LAB University of Applied Sciences

Pilvipohjaisten analyyttisen datan tietokanta-palvelujen vertailu,

Case: TPC-H-tietojoukko ja ADA-automatisointipalvelu

LAB-ammattikorkeakoulu
Insinööri (YAMK)
2024
Tuomo Valtari

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Tuomo Valtari	Opinnäytetyö, YAMK	Syksy 2024
	Sivumäärä 66	

Työn nimi

Pilvipohjaisten analyyttisen datan tietokantapalvelujen vertailu,

Case: TPC-H-tietojoukko ja ADA-automatisointipalvelu

Tutkinto ja koulutusala

Insinööri (YAMK), IoT:stä tekoälyyn

Toimeksiantajaorganisaatio (jos opinnäytetyöllä on toimeksiantaja)

Epical Finland Oy

Tiivistelmä

Tiedon hyödyntämisen tarve on noussut organisaatioissa keskiöön. Samalla vain laadukkaasti tuotettu, ajantasainen ja luotettava tieto voi toimia osana päätöksentekoa. Datan määrän ja lähteiden kasvaessa tavanomainen tietovarastointiratkaisu ei enää välttämättä palvele tai tue kaikkia analytiikan käyttötapausten tarpeita. Pilvipohjaiset palvelut ovat tuoneet modernin vaihtoehdon tavanomaisille ratkaisuille ratkaisten uuden tyyppisiä vaateita.

Tämän opinnäytetyn tarkoitus oli tutkia markkinan tunnustamia, pilvipohjaisia analyyttisen datan tietokantapalveluja tietovaraston tavanomaisilla työnkuormilla ja vertailla niitä muun muassa raa'an kyselysuorituskyvyn ja hinnoittelun osalta, mutta laajemmin myös niiden ominaisuuksien ja käytettyjen teknologioiden osalta. Työssä käytetään TPC-H-tietojoukkoa pohjana standardisoidulle tietomallille sekä joukolle tietokantakyselyitä ja ADA-automatisointipalvelua dataputkien luontiin ja automatisoitujen työnkulkujen suorittamiseen. Tutkimuksessa käytettiin vertailuanalyysia ja -arviointia vertailujen ja mittausten pohjaksi sekä konstruktiivista tutkimusotetta tukena organisaatioille esimerkiksi tietovarastoinnin ratkaisun ja dataputkien rakennuksessa.

Asiasanat

Pilvipohjainen tietokantapalvelu, dataputket, työnkulkujen automatisointi, suorituskykytestaus

Abstract

Author(s)	Type of Publication	Published
Tuomo Valtari	Master's Thesis	Autumn 2024
	Number of Pages	
	66	

Title of Publication

Comparison of cloud based analytical database services,

Case: TPC-H dataset and ADA automation tool

Degree, Field of Study

Master of Engineering, IoT to Artificial Intelligence

Organisation of the client (if the thesis work is commissioned by another party)

Epical Finland Oy

Abstract

In today's organizations, the efficient utilization of data plays a vital role. At the same time, only well-produced, up-to-date, and reliable data can be used for decision-making. However, as data volumes and sources continue to grow, traditional data warehousing solutions may fall short in supporting the various needs of analytical use cases. Cloud-based services present a modern alternative, addressing the challenges posed by increasing data volumes and complexity.

This thesis aimed to study some of the leading cloud-based analytical database services for their suitability for typical data warehouse workloads. The comparison focuses on raw performance, service features, pricing, and underlying technologies. The study uses the TPC-H dataset to establish a standardized model and as a basis for a set of database queries. Additionally, an automation tool, ADA, is utilized to create the data pipelines and run them in automated workflows. The study used comparative analysis and evaluation as the basis for comparisons and measurements, as well as a constructive research approach to help organizations address the diverse needs of data warehousing and pipeline deployments.

Keywords

Cloud warehouse, data pipelines, workflow automation, benchmarking

Sisällys

1 Johdanto 1
1.1 Toimeksiantajan esittely 1
1.2 Työn tausta ja tavoitteet 2
1.3 Tutkimuskysymykset ja rajaus 2
1.4 Tutkimusmenetelmät 3
2 Tiedon käsittelytavat, varastointi ja mallinnus 4
2.1 Datan ja tiedon määrittely 4
2.2 Dataputket ja datan orkestrointi 4
2.3 Tietoallas 6
2.4 Delta Lakehouse -konsepti 7
2.5 Medallion -arkkitehtuuri 9
2.6 Tietovarastot 9
2.7 Tietovarasto vai Data Lakehouse – vai jotain siltä väliltä? 10
2.8 Tietovarastointi ja liiketoimintatiedon järjestelmät 12
2.9 Dimensionaalinen mallinnus 13
2.10 Tähti- ja lumihiutalemalli 14
3 Palvelut 15
3.1 Pilvipohjaiset tietovarastopalvelut 15
3.1 Analytics Development Accelerator (ADA) 15
3.2 Snowflake 17
3.3 Databricks 18
3.4 Azure Synapse Analytics 20
3.5 Microsoft Fabric 21
4 Toteutuksen taustaa 23
4.1 Tietokantapalvelujen valinta 23
4.2 Pilvipalveluympäristön konfiguraatiot 23
4.3 TPC-H-datajoukko 24
4.4 Tietomalli 24
4.5 Dataputken vaiheet 25
5 Toteutus 28
5.1 Resurssien luonti 28
5.2 ADA:n yhteysmäärittelyt 31
5.3 Taulujen luonti 33
5.4 Taulujen ja rakenteiden tuonti ADA:an 34

5.5 Datakulkujen luonti ja määrittelyt 35
5.6 Datan tuonti 38
5.7 Työnkulkujen luonti 40
5.8 Microsoft Fabric – dataputket ja työnkulun luonti 44
5.9 TPC-H-tietokantakyselyt 45
6 Mittaustulokset ja vertailu 47
6.1 Mittaukset 47
6.2 Mittauksien seuranta 47
6.3 Mittaustulokset 52
6.3.1 Datan vienti kohdetauluihin 52
6.3.2 Muutokset stage-latauksiin 54
6.3.3 TPC-H-kyselytestit 54
6.4 Optimoinnilla parannusta suorituskykyyn? 55
6.5 Hinnoittelu 58
6.6 Palveluiden käyttöönotto ja käyttö 59
7 Yhteenveto ja pohdinta 60
7.1 Tutkimuskysymysten vastaukset 60
7.2 Pohdinta 60
7.3 Yhteenveto ja jatkokehitysideat 61
Lähteet 62

Lyhenteet

ACID – tietokannan ominaisuuksia, jonka tarkoitus on taata muun muassa tietojen eheys.

BI – liiketoimintatiedon analysointi ja visualisointi päätöksenteon tueksi.

Big data – Havainto sekä teoria suurista datan määristä, joiden määrien, kasvuvauhdin ja vaihtelevuuden haasteisiin pyritään vastaamaan konkreettisilla työkaluilla tai ratkaisuilla.

Blob-storage – varasto objektimuotoisten tiedostojen tallentamiseen.

Clustered Columnstore Index - Sarakkeisiin perustuva indeksi tietokannoissa.

HEAP – muistialue, jota ohjelmat käyttävät dynaamisesti.

MPP – ohjelman koordinoima massiivinen rinnakkainen prosessointi.

OLAP – tekniikka, jota käytetään suurten yritystietokantojen järjestämiseen ja liiketoimintatietojen tukemiseen.

On-premises – yrityksen omissa tiloissa sijaitseva IT-infrastruktuuri.

Python – suosittu, korkean tason yleiskäyttöinen ohjelmointikieli.

SaaS, PaaS – ohjelmisto palveluna (Software as a Service), alusta palveluna (Platform as a Service).

Siiloutuminen – tiedon eristäytyminen eri osastoille tai järjestelmiin, vastakohtana organisaation integraatio.

SKU – termi Azuren sisällä resurssin, kuten tietokannan, eri versioiden välillä.

Staging-alue – väliaikainen tallennustila datalle ennen sen varsinaista prosessointia ja sijaitsee datan lähteen ja kohteen välissä.

Tsettatavu – erittäin suuri digitaalisen tiedon tallennusyksikkö. Se biljoona gigatavua eli 10 potenssiin 21 tavua.

1 Johdanto

Nykypäivänä data on jokaisen modernin liiketoiminnan kulmakivi. Jo yksistään vuonna 2021 79 tsettatavua dataa ja määrän on arvioitu kasvavan 181 tsettatavuun vuoden 2025, joka osaltaan toimii ”data edellä” ajattelua innoittavana. Ajattelussa data ja analytiikka ajavat päätöstenteon pohjana kuin intuitio tai tavanomaiset tekemisen tavat. Se keskittyy vankistamaan liiketoimintojen suorituskykyä varmistaen kuitenkin muun muassa datan saavutettavuuden, tiedonhallinnan ja tietoturvan. (Power 2023.)

Datan integriteetti, tarkkuus ja täydellisyys ovat kriittinen osa tiedostettuja päätöksiä. Tavallinen tapa vastata vaateisiin sekä toimia osana BI-järjestelmiä on tietovarastojen käyttö, jotka mahdollistavat strukturoidun tiedon varastoimisen yhteen paikkaan – välittömästi saavutettavana ja analysoitavana. Modernien tietovarastojen hyödyntäminen toimii tehokkaana alustana liiketoiminnan kyselyille ja oivalluksien löytämiseen suurten datamassojen pohjalta. Modernit, pilvipohjaisettietovarastot tuovat vaihtoehdon tavanomaisten tietovarastojen käytölle tarjoten muun muassa skaalautuvuutta, joustavuutta ja lisää tehoa. (The Overlay Team.)

Tarve jopa reaaliaikaisen datan prosessointiin, lähteiden määrän kasvu ja jatkoprosessointi ovat lisänneet tietovarastoinnin järjestelmien suorituskyky- ja tallennusvaateita, jonka yhteydessä on alettu puhua big datasta sekä big data -prosessoinnista. (Salo 2014, 30–31). Julkiset sekä hybridi-pilvipalvelut ovat tuoneet vaihtoehdon paikallisesti tarjotuille on-premise -järjestelmille ja -resursseille. Pilvipalveluiden SaaS- ja PaaS-palveluina tarjoamat tietovarastoinnin palvelut ovat voineet korvata kokonaan erillisiä tietojärjestelmiä ja ratkaista haasteita liittyen esimerkiksi skaalautuvuuteen, ylläpitoon ja määrittelyn kompleksisuuteen liittyen.

1.1 Toimeksiantajan esittely

Toimeksiantaja Epical on vuonna 2023 perustettu, Enfo Oyj:stä omaksi pohjoismaiseksi datakonsultointiyhtiöksi eriytynyt noin 400 asiantuntijan joukko. Epical näkee vastuullisen datan hyödyntämisen tehokkaana työkaluna, jolla maailmaa voi muuttaa ja on erikoistunut muun muassa dataan ja analytiikkaan, tietoturvaan, sovelluksiin ja integraatioihin. Yhdessä palveluista on käytetty Epicalin tytäryhtiön Qivadan (vuonna 2024 Epicalin hankkima) kehittämää ADA-automatisointipalvelua.

1.2 Työn tausta ja tavoitteet

Tämä työ tähtää tietovarastoinnin ja siihen liitännäisten teknologioiden teoreettisella taustoituksella antamaan kattavan kuvan varastoinnin tavoista, tekniikoista ja käyttötilanteista. Käytännön osuudessa luodaan yhtäläiset data- ja työnkulut tietokantapalveluihin, joiden pohjalta tehdään erityyppisiä suorituskykymittauksia. Mittausten kautta luodaan vertailua työhön valikoitujen palvelujen ominaisuuksiin, teknologiaan, rajoitteisiin ja mahdollisiin käyttötapauksiin. Samalla vertaillaan niitä muidenkin tekijöiden, kuten kustannusten, käytettävyyden ja jatkohyödynnettävyyden osalta.

Käytännön osuudessa palveluiksi valikoitiin useampia alan yleisesti tunnustamia ja käyttämiä julkisen pilven tietokantapalveluja sekä toimeksiantajan kehittämä ja markkinaan tarjoama ADA-automatisointipalvelu. Työn tietopohjaksi taas julkisesti saatavilla oleva, organisaation tuotantoympäristön omainen TPCH-tietojoukko, sekä tietokannan suorituskykyä ja yhtäaikaisia modifikaatioita mittaavat SQL-tietokantakyselyt. Työn suunnittelun edetessä valittiin mukaan myös Microsoft Fabric -palvelu, vaikka palvelu ei työn toteutusvaiheen hetkellä ollut vielä tuettuna ADA:ssa.

Aluksi teoriaosuudessa käydään läpi työhön olennaisesti liitännäistä tiedon käsittelyn terminologiaa: organisaation tiedon tarpeita, varastointitapoja, käytettyjä varastointiteknologioita, dimensionaalista mallia, datan mallinnusta ja jatkojalostamista BI-järjestelmien käyttöön. Tämän jälkeen taustoitetaan työssä käytettyjä datapalveluita ja näiden olennaisia toimintoja ja ominaisuuksia. Työn toteutuksen osuuksissa käydään tarkemmin eri palveluiden käyttötarkoitusta, valittua tietopohjaa sekä data- ja työnkulkujen määrittämisen vaiheita. Mittaus ja vertailuosioissa käydään läpi ja analysoidaan mittauksia, haetaan vertailupohjaa palveluiden osalta muun muassa hinnoittelussa, käytön ja määrittelyn helppoudessa.

1.3 Tutkimuskysymykset ja rajaus

Tämän opinnäytetyön tutkimuskysymykset ovat:

1. Miten modernit pilvipohjaiset tietokantapalvelut eroavat ominaisuuksien, kuten tehokkuus ja hinta, puolesta?
2. Mitkä ovat ne olennaiset tekijät, jotka vaikuttavat sopivan tietokantapalvelun valintaan?
3. Miten yleisesti saatavilla olevat, markkinoiden yleisesti tunnustamat palvelut, vastaavat tutkimuskysymyksen 2 tarpeisiin?

Työssä on käytetty julkisesti saatavaa, standardoitua tietojoukkoa ja analyyttisen tietokannalle suunniteltuja businessorientoituja tietokantakyselyjä palvelujen

suorituskykytestaukseen. Eri tietokantapalvelujen määrää on rajattu yhteensä neljään kappaleeseen. Palvelujen hinnat on haettu valmistajien sivuilta tämän opinnäytetyön kirjoittamisen ajanhetkellä. Testausta on tehty mahdollisimman yhdenmukaisesti ja yhdenmukaisessa ympäristössä, niin että pilven resurssit on valittu esimerkiksi saman alueen sisältä verkkoyhteyksien vaikutuksen minimoimiseksi. ADA valikoitui palveluksi, sillä sitä oli onnistuneesti hyödynnetty asiakasprojekteissa, mutta lisää käyttökokemusta toivottiin osan siihen linkitetyistä palveluista osalta.

1.4 Tutkimusmenetelmät

Työssä tutkimusmenetelminä käytetään vertailuanalyysia ja -arviointia sekä konstruktiivista tutkimusotetta. Vertailuanalyysi on luonteva valinta tutkimusmenetelmäksi, sillä työssä vertaillaan useita pilvipohjaisia analyyttisia tietokantapalveluita niiden suorituskyvyn ja muiden tekijöiden pohjalta. Menetelmän avulla voidaan systemaattisesti analysoida ja arvioida eri palveluiden ominaisuuksia, suorituskykyä, hinnoittelua ja muita tärkeitä vaikuttavia tekijöitä. Vertailuanalyysi taas auttaa tunnistamaan eri palveluiden väliset yhtäläisyydet ja erot sekä muun muassa niiden vahvuudet ja heikkoudet. (Myllykorpi 2024.)

Konstruktiivinen tutkimusote puolestaan valittiin, koska työssä ei keskitytty pelkästään kuvaamaan olemassa olevia palveluita, vaan myös luomaan tietoa ja lisää ymmärrystä siitä, miten organisaatiot voivat valita itselleen mahdollisimman sopivan palvelun ja teknologian. Aiheena työ pyrkii myös ratkaisemaan haasteita mitä organisaatioilla voi olla reaalimaailman datan käsittelyn ja varastoinnin haasteiden kanssa ja löytämään asiaan uusi näkökulma tai ratkaisu – eli konstruktio. (Lukka 2001.)

2 Tiedon käsittelytavat, varastointi ja mallinnus

2.1 Datan ja tiedon määrittely

Datan synonyymina arkikielessä käytetään usein sanaa tieto. Data on kuitenkin vain raaka-aine – merkkejä ja symboleja, josta voidaan louhia informaatiota, ja informaation pohjalta muodostaa tietoa. Tiedolla lisätään ymmärrystä ja kumuloituneen tiedon pohjalta muodostetaan tietämystä. (Salo 2014, 32.)

Dataa voidaan lajitella esimerkiksi jakamalla se kahteen sen eri muotoon: paikallaan pysyvään ja liikkuvaan dataan. Paikallaan pysyvä data on helpommin hallittava ja prosessoitava muoto, kuten data tietokannassa. Virtaava data on esimerkiksi lämpötilasensorien tuottamaa dataa, joka mittaa ilman lämpötilaa tietyllä ajanhetkellä. (Salo 2014, 28.)

Toinen erittäin suosittu tapa jakaa data on sen tyypin mukaan strukturoituun, puolistrukturoituun ja strukturoimattomaan strukturoidun ollessa hyvin jäsenneltyä ja selkeästi rakenteellista dataa, kun taas strukturoimaton rakenteetonta dataa. Strukturoimaton data voi olla sosiaalisen median dataa, videokuvaa ja valokuvia, joita ei ole helppo tallentaa esimerkiksi relaatiokantaan. Puolistrukturoitu taas metatiedoilla varustettua strukturoimatonta dataa, kuten videokuvaa, josta ilmenisi sen kuvaaja, esittäjät, milloin se on kuvattu ja niin edelleen. Strukturoitu data voisi olla taas asiakastieto, jossa on asiakkaan nimi, yhteystiedot ja osoite. (Salo 2014, 27.)

2.2 Dataputket ja datan orkestrointi

Dataputki on keskeinen käsite modernissa tiedonhallinnassa, prosessoinnissa ja analysoinnissa. Käsitteenä dataputki on kuitenkin varsin laaja kattaen koko tiedon elinkaaren alkuperäisestä lähteestä ja formaatista aina sen hyödyntämiseen muun muassa analyyseissa, BI-järjestelmin luoduissa visualisoinneissa tai koneoppimisen työkuormien tehtävissä. Dataputken arkkitehtuurin muodostuu kolme ydinvaiheesta: datan kerääminen datan transformaatiot ja datan tallennus. Ennen kuin tietoa varastoidaan se sisältääkin prosessointia jonkin verran, kuten datan suodattaminen, tietojen peittäminen ja yhdistämiset, joiden tehtävä on varmistaa tarkoituksenmukaisen datan integraation ja standardisoinnin. Datajoukon kohteen ollessa relaatiotietokanta kyseisen vaiheen merkitys korostuu entisestään. (Stryker 2024.)

Dataputkia on useampaa päätyyppiä, jotka sopivat eri tarkoituksiin ja alustoille:

- Eräkäsittely: lataa dataa ”erissä” tietyin aikavälein useimmiten liiketoiminnan järjestelmien ruuhkahuippujen ulkopuolella, jolloin ne eivät kuormita kyseisten järjestelmien toimintaa.
- Datan tietovirrat: toiselta nimeltään tapahtumapohjaiset arkkitehtuurit prosessoivat dataa jatkuvasti useista eri lähteistä, kuten sensorit tai järjestelmän käyttäjän aiheuttama tapahtuma. Tapahtumat prosessoidaan ja analysoidaan ja lähetetään varastoitavaksi tai lähetetään eteenpäin analysoitavaksi.
- Dataintegraatioputket: Yhdistelevät dataa useasta lähteestä yhteen näkymään. Integraatioputket sisältävät usein ETL-prosessin vaiheet ennen datan tallentamista tietovarastoon tai -altaalle, ja ovat tärkeitä erilaisten järjestelmien käsittelyssä, jotka tuottavat epäsopivia tyyppejä tai rakenteita.
- Pilvinatiivit dataputket: Hoitaa dataputken arkkitehtuurin mukaisten tehtävien lisäksi tiedon tallentamisen ja jatkokäsittelyn vähentäen muun muassa datan siiloutumista, varmistaen sen ajankohtaisuuden ja oikeellisuuden sekä mahdollistaen itsepalvelukäyttöä.

(Stryker 2024.)

ETL-prosessi muistuttaa ja jakaa osin vaiheita dataputkien kanssa. Dataputkien yhteydessä usein puhutaan ETL-dataputkista, vaikka nämä voivat erota muun muassa putken tyypin osalta. ETL-prosessin tarkoitus on muuntaa usein useasta eri lähteestä koostuva tieto työkalujen avulla raakamuotoisesta hyödylliseksi informaatioksi – usein käyttötapauskohtaisesti. ETL-prosessi muodostuu seuraavista vaiheista sen etuliitteiden mukaisessa järjestyksessä: data haetaan lähdejärjestelmästä (Extract), datan muokkaus ja puhdistus tiedon laadun ja konsistenssin säilyttämiseksi (Transform) ja data ladataan lopulliseen kohteeseen (Load). (IBM a.)

Muunnelma tyypilliseen ETL prosessiin on ELT, jolloin datan muokkaukset ja muunnokset tehdään vasta sen tuomisen jälkeen. Tällöin tiedon laatuun, lopulliseen formaattiin ja käyttötapauksiin ei vaadita niin paljon ennakkoon suunnittelua ja työstöä. ELT:ssä ei käytetä erillistä staging-aluetta transformaatioihin vaan data ladataan suoraan kohdetauluihin, jonka jälkeen sitä vasta muokataan. (IBM a.)

ETL:stä ja ELT:stä puhutaan usein integraatiometodeina, mutta olemassa on myös muita tapoja, jotka hyödyntävät dataintegraation työkulkuja, kuten esimerkiksi CDC. CDC:ssä tarkoituksena on poimia vain muuttunut data lähteestä, jolloin ETL:n tapaan ei ladata kerralla kaikkea. Sitä voidaan myös hyödyntää reaaliaikaisesti datalakeen vietyyn dataan. ETL:ää käytetään usein lähinnä pienemmissä harvemmin päivittyvissä hyötykuormissa, kun taas

ELT tai CDC soveltuvat erityisen alati muuttuvan big dataan tai reaaliaikaisen datan tietovirtoihin. (IBM a.)

Datan orkestroinnissa ETL-prosessin tapaan integroidaan dataa useasta lähteestä. Orkestroinnissa tarjoaa kuitenkin enemmän tekniikoita datan hallintaan, kartoitukseen, mallinnukseen ja organisaation tiedon keskittämisen kautta siiloutumisen vähentämiseen ja sen saavutettavuuden parantamiseen. (Chu 2024.)

Datan orkestrointi sisältää kolme vaihetta:

- Organisointi: dataputki kerää tietoa useasta paikkaa, kuten relaatiotietokannat, API-rajapinnat ja vie kerää ne tietovarastoon tai -altaaseen.
- Muuntaminen: ETL:n tapaan data yksilöidään, puhdistetaan ja korjataan sen eri formaateista. Data menee läpi manuaalisia ja automatisoituja laatutarkastuksia ennen kuin se katsotaan validoiduksi ja voidaan esittää tuotantokannassa.
- Aktivointi: Data toimitetaan BI-työkaluille tai muille analyyttisen datan prosessointityökaluille.

Datan orkestrointi pohjautuu usein aiemmin mainittuun ELT:n, jossa vasta viimeisessä vaiheessa tehdään muunnokset. Muuntelun tapahtuessa vasta tietovarastoon latauksen jälkeen tätä dataa pohjalla voidaan hyödyntää eri käyttötapauksiin. (Poppy 2024.)

2.3 Tietoallas

Big datan yleistymisen myötä tietomäärien ja -tarpeiden kasvaessa syntyi tarve tietojen tehokkaaseen keräilyyn ja varastointiin, jonka kautta tarve tietoaltaille (Data Lake) kehittyi. Tietoallas pyrkii vastaamaan tietovarastojen rajoitteisiin, kuten tiedon formaaliin ja hierarkkiseen esittämistapaan. Erityyppisen ja -muotoisen datan varastoimisen tarpeeseen tietoaltaat vastasivat tuoden kaikki tai suurin osa datasta yhteen paikkaan sellaisenaan alkuperäisessä formaatissa. (Databricks 2024c.)

Tietoallas hyödyntää ei-hierarkkista arkkitehtuuria ja objektitietovarastoa (Object Storage) tiedon varastointiin. Objektitietovarastoissa objektit varustetaan metadatatageilla ja yksilöllisillä tunnisteilla, joka helpottaa datan paikannusta ja hakutoimintoja sekä parantaa suorituskykyä. (Databricks 2024c.)

Tietoallas sopii kaikkien datastruktuurien tallennukseen strukturoimaton ja puolistrukturoitu data mukaan lukien. Tavanomainen käyttötarkoitus tietoaltaille on datan tallennus sen alkuperäisessä muodossaan ottamatta kantaa sen enempää esimerkiksi oikeellisuuteen ja käsittely edelleen esimerkiksi tietovarastoissa. Dataa voidaan kuitenkin valitusta alustasta ja formaatista riippuen tuoda tietoaltaasta analytiikan käyttöön tai koneoppimismallin pohjaksi. (Oracle 2024.)

Ajatus kaiken datan tuomisesta yhteen ja samaan paikkaan voi olla monesti houkutteleva, mutta datamäärien alati kasvaessa ja erityyppisen tiedon lisääntyessä voidaan alkaa puhua Data Swamp:stä, eli niin kutsutusta datasuosta, jossa datan hallinta, hakeminen ja mahdollisuudet jatkojalostamiseen ovat merkittävästi heikentyneet. Datasuo on usein tulos metadatan, datakatalogin, hallintamallin ja datan puhdistuksen puutteista. (Atlan 2023.)

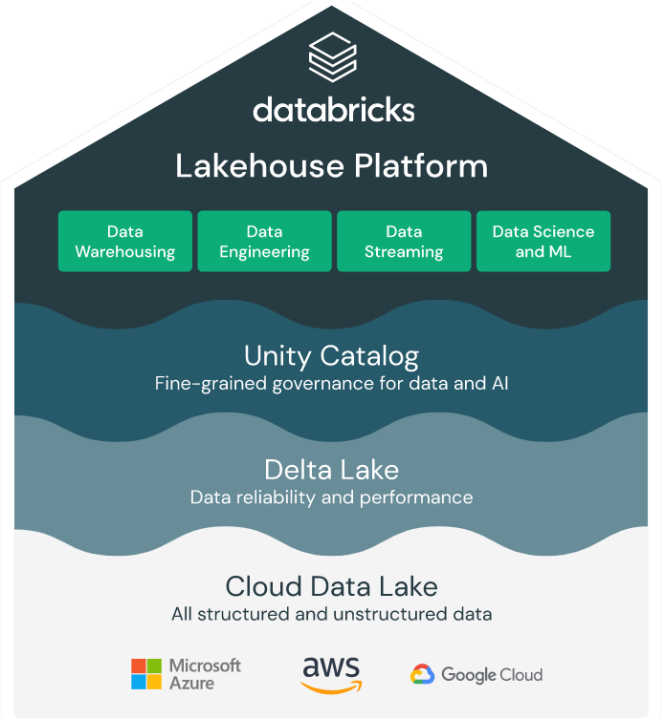
2.4 Delta Lakehouse -konsepti

Delta Lake pyrkii vastaamaan hallitsemattomien datasoiden haasteisiin tuoden datan varastointiin luotettavuutta ja rakenteita. Delta Lake on avoimen lähdekoodin projekti ja teknologia, jonka varastointikerros on suunniteltu jo olemassa olevien tietoaltaiden teknologioiden yhteyteen. Tarkoitus on säilyttää tietoaltaiden joustavuus, mutta tarjota päälle useita perinteisen tietovaraston ominaisuuksia – suorituskykyisesti ja luotettavasti. (Hewlett Packard Enterprise 2024.)

