrivi-taulu) rajataan koskemaan tarkastelukuukautena voimassa olleita sopimusrivejä ja sopimuksia. Itse F_Sopimus-taulua ei tarvita, koska kaikki tarvittavat tiedot löytyvät myös F_Sopimusrivi-taulusta. Lisäksi rajataan kirjanpidon tilien avulla tarkastelu koskemaan ainoastaan vuokra- ja asumisoikeustuottoja koskevia kirjanpidon tilejä.

```
DECLARE @Alku_KuukausiKEY int, @Loppu_KuukausiKEY int, @LoppuPvm date

SET @Alku_KuukausiKEY = YEAR(GETDATE()) * 100 + MONTH(GETDATE());
SET @LoppuPvm = EOMONTH(GETDATE(), 3)
SELECT @Loppu_KuukausiKEY = YEAR(@LoppuPvm) * 100 + MONTH(@LoppuPvm);
```

```
FROM D_Kuukausi kk
INNER JOIN D_Paiva pva
ON kk.KuukausiKEY * 100 + 1 = pva.PaivaKEY
INNER JOIN D_Paiva pvl
ON pvl.SQLPvm = EOMONTH(pva.SQLPvm, 0)
INNER JOIN D_Tila ti
ON pva.PaivaKEY BETWEEN ti.SCDAlkuPVM AND ti.SCDLoppuPVM
INNER JOIN F_Sopimusrivi sr
ON ti.HID = sr.HID
AND sr.SopimusrivinAlkamis_PaivaKEY <= kk.KuukausiKEY * 100 + 31
AND (
    sr.SopimusrivinPaattymis_PaivaKEY < 1 OR
    sr.SopimusrivinPaattymis_PaivaKEY >= kk.KuukausiKEY * 100 + 1
)
AND sr.SopimuksenAlkamis_PaivaKEY <= kk.KuukausiKEY * 100 + 31
WHERE kk.KuukausiKEY BETWEEN @Alku_KuukausiKEY AND @Loppu_KuukausiKEY
AND sr.Lahdejarjestelma = 'Tampuuri'
AND sr.SopimuksenAlkamis_PaivaKEY <= @Loppu_KuukausiKEY * 100 + 31
AND (
    sr.SopimuksenPaattymis_PaivaKEY < 1 OR
    sr.SopimuksenPaattymis_PaivaKEY >= @Alku_KuukausiKEY * 100 + 1
)
AND sr.KirjanpidonTili IN (3320, 3400, 3410)
AND sr.SopimusriviEUR > 0
```

Kuva 8. Toteumaennusteen taulujen yhdistely ja tietojen rajaukset

Kuvassa 9 on ohjelmakoodi, jolla muodostetaan varsinaiset toteumaennusterivit kuvan 8 ohjelmakoodissa yhdisteltyjen ja rajattujen tietojen pohjalta. Jotta toteumaennusterivi saadaan linkitettyä oikeaan tavoiterivin, toteumaennusteriville muodostetaan aluksi TavoiteKEY vastaavasti kuin tavoitteen puolella. Sopimuksen allekirjoituspäivä (Allekirjoitus_PaivaKEY) otetaan mukaan, jotta to-

teumaennusteen rivejä saadaan rajattua tarkastelupäivän mukaan. Itse toteumaennusteen osalta täytyy ottaa huomioon kesken kuuta alkaneet tai päättyneet sopimukset. Tätä varten täytyy laskea, kuinka monena päivänä sopimus on ollut voimassa tarkastelukuukauden aikana. Euromääräinen toteumaennuste saadaan muodostettua jakamalla sopimuksen kuukauden voimassaolopäivät kuukauden päivillä ja kertomalla luku koko kuukauden sopimusrivin hinnalla.

```
SELECT
    CAST(kk.KuukausiKEY AS varchar) + CAST(ti.TilaKEY AS varchar) TavoiteKEY,
    kk.KuukausiKEY, ti.TilaKEY, sr.SopimusNumero,
    sr.SopimuksenAlkamis_PaivaKEY, sr.SopimuksenPaattymis_PaivaKEY,
    sr.Allekirjoitus_PaivaKEY, 0 Irtisanomis_PaivaKEY,
    (1.0 + CASE
        WHEN
            SopimuksenPaattymis_PaivaKEY BETWEEN pva.PaivaKEY AND pvl.PaivaKEY AND
            (
                SopimuksenPaattymis_PaivaKEY <= SopimusrivinPaattymis_PaivaKEY OR
                SopimusrivinPaattymis_PaivaKEY = 0
            )
        THEN SopimuksenPaattymis_PaivaKEY
        WHEN
            SopimusrivinPaattymis_PaivaKEY BETWEEN pva.PaivaKEY AND pvl.PaivaKEY AND
            (
                SopimusrivinPaattymis_PaivaKEY < SopimuksenPaattymis_PaivaKEY OR
                SopimuksenPaattymis_PaivaKEY = 0
            )
        THEN SopimusrivinPaattymis_PaivaKEY
        ELSE pvl.PaivaKEY
    END -
    CASE
        WHEN
            SopimuksenAlkamis_PaivaKEY BETWEEN pva.PaivaKEY AND pvl.PaivaKEY AND
            SopimuksenAlkamis_PaivaKEY >= SopimusrivinAlkamis_PaivaKEY
        THEN SopimuksenAlkamis_PaivaKEY
        WHEN
            SopimusrivinAlkamis_PaivaKEY BETWEEN pva.PaivaKEY AND pvl.PaivaKEY AND
            SopimusrivinAlkamis_PaivaKEY > SopimuksenAlkamis_PaivaKEY
        THEN SopimusrivinAlkamis_PaivaKEY
        ELSE pva.PaivaKEY
    END)/pvl.KuukaudenPaivaNro * SopimusriviEUR ToteumaEnnusteEUR
```

Kuva 9. Sopimuksen toteumaennusteen laskeminen

Aiemmin muodostettu toteumaennuste ottaa huomioon sopimukset sopimuksen päättymispäivämäärään asti, mutta käyttöaste-ennustehistorian laskentaa varten tarvitaan toteumaennuste myös sopimuksen päättymisen jälkeiselle ajalle, mikäli sopimus päättyy tarkasteluajanjaksona ja sopimuksen päättäminen on tehty tarkasteluajanjakson aikana. Tätä varten muodostetaan erikseen toteumaennusteet edellä mainitut kriteerit täyttäville sopimuksille (kuva 10). Ennusteen muodostaminen tehdään pitkälti samoin kuin aiemman toteumaennus-

teen muodostaminen. Oleellisimmat erot aiempaan koodiin verrattuna ovat seuraavat: Sopimusriveille tuodaan irtisanomispäivä, jolla saadaan rajattua toteumia tarkastelupäivän mukaan. Toteuma lasketaan aina sopimuksen päättymispäivän mukaisen hinnan mukaan. Toteuma lasketaan koko käsitellylle aikajaksolle huolimatta siitä, että sopimus päättyy ennen aikajakson loppua. Mukaan otetaan ainoastaan sellaiset sopimukset, jotka päättyvät käsitellyllä aikajaksolla ja joiden päättäminen on tapahtunut tarkasteluaikavälillä.

```
DECLARE @TarkasteluAlku_PaivaKEY int, @TarkasteluLoppu_PaivaKEY int
SELECT @TarkasteluAlku_PaivaKEY = PaivaKEY FROM D_Paiva
WHERE SQLPvm = CAST(DATEADD(day, -29, GETDATE()) AS date)
SELECT @TarkasteluLoppu_PaivaKEY = PaivaKEY FROM D_Paiva
WHERE SQLPvm = CAST(DATEADD(day, -1, GETDATE()) AS date)

UNION ALL
SELECT
    CAST(kk.KuukausiKEY AS varchar) + CAST(ti.TilaKEY AS varchar) TavoiteKEY,
    kk.KuukausiKEY, ti.TilaKEY, sr.SopimusNumero,
    sr.SopimuksenAlkamis_PaivaKEY, sr.SopimuksenPaattymis_PaivaKEY,
    sr.Allekirjoitus_PaivaKEY, sr.Irtisanomis_PaivaKEY,
    (0.0 + CASE
        WHEN
            SopimuksenPaattymis_PaivaKEY BETWEEN pva.PaivaKEY AND pvl.PaivaKEY AND
            SopimuksenPaattymis_PaivaKEY != pvl.PaivaKEY
        THEN pvl.PaivaKEY - SopimuksenPaattymis_PaivaKEY
        ELSE pvl.KuukaudenPaivaNro
    END)/pvl.KuukaudenPaivaNro * SopimusriviEUR ToteumaEnnusteEUR
FROM D_Kuukausi kk
INNER JOIN D_Paiva pva
ON kk.KuukausiKEY * 100 + 1 = pva.PaivaKEY
INNER JOIN D_Paiva pvl
ON pvl.SQLPvm = EOMONTH(pva.SQLPvm, 0)
INNER JOIN D_Tila ti
ON pva.PaivaKEY BETWEEN ti.SCDAlkuPVM AND ti.SCDLoppuPVM
INNER JOIN F_Sopimusrivi sr
ON ti.HID = sr.HID
AND sr.SopimuksenPaattymis_PaivaKEY BETWEEN
    sr.SopimusrivinAlkamis_PaivaKEY AND sr.SopimusrivinPaattymis_PaivaKEY
AND pvl.PaivaKEY >= sr.SopimuksenPaattymis_PaivaKEY
WHERE kk.KuukausiKEY BETWEEN @Alku_KuukausiKEY AND @Loppu_KuukausiKEY
AND sr.Irtisanomis_PaivaKEY BETWEEN
    @TarkasteluAlku_PaivaKEY AND @TarkasteluLoppu_PaivaKEY
AND sr.SopimuksenPaattymis_PaivaKEY BETWEEN
    @Alku_KuukausiKEY * 100 + 1 AND @Loppu_KuukausiKEY * 100 + 31
AND sr.SopimuksenPaattymis_PaivaKEY != pvl.PaivaKEY
AND sr.Lahdejarjestelma = 'Tampuuri'
AND sr.KirjanpidonTili IN (3320, 3400, 3410)
AND sr.SopimusriviEUR > 0
```

