Tietokantasiirtymä: Matka Microsoft SQL Serveristä Snowflakeen

Tatu Tulkki

Haaga-Helia ammattikorkeakoulu
Tradenomin tutkinto
Toiminnallinen opinnäytetyö
2024

Tiivistelmä

Tekijä(t)
Tatu Tulkki

Tutkinto
Tradenomi

Opinnäytetyön nimi
Tietokantasiirtymä: Matka Microsoft SQL Serveristä Snowflakeen

Sivu- ja liitesivumäärä
48

Tämä toiminnallinen opinnäytetyö toteutettiin toimeksiantona yritykselle X. Työn tavoitteena on parantaa tietokannan skaalautuvuutta ja laskentatehoa siirtymällä Microsoft SQL Server -tietokannasta Snowflake-tietokantaan. Siirtymään kuuluu oleellisimpien tietokantakomponenttien uudelleen rakentaminen uuteen tietokantajärjestelmään.

Opinnäytetyö perustuu tietokantasiirtymän parhaisiin käytäntöihin sekä SQL Serverin ja Snowflaken arkkitehtuurien vertailuun. Kirjallisuuden tarjoamien tietojen pohjalta vertaillaan SQL Serverin ja Snowflaken ominaisuuksia ja todetaan organisaation tietojenkäsittelytarpeiden linjautuvan Snowlflaken pilvipohjaisen ja skaalautuvan arkkitehtuurin kanssa.

Tietokantasiirtymä aloitettiin analysoimalla tietokantajärjestelmän nykytila.
Tämän jälkeen tarkasteltiin teoriaa ja poimittiin parhaat käytännöt analyysissä nousseiden avainkohtien ratkaisemiseksi. Näiden pohjalta toteutettiin tietokantasiirtymä vaiheittain.

Opinnäytetyön kehityshaasteena oli nykyisen tietokantajärjestelmän laskentatehon rajallinen skaalautuvuus, joka johtaa pitkiin laskentaprosesseihin. SQL Server on luotettavasti toiminut organisaation tietokantajärjestelmänä, mutta liiketoiminnan ja datamäärän kasvaessa vanhan ratkaisun rajoitukset tulevat yhä näkyvämmäksi.

Tuloksena eniten laskentatehoa ja aikaa vievät laskentaprosessit on siirretty Snowflake-ympäristöön, joka mahdollistaa alhaisen viiveen organisaation keskeisimmissä palveluissa. Samalla kyky vastaanottaa suuria datamääriä kasvaa ja toimintavarmuus nousee.

Tämä opinnäytetyö tukee yrityksen tavoitetta kasvattaa liiketoimintaa takaamalla kyvyn hallinnoida kasvavaa datamäärää.

Asiasanat
Tietokannat, tietovarasto, Snowflake, SQL Server, tietokantasiirtymä, modernisointi

Sisällys

1 Johdanto

Datan eksponentiaalinen kasvu sekä sen yhä suurempi merkitys liiketoiminnassa muokkaavat tapaa, jolla yritykset luovat arvoa. Panostamalla datan keruuseen ja sen analysointiin, yritykset pystyvät tarjoamaan asiakkailleen ainutlaatuisia oivalluksia samalla erottautuen kilpailijoistaan. Tietokannat ovat tässä yhteydessä keskeisessä roolissa, sillä ne mahdollistavat tiedon tehokkaan säilytyksen, hallinnan ja analysoinnin. Teknologian kehittyessä ja datan määrän kasvaessa organisaatioiden on yhä kannattavampaa siirtyä hyödyntämään pilvipohjaisia tietokantaratkaisuja, jotka tarjoavat skaalautuvuutta, joustavuutta ja laskentatehoa vastaamaan nykypäivän liiketoiminnan vaatimuksiin.

1.1 Liiketoimintakonteksti

Tämä opinnäytetyö keskittyy yrityksen tietokantasiirtymäprojektiin, jossa siirrytään Microsoft SQL Server -tietokannasta Snowflake-tietokantaan. Kyseessä on 30 hengen organisaatio, jonka tarjoama palvelu kerää laajan kirjon tietoa eri datalähteistä. Nämä tiedot koostetaan yhteen keskitettyyn sovellukseen, jossa ne esitetään selkeästi ymmärrettävässä muodossa. Tästä syystä tiedonjalostus on keskeinen osa liiketoimintaa ja tietojärjestelmä toimii tärkeänä komponenttina toimeksiantajan infrastruktuurissa. Palvelu mahdollistaa asiakkaille nopean pääsyn olennaisiin tietoihin ja tukee päätöksentekoa tietopohjaisesti.

1.2 Kehitystavoite ja rajaus

Organisaation käsiteltävän datamäärän kasvaessa, virtuaalipalvelimella pyörivän Microsoft SQL Server -järjestelmän kyvykkyydessä mukautua uusiin vaatimuksiin ilmeni rajoitteita. Keskeisenä kehityshaasteena nousi esiin laskentatehon rajallinen skaalautuvuus, joka aiheutti pidentyviä viiveitä datan käsittelyssä. Tämä tilanne korosti tarvetta tarkastella erilaisia ratkaisuja raskaimpien laskentaprosessien nopeuttamiseksi.

Tämän opinnäytetyön tavoitteena on vastata organisaation kehittyviin tiedonhallintavaatimuksiin kohentamalla tietokannan skaalautuvuutta ja parantamalla laskentatehoa. Tämä saavutetaan siirtymällä nykyisestä Microsoft SQL Server -tietokannasta Snowflake-tietokantaan, mikä mahdollistaa nopean datan käsittelyn laajemmissa mittakaavoissa. Hankkeeseen kuuluu olennaisten tietokantakomponenttien uudelleensuunnittelu ja integrointi Snowflake-alustalle.

Tuloksena eniten laskentatehoa ja aikaa vievät laskentaprosessit on siirretty Snowflake-ympäristöön, joka mahdollistaa alhaisen viiveen organisaation keskeisimmissä palveluissa. Samalla kyky vastaanottaa suuria datamääriä kasvaa ja toimintavarmuus nousee.

Toiminnallinen osuus rajataan käsittelemään pelkästään Snowflake-tietokantaympäristön valmistelua ja siellä luotuja laskentaprosesseja. Valinta varmistaa, että työ pysyy tiiviisti toimeksiantajan uuden infrastruktuurin kehittämisen ympärillä. Vaikka työ syventyy tietokantojen toimintoihin ja niihin liittyviin haasteisiin, työssä ei käsitellä tietokantataulujen rakenteita tai liiketoiminnan arkaluonteisia tietoja. Tällä tavoin työ pysyy relevanttina ja kohdennettuna laajemmalle yleisölle, tarjoten yleiskatsauksen parhaisiin käytäntöihin tietokantasiirtymissä syventymättä liiaksi toimeksiantajan teknisiin yksityiskohtiin.

Vaikka työ esittää erilaisia dataputkia ja näiden ominaisuuksia, tietokannansiirtymä on itsessään jatkuva prosessi. Siirtymä ei pääty opinnäytetyön valmistumiseen vaan on osana pitkäkestoista kehitysprosessia, jota opinnäytetyö dokumentoi ja jolle se tarjoaa rakenteellisen viitekehyksen.

1.3 Käsitteet

API	Application Programming Interface. Rajapinta, joka mahdollistaa ohjelmistojen välisen tiedonvaihdon, määritellen kuinka ohjelmistot voivat käyttää toistensa toimintoja ja tietoja.
Back-End	Sovelluksen taustajärjestelmä, joka sisältää datan käsittelylogiikan ja vastaa sovelluksen toiminnallisuudesta.
Big Data	Valtava tietomäärä, jonka käsittely vaatii siihen erikoistuneita teknologioita.
CI/CD-prosessi	Continuous Integration and Continuous Deployment. Prosessi, jossa koodi integroidaan ja julkaistaan automaattisesti, vähentäen virheiden riskiä ja nopeuttaen kehitystä.
Dynaaminen taulu	Snowflaken ominaisuus, joka materialisoi tietokyselyn tulokset automaattisesti päivittyvään tauluun.
Front-End	Käyttäjälle näkyvä osa sovelluksesta, joka sisältää graafisen käyttöliittymän ja käyttäjätoiminnot.
Git	Hajautettu versionhallintajärjestelmä, jota käytetään koodin muutosten seurantaan ja yhteistyöhön ohjelmistoprojekteissa.

IP	Internet Protocol. Toimii perustana tietojen lähettämiselle verkossa määrittäen, kuinka paketit lähetetään ja vastaanotetaan.
LAN	Local Area Network. Verkko, joka kattaa rajatun alueen, kuten toimiston tai rakennuksen.
Liitoskomento	Yhdistää kaksi tai useampia taulua yhteisen sarakkeen perusteella tuottaen yhdistetyn taulukon.
Materialisoitu näkymä	Snowflaken ominaisuus, joka sisältää ennalta laskettuja ja tallennettuja tietoja kyselyiden nopeuttamiseksi.
MPP	Massively Parallel Processing. Mahdollistaa tietojen käsittelyn ja laskennan suorittamisen samanaikaisesti useilla prosessoreilla, parantaen suorituskykyä ja tehokkuutta.
MVP	Minimum Viable Product. Projektin varhainen versio, joka sisältää vain välttämättömimmät ominaisuudet sen toimivuuden testaamiseksi ja palautteen keräämiseksi.
ODBC-yhteys	Open Database Connectivity. Standardoitu rajapinta, joka mahdollistaa tiedonsiirron tietokantojen ja sovellusten välillä sekä tietokantojen kesken.
Palvelinsolmu	Server Node. Yksittäinen palvelin verkossa, joka osallistuu tietojen käsittelyyn ja tallennukseen osana laajempaa palvelinryhmää.
Pilvipohjainen	Teknologia, joka toimii internetin kautta etäpalvelimilla eikä vaadi paikallista infrastruktuuria.
POC	Proof of Concept. Toteutus, jolla testataan uuden idean tai teknologian toimivuus käytännössä.
PRD	Production. Tuotantoympäristö, jossa valmis sovellus tai palvelu on aktiivisessa käytössä ja saatavilla loppukäyttäjille.

QA	Quality Assurance. Prosessi tai toimenpide, jolla varmistetaan tuotteen tai palvelun laatu ennen sen julkaisua.
SAAS	Software as a Service. Palvelumalli, jossa ohjelmistot tarjotaan pilvipalveluna, mahdollistaen pääsyn sovelluksiin internetin kautta ilman paikallista asennusta.
SAN-verkko	Storage Area Network. Mahdollistaa suuren määrän tallennustilaa jaettavaksi useiden palvelinten kesken tehokkaasti ja joustavasti.
TCP	Transmission Control Protocol. Internetin tietoliikenneprotokolla, joka varmistaa tietojen luotettavan ja järjestetyn siirron virheitä korjaten.
Virtuaalinen tietovarasto	Itsenäinen laskentaresurssi datan käsittelyn suorittamiseen, joka skaalautuu tarpeen mukaisesti.
Versionhallinta	Menetelmä, jolla hallitaan ja seurataan tiedostojen tai projektien eri versioita samalla mahdollistaen muutosten palauttamisen ja yhteistyön.
Visual Studio Code	Microsoftin kehittämä avoimen lähdekoodin tekstieditori, joka tukee monia ohjelmointikieliä ja kehitystyökaluja.

2 Tietokannat ja dataputket

Tietokanta on järjestelmällisesti organisoitu tietojen kokoelma, joka on tyypillisesti tallennettu digitaalisesti. Tietokannan hallinta tapahtuu yleensä tietokannan hallintajärjestelmän (Database Management System, DBMS) avulla. Tieto, tietokannan hallintajärjestelmä sekä niihin liittyvät sovellukset muodostavat yhdessä tietokantajärjestelmän. (Oracle s.a..)

Yleisimmin data on mallinnettu tietokantaan riveinä ja sarakkeina useisiin eri taulukoihin, mikä tekee datan käsittelystä tehokasta. Näin dataan pääsee käsiksi helposti ja sitä voidaan hallinnoida, muokata, päivittää, kontrolloida ja järjestää tarpeen mukaisesti. Useimmat tietokannat käyttävät strukturoitua kyselykieltä (Structured Query Language, SQL) datan kirjoittamiseen ja kyselyjen suorittamiseen. (Oracle s.a..)

Tietokantaobjekteista näkymät, proseduurit ja funktiot ovat taulujen ohella perusrakenteita, jotka tekevät tietokannoista joustavia ja monikäyttöisiä. Näkymät ovat virtuaalisia taulukoita, jotka tarjoavat turvallisen ja rajoitetun pääsyn tietoihin, mahdollistaen datan tarkastelun tarvittavilla rajauksilla ilman alkuperäisten tietojen muuttamista. Proseduurit mahdollistavat toistuvien tehtävien suorittamisen ja monimutkaisten operaatioiden automatisoinnin tietokannassa. Funktiot ovat samankaltaisia proseduurien kanssa, mutta niitä käytetään yleensä tietojen käsittelyyn ja muokkaamiseen, palauttaen arvon tehtävän suorittamisen jälkeen (Kumar 22.10.2023). Yhdessä nämä objektit muodostavat työkalupakin, joka tukee datan monipuolista käsittelyä ja tehokasta hallintaa. (IBM documentation 2023a.)

2.1 Tietokantojen tyypit: OLAP, OLTP

Tietokantoja tarvitaan useisiin erilaisiin käyttötarkoituksiin, joka on johtanut erilaisten tietokantatyyppien luomiseen. Ymmärtääksemme paremmin näiden tietokantojen eroja ja käyttökohteita, voimme tehdä erottelun kahteen päätyyppiin, joita ovat (Online Transaction Processing, OLTP) ja (Online Analytical Processing, OLAP). (Sinha 16.3.2021.)

OLTP-tietokannat on suunniteltu hallitsemaan suuria määriä transaktioita lyhyessä ajassa. Niitä käytetään tyypillisesti päivittäisten toimintojen ja liiketoimintaprosessien tietojen käsittelyyn. Niiden vahvuus on kyvyssä tukea nopeita ja tehokkaita transaktioita. OLAP-tietokannat on suunniteltu analytiikkaan ja raportointiin, tarjoten mahdollisuuden suorittaa monimutkaisia kyselyitä suurilla datamäärillä. OLAP-tietokanta on ihanteellinen analysointiin, suunnitteluun ja päätöksenteon tukemiseen, mahdollistamalla suurten datamäärien tiedon rikastamisen. (Sinha 16.3.2021.)

2.2 Dataputket

Dataputket ovat automatisoituja prosesseja, jotka ovat oleellisia raakadatan jalostuksessa. Ne puhdistavat, yhdistävät ja muuntavat datan muotoon, joka on merkityksellistä ja hyödyllistä. Tähän prosessiin sisältyy saapuvan datan tarkistaminen virheiden ja puutteiden varalta, jonka jälkeen dataa standardisoidaan ja rikastetaan lisätiedoilla eri lähteistä. Lopulta prosessoitu data tuodaan esiin rajapinnoissa, mahdollistaen sen hyödyntämisen eri sovelluksissa. Rajapinta (Application Programming Interface, API) mahdollistaa eri järjestelmien, sovellusten ja palveluiden välisen vuorovaikutuksen ja tiedonvaihdon. (IBM Topics s.a..)

Dataputkien suunnittelu on riippuvainen siitä, kuinka nopeasti datan on oltava käsitellyssä muodossa. Vaatimusten perusteella voidaan valita putken malliksi joko eräajo (Batch) tai jatkuva virtaus (Stream) (IBM Topics s.a.). Eräajojen avulla käsitellään suuria datamääriä kerralla, mikä sopii tilanteisiin, joissa pidempi latenssi eli viive ei ole merkityksellinen. Toisaalta, kun dataa on saatava käyttöön reaaliajassa, on datavirtaus oikea valinta, tarjoten pääsyn jatkuvasti päivittyvään tietoon. Dataputkien latenssin määrittely on tasapainottelua kustannustehokkuuden, laskentakapasiteetin ja tarpeiden välillä. On oleellista hahmottaa, mikä on tärkeintä kullekin toiminnolla, koska mitä tiheämmin päivityksiä tehdään sitä enemmän tietokantajärjestelmä kuluttaa resursseja. (Leonard 2023, luku 1.)

2.3 Tietokantajärjestelmän suorituskyky

Tietokantajärjestelmän suorituskyky on avainasemassa sovellusten tehokkuuden ja käyttäjäkokemuksen kannalta. Suorituskyvyn keskeisiin mittareihin kuuluu vasteajat, jotka kertovat, kuinka nopeasti tietokanta kykenee vastaamaan kyselyihin. Erityisesti reaaliaikaisia päätöksiä vaativissa sovelluksissa lyhyet vasteajat ovat kriittisiä. (IBM Documentation 2023b.)

Skaalautuvuus on merkittävä tekijä, joka edistää lyhyitä vasteaikoja. Sen avulla tietokantajärjestelmän laskentakapasiteettia voidaan kasvattaa dynaamisesti vastaamaan kasvavan datan määrää tai käyttäjien lisääntymistä. Skaalautuvuus voi olla horisontaalia tai vertikaalia. Horisontaalisessa skaalauksessa tietokantajärjestelmän kapasiteettia kasvatetaan lisäämällä useita laskentayksiköitä jakamaan laskennan kuormitusta. Vertikaalisessa skaalauksessa puolestaan keskitytään yksittäisen laskentayksikön tehojen nostamiseen, jotta se voi käsitellä suurempia kuormia tehokkaammin. Kokonaisuudessaan skaalautuvuus varmistaa, ettei suorituskyky heikkene, kun tietokantajärjestelmän kuormitus kasvaa muuttuvissa olosuhteissa. (Timescale s.a..)

Tietokantajärjestelmän suorituskykyyn vaikuttavat olennaisesti partitiointi ja indeksointi. Indeksointi viittaa tietorakenteiden luomiseen, mikä mahdollistaa nopean pääsyn tietoihin määriteltyjen sarakeavainten, kuten pääavaimen perusteella. Pääavain toimii rivin yksilöllisenä tunnisteena. Tämä prosessi kohentaa kyselyjen suorituskykyä vähentämällä haettavan datan määrää, koska indeksoinnin avulla järjestelmä voi suoraan paikantaa halutut tiedot ilman koko tietorakenteen läpikäymistä. Partitioinnilla tarkoitetaan suurten datamäärien jaottelemaista pienempiin ja hallittavampiin osiin, joka voidaan tehdä esimerkiksi päivämäärän perusteella. Kunkin osion erillinen tallennus mahdollistaa tehokkaamman pääsyn datan alijoukkoihin, poistaen tarpeen kahlata koko datamassan lävitse, jotta yksi osio löydettäisiin. Yhdessä nämä muodostavat perustan tietokannan optimoinnille, mahdollistaen suuria datamäärien hallinnoinnin ilman merkittävää viivettä. (Dremio s.a..)

3 Microsoft SQL Server ja Snowflake

Seuraavassa luvussa syvennymme SQL Serverin ja Snowflaken arkkitehtuureihin kirjallisuuden pohjalta. Tämä luku on työn kannalta keskeinen, sillä ymmärrys nykyisen sekä uuden tietokantajärjestelmän toimintalogiikasta valottaa syitä, miksi toimeksiantaja on valinnut Snowflaken ratkaisuna kasvavan datamäärän hallinnointiin.

Ensimmäiseksi tutustutaan SQL Serverin ja Snowflaken arkkitehtuureihin sekä niiden keskeisiin ominaisuuksiin, jonka jälkeen vertaillaan näitä tietokantajärjestelmiä keskenään. Tässä osiossa keskitytään kummankin järjestelmän vahvuuksiin ja erityispiirteisiin, sekä arvioidaan, minkä tyyppisiin käyttötarkoituksiin kumpikin järjestelmä soveltuu parhaiten. Tämän vertailun avulla pyritään tarjoamaan yleiskuvan siitä, miten kumpikin tietokantaratkaisu voi palvella organisaation tarpeita erilaisissa datan käsittelyn ja hallinnan skenaarioissa.

3.1 SQL Server arkkitehtuuri ja toiminta

Tässä luvussa keskitytään Microsoft SQL Serveriin, joka on ollut toimeksiantajan tärkein tietokantajärjestelmä ennen tietokantasiirtymää. SQL Serverin tarkastelu on tärkeää, sillä se luo pohjan ymmärrykselle siitä, mistä lähdetään liikkeelle tietokantasiirtymässä. Alaluvuissa käydään läpi SQL Serverin arkkitehtuurin toimintaa, historiaa ja siihen liitettäviä lisäpalveluita.