Delta Lake täydentää tietoaltaan avoimen formaatin Parquet-datatiedostoja muun muassa tiedostopohjaisella ACID-transaktiolokilla ja skaalautuvalla metadatan käsittelyllä. ACID-transaktiolokilla voidaan säilyttää datan eheys ja konsistenssi myös monimutkaisissa datan käsittelyprosesseissa. Loki pitää sisällään kaikki tauluihin tehdyt muutokset tauluun tai blob-tyyppiseen hakemistoon tehdyt muutokset. Skaalautuva metadatan hallinta taas hyödyntää Apache Spark:in hajautettua prosessointitehoa käsitelläkseen kaiken metadatan jopa petatavu-kokoluokan tauluilla miljardeilla tiedostoilla pohjalla. Delta Lake -lokien käyttö nopeuttaa kyselyjä, sillä tiedostopolut ovat tallessa ja yhtäläinen data tunnistetaan ja ohitetaan tai tiedosto luetaan vain osittain. (Databricks 2024a.)

Tuki aiemmin mainittujen ominaisuuksien lisäksi datan versioinnille sekä skeemojen käytölle varmistavat datan laadun, luotettavuuden ja helpon saavutettavuuden organisaatioiden analytiikan ja datatieteiden tehtäville. Delta Lake käyttämät tekniikat toimivat alustana Data Lakehouse -konseptille. (Qlik a.)

Data Lakehouse on kehitetty yhdistämään sekä tietovaraston että tietoaltaan lainalaisuuksia toisiinsa säilyttäen tietovaraston datarakenteet ja hallintaominaisuudet, mutta samalla tietoaltaiden joustavuuden ja tuen eri muotoisen datan käyttöön. Data Lakehouse on kuitenkin vain varastoinnin konsepti ja kokoelma eri teknologioita, kun taas aiemmin mainittu Delta Lake on varsinainen teknologia ja tiedon varastointikerros Data Lakehouse -konseptin alla (kuva 1). (Databricks 2024b.)



Kuva 1. Databricks Lakehouse alustan arkkitehtuuri (Databricks 2022)

Delta Lake optimoituna tiedon varastointikerroksena tarjoaa data-alustan, eli alimman tason, Data Lakehousen käyttöön. Delta Lake'ssä data on tallennettuna avoimen formaatin parquet-tiedostoissa, jolloin näitä voidaan käyttää laajasti eri palveluiden välillä – yhteen valmistajaan sitoutumatta. (Databricks 2024a)

Lakehouse'iin voidaan tuoda dataa strukturoidusta strukturoimattomaan, joten se toimii sekä BI-työkalujen että datatieteiden työvirtojen tarpeisiin. Lakehouset tukevat ohjelmointikielistä usein datatieteilijöiden usein hyödyntämiä Pythonia ja R:ää mutta myös suorituskykyistä SQL:ää. (IBM b.)

Lakehouse -ratkaisu muodostuu tyypillisesti viidestä vaiheesta.

- Tuontikerros: kerätään data useista lähteistä yhteen ja muunnetaan se formaattiin, joka voidaan varastoida ja analysoida lakehousessa.

- Varastointikerros: kerros vastaa strukturoidun, puolistrukturoidun tai strukturoimattoman datan tallentamisesta avoimen lähdekoodin formaatteihin, kuten Parquet tai ORC.
- Metadata-kerros: koko data lakehousen perusta. Se on yhteinen katalogi, jonka tehtävä on toimittaa metadata kaikille objekteille tietoaltaassa. Kerroksella voidaan myös hyödyntää hallinnointitoimintoja, kuten ACID-transaktiot, tiedostojen kirjoittamista välimuistiin tai indeksointia nopeammille kyselyille kuin myös tuoda ennalta määrätyt skeemoja datan hallintaa tai auditointia varten.
- API-kerros: API:t tarjoavat rajapinnan analytiikan- ja kolmannen osapuolten työkaluille kyselyiden suorittamiseksi data lakehouse arkkitehtuurin tallennettuun dataan. API:n avulla voidaan myös käyttää ja prosessoida reaaliaikaista dataa välittömästi.
- Datan käyttökerros: mahdollistaa asiakassovellusten, kuten BI-työkalut, pääsyn kaikkeen data lakeen tallennettuun dataan ja metadataan.

(IBM b.)

2.5 Medallion -arkkitehtuuri

Medallion -arkkitehtuuri on suositeltu suunnittelutapa käytettäväksi erityisesti tietoallas- ja Delta Lake -toteutuksissa, ja osana Delta Lakehousen -konseptia. Tarkoituksena on inkrementaalisesti ja progressiivisesti edetä tasolta seuraavalle parantaen datan rakennetta ja laatua sen edetessä tasolta toiselle. (Databricks 2024f.)

Arkkitehtuuri koostuu kolmesta tasosta: pronssi, hopea ja kulta. Kaikki tasoista tähtäävät tiedon laadullistamiseen, mutta käytännössä vain pronssista eteenpäin sisältävät tiedon prosessointia. Eri tasot sisältävät datalle suoritettavia operaatioita ja tehtäviä: pronssitasolla tarkoitetaan raakadatan tuomista staging-alueelle tai muulle tietoalustalle, hopea taas datan validoinnin ja putsauksen operaatioita ja kultatasolla datalle tehdään vasta muokkauksia ja muita käsittelyoperaatioita. (Databricks 2024f.)

Arkkitehtuurin mukaisesti järjestys datan käsittelyssä on edetä pronssista hopean kautta kultatasolle, mutta järjestys voi myös poiketa määrityksestä ja yhdistellä eri tasoja toisiinsa tai useamman saman tason yhdistäen, jotta päästään haluttuun käsittelyn ja yhdistelyn tasoon.

2.6 Tietovarastot

Tietovarasto (data warehouse) on digitaalinen tallennusjärjestelmä, joka yhdistää ja harmonisoi dataa eri lähteistä. Tietovaraston tavoitteena on parantaa yritysten liiketoimintaa tarjoamalla mallinnettuja näkymiä BI-työkaluihin raportoitavaksi ja analysoitavaksi – lopulta

käytettäväksi päätöksenteon tueksi. Kimball & Ross (2013, 8) määrittelevät, että “Tietovarasto on kopio transaktiodatasta, joka on erityisesti jäsennelty kyselyitä ja analyysiä varten.”

Hyvin suunniteltu tietovarasto on useasti onnistuneen BI-ohjelman tai analytiikkaohjelman perusta. Prosessin tehostamiseksi datan tuontivaiheessa se on jo puhdistettu, tutkittu poikkeamien osalta ja muutoin käsitelty esimerkiksi ETL-prosessin kautta. Nämä tietovaraston tehokkaan ennakkoon suunnittelun sekä mallinnuksen ohella toimivat katalysaattorina tiedon tehokkaalle jatkohyödyntämiselle. (Kimball & Ross 2013, 7.)

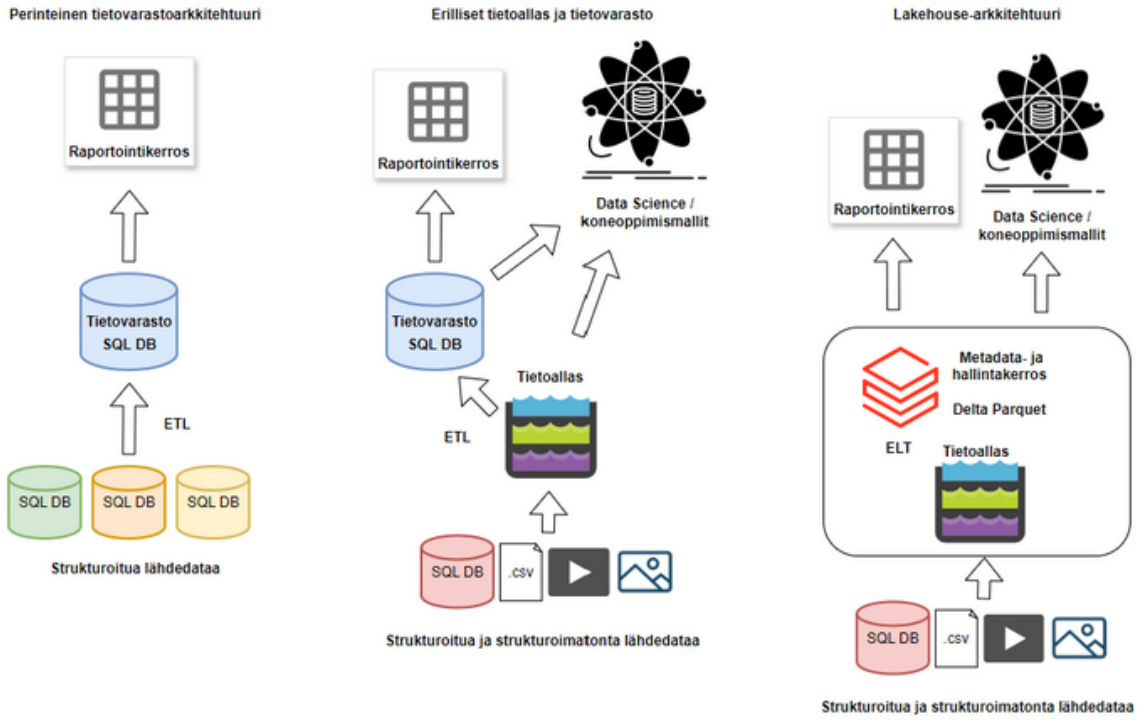
Tietovarastot ovat tiedon varastointi- ja hakujärjestelmiä (esimerkiksi tietokannoista) ja ne on erityisesti suunniteltu BI- ja OLAP-toiminnoille. Tiedot tallennetaan nykyhetkestä, mutta myös päätetyn käsittelytavan mukaan historia voidaan tallentaa faktojen lisäksi myös dimensioihin, johon viittaukset ja lähtökyselyt voivat olla useasta eri lähteestä koostettu. Teknisestä näkökulmasta tietovarasto on tietokanta, joten se koostuu tauluista, kentistä, relaatioista ja avaimista. Suurimmat hyödyt tietovaraston käytössä syntyvät kertyneen ajan suhteen. Samalla pitää kuitenkin muistaa, että sen jälkeen, kun tietovarasto on alkanut keräämään dataa, muutokset jälkikäteen sen rakenteisiin ja dataa on mahdollisesti riskaabelia ja voi aiheuttaa isoja kustannuksia. (du Mortier 2021.)

Nykyaikainen tietovarasto on suunniteltu käsittelemään strukturoitua dataa, mutta myös strukturoimatonta, kuten videotiedostoja, kuvatiedostoja tai sensoridataa. Tietovaraston luonteeseen kuuluu, että sinne ladataan dataa usein säännöllisin väliajoin eräajoissa. Pilviympäristöjen ja -palvelujen mukaantulo on kuitenkin muuttanut asetelmaa tietovarastojen tavanomaiseen käyttöön ja tuonut mukaan uusia työkaluja esimerkiksi lähes reaaliaikaiseen datan käsittelyyn (SAP). Ne ovat tuoneet vaihtoehdon tavanomaisille tietovarastoille muun muassa käytettävien resurssien skaalautuessa, jolloin esimerkiksi klusteriresurssi voi automaattisesti lisätä toisen klusterin käyttöönsä kuormituksen kasvaessa.

2.7 Tietovarasto vai Data Lakehouse – vai jotain siltä väliltä?

Perinteinen tietovarasto on vuosikymmeniä käytössä ollut ratkaisu strukturoidun datan tallentamiseen ja optimoitu erityisesti tietojen analysointia ja kyselyjä varten. Tietoallas on taas joustava ratkaisu erityisesti strukturoimattoman ja puolistrukturoidun datan tapauksissa, jossa dataa halutaan tallentaa suuria määriä alkuperäisessä muodossaan ottamatta enempää kantaa datan formaattiin, struktuuriin tai tietotyyppeihin. Yhdistelmä tietovarastosta ja tietoaltaasta mahdollistaa molempien järjestelmien ominaisuuksia, jolloin tietoaltaalle voidaan tuoda strukturoimatonta dataa ja puolistrukturoitua dataa ja viedä eteenpäin prosessin läpi tietovarastoon. Tällöin tietoa voidaan hakea ja analysoida suoraan tietovarastosta tai tietoaltaasta. Data Lakehouse on hybridiarkkitehtuuri, joka tuo yhden kokonaisuuden alle

tietoaltaan ja tietovaraston. Lakehouse toimii metadata-kerroksena tietoaltaan päällä optimoiden sen käyttöä etenkin kyselyille ja latauksille, ja mahdollistaa analytiikan ja koneoppimisen työkuormien luonnin suoraan mallin päälle. Kuvassa 2 esitelty eri tiedon varastoinnin ratkaisujen eroja. (Gates 2023.)



Kuva 2. Tietovarasto- ja lakehouse-arkkitehtuurien vertailu (Tapionsalo 2024)

Valinta sopivan tiedon varastoinnin ratkaisusta voi usein liittyä organisaation jo käytössä oleviin dataratkaisuihin tai järjestelmien valmistajiin. Nykyajan pilviratkaisut ovat kuitenkin lisänneet kenttään uusia varastoinnin ratkaisuja – usein valmistajariippumattomasti mahdollistaen laajasti eri järjestelmien käytön. Varastoinnin tavan lisäksi itse datan käyttötarve ja -tapaukset voivat määrittää ratkaisun valintaa. Esimerkiksi organisaation business-analyytikot voivat hyödyntää hyvin strukturoitua dimensionaalista tietovarastokantaa helposti kyselyjen tai BI-raporttien lähteenä. Vastakohtana tälle tietoaltaan raa'an ja suodattamattoman datan käsittelyominaisuudet voivat olla mitä datatieteilijät käyttävät koneoppimismallien pohjana. Data Lakehouse voi taas tarjota hyviä puolia molemmista tekniikoista mahdollistaen useiden eri roolien monipuolisen työskentelyn datan parissa – jopa samanaikaisesti. (Gates 2023.)

Lopuksi optimaalisen ratkaisun valinta voi koostua useasta eri tekijästä tai niiden summasta: ympäristön vaateet, tiedon hallintamallit, ajantasaisuus, historiatiedot, datan volyymi, luonne, tietoturvallisuus, liiketoiminnan vaateet, loppukäyttäjän tarpeet ja niin edelleen. Teknisten tekijöiden ohella organisaation puntarissa päätöstä tehdessä voi olla tekijät,

kuten hankkeeseen varattu budjetti ja aikataulu, käytössä oleva osaaminen sekä tietoturvavaateet. Kun hankintaan vaikuttavat tekijät on kartoitettu, voi organisaatio hakea valintaan apua esimerkiksi painotetusta päätöksentekomatriisista, jolla asettaa painotukset tekijöille niiden tärkeyden pohjalta. (Morales 2024.)

Valittu ratkaisu käyttötapauksen mukaan voi mahdollistaa koko tiedon elinkaaren lähteestä kohdejärjestelmään ja lopulta osaksi hyödyntävää järjestelmää. Tiedon käsittely tehokkaasti ja yhteensopivasti ratkaisun sisällä vaatii suunnittelua ja tietämystä sen tarjoamista ominaisuuksista ja mahdollisuuksista, mutta myös rajoitteista. (Kutay.)

2.8 Tietovarastointi ja liiketoimintatiedon järjestelmät

Tietovarastoinnin ja liiketoimintatiedon (Business Intelligence) järjestelmissä, kuten erityyppisissä dataprojekteissa, lähdetään lähes aina liikkeelle liiketoimintojen tarpeiden kautta. Liiketoimintojen tarpeet huomioiden tietovarastot suunnitellaan ja esitetään yksinkertaisesti dimensionaalisena mallina. Organisaatioissa kerätty tieto onkin usein liiketoiminnan tärkein vara. (Kimball & Ross 2013, 1–2.)

Tieto palvelee lähes aina kahta tarkoitusta organisaatioissa: operaationaalinen data tapahtumien säilytykseen ja analyyttinen data päätöksenteon tueksi. Yksinkertaistaen operationaalisiin järjestelmiin tuodaan dataa sisään ja tietovarastointi ja BI-järjestelmistä saadaan dataa ulos. Ero voidaan mitata myös käsittelyajoissa sekä tietomäärissä: transaktiojärjestelmissä puhutaan ja tarkastellaan lähes reaaliajassa tapahtuvia yksittäisistä transaktioita, kun taas tietovarasto / BI-järjestelmissä ei lähes koskaan tarkastella yksittäisiä transaktioita, data ei useinkaan ole reaaliaikaista ja ne ovat optimoitu laajoja kyselyjä varten suorituskykyiseksi ja usein sisältää tapahtumia historiassa, jotta tietoa voidaan tarkistella myös ajan funktiossa. (Kimball & Ross 2013, 2.)

Tietovarastoinnissa sekä BI-järjestelmien tavoitteena on muokata tietoa sopivaan formaattiin ja mahdollistaa sen edelleen hyödynnettävyys. Tarpeet voidaan organisaatioissa nähdä sekä järjestelmän kuin sen hyödyntäjän puolelta lähes yhtäläisenä, eli DW / BI järjestelmän tuoda tieto helposti tavoitettavaksi, esittää data yhdenmukaisesti lähteestä riippumatta, tuoda aikadimensio tapahtumille ja mahdollistaa pohjaa muutoksille. Onnistuneen järjestelmän tulisi yhdistää tietotekniikan tarpeet liiketoiminnan tarpeisiin ja olla unohtamatta kokonaisratkaisun tietoturvaa, mutta samalla vastata liiketoiminnan esittämiin kysymyksiin. (Kimball & Ross 2013, 3.)

2.9 Dimensionaalinen mallinnus

Dimensionaalinen mallinnus on yleisesti suositeltu ja hyväksytty tapa toteuttaa tietovarastointia, sillä se täyttää tarpeen toimittaa data ymmärrettävässä muodossa liiketoimintojen käyttäjille sekä tarpeen kyselyiden ripeästä prosessoinnista. Dimensionaalisessa mallinnuksessa etuina on tehdä tietovarastoista mahdollisimman yksinkertaisia käyttää. Tapa esittää sekä dataa lukijalle ymmärrettävässä muodossa että tarjota suorituskykyä kyselyihin. Helppo luettavuus on ennen kaikkea datan hyödyntäjän etu, mutta tarjoaa myös suorituskykyä tietokantaa hyödyntäville sovelluksille. (Kimball & Ross 2013, 7.)

Tietokannan normalisointi on vaiheittainen malli, jolla saadaan tietokannan rakenne tukemaan tiedon ehjää tallennusta ja tiedon tehokasta saatavuutta. Vaiheiden avulla samalla vähennetään tiedon toisteisuutta sekä parannetaan tiedon eheyden säilyttämistä. Vaiheita on useita, mutta yleisesti käytännön toteutuksissa tietokantaa pidetään normalisoituna, jos se täyttää ehdot neljänteen (4NF) vaiheeseen asti. (Wikipedia 2022.)

Vaikka dimensionaaliset mallit usein ilmennetään tietokannoissa, eroavat ne kolmannen normaalimuodon malleista (3NF) silti huomattavasti, joiden tehtävä on poistaa tiedon redundanttisuutta. Kolmas normaalimuoto jakaa tiedon erillisiin entiteetteihin, josta jokaisesta muodostuu relaatiomallin taulu. Alan termistössä usein yhdistetään 3NF -mallit entiteetti-suhde-tyyppisiin malleihin (ER). ER-diagrammeissa (ERD) esitetään entiteettien suhteita toisiinsa. Sekä 3NF että dimensionaaliset mallit voidaan esittää ER-diagrammeissa, sillä molemmat koostuvat yhdistetyistä relaatiotauluista, mutta erot mallien välillä syntyvät normalisoinnin tasossa. Dimensionaalinen malli sisältää kaiken tiedon sisällään, mitä 3NF -mallikin, mutta pakkaa tiedon formaattiin, jossa sen luettavuus säilyy, suorituskyky pysyy korkeana kyselyille sekä data on muunneltavissa. (Kimball & Ross 2013, 8.)

Erityyppisistä relaatiokannoista puuttuu puhdas erottelu loogisen rakenteen ja tiedon fyysisen tallennuksen välillä, jolloin kyselyt normalisoituun tietokantaan voivat olla hitaita. Tiedon de-normalisointi on strategia, jota käytetään jo normalisoituun tietoon erityisesti lukuoperaatioiden suorituskyvyn parantamiseksi. Kirjoitusoperaatiot puolestaan voivat hidastua, sillä tieto voi olla redundanttista ja / tai ryhmiteltyä. De-normalisoitu tieto ei ole kuitenkaan sama asia kuin tieto, jota ei ole normalisoitu, vaan sen täytyy sisältää tietty määrä normalisointia ennen sen soveltamista malliin. (Wikipedia 2024.)

Dimensionaalisessa mallissa taulujen on pääpiirteittäin kahta tyyppiä: dimensio- ja faktatauluja. Taulut sisältävät relaatiomallin pää- ja vierasavaimiin – vierasavaimet faktatauluissa ja pääavaimet faktatauluissa. Faktataulu sisältää aggregoituja tietoja, kuten myynnin avainlukuja myyntitauluissa, joten kentät ovat usein numeraalisia. Dimensiotaulussa kuvataan

taas faktataulun rivien ominaisuuksia, tiedot ovat usein tekstimuotoisia, joten sarakkeita on runsaasti. Tietokantajärjestelmässä faktataulu sisältää rivejä verrattain runsaasti sarakkeisiin nähden eli taulu on luonteeltaan ”pitkä”, kun taas dimensiotaulussa kuvailevia kenttiä voi olla useita, joten taulu on luonteeltaan ”leveä”. (IBM 2021.)

2.10 Tähti- ja lumihiutalemalli

Tähtimalli tai -skeema on yksinkertaisin esitysmuoto dimensionaalisesta mallista, jossa faktataulu on suoraan yhteydessä yhteen tai useaan dimensiotauluun tai käyttämällä vierasavainta. Tähtimallissa dimensiotaulut ovat de-normalisoituja tai ensimmäisessä vaiheessa normalisointia, joten ne sisältävät kaiken tarvittavan tiedot sisällään, eivätkä näin ollen toimi välidimensioina muille dimensioille. Tähtimalli on yksinkertaisin tapa järjestää data tietovarastossa, ja se sopiikin toteutuksiin, jossa halutaan säilyttää data erityisen luettavana sekä lukuoperaatiot tehokkaina kirjoitusoperaatioiden sekä dimensioiden koon kustannuksella ja nostaa riskiä epäkonsistentin datan muodostumiseen. (Patairya 2024.)

Yleistäen lumihiutalemalli tai -skeema on johdettu malli tähtimallista, mutta dimensiot voivat olla yhteydessä toisiinsa pää- ja vierasavain relaation kautta, jolloin ne eivät ole enää suoraan yhteyksissä faktatauluun. Dimensiotaulut ovat lumihiutalemallissa usean vaiheen mukaisesti normalisoituja, jolla on vähennetty muun muassa tiedon redundanttisuutta. Faktataulujen- ja dimensioiden mallissa on myös välidimensioita sekä mahdollisia siltatauluja. (Patairya 2024.)

Tähti- ja lumihiutalemalli voivat olla malissa yksistään tai koostua useammasta faktasta, jotka jakavat esimerkiksi samoja dimensioita, mutta toimivat kuitenkin eri analyyttisten ratkaisujen pohjana. Tähtimalli voi sopia esimerkiksi pienen organisaation myynnin tietovarastoon, jossa myynti on faktataulu ja siihen yhteydessä ovat päivämäärä, tuote ja työntekijä toimivat suorina dimensioina. Lumihiutale mallina on taas joustavampi, sillä sitä ole rajattu tiukasti tiettyyn yhteen käyttötapaukseen. Mallien väliseen vertailussa ja valinnassa huomioitavia tekijöitä on muun muassa datan luonne, tallennustilan rajat, tietokantakyselyjen tarve ja suorituskyky. Tietovarastoratkaisu sekä myös BI- tai analytiikkatyökalun valinta voi vaikuttaa kuinka erilaisia malleja voidaan hyödyntää mahdollisimman tehokkaasti suorituskykyrajoitteet huomioiden. (Patairya 2024.)

3 Palvelut

3.1 Pilvipohjaiset tietovarastopalvelut

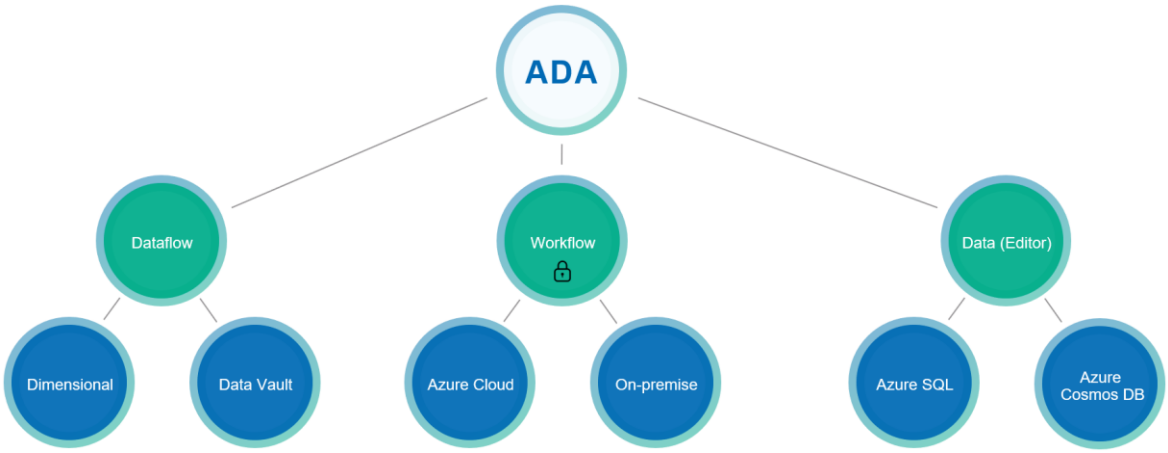
Nykyaikaiset tietovarastot ovat usein pilvipohjaisia tarjoten joustavuutta ja skaalautuvuutta tuoden vaihtoehdon paikallisesti toteutetuille resursseille tai datakeskuksille. Pilvipohjaisina SaaS- ja PaaS-palveluina ne tarjoavat palvelun päivitykset ja infrastruktuurin vähentäen näin muun muassa ylläpidon ja laiteinvestointien tarvetta. Tietovarastoresurssien skaalaus on usein hyvin nopeaa, mikäli lisäresursseja tarvitaan. Resurssien tarpeen vähentyessä voidaan resurssit skaalata alaspäin. (Qlik b.)

Työhön valitut tietovarasto- tai tietovaraston omaiset palvelut edustavat tunnettujen valmistajien ja markkinan tunnustamia analyyttisen datan tietovaraston omaisia palveluja. Työkalut tarjoavat myös eri työkaluja datan hakuun, käsittelyyn ja analysointiin. ADA:n avulla palvelussa voidaan automatisoidut työnkulut datan integraatioille vähentäen määrittelyn tarvetta. Työnkulkuihin voidaan myös käynnistää, skaalata ja sammuttaa resursseja tarpeen mukaan ja aiheuttaa selväkielisiä hälytyksiä, mikäli virhetilanteita kulkujen vaiheissa etenee.

3.1 Analytics Development Accelerator (ADA)

ADA on SaaS-pohjainen palvelu, joka on suunniteltu kiihdyttämään tietovarasto- ja data-alusta-pohjaisia projekteja tarjoamalla työkalut modernien automatisoitujen ratkaisujen implementointiin, operointiin sekä ylläpitoon. SaaS-tyyppisenä palveluna ADA ei vaadi asennuksia, mutta alle vaaditaan kuitenkin vähintäänkin oma Azure-tilaus, johon palvelu voidaan liittää. ADA toimii selaimilla web-pohjaisen käyttöliittymän kautta ja tukee monivaiheista tunnistautumista Microsoft Entraan.

ADA:n tärkeimmät ominaisuudet ovat metadatapohjainen kehitys, versionhallinta, integroitu deploy-toiminto datakuluille, tuki ulkoisille työkuormille, työkulkujen orkestrointi ja data editor analyyttiseen masterdatan hallintaan (Qivada a). ADA:n rakennetta on kuvattu kuvassa 3.