Kuva 10. Päättyvien sopimusten toteumaennusteen laskeminen

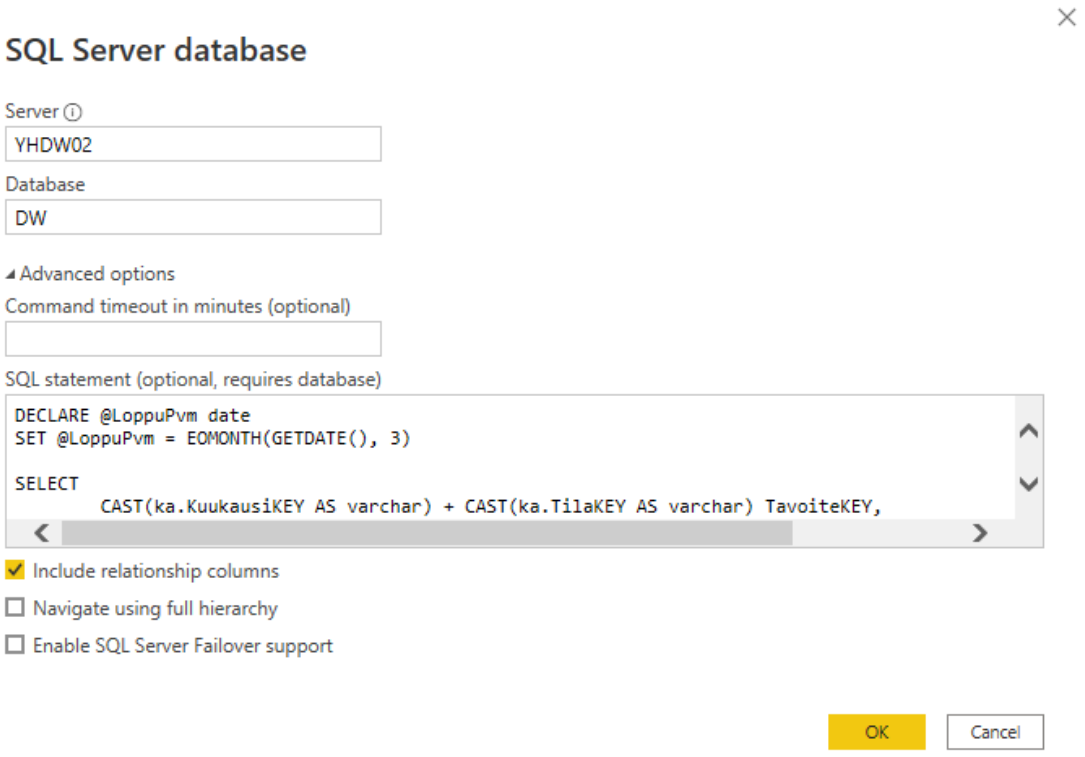
Tiedoissa on monesti viitattu PaivaKEY- ja KuukausiKEY-avaimiin, jotka ovat D_Paiva- ja D_Kuukausi-taulujen ensisijaiset avaimet. Kyseisten taulujen tietoja tarvitaan myös Power BI -raportoinnin puolella, joten taulujen tiedot haetaan mukaan käsitellyn päivämäärävälin mukaan rajattuina kuvan 10 ohjelmakoodin

mukaisesti. Muut tarvittavat taulut, joita ovat D_Omistaja, D_Kiinteisto, D_Rakennus ja D_Tila, tuodaan Power BI:n puolelle sellaisenaan.

```
SELECT *
FROM D_Paiva
WHERE SQLPvm BETWEEN
    DATEADD(day, 1, EOMONTH(GETDATE(), -1)) AND
    EOMONTH(GETDATE(), 3)

SELECT *
FROM D_Kuukausi kk
WHERE KuukausiKEY IN (
    SELECT KuukausiKEY
    FROM D_Paiva
    WHERE SQLPvm BETWEEN
        DATEADD(day, 1, EOMONTH(GETDATE(), -1)) AND
        EOMONTH(GETDATE(), 3)
    AND KuukaudenPaivaNro = 1
)
```

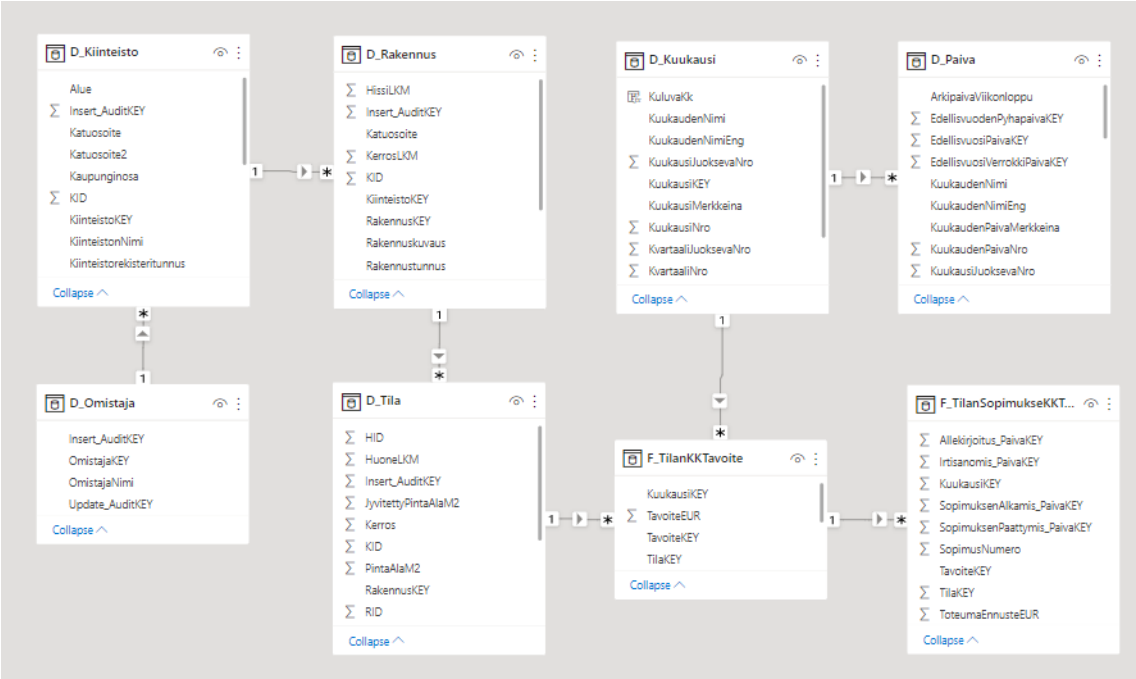
Kuva 11. Päivä- ja kuukausitaulujen tietojen tuonti

Tiedot tuodaan Power BI:n puolelle käyttäen SQL Server -yhdistintä. Kokonaan tuotavat taulut saadaan tuotua kerralla valitsemalla yhdistimen asetuksista kaikki halutut taulut. Niiden tietojen osalta, joiden tuontiin käytetään T-SQL-kielellä luotua ohjelmakoodia, tarvitaan tuonnin lisäasetuksia kuvan 12 mukaisesti. SQL lauseke -osaan (SQL statement) kirjoitetaan suoraan tuontiin käytettävä T-SQL-ohjelmakoodi. Koska tietojen muokkaus on tehty pääosin tietojen tuonnin yhteydessä, Power BI:n puolella ei ole enää tarvetta muokata tuotuja tietoja Power Query -editorilla, joka on tarkoitettu nimenomaan tuotujen tietojen muokkaukseen.



Kuva 12. Tietojen tuonti SQL-lausekkeella

Kun tiedot on saatu tuotua, muodostetaan vielä tietomalli, jossa linkitetään taulut toisiinsa avainarvojen perusteella. Kuvassa 13 on kuvattu raportin tietomalli tässä raportin muodostamisen vaiheessa käytettyjen tietojen osalta. Toisin kuin SQL Server -palvelussa, Power BI:ssä taulujen yhdistämiseen voi käyttää ainoastaan yhtä avainta. F_TilanSopimuksenKKToteumaEnnuste-taulun osalta käytetään tietojen tuonnin yhteydessä muodostettua TavoiteKEY-avainta. Muiden tietojen osalta tarvittavat avaimet olivat olemassa jo valmiiksi, joten yhdistämiseen voidaan käyttää kyseisiä avaimia. Kaikkien tämän raportin tietomallin taulujen kardinaliteetti on 1 - *, eli jokaiselle toisen taulun vierasavaimelle löytyy ainoastaan yksi arvo ensimmäisen taulun puolelta. Taulujen väliset suodattimet jätettiin vakioasetuksiin, eli 1-puolen taulu suodattaa aina *-puolen taulua.



Kuva 13. Raportin tietomalli

4.1.2 Mittarien muodostaminen

Käyttöasteella tarkoitetaan tässä yhteydessä taloudellista käyttöastetta eli toteuman prosentuaalista osuutta tavoitteesta. Kuvassa 14 on kuvattu DAX-kielinen koodi, jolla suoritetaan käyttöasteen laskenta. Toteuman osalta käytetään kaikkia tavoitteeseen kohdistuvien toteumarivien summia. Tavoitteeseen kohdistuvat toteumarivit saadaan haettua tietomallin mukaisesti tavoiteriviä SUMX-funktiolla iteroidessa DAX-kielen RELATEDTABLE-funktiolla, joka hakee kaikki kyseiseen riviin kohdistuvat toteumarivit. Koska rivejä voi olla useita, täytyy rivit summata SUMX-funktiolla. Mikäli tavoitteeseen ei kohdistu yhtään toteumaa, käytetään toteuman arvona nollaa, jotta tavoiteriville saadaan muodostettua myös tässä tapauksessa käyttöaste. Tavoitteen osalta käytetään tavoitteeseen kohdistuvien toteumien summaa, mikäli tämä on suurempi kuin tavoitteen arvo, ja muussa tapauksessa tavoitteen arvoa. Tätä ratkaisua käytetään, koska vuokrattujen asumisoikeusasuntojen osalta toteuma muodostuu suuremmaksi, kuin tilan tavoite. Näin tapahtuu, koska asumisoikeusasuntojen tavoite muodostetaan oletuksena asumisoikeushinnaston vuokrahinnastoon nähden alemman hinnan mukaan.

```
Kayttoaste =
    DIVIDE(
        COALESCE(
            SUM(F_TilanSopimuksenKKToteumaEnnuste[ToteumaEnnusteEUR]),
            0
        ),
        SUMX(
            F_TilanKKTavoite,
            MAX(
                SUMX(
                    RELATEDTABLE(F_TilanSopimuksenKKToteumaEnnuste),
                    F_TilanSopimuksenKKToteumaEnnuste[ToteumaEnnusteEUR]
                ),
                F_TilanKKTavoite[TavoiteEUR]
            )
        )
    )
```

Kuva 14. Käyttöasteen laskentaan käytetty DAX-kielinen lauseke

Käyttöastemittarin arvoa tarkastellaan edellisten 28 päivän päivien suhteen siten, että jokaisena tarkasteluajanjakson päivänä otetaan huomioon kyseisenä päivänä tai tätä ennen allekirjoitetut tai irtisanotut sopimukset. Mittarista jätetään pois kaikki tarkastelupäivän jälkeen alkaneet sopimukset ja vastaavasti irtisanomisista huomioidaan ainoastaan ne, jotka ovat tapahtuneet tarkastelupäivänä tai tätä ennen. Käyttöaste lasketaan aina kohdekuukauden viimeiselle päivälle.