SQL Server on Microsoftin kehittämä relaatiotietokannan hallintajärjestelmä (Relational Database Management System, RDBMS). Työkalu tarjoaa ratkaisun tietojen säilyttämiseen, hallintaan ja analysointiin niin pienille kuin suurille organisaatioille. Se on ansainnut merkittävän aseman RDBMS-markkinoilla pitkän historian, luotettavuuden ja tehokkuuden ansioista. Vuonna 1989 markkinoille tuodusta ensimmäisestä versiostaan lähtien, SQL Server on läpikäynyt useita teknologisia uudistuksia, mikä on osoitus yrityksen sitoutumisesta tietokantateknologian kehitykseen. (McQuillan 2015.)

SQL Serverin arkkitehtuuri nojaa kolmeen keskeiseen kerrokseen, joita ovat protokolla-, relaatio- ja tallennuskerros. Protokollakerros varmistaa käyttäjien ja tietokantapalvelimen välisen sujuvan kommunikaation. Relaatiokerros puolestaan muodostaa järjestelmän ytimen, jossa SQL-kyselyt käsitellään. Tallennuskerros huolehtii datan turvallisesta ja tehokkaasta säilytyksestä. Näiden kerrosten keskiössä on Database Engine, joka on avainasemassa datan tallennuksen, käsittelyn ja turvallisuuden kannalta. Database Engine hyödyntää Microsoftin laajentamaa versiota SQL-kielestä nimeltä Transact-SQL (T-SQL) joka tuo lisäominaisuuksia tiedonkäsittelyyn. (Terra 2023.)

SQL Server tarjoaa myös erilaisia työkaluja ja palveluita, kuten SQL Server Integration Services (SSIS) datan integrointiin, SQL Server Analysis Services (SSAS) tietojen analysointiin ja SQL

Server Reporting Services (SSRS) raportointiin. Nämä palvelut tekevät SQL Serveristä monipuolisen alustan, joka tukee kattavaa tiedonhallintaa aina datan keräämisestä ja säilyttämisestä sen analysointiin, raportointiin ja tietojen jakamiseen. (Terra 2023.)

SQL Server on ennen ollut pelkästään on-premises-järjestelmä, jolloin se vaati käyttäjältä omaa fyysistä palvelinta. Kuitenkin SQL Server on laajentanut ominaisuuksiaan pilvipalveluiden suuntaan tarjoten versioita, jotka on rakennettu muun muassa toimimaan Microsoft Azure -pilvialustalla. Tämä mahdollistaa organisaatioille mahdollisuuden valita paikallisen, pilvipohjaisen tai hybriditietokannan välillä. (Azure Migrate Documentation 2024.)

3.1.1 SQL Serverin arkkitehtuurin perusteet

SQL Serverin arkkitehtuuri koostuu kolmesta kerroksesta (Layer), jotka mahdollistavat datan käsittelyn joustavasti ja skaalautuvasti. Tässä alaluvussa tarkastellaan edellisessä luvussa mainittujen kolmen kerroksen toimintaa tarkemmalla tasolla.

Ensimmäinen osa tätä kokonaisuutta on protokollakerros (Protocol Layer), joka toimii viestinviejänä asiakasohjelmien ja tietokantapalvelimen välillä. Tämä kerros käyttää kolmea pääasiallista kommunikaatiomenetelmää, jotka on suunniteltu toimimaan erilaisissa ympäristöissä varmistaen tietokannan suorituskyvyn sekä yksittäisillä koneilla että verkoston osana. Jaettua muistia (Shared Memory) käytetään, kun asiakasohjelma ja SQL Server toimivat samalla laitteella. Kommunikointi tapahtuu jaetun muistialueen kautta, mikä mahdollistaa nopean datanvaihdon ilman verkon viiveitä. TCP/IP-protokolla (Transmission Control Protocol/ Internet Protocol) mahdollistaa asiakasohjelman ja SQL Serverin välisen kommunikoinnin, vaikka ne sijaitsisivat fyysisesti eri koneilla tai verkkoalueilla. Tämä on yleisin tapa yhdistää kaukaiset järjestelmät toisiinsa ja tarjoaa luotettavan ja standardoidun tavan datan siirtämiseen. Nimetyt putket (Named Pipes) on suunniteltu lähiverkon (Local Area Network, LAN) sisäiseen kommunikointiin. Se mahdollistaa tehokkaan datan siirron paikallisissa verkostoissa käyttäen erityistä protokollaa, joka on optimoitu lyhyiden etäisyyksien dataliikenteeseen. (Terra 2023.)

Toinen komponentti on nimeltään Relational Engine, joka tunnetaan myös kyselyprosessorina. Sen tehtävänä on määrittää, mitä kysely pyrkii tekemään ja kuinka tämä voidaan parhaiten toteuttaa. Kyselyprosessori koostuu kolmesta pääkomponentista, joita ovat kyselyn jäsentäjä (CMD Parser), Optimoija (Optimizer) ja Kyselyn ajaja (Query Executor), jotka yhdessä varmistavat käyttäjien kyselyjen tehokkaan ja tarkoituksenmukaisen suorituksen. Kyselyn jäsentäjä aloittaa prosessin tarkistamalla kyselyn pätevyyden ja mahdolliset syntaksivirheet. Tämän jälkeen se luo kyselystä kyselypuun (Query Tree), joka on kyselyn rakenteellinen esitys. Jäsentäjä on kyselyprosessorin ensimmäinen komponentti, joka toimii ensimmäisenä suodattimena ja varmistaa kyselyn

virheettömyyden ennen sen etenemistä. Optimoija käyttää algoritmeja parantaakseen kyselyn suoritusaikaa. Sen tavoitteena on löytää niin sanotusti halvin suorituspolku kyselyn toteuttamiseksi. Eli komponentti tavoittelee resurssien käytön minimointia, eikä välttämättä nopeinta suoritusaikaa. Optimoija luo suorituspolun, joka määrittelee, miten dataa haetaan ja käsitellään. Kun optimoija on määrittänyt suorituspolun, kyselyn ajaja ottaa vastuun. Se luo juuri määritetyn suorituspolun ja vastaanottaa kyselystä syntyneen tuloksen tallennusmoottori-komponentilta (Storage Engine). Ajaja julkaisee tulokset protokollakerrokseen, josta tulokset lähetetään lopulta käyttäjälle. (Terra 2023.)

Tallennusmoottori vastaa datan tallentamisesta säilytysjärjestelmään, kuten SAN-verkkoon (Storage Area Network) tai kovalevylle ja hakee sitä tarvittaessa. Komponentti käyttää kolmea erilaista tiedostotyyppiä datan hallintaan, joita ovat primääritiedostot, sekundääritiedostot ja lokitiedostot. Primääritiedosto sisältää tietokannan datan sekä järjestelmän metatiedot. Sekundääritiedostot ovat vapaaehtoisesti käyttäjän määrittelemiä ja lokitiedostot ylläpitävät kaikkia tietokantatapahtumia mahdollistaen palauttamisen ja datan eheyden säilymisen. (Terra 2023.)

Tallennusmoottori käyttää seuraavia komponentteja, joiden avulla datan säilöminen, hallinta ja nouto mahdollistetaan:

- **Access Method**: Välittää tiedon kyselynsuorittajan ja puskurihallinnan (Buffer Manager) välillä varmistaen sujuvan tiedonsiirron
- **Buffer Manager**: Hallinnoi datan väliaikaista tallennusta, nopeuttaen datan käsittelyä ja kyselyjen suoritusta. Se koostuu kolmesta alamoduulista, joita ovat:
    - **Plan Cache**: Säilyttää valmiita suoritussuunnitelmia kyselyjen nopeuttamiseksi
    - **Data Parsing**: Valmistelee ja jäsentää datan kyselyjä varten
    - **Dirty Pages**: Sisältää muutokset, jotka odottavat tietokantaan kirjoittamista
- **Transaction Manager**: Hallitsee transaktioita, käyttäen lukitus- ja lokimekanismeja varmistaakseen transaktioiden eheyden

3.1.2 SQL Serverin kehitys ja versiot

Microsoft SQL Serverin kehityskaari alkoi vuonna 1989. Haastaakseen Oraclea, joka dominoi 1980-luvun tietokantamarkkinoita, Microsoft aloitti yhteistyön Sybase-nimisen yrityksen kanssa kehittääkseen oman tietokantajärjestelmänsä. Tämä yhteistyö jatkui aina 1990-luvun puoliväliin asti, jolloin Microsoft ja Sybase erkanivat omille poluilleen. Tuosta hetkestä alkaen Microsoft omaksui täysin uuden suunnan SQL Serverin kehittämisessä, mikä huipentui uudistetun version 7.0 julkaisuun vuonna 1998. Järjestelmän edistyksellisyyden lisäksi julkaisu edustaa Microsoftin sitoutumista tietokantateknologian itsenäiseen kehitykseen. (Introduction to SQL Server 2015.)

SQL Server on kasvanut tarjoamaan useita versioita järjestelmästä, jotka palvelevat erilaisia käyttötarpeita. Enterprise-versio on suunniteltu suurille yrityksille ja tarjoaa korkeaa suorituskykyä, turvallisuutta ja skaalautuvuutta. Business Intelligence -versio tukee BI-ratkaisuja ja tiedon analysointia, kun taas Standard-versio palvelee yleisimmin tuotantoympäristöjä tarjoten vankan tiedonhallinnan. Web-versio tukee verkkosivustoja ja Developer-versio on identtinen Enterprise-version kanssa, mutta tarkoitettu kehitys- ja testauskäyttöön. Express-versio on ilmainen ja soveltuu pienimuotoisiin sovelluksiin ja kehitystyöhön. (McQuillan 2015.)

3.1.3 SQL Serverin palvelut

Microsoft SQL Server tietokantapalvelimen monipuoliset palvelut tarjoavat vankan perustan nykyaikaiselle tiedonhallinnalle. Sen ytimessä on Database Engine, joka vastaa datan tallennuksesta, suojaamisesta ja nopeista transaktioprosesseista. Tämän tukena on palveluita, jotka mahdollistavat esimerkiksi prosessien automatisoinnin, analytiikkaratkaisuja ja joustavuutta verkkoarkkitehtuuriin. Arkkitehtuurin peruskomponentit ja tukena toimivat palvelut luovat kattavan tietokantajärjestelmän. (Terra 2023.)

SQL Serverin tarjoama automaatiotyökalu on yksi sen vahvuuksista. SQL Server Agent toimii tehtävien aikatauluttajana mahdollistaen ajoitettujen toimintojen, kuten datan varmuuskopioinnin, siirron ja muiden rutiininomaisten ylläpitotoimien automatisoinnin. Tämä vähentää manuaalisen työn tarvetta ja parantaa järjestelmän tehokkuutta. (Terra 2023.)

Yhteydenpidossa ja verkkoarkkitehtuurin hallinnassa SQL Server Browser on oleellinen varsinkin moni-instanssiympäristöissä. Se kuuntelee saapuvia yhteyspyyntöjä ja ohjaa ne oikeille SQL Server -instansseille parantaen tietokanta-arkkitehtuurin joustavuutta ja saatavuutta. (Terra 2023.)

SQL Server Analysis Services (SSAS) mahdollistaa laajojen datamäärien tehokkaan analysoinnin ja koneoppimissovellusten kehittämisen. Yhdistämällä SSAS-palvelun R- ja Python-ohjelmointikielien kanssa organisaatiot voivat luoda analytiikkaratkaisuja, jotka tukevat monipuolista tiedonkeruuta ja syventävät ymmärrystä liiketoiminnan prosesseista. SQL Server Reporting Services (SSRS) puolestaan tarjoaa kattavat raportointiominaisuudet ja päätöksenteon tukityökalut. SQL Server Integration Services (SSIS) täydentää kokonaisuutta tarjoamalla datan muunnos- ja latauspalveluita, jotka muuttavat raakadatan hyödylliseksi informaatioksi. (Terra 2023.)

3.2 Snowflake arkkitehtuuri ja toiminta

Tässä luvussa syvennytään Snowflaken arkkitehtuuriin ja luvun tavoitteena antaa lukijalle monipuolinen ymmärrys sen toimintaperiaatteista. Tiedon avulla saadaan selkeämpi kuva Snowflaken ydinkomponenteista. Alaluvuissa käsitellään kyseisen tietokantajärjestelmän pilvipohjaisuutta, suorituskykyä, skaalautuvuutta, tallennuskykyä, kustannuksia ja hallinnointia. Näiden läpikäynti valaisee, miten Snowflake eroaa muista tietokantajärjestelmistä.

3.2.1 Pilvipohjainen arkkitehtuuri ja ydinkomponentit

Viime vuosina Snowflake on noussut esiin merkittävänä toimijana tietokantajärjestelmien kentällä. Esimerkiksi, kun Snowflake esitteli pilvipohjaisen Software as a Service (SaaS) -mallinsa muutkin palveluntarjoajat alkoivat omaksumaan samankaltaisia lähestymistapoja. Snowflake hyödyntää useita julkisia pilvipalveluita, jotka antavat työkalut suurien datamäärien käsittelyyn ja skaalaamiseen. (Bell 2022, luku 1.)

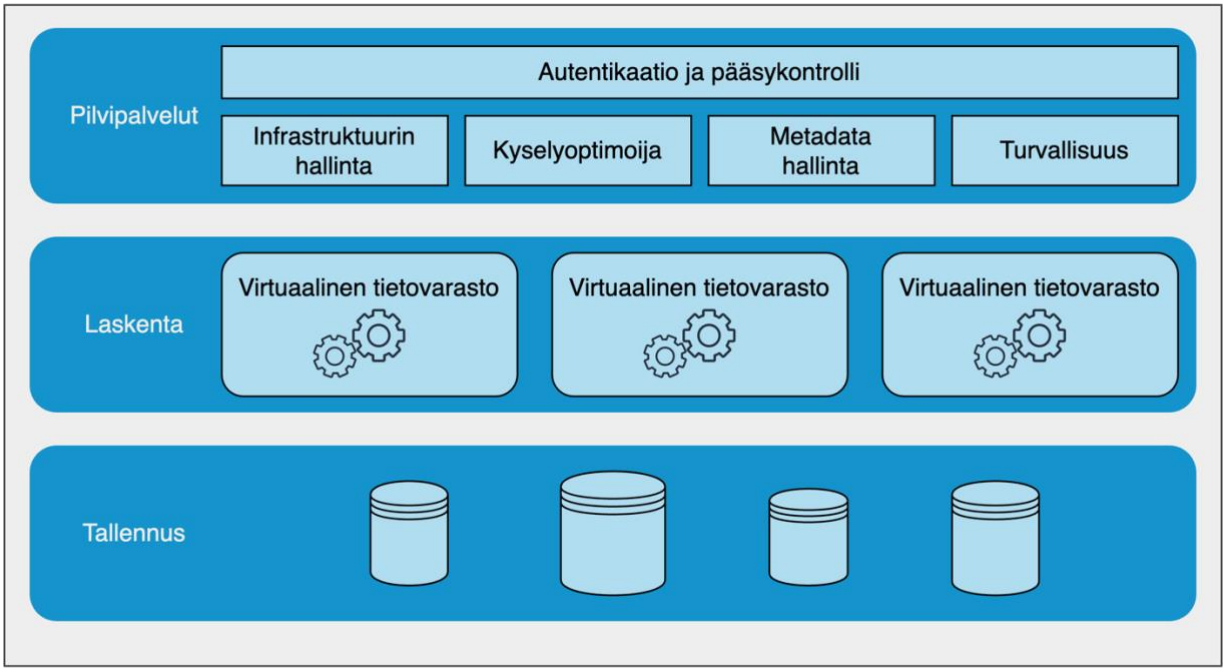
Snowflaken perustajat tunnistivat vuonna 2012, että Big Datan skaalaus oli edelleen merkittävä haaste. Tämän havainnon siivittämänä he lähtivät luomaan uudenlaista tietokantajärjestelmää, jonka perustana on pilviarkkitehtuuri. Kyseisessä lähestymistavassa koko tietokanta-alusta rakennettiin toimimaan pilvipohjaisesti, joka loi perustan merkittäville uudistuksille niin tietokantojen nopeudessa, skaalautuvuudessa kuin käyttäjäystävällisyydessäkin. Pilvipalveluiden tuomien etujen ansiosta asiakkaat pystyvät ylittämään kustannukset, jotka liittyvät siirtymiseen perinteisistä tietokantaratkaisuista Snowflakeen. (Bell 2022, luku 1.)

Snowflake oli ensimmäinen tietokanta-alusta, joka suunniteltiin ja rakennettiin alusta alkaen toimimaan pilvessä. Tämä lähtökohta antoi Snowflakelle mahdollisuuden kehittää teknologiaansa ilman perinteisten paikallisten RDBMS-järjestelmien rajoituksia. Sen SQL-tietokantamoottorin automaattinen skaalaus erottuu merkittävästi perinteisistä RDBMS-järjestelmistä, sillä se perustuu rinnakkaiskäsittelyyn (Massively Parallel Processing, MPP). Tämä teknologinen innovaatio erottaa Snowflaken kilpailijoistaan, kuten RedShiftistä, BigQuerystä ja Databricksista, jotka ovat mukailleet perinteisiä paikallisia ratkaisuja pilvipohjaiseen käyttöön. (Bell 2022, luku 1.)

Snowflaken arkkitehtuuri on hybridi, joka yhdistää Shared-Disk- ja jaetun Shared-Nothing-rakenteet. Tällä lähestymistavalla Snowflake on erottunut merkittävästi RDBMS-kentällä, hyödyntäen molempien arkkitehtuurien vahvuuksia, tarjoten joustavuutta ja tehokkuutta. Ennen Snowflakea RDBMS-arkkitehtuurin pääsuuntauksina olivat joko Shared-Nothing tai Shared-Disk-mallit. Shared-Nothing -mallissa data jaetaan ja prosessoidaan erillisillä palvelinsolmuilla (Server Nodes), joka tarkoittaa, että solmut eivät jaa tallennustilaa tai muistia keskenään. Shared-Disk-

mallissa taas data on kaikkien solmujen saatavilla, eli ne käyttävät yhteistä tallennustilaa. (Avila 2022, luku 2.)

Snowflaken teknologia perustuu kolmikerroksiseen arkkitehtuuriin, johon kuuluu pilvipalvelut (Cloud Services), laskenta (Compute) ja tallennus (Storage). Kerrosrakennetta esitellään kuvassa 1. Jokainen kerros on erityisesti suunniteltu tarjoamaan tiettyjä toimintoja, jotka yhdessä muodostavat tehokkaan ja joustavan tietokantaympäristön. (Avila 2022, luku 2.)



Kuva 1: Snowflaken kolmikerroksinen arkkitehtuuri (Snowflake documentation s.a. a)

Pilvipalvelukerros toimii Snowflaken aivoina, joka tarjoaa valikoiman palveluita, kuten optimointi-, hallinta-, transaktio-, tietoturva- sekä metatiedon hallintapalveluita. Tämä kerros on avainasemassa tietojen jakamisen ja yhteistyön hallinnassa, mahdollistaen keskitetyn tietoturvan ja datan hallinnan. Sen ansiosta Snowflake tarjoaa läpinäkyvyyttä kyselyhistoriaan ja hallitsee tehokkaasti metatietoa, kyselyn optimointia sekä tietojen jakamispalveluita. (Avila 2022, luku 2.)

Laskentakerros on täysin erillään pilvipalvelu- ja tallennuskerroksesta. Tämä kerros käyttää virtuaalisia varastoja eli MPP-laskentaklustereita (Bell 2022, luku 1). Tämän ansioista eri kokoiset kyselyjä käsittelevät tietovarastot (Warehouse) pystyvät toimimaan itsenäisesti ja yhtäaikaisesti. Yksi virtuaalinen varasto voi esimerkiksi keskittyä datan lataamiseen, kun toinen suorittaa kyselyjä. Laskentakerros mahdollistaa samanaikaisen pääsyn dataan ilman pullonkauloja tai konflikteja, jotka ovat yleisiä perinteisissä tietokannoissa. (Avila 2022, luku 2.)

Tallennuskerros on erotettu laskentakerroksesta, joka eroaa ratkaisuna perinteisistä tietokantajärjestelmistä, joissa laskenta ja tallennus ovat toisiinsa sidottuja. Tämä erottelu mahdollistaa tarpeisiin mukautuvan infrastruktuurin hallinnan, joka voi ilmetä kustannustehokkaana ratkaisuna, koska asiakas maksaa käytön perusteella. (Bell 2022, luku 1.)

Tallennuskerros hyödyntää älykkäästi valitun pilvipalveluntarjoajan natiivia Blob-tallennusta (AWS:ssä S3, Azuressa Azure Blob ja Googlessa Google Cloud Storage), käyttäen sarakepohjaista tietokanta-arkkitehtuuria, jossa raakatiedostot ovat pakattuja ja salattuja pilvipalvelussa. Tämä datan pakkaus hyödyntää mikropartitioita. Optimoitu sarakepohjainen varastointi mahdollistaa sekunnin murto-osan vastausajat ja tehokkaan tiivistyksen. Optimointi johtaa 3–5 kertaiseen datakompression nopeuteen verrattaessa perinteisiin tietokantajärjestelmiin. (Bell 2022, luku 1.)