Kuva 3. ADA-toiminnot (Qivada a)

Dataflow-osio hoitaa muun muassa datakulkujen luonnin Microsoft Synapseen, Microsoft Fabriciin tai Databricsiin, automatisoidusta työnkulkujen luonnista ja sekä datakulkujen lukitsemisesta. Workflow-osiossa voidaan suorittaa tyypillisiä analyyttisen tietokannan operaatioita, kuten dataputket, pysäytys, palvelujen skaalaus, prosessointi sekä monitoroida ja palauttaa työnkulut. Data (Editor) -osiossa tuodaan tiedon metadatarakenteet palveluun, post-execute SQL:llä viedään business-säännöt kantaan. Data-editorilla ja voidaan livenä muokata ladattuja tauluja ja suorittaa analyyttisten masterdata-tietojen tuonti ja määrittely. (Qivada a.)

ADA valikoitui työhön, sillä sitä oli jo onnistuneesti hyödynnetty useissa Enfon ja Epicalin asiakashankkeissa. Kuitenkin esimerkiksi Snowflake-yhteystyyppi oli tuotu palveluun ajankohdallisesti ennen suunnittelua, joten käytännön kokemusta käytöstä haluttiin. Myös Fabric-yhteys oli kehityksen alla ja alustasta haluttiin samalla käyttökokemusta liittymän kehityksen aikana. (Qivada b.)

Työssä ADA:n ominaisuuksista hyödynnetään dataflow-osiosta dimensionaalisen mallien käyttöä ja luodaan ympäristökohtaiset datatyönkulut. Data-puolelta käytetään Azure SQL:ää, Databricks Unity Catalog -katalogin tauluja sekä Snowflakeen sen DW-puolta, jotta saadaan datan metatietorakenteet palveluun näiden hyödyntämiseksi kenttien kartoituksissa. Workflow-ominaisuutta taas hyödynnetään Synapsen dataputkien orkestrointiin.

ADA on hankittavissa Azure Marketplacen kautta. Hinnoittelussa se tukee pay-as-you-go tyyppistä eri resurssien käyttöön pohjautuvaa maksua. Nämä eri resurssit jakautuvat eri osioiden mukaan: dataflow:t, workflow:t ja data editor. Tämän lisäksi peritään tenantista kiinteä maksu per kuukausi, mutta sitä ei peritä, mikäli käyttö ylittää tuon kiinteän maksun määrän tai tenant on pysäytettynä koko kuukauden ajan. (Qivada c.)

3.2 Snowflake

Snowflake on SaaS-tyyppinen pilvipohjainen tietokanta- ja datajärjestelmä, joka on suunniteltu käsittelemään suuria määriä tietoja skaalautuen kapasiteetin ja laskentaresurssin osalta. Snowflake on suunniteltu niin, ettei se ole yhteen pilvipalveluun sidonnainen, vaan alusta voidaan ottaa käyttöön kirjoitushetkellä kolmessa suuressa: Microsoft Azure:ssa, Amazon AWS:ssä ja Google Cloud:ssa. Käyttöönotettaessa Snowflaken arkkitehtuurin kolme osaa varastointi, laskenta sekä pilvipalvelut viedään ja hallitaan kokonaan valitussa pilvipalvelussa. (Snowflake 2024a.)

Kun dataa tuodaan Snowflake:en järjestelmä organisoi sen sisäisesti optimoituun ja kompressoituun kolumnimaiseen formaattiin ja tämä optimoitu data varastoidaan sen pilvivarastoon. Näihin dataobjekteihin ei käyttäjällä ole Snowflake:ssä suoraa pääsyä, mutta niiden sisältämä data on kyseltävissä SQL-kielellä. Snowflake tukee datan tuomista laajasti muiden palveluiden tietoaltaiden tiedostoista sisäiseen Snowflaken stageen palveluun sen eri muodoissaan strukturoimattomasta, puolistrukturoituun ja strukturoituun tietoon. (Snowflake 2024b.)

Tavanomaisesta tietovarastosta poiketen Snowflake erottaa fyysisesti tallennuksen, laskennan ja palvelut (kuten metatiedon ja käyttäjähallinnan), mutta yhdistää ne loogisesti toisiinsa. Näin ollen esimerkiksi laajat tietokantakyselyt eivät rampauta varastokapasiteetin toimintaan laskevasti tai toisinpäin. Tietovarastoissa Snowflake tukee relaatiomuotoista SQL-tietokantaa, ja tietokantakyselyt tukevat standardin mukaista SQL:ää sisältäen merkittäviä ANSI SQL:n osia. Snowflake ajaa kyselyt prosessointikerroksessa hyödyntäen MPP-laskentaklustereja jakaen prosessointitehoa virtuaalisten tietovarastojen välillä. (Snowflake 2024c.)

Snowflake tukee myös Data Lakehouse -arkkitehtuuria hyödyntäen avointa Apache Polaris -katalogia. Katalogilla on tuki esimerkiksi avointen taulujen formaatille, kuten Apace Iceberg sekä tuki API-rajapintojen kautta myös laajasti muille palveluille. (Snowflake 2024d.)

Snowflake kutsuu graafista käyttöliittymäänsä nimellä SnowSight. Käyttöliittymän avulla hoituvat suurimmat osat toiminnoista, kuten tietokantakyselyt, ja se tarjoaa myös suoria työkaluja datan visualisoimiseksi tulosten pohjalta. Käyttöliittymän kautta voidaan myös monitoroida aiempien kyselyjen suoriutumista. Graafisen käyttöliittymän lisäksi Snowflake tukee yhdistämistä toimintoihinsa muun muassa komentorivipohjaisesti SnowSQL-asiakassovelluksella. (Snowflake 2024e.)

Snowflake:n hinnoittelu jakautuu kolmeen osaan:

- Laskenta: resurssit kuten virtuaaliset tietovarastot, Snowpipe, automaattinen klusterointi ja tehtävät.
- Varastointi: data, joka on tallennettuna Snowflake:en sekä datasiirrot järjestelmän sisään ja ulos.
- Pilvipalvelut: tehtävät kuten käyttäjien autentikointi ja roolien hallinta.

Hinnoittelumalli pohjautuu krediittipohjaiseen hinnoitteluun. Krediittejä voi joko hankkia ennakkoon, jolloin jokainen palveluista vähentää sitä kerrallaan tai sitten tehdä sopimuksen käytöstä, jolloin sopimuksen pohjalta saa tyypillisesti tietyn määrän kreditejä vuoden ajaksi. Käyttö palveluiden välillä voi vaihdella, jolloin kokonaishinnan arvioiminen voi olla hankalaa laskentakapasiteetin ollessa usein kallein elementti. Valitun tilin taso myös vaikuttaa osaltaan klusterien hinnoitteluun. (Lingappa 2024.)

3.3 Databricks

Databricks on PaaS-tyyppinen pilvipohjainen hajautetun laskennan ja avoimen lähdekoodin data-, analytiikka- ja tekoälyalusta. Databricks esitteli alkujaan ”lakehouse” -konseptin yhdistäen tietoaltaan ja tietovaraston ominaisuuksia yhden tuotteen alle saumattomaksi kokonaisuudeksi, jota nykyään kutsutaan Databricks’ssä Data Lakehouse -arkkitehtuuriksi. Palvelu on tämän opinnäytetyön kirjoitushetkellä tuotavissa suurimpien pilvipalvelujen yhteyteen. Databricksin Data Intelligence -alusta integroituu suoraan pilven varastoon ja käyttäjän tilin tietoturvaan ja se hallinnoi ja ottaa käyttöön valitun pilviarkkitehtuurin automaattisesti. (Microsoft 2024g.)

Azuressa Databricks hyödyntää generatiivista tekoälyä data lakehousen kanssa ymmärtääkseen datan uniikkia semantiikkaa, jonka jälkeen se optimoi suorituskykyä ja hallitsee rakennettua infrastruktuuria käyttäjän business-tarpeisiin soveltuvaksi. Luonnollisen kielen prosessointi oppii käyttäjän kysymysten pohjalta business-termistöä. Luonnollisen kielen tulkki auttaa myös kirjoittamaan koodia notebookien sisällä, ongelmien ratkaisuun ja vastaamaan dokumentaation kysymyksiin. (Microsoft 2024g.)

Databricks pohjautuu Apache Spark-projektiin luoden tehokkaan abstraktiotason Spark-moottorin päälle. Toiminnoista se käyttää muun muassa Sparkin hajautettua laskentaa sekä RAM-muistissa tapahtuvaa prosessointia. Yksi Databricksin erottavista ominaisuuksista on, että optimoi työkuormien suorituskykyä säätämällä älykkäiden algoritmien avulla dynaamisesti Sparkin konfiguraatiota, jolla mahdollistetaan ripeämmät suoritusajat ja tätä kautta suoraan rahallisia säästöjä resurssien kustannuksiin. (Escórcio 2024.)

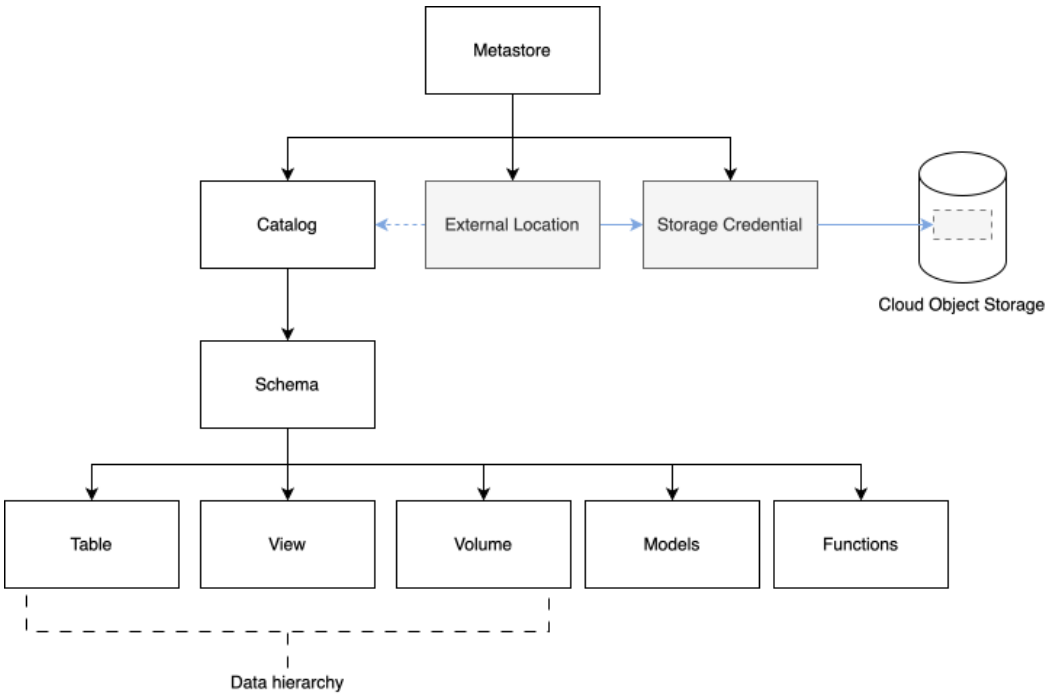
Tietoa prosessoitaessa prosessoitava yksikkö on tyypillisesti Spark Dataframe, joka on datarakenne 2-ulotteinen sarakemuotoinen taulu, muistuttaen hyvin paljon laskentataulukkoa.

Jokainen Dataframe pitää sisällään skeeman, joka määrittää jokaisen sarakkeen nimen ja datatyypin. Puuttuvat tai virheelliset arvot tallentuvat tyhjänä. (Databricks 2024e.)

Dataframen prosessointi tapahtuu laskentaklusterin muistissa ja sitä voidaan muokata ohjelmointikielten, kuten Pythonin, R:n tai SQL:n avulla datan prosessointiputkissa. Databricks kehityksen perusta on notebookeissa, jotka pitävät sisällään koodia komentosoluissa, ja toimivat pohjana prosessointiputkille. Yhteen komentosoluun kirjoitetaan esimerkiksi SQL-koodi, jonka avulla Spark Dataframe muodostuu. Yksi notebook voi koostua useammasta komentosolusta sisältäen useampaakin kieltä ja yksi notebook voidaan ketjuttaa useampaan luoden yhden dataputken. Tällöin notebook hoitaa esimerkiksi yhtä tiedon käsittelyn vaihetta Medallion Lakehouse -arkkitehtuurissa siirtyen eteenpäin putkessa. Prosessointiputken lopputulemana tallennetaan usein pilveen Delta Lake -formaattiin, josta se on käytettävissä lakehouse-ratkaisun pohjana. (Anttila 2023.)

Unity Catalog lisää datan näkyvyyttä ja hallittavuutta tuoden muun muassa datan hallintamallin ja käyttäjien hallinnan yhteen keskitettyyn paikkaan. Unity Catalog toimii myös metavarastona. Metavarasto toimii keskitettynä hakemistona koordinoiden hakuja metadatan pohjalta ja säätämällä käyttöoikeuksia ilman itse datan tallennusta. (Gomezm 2024.)

Poiketen tavallisista metavarastoista Unity Catalog lisää yhden tason nimiavaruuteen – katalogin. Lisäämällä useamman katalogin esimerkiksi kehitysympäristön mukaan, jotka jakavat saman skeeman ja taulut, voidaan ottaa käyttöönottoputki käyttöön ilman koodimuutoksia (kuva 4). (Gomezm 2024.)



Kuva 4. Metastore (Gomezm 2024)

Databricks tukee myös SQL-tietovarastoja erillisen SQL warehouse -laskentaresurssin avulla. Resurssi on skaalattavissa yhtä lailla muiden laskentaresurssien tapaan ja on optimoitu erityisesti SQL kyselyt ja analyyttiset työkuormia varten. SQL-tietovarastot integroituvat saumattomasti Unity Catalog:n kanssa mahdollistaen taulujen ja näkymien luomisen skeemaan. SQL-tietovarastot ovat myös esimerkiksi BI-työkalujen suoraan luettavissa. (Microsoft 2024a.)

Databricks:n hinnoittelumalli on pay-as-you-go mallia, jossa laskutetaan käytön perusteella. Laskennan yksikkö on Databricks Unit (DBU), joka kuvastaa laskennallisia resursseja. DBU käyttö pohjautuu tekijöihin, kuten klusterin koko, ajoaika ja käyttöönotetut ominaisuudet. Muita hinnoitteluun vaikuttavia tekijöitä ovat muun muassa Databricks-tilin taso, pohjalle valittu pilvipalvelu sekä tiedon varastoon käytetty resurssi. (Marattha 2024.)

3.4 Azure Synapse Analytics

Azure Synapse Analytics on Microsoftin kehittämä PaaS-tyyppinen pilvipohjainen analytiikka-alusta, joka yhdistää tietovarastoinnin, tietoaltaat, big data -analytiikan ja datan integroinnin saumattomasti yhden hallitun palvelun alle. Se tarjoaa sekä palvelimettomia että dedikoituja resursseja tietovarastoinnin tarpeisiin. Alusta kattaa koko putken tiedon tuonnista, valmistelusta, hallinnasta ja datan jakamisesta edistyneen analytiikan käyttöön. (Microsoft 2023.)

Azure Synapsen pohjalla on pilvinatiivi, hajautettu SQL-prosessointimoottori, joka on rakennettu SQL Server -pohjalta etenkin vaativat tietovarastoinnin hyötykuormat huomioiden. Muiden MPP-laskennan tyyppisten resurssien tapaan se erittelee datan ja laskennan toisistaan, joista laskutetaan kustakin erikseen. Tietovarastoresurssi perustuu DWU-pohjaiseen laskentatehoon ja resurssia voi skaalata halutessaan ylös ja alaspäin sekä pysäyttää kokonaan. (Qlick b.)

Synapse Studio tarjoaa graafisen työtilan datan valmisteluun, datan hallintaan, tutkintaan ja tietovarastokäyttöön. Muiden toiminnallisuuden ohella käyttöliittymän alta voidaan myös linkittää muita palveluja Synapse:en. Integration Runtime (IR) on laskentainfrastruktuuri, Azure Synapse pipelines -käyttöä varten mahdollistaen tuetut toiminnallisuudet liitetyissä palveluissa. (Microsoft 2024b.)

Synapsen hinnoittelu pohjautuu myös pay-as-you-go tyyppiseen resurssien utilisointiin ja käyttöön. Dedikoituja resursseja voi myös varata ennakkoon tietyksi ajaksi käyttöön, jolloin resurssin pysäytys ei pysäytä käyttöä. Datan varastoinnista ja varmuuskopion omaisten snapshotien ottamisesta sekä myös esimerkiksi integroinnin alaisten dataputkiaktiviteettien käytöstä velotetaan erikseen. (Microsoft 2024f.)

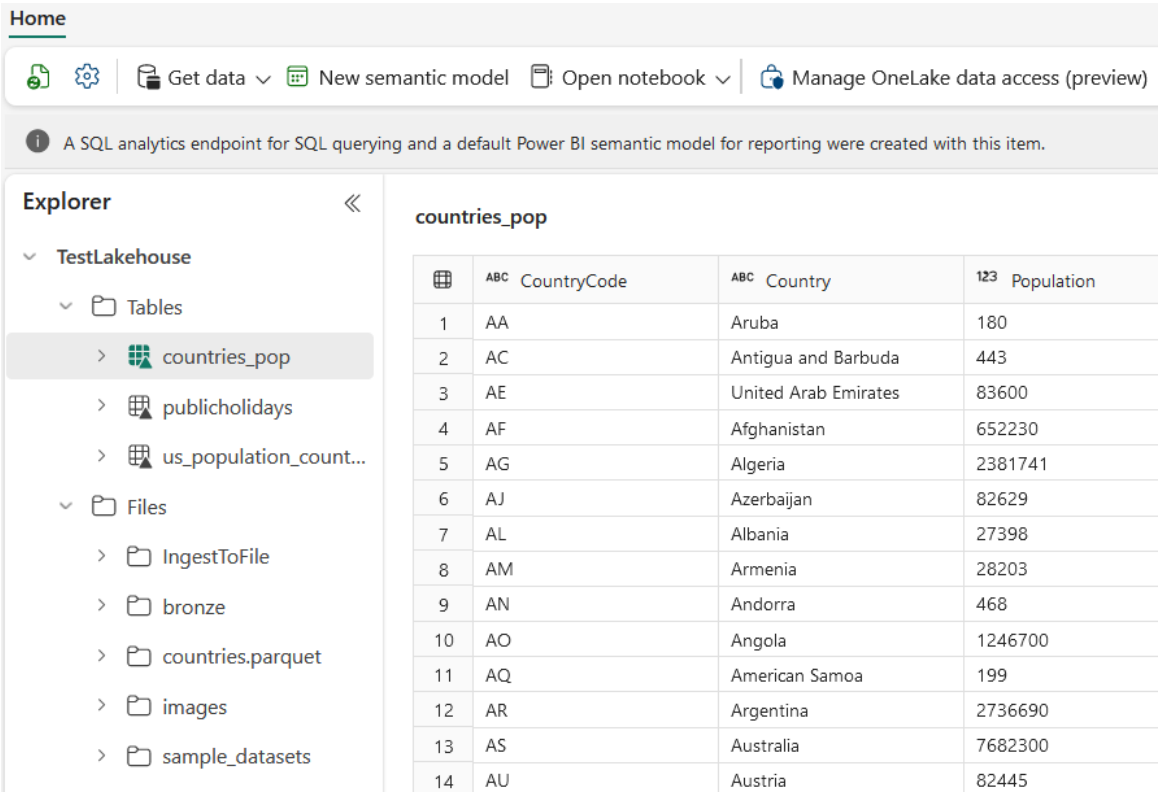
3.5 Microsoft Fabric

Microsoft Fabric on SaaS-tyyppinen pilvipohjainen analytiikan kaikki yhdessä palvelu, joka vastaa koko dataputkesta alkulähteestä aina BI-raportille saakka. Tarkoitus on, että kaikki tarvittavat palvelut löytyvät tai ovat mahdollisia integroida tuotteen alle. Se yhdistää toisistaan erilliset tuotteet, kuten Power BI, Azure Synapse Analytics ja Data Factory saman ympäristön yhteyteen. (Microsoft 2024c.)

Fabric tukee data lakehouse-konseptia, tietovarastoja sekä OneLakeä. OneLake toimii tietoaltaana koko alustalle yksinkertaistaen datan hallintaa. Fabric-tietovarastot pohjautuvat tietoaltaiden ja tietovaraston yhdistämiseen. Fabric ei kuitenkaan tekniikaltaan ole tavanomainen yritystason tietovarasto vaan enemmänkin allastyyppinen tietovarasto, joka tukee kahta elementtiä: Fabric data warehousea sekä SQL analytics endpoint:ia. Fabric data warehouse onkin nopea ja yksinkertainen ratkaisu tyypillisen SQL-pohjaisen analyyttisen tietovaraston toteuttamiseksi. (Microsoft 2024d.)

Fabric Lakehouse generoi sekä Lakehousen että SQL analytics endpoint:n, jonka avulla voidaan siirtyä sujuvasti allasnäkymän ja Lakehouse-näkymän välillä. Tämä mahdollistaa, että erityyppiset käyttäjät datainsinööreistä datatieteilijöihin voivat hyödyntää samaa resurssia. (Microsoft 2024d.)

Fabricissa lakehouse-datan tarkasteluun ja käsittelyyn voidaan hyödyntää palvelun Lakehouse Explorer -portaalia (kuva 5). Tämä graafinen portaali listaa hakemistoihin Delta- tai Parquet -tiedostot, deltataulut ja käyttöliittymän, jonka kautta näille voidaan myös suorittaa erilaisia käsittelyoperaatioita. Automaattisesti luodun SQL-päätepisteen kautta päästään käsiksi oletuksena luotuun semanttiseen malliin, jota voidaan suoraan hyödyntää esimerkiksi BI-työkalun raportin pohjana.



Kuva 5. Lakehouse Explorer

Fabric tarjoaa jaetusta varannosta Fabric-kapasiteettia, jonka tehon määritelmä on CU, joka toimii hieman tietokoneen prosessorin tapaisesti, ja se on skaalattavissa ripeästi korkeammalle SKU:lle. Fabric hyödyntää myös bursting ja smoothing -teknologioita, jolloin resurssien hetkellisesti täyttyessä se voi lisätä resursseja. Tällöin esimerkiksi kysely suoriutuu nopeammin kuin alkuperäisellä kapasiteetilla. Tällöin tuo ajallinen erotus otetaan smoothing:lla takaisin, eli CU-tehoa tasataan tietyksi ajaksi. (Microsoft 2024e.)

Hinnoittelumalli on yksinkertainen – maksetaan Fabric-kapasiteetista ja OneLakestä erikseen. Hinnoittelumalli on pay-as-you-go -tyyppinen käytettyjen resurssien perusteella, mutta myös kapasiteetin varaus onnistuu ennakkoon määritellyksi ajaksi, jolloin hinta laskee. Tietyissä tapauksissa pay-as-you-go -tyyppisessä laskutuksessa kuluja voi kuitenkin tulla bursting -teknologian käytöstä, mikäli smoothing:ia ei ilmene tai resurssi suljetaan saman tien, jolloin smoothing ei ehdi käyttämään CU:n aikaa, minkä bursting on käyttänyt. Ennakkoon varatussa kapasiteetissa ominaisuutta ei kuitenkaan esiinny (Nikola 2023).

4 Toteutuksen taustaa

4.1 Tietokantapalvelujen valinta

Toteutuksessa lähdettiin pohtimaan soveltuvia tuotteita ensin niiden tuen perusteella ADA-automatisointipalveluun, mutta kuitenkin niin, että ei jumiuduta yhteen valmistajaan. Mukaan valikointiin myös ADA:a sillä hetkellä tukematon palvelu, Microsoft Fabric, joka oli mielenkiintoinen erityisesti sen uuden tyyppisen lähestymistavan vuoksi.

Tavoitteena oli testata palvelujen kykyjä ja soveltuvuutta tietovaraston tavanomaisiin säännöllisiin eräajoihin, alustojen ominaisuuksia ja hyödyntämistä. Integraation alaisten datakulkujen määrittelyyn ja näiden pohjalta automatisoitujen työkulkujen luontiin käytettiin ADA-palvelua, jossa ominaisuus on tuettuna. Analyyttisen datan tietovarastoinnin alustoiksi ja suorituskykytestauksen pohjaksi valikoituivat lopulta seuraavat tyypit:

a. Synapse Dedicated Pool,
b. Azure Databricks ja Unity Catalog,
c. Microsoft Fabric Warehouse,
d. Snowflake Data Warehouse.

Yllä olevista palveluista pyrittiin valikoimaan yllä olevista palveluista paperilla mahdollisimman yhtäläisen suorituskyvyn ja hinnoittelun omaavia resursseja testien yhdenmukaisten mittausten saavuttamiseksi. Kuitenkin samalla pidettiin mielessä säästöaspekti valiten mahdollisimman pienen kustannuksen resursseja, jolloin edullisemman resurssin soveltuvuutta käyttötapauksiin voitiin verrata astetta suorituskykyisempiin resursseihin.

4.2 Pilvipalveluympäristön konfiguraatiot

Käytännön osuudessa pyrittiin minimoimaan pilviympäristön ja palveluiden välisiä eroavaisuuksia, erityisesti tiedonsiirron viiveitä, valitsemalla tietovarastopalvelut ja tiedon varastoinnin resurssit mahdollisuuksien mukaan samalta geologiselta alueelta. Tämän vuoksi valittiin Azuren palvelut ja niihin liittyvät tallennustilit suoraan ‘North Europe’ -alueelta, sekä sama valinta tehtiin myös Azureen integroituville palveluille, kuten Databricksille ja Snowflakelle.

Pilvitietovarastojen palveluiden edellytettiin olevan skaalattavissa, jolloin datamäärien ja hyötykuormien muuttuessa voitiin resursseja lisätä tai laskea tarpeen mukaan. Samalla tavoitteena oli luoda vertailukohtaa resurssien välillä palveluiden sisällä.

4.3 TPC-H-datajoukko

Käytännön toteutukseen käytettäväksi datajoukoksi valittiin TPC-yrityksen julkaisema suorituskykytestauksessa yleisesti käytetty, päätöksenteon tueksi tarkoitettu TPC-H. TPC on voittoa tavoittelematon yritys, jonka ideana on luoda datasentrisiä, objektiivisia ja alalle helpommin todennettavia standardinmukaisia suorituskykytestejä transaktiokäsittelyyn sekä tietokantatesteihin tarkoitettuja mittareita. TPC organisaation käyttämät testit ovat käytössä järjestelmien suorituskyvyn arvioinneissa, niitä on luotu useita eri tarkoituksiin ja niiden tulokset on julkaistu TPC:n sivuilla. (TPC.)

TPC-H datasetistä Snowflake tarjoaa eri skeemoja seuraavasti tietokannassaan: TPCH_SF1, TPCH_SF10, TPCH_SF100 ja TPCH_SF1000. Ero näiden välillä on rivien määrässä, joka suhteessa eri datasettien välillä on seuraava:

a. TPCH_SF1: Sisältää perus rivimäärän (useampi miljoona elementtiä),
b. TPCH_SF10: Perusrivimäärän x 10 (useampi kymmenen miljoonaa elementtiä),
c. TPCH_SF100: Perusrivimäärä x 100 (useampi sata miljoonaa elementtiä),
d. TPCH_SF1000: Perusrivimäärä x 1000 (useampi miljardi elementtiä).