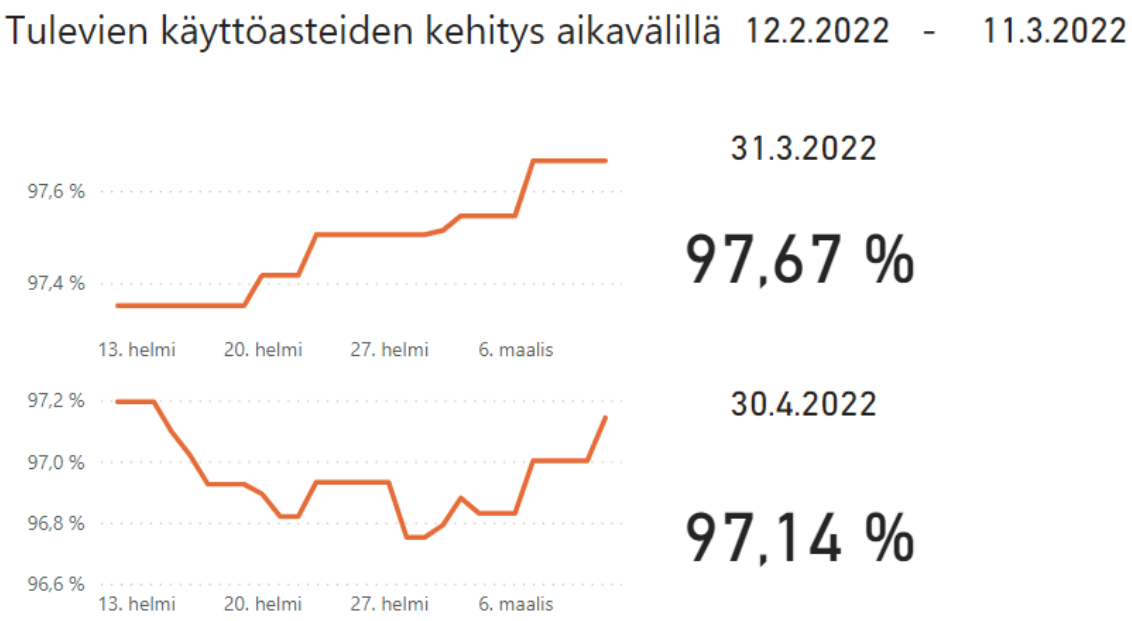
Jotta aiempi käyttöastemittari saadaan toimimaan näiden rajausten kanssa, tehdään uusi mittari kuvan 15 mukaisella DAX-koodilla. Tarkastelu_PaivaKEY-muuttujasta on hyvä huomata se, että muuttaja poimii senhetkisen suodatinkontekstin mukaisen viimeisimmän päivän avaimen, jolloin tarkasteluajanjaksolle voidaan muodostaa helposti kuvaaja päivämäärien suhteen valitsemalla kuvaajan akseliksi D_Päivä-taulun päivämäärät. Kohdekuukausi taas määritellään mittarin suodatinkontekstiin kiinteästi käyttämällä apuna mittaria KK1ViimPv, joka laskee kuluvan kuukauden viimeisen päivän. Lisäksi mittarin suodattimiin lisätään ehdot, jolla saadaan rajattua toteumista pois ne tapahtumat, jotka ovat tapahtuneet tulevaisuudessa suhteessa senhetkiseen tarkastelupäivään.

```
Kk1Kayttoaste =
VAR Tarkastelu_PaivaKEY = MAX(D_Paiva[PaivaKEY])

VAR KohdeKk_KuukausiKEY = YEAR([KK1ViimPv]) * 100 + MONTH([KK1ViimPv])
VAR KohdeKkViim_PaivaKEY = KohdeKk_KuukausiKEY * 100 + DAY([KK1ViimPv])

RETURN CALCULATE(
    [Kayttoaste],
    F_TilanKKTavoite[KuukausiKEY] = KohdeKk_KuukausiKEY,
    AND (
        F_TilanSopimuksenKKToteumaEnnuste[Allekirjoitus_PaivaKEY] <= Tarkastelu_PaivaKEY,
        (
            F_TilanSopimuksenKKToteumaEnnuste[SopimuksenPaattymis_PaivaKEY] >= KohdeKkViim_PaivaKEY ||
            F_TilanSopimuksenKKToteumaEnnuste[Irtisanomis_PaivaKEY] > Tarkastelu_PaivaKEY ||
            F_TilanSopimuksenKKToteumaEnnuste[Irtisanomis_PaivaKEY] = 0
        )
    )
)
```

Kuva 15. Kuluvan kuukauden käyttöaste tarkastelupäivälle

Koska käyttöaste-ennustetta halutaan näyttää myös muutaman seuraavan kuukauden osalta, muodostetaan omat mittarit myös seuraaville kuukausille. Mittari muodostetaan lähes samoin kuin kuluvan kuukauden mittari. Ainoana erona on kohdekuukauden viimeisimmän päivän viittausten vaihto osoittamaan seuraavan kuukauden viimeistä päivää.

Näiden mittarien avulla saadaan muodostettua kuvan 16 mukaiset visualisoinnit. Kuvassa näytetään ainoastaan osa muodostetuista mittareista, mutta muiden kuukausien mittarit ovat vastaavia kuin kuvassa esitetyt mittarit. Kuluvalle ja kolmelle seuraavalle kuukaudelle muodostetaan käyttöastegraafi, joka kuvaa käyttöasteen kehitystä edellisen 28 päivän päivien suhteen. X-akselilla on kuluvan kuukauden päivät ja y-akselilla käyttöasteprosentti. Lisäksi näytetään korttimuotoisena visualisointina kunkin kuukauden lopun käyttöasteprosentin ennuste. Kortin yläpuolelle tuodaan selvyyden vuoksi päivämäärä, jolle käyttöastelukema on muodostettu. Otsikkoon tuodaan varsinaisen otsikon lisäksi tarkasteluajanjakson päivämääräväli.



Kuva 16. Tulevan käyttöasteen visualisointi

4.2 Alkavien sopimusten ja vapautuvien tilojen mittari

Käyttöaste on suoraan sidoksissa alkavien sopimusten ja vapautuvien tilojen määrään. Alkavien sopimusten ja vapautuvien tilojen mittarin antaa kuitenkin uutta tietoa tilanteen tarkasteluun pelkkään käyttöastemittariin verrattuna. Tästä syystä onkin luonnollista liittää myös tämä mittari osaksi samaa raporttia, jolla tarkastellaan käyttöastetta.

Laskemalla edellisen 28 päivän osalta kumulatiivinen päiväkohtainen summa sekä alkavista sopimuksista että vapautuvista tiloista nähdään helposti, ollaanko niin sanotusti pinnalla mittarin kuukauden osalta. Mikäli alkavia sopimuksia on enemmän kuin vapautuvia tiloja, käyttöaste todennäköisesti kasvaa. Kappalemääräinen mittari tuo käyttöastemittariin verrattuna mukaan myös huomattavasti enemmän konkretiaa, koska mittarista voidaan seurata suoraan niiden uusien sopimusten määrää, jotka on esimerkiksi juuri edellisenä päivänä allekirjoitettu. Vastaavasti voidaan tarkastella vapautuvien tilojen määrää niiden sopimusten osalta, joiden irtisanomiset on käsitelty jonain tarkasteluajanjakson päivänä.

Mittarin luvut saadaan laskettua samaa tietosisältöä käyttäen kuin käyttöaste-mittarin tiedot, eli tiedot ovat haettavissa toimeksiantajayrityksen tietovarastosta. Toisin kuin nykyisin käytössä olevassa kuukausittaisessa mittarissa, uudessa mittarissa valittiin sopimusten päättymispäivän sijaan tarkasteltavaksi muuttujaksi päättymispäivää seuraava päivä, jolloin tila pääsääntöisesti vapautuu ja on vuokrattavissa. Koska etenkin vuokrasopimukset päättyvät useimmiten kuun viimeisenä päivänä, saadaan tällä valinnalla alkavien ja päättyvien sopimusten luvut vertailukelpoisiksi kuukausitasolla.

Toinen seikka, joka täytyy ottaa huomioon alkavien sopimusten ja vapautuvien tilojen lukumäärän vertailussa, on uudiskohteet, joissa ei ole ollut yhtään aiempaa sopimusta. Näiden tilojen alkavat sopimukset on hyvä saada eriteltyä muista sopimuksista, jotta mittarin lukujen vertailukelpoisuus säilyy. Myös tämä tieto pystytään päättelemään melko vaivattomasti nykyisen tietovaraston tietojen perusteella.

4.2.1 Tietolähteet, tietojen haku ja -muokkaus

Sopimusmäärien mittaria varten tietomalliin tarvitaan taulu, jonka kautta saadaan laskettua kappalemäärät allekirjoitetuista ja irtisanotuista sopimuksista. Tiedot löytyvät toteumataulusta, mutta toteumataulussa sopimuksilla voi olla tuplarivejä, joten helpoimmin laskenta saadaan toteutettua tuomalla uusi sopimustaulu tietomalliin. Kuvassa 17 on esitelty tietojen tuontiin käytetty T-SQL-koodi. Tuonti tapahtuu melko suoraviivaisesti sopimukset sisältävän F_Sopimus-taulun tietojen avulla. Tilojen valintaan käytetään F_Kayttoaste_KK-faktataulua vastaavasti kuin tavoitetaulun tuonnin osalta. Tuotavat sopimukset rajataan kohdekuukausivälillä voimassa oleviin kuukausiin sopimusten alkamis- ja päättymispäivämäärien avulla. Ainoa uusi tieto, jota tietovarastosta ei löydy valmiiksi, on tieto siitä, onko kyseessä tilan ensimmäinen sopimus vai ei.

```
SELECT
    ka.KuukausiKEY, ti.TilaKEY, s.SopimusNumero,
    CASE WHEN es.HID IS NOT NULL THEN 1 ELSE 0 END AS EnsimmainenSopimus,
    s.SopimuksenAlkamis_PaivaKEY,
    s.SopimuksenPaattymis_PaivaKEY,
    s.Allekirjoitus_PaivaKEY,
    s.Irtisanomis_PaivaKEY
FROM F_Kayttoaste_KK ka
INNER JOIN D_Tila ti
ON ka.TilaKEY = ti.TilaKEY
INNER JOIN F_Sopimus s
ON ti.HID = s.HID
AND s.SopimuksenAlkamis_PaivaKEY <= ka.KuukausiKEY * 100 + 31
AND (
    s.SopimuksenPaattymis_PaivaKEY < 1 OR
    s.SopimuksenPaattymis_PaivaKEY >= ka.KuukausiKEY * 100 + 1
)
LEFT JOIN TilanEnsimmainenSopimus es
ON s.HID = es.HID
AND s.SopimusNumero = es.SopimusNumero
AND s.Lahdejarjestelma = es.Lahdejarjestelma
WHERE ka.KuukausiKEY BETWEEN
    YEAR(GETDATE()) * 100 + MONTH(GETDATE()) AND
    YEAR(@LoppuPvm) * 100 + MONTH(@LoppuPvm)
AND ka.TilojaKPL = 1
AND ka.MyyntiAlkanutKPL = 0
AND ti.TilanKayttotarkoitus = 'Asuinhuoneisto'
AND ti.PintaAlaM2 IS NOT NULL
AND ti.PintaAlaM2 > 0
AND s.Lahdejarjestelma = 'Tampuuri'
```

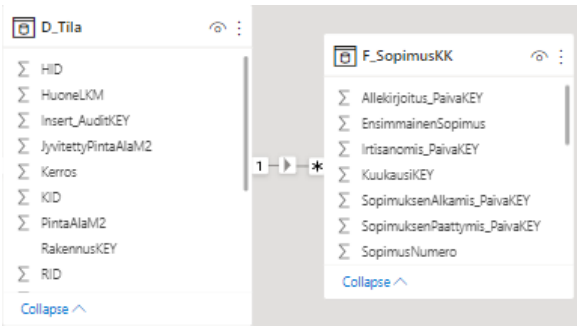
Kuva 17. Sopimustietojen tuonnin T-SQL-koodi

Tilan ensimmäisen sopimuksen laskenta hoituu yksinkertaisesti järjestämällä tilan sopimukset alkamispäivämäärän mukaiseen järjestykseen ja valitsemalla ensimmäinen sopimus kunkin tilan kohdalta. Kuvassa 18 on kuvattu tähän käytetty koodi. Sopimukset haetaan tiloittain järjestettynä ja numeroituna sisemmässä kyselyssä ja ulommassa kyselyssä valitaan ainoastaan kunkin tilan osalta ensimmäinen rivi, eli vanhin sopimus.

```
WITH TilanEnsimmainenSopimus AS (
    SELECT s.HID, s.SopimusNumero, s.Lahdejarjestelma FROM (
        SELECT
            s.HID, s.SopimusNumero, s.Lahdejarjestelma, ROW_NUMBER()
            OVER (PARTITION BY s.HID ORDER BY s.SopimuksenAlkamis_PaivaKEY) rivi
        FROM F_Sopimus s
        WHERE s.SopimuksenPaattymis_PaivaKEY = 0 OR
        s.SopimuksenPaattymis_PaivaKEY - s.SopimuksenAlkamis_PaivaKEY > 1
    ) s
    WHERE rivi = 1
)
```

Kuva 18. Tilan ensimmäisen sopimuksen haku

Haetut tiedot tuodaan tietomalliin F_SopimusKK-nimiseen tauluun, joka liitetään D_Tila-tauluun TilaKEY-avaimella. Jokaisen tilaan voi kohdistua useampi sopimus, joten yhteyden kardinaliteetti on 1 - *. Myöskään suodatusta ei ole tässä yhteydessä tarvetta muuttaa oletusarvoista, joten suodatus tapahtuu D_Tila-taulun suunnasta F_SopimusKK-tauluun. Kuvassa 19 on esitelty tietomalliin lisätty taulu ja D_Tila-taulu, johon uusi taulu liittyy.