Perinteiset RDBMS-järjestelmät vaativat usein Snowflakea enemmän manuaalista työtä ja osaamista. Snowflake skaalaa ja hallinnoi dataa automaattisesti ja erottuu näin kilpailijoistaan. Se hyödyntää metatietoja luodakseen itseindeksoivan ja itseoptimoivan tietokantajärjestelmän. Tämä innovaatio on muuttanut perusteellisesti ylläpidon kustannusrakennetta organisaatioissa vähentämällä monimutkaisuutta ja tarvetta tietokantojen manuaaliselle ylläpidolle. (Bell 2022, luku 1.)

3.2.2 Tiedon jakaminen ja hallinta

Snowflaken tarjoama ratkaisu tiedon jakamiselle ja hallinnalle muuttaa tapaa, jolla organisaatiot käsittelevät ja jakavat tietoaan. Paikallisten tietokantojen datan yhdistäminen eri lähteistä on usein hallinnoinnin ja suorituskyvyn näkökulmasta haasteellista. Prosessissa joutuu usein kopioimaan ja siirtämään dataa tarpeettomasti. Snowflake mahdollistaa datan kyselyn ja analysoinnin useiden tietokantojen välillä. Tämä poistaa monia aikaisempia esteitä, jotka liittyivät datan hallintaan. Ratkaisu yksinkertaistaa tiedon hallintaa ja analysointia, joka vähentää tarvetta datan kopiointiin ja siirtoon. Tämän hyöty konkretisoituu yrityksissä, joilla on monimutkaisia ja laajoja tietokantoja. (Bell 2022, luku 1.)

Snowflake-tilien välinen datan siirto on tehtävissä eri alueiden ja pilvipalveluntarjoajien välillä. Tämä ominaisuus mahdollistaa luotettavien ja tehokkaiden dataputkien luomisen eri Snowflake tilien välille. Tietoa voi siirtää tililtä toiselle riiputtamatta pilvipalveluntarjoajasta, alueesta tai organisaatiosta. (Bell 2022, luku 1.)

Snowflaken Time Travel -ominaisuuden avulla käyttäjät voivat palata tietokantansa aiempaan versioon valitsemallaan ajankohdalla. Tämä on hyödyllistä virheiden korjaamisessa, sillä perinteisissä RDBMS-järjestelmissä virheen korjaaminen saattaa vaatia varmuuskopion palauttamisen ja vian korjaamisen, mikä vie usein tunteja tai päiviä. Snowflakessa käyttäjät voivat sen sijaan korjata virheitään muutamassa minuutissa. (Avila 2022, luku 2.)

Zero Copy Cloningin ominaisuudella organisaatiot voivat joustavasti ja nopeasti kloonata dataa ja suorittaa sille kattavia testejä ilman riskiä alkuperäisen datan eheyden heikkenemisestä. Ominaisuuden ansioista voidaan reagoida ketterästi liiketoiminnan tarpeisiin, päivittämällä, testaamalla ja käyttämällä tuotantodatan kopioita ilman suorituskyvyn kompromisseja tai liiallisia kustannuksia. (Bell 2022, luku 1.)

3.2.3 Suorituskyky ja optimointiominaisuudet

Suorituskyky ja sen optimointimahdollisuudet ovat Snowflaken vahvuuksia. Mikropartitiot ovat yksi näistä Snowflaken keskeisistä arkkitehtuurisista konsepteista, jotka mahdollistavat suurten tietomäärien käsittelyn tehokkaasti. Snowflake automaattisesti jakaa ja ryhmittelee taulujen rivit tiivistettyihin mikropartitioihin, joiden koko on 50–500 megatavua (MB). Mikropartitiot ovat muuttumattomia tiedostoja, jotka jaetaan automaattisesti vastaanottojärjestyksen mukaan, ellei auto-klusterointia ole määritelty toisin. Mikropartitioiden klusterointi mahdollistaa datan tehokkaan karsimisen (Pruning) ja tasaisen jakelun solmujen kesken. Tämä on osa arkkitehtuuria, jonka ansiosta Snowflake pystyy käsittelemään minkä kokoluokan datasetteja tahansa, jopa petatavuja (B). (Bell 2022, luku 3.)

Pruning on kyselyoptimointi-prosessi, joka karsii tarpeettomat mikropartitiot, vähentäen samalla I/O-kuormaa (Input/Output), laskentatehoa ja kyselyn suorittamiseen vaadittavaa kokonaistyötä. Tämä nopeuttaa kyselyjä, jotka tarvitsevat pääsyn vain pieneen osaan partioista. Snowflaken jatkuvasti päivittyvä metadata mahdollistaa tarkasti karsitut kyselyt. Mikropartioiden metatietojen ansioista kyselyt ovat optimoituja. (Bell 2022, luku 3.)

Snowflake käyttää klusteriavaimia ja automatisoitua klusterointia datan järjestämiseen. Tämä järjestely tapahtuu luontaisesti aikapohjaisissa datatyypeissä, kuten päivämäärissä ja aikaleimoissa, mutta se voi vaihdella riippuen datan tyypistä ja taulukon järjestyksestä. Automaattinen uudelleenklusterointi helpottaa ylläpitoa ja järjestystä, vaikka se tuo mukanaan myös kustannuksia. (Bell 2022, luku 3.)

Reklusterointi optimoi mikropartitiot klusteriavainten perusteella, jotta metadata ja partitiot ovat valmiita karsintaa varten. Tämä prosessi on täysin automatisoitu ja parantaa työkuormien historian perusteella suorituskykyä. (Bell 2022, luku 3.)

Lisäksi Snowflaken välimuisti (Cache) säästää kyselytuloksia 24 tunnin ajaksi mahdollistaen toistuvien kyselyjen nopean suorituksen. Tämä ominaisuus vähentää päällekkäisiä kyselyjä ja siihen liittyviä kustannuksia. (Avila 2022, luku 2.)

3.2.4 Versiot ja pilvipalveluntarjoajat

Snowflake-tiliä luodessa on tärkeää ymmärtää kolme keskeistä valintaa, joita ovat tilin versio, pilvipalvelujen tarjoaja ja pilvipalvelun sijaintialue. Tämä prosessi vaatii huolellista suunnittelua ja konfiguraatiota. Jokainen Snowflake-tili sijaitsee tietyllä pilvipalvelun tarjoajalla ja alueella, mikä vaikuttaa tietojen siirtoon, jakamiseen ja käytettävyyteen organisaation sisällä. Snowflaken tarjoama replikointi ja datan jakaminen mahdollistavat tietojen turvallisen jakamisen eri alueiden ja pilvipalvelutarjoajien välillä. (Snowflake documentation s.a. b.)

Snowflaken tarjoamat neljä versiota eroavat toisistaan kustannusten, käytettävissä olevien ominaisuuksien ja tietoturvan tason mukaan. Standard Edition on edullisin vaihtoehto, joka soveltuu monille yrityksille tarjoten helppokäyttöisyyttä, skaalautuvuutta ja suorituskykyä. Enterprise Edition lisää ominaisuuksia, kuten 90 päivän Time Travel -toiminnon, moniklusterivarastot ja muita yritystason palveluita. Business Critical Edition tarjoaa lisäsuojauksia, kuten asiakkaan hallinnoimat salausavaimet ja tietojen suojauksen, kun taas Virtual Private Snowflake Edition tarjoaa korkeimman tietoturvatason eristetyllä ympäristöllä ja metatietovarastolla. (Bell 2022, luku 2.)

Vaikka Snowflaken käyttöliittymä ja suorituskyky ovat samanlaiset riippumatta valitusta pilvipalveluntarjoajasta, on AWS ollut suosituin valinta sen kypsyyden ja laajamittaisen käytön vuoksi. Tämä johtuu siitä, että Snowflaken alkuperäinen versio kehitettiin AWS:n pohjalta. (Bell 2022, luku 2.)

3.2.5 Selainpohjaisuus

Snowflakella on Data Software as a Service (DSaaS)-malli, joka toimii osana sen pilvipohjaista tietoalustaa. Tämä tarkoittaa, että käyttäjien ei tarvitse asentaa, konfiguroida tai hallinnoida mitään ohjelmistoja tai laitteistoja. Snowflake huolehtii tietokannan jatkuvasta ylläpidosta sekä siihen liittyvistä kysely-, turvallisuus-, ja hallintapalveluista. Tämä on muuttanut näkemystä siitä, miten tietotekniikan infrastruktuuria ylläpidetään, sillä jatkuvat laitteisto- ja ohjelmistopäivitykset eivät enää ole tarpeen. Koska Snowflake toimii täysin pilvessä, kaikki sen DSaaS-komponentit pyörivät kunkin pilvipalveluntarjoajan infrastruktuurissa, lukuun ottamatta valinnaisia liittimiä ja ajureita. (Bell 2022, luku 2.)

3.3 Vertailu: SQL Server, Snowflake

Tietokantajärjestelmän valinta on yksi keskeisimmistä päätöksistä, joka vaikuttaa organisaation tiedonkäsittelyn suorituskykyyn, joustavuuteen ja kustannustehokkuuteen. Nykyaikaisessa liiketoimintaympäristössä, jossa datan määrä kasvaa eksponentiaalisesti sen käsittelyn nopeus on huomioitava tietokantaratkaisua valittaessa. On tärkeää valita alusta, joka ei ainoastaan vastaa nykyisiä tarpeita, mutta myös mukautuu tulevaisuuden vaatimuksiin. SQL Server ja Snowflake ovat molemmat suosittuja tietokantajärjestelmiä, joiden arkkitehtuurisilla lähestymistavoilla on merkittäviä eroja. Näiden erojen ymmärtäminen on avainasemassa, jos organisaatio on tekemässä valintaa näiden kahden ratkaisun välillä. (Mukhtiar 6.4.2023.)

SQL Server on perinteinen ja luotettava tietokantajärjestelmä, joka on toiminut vuosikymmenien ajan monien yritysten tiedonhallinnan selkärankana. Järjestelmä tarjoaa vankan pohjan transaktioiden käsittelylle ja monimutkaisille kyselyille. Sen kyky toimia paikallisesti tarjoaa tietyt turvallisuus- ja suorituskykyedut, mutta samalla asettaa haasteita muun muassa skaalautuvuuden osalta. (Araujo 21.3.2023.)

Snowflaken pilvipohjainen tietokantaratkaisu muuttaa tapaa, jolla organisaatiot käsittelevät ja analysoivat tietoaan. Sen arkkitehtuuri ja pilvipohjainen luonne mahdollistavat uudenlaista skaalautuvuutta ja joustavuutta, mikä tekee siitä varteenotettavan vaihtoehdon suurille datamäärille, jotka vaativat merkittäviä laskentaresursseja. (Influxdata s.a..)

Tietokantajärjestelmää valittaessa on tärkeää ottaa huomioon useita tekijöitä, joihin kuuluu alustan suorituskyky, luotettavuus, skaalautuvuus, joustavuus, käytön helppous, intuitiivisuus sekä kustannustehokkuus. Näiden tekijöiden ymmärtäminen ja arviointi suhteessa organisaation tarpeisiin on kriittistä optimaalisen ratkaisun löytämiseksi. (Araujo 21.3.2023.)

3.3.1 Arkkitehtuuriset erot

SQL Server on rakennettu tarjoamaan erinomaista suorituskykyä ja luotettavuutta vakio-olosuhteissa. Järjestelmä on ihanteellinen organisaatioille, jotka vaativat korkeaa transaktioiden suorituskykyä paikallisessa infrastruktuurissa. SQL Serverissä voi ilmetä puutteita, jos organisaatio kohtaa nopeaa kasvua ja samalla suurten datamäärien äkillisiä piikkejä. Tämä johtaa usein suorituskyvyn heikkenemiseen ja laitteiston lisäinvestoinnin tarpeeseen. (Araujo 21.3.2023.)

Snowflaken pilvipohjainen tietokantapalvelu erottaa laskennan tallennuksesta, mikä mahdollistaa lähes rajattoman skaalautuvuuden. Täten datamäärän äkillinen piikki on ratkaistavissa mukauttamalla resurssit käyttötarpeen mukaiseksi. Arkkitehtuurin avulla pienikin organisaatio voi päästä käsiksi suureen laskentatehoon hetkellisesti, ilman pitkäaikaisia investointeja tai

kustannuksia. Tämä tekee Snowflakesta erityisen houkuttelevan valinnan organisaatioille, jotka käsittelevät suuria datamääriä tai tavoittelevat nopeaa kasvua. (Araujo 21.3.2023.)

SQL Server tarjoaa kattavasti kustomointimahdollisuuksia, jotka sopivat organisaatioille, joilla on tarve tiukempaan ympäristön hallintaan. On-Premises ratkaisu mahdollistaa suurempaa kontrollia esimerkiksi turvallisuudesta ja suorituskyvyn optimoinnista. Korkea mukauttamisen aste vaatii kuitenkin enemmän resursseja sekä ylläpidon että päivitysten osalta, mikä voi lisätä kokonaiskustannuksia. (Araujo 21.3.2023.)

Snowflake tarjoaa erilaisen lähestymistavan painottaen käyttöönoton nopeutta ja helppoutta. Palvelu minimoi paikallisen ylläpidon tarpeen automatisoimalla monet rutiininomaiset ylläpitotehtävät kuten päivittämisen. Vaikka tämä vähentää organisaation hallinnollista taakkaa, se voi myös rajoittaa mukauttamisen syvyyttä verrattuna perinteisiin paikallisiin järjestelmiin. (Araujo 21.3.2023.)

SQL Serverin kustannukset voivat näyttää edullisemmalta erityisesti organisaatioille, joilla on jo käyttöön tarvittava infrastruktuuri asennettuna. Kuitenkin pitkäaikaiset ylläpito- ja päivityskustannukset voivat merkittävästi kasvattaa kokonaiskustannuksia, varsinkin jos datan määrä tai käsittelyvaatimukset kasvavat. (Araujo 21.3.2023.)

Snowflaken Pay as You Go -malliin perustuva hinnoittelu on kasvattanut suosiota antamalla organisaatioille kyvyn vaikuttaa kustannuksiin (Snowflake s.a.). Kuitenkin itse laskentatehon hinta on usein kalliimpi pilvipalveluissa kuin On-Premises mallissa. Snowflaken ja SQL Serverin kustannustehokkuuden vertailu on siis haastavaa ilman tapauskohtaista tarkastelua. (Araujo 21.3.2023.)

3.3.2 Yhteenveto

Kun vertaillaan SQL Serverin ja Snowflaken tarjoamia tietokantaratkaisuja, on selvää, että kumpikin tarjoaa ainutlaatuisia etuja ja kohtaa erilaisia haasteita. Se, mikä etu tai haaste on valinnan kannalta merkittävä tekijä, määräytyy pitkälti organisaation käyttötapauksien mukaan. Usein valinnan kannalta merkittävimpiä tarkastelun kohteita ovat käyttötarkoitus, datan määrä, kasvutavoitteet ja kustomointitarpeet. (Araujo 21.3.2023.)

SQL Server saattaa olla parempi valinta organisaatioille, jotka tarvitsevat vankkaa transaktioiden käsittelykykyä, kohtaavat pienempiä datamääriä tai joiden on täytettävä tiukat turvallisuusvaatimukset paikallisessa hosting-ympäristössä. Organisaatiolla tulisi olla myös resursseja tai asiantuntemusta, jonka avulla syntyy valmius ylläpitää tätä tietokantainfrastruktuuria.

Näiden täyttyessä SQL Server on kustannustehokas, luotettava ja pitkäaikainen valinta tietokantatarpeille. (Araujo 21.3.2023.)

Snowflake tarjoaa etuja organisaatioille, jotka käsittelevät suuria datamääriä tai joiden datan käsittelyvaatimukset ovat nopeasti muuttuvia. Sen pilvipohjainen arkkitehtuuri on myös toimiva suurille koneoppimisprojekteille ja nopeasti kasvaville yrityksille. Snowflake mahdollistaa organisaatioiden sopeutumisen muuttuviin datan käsittelytarpeisiin ilman merkittäviä etukäteen tehtyjä investointeja tai pitkäaikaista sitoumista, mikä tekee siitä erinomaisen valinnan joustavuutta vaativiin ympäristöihin. (Araujo 21.3.2023.)

4 Tietokantasiirtymän hallinta

Tässä luvussa käsitellään tietokantajärjestelmän elinkaarta ja sen osana olevaa tietokantasiirtymä prosessia. Tämä auttaa tunnistamaan merkkejä tietokannan modernisoinnin tarpeesta ja erilaisista toteutustasoista sekä tavoista. Luku käy läpi, miten tietokantajärjestelmän uudistaminen voidaan toteuttaa onnistuneesti, samalla erilaisia riskejä huomioiden. Loppupuolella käsitellään, kuinka uutta järjestelmää ylläpidetään tehokkaasti, varmistaen sen pitkäikäisyyden ja toimivuuden. Kokonaisuudessaan teksti pyrkii tarjoamaan monipuolisen ymmärryksen tietokantasiirtymän eri vaiheista ja niiden tärkeydestä, antaen tarvittavat tiedot ja työkalut siirtymäprojektin onnistuneeseen läpivientiin.

4.1 Legacy-järjestelmän ominaisuudet ja haasteet

Tietokantajärjestelmän elinkaari alkaa kehitysvaiheesta, jonka jälkeen siirrytään ylläpitovaiheeseen. Ylläpitovaihe on tyypillisesti elinkaaren pisin osa, jonka aikana järjestelmään tehdään pienimuotoisia parannuksia ja korjauksia tarpeen mukaan. Vaikka ylläpito pitää järjestelmän toimintakunnossa, jossain kohtaa se ei riitä vastaamaan jatkuvasti muuttuvia teknologisia ja liiketoiminnallisia vaatimuksia. Tästä syystä järjestelmä vaatii ajan saatossa laajempia uudistuksia tai kokonaisvaltaista päivitystä, jotta se pysyy tehokkaana ja turvallisena. Elinkaaren loppuvaiheessa ilmenee väistämättä järjestelmän täydellinen uudelleenrakentaminen tai lopetus. Elinkaaren loppuvaiheen päätös tehdään, kun vanhentuneet teknologiat ja arkkitehtuurit alkavat rajoittaa toiminnallisuutta ja skaalautuvuutta. Tällaiset vanhentuneet järjestelmät tunnetaan yleisesti Legacy-järjestelminä. Ne voivat aiheuttaa merkittäviä haasteita ylläpidossa ja vaikeuksia uudempiin teknologioihin integroidessa, minkä vuoksi niiden korvaaminen tai modernisointi nousee usein keskeiseksi kysymykseksi ohjelmistokehityksessä. Legacy-järjestelmän määritelmä kattaa niin ohjelmistot, laitteistot kuin tietokantajärjestelmätkin, jotka ovat teknologisesti vanhentuneita, mutta edelleen kriittisiä liiketoiminnan kannalta. (Hussain, Bhatti & Rasool 2017.)

Legacy-järjestelmäksi kutsutaan vanhentunutta teknologiaratkaisua, vaikka järjestelmän iän lisäksi on muitakin tekijöitä otettava huomioon määritelmässä (Hussain, Bhatti & Rasool 2017). Keskeisiä vanhentumisen piirteitä ovat puutteellinen dokumentaatio, suuri koodimäärä ja järjestelmän periytyminen kehittäjältä toiselle. Monet modernisoitavat järjestelmät on alun perin suunniteltu ja toteutettu aikana, jolloin nykyaikaiset ohjelmistokehitysmenetelmät eivät olleet käytössä. Tämän vuoksi niiden koodi on usein raskasta ja dokumentaatio on jäänyt puutteelliseksi tai vanhentuneeksi. Lisäksi järjestelmän ylläpito on siirtynyt kehittäjältä toiselle, mikä johtaa tilanteeseen, jossa alkuperäiset suunnittelijat eivät enää ole mukana ylläpidossa. (Birchall 2016.)

Näiden seikkojen yhdistelmä tekee vanhentuneiden järjestelmien ylläpitämisestä ja jatkokehittämisestä erityisen haastavaa. Ohjelmistojen vanhentuneet kielet ja arkkitehtuurit aiheuttavat yhteensopivuusongelmia nykyisten teknologioiden kanssa. Heikko dokumentaatio vaikeuttaa järjestelmän ymmärtämistä ja mahdollisten ongelmien ratkaisemista. Lisäksi näiden järjestelmien hallinnointi vaatii erityistä osaamista, jonka löytäminen on päivä päivältä haastavampaa. Näistä syistä vanhentuneiden järjestelmien modernisointi tai korvaaminen uudemmilla ratkaisuilla on usein välttämätöntä organisaation tehokkuuden ja turvallisuuden takaamiseksi. (Birchall 2016.)