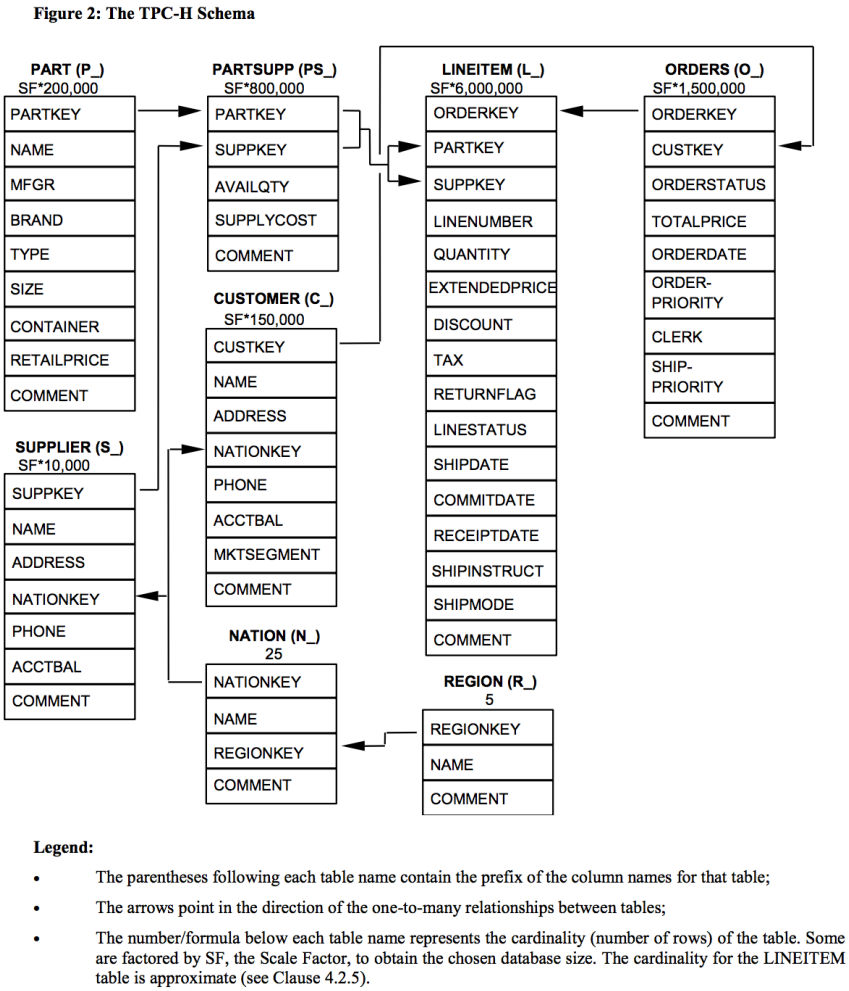
(Snowflake 2024f.)

TPC-H-suorituskykytesti mittaa järjestelmän kykyä käsitellä monimutkaisia kyselyitä huomattavan suurilla tietomäärillä. Käytännössä testi sisältää liiketoimintaorientoituneita ad hoc-kyselyitä ja samanaikaisia rakenteiden muutoksia dataan. TPC-H on kuitenkin vain yksi testeistä, jonka lisäksi löytyy esimerkiksi tätä laajempi TPC-DS, jossa kyselyt ovat monimutkaisempia ja tietomalli suuri. Osa OLAP-järjestelmistä ei kuitenkaan suoriudu edes TPC-H testeistä. (Snowflake 2024f.)

Toteutusta varten valikoidaan TPCH_SF10, joka on riittävän suuri datan ja rivimäärien osalta tuomaan eroja eri palveluiden välille, mutta kuitenkin kuvastaa suhteellisen tavanomaisia keskisuuren organisaation tietovaraston kokoa.

4.4 Tietomalli

Datasetin pohjalta muodostuu dimensionaalinen lumihiutalemalli, joka koostuu kahdeksasta erillisestä taulusta. TPC_H-skeema sisältää seuraavat taulut, attribuutit ja relaatiot (kuva 6).



Kuva 6. TPC-H skeema (Snowflake f)

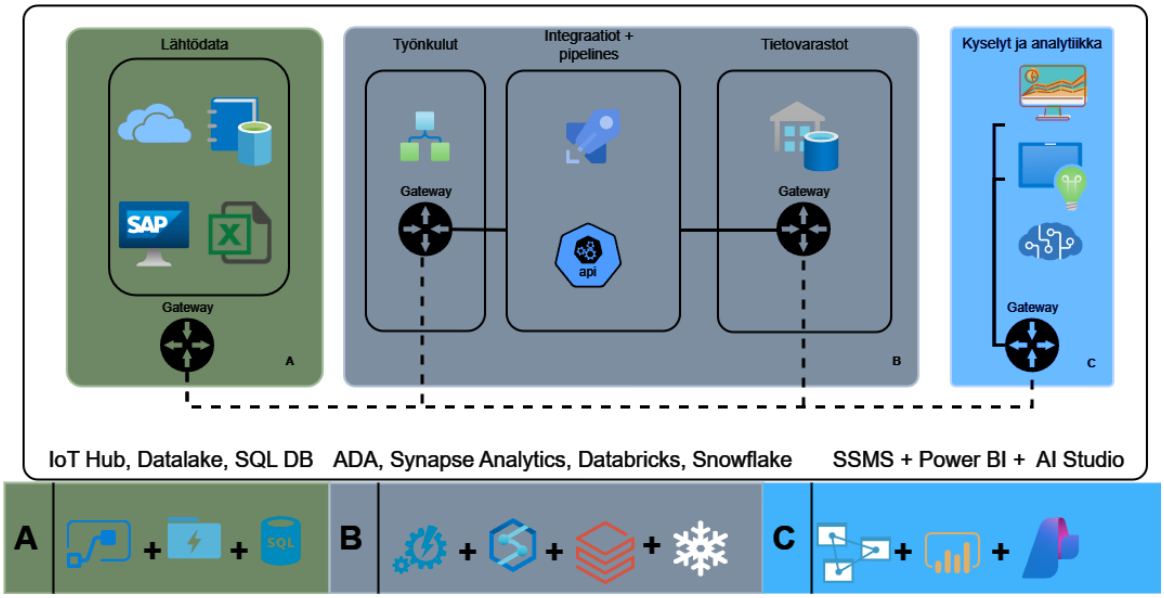
Tiedot viedään sarakemuotoisena lähteestä erillisen tietovaraston staging-alueen kautta kohdetauluihin. Käytetään sarakkeiden nimeämisessä päätteitä, mutta tuodaan kentän erottamiseksi myös kenttien nimille etuliite kuvassa sulkeissa olevien etumerkkien mukaisesti.

4.5 Dataputken vaiheet

Kuvassa 7 dataputken havainnekuva, jossa esitetty kolmessa osiossa dataputki datan alkulähteiltä prosessoinnin ja tietovaraston kautta sen hyödyntämiseen asti. Ensimmäisessä osiossa lähtödataa kerätään useista eri lähteistä, kuten sovelluksista, tietokannoista tai sensoreista. Data tallennetaan joko raakamuotoisena tietoaltaaseen tai strukturoituna tietovarastoon. Tietoallas mahdollistaa datan tallentamisen yhteen paikkaan kaikista lähteistä ja hyödyntämisen suoraan esimerkiksi analytiikan käyttöön, kun taas tietokanta sisältää strukturoitua tietoa tietovaraston analytiikan pohjalle.

Toisessa osiossa luodut työnkulut käskyttävät integraatioputkia, jotka voivat ETL-prosessien omaisesti puhdistaa, prosessoida ja rikastaa dataa ennen sen viemistä tietovarastokantaan, joka on optimoitu erityisesti analyyttisen datan käytölle. Dimensionaalinen mallinnus tietovarastokannassa helpottaa datan ymmärtämistä sekä analysointia liiketoiminnan näkökulmasta.

Kolmannessa osiossa hyödynnetään tiedon analysointiin työkaluja, kuten SSMS:ää ja Power BI:tä esimerkiksi kyselyjen generoimiseksi tai datan visualisoimiseksi. Koneoppimisen tai tekoälyn työkalut puolestaan hyödyntävät dataa ennusteiden tekemiseen, poikkeamien havaitsemiseen tai tekoälymallin opettamiseksi. Dataa voidaan käyttää sekä tietovaraston lähteistä että suoraan raakamuotoisena tietoaltaalta tai yhdistelemällä molempia. Tekoälymallit voivat hyödyntää dataa esimerkiksi asiakaskäyttäytymisen ennustamiseen tai prosessien optimointiin.



Kuva 7. Dataputken vaiheet

Käytännön osuudessa dataputkesta hyödynnetään pitkälti vain keskimmäistä B-osiota. Tietovarastoon on tuotu aiemmin kuvattu TPC_H-datasetti dimensionaalisiin staging-alueen tauluihin valmiiksi kolumnimuotoisena ja määritellyillä tietotyypeillä. Testejä varten ajetaan työnkulkujen avulla data ensin tyhjiin kohdetauluihin. Muutetaan dummy-kentän arvoja, jotta voidaan testata dimensio- ja faktataulujen muutosten vaikutuksia Update ja Delete-Insert SQL-operaatioilla.

Kyseinen osio voidaan tehtävien mukaan jakaa kolmeen tehtävään: datakulkujen ja automatisoitujen työnkulujen luonti, integroinnit ja dataputket, eräajot ja datan vienti tietovarastoon. Dataputken luontiin käytetään joko ADA-palvelun sisäistä dataflow-toimintoa, joka luo dataintegraatiot valittuun palveluun, kuten Synapseen, tai käytetään palvelun sisäistä

työputken luontityökalua Microsoft Fabricin tapauksessa. Luodaan vielä erikseen datakulut staging-latausten lisäksi dimensionaali- ja faktataulujen latauksille.

Luodaan myös tietovarastoon kohdetaulut vastaavine tietotyyppeineen tietovaraston tai lakehousen tauluille ennen ajojen käynnistystä. Ajettavat työnkulut käynnistävät datankulut määriteltyjen riippuvuuksien mukaisesti, jota ennen resurssit tulee myös käynnistää pilviportaalissa, mikäli ne on pysäytetty tai lisätä resurssin käynnistys työnkulkuun ennen sen ajamista.

Tietovaraston tai lakehousen tiedot tallentuvat ennalta kartoitettuihin tietokannan taulujen tai Databricks'ssä deltataulujen kenttiin. Kun data on viety kohdetauluihin, tehdään muutokset staging-taulujen tietoihin muuttamalla dummy-kenttien tietoa arvosta 0 arvoon 1 tai toisinpäin. Lopuksi ajetaan vielä tietokantakyselyt palvelukohtaisesti kyselyissä.

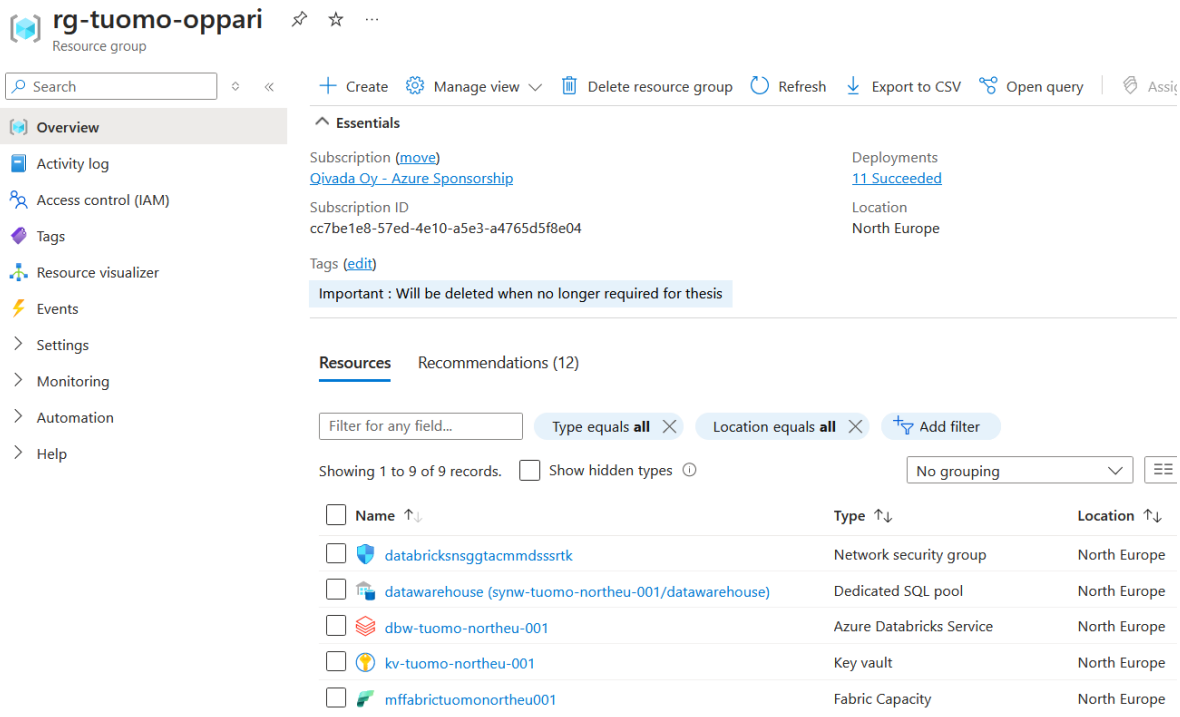
Tuotannonomaisissa ympäristöissä tietokantakyselyistä voisi tehdä esimerkiksi näkymät tietokantaan, jolloin niitä voisi hyödyntää suoraan viimeisen osion BI-työkaluissa. Mikäli ympäristöä ajettaisiin säännöllisesti eräajoissa, erillisen staging-alueen voisi siirtää esimerkiksi tietoaltaan prosessoitavaksi säästäen tietovaraston resursseja analytiikan käyttöön (Parkhouse 2021).

5 Toteutus

5.1 Resurssien luonti

Toteutuksessa oletuksena on, että käyttäjä omaa pilvipalveluun riittävät oikeudet, jotta tarpeelliset resurssit voidaan luoda Azure-tilauksen alle sekä liitännäisiin palveluihin. Näin on usein oletuksena siinä tapauksessa, jos tarkoitukseen on käytetty esimerkiksi monen oppilaitoksen tarjoamaa opiskelijatilausta. Organisaatioissa ei usein ole riittäviä oikeuksia, vaan lupia täytyy pyytää yrityksen IT-osastolta tai muulta hallinnoivalta taholta. Tämä voi viedä aikaa ja hidastaa itse toteutusta.

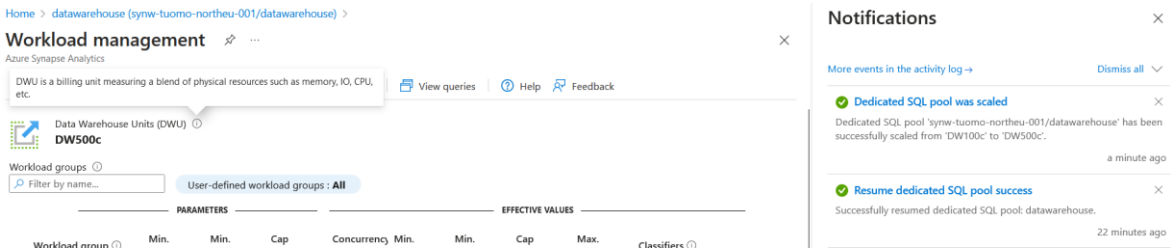
Kuvassa 8 resursseja varten Azure:ssa luodaan ensiksi resurssiryhmä ”rg-tuomo-oppari”, jonka alle tuodaan suoraan kaikki työssä käytetyt resurssit ja palvelut. Tämä toimii testauksessa hyvin, sillä ympäristö halutaan pitää mahdollisimman yksinkertaisena testauksia varten. Tuotantoympäristössä resurssiryhmiä tyypillisesti on useampia palvelu- tai palvelukokonaisuuskohtaisesti. Resurssiryhmien välille voidaan luoda virtuaaliverkon päätepisteitä ja muokata pääsyhallintaa esimerkiksi käytetyn palvelun mukaan tai esimerkiksi erottamaan kehitys ja tuotantoympäristöt toisistaan.



Kuva 8. Azure resurssien luonti

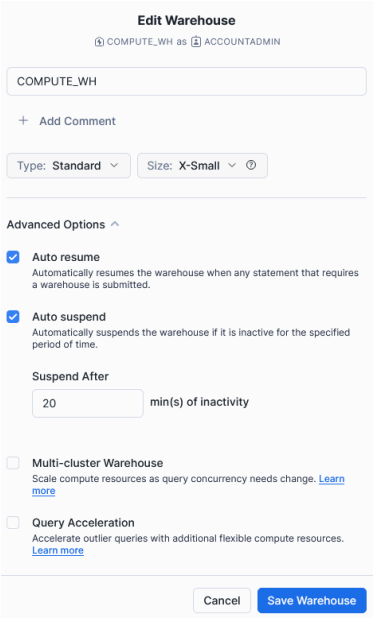
Synapse Analytics palvelua varten luodaan Synapse Workspace -resurssi, jonka alle liitetään storage account - Azure Data Lake Storage Gen2 tyyppinen -tietoallas. Tietoallasta hyödynnetään muun muassa Snowflaken dataputken lastauslaiturin interim stage -

toimintoa sekä Databricks:n Unity Catalog:n metatietovarastolle. Synapse -työtilan alle luodaan dedikoitu SQL tietovarasto -resurssi, jolle määritetään muun muassa DWU:n taso (kuva 9).



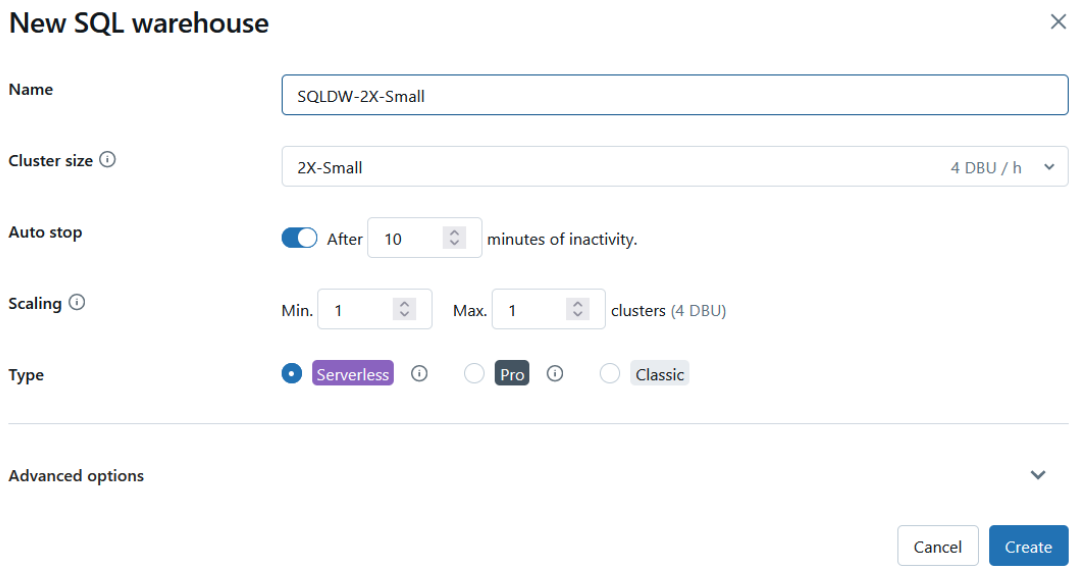
Kuva 9. Synapse dedicated SQL pool

Snowflake:n alle luodaan sekä virtuaalinen tietovarasto sekä tietokantaa varten Snowflake -tilauksen alle tietokanta. Snowflake:ssä luodaan myös laskentaresurssi ”COMPUTE_WH”, jolle määritetään muun muassa sen koko ja klusterien määrä ja päällä oloaika (kuva 10).



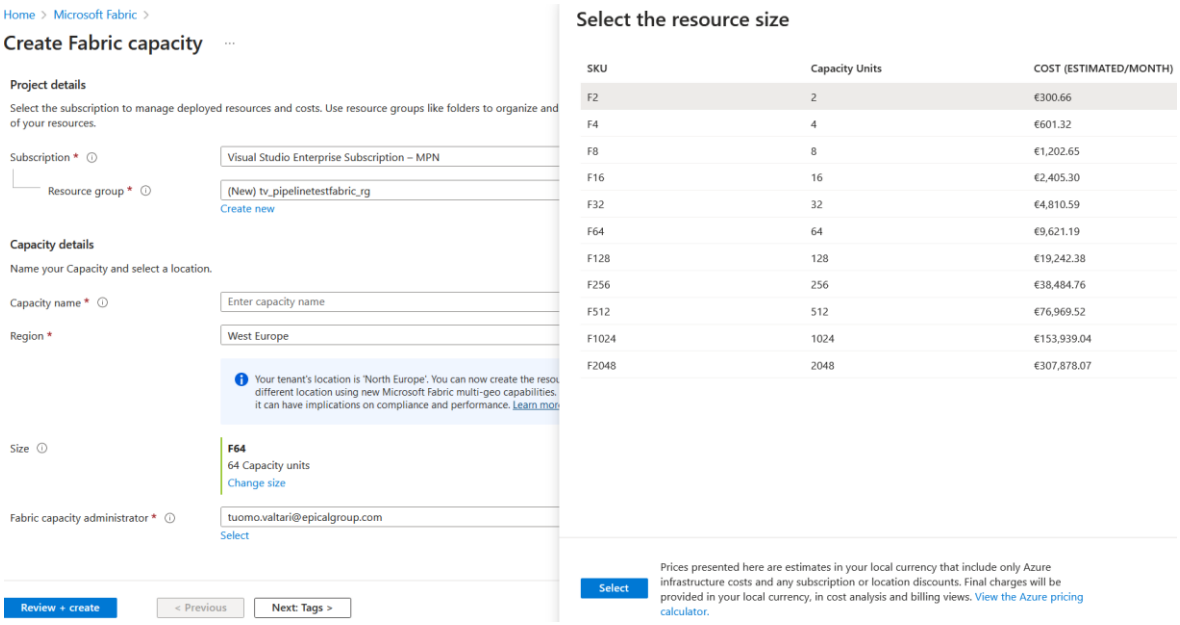
Kuva 10. Snowflake virtual warehouse

Databricks:ssä otetaan Unity Catalog käyttöön (oletuksena käytössä uudemmissa ympäristöissä), jonka jälkeen luodaan SQL Warehouse -tyyppinen laskentaklusteri. Nimetään laskentamoottori, valitaan serverless -tyyppi, klusterin koko, skaalauksen taso ja tyyppi (kuva 11).



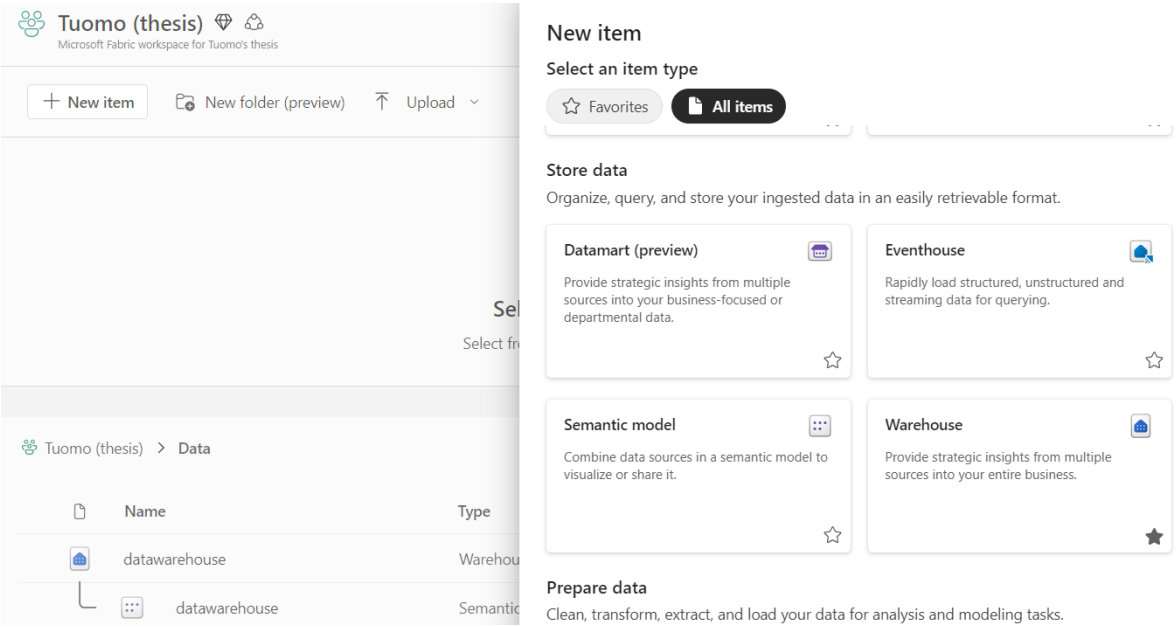
Kuva 11. Databricks SQL warehouse

Microsoft Fabric:ia varten luodaan Azureen Fabric -kapasiteetti, jolle valikoidaan resurssiryhmä, regioona sekä kapasiteetin koko (kuva 12).



Kuva 12. Fabric kapasiteetti

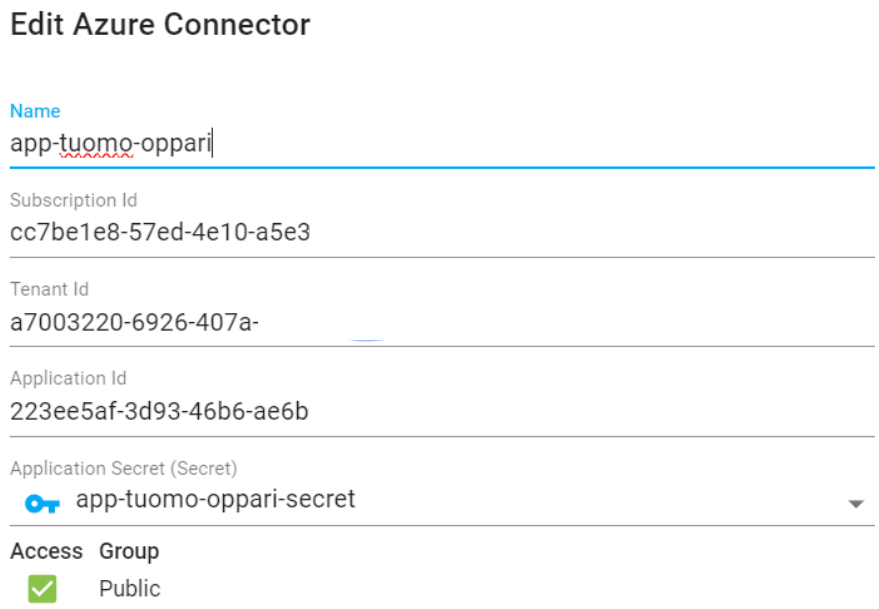
Fabricissa sen palvelut ovat sidoksissa käytettyyn kapasiteettiresurssiin tilauksen alla, ja ovat käytettävissä, kun kapasiteetti vain on käynnissä. Resurssit tulee kuitenkin luoda erikseen tarpeiden mukaan, esimerkiksi tietovarastoa varten Warehouse-resurssi (kuva 13).



Kuva 13. Fabric warehouse

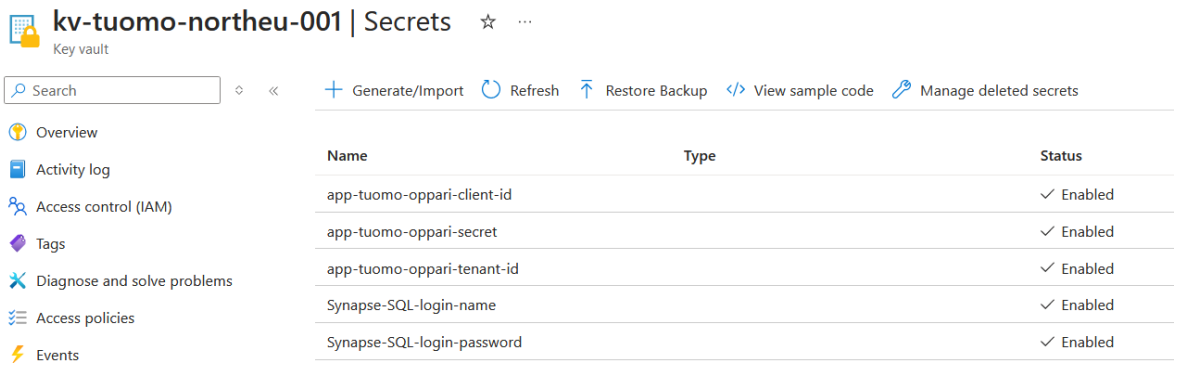
5.2 ADA:n yhteysmäärittelyt

ADA:a hyödyntämiseksi käytetyissä palveluissa, lisätään se Azuressa Microsoft Entraan applikaatiorekisteröinnillä, josta saadaan sovellukselle Application Secret. Application Secretin sekä tenant tietojen avulla saadaan lisättyä ja määritettyä ADA:an Azure Connector sen hyödyntämiseksi Azuren eri palveluissa (kuva 14).