Kuva 19. F_SopimusKK-taulu tietomallissa

4.2.2 Mittarien muodostaminen

Sopimusmääriä varten muodostetaan kolme erillistä mittaria. Näistä ensimmäinen on kohdekuukautena alkavien sopimusten mittari. Mittari näyttää kohdekuukautena alkavien sopimusten kumulatiivista määrää siten, että lukemaan lasketaan mukaan ainoastaan tarkastelupäivänä tai tätä ennen allekirjoitetut sopimukset. Lisäksi suodatetaan pois tilojen ensimmäiset sopimukset. Koska tässä tapauksessa lasketaan sopimusten lukumäärää, eikä summata yhteen euromääriä, käytetään mittarin muodostamiseen DAX-kielen COUNT-funktiota, kuten kuvan 20 koodista nähdään. Mikäli tarkastelupäivälle ei tulisi yhtään sopimusta, mittari palauttaisi tyhjän arvon, ellei mittarin lopussa rivillä 17 olisi summattu mittarin arvoa lukuun 0. Saman saisi aikaan myös käyttämällä esimerkiksi IF-lauseketta, mutta tässä käytetty tekniikka on lyhyempi. Uusien alkavien sopimusten mittarin on muuten täysin sama kuin alkavien sopimusten mittari, mutta rivillä 15 valitaan mukaan tarkasteluun ainoastaan tilojen ensimmäiset sopimukset.

```
Kk1Alkavat =
VAR Tarkastelu_PaivaKEY = MAX(D_Paiva[PaivaKEY])

VAR KohdeKk_KuukausiKEY = YEAR([KK1ViimPv]) * 100 + MONTH([KK1ViimPv])
VAR KohdeKKEns_PaivaKEY = KohdeKk_KuukausiKEY * 100 + 1
VAR KohdeKkViim_PaivaKEY = KohdeKk_KuukausiKEY * 100 + DAY([KK1ViimPv])

RETURN CALCULATE(
    COUNT(F_SopimusKK[SopimusNumero]),
    F_SopimusKK[KuukausiKEY] = KohdeKk_KuukausiKEY,
    (
        F_SopimusKK[Allekirjoitus_PaivaKEY] <= Tarkastelu_PaivaKEY &&
        F_SopimusKK[SopimuksenAlkamis_PaivaKEY] >= KohdeKKEns_PaivaKEY &&
        F_SopimusKK[SopimuksenAlkamis_PaivaKEY] <= KohdeKkViim_PaivaKEY &&
        F_SopimusKK[EnsimmainenSopimus] = 0
    )
) + 0
```

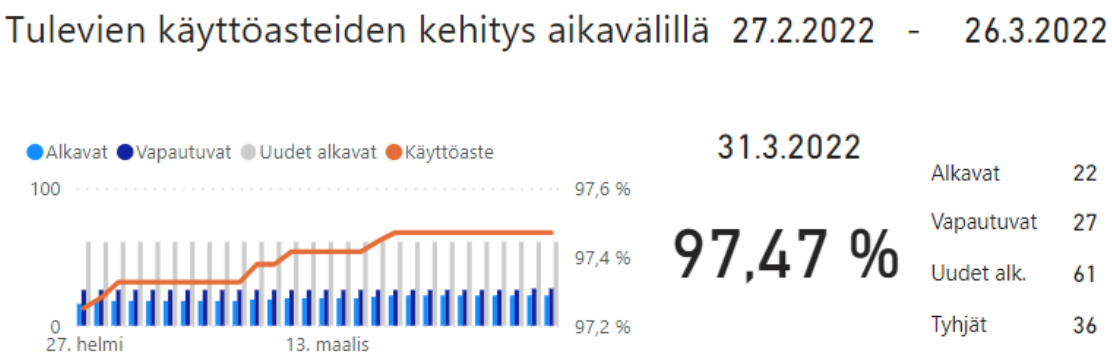
Kuva 20. Alkavien sopimusten mittari

Päättyvien sopimusten mittari muodostetaan lähes samoin kuin alkavien sopimusten mittari. Erona alkavien sopimusten mittariin on laskettavien sopimusten valinta alkamispäivän sijaan päättymispäivän mukaan. Lisäksi tarkastelupäivän mukaan suodatus tehdään allekirjoituspäivän sijaan irtisanomispäivän mukaan. Tämän mittarin osalta ei ole myöskään tarvetta tarkastella erikseen uusia sopimuksia, joten kyseinen suodatin voidaan jättää pois. Kuvassa 21 on ensimmäisen kuukauden mittarin muodostamiseen käytetty koodi. Muiden kuukausien mittareiden osalta ainoana erona on KK1ViimPv-viittauksen vaihtaminen halutun kuukauden viittaukseen.

```
Kk1Vapautuvat =
VAR Tarkastelu_PaivaKEY = MAX(D_Paiva[PaivaKEY])

VAR KohdeKk_KuukausiKEY = YEAR([KK1ViimPv]) * 100 + MONTH([KK1ViimPv])
VAR KohdeKKEns_PaivaKEY = KohdeKk_KuukausiKEY * 100 + 1
VAR KohdeKkViim_PaivaKEY = KohdeKk_KuukausiKEY * 100 + DAY([KK1ViimPv])

RETURN CALCULATE(
    COUNT(F_SopimusKK[SopimusNumero]),
    F_SopimusKK[KuukausiKEY] = KohdeKk_KuukausiKEY,
    (
        F_SopimusKK[Irtisanomis_PaivaKEY] <= Tarkastelu_PaivaKEY &&
        F_SopimusKK[Vapautumis_PaivaKEY] >= KohdeKKEns_PaivaKEY &&
        F_SopimusKK[Vapautumis_PaivaKEY] <= KohdeKkViim_PaivaKEY
    )
) + 0
```

Kuva 21. Päättyvien sopimusten mittarin koodi

Yllä muodostetut mittarit tuodaan samaan visualisointiin aiemmin tehdyn käyttöastemittarin kanssa. Alkavien- ja uusien alkavien sopimusten, sekä vapautuvien tilojen mittarin esitysmuodoksi valittiin ”Line and clustered column chart”, koska graafissa täytyy saada näytettyä rinnakkain useamman mittarin arvot. Lisäksi mittareille tehtiin erilliset korttimuotoiset visualisoinnin vastaavasti, kuin käyttöasteelle. Kuvassa 22 näkyy osa lopputuloksena syntyneistä visualisoinneista. Muiden kuukausien visualisoinnit ovat vastaavat kuin ensimmäisen kuukauden.



Kuva 22. Alkavien, uusien alkavien ja päättyvien sopimusten mittarien visualisoinnit.

4.3 Ohjelmistorobotin toteutus

Osaa raportoinnissa käytetyistä tiedoista ei pysty hakemaan suoraan mistään valmiista tietolähteestä tai minkään valmiin rajapinnan kautta. Näiden tietojen hakuun päätettiin rakentaa ohjelmistobotti, joka hakee tiedot suoraan ohjelmiston käyttöliittymän kautta lähes ihmistä vastaavalla tavalla. Ihmisestä poiketen robotti pystyy kuitenkin käyttämään tiettyjä oikoteitä, kuten avaamaan esimerkiksi suoraan vaadittavan näkymän klikkaamatta toimintoa auki käyttöliittymän kautta ja käyttämään käyttöliittymää huomattavasti nopeammin kuin ihminen. Ohjelmistorobotteja voidaan toteuttaa esimerkiksi kaupallisilla low-code-, eli vähäisen ohjelmakoodin ratkaisuilla tai rakentamalla robotti koodaamalla robotin logiikka yleiskäyttöisellä ohjelmointikielellä.

4.3.1 Käytettävän ohjelmiston valinta

Tässä työssä päädyttiin käyttämään jo aiemmin tutuksi tullutta Microsoftin kehittämää avoimen lähdekoodin Playwright-kirjastoa, jota pystyy käyttämään useammalla eri ohjelmointikielellä. Playwright-kirjasto toimii ohjaamalla halutun selaimen käyttöliittymää ohjelmallisesti. Toinen tapa hakea tietoa selainkäyttöisten ohjelmien käyttöliittymästä on käyttää pelkkiä http-kutsuja, joilla selaimet hakevat tiedot palvelimelta, mutta tämä ei nykyään useimmiten toimi kovin hyvin, koska ohjelmistojen käyttöliittymät on toteutettu suurelta osin käyttäen käyttäjän selaimessa suoritettavaa JavaScript-koodia. Vaikka kirjasto käyttääkin selaimen käyttöliittymää, voidaan robotin toiminnot suorittaa ilman näkyvää selaimen ikkunaa. Tämä mahdollistaa robottien suorituksen esimerkiksi ajastetusti palvelimelta. Playwright ei ole ainoa kirjasto, jolla selaimen käyttöliittymää voidaan käyttää ohjelmallisesti. Toinen pidemmän historian omaava kirjasto, jota käytetään usein vastaaviin käyttötarkoituksiin, on Selenium. Playwrightiin verrattuna Seleniumin käyttäminen vaatii kuitenkin huomattavasti enemmän koodia, koska Seleniumissa joutuu toteuttamaan itse käyttöliittymän elementtien odotukseen liittyvän logiikan, joka taas Playwrightissa hoituu pääasiassa automaattisesti kirjastoon sisäänrakennetulla logiikalla. Kaupallisia ohjelmistorobottiratkaisuita ei lähdetty ottamaan käyttöön tätä insinöörityötä varten ohjelmistojen korkeiden lisensointikustannusten takia ja myös siitä syystä, että näistä ohjelmistoista ei ollut aiempaa kokemusta.

4.3.2 Ohjelmistorobotin rakenne

Ohjelmistorobotin suunnittelun lähtökohtana olivat seuraavat tarpeet. Robotin pitää pystyä kirjautumaan Tampuuriin, hakemaan halutut tiedot ohjelmiston käyttöliittymästä ja tallentamaan halutut tiedot tietokantaan raportointia varten. Lisäksi robotin onnistuneen suorituksen seuraamiseksi tarvittiin tieto siitä, onko kukin robotin suorittama vaihe onnistunut. Virheiden havainnoinnin lisäksi tarvittiin keino välittää tieto epäonnistumisesta eteenpäin oikealle taholle. Robotille oli suunniteltu jo tässä vaiheessa useampi suoritettava tehtäväkokonaisuus,

joissa osa tehtäväkokonaisuuksien vaiheista on samoja, joten suoritettavien vaiheiden tai osatehtävien olisi hyvä olla toistettavissa useassa eri tehtäväkokonaisuudessa ilman, että näihin käytettyä koodia täytyisi toistaa.

Näiden määritysten pohjalta ohjelmistorobotti päätettiin rakentaa siten, että jokainen käyttöliittymässä suoritettu toimintokokonaisuus pilkotaan omaksi tehtäväksi ja tehtäväkokonaisuudet muodostetaan koostamalla näitä tehtäviä peräkkäisiksi vaiheiksi. Tehtäväkokonaisuuksien suorittamiseen tehtiin oma moduuli, joka suorittaa tehtäväkokonaisuuden tehtävät annetussa järjestyksessä, kerää tehtävien suorituksessa mahdollisesti aiheutuneet poikkeamat ja raportoi poikkeamat eteenpäin sähköpostitse. Sähköpostiin liitetään tehtäväkokonaisuuden lokitiedosto, sekä kuva selaimen käyttöliittymästä siinä tilassa, jossa selain oli, kun poikkeus tapahtui. Tietokantaan liittyvä toiminallisuus, kuten tietojen tallennus tietokantaan, sisällytettiin myös tehtäväkokonaisuuksien osatehtäviksi. Ohjelmisto jaettiin seuraaviin moduulikokonaisuuksiin:

- Core-kokonaisuus sisältää ohjelmiston ydintoiminannallisuuteen liittyvät moduulit, kuten asetusten hallinnan, tehtävälistojen suorittamisen toiminnallisuudet, sekä sähköpostivirheraportoinnin moduulin.