4.2 Arviointi uusimisen tarpeesta

Tietokantajärjestelmien modernisointi tai siirtymä vaatii huolellista harkintaa ja ajoitusta, jotta voidaan välttää liiallisen aikapaineen aiheuttamat kompromissit infrastruktuurin laadussa. Uusimisen tarve tulee tunnistaa varhaisessa vaiheessa useiden merkkien perusteella. Nämä merkit ilmenevät, kun modernisoitava järjestelmä alkaa vaikuttaa liiketoiminnan arvoon haitallisesti esimerkiksi heikentämällä asiakaskokemusta tai rajoittamalla uusien palveluiden kehitystä. Yksi selvä merkki on ylläpitokustannusten kasvu tai kun järjestelmästä tulee niin jäykkä, ettei se salli tarvittavia päivityksiä tai muutoksia. (Hussain, Bhatti & Rasool 2017.)

Ongelmaksi voi nousta, että vanhentunut järjestelmä on teknologisesti yhteensopimaton uudempien järjestelmien kanssa, mikä rajoittaa integraatioiden mahdollisuuksia ja estää järjestelmien tehokkaan yhteistyön. Kun yksi tai useampi näistä haasteista tulee esiin, on välttämätöntä suorittaa kattava järjestelmän nykytilan analyysi. Tämän analyysin tulosten pohjalta voidaan päätellä, tuleeko koko järjestelmä uusia tai onko osittainen päivitys riittävä. Tässä yhteydessä on tärkeää arvioida, kuinka uusiminen vaikuttaa liiketoiminnan kokonaisvaltaiseen suorituskykyyn ja tulevaisuuden kasvumahdollisuuksiin. Näin ollen systemaattinen ja ennakoiva lähestymistapa tietokantajärjestelmän uusimisessa tukee organisaation pitkän aikavälin strategisia tavoitteita, vaikka modernisointiprosessi osoittautuisi aikaa vieväksi tai kalliiksi lyhyellä aikavälillä. (Hussain, Bhatti & Rasool 2017.)

4.3 Modernisoinnin lähestymistavat ja valmisteluvaihe

Modernisoidessa Legacy-tietokantajärjestelmää valinta oikean strategian suhteen on kriittinen. Neljä keskeistä lähestymistapaa tarjoavat vaihtoehtoisia polkuja, jotka sopivat erilaisiin organisaation tarpeisiin ja haasteisiin. Big Bang -lähestymistapa edustaa kokonaisvaltaista uudistusta, jossa sekä tietokanta että laitteisto päivitetään samanaikaisesti. Tämä menetelmä mahdollistaa nopean siirtymisen uuteen teknologiaan, mutta sisältää myös merkittäviä riskejä liittyen liiketoiminnan jatkuvuuden katkoksiin ja suureen investointitarpeeseen. Database First -

strategiassa puolestaan aloitetaan tietokannan modernisoinnilla säilyttäen samalla yhteys vanhaan järjestelmään. Tämä lähestymistapa vähentää operatiivista riskiä ja mahdollistaa sujuvamman siirtymisen. Lähestymistapa soveltuu erityisesti silloin, kun tietokanta on organisaation toiminnan ydin ja sen päivitys tuo välittömästi merkittävää lisäarvoa. (Fanelli, Simons & Banerjee 2016.)

Database Last -lähestymistapa keskittyy ensin muihin järjestelmän osiin, kuten sovelluksiin ja käyttöliittymiin, päivittäen tietokannan myöhemmässä vaiheessa. Tämä voi olla hyödyllistä, kun tietokanta on vakaampi ja vähemmän altis häiriöille. Hybrid -lähestymistapa yhdistää näiden kolmen strategian elementtejä tarjoten räätälöidyn ratkaisun, joka vastaa organisaation erityistarpeita ja tavoitteita. (Fanelli, Simons & Banerjee 2016.)

Kunkin lähestymistavan valinta riippuu monista tekijöistä, kuten organisaation riskinsietokyvystä, resurssien määrästä ja liiketoiminnan jatkuvuuden tarpeista. On tärkeää, että modernisointiprojekti suunnitellaan huolellisesti perustuen tarkkaan nykytilan analyysiin ja strategiseen suunnitteluun. (Fanelli, Simons & Banerjee 2016.)

Kun sopivin lähestymistapa järjestelmän modernisointiin on valittu, aloitetaan valmisteluvaihe. Tässä vaiheessa on keskeistä varmistaa, että vanhentuneen järjestelmän dokumentaatio, vaatimusmäärityksет ja suunnitelmat ovat ajan tasalla. Yksityiskohtainen ymmärrys järjestelmän alkuperäisestä tarkoituksesta ja rakenteesta on oleellista, jotta uudistus voidaan toteuttaa suunnitellusti ja ilman odottamattomia esteitä. (Rashid et al. 2013.)

Valmisteluvaiheessa tarkastellaan myös järjestelmän nykyisiä toiminnallisuuksia kriittisesti. Vuosien mittaan kertyneet, mutta nykyään tarpeettomat tai vanhentuneet ominaisuudet tunnistetaan ja niiden tarpeellisuus arvioidaan. Poistamalla nämä käyttämättömät osat modernisointiprojektin työmäärä ja kustannukset voidaan pitää hallinnassa, mikä edistää tehokkaampaa uudistusta. Valmistelutyö luo vankan perustan modernisointiprojektille, jonka pyrkimyksenä ei ole pelkästään vanhan järjestelmän korvaaminen. Sen sijaan modernisoinnin keskeisenä tavoitteena on saavuttaa huomattavia parannuksia, jotka tukevat pitkän aikavälin tavoitteita. (Rashid et al. 2013.)

Järjestelmän modernisointi on moniulotteinen prosessi, joka sisältää useita teknisiä vaiheita, kuten käänteisohjelmointia (Reverse Engineering), ohjelmointia (Forward Engineering), uudelleendokumentointia, kääntämistä ja uudelleenjärjestelyä. Käänteisohjelmoinnin kautta pyritään saamaan syvällinen käsitys järjestelmän toiminnasta. Tämä prosessi alkaa järjestelmän perusteellisella analysoinnilla ja sisäisten riippuvuuksien tunnistamisella, joka on oleellista parannettavien osa-alueiden löytämiselle. Kattava ymmärrys järjestelmästä auttaa pienentämään

uudelleenohjelmoinnin kustannuksia ja riskejä tarjoten samalla perustan uudistukselle. (Rashid et al. 2013.)

Modernisointiprosessissa hyödynnetään kokonaisuudessaan tai osittain ohjelmointia, joka suoritetaan etukäteen määriteltyjen suunnitelmien mukaisesti. Tässä lähestymistavassa järjestelmä tai sen komponentti luodaan alusta alkaen hyödyntäen perusteellisesti mietittyjä suunnitelmia ja tavoitteita, jotka ohjaavat rakentamisprosessia. Ohjelmointi mahdollistaa modernien ominaisuuksien ja toiminnallisuuksien sujuvan yhdistämisen järjestelmään. (Rashid et al. 2013.)

Lisäksi modernisoinnissa korostuu uudelleendokumentoinnin ja uudelleenjärjestelyn rooli. Uudelleendokumentointi varmistaa, että kaikki järjestelmän osat ovat kunnolla dokumentoituja, mikä helpottaa tulevaa ylläpitoa ja mahdollisia jatkokehitystä. Uudelleenjärjestely puolestaan keskittyy järjestelmän rakenteen optimointiin ja varmistaa, että uusi järjestelmä on mahdollisimman tehokas ja toimiva. (Rashid et al. 2013.)

4.4 Tavoitteet ja hyödyt

Modernisointiprojektien päätavoite on yleensä tietokantajärjestelmän laadun ja ylläpidettävyyden parantaminen. Tämä käsittää useita näkökulmia, joista keskeisin on järjestelmän kehittäminen ja ylläpidon kustannustehokkuuden varmistaminen. Laatu viittaa tässä yhteydessä sekä järjestelmän sisäisiin että ulkoisiin ominaisuuksiin. Sisäiset ominaisuudet kattavat muun muassa koodin selkeyden, joustavuuden ja uudelleenkäytettävyyden, kun taas ulkoiset ominaisuudet liittyvät esimerkiksi käyttöliittymän intuitiivisuuteen ja käyttäjäkokemuksen sujuvuuteen. (Birchall 2016.)

Modernisoinnin myötä on mahdollista korjata aiemmin tunnistettuja virheitä ja puutteita, mikä parantaa järjestelmän luotettavuutta ja suorituskykyä. Dokumentaation päivitys ja parannus ovat myös olennaisia, sillä ne mahdollistavat tehokkaamman järjestelmän ylläpidon ja kehittämisen. Nykyisten standardien ja teknisten vaatimusten mukaan suunnittelu edistää myös tietokantajärjestelmän yhteensopivuutta muiden teknologioiden ja järjestelmien kanssa, mikä on erityisen tärkeää nopeasti muuttuvassa teknologisessa ympäristössä. (Birchall 2016.)

Lisäksi modernisointi voi tarjota mahdollisuuksia järjestelmän toiminnallisuuden laajentamiseen ja uusien ominaisuuksien lisäämiseen, mikä tukee organisaation pitkän aikavälin strategisia tavoitteita. Esimerkiksi datan hallinta, analysointi ja turvatoimenpiteet ovat ominaisuuksia, joita on arvokasta parantaa järjestelmää modernisoidessa. (Birchall 2016.)

4.5 Riskien hallinta ja haasteiden ennakointi

Riskit ovat usein suuremmat uudistaessa vanhaa järjestelmää kuin perinteisessä ohjelmistoprojektissa. Tämä johtuu siitä, että olemassa oleva järjestelmä perustuu vanhentuneeseen toteutukseen, ja sen keskeiset toiminnot tulee suunnitella uudelleen hyödyntäen nykyaikaista teknologiaa. Voi olla haastavaa, että uudistuksen tulee säilyttää alkuperäisen järjestelmän toiminnallisuus, mutta samalla parantaa sen tehokkuutta ja käytettävyyttä täysin uudella toteutustavalla. Tämän lisäksi vanhettunut teknologia ja arkkitehtuuri saattaa haitata uusien toimintojen lisäämistä. Myös olemassa olevan datan siirto uuteen järjestelmään voi osoittautua monimutkaiseksi. (Rashid et al. 2013.)

Modernisoinnin riskien ymmärtäminen ja niiden asianmukainen hallinta ovat keskeisiä tekijöitä projektin onnistumiselle. Tämä edellyttää huolellista suunnittelua, laadunvarmistusta ja riskien ennakointia koko projektin ajan. Seuraavat alaluvut tarkastelevat riskejä kustannukset, laatu ja suorituskyky huomioon ottaen. (Rashid et al. 2013.)

4.5.1 Kustannusriskit

Legacy-järjestelmiä uudistaessa tavoitteena on usein kustannustehokkuuden parantaminen uusien teknologioiden avulla. Modernisointi voi kuitenkin tuoda mukanaan taloudellisia riskejä. Esimerkiksi uudistuksen kokonaiskustannukset voivat ylittää alkuperäiset arviot, erityisesti jos projektissa ilmenee suunnittelemattomia teknisiä haasteita tai sen laajuutta on aliarvioitu. (Rashid et al. 2013.)

Toinen merkittävä riski liittyy ylläpitokustannusten ennakoitua hitaampaan laskuun. Uuden järjestelmän ylläpito voi vaatia odottamattomia resursseja. Muun muassa jatkuvat päivitykset tai integrointihaasteet voivat nostaa pitkän aikavälin kustannuksia. Tämän vuoksi on tärkeää, että modernisointiprojektin taloudellinen kannattavuus arvioidaan huolellisesti ottaen huomioon sekä välittömät että pitkän aikavälin kustannukset. (Rashid et al. 2013.)

4.5.2 Laatu ja suunnittelun epäjohdonmukaisuus

Modernisointiprojektien yhteydessä esiintyy usein haasteita, jotka liittyvät projektin laatuun ja suunnittelun johdonmukaisuuteen. Yksi keskeisistä riskeistä on, että uudistusprosessin laatu ei vastaa odotuksia, mikä voi johtua puutteellisesta suunnittelusta, resurssien allokoinnista tai teknisten vaatimusten ymmärtämättömyydestä. Tämän seurauksena projektin lopputulos voi olla epäjohdonmukainen tai ei täytä alkuperäisiä tavoitteita, mikä voi vaikuttaa sekä järjestelmän toimivuuteen että käyttäjäkokemukseen. (Rashid et al. 2013.)

Lisäksi suunnittelun epäjohdonmukaisuus voi aiheuttaa taloudellisia riskejä, kuten odottamattoman kalliita varmuuskopiointiratkaisuja tai ylimääräisiä kuluja projektin hallinnoinnissa ja seurannassa. Nämä kustannukset voivat ilmetä muun muassa odottamattomina menoina projektinhallintatyökaluissa tai viivästyneissä aikataulussa, jotka johtavat lisätyövoiman tarpeeseen. Jos uusi toteutus ei toimi luotettavasti, voi laadunvalvonta aiheuttaa kulueriä. Tällaiset kustannukset ovat usein seurausta siitä, että suunnitteluvaiheessa ei ole huolellisesti arvioitu kaikkia projektin näkökohtia tai budjetoitu riittävästi varoja mahdollisiin ongelmatilanteisiin. (Rashid et al. 2013.)

4.5.3 Suorituskyvyn riskit

Tietokantajärjestelmän modernisoinnissa keskeistä on suorituskyvyn vaatimusten huomioiminen. Vaatimukset tulee mitoittaa niin, että ne vastaavat tulevaisuuden tarpeita ottaen huomioon resurssien käytön keskiarvot ja pidemmän aikavälin ennusteet. Jos suorituskyvyn vaatimuksia ei analysoida riittävästi, uusi järjestelmä saattaa jäädä lyhytikäiseksi ratkaisuksi. (Db Pro Services s.a..)

Suorituskyvyn riskit ilmenevät muun muassa yhteensopivuusongelmina uusien ja vanhojen järjestelmien välillä, mikä voi heikentää suorituskykyä ja toimintavarmuutta. Tämän estämiseksi modernisointiprosessissa on keskeistä suorittaa perusteellinen suunnittelu ja testaus. Näiden avulla varmistetaan, että uusi järjestelmä täyttää nykyiset vaatimukset, mutta toimii myös pitkäaikaisena ratkaisuna tulevaisuuden tarpeisiin. (Rashid et al. 2013.)

4.6 Ylläpitovaihe

Ylläpitovaihe muodostaa merkittävän osan tietokantajärjestelmän elinkaaren kustannuksista. Järjestelmän kehitysvaihe saattaa kestää vain muutamia kuukausia, mutta ylläpito voi jatkua useita vuosia, mikä nostaa ylläpidon osuuden jopa 50–90 prosenttiin koko projektin kustannuksista. Tämän vuoksi on välttämätöntä kiinnittää huomioita ylläpidon kustannustehokkuuteen ja varmistaa, että resurssit kohdistetaan järkevästi. Ylläpidon tehokkuus kustannusten säästön lisäksi myös takaa järjestelmän pitkäaikaisen suorituskyvyn ja luotettavuuden. Jatkuvan seurannan ja arvioinnin avulla voidaan tunnistaa ja korjata tehottomuudet, jolloin ylläpitokustannukset pysyvät hallinnassa ja järjestelmän elinkaari maksimoidaan. (Vasconcelos, Kimble, Carreteiro & Rocha 2017.)

Koodin hallinnan ja vähentämisen merkitys korostuu uudistetun järjestelmän ylläpitovaiheessa. Yksinkertaistamalla lähdekoodia ja poistamalla tarpeettomat toiminnallisuudet voidaan parantaa järjestelmän suorituskykyä ja vähentää ylläpidon monimutkaisuutta. Koodin vähentämisen prosessi vaatii huolellista suunnittelua, jolla varmistetaan, ettei poistettavalla koodilla ole tuntemattomia tai piilossa olevia riippuvuuksia muihin järjestelmän osiin. Nykyaikaiset työkalut tarjoavat merkittävää

apua tässä prosessissa mahdollistaen koodin tehokkaan analysoinnin ja riippuvuuksien tunnistamisen. Koodin yksinkertaistaminen ja optimointi ovat keskeisiä tekijöitä ylläpidon tehokkuudessa, sillä ne vähentävät järjestelmän ylläpitäjien kuormitusta ja auttavat pitämään ylläpidon kustannukset kurissa. (Seacord, Plakosh & Lewis 2003, luku 17.3.)

Tässä luvussa olemme tarkastelleet tietokantajärjestelmän elinkaarta ja tietokantasiirtymän prosessia. Luku käsittelee tietokannan modernisoinnin tarpeen merkkejä, toteutustapoja sekä antaa ohjeita riskien hallintaan ja uuden järjestelmän tehokkaaseen ylläpitoon. Ymmärrys näistä vaiheista on oleellista toimeksiantajalle, joka toteuttaa tietokantasiirtymää Snowflakeen. Luvussa esitetyt tiedot muodostavat perustan seuraavalle luvulle, jossa keskitytään siirtymäprojektin käytännön toteutukseen.

5 Siirtymäprosessi ja toteutus

Tässä luvussa käsitellään yritys X:n tietokantasiirtymäprosessia SQL Serveristä Snowflakeen. Siirtymä on osa laajempaa tietojärjestelmän modernisointia. Puhuttaessa tietojärjestelmästä tarkoitetaan kokonaisuutta, joka sisältää kaikki järjestelmät ja komponentit, joita tarvitaan, että laadukas ja eheä data on toimeksiantajan luoman sovelluksen saatavilla. Luku alkaa tietokantasiirtymän vaiheittaisesta toteutuksesta ja sen merkityksestä. Alaluvuissa selvitetään, miten erilaiset ratkaisumallit dataputkien toteutuksessa vaikuttavat tietokannan suorituskykyyn ja kustannuksiin sekä miksi tietyt toteutustavat sopivat parhaiten organisaation tarpeisiin. Lisäksi käydään läpi erilaiset testausmenetelmät datan eheyden varmistamiseksi, mikä on keskeinen osa sujuvaa siirtymää.

On tärkeää huomioida, että tässä luvussa SQL Serveristä puhuttaessa viitataan toimeksiantajan SQL Server toteutukseen, eikä luku käsittele SQL Serveriä kokonaisuudessaan. Tämä johtuu siitä, että kyseisestä Microsoftin tietokantajärjestelmästä on useita eri versioita. Organisaatio on luonut aikaisemman tietokantajärjestelmän virtuaalisen SQL Serverin päälle, jonka ominaisuudet eivät välttämättä päde jokaiseen versioon. Tämän takia toteamuksia ei ole tarkoitettu yleispäteviksi vaan kyseistä ratkaisua kuvailevaksi.

Siirtymään ryhtynyt toimeksiantaja on perustamisesta lähtien käyttänyt SQL Serveriä keskeisenä tietokantaratkaisunaan. Melkein kymmenen vuoden ajan SQL Server on toiminut infrastruktuurin pohjana, tarjoten tukea organisaation kasvulle ja palveluiden laajentumiselle. Tänä aikana palveluvalikoima, että asiakaskunta onkin kasvaneet merkittävästi. Kuitenkin tietokannan monimutkaistuessa ja datamäärän kasvaessa laskentaprosessit tietokannassa ovat muuttuneet aikaa vieviksi ja suurimpien ajojen suorittamiseen saattaa kulua useita tunteja. Vaikka nykyinen tietokantaympäristö toimii luotettavasti, niin modernisointiprosessin proaktiivinen tarkastelu on oleellista, jotta organisaation kasvutavoitteet ja tietokannan laskentakyky pysyvät linjassa toistensa kanssa. Tämän tarpeen ajamana alkoi selvitystyö erilaisista vaihtoehdoista, jotka mahdollistaisivat jatkossakin vastaavanlaisen kasvun tukemisen. Uuden järjestelmän tulee tarjota ratkaisu kasvavaan skaalaustarpeeseen ja toimia alustana, jonka avulla tietokantajärjestelmä ja laajempi tietojärjestelmä voidaan kattavasti modernisoida vastaamaan tulevaisuuden vaatimuksia.

Valinta oikean tietokantateknologian suhteen on tärkeä päätös organisaation tietojenkäsittelykyvyn ja tulevaisuuden kasvun kannalta. Organisaation tavoitteena on löytää skaalautuva ja pilvipohjainen tietokantaratkaisu, joka yhdistää tehokkuuden, luotettavuuden ja uusimman teknologian hyödyntämisen järjestelmässään. Organisaatio analysoi ratkaisuvaihtoehtoja ja päätyi kahteen parhaimpaan, joita ovat joko lisäinvestointi nykyiseen SQL Serveriin tai siirtyminen Snowflake-tietokantaan. SQL Serverin etuna on sen olemassa oleva ja toimiva arkkitehtuuri, mikä

tarjoaa riskittömän polun suoritusaikojen parantamiseen. Kaikki aiemmin rakennetut ratkaisut jatkaisivat toimintaansa saumattomasti. Toisaalta Snowflake kykenee tarjoamaan laajemmin skaalautuvuutta mahdollistaen joustavan resurssien mukauttamisen kasvavan datamäärän hallintaan.