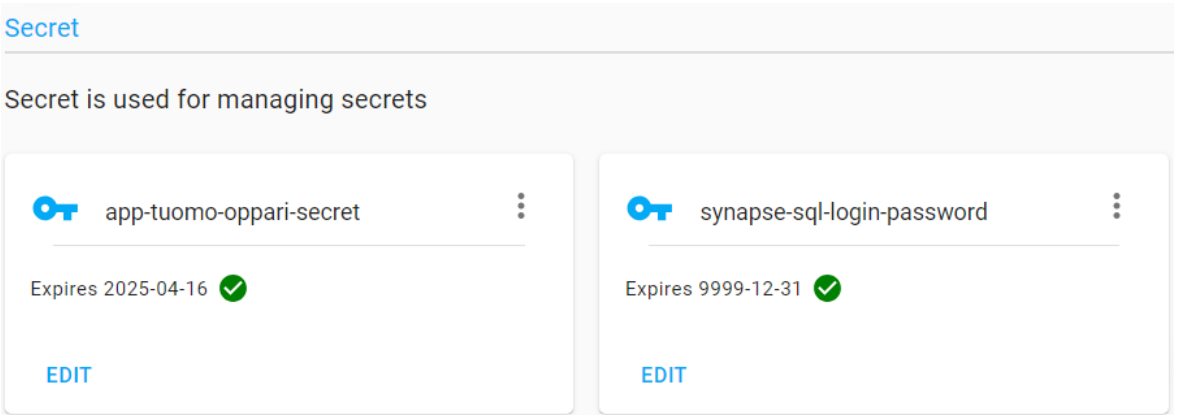


Kuva 14. ADA Azure Connector

Jotta vältytään käyttämästä muun muassa tietokantapalveluja suoraan avoimen tekstin tunnuksilla ja salasanoilla yhteysparametreissä, luodaan Azuren key vaultiin halutut secretit (kuva 15) määritetään ADA hyödyntämään secretejä ympäristökohtaisesti (kuva 16).

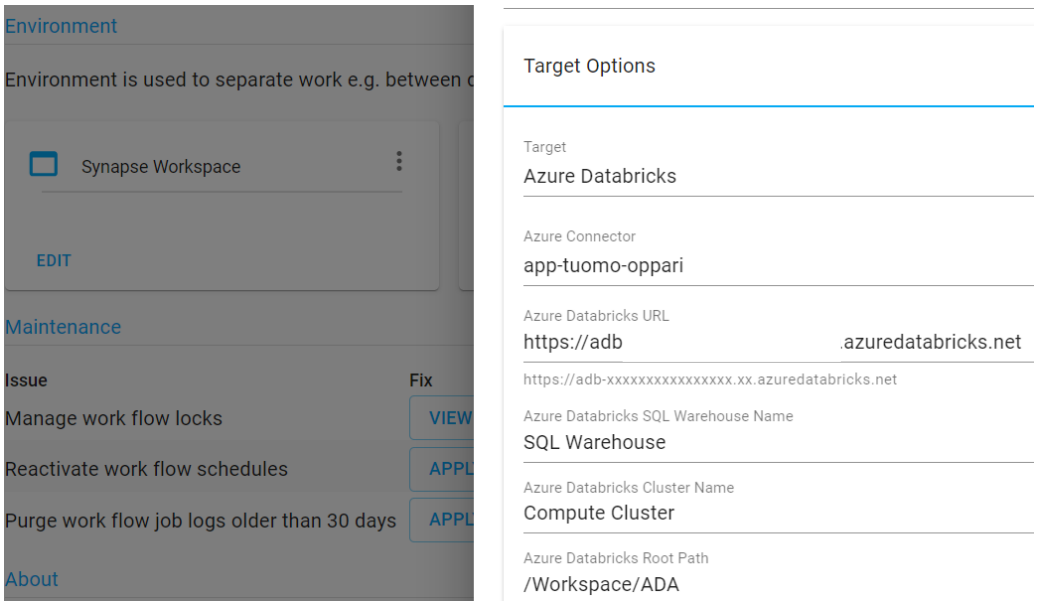


Kuva 15. Azure key vault secret



Kuva 16. ADA secretien hallinta

Environment kohdassa yhden ympäristön tapauksessa voitaisiin määritellä esimerkiksi kehitys, testi ja tuotantoympäristöt eri parametrein. Työssä käytetään useampaa palvelua ja vain kehityksen työtiloja, joten määritellään Synapse ja Databricks erikseen omina ympäristöinä ADA:an, joihin käytetyn palvelun mukaan määritellään muun muassa SQL Warehouse sekä Compute-resurssien tiedot (kuva 17).



Kuva 17. ADA environment määrittely

5.3 Taulujen luonti

Data on TPC-H-datasetissä valmiiksi sarakemuotoisesta, joten sille ei vaadita ETL-käsittelyä sen tiedostoformaatin tai kenttien kartoituksen osalta, vaan voidaan käyttää nimeämisessä samaa logiikkaa. TPC-H-datasetin dataa ei kuitenkaan ole jokaisessa palvelussa suoraan lähteestä saatavilla, joten luodaan erillinen alue sekä lähdetauluille että laiturialue stage-tyyppisille latauksille. Tällä myös pyritään eliminoimaan muun muassa erot tiedonsiirrossa lähdejärjestelmän ja kohdejärjestelmän välillä. Käytetään taulujen luontilausekkeissa yhtäläisiä tietotyyppejä lähdetauluista kohteeseen viedessä, jotta erillisiä konversioita ei datalle tarvita sen kohdetauluun viemisessä.

Käytetään nimeämiseen seuraavaa muotoa: stage -taulun ilmentämiseksi x-kirjainta nimen alkuun, jonka jälkeen lisätään lähde datalle, esimerkiksi ”SnowflakeData” ja tämän jälkeen käsiteltävä entiteetti, kuten ”Customer”. Esimerkin mukaisesti Customer taululle Snowflake:ssä: ”X_SnowflakeData_Customer”.

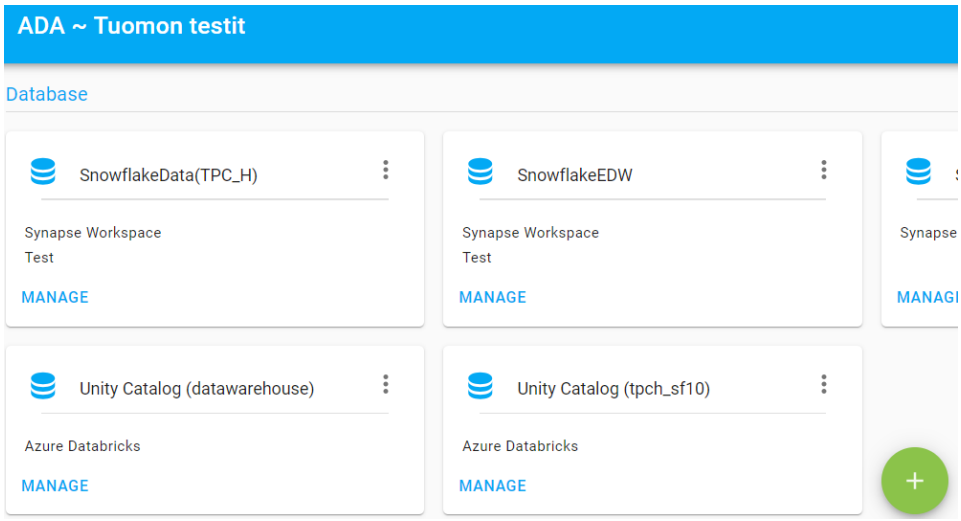
Laiturialueelle tuotaessa luodaan samalla ”DUMMY” -kenttä Integer-datatyypillä ja asetetaan kenttään staattinen arvo nolla. Muutetaan ensimmäisten latausten jälkeen kentän arvoa kaikille riveille lukuun yksi, jotta voidaan todentaa muutosten vaikutus ajoketjun suoritukseen.

Jaetaan kohdetaulut niiden tyypin mukaisesti dimensio-, ja faktatauluihin. Nimeämisessä käytetään tauluille etuliitteinä D_, tai F_ taulun tyypin mukaan, jonka jälkeen kyseisen taulun nimi, eli esimerkiksi D_Customer. Käytetään kohdetauluille oletusskeemaa ”dbo” muissa

palveluissa kuin Databricks:ssä, jossa käytetään skeemaa ”datawarehouse” Unity Catalog:n metastoren katalogin ”unity-catalog” alla.

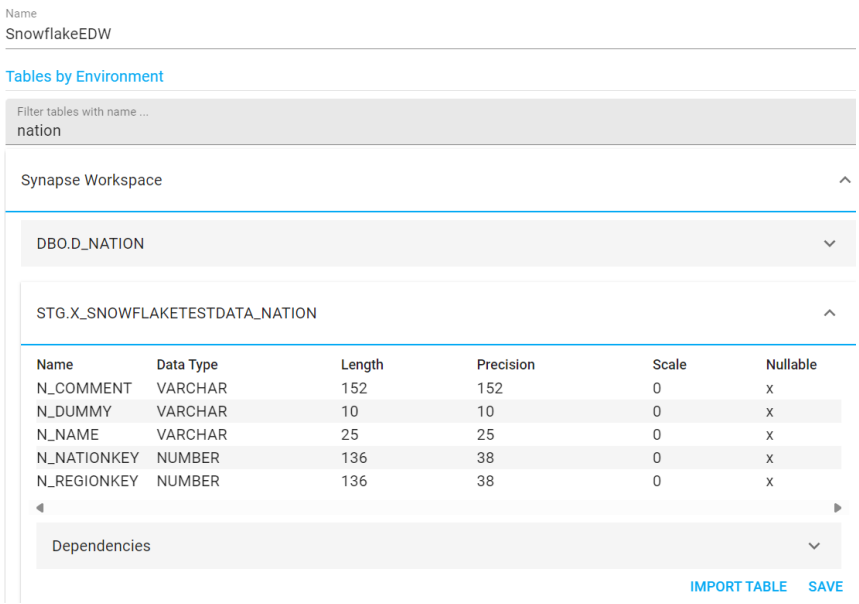
5.4 Taulujen ja rakenteiden tuonti ADA:an

Jotta voimme käyttää tauluja ADA:ssa dataputkien määrittelyiden pohjana, tulee taulut ja niiden rakenteet ensiksi tuoda palvelukohtaisesti palveluun. Lisätään määrittelyt Storage-toiminnallisuuden alta sekä parametrisoinnit ympäristöittäin ja palvelukohtaisesti (kuva 18).



Kuva 18. ADA storage osio

Hallinta tietolähteille tehdään Manage-painikkeella, josta voidaan ympäristökohtaisesti tuoda tietolähteen näkymät, taulut ja tietotyypit, jota voidaan suodattaa taulukohtaisesti (kuva 19).

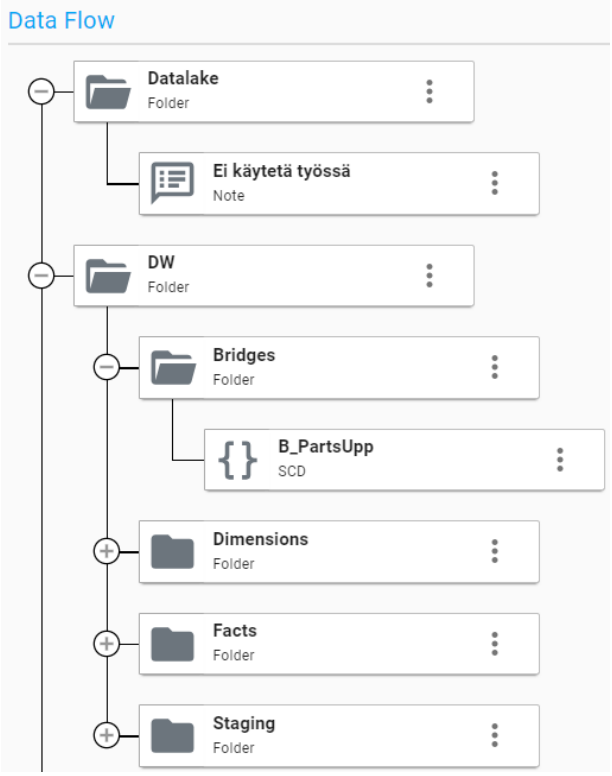


Kuva 19. ADA metadatan tuonti

Lähdekysely on myös mahdollista tehdä suoraan SQL-kyselynä kantaan, jolloin voidaan laajemmin muokata ja tehdä esimerkiksi useamman taulun yhdistelevän kyselyn pohjalta uusi lähdekysely tiedon tuontia varten. Tällöin ADA tarkastaa tiedon tuonnin vaiheessa, että kysely on muutoin validi.

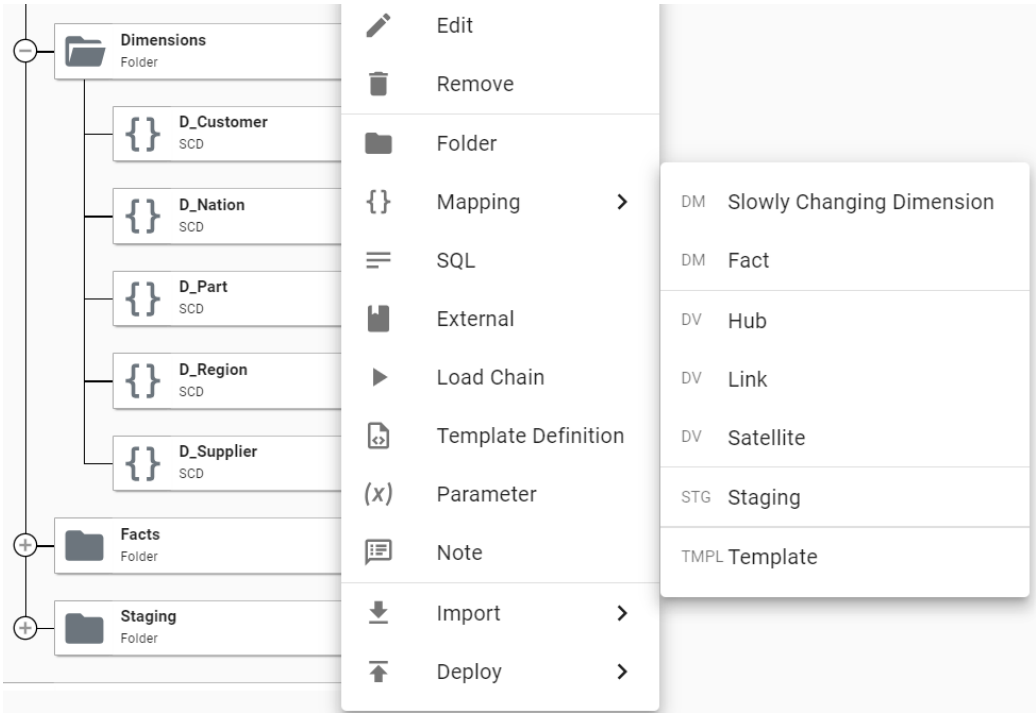
5.5 Datakulkujen luonti ja määrittelyt

Datakulkujen luonti ja muokkaus ADA:ssa tehdään ”Data Flow” -toiminnallisuuden alta. Data flow:n alle voidaan luoda hakemistot ja alihakemistot selkeyttämään niiden eri lähteitä tai esimerkiksi latauksen tyyppiä (kuva 20). Hakemistot toimivat hierarkkisesti, eli esimerkiksi deploy ylemmällä tasolla valitsee kaikki alemmatkin hakemistot ja komponentit.



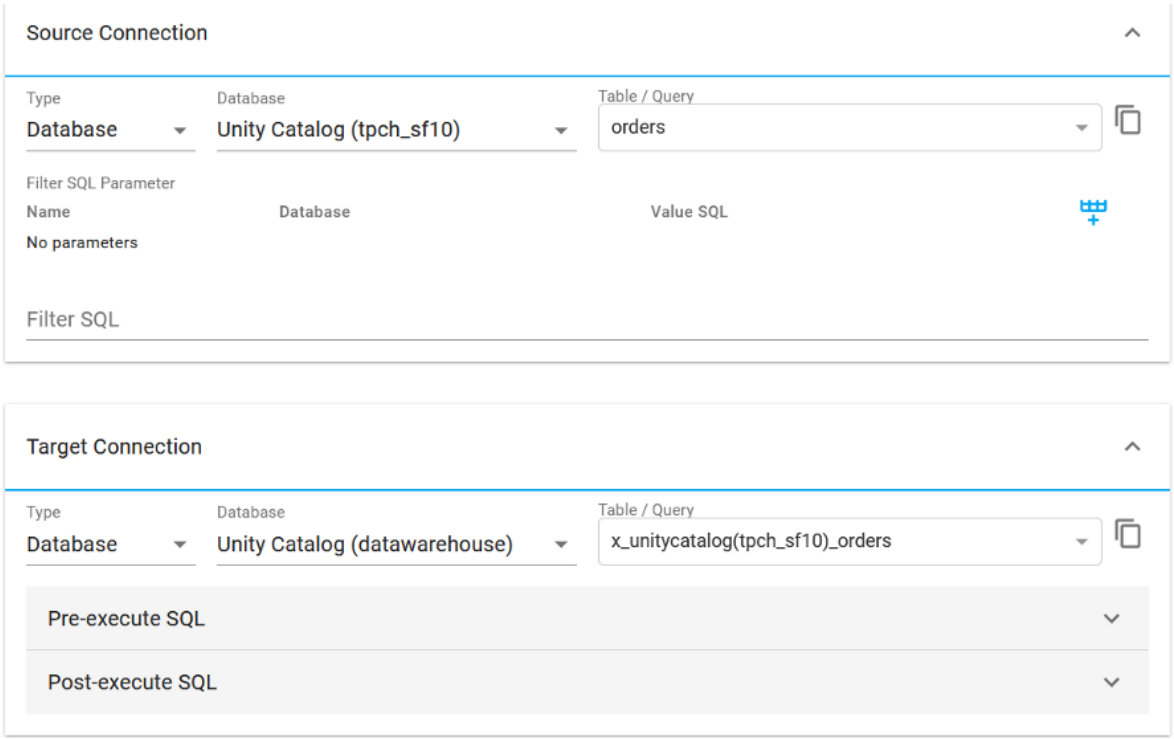
Kuva 20. ADA data flow osio

Datakulun tyyppi määritellään latausta luodessa. Työssä käytetään valmiiksi määriteltyjä ympäristökohtaisia pohjia fakta- dimensio-, ja staging -latausten kartoituksille, mutta optiona on käyttää esimerkiksi templateja, joilla voidaan Databricks notebookeja ja määrittää niille latauskohtaisesti halutut parametrit. Valmiiksi luotuihin kartoituksiin voidaan käyttää suoraan ADA:n omaa määrittelytyökalua, jolloin halutut datan kartoitukset, konversiot ja muut tekijät määritetään lähde- ja kohdetaulun pohjalta (kuva 21).



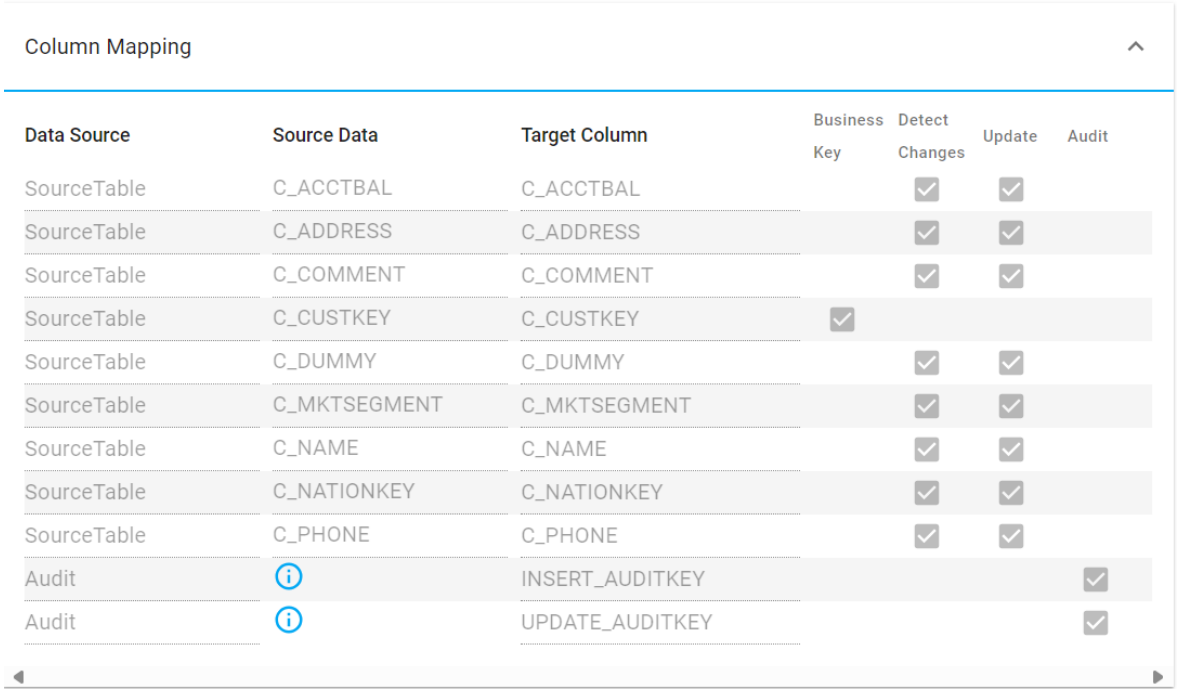
Kuva 21. ADA kartoitus latauksille

ADA:ssa datakulun määrittely tehdään ympäristökohtaisesti lähdetietojen ja kohdetietojen määrittelyllä (kuva 22). Options -valikon alta valitaan myös kyseiselle lataukselle sopiva tyyppi muun muassa muutosten hallintaan.



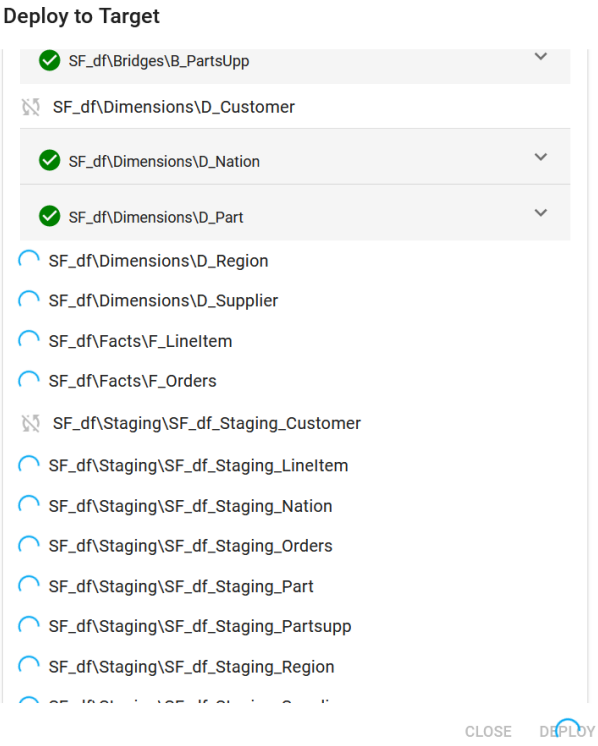
Kuva 22. ADA latauksen määrittely

Faktojen ja dimensioiden tapauksessa tehdään myös lähteen, eli saman entiteetin staging- ja kohdetaulujen välillä sekä businessavaimen ja dimensioiden tapauksessa muutoskenttien määrittelyt. Mikäli kenttien määritykset sekä lähteessä tai lähdekyselyssä ovat yhtäläiset, ADA tunnistaa kohdistettavat kentät automaattisesti (kuva 23).



Kuva 23. ADA kenttien kartoitus

Kun latauksen tyyppi, kenttämääritykset ja muut halutut määritykset on tehty, tehdään latauksien vienti ympäristö- ja palvelukohtaisesti ”Deploy to Target” avulla. ADA näyttää laturikohtaisesti, mikäli määrityksissä on tapahtunut muutoksia kyseisen ympäristön kohdalla (kuva 24).



Kuva 24. ADA työnkulkujen vienti ympäristöön

ADA:sta viennin jälkeen dataputket ilmaantuvat palveluiden alle orkestroitaviksi. Snowflake ei kyseisiä luotuja dataputkia näytä erikseen palvelussaan, mutta Synapsen kautta ajoja ajettaessa, voidaan suoritettuja SQL-käskyjä seurata Snowflaken Monitoring -välilehden kautta.