- Database-kokonaisuus sisältää tietokantayhteyden hallinnan, tietokantataulujen ORM (Object Relational Mapping) määrittelyt, tietokannan käyttämiseen liittyvän toiminnallisuuden sekä tietojen tietokantaan tallennukseen käytetyt tehtävät.

- TampuuriUI-kokonaisuus sisältää Tampuurin käyttöliittymän käyttöön liittyvän toiminnallisuuden jaoteltuna tehtäväkokonaisuuden mukaisiin moduuleihin.

- Tasklists-kokonaisuus sisältää tehtävälistojen määritykset. Tehtävälistat koostavat osatehtävistä tehtäväkokonaisuuksia, joilla saadaan toteutettua haluttu toiminto, esimerkiksi haettua hakemusten tiedot tietokantaan.

Kuvassa 23 on Tampuuriin kirjautumiseen käytetyn tehtävän lähdekoodi. Tehtävät ovat Task-luokasta periytyviä luokkia, joiden __init__-metodissa määritellään tehtävän tarvitsemat perustiedot ja __call__-metodissa suoritetaan tehtävän toiminnot. Tehtävän __call__-metodin parametrina annettu context-muuttuja pitää sisällään esimerkiksi viittauksen selaimen käynnissä olevan instanssiin, tehtävän mahdollisesti vaatimat aiemmassa tehtävässä haetut tiedot sekä tehtävän tuottamat eteenpäin välitettävät tiedot.

```
5   class LoginTask(Task):
6       def __init__(self, config: Config):
7           self.name = "Login to Tampuuri"
8           self.task_type = TaskType.Browser
9           self._config = config
10
11      def __call__(self, context: Context):
12          page = context.page
13          page.goto(f"{self._config.baseurl}/v3/")
14          page.fill("text='Käyttäjätunnus:'", self._config.credentials.username)
15          page.fill("text='Salasana:'", self._config.credentials.password)
16          page.click("#btnKirjaudu")
17          page.frame_locator('[name="Header"]').locator("text=Etusivu").wait_for()
```

Kuva 23. Tampuuriin kirjautumisen tehtävä

Tehtävälistat muodostetaan tekemällä yksinkertainen lista, jonka jäseniksi liitetään tehtäväinstansseja halutussa järjestyksessä. Tehtäväinstanssia muodostaessa tehtävälle annetaan tarvittavat kiinteät asetustiedot sen mukaan, mitä tietoja kukin tehtävän vaatii. Kuvassa 24 on esimerkki irtisanomisten haun tehtävälistan alkuosasta. Tehtävälistassa lasketaan aluksi haettaville tiedoille aikaväli, jonka jälkeen aloitetaan itse tehtävälistan muodostus. Esimerkkitehtävälistan ensimmäisenä vaiheena on tietokantayhteyden alustus, toisena Tampuuriin kirjautuminen, kolmantena irtisanomislistauksen haku. Tämän jälkeen avataan ensin yhteys irtisanomisten tietokantaan, haetaan aiemmin kirjatut irtisanomiset ja tulkitaan irtisanomislistauksesta irtisanomisten avainarvot. Tehtävälistaan voi liittää vapaasti tarvittavan määrän kunkin tehtäväkokonaisuuden vaatimia tehtäviä ja myöhemmät tehtävät pystyvät käyttämään hyväkseen aiempien tehtävien tuottamia tietoja. Näin tehtäviä koostamalla saadaan muodostettua tehtäväkokonaisuudet jokaiseen eteen tulevaan tarpeeseen.

```
def create_tasklist(config: Config) -> List[Task]:
    start_date = date.today() - timedelta(days=config.reportdays)
    end_date = date.today()

    return [
        app.database.common.InitializeDBConnectionTask(config=config),
        app.tampuuriui.common.LoginTask(config=config),
        app.tampuuriui.workqueue.SearchWorkQueueTask(
            config=config,
            seach_params=app.tampuuriui.workqueue.SearchParams(
                task_class="Sopimukset",
                task_type="Sopimuksen irtisanominen",
                states=["Valmis", "Hylätty"],
                start_date=start_date,
                end_date=end_date,
            ),
        ),
        app.database.termination_notices.InitializeTerminationNoticesDBTask(
            output_key="notices_db"
        ),
        app.database.termination_notices.GetReportedTerminationNoticeIDsTask(
            database_key="notices_db", output_key="reported_ids", start_date=start_date
        ),
        app.tampuuriui.workqueue.GetTerminationNoticeIdsTask(
            filter_ids_key="reported_ids", output_key="notice_ids"
        ),
```

Kuva 24. Esimerkki irtisanomisten haun tehtävälistan alkuosasta

Tehtäväkokonaisuuksien suorittaminen tapahtuu Executor-luokassa (kuva 25), joka saa luokan alustamisen yhteydessä parametrina suoritettavan tehtävälistan. Luokka alustaa myös ennen tehtävälistan suoritusta selaininstanssin ja context-muuttujan. Tehtävälistan tehtävien suorituksen yhteydessä run_tasklist-metodi välittää parametrina tehtäville context-muuttujan. Tehtävien suorituksen poikkeuksien hallinta tapahtuu _execute_task-metodissa, joka kirjaa myös tiedon tehtävän suorituksesta lokitiedostoon ja välittää mahdolliset poikkeukset, näihin liittyvän lokitiedoston ja kuvan selaimen tilasta eteenpäin sähköpostin välityksellä. Tässä ohjelmistorobotin versiossa poikkeustilanne keskeyttää aina robotin suorituksen. Poikkeustilanteen hallintaan voitaisiin rakentaa myös esimerkiksi mahdollisuus uudelleenyrityksiin, mutta tässä tapauksessa tarvetta tällaiselle toiminnallisuudelle ei ollut.

```python
class Executor:
    def __init__(self, tasklist: List[Task], smtp_logger: SMTPLogger):
        self._smtp_logger = smtp_logger
        self._tasklist = tasklist

    def _execute_task(self, task: Task, context: Context) -> None:
        try:
            LOG.info("Running task %s", task.name)
            task(context)
            LOG.info("Task %s succeeded", task.name)
        except Exception:
            LOG.exception("Task %s failed:\n", task.name)
            self._smtp_logger.send_logs(io.BytesIO(context.page.screenshot()))
            raise RuntimeError(f"Step {task.name} failed")

    def run_tasklist(
        self,
        headless: bool,
    ) -> None:
        with sync_playwright() as pw:
            browser = pw.chromium.launch(headless=headless)
            page = browser.new_page()
            context = Context(page=page, output={})
            for task in self._tasklist:
                self._execute_task(task, context)
```

Kuva 25. Tehtävälistoja suorittava Executor-luokka

Moduulikokonaisuuksien lisäksi ohjelmiston juuressa on ohjelmiston käynnistämiseen tarvittava main.py-moduuli, ohjelmiston asetustiedosto config.yaml, sekä projektin määrittelyyn käytettäviä tiedostoja. Tällä rakenteella jo toteutettuja osatehtäviä pystyy yhdistelemään vapaasti uusiksi tehtäväkokonaisuuksiksi tekemällä uuden tehtävälistan Tasklists-kokonaisuuden alle ja uudet Database- tai TampuuriUI-kokonaisuuksiin toteutettavat osatehtävät ovat automaattisesti käytettävissä myös kaikissa muissa tehtäväkokonaisuuksissa.

Ohjelmiston suorituksen voi ajastaa tapahtumaan säännöllisesti käyttämällä esimerkiksi Linux-järjestelmien Crontab-ohjelmaa tai Windowsin Task Scheduleria. Tieto mahdollisesti epäonnistuneesta suorituksesta saadaan sähköpostitse. Mikäli ongelman syy ei selviä sähköpostista, ohjelmiston suorituksen voi ongelman selvittämiseksi toistaa siten, että selaimen ikkuna on näkyvissä.

4.4 Irtisanomisten käsittelyajan mittari

Vuokrasopimusten irtisanomiset tehdään pääsääntöisesti sähköisesti, mutta irtisanominen vaatii käsittelyä taustajärjestelmässä ennen kuin päättäminen saadaan kirjattua sopimukselle ja sopimus päätettyä. Vapautuvaa asuntoa markkinoidaan asiakkaille vasta sen jälkeen, kun sopimus on päätetty, joten irtisanominen pitäisi saada käsiteltyä mahdollisimman pian irtisanomisen saapumisen jälkeen. Sillä, kuinka pitkään asuntoa on markkinoitu asiakkaille, voidaan tästä syystä olettaa olevan yhteys siihen, koska asunto saadaan vuokrattua. Tätä kautta syntyy myös välillinen yhteys käyttöasteeseen. Edellä mainituista syistä johtuen irtisanomisten käsittelyajan mittari on hyvä ehdokas myös nyt muodostettavaan raporttiin.

4.4.1 Tietolähteet, tietojen haku ja muokkaus

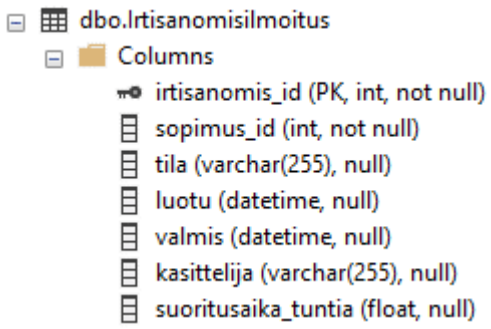
Tieto sopimuksille kirjautuista irtisanomisista löytyy tietovarastosta, mutta tietoa irtisanomisten käsittelyajasta ei kyseisestä tietolähteestä löydy, joten tämän mittarin muodostamiseksi vaadittavat tiedot täytyy hakea muualta. Käyttävissä olevien tietolähteiden analysoinnin tuloksena päädyttiin siihen, että tieto on haettavissa ainoastaan Tampuurin käyttöliittymän kautta.

Irtisanomisten käsittelyaika lasketaan irtisanomisen saapumisajan ja valmiiksi- tai hylätyksi kuittaamisajan erotuksesta. Irtisanomiset käsitellään Tampuurissa asiakaspalvelumoduulin työjono-toiminnon kautta. Toiminnolla voi hakea halutulla aikavälillä jätetyt tietyn tilan omaavat tehtävät ja avata kyseiset tehtävät käsittelyä tai tarkastelua varten. Irtisanomisen saapumisaika löytyy tehtävän käsittelylomakkeen muutoslokilta, kuten myös tieto tehtävän kuittausajasta. Lisäksi lomakkeelta löytyy tieto siitä, mitä sopimusta tehtävä koskee. Tietoa siitä, mitä sopimusta irtisanominen koskee, tarvitaan, mikäli irtisanomisia halutaan tarkastella erikseen esimerkiksi asumismuodon mukaan.

Irtisanomisten hakua varten muodostettiin ohjelmistorobotille tehtävälista, joka suorittaa seuraavat tehtävät:

1. Alustaa tietokantayhteyden.
2. Kirjautuu Tampuuriin.
3. Hakee irtisanomislistauksen annetuilla parametreilla.
4. Alustaa yhteyden irtisanomisten tietokantaan.
5. Hakee aiemmin kirjatut irtisanomiset tietokannasta.
6. Hakee irtisanomiset halutulta aikaväliltä ja suodattaa pois aiemmin kirjatut irtisanomiset.
7. Avaa irtisanomiset ja hakee tarvittavat tiedot irtisanomiselta.
8. Kirjaa irtisanomisten tiedot tietokantaan.