Näiden kahden vertailuprosessissa on oleellista tarkastella kunkin vaihtoehdon hyötyjä ja riskejä pitkällä aikavälillä. SQL Serverin jatkokehitys ilmenee riskittömänä ja helppona vaihtoehtona valmiiksi rakennetun infrastruktuurin takia, kun taas Snowflaken tarjoama tehokkuus ja skaalautuvuus saattaa olla ratkaisuna pitkäaikaisempi. Kuitenkaan Snowflaken implementointi ei olisi pieni kehitysprojekti, vaan edellyttäisi koko tietojärjestelmäarkkitehtuurin perusteellista uudelleensuunnittelua, mikä asettaa sekä haasteita että mahdollisuuksia. Harkinnan ja potentiaalisten hyötyjen tunnistamisen jälkeen organisaatio päätyi siirtymään Snowflake-tietokantaan. Tämä päätös siirtymästä perustuu huolelliseen arviointiin, jonka tuloksena syntyi näkemys, että Snowflake kykenee tarjoamaan paremmat valmiudet vastata tulevaisuuden tarpeisiin.

5.1 Tietokantasiirtymän vaiheet

Tietokantasiirtymä on moniulotteinen kehitysprojekti, jossa on otettava huomioon erilaiset vaatimukset, joita aiempi tietokantajärjestelmä on ajan saatossa asettanut. Tämä edeltävä järjestelmä on jalostanut ja muokannut dataa tietyllä tavalla, luoden rajapinnoille odotuksia siitä, missä muodossa datan tulisi olla. Lisäksi järjestelmä on integroitunut erilaisiin tietolähteisiin ja kolmannen osapuolen palveluihin, joihin myös uuden tietokannan pitäisi pystyä yhdistymään. Siirtymä on onnistunut Snowflaken pystyessä vastaamaan jo luodun infrastruktuurin odotuksiin.

Siirtymäprosessin aloitusmalli on ensisijaisen tärkeä päätös. Aloitusmalleista kerrotaan enemmän tietoperustan luvussa 4.3. Modernisointiprosessi voidaan aloittaa tyypillisesti kolmesta eri lähtökohdasta. Kehitys voidaan aloittaa sovelluksesta ja siihen kuuluvien ohjeisjärjestelmistä (Database Last), tietokannasta (Database First) tai kehittää komponentteja samanaikaisesti (Big Bang). Harkinnan jälkeen valittiin Database First -lähestymistapa, jossa kehitetään ensin tietokanta mahdollisimman valmiiksi ennen sen integroimista sovellukseen. Tämä malli valikoitui, koska tietokanta on keskeinen liiketoiminnan mahdollistaja ja tärkeä osa palvelua. Tavoitteena on datan pitkälle viety rikastaminen, minkä vuoksi tietokantalähtöinen lähestymistapa on luonteva ja sillä on merkittävä vaikutus palvelun laatuun.

Jotta siirtymä voidaan toteuttaa hallitusti ja riskejä minimoiden, kehitettiin vaiheittainen prosessi etenemiselle. Tämä prosessi mahdollistaa keskittymisen kriittisimpiin tekijöihin kunkin projektin vaiheen aikana. Näin taataan, että jokainen vaihe tuottaa lisäarvoa ja edistää päämäärään

pyrkimistä. Tulevissa alaluvuissa tutustutaan tarkemmin näihin vaiheisiin ja kuvataan, mitä kukin niistä pitää sisällään.

5.1.1 Proof of Concept

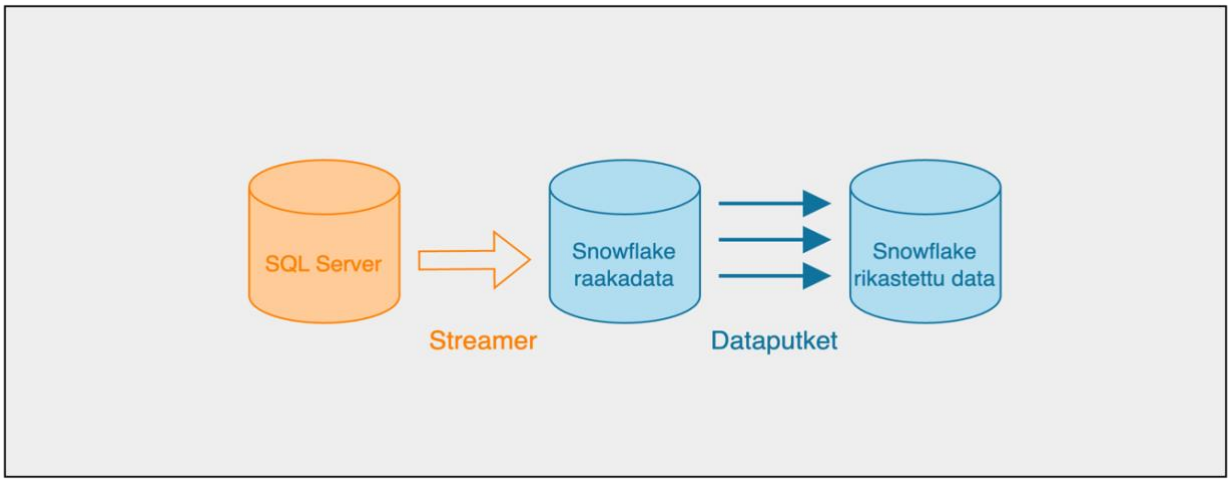
Siirtymä Snowflake-tietokantaan aloitettiin kolmivaiheisella prosessilla, jonka ensimmäinen askel on Proof of Concept (POC). Tässä vaiheessa organisaatio teki alustavan investoinnin Snowflakeen, mutta varmistaen, että toiseen ratkaisuun päätyminen ei aiheuttaisi merkittäviä taloudellisia haittoja. POC-vaiheen tavoitteena on varmistaa, että Snowflaken teknologia kykenee vastaamaan organisaation laskenta- ja datankäsittelytarpeisiin tehokkaasti ja luotettavasti. Kokeilu keskittyi yhden merkittävän laskentaprosessin hahmotteluun, johon sisältyy useita suuria tauluja, proseduureja ja funktioita. Laskentaprosessin tarkemmat yksityiskohdat ovat luottamuksellisia ja jäävät sen vuoksi tämän vaiheen kuvauksen ulkopuolelle.

Tämän prosessin onnistunut toteutus Snowflake-alustalla osoitti teknologian soveltuvuutta organisaation tarpeisiin ja vahvisti luottamusta järjestelmän suorituskykyyn. Merkittävästi nopeampi suorituskyky verrattuna nykyisen ympäristön vastaavaan prosessiin tarjosi konkreettisen todisteen Snowflaken teknologisista eduista. Onnistunut POC-vaihe toimii perustana siirtymäprosessin seuraaville vaiheille alustaen samalla Snowflake-tietokannan täysimittaista hyödyntämistä. Kokeilu osoitti Snowflaken arkkitehtuurin pystyvän organisaatiolle oleelliseen laskentaprosessiin samalla saavuttaen merkittäviä parannuksia käsittelyn nopeudessa ja tehokkuudessa.

5.1.2 Minimum Viable Product

Seuraava vaihe tietokantasiirtymässä on Minimum Viable Product (MVP), joka keskittyi tietokannan perustan luomiseen. Tämän vaiheen tavoite on muodostaa kattava ja vankka perusta tietokannalle, joka käsittää kaikki keskeiset taulut ja niiden pohjalta rakennetut dataputket. Nämä dataputket on suunniteltu erityisesti datan jalostamiseen varmistaen samalla tiedon eheyden ja luotettavuuden.

MVP-vaiheen toinen merkittävä tavoite on Streamer-nimisen dataputken kehittäminen, joka toimii siltana SQL Serverin ja Snowflaken välillä. Tämä putki mahdollistaa muutosten viemisen SQL Serveristä Snowflakeen reaaliajassa varmistaen, että molemmat tietokannat pysyvät yhtenäisinä ja ajantasaisina. Käytännössä Snowflaken ainoa datalähde on tässä kehitysvaiheessa SQL Server ja kaikki SQL Serverin vastaanottamat tiedot siirtyvät Snowflaken Streamer-putken avulla. Kuvailtu arkkitehtuuri on visualisoitu kuvassa 2.



Kuva 2: MVP-osion arkkitehtuuri

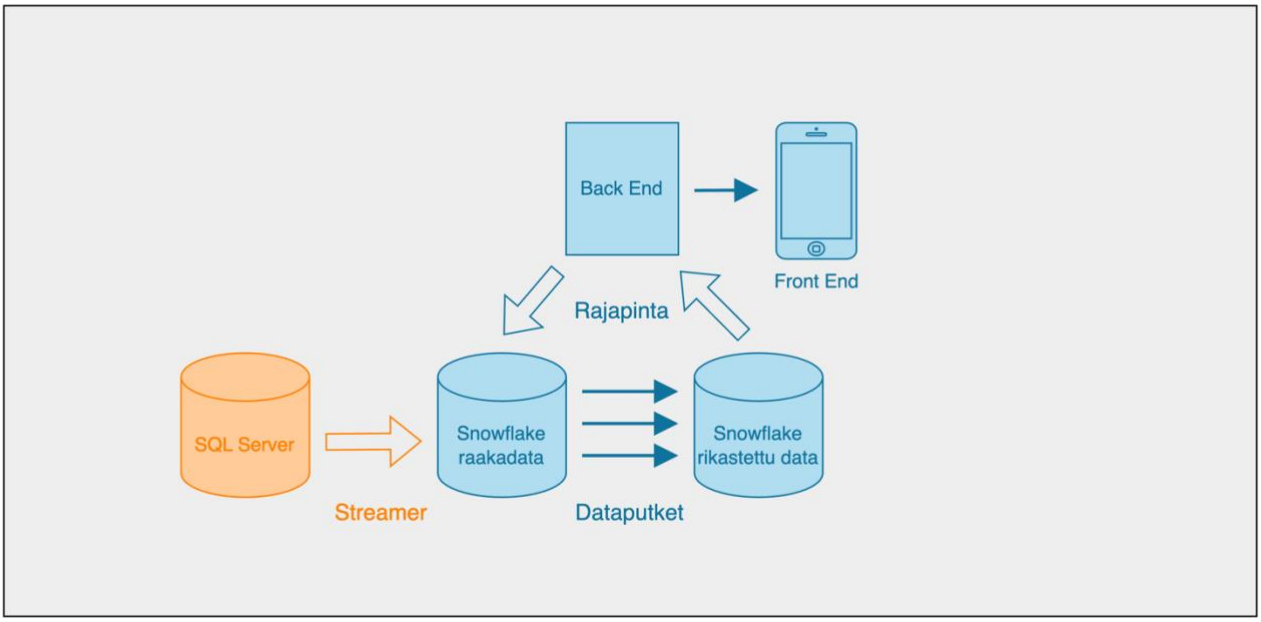
Molempien tietokantajärjestelmien ajantasaisuus takaa, että data on johdonmukaista ja vertailukelpoista tietokannasta huolimatta, mikä on oleellista kehitysprosessin aikana. Näin voidaan luoda Snowflaken infrastruktuurin perusta hallitusti. Perustaa voi jatkokehittää luomalla muitakin datalähteitä järjestelmälle SQL Serverin lisäksi.

5.1.3 Hybridikehitysvaihe

Tietokantasiirtymä prosessin Proof of Concept ja Minimum Viable Product -vaiheiden suorittaminen on johdattanut kehitysprosessin kolmanteen vaiheeseen nimeltä Hybridikehitysvaihe. Tämä vaihe on luonnollinen jatkumo aiemmille vaiheille, joissa keskityttiin Snowflake-tietokannan itsenäiseen kehitykseen ilman yhteyttä tuotteena toimivaan sovellukseen. Eli Snowflake tietokantaa kehitettiin samalla, kun SQL Server toimi ainoana tietokantana tuotannossa toimivalle sovellukselle. Tämä strategia on mahdollistanut operatiivisen riskin minimoinnin ja sujuvan aloituksen modernisointiprosessiin.

Hybridikehitysvaihe erottuu aiemmista vaiheista sillä, että se edellyttää tiivistä yhteistyötä Front-End, Back-End ja tietokantakehityksen välillä. Tässä vaiheessa organisaation johto määrittelee, mikä sovelluksen palvelu tai toiminnallisuus on seuraavaksi kehitettävä tai päivitettävä. Tämän päätöksen perusteella kehitystiimit työskentelevät yhdessä varmistaakseen, että kaikki tarvittavat osa-alueet tukevat toisiaan parhaalla mahdollisella tavalla. Käytännössä tämä tarkoittaa, että modernisointiprosessi ei ole enää itsenäistä kehitystä. Snowflake-tietokanta yhdistetään rajapintojen avulla eri palveluihin, jotka pyytävät tietokannalta dataa. Rajapintojen kautta siirtyvä data tulee olla halutussa muodossa, jotta Back-End kehittäjät kykenevät hyödyntämään tätä. Käytännössä hybridikehitysvaiheessa Back-End-tiimi määrittelee tarvitsemansa datan, jonka

pohjalta tietokantaa kehitetään vastaamaan näitä tarpeita. Kuva 3 havainnollistaa, mitä uutta hybridikehitysvaihe on tuonut mukanaan verrattuna aikaisempaan arkkitehtuurikuvaan.



Kuva 3: Hybridikehitysvaiheen arkkitehtuuri

Hybridikehitysvaiheen myötä tietokanta ei ole enää erillinen entiteetti kehitysprosessissa, vaan se on osa kokonaisvaltaista kehitystyötä. Tässä lähestymistavassa tietokannan rooli on mukautuva ja vastaa reaaliaikaisesti muiden kehitystiimien tarpeita. Näin ollen tietokanta kehittyy symbioosissa sovelluskehityksen kanssa, mikä mahdollistaa kohdennetun kehitysprosessin, jossa tuotetta parannetaan yksi osuus kerrallaan hallitusti.

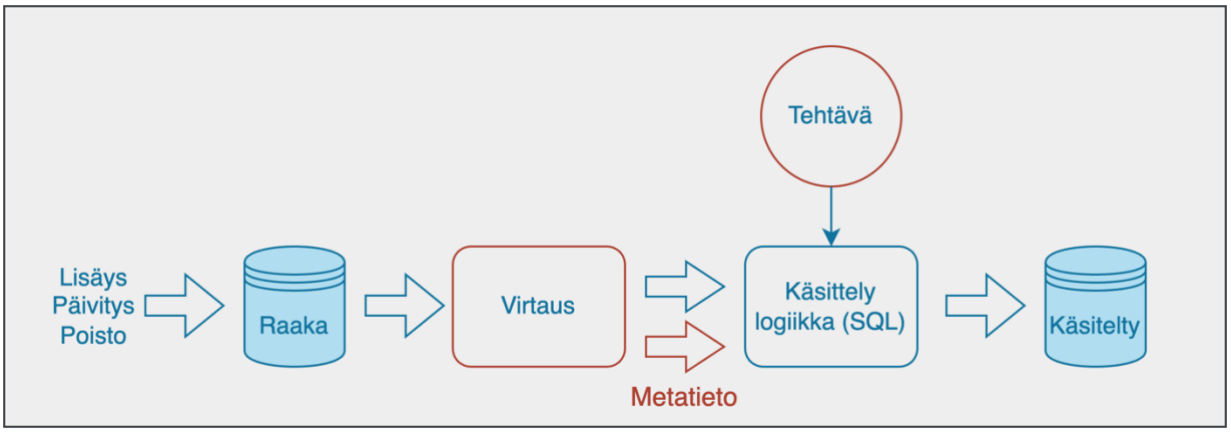
5.2 Dataputkien toteutus

Tavoitteena on löytää Snowflaken toiminnallisuuksista mahdollisimman yksinkertainen, luotettava ja tehokas ratkaisu dataputkille, jotka tuottaisivat käsiteltyä ja laadukasta dataa. Tätä päämäärää tavoitellessa kartoitettiin ja kokeiltiin jokaista Snowflaken tarjoamaa toimintoja, jotka voisivat osoittautua hyödylliseksi organisaation tarpeisiin. Tavoiteltu lopputulos tutkinnalle on kattava näkemys näistä ominaisuuksista, niiden hyödyistä sekä rajoitteista. Näin varmistetaan, että tuleva tietokantajärjestelmä on tehty parhaalla mahdollisella tavalla.

5.2.1 Ratkaisumalli A: Stream & Task

Aluksi kokeilin hyödyntää Snowflaken stream- ja task-toimintoja dataputken rakentamiseen. Prosessi alkoi lähdetaulun luomisella, joka vastaanottaa raakadataa tietokannan ulkopuolelta. Tämän jälkeen loin virtauksen (Stream), joka tallentaa lähdetaulun muutokset, kuten uudet rivit,

poistot tai arvojen muutokset reaaliajassa. Virtaus muistuttaa hieman taulua, lisäten rivejä jokaisesta muutoksesta ja niiden ajankohdasta. Seuraava askel oli tehtävän (Task) määrittäminen. Tehtävä tarkastaa säännöllisin väliajoin, onko uusia rivejä ilmestynyt virtaukseen, suorittaa määritellyt muutokset datalle ja siirtää käsitellyn datan kohdetauluun. Dataputki on visualisoitu kokonaisuudessaan kuvassa 4.



Kuva 4: Stream & Task dataputki (Medium 2020)

Ensimmäisen dataputken luomisen jälkeen ilmeni pian, että vaikka stream- ja task-menetelmä tarjoaa vakaan tavan käsitellä dataa, se tuo mukanaan myös omia haasteita. Yksi ongelma on prosessin kustannukset, erityisesti kun tehtävä asetetaan ajettavaksi tihein väliajoin. Koska tehtävä tarkastaa virtauksen jatkuvasti tietovarasto joutuu käynnistämään itseään tauotta, mikä johtaa korkeisiin kustannuksiin. Lisäksi tämä lähestymistapa osoittautui jäykäksi, sillä se ei salli joustavaa datan uudelleenkäsittelyä ilman merkittäviä kiertoteitä, kuten koko kohdetaulun datan poistamista ja lähdetaulun muokkaamista. Myös liitosten luominen on haastavaa, koska siihen vaaditaan useampaa virtausta, jotka saattavat vastaanottaa dataa eri aikoihin ja tämän takia vaatia enemmän määrittelyä.

Ensimmäisen dataputken kehityskokemuksen perusteella todettiin, että se ei vastaa organisaation näkemystä ideaalista infrastruktuurin rakenteesta. Menetelmän kustannustehokkuus, joustavuuden puute ja ylläpidon monimutkaisuus ovat merkittäviä esteitä. Tämän seurauksena päätettiin jatkaa Snowflaken tarjoamien toiminnallisuuksien tutkimista. Tavoitteena on löytää ratkaisu palvelemaan organisaation tarvetta kustannustehokkaasta ja joustavasta rakenteesta dataputkille.