5.6 Datan tuonti

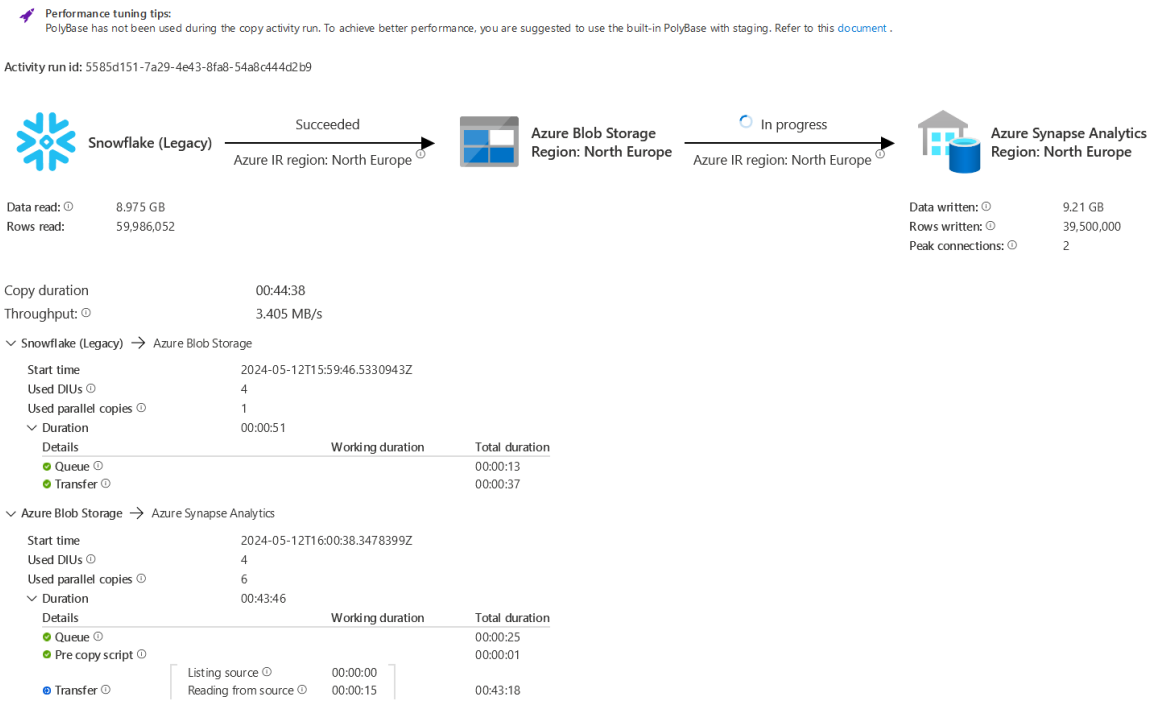
Datan tuonti aloitetaan staging-alueen latauksilla. Staging-alueelle tuotaessa määritetään lähteeksi SF_10 datasetti ja sen taulut. Lisätään myös kenttä ”dummy”, jotta voimme muuttaa sen arvoa jälkikäteen DELETE-INSERT ja UPDATE SQL-operaatioiden testausta varten. Staging:lle määritetään myös tietotyypit mahdollisimman yhtäläisiksi jo kohdetaulujen kanssa, jotta erillisiä kenttien konversioita ei tarvita taulujen välillä (kuva 25).

```
CREATE TABLE [stg].[X_SnowflakeTestData_customer]
(
    [c_custkey] [int] NULL,
    [c_name] [varchar](25) NULL,
    [c_address] [varchar](40) NULL,
    [c_nationkey] [int] NULL,
    [c_phone] [char](15) NULL,
    [c_acctbal] [decimal](15, 2) NULL,
    [c_mktsegment] [char](10) NULL,
    [c_comment] [varchar](117) NULL,
    [c_dummy] [varchar](10) NULL
)
```

```
create TABLE datawarehouse.dbo.D_CUSTOMER (
    C_CUSTKEY BIGINT NOT NULL,
    C_NAME VARCHAR(25),
    C_ADDRESS VARCHAR(40),
    C_NATIONKEY BIGINT,
    C_PHONE VARCHAR(15),
    C_ACCTBAL DECIMAL(15,2),
    C_MKTSEGMENT VARCHAR(10),
    C_COMMENT VARCHAR(117),
    C_DUMMY VARCHAR(10),
    INSERT_AUDITKEY BIGINT,
    UPDATE AUDITKEY BIGINT,
```

Kuva 25. Taulujen luonti tietokantaan

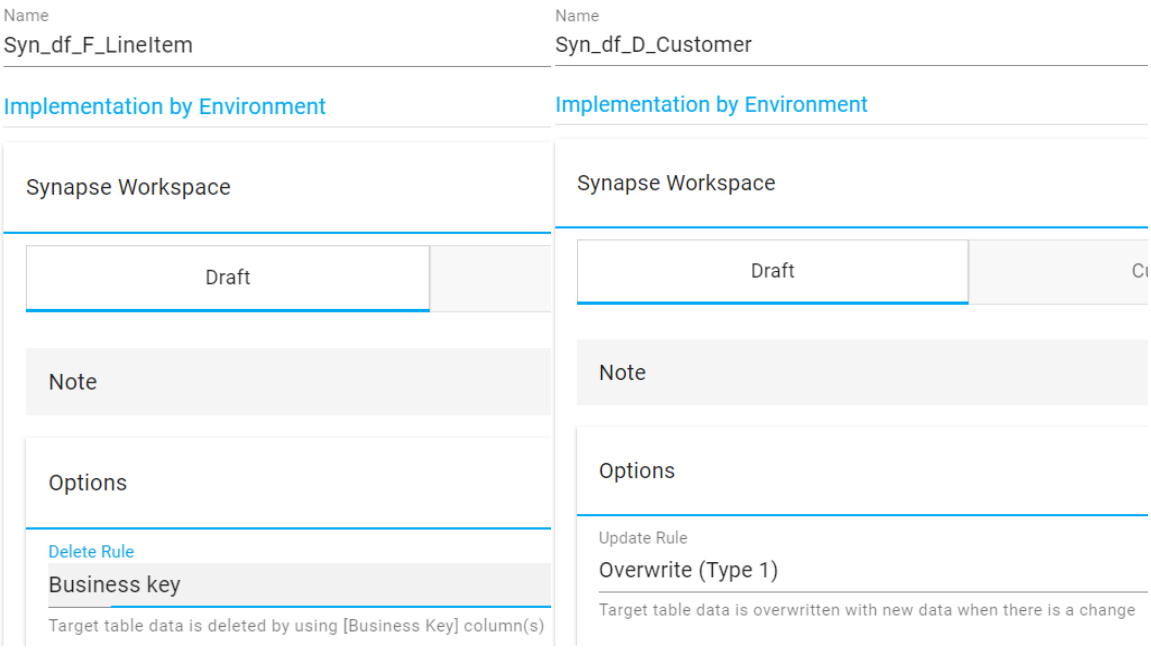
Dataa staging-alueelle siirretään lähteestä suoraan Copy-data operaatiolla lähdetaulusta kohdetauluun (kuva 26). Snowflakeä käytettäessä latauksen kohteena tulee välissä käyttää blob-storage-tyyppistä interim staging -varastoa, jolle data siirretään ennen sen kohdetauluun vientiä. Myös Synapse ja Fabric latauksissa käytetään interim staging storea, sillä lähteenä on hyödynnetty suoraan Snowflaken SF10-datasetin tauluja. Synapsen sisäisissä latauksissa suorituskykyisempi valinta voisi olla esim. PolyBase isoille datamäärille tai COPY command -komento, jolloin vältetään interim stage -välivaihe, mikäli vain lähtöformaatti on oikeanlainen.



Kuva 26. Datan siirtyminen staging-alueelle

DataBricksin osalta vastaavia määrityksiä staging-alueelle ei ole tehty, vaan dataa ladataan suoraan deltalaken tauluihin copy -operaatiolla palveluun ladatuista lähdetauluista. Tarkoitukseen käytetään Insert-Overwrite tyyppistä sql-lauseketta.

Kohdetauluihin datan tuomiseen käytetään dimensiotauluille hitaasti muuttuvan dimension tyyppiä 1, eli yli kirjoitetaan päivittyneet tiedot ja faktatauluille taas business-avaimen perusteella rivien poisto ja -lisäys. Määritys eri taulujen lataustyypeille voidaan tehdä ADA:n data flow:n määritysten alla ympäristökohtaisesti (kuva 27).

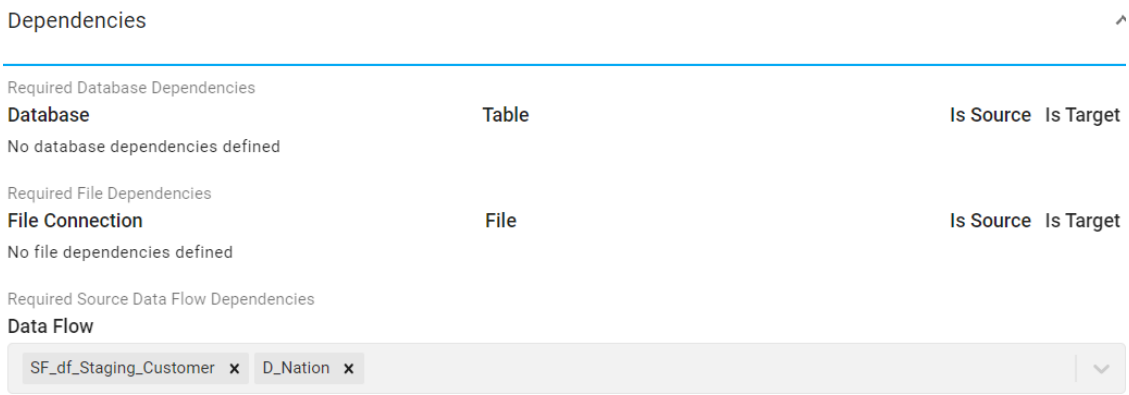


Kuva 27. Latausten tyypit faktat ja dimensiot

Datan tuonti fakta- ja dimensiotauluihin tuonti voi tavanomaisesti sisältää muun muassa tiedon yhdistämistä, aggregointia ja muokkausta tietokannan taulun formaattiin. Puhtaissa suorituskyvyn testauksissa pyritään minimoimaan eri muunnosten ja muokkausten osuutta, joten data viedään staging-alueelta suoraan vastaaviin kenttiin kohdedimensiossa ja -faktoissa saman palvelun sisällä. Itse datan muokkaus tehdään suoraan staging-alueen tauluihin, jolloin eri operaatioiden vaikutuksen voidaan todeta kohdetaulujen ajoketjuissa.

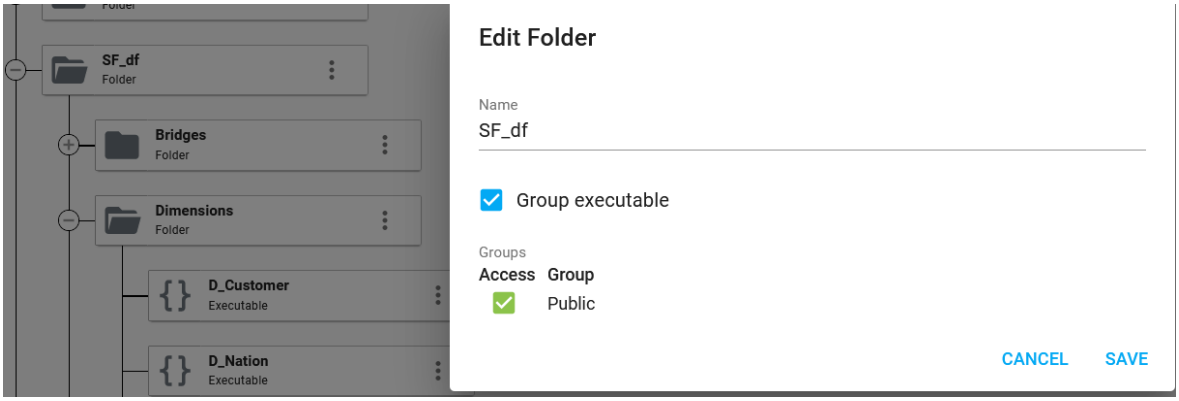
5.7 Työnkulkujen luonti

Työnkulkujen luonnissa ja niiden suorittamisessa hyödynnetään ADA:n ”Workflow” -toiminnallisuutta. Asetetaan ensivaiheessa riippuvuudet muiden ajoketjujen suoritukseen jokaisen taulun kohdalle, joiden yhteisvaikutusten mukaisesti niiden suoritusjärjestys määräytyy työnkulkuun (kuva 28).



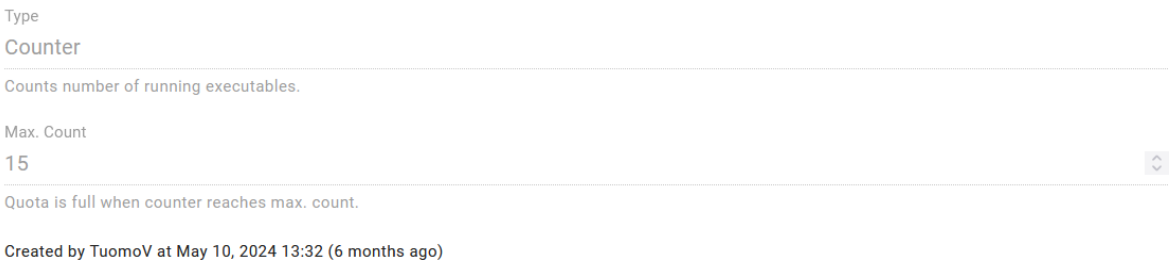
Kuva 28. ADA riippuvuuksien määritys

Tehdään ADA:ssa halutuille workflow:ille deploy ympäristökohtaisesti, jonka jälkeen näitä voidaan hyödyntää workflown:n määrittelyssä. Deploy luo kuluista Executable:n Workflow -välilehden alle. Jotta jokaista kulkua ei tarvitse erikseen määrittää workflow alle, voimme kansiotasolla määrittää kulut erilliseksi ajoryhmäksi (kuva 29).



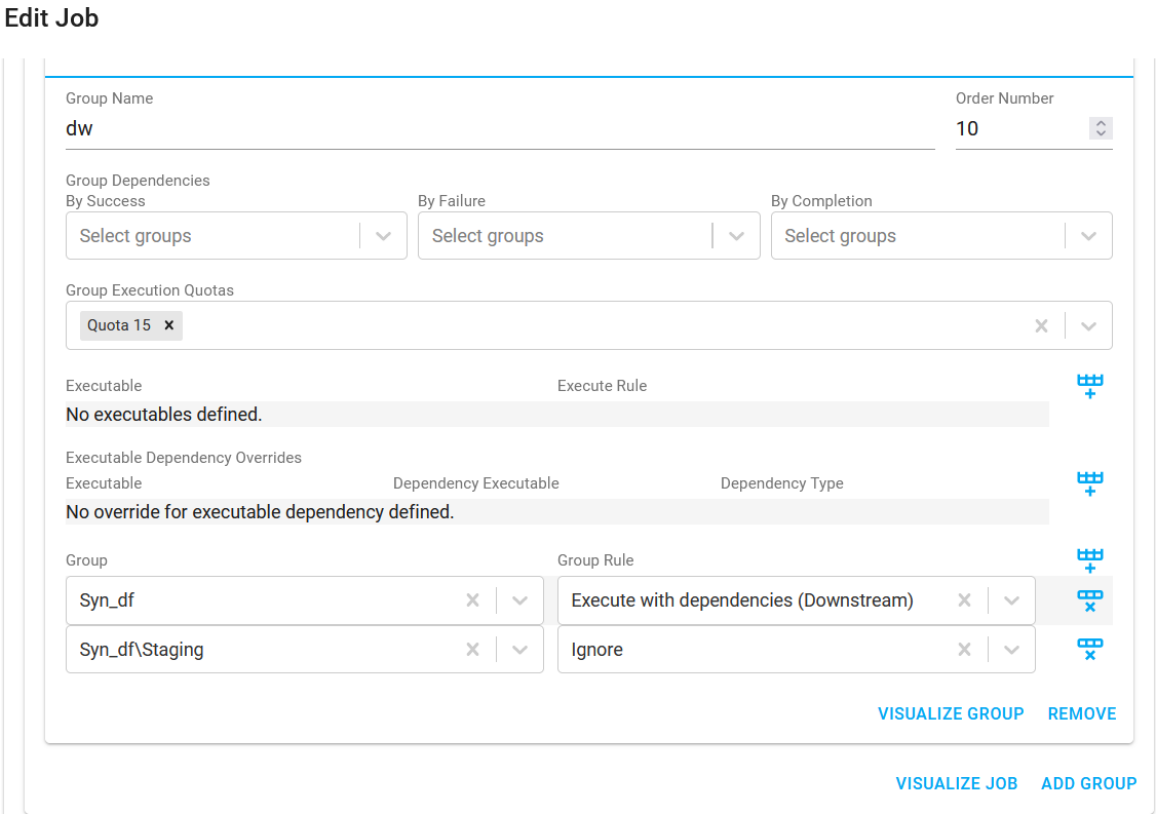
Kuva 29. ADA hakemisto executable:ksi

Luodaan myös oma ”Quota” -elementti, jonka sisälle voidaan määrittää raja-arvo yhtäaikaisten kulkujen suoritukselle ajoketjussa (kuva 30). ”Quota” -elementtiä voidaan hyödyntää työkulun määrittelyssä ”Execution Group” alla määrittelyssä.



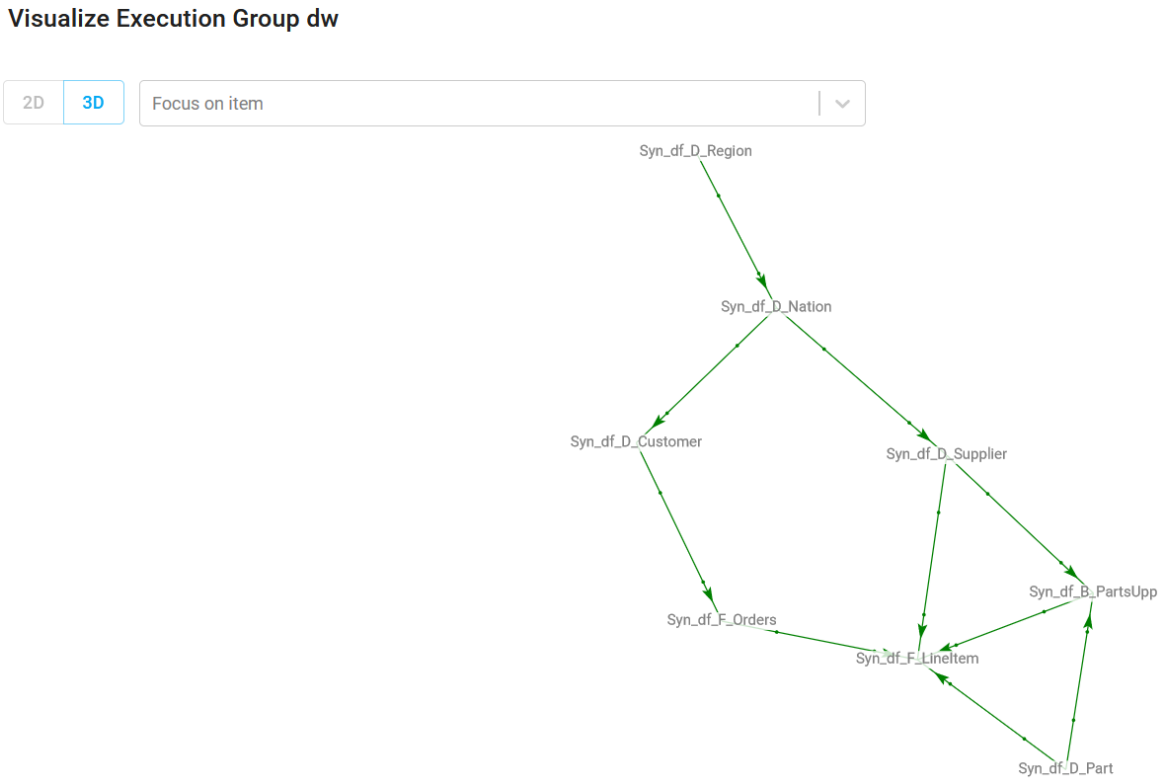
Kuva 30. ADA quota counter

Luodaan työkulut group executable -kansioiden pohjalta. Luomalla useamman group executablen erityyppisille latauksille voimme luoda erilliset ajoketjut kaikille liittyville latauksille, staging -latauksille että vain kohdetauluihin viemiseksi. Kuvassa 31 viedään halutut suoritusryhmät Execution Groups alle haluttuun ryhmään. Voimme sekä määrittää että rajata pois suoritusryhmiä tai yksittäisiä kulkuja.



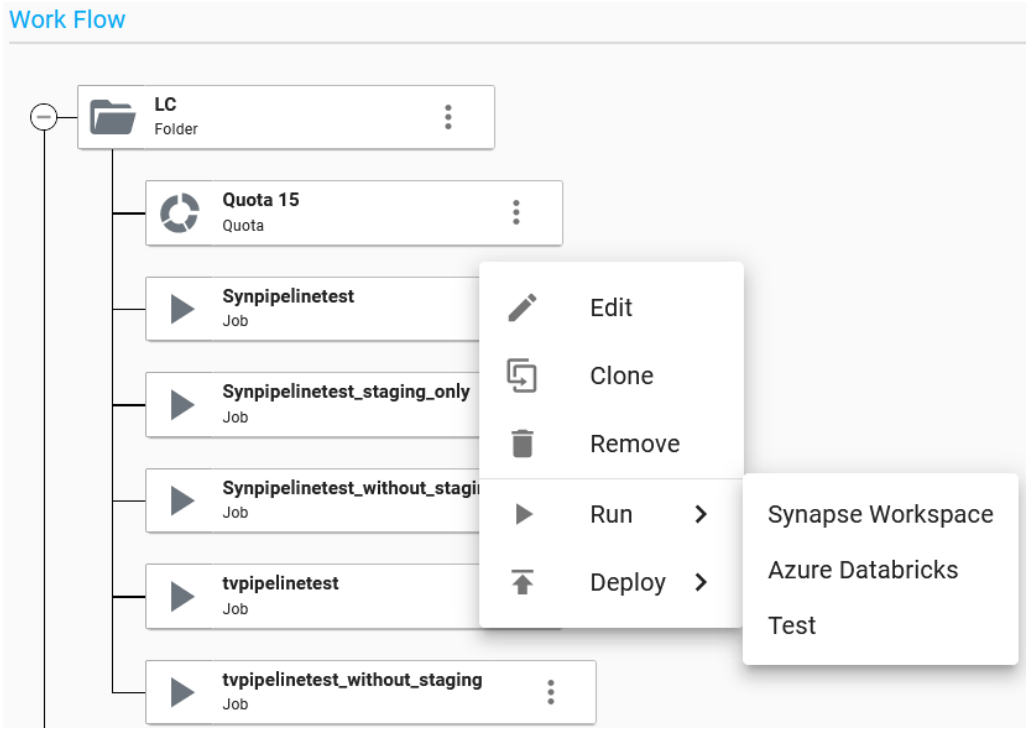
Kuva 31. ADA workflow muokkaus

Työnkulkuun määriteltyjä executableja ja näiden riippuvaisuuksia toisiin ajoihin voidaan tarkastella “Visualize Group” -painikkeella (kuva 32). Mikäli execution groupeja on useampi per työnkulku, voidaan näiden riippuvuuksia tarkastella “Visualize Job” -painikkeella. Tällöin voidaan asettaa riippuvuus seuraavaan execution group:iin, mikäli esimerkiksi kyseinen execution group suoriutuu loppuun asti onnistuneesti.



Kuva 32. ADA execution group -visualisointi

Luotua työnkulkua voidaan ajaa ympäristökohtaisesti käyttämällä “Run” -toimintoa ja valitsemalla haluttu ympäristö (kuva 33). Manuaalinen työnkulun käynnistys varmistaa vielä tämän jälkeen, jotta varmasti valitaan oikea ympäristö ja työnkulku virheiden vähentämiseksi.

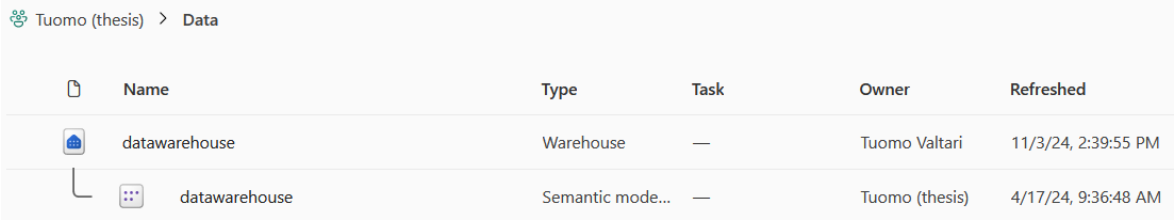


Kuva 33. ADA työnkulun manuaalinen ajo

5.8 Microsoft Fabric – dataputket ja työnkulun luonti

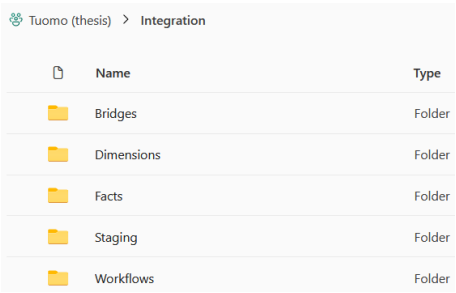
Microsoft Fabricilla ei vielä tämän opinnäytetyön toteutusvaiheessa ollut tukea ADA:aan rajapinnan rajoitteiden vuoksi, joten työssä päädyttiin käyttämään Fabricin sisäisiä työkaluja data- ja työnkulkujen luontiin. Työn kirjoituksen loppuvaiheilla tuki kuitenkin on jo olemassa ja näin ollen työkalua voidaan hyödyntää muiden palveluiden tapaan ADA:ssa. Työn toteutukseen ympäristö ja työkulut pyrittiin muutoin rakentamaan mahdollisimman yhtäläiseksi Synapse-toteutusten kanssa.

Luodaan tarkoitusta varten ensiksi yhteinen työtila ja kansiot tämän pohjalle näiden käyttötarpeen perusteella. Kuvassa 34 datahakemiston alle luodaan Fabric warehouse -resurssi, joka toimii tietovaraston tapaisesti pohjana tietokannalle sekä SQL-kyselytyökaluna.



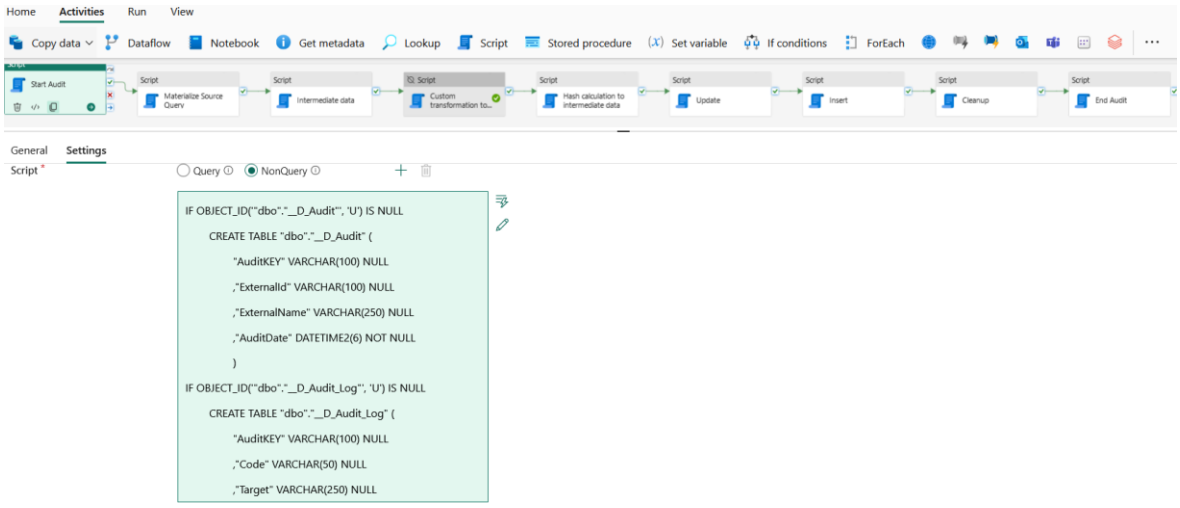
Kuva 34. Fabric warehouse

Hakemiston ”Integration” alle luodaan taas dataputket. Latauksen tyypin mukaan jaettujen hakemistojen lisäksi luodaan hakemisto ”Workflows” työkululle (kuva 35).



Kuva 35. Fabric hakemistorakenne

Dataputkien määrittelyssä käytetään ”Script”-elementtejä ja SQL-lausekkeita. Tuodaan siis vastaavat elementit ja liitännäiset SQL-lausekkeet muunnettuna Fabriciin yhteensopiviksi (kuva 36).



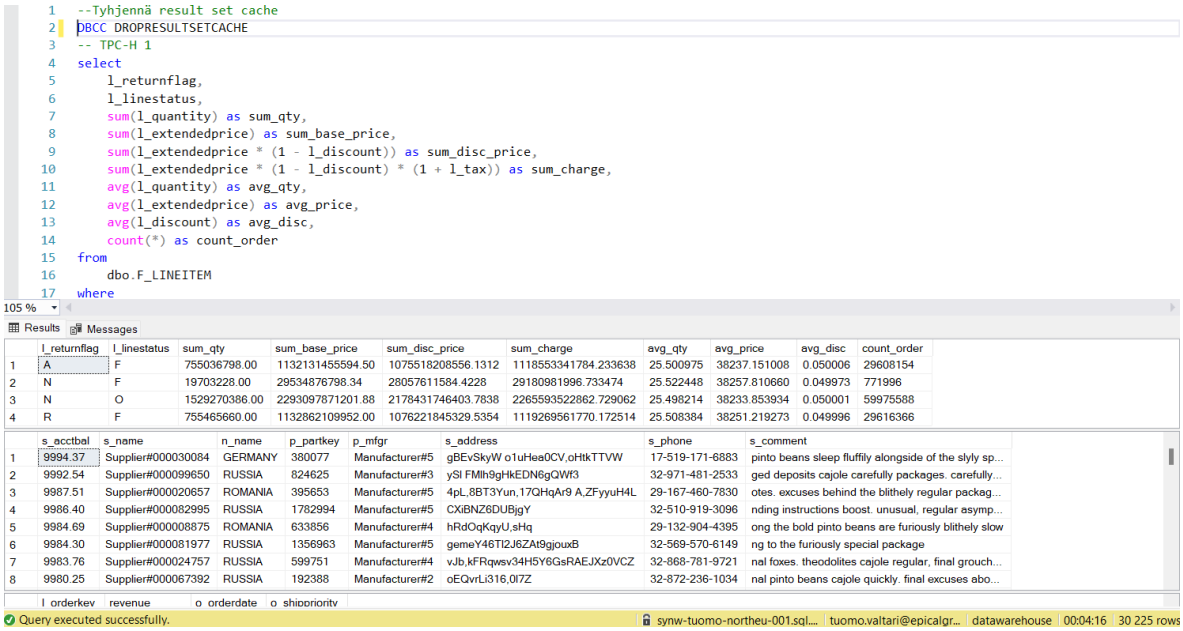
Kuva 36. Fabric data pipeline luonti

Työnkulkua varten luodaan oma dataputki, jossa käytetään ”Invoke Pipeline” -tyypin elementtejä. Yhdistetään seuraavat ”Invoke Pipeline” -elementit, joilla saadaan riippuvuudet määriteltyä toisiin ”Invoke Pipeline” -elementteihin. Työnkulkua voidaan ajaa suoraan työtilasta ”Workflow” kautta käynnistämällä, se voidaan ajastaa tai liittää se johonkin toiseen tehtävään tietovarastoinnin automatisoimiseksi.