Irtisanomisten tiedot kirjataan kuvan 26 mukaiseen tietokantatauluun. Vaikka tauluun kirjataankin sekä tehtävän luonti- että valmistumisaika, tehtäville lasketaan valmiiksi myös suoritusaika, koska aikamääreet saattavat osua kesä- ja talviajan vaihtoajankohtaan, jolloin suoritusajan laskeminen rivin aikamääreistä tuottaisi väärän tuloksen. Irtisanomisesta kirjataan myös sen sopimuksen tunniste, jota irtisanominen koskee, jotta irtisanomiset voidaan suodattaa raportilla esimerkiksi omistajittain tai asumismuodon mukaan.



Kuva 26. Irtisanomisilmoitusten tietokantataulu

4.4.2 Mittarin muodostaminen

Irtisanomisten käsittelyajalle muodostetaan mittari, joka laskee liukuvaa keskiarvoa tarkasteluhetkestä katsottuna edellisen 7 päivän aikana käsitellyistä irtisanomisista. Liukuvaa keskiarvoa käytetään tasaamaan raportoituja arvoja, koska irtisanomisia tulee tiettyinä ajankohtina melko vähän ja esimerkiksi viikonloppuina ei välttämättä käsitellä irtisanomisia. Kuvassa 27 on esitelty mittarin laskentaan käytetty DAX-koodi.

```
IrtisanomisteKasittelyaika_7pv_ka =
VAR Loppuaika = MAX(D_Paiva[SQLPvm])
VAR Alkuaika = Loppuaika - 7
RETURN CALCULATE(
    AVERAGE(Irtisanomisilmoitus[suoritusaika_tuntia]),
    ALL(D_Paiva),
    ALL(F_SopimusKK),
    D_Paiva[SQLPvm] >= Alkuaika && D_Paiva[SQLPvm] <= Loppuaika
)
```

Kuva 27. Irtisanomisten käsittelyajan mittari

Mittari visualisoidaan samaan tapaan kuin aiemmat mittarit, eli viivakaaviona edellisen 28 päivän ajalta, sekä koko aikajakson keskiarvoa kuvaavaa lukuarvoa näyttävänä mittarina (kuva 28). Mittarin visualisoinnit pyrittiin pitämään mahdollisimman samankaltaisina, kuin aiemmat visualisoinnit, jotta mittaristosta muodostuu eheä kokonaisuus ja jotta raportti olisi helppokäyttöinen ja selkeä.

Kuva 28. Irtisanomisen käsittelyajan mittarin visualisoinnit

4.5 Hakemusten käsittelyajan mittari

Vuokra- ja asumisoikeusasunnon saadakseen asiakkaiden tulee useimpien vuokranantajien osalta täyttää hakemus. Usein asiakkaat tekevät hakemuksen useammalla toimijalle, joten nopeimmin sopivaa asuntoa asiakkaalle tarjoavan tahon voidaan olettaa olevan etulyöntiasemassa. Täten myös hakemusten käsittelyajalla voidaan olettaa olevan välillinen yhteys käyttöasteeseen. Toinen mittarin valintaa puoltaneista seikoista on se, että työntekijöillä on suora vaikutusmahdollisuus mittarin arvoihin. Edellä mainituista syistä johtuen mittari sopii hyvin käyttöasteen kehitystä seuraavaan mittaristoon ja antaa lisäarvoa mittaristolle.

4.5.1 Tietolähteet, tietojen haku ja muokkaus

Vuokra- ja asumisoikeushakemukset tehdään pääosin sähköisesti ja nekin hakemukset, joita ei ole tehty sähköisesti, kirjataan Tampuuriin käsittelyä varten. Vaikka hakemuksista muodostetaan kerran vuorokaudessa tietokokonaisuus Tampuurin tietovarastoon raportointia varten, tiedot eivät sisällä hakemusten käsittelyajan laskemiseen tarvittavia tietoja. Myöskään Tampuurin ohjelmointirajapinta ei pidä sisällään tarvittavia tietoja, joten ainoaksi vaihtoehdoksi jää hakea tiedot Tampuurin käyttöliittymän kautta.

Hakemusten tiedot löytyvät Tampuurista asiakaspalvelumoduulin hakemukset-toiminnosta. Toiminnolla pystyy hakemaan listauksen tietyllä aikavälillä tulleista hakemuksista ja avaamaan hakemukset käsittelyä varten. Tiedot hakemuksen käsittelystä löytyvät hakemuksen lokiosiosta. Hakemusten käsittelyajan hakemista varten muodostettiin ohjelmistorobotille tehtävälista, joka suorittaa seuraavat tehtävät:

1. Alustetaan tietokantayhteys.

2. Kirjaudutaan Tampuuriin.

3. Haetaan hakemukset halutuilla parametreilla rajattunan.

4. Alustaan yhteys hakemustietokantaan.

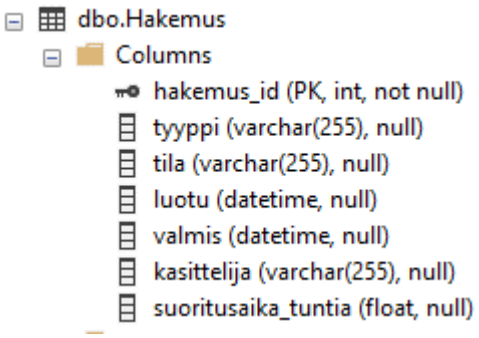
5. Haetaan listaus aiemmin kirjattujen hakemusten yksilöivistä arvoista.

6. Käydään läpi hakemuslistaus ja suodatetaan pois aiemmin kirjatut hakemukset.

7. Avataan hakemukset ja tulkitaan tarvittavat tiedot hakemuksilta.

8. Tallennetaan hakemusten tiedot tietokantaan.

Hakemusten tiedot kirjataan kuvan 29 mukaiseen tietokantatauluun. Hakemusta ei pysty läheskään aina linkittämään suoraan mihinkään tiettyyn asuntoon tai kiinteistöön, joten hakemustietojen osalta tietoja ei voida liittää tietomallin kohdetietoihin. Sen sijaan hakemukselta löytyy tieto siitä, onko kyseessä asumisoikeushakemus vai vuokrahakemus. Tätä tietoa voidaan käyttää raportilla tietojen suodatukseen asumismuodon perusteella.



Kuva 29. Hakemusten tietokantataulu

4.5.2 Mittarin muodostaminen

Hakemusten käsittelyajan raportointia varten tehdään mittari, joka laskee tarkasteluhetkestä katsottuna edellisen 7 päivän liukuvaa keskiarvoa. 7 päivän keskiarvoa käytetään tasaamaan käsittelyajan ulkopuolella tulleiden hakemus-

ten vaikutusta mittariin. Liukuvalla keskiarvolla saadaan tasattua myös päiväkohtaisen hakemusmäärän aiheuttamat vaihtelut. Kuvassa 30 on mittarin laskemisessa käytetty DAX-koodi.

```
1 HakemustenKasittelyaika_7pv_ka =
2 VAR Loppuaika = MAX(D_Paiva[SQLPvm])
3 VAR Alkuaika = Loppuaika - 6
4 RETURN CALCULATE(
5     AVERAGE(Hakemus[suoritusaika_tuntia]),
6     ALL(D_Paiva),
7     D_Paiva[SQLPvm] >= Alkuaika && D_Paiva[SQLPvm] <= Loppuaika
8 )
```

Kuva 30. Hakemusten käsittelyajan mittarin koodi

Myös tämän mittarin visualisointi toteutettiin viivakaavion ja koko tarkasteluaikavälin keskiarvoa kuvaavana numeroarvona. Näin hakemusten ja irtisanomisten käsittelyajan mittareista saatiin visuaalisesti vastaavat ja keskenään vertailukelpoiset. Kuvassa 31 on tuloksena syntynyt visualisointikokonaisuus.

Kuva 31. Hakemusten käsittelyajan visualisoinnit

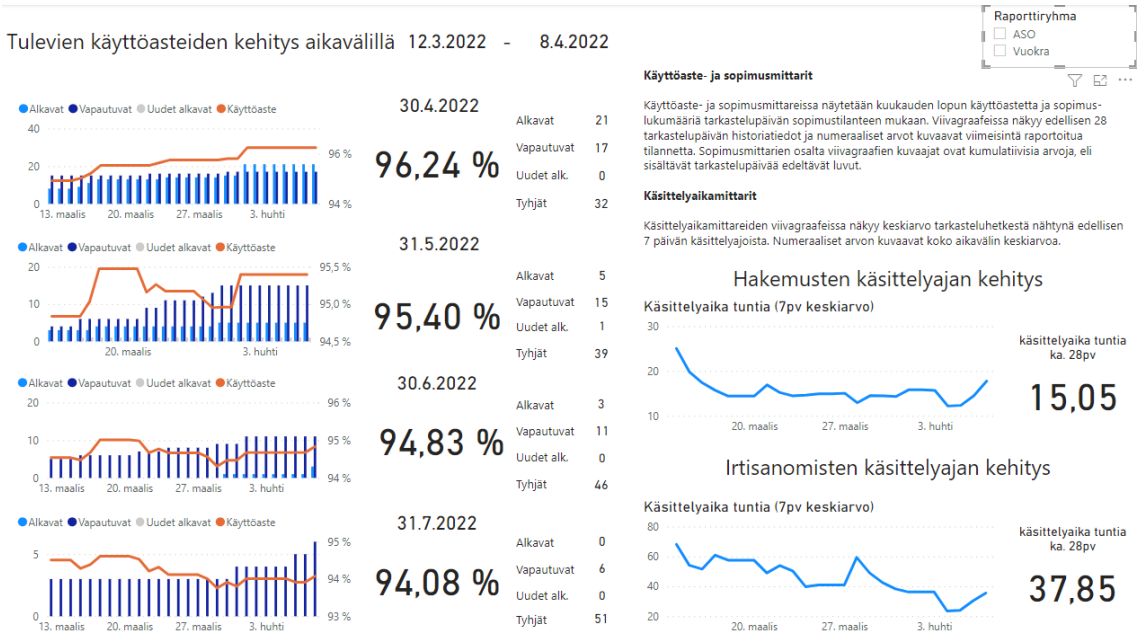
4.6 Raportin koostaminen

Power BI -palveluun voi muodostaa loppukäyttäjille tulosten tarkastelua varten raportteja ja koontinäyttöjä. Raportit voivat olla monisivuisia, käyttävät yleensä

yhtä tietolähdettä, sisältävät monesti yksityiskohtaisempia tietoja kuin koontinäytöt, sekä mahdollistavat suodatusvalintojen muuttamisen. Koontinäytöt taas voivat koostaa samalla näytölle visualisointeja useammalta eri raportilta ja ovat yksisivuisia. (Johdatus raporttinäkymiin Power BI-kehittäjille 2022.)

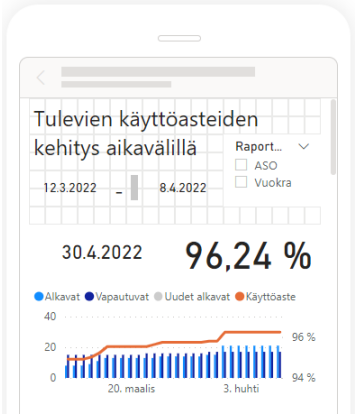
Koontinäyttöjä käytetään monesti kertomaan tarinaa koostamalla sopivia visualisointeja useammasta tietolähteestä yhdelle näytölle. Koontinäytölle voidaan tuoda esimerkiksi visualisoinnit, joilla voidaan seurata asiakastyytyväisyyden yhteyttä käyttöasteeseen tai käyttöasteen yhteyttä liiketoiminta-alueen tulokseen. Koontinäytöt mahdollistavat myös esimerkiksi kokonaisnäkymän muodostamisen jonkin liiketoiminta-alueen toimintaan tuomalla yhteen näyttöön esimerkiksi vuokraukseen, asuntojen kunnossapitoon ja isännöintiin liittyviä tietoja.