5.2.2 Ratkaisumalli B: Näkymät ja materialisoidut näkymät

Seuraavaksi tarkastelin näkymien (View) ja materialisoitujen näkymien (Materialized View) mahdollisuutta toimia perustana dataputkien rakentamisessa. Näkymien ja materialisoitujen

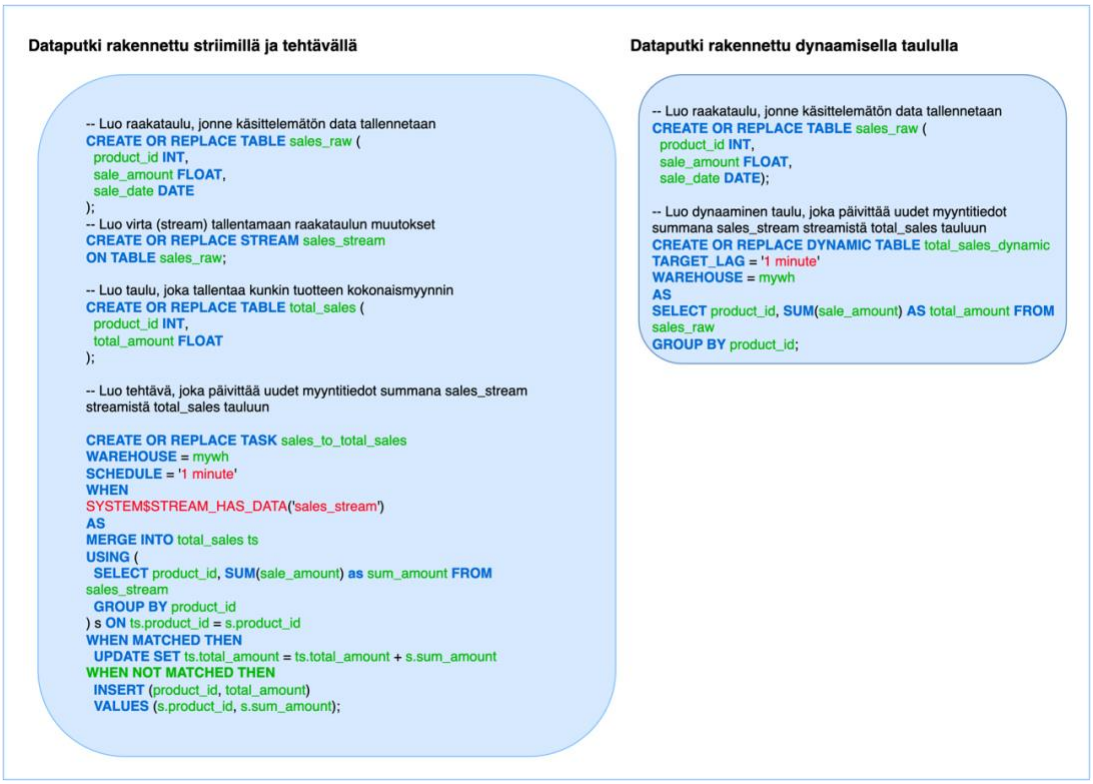
näkymien etuna on, että ne eivät tuo lisäviivettä datan käsittelyyn toisin kuin tehtäväpohjaiset dataputket, joissa muutokset eivät näy reaaliajassa. Näkymät ovat käytännössä nimikkeen alle tallennettuja kyselyjä, jotka mahdollistavat aina ajantasaisen tiedon esittämisen. Tämä tekee niistä houkuttelevan vaihtoehdon tapauksissa, joissa on tarve esittää juuri vastaanotettu tieto nopeasti. Esimerkiksi käyttöliittymässä voi olla useita tilanteita, jossa halutaan heti näyttää käyttäjälle hänen tekemänsä toiminnan tulokset eikä muutaman minuutin kuluttua.

Perinteiset näkymät eivät välttämättä ole kuitenkaan paras ratkaisu datan puhdistukseen ja muokkaukseen, sillä ne eivät tallenna tietoa pitkäaikaisesti. Ratkaisuna tähän ongelmaan kokeilin materialisoituja näkymiä, jotka tallentavat kyselyjen tulokset, tehden niistä kustannustehokkaan ratkaisun tilanteissa, joissa samoja tuloksia tarvitaan toistuvasti. Materialisoitujen näkymien avulla voidaan käsitellä data haluttuun muotoon niin, että seuraavan kerran kun tulosta tarvitaan, voidaan hakea aiemmin saatu tulos. Näin samaa raskasta käsittelyprosessia ei tarvitse tehdä uudelleen. Törmäsin kuitenkin rajoitteiseen materialisoidussa näkymässä, sillä sitä ei voi luoda kyselyn pohjalta, jossa on useampi taulu liitetty toisiinsa. Vaikka kyseisen toiminnon tallennusominaisuus tarjoaa etuja, liitoskomentojen tarve useimmissa dataputkissa teki niiden käytöstä mahdotonta useimmissa tilanteissa.

5.2.3 Ratkaisumalli C: Dynaamiset taulut

Tässä kohtaa huomattiin näkymien ja materialisoitujen näkymien olevan ratkaisu käyttöliittymän komponenteille, jotka vaativat nopean datan toimituksen käyttäjille. Edelleen etsitään ratkaisua tilanteisiin, joissa käsiteltyä tietoa tarvitaan toistuvasti, mutta laskennan välittömyys ei ole kriittistä. Tutkinta suuntautui seuraavaksi Snowflaken dynaamisiin tauluihin, jotka olivat tuolloin Public Preview -vaiheessa eli kyseessä on uusi ominaisuus, joka on kaikkien käyttäjien testattavissa. Dynaamiset taulut tarjoavat käytännössä automatisoidun version Stream- ja Task -dataputkesta, mutta ovat huomattavasti yksinkertaisempia toteuttaa ja hallinnoida. Toisin kuin aikaisemmat lähestymistavat, jotka vaativat useiden komponenttien luomista, dynaamisten taulujen käyttöönotto vaatii vain itse dynaamisen taulun määrittelyn. Tämä ilmenee myös kuvasta 5, jossa vertaillaan, kuinka monta riviä koodia vaaditaan tekemään sama dataputki, joko Stream- ja Task-toiminnoilla tai

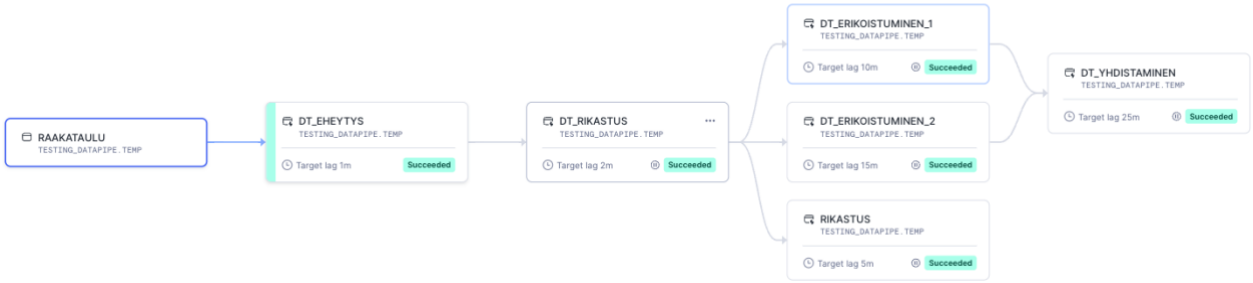
dynaamisen taulun avulla.



Kuva 5: Stream- ja Task -dataputki verrattuna dynaamiseen tauluun (Snowflake documentation s.a. c)

Dynaamiset taulut ovat osoittautuneet ratkaisuksi, joka merkittävästi yksinkertaistaa dataputkien luomista ja hallintaa. Vähentämällä koodin ja eri komponenttien määrää alennetaan samalla virheiden riskiä ja mahdollistetaan järjestelmän rakentaminen hallinnoitavaksi ja seurattavaksi. Testauksen perusteella dynaamiset taulut ovat tehokas ratkaisu useisiin haasteisiin, joita ilmeni edellisiä toimintoja tarkasteltaessa vähentäen samalla merkittävästi kustannuksia.

Dynaamisten taulujen käyttö tarjoaa joustavuutta datan käsittelyyn, koska ominaisuuden ansiosta käyttäjät voivat luoda monimutkaisia liitoskyselyjä suoraan taulun määrittelyn yhteydessä. Snowflaken käyttöliittymän dynaamisten taulujen näkymä tarjoaa kokonaisvaltaisen kuvan kaikista toisiinsa linkitetyistä tauluista, joka helpottaa seurantaa ja hallinnointia. Kuvassa 6 esitetty näkymä informoi käyttäjää siitä, onko dynaaminen taulu aktiivinen, onko se kohdannut virheitä tai onko prosessi keskeytetty. Lisäksi tämä näkymä tarjoaa arvokasta tietoa datan käsittelyprosessista, kuten kunkin prosessin vaiheen viiveen ja näkymä mahdollistaa syventymisen mihin tahansa yksittäisen dynaamisen taulun tarkasteluun.



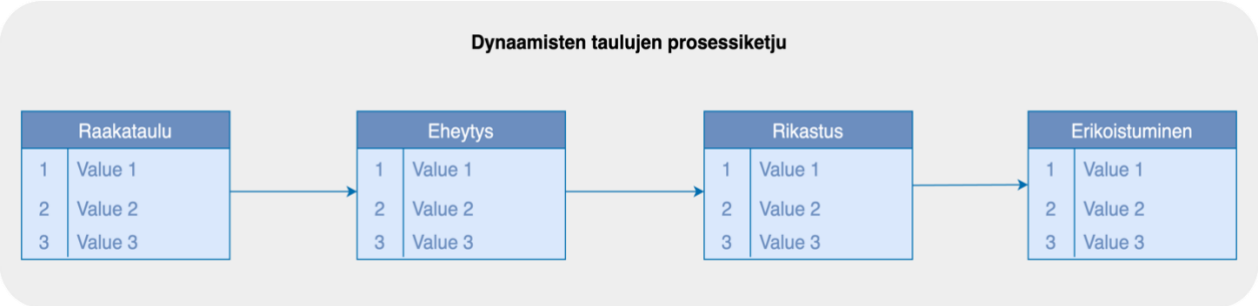
Kuva 6: Dynaamisten taulujen näkymä Snowflake-käyttöliittymässä

Dynaamiset taulut toimivat työkaluna suurten datamäärien käsittelyyn, sillä ne pystyvät päivittämään tietoja tehokkaasti keskittymällä vain uuteen tai muuttuneeseen dataan. Tämä lähestymistapa auttaa säästämään resursseja varsinkin prosesseissa, joissa datamäärät ovat suuria. Jos tarve vaatii, dynaamisen taulun sisältö on myös helppo päivittää täysin. Sen voi tehdä yksinkertaisesti poistamalla taulun ja luomalla tämän uudelleen tarvittavin muutoksin. Tämä mahdollistaa koko datajoukon päivittämisen. Menettely on suoraviivaisempi ja helpompi toteuttaa verrattuna Stream- ja Task-dataputkeen. Edellä mainitun dataputken muutokset vaativat usein uudelleen määrittelyä sen jokaisen komponentin osalta, eli niin lähdetaulussa, virtauksessa kuin kohdetaulussa.

Dynaamisten taulujen hyötynä voidaan mainita:

- Hallinnointi ja seuranta dynaamisten taulujen näkymästä
- Helpottaa join-liitoksia
- Uudelleen määrittely vaatii vähemmän toimenpiteitä
- Vähemmän koodia
- Kustannustehokas

Testaamisen jälkeen dynaamiset taulut valittiin dataputkien peruskomponenteiksi tilanteissa, joissa laskettua kyselytulosta tarvitaan toistuvasti, mutta ensimmäisessä laskentakerrassa voi olla minuuttien tai kymmenien minuuttien viive. Suurin osa datatarpeista osuu tähän kategoriaan, jossa uusi data ei tarvitse olla välittömästi esiteltävissä ja ratkaisuna näihin tilanteisiin kehitettiin monivaiheiset dynaamisten taulujen prosessiketjut. Prosessiketjujen rakenteen voi nähdä kuvasta 7. Näiden avulla data käsitellään järjestelmällisesti, skaalautuvasti ja tehokkaasti, mutta ottaen kustannukset huomioon. Rakenne selkeyttää infrastruktuuria ja optimoi resurssien käyttöä erityisesti laskennallisesti vaativissa prosesseissa.



Kuva 7: Dynaamisten taulujen prosessiketju

Prosessissa on useita vaiheita ja jokaiselle raakataululle tehdään tietyt toimenpiteet, jotka varmistavat datan laadun. Nämä toimenpiteet suoritetaan kolmessa eri dynaamisen taulun kerroksessa. Ensimmäisessä kerroksessa varmistetaan datan eheys ja valmistellaan data liitoskyselyjä ja laskentaa varten. Valmisteluun kuuluu duplikaattien poisto ja yksinkertainen datan käsittely.

Toisessa kerroksessa siirrytään datan rikastamiseen, mikä on prosessin ydin. Tässä vaiheessa teemme liitoksia eri dynaamisten taulujen välille monimutkaisten kyselyjen suorittamiseksi, mikä mahdollistaa aiempaa syvällisemmän tiedon jalostamisen. Tämän vaihe muuttaa raakadatan asiakkaille helposti ymmärrettäväksi ja arvokkaaksi tiedoksi.

Kolmas vaihe prosessissa ei ole aina välttämätön, mutta arvokas tietyissä skenaarioissa. Tässä data supistetaan tai jalostetaan vastaamaan jotain tiettyä tarvetta. Esimerkiksi rajapinta tarvitsee vain muutamaa saraketta tietyillä ehdoilla jättimäisestä taulusta tai dataa täytyy rikastaa yksittäistä toimintoa varten, mutta se ei hyödytä mitään muuta toiminnallisuutta. Näissä tilanteissa voidaan tapauskohtaisesti luoda jatkojalostettu taulu, joka yksinkertaistaa tai nopeuttaa Back-End kehittäjän työtä.

5.3 Käyttöönotto ja testaus

Snowflakeen on nyt luotu kaikki tarpeelliset dynaamiset taulut, jotka käsittelevät datan toivottuun muotoon. Seuraavaksi on tärkeää varmistaa, että datan lopullinen muoto on varmasti eheää ja laadukasta. Voidaan todentaa tämä vertaamalla SQL Serverin ja Snowflaken tuottamia lopputuloksia keskenään. Koska tiedetään SQL Serverin datan olevan halutussa muodossa, samankaltaiset lopputulokset Snowflakessa osoittaisivat datan olevan luotettavaa. Haasteena on, että nämä kaksi tietokantajärjestelmää suorittavat laskennan eri tavoin, minkä vuoksi taulujen sarakkeita ei voi verrata yksi yhteen.

Toimeksiantajan SQL Server -ratkaisu on suunniteltu käsittelemään dataa omat vahvuudet ja heikkoudet huomioon ottaen. Tässä SQL Server toteutuksessa voi ilmetä haasteita suurten datamäärien reaaliaikaisessa käsittelyssä ilman merkittävää viivettä. Sovelluksen tarjoaman palvelun kannalta on kriittistä pitää latenssi mahdollisimman alhaisena, jotta käyttökokemus pysyy laadukkaana. Tästä syystä SQL Server on rakennettu suorittamaan suuret laskentaprosessit yöllä, kun tietokanta ei vastaanota sovellukselta pyyntöjä tai kehittäjät eivät tee muutoksia tietokantaan. Näin tietokanta voi keskittyä suurten laskentaprosessien suorittamiseen. Vaikka dataa rikastetaan mahdollisimman pitkälle yöajojen aikana, niin sovellus vaatii usein vielä funktioiden suorittamista. Tämä johtuu siitä, että jos kaikki mahdolliset parametrien yhdistelmät laskettaisiin valmiiksi tauluihin, niin taulujen rivimäärät kasvaisivat eksponentiaalisesti. Jokaisen yhdistelmän valmiiksi laskeminen on käytännössä mahdotonta. Sen takia SQL Serverin ratkaisu on rakennettu hyödyntämään funktioita, jotta laskenta tehdään vain, kun sovellus pyytää funktion suorittamista tiettyjen parametrien osalta.

Snowflaken skaalautuvuus ja laskennan tehokkuus mahdollistavat raskaiden laskentaprosessien suorittamisen vuorokauden ympäri. Dynaamisiin tauluihin voidaan laskea monimutkaisempia laskelmia ilman, että tarvitaan erillisiä funktioita, joita SQL Server vaatisi. Näin voidaan vähentää funktioiden määrää, mutta ei poistaa niiden tarvetta kokonaan. Snowflaken edullinen datan säilytys ja tehokas laskenta mahdollistaa kattavan ennalta laskennan. Tämä eroaa toimeksiantajan SQL Server ratkaisusta, joka hyödyntää suurimmaksi osaksi parametrisoituja funktioita. Funktioiden avulla tietokanta käsittelee dataa vain tarpeen tullen ja tietyllä rajauksella.

SQL Serverin ja Snowflaken arkkitehtuuriset erot tarkoittavat, että niiden valmiiksi laskettujen taulujen sarakkeet eivät välttämättä täsmää. Snowflakessa voi olla enemmän sarakkeita, jotka SQL Serverissä laskettaisiin tapauskohtaisesti. Tämän vuoksi suora vertailu ei ole mahdollista ja on kehitettävä useita menetelmiä lopputulosten testaamiseen ja vertailuun.

5.3.1 Sarakekohtainen testaaminen

Tässä luvussa esitellään eri menetelmiä datan laadun testaukseen, joilla pyritään varmistamaan datan eheys. Yksi tehokas menetelmä on sarakekohtainen vertailu, joka kohdistuu niihin sarakkeisiin, jotka ovat identtisessä muodossa molemmissa tietokannoissa. Tämä voidaan suorittaa koneellisesti ODBC-yhteyden kautta käyttämällä SQL Serverin Linked Server -palvelua yhteyden luomiseksi. On mahdollista luoda SQL Serveriin proseduureja, jotka vertailevat sarakkeiden sisältöä, rivien määriä tai päivämääriä. Esimerkiksi kehitin proseduurin, joka vertailee rivien määrää tietyn aikavälin sisällä molempien tietokantojen tauluissa. Jos SQL Serverin ja Snowflaken taulujen rivimäärät täsmäävät valitulla aikavälillä, testi katsotaan onnistuneeksi. Mikäli rivimäärät eroavat, testi epäonnistuu ja proseduuri lähettää testin tuloksen eteenpäin. On myös

mahdollista suorittaa testejä, joissa lasketaan yhteen tietyn aikavälin sisällä olevat numeeriset arvot. Jos tulokset poikkeavat toisistaan, datassa on selkeästi eroavaisuuksia.

On kuitenkin tärkeää huomioida, että pelkkä rivimäärän täsmääminen ei itsessään takaa datan eheyttä, sillä se voi olla myös yhteensattuma. Samoin pelkästään numeeristen arvojen yhteenlasku ei välttämättä kerro taulujen olevan identtisiä. Kuitenkin jos useampi tarkistussummalaskenta osoittautuu onnistuneeksi, todennäköisyys on suuri sille, että taulujen välinen data on yhdenmukaista.

5.3.2 Sovelluksen avulla testaaminen

Toinen datan eheyden testausmenetelmä suoritetaan luomalla testiversio sovelluksesta, joka hyödyntää Snowflake-tietokantaa. Testiversion tuomia arvoja verrataan vastaavan sovelluksen arvoihin, joka on yhdistetty SQL Serveriin. Tämä mahdollistaa arvojen vertailun tilanteissa, joissa suoraa sarakkeiden vastaavuutta ei ole tietokantojen tauluissa. Toisistaan eroavat taulut ilmenevät esimerkiksi, kun sovelluksen rajapinta pyytää tietokannasta jonkin funktion tulosta tietyillä parametreilla, kun taas toisessa tietokannassa arvo on jo valmiiksi laskettu. Tämän testausmetodin avulla voidaan varmistaa, että data on yhteneväistä, vaikka yksittäisten sarakkeiden vastaavuutta ei voida suoraan osoittaa. Kun samaa sovellusta eri tietokantojen yhteyksillä on tarpeeksi kattavasti testattu, niin voidaan todeta, että eri laskentalogiikat tuottavan saman ja oikean lopputuloksen.

Testausmenetelmien hyväksytyn suorituksen jälkeen voidaan vahvistaa dataputkien tuottavan laadukasta ja yhtenäistä dataa. Myös tulevaisuudessa uusien dataputkien tarve ilmenee väistämättä. Luvussa hahmoteltu prosessi auttaa ymmärtämään, mitkä kehitysmenetelmät ja erilaisiin tilanteisiin soveltuvat tietokantaobjektit, palvelevat organisaation tarpeita parhaiten. Dataputken luomisen jälkeen voidaan soveltaa testausmenetelmiä ja näin varmistaa uuden putken tuottavan korkealaatuista dataa. Tämä muodostaa kehitysprosessin, joka takaa tiedon laadun ja rikastusprosessin tehokkuuden. Prosessi tarjoaa vankan pohjan tulevaisuuden tietokantavaatimusten kehittämiseksi.

6 Snowflaken käyttöönoton infrastruktuuri

Projektin alkuvaiheessa, jo ennen kuin tietokantasiirtymä etenee kehitysvaiheeseen, on välttämätöntä tunnistaa ja toteuttaa joukko valmistelutoimenpiteitä. Näiden toimenpiteiden avulla pidetään huoli, että dataputkien kehitys on alusta alkaen systemaattista ja tehokasta. Keskeisiin valmistelutoimenpiteisiin kuuluu tehokas versionhallinta, selkeä roolihierarkia, ohjelmointiympäristön keskittäminen sekä CI/CD-prosessien luominen. Tämä lähestymistapa on tärkeä Snowflaken onnistuneeseen käyttöönottoon. Seuraavat alaluvut käsittelevät näiden toimintojen suunnittelua, toteutusta ja lopputulemaa.

6.1 Versiohallinta

Versionhallinta on keskeinen vaatimus tietokannan kehityksessä, tarjoten työkalun muutosten seurantaan ja dokumentointiin. Sen avulla luodaan selkeä ja läpinäkyvä muutoshistoria, joka yksinkertaistaa hallinnointia sekä varmistaa kaikkien päivitysten olevan turvallisesti palautettavissa. Hyvin implementoitu versionhallinta mahdollistaa virheiden nopean tunnistamisen ja korjaamisen palauttamalla järjestelmän virheettömään tilaan. Tämä lisää toimintavarmuutta ja luotettavuutta vähentämällä operatiivisia riskejä. On oleellista minimoida potentiaaliset riskit, vaikka tarvetta palautukselle hyvin luodussa kehitysprosessissa ei välttämättä koskaan syntyisikään. Versionhallinta on siten perusta tietokannan kestävälle kehitykselle ja ylläpidolle.