5.9 TPC-H-tietokantakyselyt

TPC-H suorituskykytestauksessa käytetään 22 kpl:tta ad hoc SQL-kyselyjä, jossa hyödynnetään laajasti erityyppisiä SQL-operaatioita, kuten esimerkiksi Join:t, Subquery:t, Group by, Order by ja Create. Tarkoitusta varten muutetaan SQL-kyselyt vastaamaan palvelukohtaisesti tuettuja operaatioita.

Kyselytyökalu ja -moottori vaihtelee palveluittain. Esimerkiksi Synapsen ja Fabricin tapauksissa voidaan käyttää kyselyiden suorittamiseen Microsoft SQL Server Management Studiota (kuva 37). Muiden tapauksessa hyödynnetään suoraan palveluiden omia kyselytyökaluja, sillä asetuksen vaativat erillistä määrittelyä ja ajureita. Mikäli palvelut keräävät kyselyiden tuloksia välimuistiin, otetaan tämä ominaisuus pois käytöstä. Microsoft Fabricissa välimuistin käyttö on kiinteänä osana moottoria, eikä asetusta voi muuttaa erikseen käyttäjän toimesta.



```
--Tyhjennä result set cache
DBCC DROPRESULTSETCACHE
-- TPC-H 1
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    dbo.F_LINEITEM
where
```

105 %

Results | Messages

	l_returnflag	l_linestatus	sum_qty	sum_base_price	sum_disc_price	sum_charge	avg_qty	avg_price	avg_disc	count_order
1	A	F	755036798.00	1132131455594.50	1075518208556.1312	1118553341784.233638	25.500975	38237.151008	0.050006	29608154
2	N	F	19703228.00	29534876798.34	28057611584.4228	29180981996.733474	25.522448	38257.810660	0.049973	771996
3	N	O	1529270386.00	22930978712O1.88	2178431746403.7838	2265593522862.729062	25.498214	38233.853934	0.050001	59975588
4	R	F	755465660.00	1132862109952.00	1076221845329.5354	1119269561770.172514	25.508384	38251.219273	0.049996	29616366

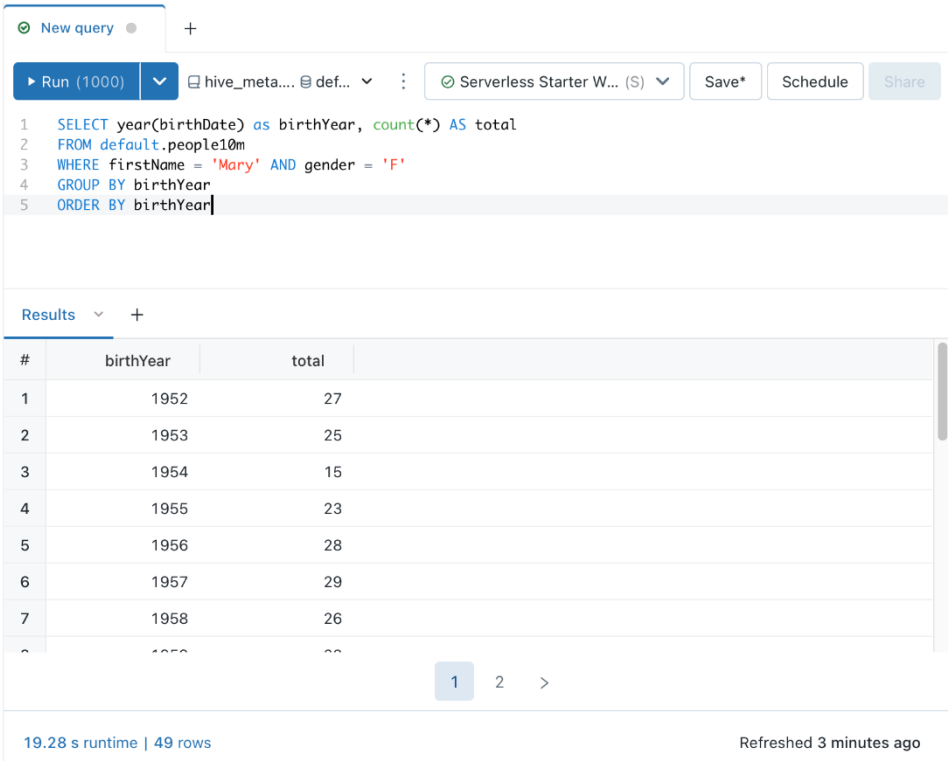
	s_acctbal	s_name	n_name	p_partkey	p_mfgr	s_address	s_phone	s_comment
1	9994.37	Supplier#000030084	GERMANY	380077	Manufacturer#5	gBEvSkyW o1uHea0CV,oHtkTTVW	17-519-171-6883	pinto beans sleep fluffily alongside of the slyly sp...
2	9992.54	Supplier#000099650	RUSSIA	824625	Manufacturer#3	ySl FMlh9gHkEDN6gQWf3	32-971-481-2533	ged deposits cajole carefully packages. carefully...
3	9987.51	Supplier#000020657	ROMANIA	395653	Manufacturer#5	4pL,8BT3Yun,17QHqAr9 A,ZFyyuH4L	29-167-460-7830	otes. excuses behind the blithely regular packag...
4	9986.40	Supplier#000082995	RUSSIA	1782994	Manufacturer#5	CXiBNZ6DUBjgY	32-510-919-3096	nding instructions boost. unusual, regular asymp...
5	9984.69	Supplier#000008875	ROMANIA	633856	Manufacturer#4	hRdOqKqyU,sHq	29-132-904-4395	ong the bold pinto beans are furiously blithely slow
6	9984.30	Supplier#000081977	RUSSIA	1356963	Manufacturer#5	gemeY46TI2J6ZAt9gjouxB	32-569-570-6149	ng to the furiously special package
7	9983.76	Supplier#000024757	RUSSIA	599751	Manufacturer#4	vJb,kFRqwsv34H5Y6GsRAEJXz0VCZ	32-868-781-9721	nal foxes. theodolites cajole regular, final grouch...
8	9980.25	Supplier#000067392	RUSSIA	192388	Manufacturer#2	oEQvrLi316,0I7Z	32-872-236-1034	nal pinto beans cajole quickly. final excuses abo...

	l_orderkey	revenue	o_orderdate	o_shippriority

Query executed successfully. | synw-tuomo-northeu-001.sql... | tuomo.valtari@epicalgr... | datawarehouse | 00:04:16 | 30 225 rows

Kuva 37. TPC-H-tietokantakysely

Databricks'ssä kannattaa SQL editorissa SQL-kyselyjä ajaessa huomioida, että kaikille kyselyille on oletuksena LIMIT 1000, joka rajaa tulosten määrän 1000 riviin (kuva 38). Tämän asetuksen voi muuttaa valitsemalla ylälaidasta kyseisen luvun kohdalta valikon ja poistamalla rajoitteen.



New query +

Run (1000) | hive_meta.... def... | Serverless Starter W... (S) | Save* | Schedule | Share

```
SELECT year(birthDate) as birthYear, count(*) AS total
FROM default.people10m
WHERE firstName = 'Mary' AND gender = 'F'
GROUP BY birthYear
ORDER BY birthYear
```

Results +

#	birthYear	total
1	1952	27
2	1953	25
3	1954	15
4	1955	23
5	1956	28
6	1957	29
7	1958	26

1 2 >

19.28 s runtime | 49 rows

Refreshed 3 minutes ago

Kuva 38. Databricks SQL editor

6 Mittaustulokset ja vertailu

6.1 Mittaukset

Suorituskykymittauksissa mitataan suoritusaikoja palveluissa seuraavissa tietovaraston omaisissa työkuormissa:

- Datan vienti stage-alueelta kohdetauluihin.
- UPDATE ja DELETE-INSERT-operaatiot operaatiot kohdetaulujen mukaan.
- 22 kpl operatiivisen järjestelmän ad hoc SQL-kyselyn suorittaminen kantaa vastaan.

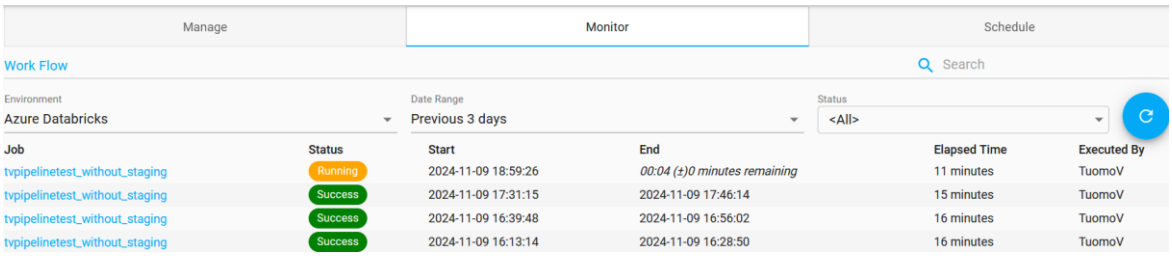
Mittaukset iteroitiin palvelukohtaisesti kolme kertaa per valittu palvelun taso ja per operaatio. Arvoista merkittiin ylös pienin, suurin sekä suoritusten keskiarvo. Tällä valinnalla pyrittiin tasoittamaan vaihteluja, joita erityisesti jaettujen resurssien osalta voi olla esimerkiksi tietyllä ajan hetkellä. Mittaukset dokumentoitiin taulukkoon tietokantapalveluittain resurssin palvelutason mukaisesti sekä itse tulokset edellä mainittujen mittausten suureiden mukaan.

Mittauksista saadut suureet ovat tehtyjen ajojen tai kyselyjen kesto minuuteissa tai sekunneissa. Minuuttien osalta pyöristettiin tulos lähimmän minuutin tasolle, sillä ajan ilmoitustavat poikkeavat toisistaan palvelukohtaisesti, joten sekuntitason vertailu ei ole mielekästä. Tietovarastojen säännölliset eräajot tuotantoympäristöissä ajoittuvat usein hiljaiseen päivän aikaan, eikä tällöin tarvetta sen käytölle synny. TPC-H SQL-kyselyissä merkitään tulokset sekuntitasolla, sillä liiketoiminnan omaisia kyselyjä ajetaan luultavasti tiettyä tarkoitusta varten, tiettyjen henkilöiden toimesta, ja kyselyt voi myös esimerkiksi olla osana rakennetun raportin live-kyselymoottoria, jolloin odotusaika merkitsee enemmän.

6.2 Mittauksien seuranta

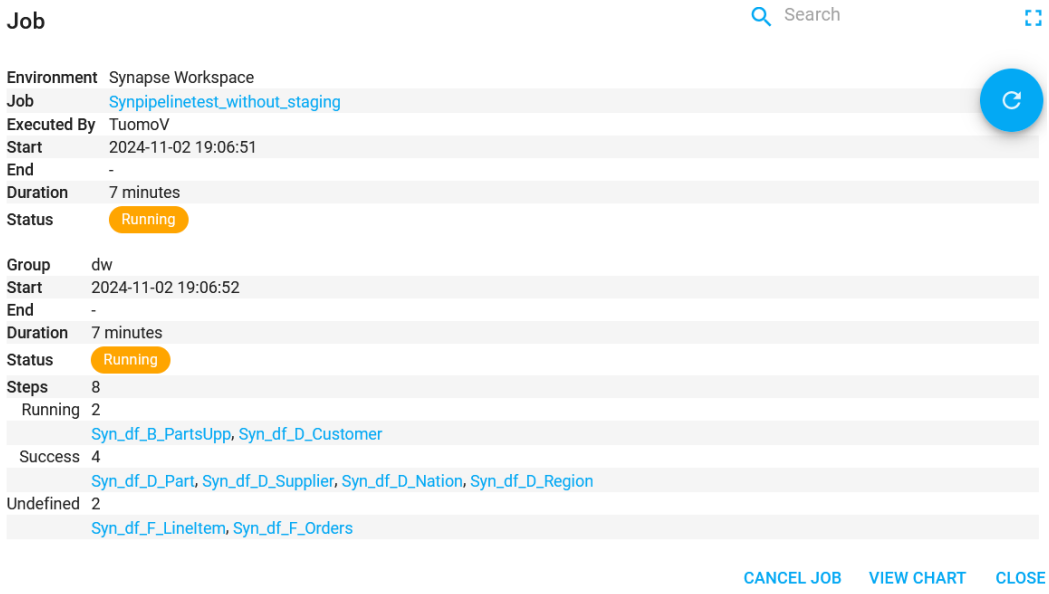
Mittauksien seurantaan käytettiin sekä ADA:n sisäisiä monitorointityökaluja, että muita palvelujen sisäisiä työkaluja, mikäli tukea ADA-seurannalle ei ole. Kyselyjen seurantaan käytetään joko SQL Server Management Studio -työkalua tai palvelujen sisäisiä monitorointityökaluja, mikäli Management Studio ei ollut helposti hyödynnettävissä tietokantapalvelun kanssa.

ADA:ssa käynnistettyjä työnkulkuja voidaan seurata Monitor-välilehden kautta. Tietoja voidaan suodattaa ympäristön, viimeisimmän suoritusajankohdan tai suorituksen statuksen mukaan ja järjestää halutusti (kuva 39).



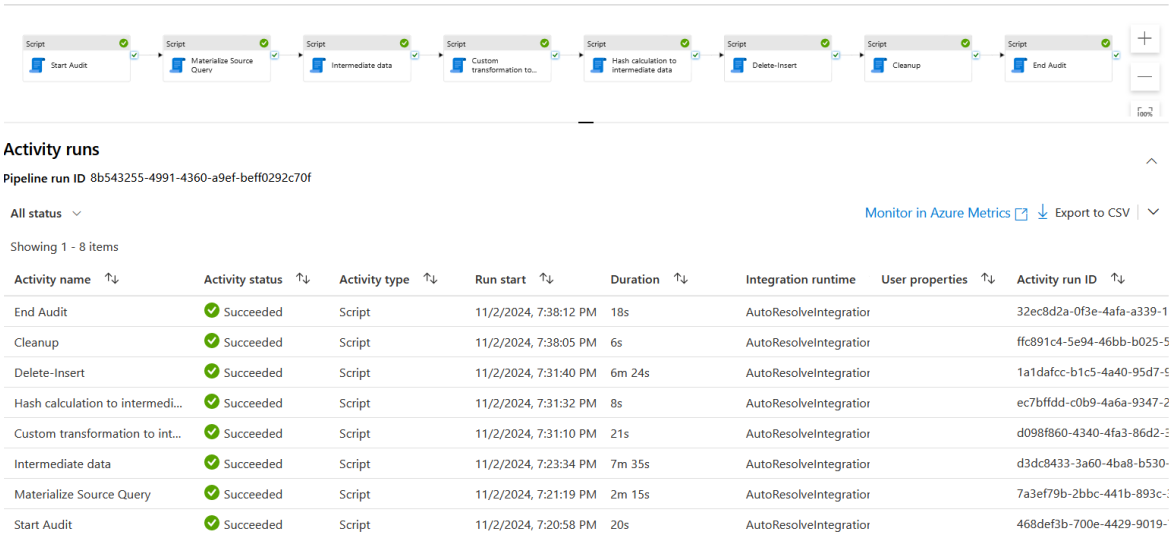
Kuva 39. ADA monitor

Avaamalla ajon päästään käsiksi tarkempiin suorituksen tietoihin ja voidaan tarkastella missä vaiheessa suoritus on menossa. Järjestelmä antaa myös ennusteen päättymisajankohdasta pohjautuen muun muassa aiempien jobien suorituksiin (kuva 40).



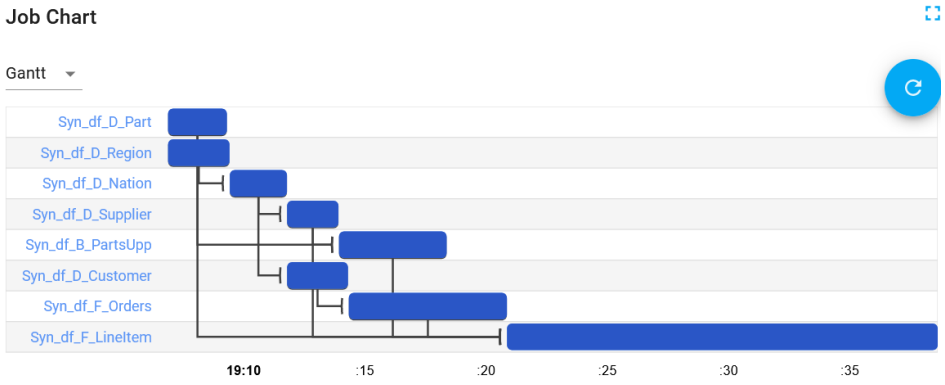
Kuva 40. ADA ajoketjun vaihe

Suorituksessa olevan ajon “Job” -linkin kautta pääsee Synapseen, josta voidaan nähdä kyseisen suorituksen ajoketjun etenemisvaihetta ja tilaa. Mikäli ajo epäonnistuu, voidaan tarkemmin vaiheen kautta päästä käsiksi myös virheen aiheuttaneen elementin SQL-koodiin tai järjestelmän antamaan virheviestiin.



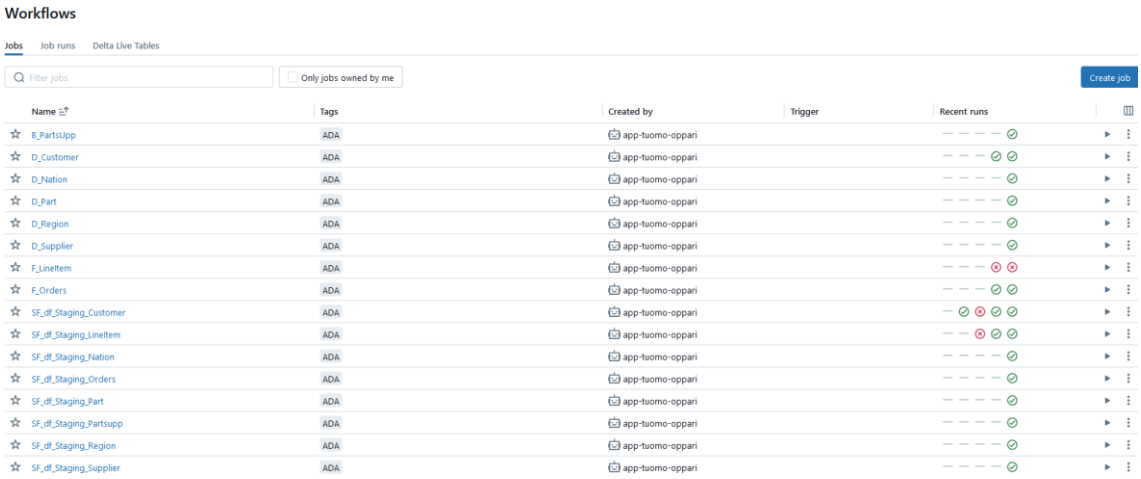
Kuva 41. Synapsen dataputken ajo

Yksittäisten ajojen suoriutumista, riippuvuuksia, ajojen rinnakkaisuutta ja suoriutumissaikoja voidaan tarkastella visuaalisesti Gantt-kaavion kautta valitsemalla “View Chart” Job-ikkunassa (kuva 42).



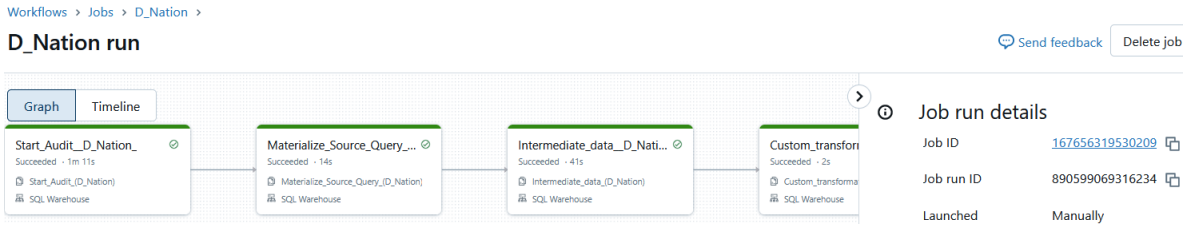
Kuva 42. ADA Gantt-kuvaaja

Monitorointi ADA:ssa hoituu tällä tapaa sekä Synapsen dedikoidun tietokannan ja Snowflaken käyttötapauksissa. Databricks:ssä ADA:aa hyödynnetään samaan tapaan työnkulkujen luontiin, suoritukseen ja ajojen monitorointiin. Mikäli Databricks:ssä halutaan tarkempaa tietoa ajoista, voidaan hyödyntää palvelun ”Workflows” valikon alta löytyvää ”Jobs” välilehteä. ”Jobs” näyttää kaikki palveluun luodut työnkulut sekä tiedot viimeisistä suorituksista sekä niiden yksittäiset ajot onnistuvat täältä (kuva 43).



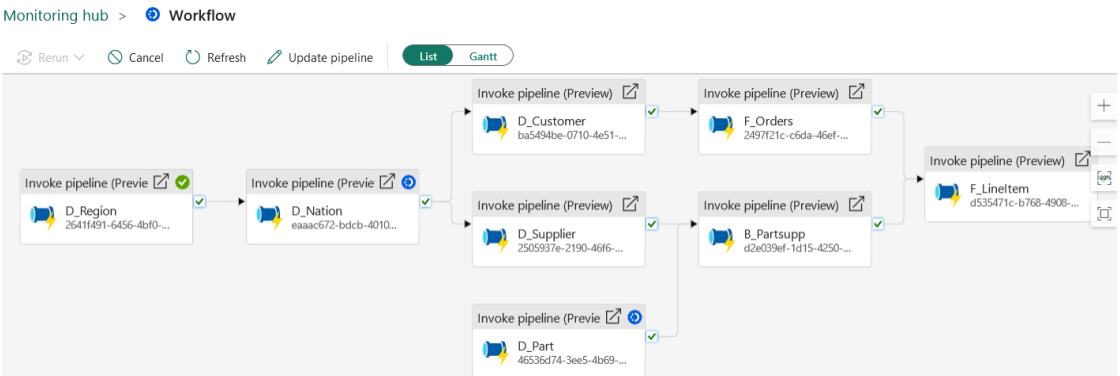
Kuva 43. Databricks Jobs -välilehti

Suorituksen vaihetta voidaan katsoa samaan tapaisesti kuin Synapsen Monitor-välilehden alta, josta nähdään mistä elementeistä ajo koostuu ja mitkä niiden suoriutumisajat sekä myös tarkempaa tietoa, mikäli ajot ovat epäonnistuneet. Valitsemalla yksittäisen ajon voidaan tarkastella sen vaihetta tai valmistuneen kulun eri vaiheita (kuva 44).



Kuva 44. Databricks ajon sisältö

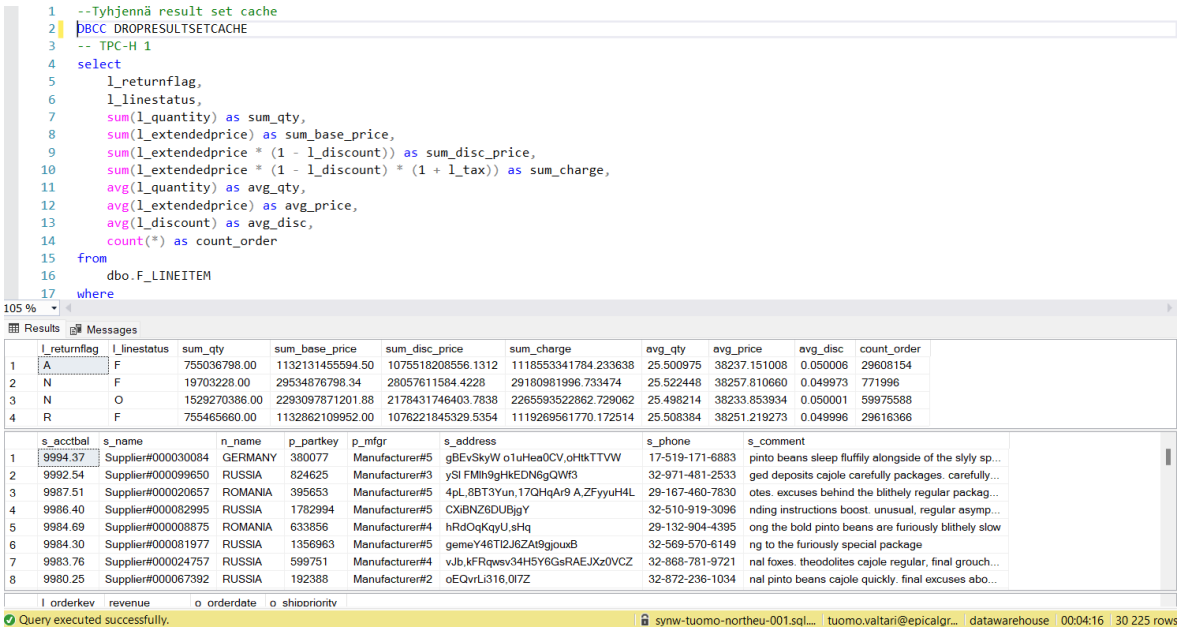
Microsoft Fabricissa työnkulun etenemistä voidaan seurata Monitoring Hub -välilehden kautta, josta nähdään suorituksen tila ja kyseisellä hetkellä käynnissä olevat dataputket (kuva 45). Tarkemmin yksittäisten ajojen sisään pääsee pureutumaan Workflow-alta valitsemalla haluttu elementti. Itse suoritusaika pitää kuitenkin laskea aloitus ja päättymisajan perusteella tai katsomalla koko ajon etenemistä Gantt-kuvaajasta.



Kuva 45. Fabric työnkulun monitorointi

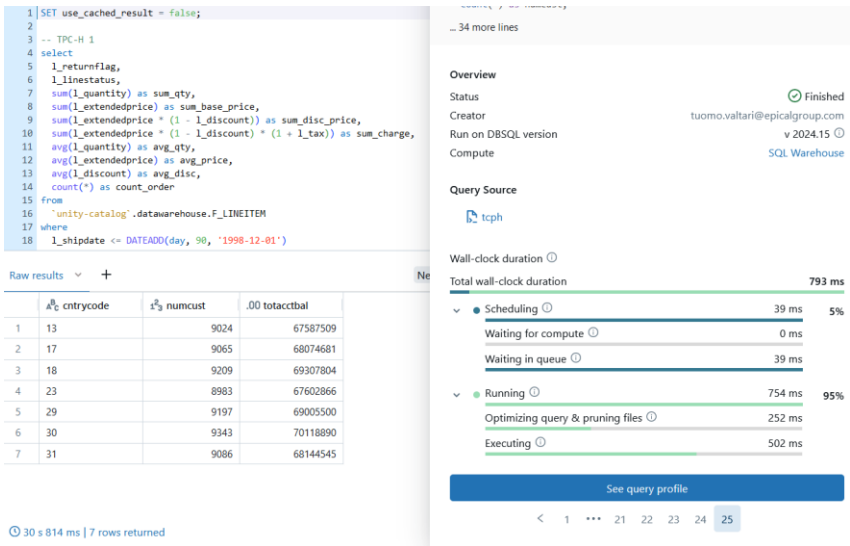
.