Tähän insinöörityöhön liittyvien tietojen esittämiseen käytetään yhtä raporttia, koska raportin tiedot liittyvää kiinteästi yhteen ja raportille ei ole tarvetta hakea tietoja toisilta raporteilta. Käyttöaste- ja sopimusmäärien visualisoinnit näytetään kuluvan ja kolmen seuraavan kuukauden osalta. Näytettävien kuukausien lukumäärä valittiin, koska asumisoikeusasunnoissa on kolmen kuukauden irtisanomisaika ja myös asumisoikeusasuntojen irtisanomisten kertyminen haluttiin näyttää raportilla. Raportilla näytetään käyttöaste-, sopimus- ja käsittelyaikatietojen osalta edellisen neljän viikon, eli 28 päivän historiaa. Neljän viikon historiaan päädyttiin, jotta graafeihin saadaan näkyville kuukauden sisällä tapahtuva vaihtelu. Hyvä esimerkki kuukauden sisällä tapahtuvasta vaihtelusta on vuokraasuntojen irtisanomiset, joista suuri osa tehdään tyypillisesti lähellä kuun loppua, eli viimeistä mahdollista irtisanomispäivää. Lisäksi raportille tuotiin osittaja, jolla voi rajata raportilla näyttävän tiedot haluttuun asumismuotoon, sekä tekstikenttä, jossa on ohje raportin lukujen tulkintaa varten. Muodostetut visualisoinnit saatiin koostettua yhdelle raporttisivulle (kuva 32), josta näkee helposti yhdellä silmäyksellä raporttihetken tilanteen sekä raportin arvojen kehityksen edellisen neljän viikon osalta.



Kuva 32. Aiemmat visualisoinnit yhdistettynä raportiksi

Raportille voi tehdä myös erillisen mobiililaitteille tarkoitetun näkymän. Mobiililaitenäkymässä visualisoinnit voivat olla järjestetty täysin eri järjestykseen kuin työpöytänäkymässä ja kaikkia visualisointeja ei välttämättä tarvitse edes näyttää. Koska mobiililaitteiden näytön koko on rajoitettu esimerkiksi tietokoneen näyttöön verrattuna, näkymään tuodaankin usein vain raportin keskeisimmät tiedot. Koska osalla muodostetun raportin käyttäjistä on käytössään Power BI:n mobiilisovellus, muodostettiin raportille mobiilinäkymä (kuva 33), jossa osa visualisoinneista jätettiin pois.



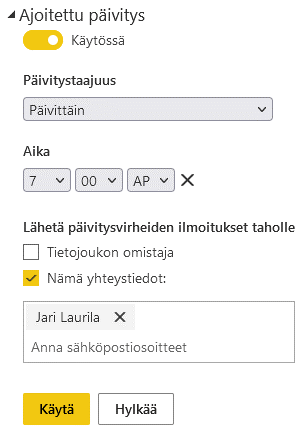
Kuva 33. Raportin mobiilinäkymä

4.7 Tietojen päivityksen automatisointi

Muodostettu raportti pitää sisällään päivätason tietoja, joten myös raportin tietosisältö täytyy saada päivittymään kerran vuorokaudessa. Koska päivitysfrekvenssi on näin tiheä, käsin tapahtuva päivittäminen muodostuisi turhan raskaaksi. Tästä syystä raportin tiedot täytyy saada päivittymään automaattisesti, ja tieto mahdollisista päivityksen virheistä täytyy saada siirtymään raportin ylläpidosta vastaavalle taholle.

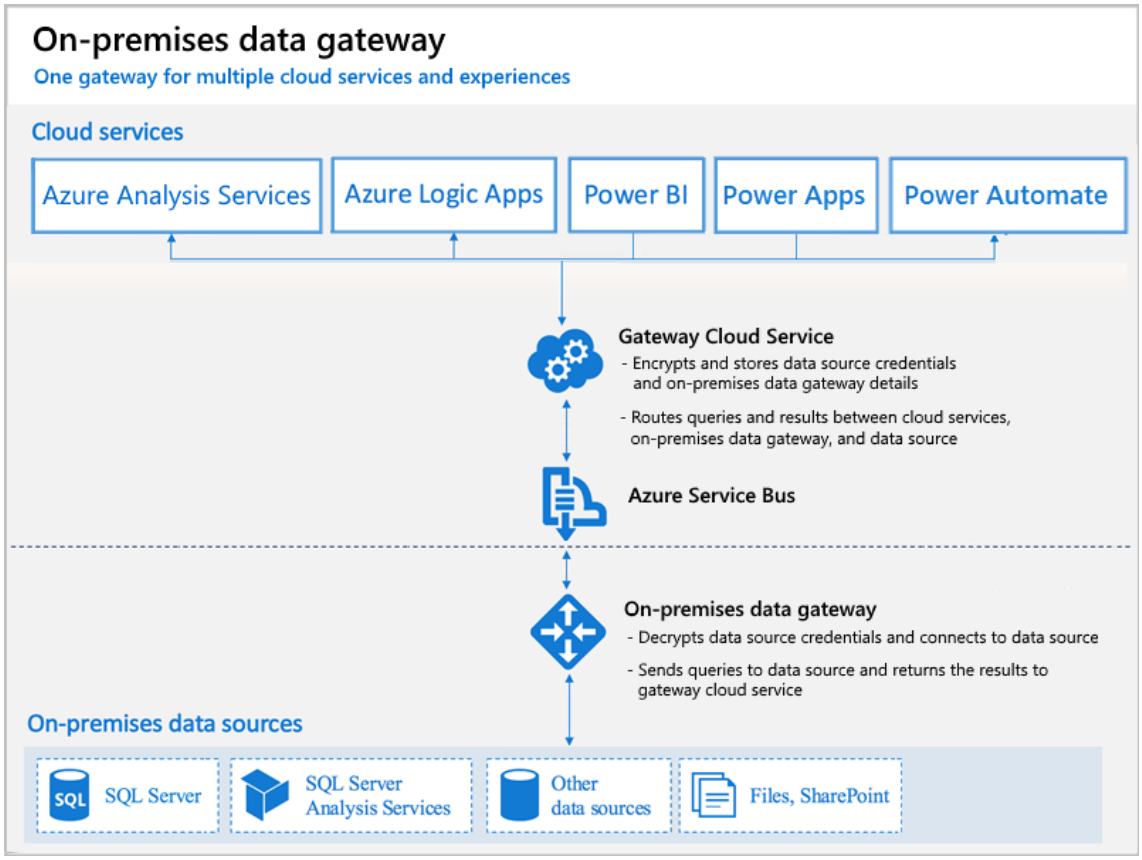
Osa raportin tiedoista haetaan jo valmiiksi ajastuksen kautta päivittyvästä tietokanta-aineistosta, joten tämän aineiston osalta tehtäväksi jää ainoastaan Power BI:n tietojenhaun automatisointi. Loput raportin tiedot ovat sellaisia, jotka haetaan ohjelmistorobotin avulla käyttöliittymästä. Näiden tietojen osalta vaaditaan Power BI:n tietojenhaun automatisoinnin lisäksi myös ohjelmistorobotin automatisointia.

Power BI -palvelussa on sisäänrakennettu mahdollisuus tietojenhaun automatisointiin. Riippuen käytetystä ohjelmistolisenssistä tiedot voi asettaa päivittymään esimerkiksi kerran vuorokaudessa, mikä on tämän insinöörityön raportille riittävä päivityssykli. Palvelussa on myös valmis tuki sille, että mikäli tietoja päivittäessä tapahtuu virheitä, lähetetään tieto virheestä halutuille tahoille. Palvelu kerää lisäksi historiatietoa tietojenpäivityksen onnistumisesta. Kuvassa 34 on kuvattu Power BI -palvelusta löytyvät tietojenpäivitykseen liittyvät asetukset.



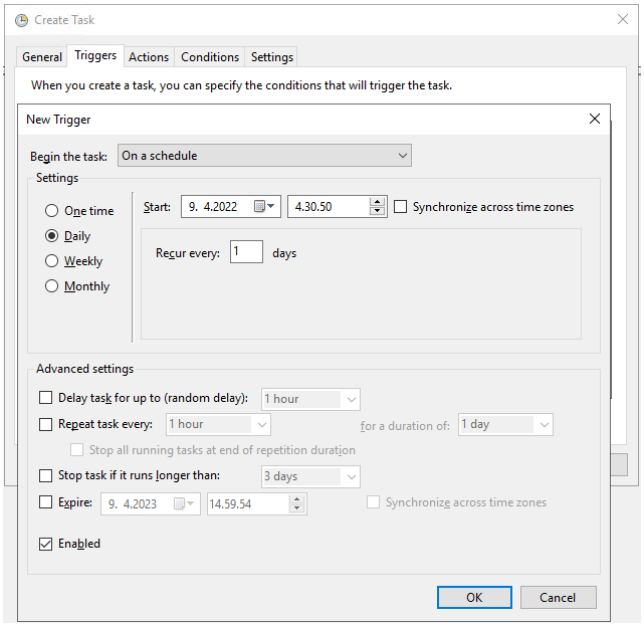
Kuva 34. Power BI -palvelun tietojen päivityksen asetukset

Mikäli Power BI -palvelu pääsee suoraan käsiksi raportin tietolähteeseen, ei tarvita muita komponentteja palvelusta valmiiksi löytyvien lisäksi. Mikäli tiedot sijaitsevat kuitenkin sellaisessa palvelussa, johon ei pääse käsiksi julkisen tietoverkon kautta, täytyy asentaa paikallinen tietoyhdyskäytävä sellaiseen sijaintiin, josta tietoihin pääsee käsiksi. Yhdyskäytävä mahdollistaa tietojen haun esimerkiksi yrityksen oman sisäverkon palvelimilta tai ulkoisesta fyysisestä tai virtuaalisesta konesalista. Yhdyskäytävä mahdollistaa tietojen haun esimerkiksi tietokantapalvelimilta tai verkkolevyiltä. Kuvassa 35 on kuvattu paikallisen tietoyhdyskäytävän toimintaperiaate. Tämän insinöörityön tietojoukon tiedot sijaitsevat ulkoisessa virtuaalisessa konesalissa, joten tietojen hakuun käytetään yhdyskäytäväyhteyttä.



Kuva 35. Paikallisen tietoyhdyskäytävän toimintaperiaate (Microsoft 2022.)

Ohjelmistorobotin automatisointi voidaan toteuttaa käyttäen esimerkiksi käyttöjärjestelmän sisäänrakennettua tehtävienajastustoimintoa. Tämän työn tapauksessa robotti asennettiin Windows-käyttöjärjestelmää käyttävälle palvelimelle, joten ajastus toteutettiin Windowsin Task Scheduler -toiminnolla (kuva 36). Ajastuksia tehdessä tulee huomata se, että lähdetietojen haun täytyy olla valmiina hyvissä ajoin ennen Power BI:n ajastettua tietojenpäivitysajoa. Task Schedulerissa ei ole sisäänrakennettuna toiminnallisuutta, jolla voisi ilmoittaa epäonnistuneesta ajosta eteenpäin, joten tehtävien valvonta täytyy toteuttaa jollakin toisella tavalla. Tässä tapauksessa tehtävän valvonta hoituu robotin sisäänrakennetulla toiminnallisuudella, joka valvoo tehtävien onnistumista ja lähettää mahdollisen virheen kohdatessaan tiedot virheestä eteenpäin ohjelman asetuksissa määritellylle taholle sähköpostin välityksellä.