Versionhallinta on avainasemassa myös tiimien välisen yhteistyön tehostamisessa, erityisesti monikehittäjäprojekteissa, jossa työskennellään saman tietokannan parissa. Se ehkäisee tehokkaasti konflikteja tarjoamalla selkeät mekanismit muutosten hallintaan ja yhdistämiseen. Lisäksi versionhallintajärjestelmät mahdollistavat täydellisten tietokantaympäristöjen kopioiden luomisen, joissa kehittäjät voivat turvallisesti testata uusia ominaisuuksia ilman vaikutusta tuotantoympäristöön. Kattava testaus QA-ympäristössä (Quality assurance) ennen muutosten siirtämistä tuotantoon varmistaa toimintavarmuuden samalla vähentäen manuaalista työtä ja minimoiden inhimillisten virheiden mahdollisuutta.

Snowflake-tietokannalle valittiin sopiva versionhallintajärjestelmä arvioimalla huolellisesti organisaation keskeiset tarpeet. Tärkeinä kriteereinä ovat käyttäjäystävällisyys, matala oppimiskynnys ja yhteensopivuus Snowflaken kanssa. Arvioinnin jälkeen päädyttiin neljään potentiaaliseen ehdokkaaseen, joita ovat Flyway, Liquibase, DBT ja Schemachange, joilla kaikilla on omat etunsa ja haasteensa.

Huomioiden organisaation tarpeet ja tavoitteet versionhallinnan suhteen, päätös kallistui Schemachange-versionhallintajärjestelmän suuntaan. Tämä valinta perustui moniin tekijöihin, kuten Schemachangen selkeyteen ja käyttöönoton helppouteen, mitkä sen yksinkertainen toteutus Python-kirjastona mahdollistaa. Vaikka muut vertailussa olleet järjestelmät esittelivät laajan ominaisuusvalikoiman, Schemachange erottautui keskittymällä niihin avaintoimintoihin, jotka vastaavat tarkasti organisaation versionhallinnan keskeisiä vaatimuksia. Schemachange mahdollistaa uusien testiympäristöjen nopean pystytyksen, tietokannan palautuksen aikaisempiin versioihin ja kehitysmuutosten turvallisen siirron tuotantoympäristöön, mikä tekee siitä hyvän valinnan tukemaan kehitysprosesseja (Gupta 15.8.2021).

6.2 Ohjelmointiympäristön valinta

Keskitetyksi ohjelmointiympäristöksi valittiin Visual Studio Code, erityisesti sen Snowflake-laajennuksen vuoksi. Tämä laajennus sallii tietokantamuutosten tekemisen suoraan Visual Studio Coden kautta, välttäen näin Snowflaken oman käyttöliittymän tarpeen. Keskeinen syy ohjelmointiympäristön vaihtoon oli Visual Studio Coden saumaton integraatio Gitin ja Schemachangen kanssa, mikä keskittää kehitysprosessin yhteen paikkaan. Tämä yksinkertaistaa koodin muokkausta, validointia ja tallentamista vähentäen samalla manuaalista työtä ja inhimillisen virheen riskiä. Ilman tätä keskitettyä lähestymistapaa kehittäjien täytyisi tehdä muutokset Snowflaken käyttöliittymässä, siirtää koodi manuaalisesti Gitiin ja sen jälkeen Schemachangeen. Toisin sanoen kehittäjän tulisi kopioida sama koodi kolmeen eri paikkaan. Tämä ei ainoastaan ole aikaa vievää, vaan lisää myös unohduksien ja virheiden mahdollisuutta.

6.3 CI/CD prosessi

Keskittämällä lähdekoodin hallinnointi, Git ja Schemachange yhteen ohjelmointiympäristöön on jo otettu askel kohti saumattomampaa kehitysprosessia. Siitä huolimatta haasteeksi nousee se, että Gitin käyttö ja Schemachange-versiohallinnan päivitykset vaativat erilliset ajot. Nämä kaksi erillistä toimenpidettä nostaa riskiä siihen, että vahingossa kehittäjä päivittää vain toisen näistä, jättäen järjestelmän puutteelliseen tilaan. Ratkaisuna suunnittelin ja toteutin CI/CD-prosessin, jonka myötä kehittäjän viedessä muutokset Gittiin, Schemachange ajaa itsensä automaattisesti varmistaen, että koko kehitysketju on ajan tasalla ja synkronoitu. Tämän automatisoidun prosessin ansiosta pystytään minimoida manuaalisen työn määrä. Samalla potentiaalisten virheiden mahdollisuus laskee merkittävästi pienemmäksi, joka edistää kehitystiimin tehokkuutta ja koodin laatua.

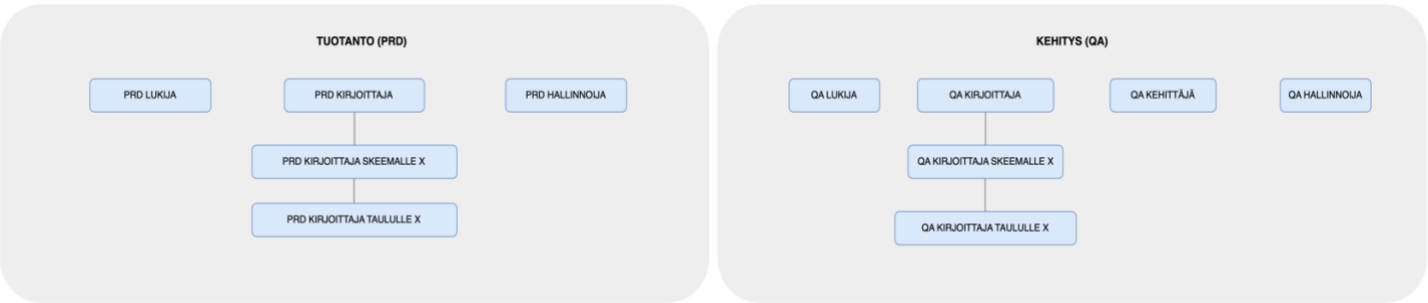
6.4 Roolihierarkia

Snowflakessa käyttöoikeuksien hallinta on toteutettu käyttäjä- ja roolipohjaisesti, mikä varmistaa turvallisen ja joustavan pääsyn tietokantaresursseihin. Käyttäjille myönnetään oikeudet tietyille rooleille ja tietovarastoille mahdollistaen pääsyn rajaamisen tarpeiden mukaiseksi. Roolit määrittelevät käyttäjän pääsyn tietokanta-, skeema- ja taulutasolla. Rooli määrittää myös, pystyykö sillä lukemaan, kirjoittamaan, muokkaamaan tai poistamaan tietokantaobjekteja.

Järjestelmällisen roolihierarkian suunnittelu on oleellista, jotta voidaan varmistaa, että käyttäjillä on vain tarvittavat käyttöoikeudet ja että roolien määrä pysyy hallinnassa. Ajan myötä, kun tietokanta kasvaa ja kehittyy, roolien määrä voi paisua ja niiden hallinta voi muuttua haasteelliseksi. Tämä voi johtaa sekavaan tilanteeseen, jossa roolien tarkoituksen ja tarpeellisuuden ymmärtäminen on vaikeaa ja rooleille voi kertyä liikaa oikeuksia.

Ratkaisuna tähän haasteeseen luotiin selkeästi määritellyt ylätason roolit, jotka jakavat oikeudet systemaattisesti tuotanto- ja kehitysympäristöjen kesken. On oleellista, että samalla roolilla ei ole mahdollisuutta muokata molempia ympäristöjä, varmistaen ympäristöjen välisen selkeän erottelun. Kummassakin kehitysympäristössä on sama roolirakenne, johon kuuluu lukija, kirjoittaja ja hallinnoija. Ainoana erona on, että tuotantoympäristöllä ei ole kehittäjäroolia, koska siihen ei tehdä suoria muutoksia. Tämä mahdollistaa oikeuksien tarkemman määrittelyn käyttötarkoituksen ja tarvittavan pääsyn tason mukaan.

Suurin osa käyttötarpeista mahtuu näiden roolien sisälle. Kuitenkin voidaan tarpeen mukaan määritellä näihin linkittyviä tytärrooleja, jotka ovat esimerkiksi skeema- tai taulukohtaisia. Näin jokainen rooli on räätälöity tarkasti tietylle käyttötarkoitukselle ja käyttäjäryhmälle, kuten kuvassa 8 on esitetty. Tämä lähestymistapa varmistaa, että käyttöoikeudet ovat aina ajantasaiset ja selkeät, eivätkä rönsyile ajansaatossa samalla tavalla kuin strukturoimaton toteutus.



Kuva 8: Roolihierarkian rakenne

7 Tulokset ja arviointi

Tässä luvussa käsitellään modernisointiprosessin saavutettuja tuloksia, hyötyjä, riskejä sekä jatkokehityksen askeleita. Hyötyjen osalta syvennytään siihen, mitä kaikkea projekti on tähän mennessä saavuttanut. Miten tietokantasiirtymä on parantanut tehokkuutta, ylläpidon sujuvuutta ja asiakaskokemusta. Tämän jälkeen siirrytään tutkimaan siirtymän riskejä yleisellä tasolla, kuinka on onnistuttu näiden riskien hallinnassa ja minkälaisia riskejä hankkeella mahdollisesti piilee tulevaisuudessa. Lopuksi käydään läpi koko projekti vaihe vaiheelta ja hahmotellaan, mitä kehitysaskeleita projektille on odotettavissa jatkossa.

7.1 Modernisointiprosessin arvo

SQL Server on melkein kymmenen vuoden ajan toiminut organisaation luotettavana tietokantaratkaisuna, tukien palveluiden monipuolistamista ja data- sekä asiakasmäärän kasvua. Kuitenkin, kuten pitkäikäisissä järjestelmissä ajan mittaan käy, vanhemman teknologian kyvykkyydet eivät enää vastaa organisaation tulevaisuuden kasvaviin tarpeisiin. Tämä on luonnollinen seuraus järjestelmän iästä, organisaation suunnan vaihdoksista ja teknologisten vaatimusten kasvusta.

Ajan myötä organisaation tavoitteet ja palvelutarjonta ovat kokeneet muutoksia, mikä on väistämättä heijastunut tietokantajärjestelmän rakenteeseen. Erilaisia toimintoja on kehitetty vastaamaan kulloinkin vallinneita tarpeita. Kuitenkin, kun nämä tarpeet muuttuvat tai jäävät taakse, järjestelmään kertyy toimintoja, jotka ovat menettäneet alkuperäisen merkityksensä tai käyttötarkoituksensa, aiheuttaen tarpeettomia monimutkaisuuksia. Näiden käyttämättömien ja muuttuneiden toimintojen kertyessä, niiden hallinnointi ja ylläpito monimutkaistuvat. Koodin määrä kasvaa, kun vanhoja komponentteja yritetään sopeuttaa uusiin tarkoituksiin, mikä saattaa johtaa epätehokkuuteen ja turhiin osiin järjestelmässä. Vaikka dokumentaation päivitykseen pyritäänkin säännöllisesti, vanhentuneita toimintoja koskevat jäänteet voivat jäädä dokumentaatioon, varsinkin kun organisaation strategiset suunnitelmat muuttuvat. Tämän seurauksena syntyy tilanne, jossa järjestelmän yksittäiset osat eivät välttämättä toimi optimaalisesti. Lisäksi järjestelmän hallinnointi muuttuu haasteellisemmaksi kehittäjille sekä erityisesti henkilöille, joille rakenne ja logiikka eivät ole ennestään tuttuja.

Kyseessä on väistämätön ilmiö, joka seuraa tietokantajärjestelmien luonnollista elinkaarta. Ei ole aina ilmeistä, mitkä toiminnot ovat jääneet pysyvästi tarpeettomiksi tai mikä osa dokumentaatiosta on vanhentunutta. Huolellisesta hallinnasta huolimatta, historialliset jäljet ja menneiden aikojen päätökset heijastuvat edelleen järjestelmässä. Tämä ei siis ole merkki huonosta ylläpidosta, vaan

pikemminkin todistusaineistoa siitä, kuinka tietokantajärjestelmän päivittäminen voi tuoda kaivattua selkeyttä ja järjestelmällisyyttä infrastruktuuriin.

Modernisointiprosessin suurin arvo piilee mahdollisuudessa uudelleenarvioida, mitkä toiminnallisuudet ovat todella tarpeellisia. Kehittäjillä on tilaisuus kirjoittaa dokumentaatio uudelleen ja ottaa käyttöön uusia toiminnallisuuksia, joita vanha teknologia ei mahdollistanut. Lisäksi voidaan hyödyntää uusimpia kehitysmenetelmiä, jotka eivät olleet saatavilla alkuperäistä järjestelmää luodessa. Tämä lista tuo esiin kaikki hyödyt, joita modernisointiprosessi on tuonut organisaatiolle.

- Operatiivisen tehokkuuden parannus:
    - Tarpeettomien toimintojen karsiminen ja prosessien virtaviivaistaminen
    - Tietokannan koodin yhtenäistäminen pääasiassa SQL- ja Python -ohjelmointikieliin
    - Dokumentaation uudelleenkirjoitus ja ajantasaistaminen
- Teknologiset parannukset:
    - Tallennuskapasiteetin ja laskentatehon kasvattaminen
    - Skaalautuva laskentateho
- Ylläpidon tehostaminen:
    - Snowflaken uniikit toiminnot helpottavat dataputkien hallinnointia
    - Skaalautuva laskentakyky lisää järjestelmän luotettavuutta datapiikkien kohdalla
    - Infrastruktuurin ylläpidon ulkoistaminen
- Asiakaskokemuksen parannus:
    - Sovellusten suurien laskentojen vasteajan lyhentyminen
    - Palveluvalikoiman laajentuminen uuden teknologian myötä
- Integraatioarkkitehtuurin yksinkertaistaminen
    - Snowflake mahdollistaa erilaisia tiedonsiirtotapoja, jotka vähentävät erillisten integraatioiden tarvetta
- Automatisointi:
    - Snowflake kehittää aktiivisesti uusia ominaisuuksia, jotka mahdollistavat prosessien automatisointia

Dataputkien, funktioiden, proseduurien ja muun tietokantaan liittyvän logiikan kriittinen tarkastelu ja arviointi mahdollistaa tarpeettomien toiminnallisuuksien karsimisen. Tämä selkeyttää ja tehostaa koodipohjaa. Yhtenäistämistä auttoi myös tietokannassa ohjelmointikielien määrän vähentäminen kahteen. Tämä pienentää ohjelmointikielien osaamisvaatimuksia ja varmistaa, että kaikki kehittäjät pystyvät tarkastelemaan jokaista ympäristössä olevaa toimintoa. Uuden tietokantajärjestelmän

luomisessa avautuu myös mahdollisuus kirjoittaa dokumentaatio uudelleen. Laadukas ja helposti saatavilla oleva dokumentaatio paljastaa kehitysprosessin logiikan ja tavoitteet, tarjoten tuleville kehittäjille ymmärryksen järjestelmän rakenteesta ja sen kehityshistoriasta. Tieto siitä, miksi nykyiset toiminnot luotiin on yhtä oleellista kuin itse tekninen toteutus kokonaisuuden hahmottamisessa ja jatkokehityksessä.

Tietokantajärjestelmän kyvykkyyttä tarkastellessa Snowflake on tuonut merkittäviä edistysaskeleita organisaation tietokannan tallennuskapasiteettiin ja laskentatehoon. Modernisaatio mahdollistaa asiakkaiden toivomien uusien palveluiden kehittämisen, joiden pullonkaulaksi ilmeni aikaisemmin SQL Serverin laskentateho. Uusien palveluiden kehittämisen lisäksi laskentateho tarjoaa myös ylläpidon toimintavarmuutta. Siirtymä ratkaisee aiemmin SQL Server -toteutuksen kanssa kohdatut haasteet, jossa kasvanut datan määrä ja laskentatehon rajallisuus johtivat pitkiin laskentaprosesseihin. Pitkät laskennat täytyi ajaa yön aikana ja riskinä oli, että jos laskenta epäonnistuu, viivästykset datan päivityksessä nousee useisiin tunteihin, koska ajo pitää suorittaa uudestaan. Nykyään suurimmatkin laskentaprosessit voidaan suorittaa minuuteissa, tarjoten varmuutta datan käsittelyyn tilanteissa, jossa virheitä saattaa ilmetä.

Snowflaken dynaamiset taulut ovat yksinkertaistaneet ylläpitoa merkittävästi. Ne vähentävät tarvittavan koodin ja komponenttien määrää ja helpottavat seurantaa, joko yksittäisinä objekteina tai kokonaisuutena.

On tärkeää kuitenkin huomioida, että SQL Server relationaalisena tietokantana saattaa tarjota yhtä nopeat tai lyhyemmät vasteajat yksinkertaisissa kyselyissä. Snowflake erottuu kuitenkin positiivisesti, kun kyseessä on vaativa laskentaprosessi. Raskaat kyselyt ovat nopeampia Snowflakessa kuin SQL Serverissä, jolloin käyttöliittymän sivujen latausajassa ei ole enää merkittäviä eroja.

Lisäksi siirtymä on nostanut tietokantaratkaisun yhteensopivuutta muiden organisaatioiden kanssa ja vähentänyt erillisten integraatioiden tarvetta. Monipuoliset tiedonsiirtomahdollisuudet tekevät Snowflake tilien välisestä tiedonjaosta saumatonta. Tiedonjakamisen kyvykkyydet tuovat uusia mahdollisuuksia tehdä yhteystyötä nykyisten ja tulevien asiakkaiden kanssa erottaen organisaation muista. Snowflake myös jatkuvasti kehittää uusia ominaisuuksia, kuten automatisointia tukevia toimintoja. Esimerkkinä toimii private preview -vaiheessa oleva tekoälyä hyödyntävä PDF-lukija, jonka käyttöönotto vähentäisi manuaalisen työn tarvetta. Lukuisat kehitteillä olevat ominaisuudet ilmentävät Snowflaken tahtotilaa pysyä ajan tasalla tietokantateknologian kehityksen kanssa samalla tuoden uusia mahdollisuuksia organisaation liiketoimintatarpeille.

7.2 Modernisointiprosessin riskit

Siirtyminen vanhasta järjestelmästä uuteen sisältää suurempia riskejä verrattuna tavanomaisiin ohjelmistoprojekteihin. Uudelta tietokantajärjestelmältä odotetaan vähintään samantasoisia teknisiä ominaisuuksia kuten laskentatehoa ja tallennuskapasiteettia kuin mitä aikaisempi järjestelmä tarjoaa. Jos aiempi järjestelmä ei ole vanhentunutta Legacy-arkkitehtuuria niin jokaisen suorituskyvyllisen osa-alueen parantaminen on paljon vaadittu uudelta järjestelmältä. Tämän lisäksi uusi tietokantajärjestelmä täytyy olla integroitavissa olemassa oleviin prosesseihin ja sen on pystyttävä tekemään kaikki samat toiminnot mitä vanha järjestelmä. Modernisointiprosessi on luonteeltaan monimutkainen ja riskejä sisältävä kokonaisuus, koska vanha järjestelmä on muovannut kapean muotin, mihin uuden järjestelmän tulee mahtua.

Seuraavaksi esittelen listan tyypillisistä tietokantasiirtymän haasteista, jotka otettiin huomioon modernisointiprojektissa.

- Yhteensopivuusriskit
    - Toiminnot eivät ole rakennettavissa uudessa järjestelmässä
    - Integraatio uuden ja vanhan järjestelmän välillä mahdoton
    - Uudelleen rakentamisen epäonnistuminen vanhan järjestelmän monimutkaisuuden takia
- Suorituskykyriskit
    - Liian suuri viive käyttöliittymässä uudesta järjestelmästä
    - Laskentatehon riittämättömyys monimutkaisissa laskentaprosesseissa
- Valmisteluriskit
    - Liian vajavainen taustatyö ja suunnittelu
    - Resurssien riittämättömyys

Tietokantasiirtymän vaativan luonteen vuoksi lähestyimme modernisointiprosessia kattavalla suunnittelulla ja vaiheittaisella toteutuksella. Vaiheisiin kuuluu Proof of Concept, Minimal Viable Product ja hybridikehitys. Tämän lähestymistavan perustana on mahdollisuus tunnistaa ja ratkaista kriittiset haasteet prosessin eri vaiheissa. Jokainen vaihe on suunniteltu varmistamaan, että prossissa edetään mahdollisimman tehokkaasti samalla riskejä minimoiden.

Organisaatio on päässyt modernisaatioprosessissa hybridikehitysvaiheeseen ja onnistunut ratkaisemaan suurimman osan listan riskeistä. "Proof of Concept" -vaiheen myötä saatiin viitteitä suorituskyvyn riittävyydestä. Vaiheessa osoitettiin, että järjestelmä kykenee suorittamaan

monimutkaisia laskentaprosesseja minimaalisella viiveellä. Yhteensopivuusriskit puolestaan ratkaistiin "Minimal Viable Product" -vaiheessa, jossa luotiin toimivat dataputket uuteen järjestelmään ja “Streamer” nimisen yhteyden SQL Serverin ja Snowflaken välille. Streamer osoitti, että vanhan ja uuden järjestelmän saa tehokkasti integroitua keskenään. Valmisteluriskit eivät myöskään nousseet akuutiksi huolenaiheeksi ja molemmat kehitysvaiheet etenivät suunnitelman mukaisesti, josta voi päätellä modernisointiprosessiin valmistautumisen olleen kattava.

7.3 Jatkokehitys

Tarjotakseni selkeän kuvan jatkokehitysaskelista esittelen ensin listauksen kaikista vaiheista tietokantasiirtymäprosessissa, jotka on tuonut projektin nykyiseen pisteeseen. Tiivistys projektin vaiheista auttaa ymmärtämään, mitä kaikkea olemme saavuttaneet tähän mennessä ja mitä tulevaisuudessa täytyy tehdä, jotta voimme onnistuneesti asettaa projektin päätökseen.

- Snowflake-ympäristön valmistelu:
    - **Tietokantaversionhallinta:** Mahdollistaa palautuksen, samanaikaisen kehityksen ja ympäristöjen ajantasaisuuden hallinnoinnin
    - **Roolihierarkia:** Systematisoitu oikeuksien hallinta
    - **Ohjelmointiympäristön keskitys:** Mahdollistaa kaikkien toimenpiteiden tekemisen samassa ohjemointiympäristössä
    - **CI/CD-prosessit:** Vähentää virheiden riskiä ja manuaalista työtä
- Dataputkien Rakentaminen:
    - **Käytännön testaus:** Eri dataputkimenetelmien testaus organisaation tarpeiden mukaan
    - **Dataputkien toteutus:** Parhaiden käytäntöjen pohjalta rakennetut dataputket, jotka tukevat keskeisiä laskentaprosesseja
    - **Streamer:** Dataputki SQL Server ja Snowflake tietokantojen välille
    - **Rajapinta:** Snowflaken ja organisaation tuotteen välille luotu rajapinta, joka mahdollistavat datan suoran haun Snowflakesta

Listauksen työvaiheet ovat valmiita. Nykyisessä pisteessä ollaan tilanteessa, jossa Snowflake on valmis toimimaan tietokantana organisaation erilaisille palveluille. On jo onnistuneesti luotu yksi tuote, joka käyttää Snowflakea tietokantanaan hakien ja vieden siihen tietoa. Nyt organisaatio määrittelee seuraavan osa-alueen sovelluksesta, joka tullaan siirtämään Snowflaken päälle. Tämä prosessi jatkuu samalla kaavalla, kunnes skaalautuvaa laskentaa tarvitsevat palvelut ovat

integroitu uuden tietokantajärjestelmän kanssa. Lopullisena tavoitteena on siirtää kaikki raskasta tiedonkäsittelyä tai laskentaa vaativat toiminnot Snowflakeen.

Vaikka merkittäviä etappeja on saavutettu, prosessi on edelleen kesken ja haasteiden esiintyminen kuuluu osana tämän luonteiseen hankkeeseen. Seuraavaksi keskitytään tunnistamaan, mitä riskejä voidaan kohdata vielä tässä vaiheessa projektia ja kuinka näihin haasteisiin vastataan.

On päästy modernisointiprosessin hybridikehitysvaiheeseen, jossa integroidaan sovelluksen yksittäiset palvelut Snowflake-tietokantaan rajapintojen kautta ja nämä sovelluksen palvelut tallentavat ja pyytä tietoa kannasta. Koko sovelluksen laajamittainen siirtyminen uuden tietokannan käyttöön on kuitenkin vielä näkemättä. Mahdollisuutena on, että ylläpitokustannukset ovat laskettua suuremmat uudessa järjestelmässä. On kuitenkin useita erilaisia tapoja, miten pienentää suurten ylläpitokustannusten riskiä.

Kustannusten tehokas hallinta edellyttää optimaalisen viiveen määrittämistä, joka yhdistää käyttäjätyytyväisyyden ja kustannustehokkuuden. Dataputket voidaan luoda käytännössä välittömiksi. Jatkuva tarve ajantasaisuuteen kuitenkin lisää Snowflake-tietokannan käyttämien tietovarastojen määrää ja kokoa, mikä nostaa kustannuksia merkittävästi. Välittömän datan tarjoamisen kustannukset eivät välttämättä lisää palvelun arvoa odotetulla tavalla. Ratkaisuna tähän palvelua kehittäessä on oleellista suunnitella infrastruktuuri muotoon, jossa havaitaan tilanteet, missä viiveen vähäisyys tuottaa paljon hyötyjä sekä tilanteet, jossa viive ei ole erityisen haitallista. Dataputkia suunnitellessa on keskeistä arvioida, täytyykö tiedon olla saatavilla välittömästi vai onko minuuttien tai tuntien viive hyväksyttävä.

Viiveen optimoinnin ohella on oleellista rakentaa infrastruktuuri tavalla, joka maksimoi aktiivisten tietovarastojen käytön. Snowflaken hinnoittelumalli perustuu tietovaraston aktiiviseen käyttöaikaan, joten tietovarastoista veloitetaan vain päälläoloajalta. Snowflake määrittää automaattisesti tietovaraston aktiivisuuden kestoksi kymmenen minuuttia. Tavoitteena on pienentää tämä ajanjakso minimiin, joten asetetaan päälläoloaika yhteen minuuttiin. Tämä mahdollistaa kustannussäästöt lyhyemmissä kyselyissä, erityisesti kun suoritetaan harvemmin toistuvia tehtäviä. Asetuksen muutos tuo jo yksinään merkittäviä säästöjä, mutta ei ratkaise haastetta.

Lisäksi on tärkeää optimoida tietovarastojen käyttö siten, että niiden aktiivinen käyttöaika hyödynnetään täysimääräisesti. Esimerkiksi jos tehtävä, joka aktivoi tietovaraston, ajoitetaan minuutin välein ajettavaksi, ja kunkin tehtävän suoritus kestää viisi sekuntia, käytännössä tietovarastoa hyödynnetään vain kaksi tuntia vuorokaudessa. Vaikka tietovarastoa veloitettaisiin kahdenkymmenenneljän tunnin käytöstä, todellinen hyödyntämisaika jää huomattavasti lyhyemmäksi. Viidenkymmenenviiden sekunnin hukka voi itsessään tuntua pieneltä, mutta näistä

pienistä kuluista voi syntyä merkittävä vaikuttaja, kun kokonaisuus on tarpeeksi suuri. Vaikka tällä hetkellä toimeksiantaja on implementoinut vasta muutamia osia sovelluksesta, jotka hyödyntävät Snowflakea, on tärkeää suunnitella tehokas rakenne ajoissa. Näin ylläpitokulut eivät kasva liiallisesti siirtymän loppupuolella. Kuten taulukosta numero 1 ilmenee, Snowflaken tietovarastojen hinta kreditteinä tunnissa kaksinkertaistuu melkeinpä jokaisen kokoluokan noustessa. Tämä taulukko havainnollistaa hintarakenteen progressiivista kasvua pienimmästä XS-koosta aina suurimpaan 6X-L-kokoon.

Tietovaraston tyyppi	XS	S	M	L	XL	2X-L	3X-L	4X-L	5X-L	6X-L
Hinta kreditteinä/ tunti	1	2	4	6	16	32	64	128	256	512

Taulukko 1: Snowflake tietovaraston tuntihinnoittelu kreditteinä (Snowflake documentation s.a. d)

Tavoitteena ylläpitokustannusten optimoinnissa on kyselyjen ja prosessien aikataulutus niin, että ne toimivat peräkkäin saman tietovaraston aktiivisen käyttöjakson aikana. Tämä tarkoittaa, että pyritään yhdistämään useita lyhyitä kyselyjä yhdeksi jonoksi, välttäen niiden erillistä suorittamista. Tämä menetelmä on teoriassa yksinkertainen, mutta kohtaa käytännön haasteita, sillä monet kyselyt on suoritettava tietyn ikkunan sisällä. On mahdotonta toteuttaa kaikki tietokantakyselyt yhdessä lyhyessä aikaikkunassa, minkä jälkeen tietovarastoja ei tarvitsisi käyttää loppupäivänä. Ratkaisuna pyritään kehittämään funktiot, proseduurit ja dataputket jonomuodossa, jolloin yhden ajon jälkeen seuraava aktivoituu automaattisesti. Tämä lähestymistapa muuttaa lyhyet erilliset tehtävät pidemmäksi ja yhtenäiseksi toimintajaksoksi, tehostaen tietovaraston käyttöä.

Lisäksi tavoitteena on keskittää mahdollisimman paljon kuormituksesta yksittäiselle tietovarastolle, sillä näin voidaan maksimoida yhden tietovaraston käyttökapasiteetti. Keskittämällä kuormitusta pystytään minimoimaan päälläoloaikaa, jolloin tietovarasto ei suorita laskentaprosessia, edistäen kustannustehokkuutta sekä resurssien optimointia.

Modernisointiprosessi jatkuu vaiheittaiseen tapaan yksi palvelu kerrallaan, mutta tätä tehdessä on muistettava jatkaa tietokantaympäristön kehitystä. Snowflake on kehittämässä ja julkistamassa useita uusia ominaisuuksia. Tietokantasiirtymäprosessi etenee vaiheittaisella toteutuksella, mutta tämän yhteydessä on tärkeää pysyä tietoisena tietokantamaailman kehityksestä. Tämän takia jatkokehitykseen kuuluu oleellisena osana myös uusien toimintojen tutkinta, testaaminen ja parhaassa tapauksessa käyttöönotto. Esimerkkinä toimii Snowflaken useampi markkinoille ilmestyvä tekoälypohjainen toiminto, jotka saattavat antaa mahdollisuuden laajemmalle automaatiolle, vähentäen samalla manuaalista työtä ja ylläpitoa. Uuden tietokantajärjestelmän

käyttöönotto on vaatinut huomattavia ponnistuksia, ja tämän jatkuvalla kehittämisellä on tavoitteena vastata kasvaviin tiedonkäsittelytarpeisiin niin pitkään kuin mahdollista. Näin voimme varmistaa pitkän elinkaareen tuoreelle järjestelmälle.

7.4 Oma oppiminen

Tämän toiminnallisen opinnäytetyön myötä olen kokenut merkittävää ammatillista kehittymistä sekä teknisellä että akateemisella tasolla. Akateeminen kehitykseni käsittää asiatekstin kirjoittamisen taidon, tietolähteiden kriittisen arvioinnin, aihealueen tarkemman rajauksen sekä itsenäisen työskentelyn hallinnan. Opinnäytetyön edetessä nämä taidot heijastuvat työssäni siten, että uusi teksti on tiiviimpää, laadukkaiden lähteiden valinta nopeampaa ja kyky rakentaa selkeämpiä lukuja ja kappaleita kehittyy.

Teknisen kehityksen osalta keskityn tietokantajärjestelmien perustoimintojen hallintaan, eri tietokanta-arkkitehtuurien erojen ymmärtämiseen sekä tietokantasiirtymäprosessin kokonaisvaltaiseen hahmottamiseen. Nämä kolme aluetta muodostavat laajoja kokonaisuuksia. Tietokantajärjestelmien perustoimintojen ymmärtäminen kattaa tutustumisen ja ymmärryksen erilaisista tietokantaobjekteista. Tietokanta-arkkitehtuurien syventämisessä tutkin SQL Serverin ja Snowflaken toiminnan perusteita, mikä edesauttaa nopeiden ja kustannustehokkaiden ratkaisujen kehittämistä dataputkien, kyselyiden ja funktioiden osalta. Tietokantasiirtymäprosessin hallinta sisältää analyysien tekemisen, siirtymän valmistelun, tietokantaympäristön valmistelun, vaiheittaisen toteutuksen, uusien dataputkien kehittämisen, niiden testaamisen ja laadukkaan datan toimittamisen.

Työskentely suuren projektin parissa on tuonut esiin muitakin keskeisiä näkökulmia, kuten projektinhallinnan merkityksen, tiimityöskentelyn tärkeyden, tarkan suunnittelun ja selkeiden tavoitteiden asettamisen. Näiden taitojen kehittyminen on ollut oleellista projektin onnistumiselle, ja ne ovat osoittautuneet yhtä tärkeiksi kuin teknisen asiantuntemuksen syventäminen. Projektinhallinta on vaatinut jatkuvaa priorisointia, resurssien tehokasta jakamista ja joustavuutta muuttuvien olosuhteiden edessä. Tiimityöskentelyn ohella olen oppinut arvostamaan siirtymäprojektiin osallistuneiden syvää ja eri osa-alueisiin keskittynyttä asiantuntijuutta. Selkeiden tavoitteiden asettaminen on puolestaan varmistanut, että kaikki tiimin jäsenet työskentelevät yhteisten päämäärien eteen, pitäen projektin aikataulussa.

Vaikka opinnäytetyöprosessi on tarjonnut arvokkaita oppimiskokemuksia, ne eivät ole tulleet helpolla. Työskennellessäni ensimmäistä kertaa Snowflake-alustan ja laajan tietokantasiirtymäprojektin parissa, olen kohdannut useita haasteita. Kukin haaste on vaatinut perusteellista tutkimusta ja kokeiluja, jotka eivät ole aina heti johtaneet optimaalisiin ratkaisuihin.

Tämä prosessi on opettanut minulle kärsivällisyyttä sekä kyvyn sopeutua muuttuviin suunnitelmiin. Olen kiitollinen niille, jotka ovat tukeneet minua matkallani, ja odotan innolla, miten voin hyödyntää näitä oppeja tulevaisuuden hankkeissani.

Lähteet

Araujo, G. 21.3.2023. SQL Server vs. Snowflake: The Data Warehouse Smackdown. Medium. Blogi. Luettavissa: https://medium.com/ai-genesis/choosing-snowflake-vs-sql-server-for-a-data-warehouse-%EF%B8%8F-9c9b0d6b1115. Luettu: 10.2.2024.

Assaf, W., D'Antoni, J., Davidson, L., Goguri, D., Longoria, M., Noble, E., West, R., & Zacharias, M. 2023. SQL Server 2022 Administration Inside Out. Microsoft Press. E-kirja. Luettu: 10.02.2024.

Avila, J.K. 2022. Snowflake: The Definitive Guide. O'Reilly Media. E-kirja. Luettu: 15.02.2024.

Azure Migrate Documentation 2024. Migrate machines as physical servers to Azure. Luettavissa: https://learn.microsoft.com/en-us/azure/migrate/tutorial-migrate-physical-virtual-machines. Luettu: 02.02.2024.

Bell, F., Chirumamilla, R., Joshi, B.B., Lindstrom, B., Soni, R., & Videkar, S. 2022. Snowflake Essen-tials: Getting Started with Big Data in the Cloud. Apress. E-kirja. Luettu: 20.01.2024.

Birchall, C. 2016. Re-Engineering Legacy Software. Manning Publications. E-kirja. Luettu: 20.1.2024.

Dremio s.a. Indexing and Partitioning. Luettavissa: https://www.dremio.com/wiki/indexing-and-partitioning/. Luettu: 08.04.2024.

Fanelli, T.C., Simons, S.C. & Banerjee, S. 2016. A Systematic Framework for Modernizing Legacy Application Systems. IEEE. Elektroninen tietoaineisto. Luettu: 14.10.2024.

Gupta, R. 15.08.2021. Database Change Management Tool: SchemaChange with Snowflake Overview. Medium. Blogi. Luettavissa: https://blog.devgenius.io/database-change-management-tool-schemachange-with-snowflake-overview-b62dec744e0a. Luettu: 05.04.2024.

Hussain, S.M., Bhatti, S.N. & Rasool, M.F.U. 19.-21.4.2017. Legacy system and ways of its evolution. IEEE. Elektroninen tietoaineisto. Luettu: 12.01.2024.

IBM Documentation 2023b. Database performance and query optimization. Luettavissa: https://www.ibm.com/docs/en/i/7.3?topic=database-performance-query-optimization. Luettu: 08.04.2024.

IBM documentation. 2023a. Database objects. Luettavissa: https://www.ibm.com/docs/en/db2/11.1?topic=administration-database-objects. Luettu: 07.04.2024.

IBM Topics. s.a. What is a data pipeline? Luettavissa: https://www.ibm.com/topics/data-pipeline. Luettu: 07.04.2024.

Influxdata. s.a. Snowflake vs SQL Server. Luettavissa: https://www.influxdata.com/comparison/snowflake-vs-sqlserver/. Luettu: 25.02.2024.

Isomäki, S. Milloin tietokanta-alustan modernisointi on perusteltua? Luettavissa: https://dbproservices.fi/data-platform/milloin-tietokanta-alustan-modernisointi-on-perusteltua/. Luettu: 1.1.2024.

Kumar, R. 22.10.2023. Difference between SQL Functions and Stored Procedures. Medium. Blogi. Luettavissa: https://medium.com/javatute-blog/difference-between-sql-functions-and-stored-procedures-dcbadbb2fb82. Luettu: 07.04.2024.

Leonard, S. 2023. Cost-Effective Data Pipelines. O'Reilly Media. E-kirja. Luettu: 07.04.2024.

McQuillan, M. 2015. Introducing SQL Server. Apress. E-kirja. Luettu: 04.02.2024.

Medium 2020. How to Automate Data Pipelines with Snowflake Streams and Tasks. Luettavissa: https://medium.com/slalom-data-ai/how-to-automate-data-pipelines-with-snowflake-streams-and-tasks-2b5b77ddff6a. Luettu: 22.02.2024.

Mukhtiar, M. 6.4.2023. Snowflake vs SQL Server: Choosing the Right Data Warehouse for Your Needs. Medium. Blogi. Luettavissa: https://medium.com/@rao.mohsin.54/snowflake-vs-sql-server-choosing-the-right-data-warehouse-for-your-needs-137f27bea85d. Luettu: 25.02.2024.

Nasir, R., Salam, M., ShahSani, R.K., & Alam, F. 2023. Analysis of Risks in Re-Engineering Software Systems. Research Gate, 73, 11. Luettavissa: https://www.researchgate.net/publication/272864256_Analysis_of_Risks_in_Re-Engineering_Software_Systems. Luettu: 25.1.2024.

Oracle. s.a. What Is a Database? Luettavissa: https://www.oracle.com/database/what-is-database/. Luettu: 07.04.2024.

Sinha, T. 16.3.2021. OLAP vs. OLTP: What's the Difference? IBM Blog. Blogi. Luettavissa: https://www.ibm.com/blog/olap-vs-oltp/. Luettu: 07.04.2024.

Snowflake. s.a. Why and How to Use Consumption Model Pricing for SaaS and Cloud Services. Luettavissa: https://www.snowflake.com/trending/why-and-how-use-consumption-model-pricing/. Luettu: 27.02.2024.

Snowflake documentation s.a.a. Key Concepts & Architecture. Luettavissa: https://docs.snowflake.com/en/user-guide/intro-key-concepts. Luettu: 22.02.2024.

Snowflake documentation s.a.b. Sharing data securely across regions and cloud platforms. Luettavissa: https://docs.snowflake.com/en/user-guide/secure-data-sharing-across-regions-platforms. Luettu: 29.02.2024.

Snowflake documentation s.a.c. Dynamic tables. Luettavissa: https://docs.snowflake.com/en/user-guide/dynamic-tables-about. Luettu: 26.02.2024.

Snowflake documentation s.a.d. Understanding compute cost. Luettavissa: https://docs.snowflake.com/en/user-guide/cost-understanding-compute. Luettu: 27.02.2024.

Seacord, R. C., Plakosh, D. & Lewis, G. A. 2003. Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. Addison-Wesley Professional. E-kirja. Luettu: 21.02.2024.

Terra, J. 2023. SQL Server Architecture Explained. Luettavissa: https://www.simplilearn.com/what-is-microsoft-sql-server-architecture-article. Luettu: 10.02.2024.

Timescale s.a. Building a Scalable Database. Luettavissa: https://www.timescale.com/learn/building-a-scalable-database. Luettu: 08.04.2024.

Vasconcelos, J. B. de, Kimble, C., Carreteiro, P. & Rocha, Á. 2017. The application of knowledge management to software evolution. Sciencedirect, 37, s. 1499–1506.