Kuva 36. Tehtävän ajastus Windows Task Schedulerilla

5 Tulokset ja jatkokehitysmahdollisuudet

Ajatus tämän insinöörityön raportin toteuttamisesta oli ollut esillä jo pidemmän aikaa ennen työn aloittamista, joten työn toteuttamisen suunnittelemisen kanssa päästiin nopeasti vauhtiin, eikä alustava suunnitelma vaatinut pitkää iterointikierrosta. Toki itse toteutusta tehdessä tuli muutamaan kertaan eteen tilanteita,

joissa alkuperäinen ajatus siitä, miten joku toiminnallisuus tulee toteuttaa, ei toiminut ja toteutus täytyi tästä syystä suunnitella uudestaan. Muutamaa teknistä toteutusta tehdessä itse toteutus tuli tehtyä uudestaan useamman kerran, joka on aiemman kokemuksen perusteella varsin normaalia vastaavia projekteja toteuttaessa. Suurimmat haasteet työn tekemisessä liittyivät tietojen muokkaamiseen tietojen haun yhteydessä SQL-skripteillä, sekä Power BI:n mittareiden laskennassa käyttämän käsitemallin ymmärtämiseen. Nämäkin haasteet saatiin kuitenkin useamman väliversion jälkeen ratkaistua ja samalla tuli opittua paljon uutta etenkin Power BI:n mittareiden rakentamiseen ja ohjelman käsitemalliin liittyen.

5.1 Lopullinen raportti

Insinöörityön tavoitteena oli luoda proaktiivinen eli ennakoiva mittari käyttöasteen seurantaa, sekä luoda kaksi uutta mittaria, joista ensimmäinen seuraa irtisanomisten käsittelyaikaa ja toinen hakemusten käsittelyaikaa. Käyttöastemittarin tavoitteena oli siirtää katse historian seuraamisesta tulevaan ja mahdollistaa tätä kautta mittarin lopullisiin arvoihin vaikuttaminen. Irtisanomisten ja hakemusten käsittelyaikojen mittareiden suhteen tavoitteena oli tuoda täysin uutta tietoa toiminnan ohjaamisen tueksi. Lisäksi molempien kokonaisuuksien osalta haluttiin siirtyä kuukausitasoisesta seurannasta päivätasoiseen seurantaan, jotta päivittäin tehty työ saadaan paremmin näkyville ja raportointi on reaaliaikaisempaa.

Lopullinen raportti täyttää mielestäni hyvin alkuperäiset tavoitteet. Raportilta pystyy seuraamaan käyttöasteen kehitystä useamman kuukauden tulevaisuuteen ja käyttöasteen kehityksen historiaa saa seurattua päivätasolla neljä viikkoa taaksepäin. Lisäksi uudet mittarit saatiin visualisoitua selkeästi samaan raporttikokonaisuuteen käyttöastemittarien kanssa. Raportin soveltuvuus päivittäiseen käyttöön tulee arvioitua tarkemmin sen jälkeen, kun raportti julkaistaan raportin koko käyttäjäkunnalle.

Käyttöastelukujen ennustaminen tehdään tietolähteen rajoituksista johtuen osittain eri tiedoista kuin aiemman kuukausittaisen käyttöasteraportin, joten uusi raportti ei välttämättä täsmää täysin aiempaan raporttiin. Eroja pyrittiin kuitenkin minimoimaan käyttämällä aina kun mahdollista samaa tietolähdettä kuin aiempi raportti. Uuden raportin lukuja vertailtiin aiemman raportin lukuihin ja uuden raportin käyttöaste oli esimerkiksi maaliskuun 2022 lukujen osalta 0,01 prosenttiyksikön sisällä vanhan raportin luvuista, joten uuden raportin luvut näyttivät täsmäävän varsin hyvin vanhaan raporttiin.

Uuden raportin käyttöastemittari käyttää tietolähteenään asiakkaiden sopimukselle merkittyjä maksulajeja, eikä muodostettuja laskuja, kuten aiemmat raportit. Tampuurissa pystyy käsin muuttamaan muodostettujen laskujen summia, sekä poistamaan laskulta maksulajeja. Mikäli näin tehdään, eivät vanhan ja uuden raportin luvut täsmää. Muodostettujen laskujen summien muuttaminen käsin tai maksulajien poistamien laskuilta ei ole suositeltu toimintapa, joten edellä mainittua virhettä ei pitäisi esiintyä. Toinen virhelähde voi olla se, että sopimuksen maksulajille muutetaan hintaa, mutta sopimukselle ei muodosteta uusia laskuja muutetuilla hinnoilla. Myöskään tätä virhettä ei pitäisi normaalitilanteessa esiintyä. Vaikka edellä mainittuja virhetilanteita ei pitäisi esiintyä, olisi virheellisten tietojen välttämiseksi hyvä laatia raportti, jolla saisi listattua edellä mainitut kriteerit täyttävät sopimukset.

5.2 Jatkokehitysmahdollisuudet

Tässä työssä toteutetussa raportissa keskityttiin koko manageerausta koskevien käyttöaste- ja sopimusmittarien lisäksi ainoastaan vuokrauksen toimintaan liittyviin mittareihin. Asuntojen käyttöasteeseen vaikuttavia tekijöitä on kuitenkin huomattavasti enemmän, ja osa näistä tekijöistä ei liity suoraan vuokrauksen toimintaan.

Tämän työn puitteissa muodostetun raportin lähtökohtana on vapautuvien asuntojen mahdollisimman tehokas uudelleenvuokraaminen. Toinen lähestymis-

kulma on se, että pyritään välttämään sitä, että asiakkaat edes vaihtavat asuntoa. Tämä ei toki kaikissa tilanteissa ole mahdollista, mikäli muuton syynä on esimerkiksi muutto toiselle paikkakunnalle tai asunnon koon epäsopivuus perhetilanteen muutoksesta johtuen. Näidenkin jälkeen jää jäljelle kuitenkin tilanteita, joissa irtisanominen olisi ollut vältettävissä, mikäli jokin asia esimerkiksi asunnon kunnossapitoon, kiinteistön kunnossapitoon tai isännöinnin toimintaan liittyen olisi tehty toisin. Näihin tilanteisiin liittyen voisi muodostaa mittareita, joilla seurattaisiin vaihtuvuutta ennakkoon samaan tapaan kuin käyttöastetta. Näiden mittarien osalta olisi hyödyllistä tarkastella tietoja sekä asunto-, että kiinteistökohtaisesti, jotta päästäisiin kiinni poikkeuksellisen vaihtuvuuden syihin ja pystyttäisiin puuttumaan ongelmiin mahdollisimman aikaisessa vaiheessa.

Vaihtuvuuden syiden selvittämiseen työkaluja voisi löytyä asiakkaille lähetettyjen kyselyjen tulosten analysoinnista. Joidenkin kyselyiden osalta haasteena on tosin se, että kyselyyn ei saada vastauksia riittävästi, jotta tuloksista pystyisi tekemään luotettavia analyysejä. Yksi tapa ratkaista asia on laskea tuloksista keskiarvoja pidemmältä aikaväliltä. Tämän lähestymistavan ongelmaksi voi muodostua mittarin hidas reagointi. Toinen työkalu syiden selvittämiseen voisi olla kyselyiden avoimien vastauksien tuominen vaihtuvuusraportin rinnalle, jolloin myös yksittäisistä vastauksista pystyttäisiin tulkitsemaan ongelman syitä paremmin.

Yksi mielenkiintoinen tietolähde ongelmakohteiden tai -asuntojen selvittämiseen voisi olla myös yrityksen asiakkaiden kanssa viestintään käytetty ohjelmisto. Lähes kaikki asiakkaan kanssa käyty viestintä kulkee kyseisen sovelluksen kautta ja tietoa järjestelmästä voisi kerätä esimerkiksi luokittelemalla tiettyä ongelmaluokkaa koskevat yhteydenotot järjestelmässä asiakaspalvelijan toimesta. Ongelmaluokkia voisi muodostaa esimerkiksi huollon toiminnalle tai asumisen häiriöille. Luokitellut yhteydenotot voitaisiin kerätä järjestelmästä ja yhdistää taustatietojen kautta asuntotasolle, jolloin saataisiin muodostettua hyvä ja lähes reaaliaikainen ongelmaindikaattori. Tästä indikaattorista voisi muodostaa oman raportin ja tuoda saman tiedon myös vaihtuvuutta seuraavan mittarin rinnalle.

Mittarit, jotka mahdollistaisivat nopean puuttumisen vaihtuvuuden syihin, olisivat hyvä lisä aiempien mittareiden rinnalle ja ulottaisivat mittarin koskemaan paremmin myös manageerauksen muuta toimintaa vuokrauksen lisäksi ja mahdollistaisivat puuttumisen vaihtuvuuteen nykyistä aikaisemmassa vaiheessa.

Lähteet

Enho, Heidi. 2021. Verkkoaineisto. Power BI – hinnat ja lisenssit. <https://hex-celligent.fi/2021/10/10/power-bi-hinnat-ja-lisenssit-v2/>. Luettu 12.2.2022.

Data Flair. Verkkoaineisto <https://data-flair.training/blogs/power-bi-tutorial/>. Power BI Tutorial – A Complete Guide on Introduction to Power BI. Luettu 12.2.2022.

Gramlich, Michael. 2020. Verkkoaineisto. What is structured, semi structured and unstructured data? <https://www.michael-gramlich.com/what-is-structured-semi-structured-and-unstructured-data/>. Luettu 12.2.2022.

Hämäläinen, Virpi, Maula, Hanna & Suominen, Kimmo. 2016. Digiajan strategia. Helsinki: Alma.

Ikäheimo, Seppo, Malmi, Teemu & Walden, Risto 2019. Yrityksen laskentatoimi. 8., uudistettu painos. Helsinki: Alma Talent Oy.

Laki asuinhuoneiston vuokrauksesta. 1995. 481/31.3.1995

Laki asumisoikeusasunnoista. 2021. 393/7.5.2021

Microsoft. 2022. Verkkoaineisto. Mikä Power BI on? <https://docs.micro-soft.com/fi-fi/power-bi/fundamentals/power-bi-overview>. Luettu 12.2.2022.

Microsoft. 2022. Verkkoaineisto. Johdatus raporttinäkymiin Power BI-kehittäjille. https://docs.microsoft.com/fi-fi/power-bi/create-reports/service-dashboards. Luettu 9.4.2022.

Rainardi, Vincent. 2008. Building a Data Warehouse. Apress.

Ray, Saikat & Villa, Arthur & Rashid, Navid & Vincent, Paul & Guttridge, Keith & Alexander, Melanie. 2021. Verkkoaineisto. <https://www.gart-ner.com/doc/reprints?id=1-26Q65VFT&ct=210706&st=sb>. Luettu 13.2.2022.

Tanskanen, Aapo. 2018. Verkkoaineisto. Intro to process automation with RPA. <https://gofore.com/en/intro-to-process-automation-with-rpa/>. Luettu 13.2.2022.

Tapper, Adam. 2021. Verkkoaineisto. Playwright vs WebDriver: The Future of Browser Automation. https://medium.com/slalom-build/playwright-vs-webdriver-the-future-of-browser-automation-854a7ae63218. Luettu 13.2.2022.

Taulli, Tom. 2020. Robotic Process Automation Handbook: A Guide to Implementing RPA Systems. Apress.

Verbeeck, Koen. 2020. Verkkoaineisto. Power BI Desktop Data Source Considerations. <https://www.mssqltips.com/sqlservertip/6428/power-bi-desktop-data-source-considerations/>. Luettu 12.2.2022.

YH Kodit Oy. Vastuullisuuskertomus 2020. 2021. Verkkoaineisto. <https://www.yhkodit.fi/content/uploads/2021/04/YHKodit_Vastuullisuuskertomus_2020.pdf>. Luettu 6.2.2022