Kyselyjen seurantaan käytetään palvelujen omien monitorointityökalujen lukemia. Microsoft SQL Server Management studion tapauksessa kyselyt antavat sekä kyselyn tuloksen että näihin kuluneen ajan samaan ikkunaan (kuva 46).



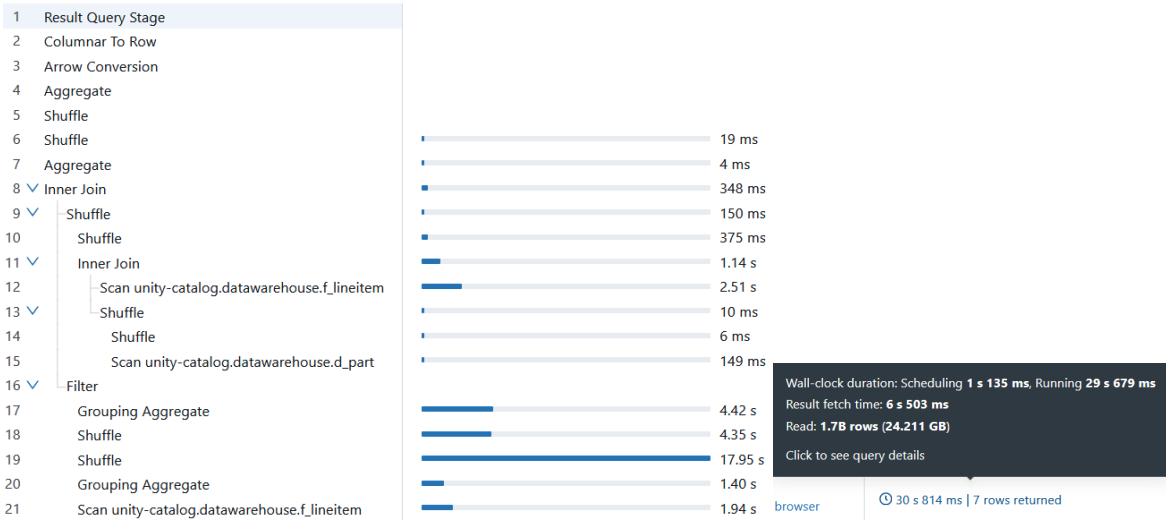
Kuva 46. SQL Server Management Studio (SSMS)

Databricks'ssä kyselyitä voidaan luoda ja tallentaa "Queries" sivun alle. "Query history" näyttää taas aiemmin suoritettujen kyselyjen historiaa ja tietoja. Sivulta pääsee myös tarkemmin käsiksi muun muassa siinä luettujen rivien ja datan määrään. Kyselytyökalu ei kuitenkaan palauta useamman kyselyn tuloksia, vaan näyttää vain viimeisimmän kyselyn tuloksen poiketen näin toimintalogiikaltaan SSMS:stä (kuva 47).



Kuva 47. Databricks kyselyn tulokset

Työkalu näyttää myös kyselyjen etenemistä tarkemmalla operaatiotasolla, josta voidaan pureutua yksittäisten operaatioiden sisälle tarkemmin ja nähdä suoritusaikoja. Kaikki riveistä eivät kuitenkaan ole SQL-operaatioita, vaan esimerkiksi ”Shuffle” on yksi Apache Sparkin hitaimmista operaatioista. Käytännössä se siirtää vain dataa klusterin suoritusnodejen välillä (kuva 48).



Kuva 48. SQL-kyselyn suoriutuminen ja ajat

6.3 Mittaustulokset

Mittaustulokset kerättiin pääosin huhti-toukokuun 2024 aikana. Optimointia tehtiin kuitenkin tauluihin vielä lokakuussa 2024, jolloin tulokset hieman poikkesivat aiemmista. Esimerkiksi Fabriciin oli testien välissä tullut päivitys, jonka takia vanhat invoke pipelinet putosivat pois käytöstä ja niitä varten piti luoda kokonaan uudet päivitetyt invoke pipelinet.

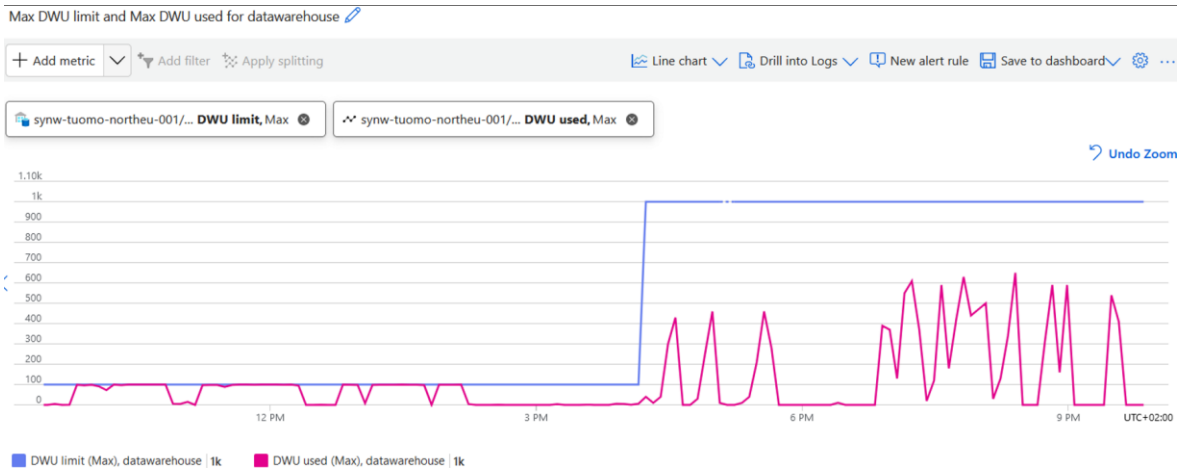
6.3.1 Datan vienti kohdetauluihin

Datan viennissä stage-alueelta tyhjiin kohdetauluihin tulosten keskiarvojen hitaimman ja nopeimman välillä oli enintään 6 min eli 35 % ero (taulukko 1).

DB palvelu	Resurssin taso	Ka. 3 ajoa (min)	Min. (min)	Max. (min)
Synapse Dedicated Pool	DW100	23	22	25
Synapse Dedicated Pool	DW400	17	16	19
Databricks SQL WH	2X-Small (serverless)	17	17	17
Databricks SQL WH	Medium (serverless)	17	17	17
Snowflake Warehouse	X-Small	21	20	22
Snowflake Warehouse	Medium	19	18	22
Microsoft Fabric WH	F2	19	18	21
Microsoft Fabric WH	F64	19	18	20

Taulukko 1. Datan kohdetauluihin ajot ja suoritusajat

Mittauksissa hitain Synapse Analytics dedikoitu allas selkeästi kärsi resurssin täyttäessä DWU-limiitin DWU100-tasolla, mutta kun resursseja lisäsi DWU400-tasolle, ei resurssin teho enää rajoittanut ja tulos oli parhaimmistoa kirjoitusoperaatioiden osalta (kuva 49).



Kuva 49. Synapse SWL dedicated pool monitor

Databricks suoriutui selkeästi tasaisimmin, eikä tulos poikennut 2X-Small ja Medium koon SQL warehouse klusterien välillä. Hyvään tulokseen on varmasti vaikuttanut Databricksin hyödyntämä Delta Lake -formaatti, joka on optimoitu datan tuomiseen. Tuloksiin on luultavasti vaikuttanut myös Truncate-komennon käyttö taulujen puhdistuksessa. Truncate tyhjentää vain loogisen kerroksen, mutta ei alla olevia Delta Laken Parquet-tiedostoja. Tähän tarkoitukseen tulee tauluun käyttää Vacuum-komentoa, mikä tyhjentää myös fyysisesti tiedostot Delta Lakestä. Näin ollen vaikuttaa siis, että 2X-Small-klusteri on hyötynyt operaatioissa fyysisen datan säilymisestä, mutta tulos on silti hyvää tasoa M-resurssilla. Microsoft Fabricissa tulokset F2 ja F64 resurssien välillä ovat hyvin tasaväkiset. Tätä voi selittää Fabricin hyödyntämät Smoothing ja Bursting teknologiat, jossa viimeiseksi mainittu tarkoittaa, että Fabric voi lisätä hetkellisesti enemmän resursseja työkuorman käyttöön, jota se sitten tasoittaa Smoothing-teknologian avulla ajan kanssa. Fabriciin luotiin työnkulut käsin, mikä

voisi selittää erilaisia tuloksia F64 kapasiteetin osalta verrattuna muihin ratkaisuihin. Snowflake:ssä erot ovat suhteessa melko pieniä ja selittynevät laskentakapasiteetin täyttymisellä Synapse Analytics dedikoidun altaan tapaan.

6.3.2 Muutokset stage-latauksiin

Toisena testattiin muutoksia taulujen staging-alueelle ennen datan tuontia kohdetauluihin olemassa olevan datan päälle (taulukko 2).

DB palvelu	Resurssin taso	Ka. 3 ajoa (min)	Min. (min)	Max. (min)
Synapse Dedicated Pool	DWU 100	30	26	38
Synapse Dedicated Pool	DWU 400	20	18	22
Databricks SQL WH	2X-Small (serverless)	15	14	15
Databricks SQL WH	Medium (serverless)	15	14	15
Snowflake Warehouse	X-Small	21	20	22
Snowflake Warehouse	Medium	19	18	22
Microsoft Fabric WH	F2	25	25	26
Microsoft Fabric WH	F64	20	16	26

Taulukko 2. Muutosajojen suorituskykymittaukset

Muutosoperaatioissa alkoi selkeästi erottua Synapse- ja Fabric-tietokatojen resurssit suoritusaikojen kasvaessa suurempien resurssien suoriutuessa kuitenkin vielä kohtalaisesti. Databricks hyödyntää mallikkaasti Delta Lake-puolta. Snowflake suoriutui tästäkin testistä tasaisesti molempien resurssien osalta.

6.3.3 TPC-H-kyselytestit

TPCH-suorituskykytestit toivat jo isommin eroja esiin (taulukko 3). Microsoft Fabric oli ainoa, josta vertailukelpoista tulosta ei saanut, sillä järjestelmästä ei saanut poistettua käytöstä välimuistia. Erot hitaimman ja nopeimman järjestelmän välillä ovat n. 17 minuuttia – eroa yli 7500 %, mikä alkaa olla minkä vain analyyttisen järjestelmän rampauttava aika. Voidaan kuitenkin sanoa, että kaikki palvelut läpäisivät testin onnistuneesti.

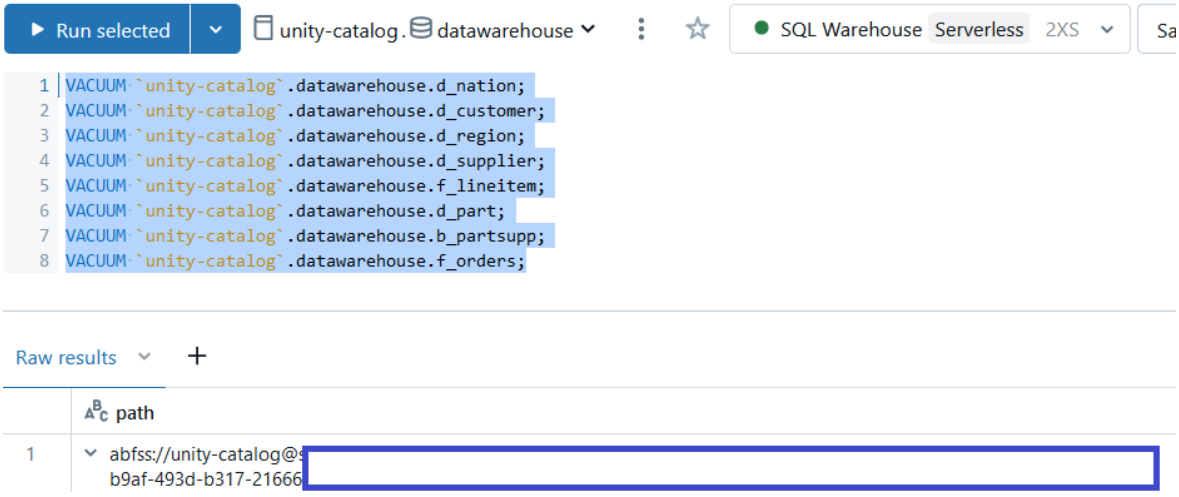
DB palvelu	Resurssin taso	Ka. 3 ajoa (s)	Min. (s)	Max. (s)
Synapse DP	DWU 100	1090	990	1230
Synapse DP	DWU 400	250	210	330
Databricks SQL WH	2X-Small (serverless)	64	52	66
Databricks SQL WH	Medium (serverless)	33	29	40
Snowflake WH	X-Small	24	18	35
Snowflake WH	Medium	15	14	16
Microsoft Fabric WH	F2	Autocache	20	120
Microsoft Fabric WH	F64	Autocache	15	21

Taulukko 3. TPC-H-kyselyjen suoritusajat

Tuloksissa tulee huomioida, ettei tauluja tietokannassa ole optimoitu millään lailla esimerkiksi indeksointien, avainnoksien tai taulujen partitioinnin osalta, joten tavallista tähtimallia ei muodostu. Kuitenkin Snowflake suoriutuu varsin ripeästi kyselyistä molempien resurssien osalta. Databricks jälleen isommalla kapasiteetilla toimii vielä kohtalaisesti pienemmän kapasiteetin kyselyaikojen lähes tuplaantuessa.

6.4 Optimoinnilla parannusta suorituskykyyn?

Kokeillaan vielä hieman optimoida taulujen hajautus- ja indeksointiominaisuuksia Synapsen puolella, ja lisätään samalla tehoja, jotta nähdään kuinka paljon optimoinnilla, voidaan vaikuttaa palvelun käyttöön. Kokeillaan myös Fabricista vielä F32 tason resurssia hakien lisää näkemystä resurssin SKU:n vaikutuksesta. Tehdään samalla Databricksille testi deltalake-liitännäisten tietojen poistosta, mikäli tällä on vaikutusta pienemmistä resurssiin (kuva 50).



Kuva 50. VACUUM komento Parquet-tiedostoille

Dedikoituun Synapsen kantaan luotiin taulut uusiksi lisäämällä tauluun tauluille pääavaimet. Hajautetun taulun metodit määriteltiin taulukohtaisesti, jossa faktoille ja isoimmille dimensioille asetettiin Hash-tyyppinen hajautus, jolla pyrittiin erityisesti optimoimaan Join-

operaatioita taulujen välillä (kuva 51). Taulujen Join-operaatioiden yhteydessä käytetyt vierasavaimet määritettiin ei-klusteroiduiksi indekseiksi. Pienemmät dimensiot luotiin taas Heap-tauluna Replicate-tyypin hajautuksella. Valituilla taulun hajautuksen tavoilla sekä indeksoinneilla tasapainoiltiin tauluun kirjoituksen suorituskyvyn ja kyselyiden nopeuden kanssa. (Microsoft 2022.)

```
CREATE TABLE [dbo].[F_LINEITEM]
(
    [L_ORDERKEY] [bigint] NOT NULL,
    [L_PARTKEY] [bigint] NULL,
    [L_SUPPKEY] [bigint] NULL,
    [L_LINEDECIMAL] [bigint] NOT NULL,
    [L_QUANTITY] [decimal](15, 2) NULL,
    [L_EXTENDEDPRICE] [decimal](15, 2) NULL,
    [L_DISCOUNT] [decimal](15, 2) NULL,
    [L_TAX] [decimal](15, 2) NULL,
    [L_RETURNFLAG] [varchar](1) NULL,
    [L_LINESTATUS] [varchar](1) NULL,
    [L_SHIPDATE] [date] NULL,
    [L_COMMITDATE] [date] NULL,
    [L_RECEIPTDATE] [date] NULL,
    [L_SHIPINSTRUCT] [varchar](25) NULL,
    [L_SHIPMODE] [varchar](10) NULL,
    [L_COMMENT] [varchar](44) NULL,
    [L_DUMMY] [varchar](10) NULL,
    [INSERT_AUDITKEY] [bigint] NULL,
    [UPDATE_AUDITKEY] [bigint] NULL,
    UNIQUE (L_LINEDECIMAL) NOT ENFORCED
)
WITH
(
    DISTRIBUTION = HASH(L_LINEDECIMAL),
    CLUSTERED COLUMNSTORE INDEX
);

CREATE NONCLUSTERED INDEX idx_lineitem_orderkey ON [dbo].[F_LINEITEM] (L_ORDERKEY);
CREATE NONCLUSTERED INDEX idx_lineitem_partkey ON [dbo].[F_LINEITEM] (L_PARTKEY);
CREATE NONCLUSTERED INDEX idx_lineitem_suppkey ON [dbo].[F_LINEITEM] (L_SUPPKEY);
CREATE NONCLUSTERED INDEX idx_lineitem_shipdate ON [dbo].[F_LINEITEM] (L_SHIPDATE);
CREATE NONCLUSTERED INDEX idx_lineitem_commitdate ON [dbo].[F_LINEITEM] (L_COMMITDATE);
CREATE NONCLUSTERED INDEX idx_lineitem_receiptdate ON [dbo].[F_LINEITEM] (L_RECEIPTDATE);
```

Kuva 51. Faktataulun optimointia

Palveluihin ja tauluihin tehdyillä muutoksilla havaittiin olleen vaikutusta suoritusaikoihin sekä viedessä tyhjiin kohdetauluihin että datan muutostapauksissa. Synapsen dedikoitu DWU 1000 tason optimoitu tietokanta menee jopa kirjoitustapahtumissa Databricks:n ohi. Delta Lake tiedostojen puhdistuksen jälkeen myös Databricks SQL Warehousen 2X-Small-resurssi hidastuu ja jää jälkeen Medium-tason resurssista. Microsoft Fabric F32 kapasiteetti on jopa nopeampi kirjoitusoperaatiossa verrattuna F64-kapasiteettin – johtopäätöksenä aika lienee korjannut tai parantanut palvelun toimintaa. Järjestelmän työketju on myös

rakennettu uusiksi, sillä aiemmat työketjut ovat korvautuneet uusilla ajojen välissä (taulukko 4).

DB palvelu	Resurssin taso	Kohdetaulut tyhjiä			Datan muutos staging:llä		
		Ka. 3 ajoa (min)	Min. (min)	Max. (min)	Ka. 3 ajoa (min)	Min. (min)	Max. (min)
Synapse DP	DW100	23	22	25	30	26	38
Synapse DP	DW400	17	16	19	20	18	22
Synapse DP	DW1000 (opt)	15	14	16	17	16	17
Databricks SQL WH	2X-Small (serverless)	19	18	19	19	19	19
Databricks SQL WH	Medium (serverless)	17	17	17	15	14	15
Snowflake	X-Small	21	20	22	21	20	22
Snowflake	Medium	19	18	22	19	18	22
Microsoft Fabric WH	F2	19	18	21	25	25	26
Microsoft Fabric WH	F32	17	17	18	22	22	23
Microsoft Fabric WH	F64	19	18	20	20	16	26

Taulukko 4. Optimoitujen tai säädettyjen resurssien suorituskyky tauluajoissa

Synapsen optimoidun tietokannan osalta katsottiin tuloksia vielä tietokantakyselyjen osalta. Taulukossa 5 nähdään taulujen optimoinnin jälkeen kyselyn suoritusaikojen putoavan lähes neljäsosaan alkuperäisiin verrattuna. Optimoimalla lisää tietomallia ja tauluja esimerkiksi indeksien osalta, tuloksia voisi luultavasti entisestään parantaa. Vastakohtana keskittymällä erityisesti kyselyjen optimointiin tai indeksien laajempaan optimointiin voi tuloksena olla datan tuontiprosessin hidastuminen.

Tietovarastoresurssi	Resurssin taso	Ka. 3 ajoa (s)	Min. (s)	Max. (s)
Synapse DP	DW100	1090	990	1230
Synapse DP	DW400	250	210	330
Synapse DP	DW1000 (opt)	82	82	83
Unity Catalog	2X-Small (serverless)	122	117	132

Taulukko 5. Kyselyjen suorituskyky muutoksien jälkeen

6.5 Hinnoittelu

Palvelut osiossa tarkasteltiin palvelukohtaisesti hintaan vaikuttavia tekijöitä. Monesti hinta on useamman osatekijän summa: laskentaresurssit, data, pilvipalvelun palvelut ja palveluun valittu taso. Datan varastoinnin lisäksi sen hakemisesta ja viemisestä saatetaan veloittaa erikseen datamäärien perusteella.

Joskus suoranta hintavertailu kahden tuotteen välillä voi olla varsin hankalaa, sillä kaikki vaikuttavat tekijät eivät välttämättä näy palvelua testiluontoisesti kokeilemalla tai käyttämällä vain tiettyyn käyttötapaukseen. Kaikkien palveluiden osalta verkosta löytyy kuitenkin hintalaskurit, joiden avulla vertailua voi harjoittaa kattavasti. Myös pilvipalveluiden sisältä löytyy kulujen tarkkailuun toimintoja, joilla jo kulutettujen resurssien lisäksi voidaan analysoida ja ennustaa tulevaa.

Taulukossa 6 eri palveluiden tietovarastoresurssien pay-as-you-go -tyylistä hinnoittelua tarkasteltiin vielä tämän opinnäytetyön kirjoituksen hetkellä, joka on kuitenkin vain yksi osa kokonaisuudesta. Databricksin osalta Serverless-kapasiteetin hinta sisältää myös pilvi-instanssin maksun. Osassa palveluista ilmoitettu pelkästään dollarihinnat, josta euromäärät on laskettu valuuttamuunnoksella, jossa 1 € vastaa 1,0488 dollaria. Palveluiden tai alustan aluetiedoksi on valittu Pohjois-Eurooppa.

DB palvelu	Resurssi	€ Per h	€ per pv	€ per kk (22 pv)	HUOM!
Microsoft Fabric	F2	0,35	8,45	185,86	
Synapse Dedicated Pool	DW100	1,29	30,91	680,06	
Databricks SQL WH	2X-Small (Serverless)	3,47	83,28	1832,16	Enterprise taso
Snowflake WH	X-Small	3,72	89,28	1964,16	Premium Plan
Synapse Dedicated Pool	DW400	5,15	123,58	2718,67	
Microsoft Fabric WH	F32	5,63	135,14	2973,17	
Microsoft Fabric WH	F64	11,26	270,26	5945,81	
Synapse Dedicated Pool	DW1000	12,87	308,95	6796,94	
Databricks SQL WH	Medium (Serverless)	13,88	333,12	7328,64	Premium Plan
Snowflake WH	Medium	14,36	344,64	7582,08	Enterprise taso

Taulukko 6. Eri resurssien hinnat palveluittain

Taulukon hinnat laskettiin sillä ajatuksella, että resurssit olisivat aina työpäivinä päällä. Tavanomaisempi tietovarastojen käyttötapa voisi olla joko pysäyttää resurssit businessaikojen ulkopuolella tai skaalata niitä vaihtuvien kuormien mukaan. Osa palveluista mahdollistaa myös ennalta varattua kapasiteettiä tietyksi ajaksi, jolloin saavutetaan säästöjä normaaliin, käyttöön perustuvaan hinnoitteluun verrattuna. Tällöin esimerkiksi isommat eräajot tai kyselyt voitaisiin suorittaa isomman resurssin alla ja kevyempiä tehtäviä hoidettaisiin kevyemmillä resursseilla, jotka nekin suoriutuvat tavanomaisista tehtävistä varsin kohtalaisesti.

6.6 Palveluiden käyttöönotto ja käyttö

Pilvipohjaisten pay-as-you-go -tyyppisten palvelujen käyttöönotto sujuu usein muutamalla hiiren painalluksella ja maksutiedot syöttämällä. Palveluista esimerkiksi Fabric tarjoaa muutaman kuukauden triallisenssillä palvelun kokeilua ilmaiseksi. Usein haasteet syntyvät enemmän ulkoa tulevista tietoturva ja käsittelyn vaateista – usein tietysti tietoturvan näkökulmasta hyvästä syystä. On mahdollista, että tiettyjä toimintoja ei saa itse päälle tai oikeudet eivät riitä, jolloin asioiden käyttöönotto saattaa viivästyä.

Nykyaikaiset pilvipohjaiset palvelut ovat toimintojensa lisäksi käyttöliittymältään varsin monipuolisia ja intuitiivisia – useimmat ydintoiminnot hoituvat graafisen käyttöliittymän kautta, mutta myös esimerkiksi API-rajapintoja hyödyntämällä tai komentorivipohjalta, ja ne tarjoavat laaja-alaisesti työkaluja käyttöön. Toinen palvelu tai teknologia pohjalla voi sopia paremmin esimerkiksi organisaation business-analyytikolle ja tukea tehokkaasti businessorientoituneita työnkulkuja, kun toinen taas datatieteilijän tarpeisiin ja tukea edistyneemmän analytiikan toimintoja esimerkiksi Python-kielen käytön mahdollistamalla käyttöliittymässä. Avoimen formaatin teknologiat mahdollistavat työskentelyn eri työkalujen kautta, joten yhteen tuotteeseen tai valmistajaan ei tarvitse usein enää sitoutua.

7 Yhteenveto ja pohdinta

7.1 Tutkimuskysymysten vastaukset

Työssä käytiin läpi teoreettisesti ensin tiedon käsittelyä, -varastointia, eri käytössä olevia teknologioita sekä tiedon mallintamista ja vie. Tämän jälkeen tutkittiin pilvipohjaisten tietokannan omaisten palveluiden toimintaa ja ydinominaisuuksia. Käytännön osuudessa tehtiin käytännön testausta näissä valituissa palveluissa, pohdittiin ja kokeiltiin optimoinnin merkitystä ja käytiin muun muassa läpi palveluiden välistä hinnoittelua ja käyttöä. Ympäristö saatiin rakennettua lopulta pienien haasteiden kautta melko yhtäläiseksi testausta ajatellen ja testit saatiin suoritettua käyttämällä työnkulkuja automatisoituihin ajoihin.

Ensimmäisenä tutkimuskysymyksenä oli tutkia, miten modernit pilvipohjaiset tietokantapalvelut eroavat ominaisuuksien, kuten tehokkuus ja hinta, puolesta. Tähän saatiin vastaus eri tavoin palvelujen suorituskykyä mitanneilla testeillä, taustoittamalla niiden olennaisia ominaisuuksia sekä varastointi- ja prosessointitapoja. Palvelujen hintoja tarkasteltiin laskentakapasiteetin perusteella sekä läpikäytiin muita vaikuttavia hintatekijöitä.

Toisena tutkimuskysymyksenä oli tutkia mitkä olennaiset tekijät vaikuttavat sopivan tietokantapalvelun valintaan. Kysymyksen vastausta varten taustoitettiin teoriaosuudessa tiedon varastoinnin eri tapoja. Palvelujen ominaisuuksia vertailtiin ja tätä kautta vastattiin mitä toimintoja ne mahdollistavat. Lähtödatan muodon ja jatkohyödyntämisen näkökulmaa tuotiin taas liiketoiminnon eri käyttäjien tarpeiden kartoittamiseksi käyttötapausten kautta.

Kolmantena tutkimuskysymyksenä oli pohtia palveluiden soveltuvuutta toisen tutkimuskysymyksen kohtiin. Kysymyksen vastausta varten esiteltiin muun muassa datalake ja lakehouse-arkkitehtuuria sekä näiden eroja tavanomaiseen strukturoituun tietovarastototeutukseen. Eri palvelujen ominaisuuksia kartoittamalla haettiin eroja palvelujen välille. Suorituskyky- ja hinnoittelun vertailuilla löydettiin eroja palvelujen välillä ja eri käyttötapauksiin soveltuvuutta.

7.2 Pohdinta

Käytännön osuudessa käytetyt palvelut onnistuttiin lopulta rakentamaan ratkaisua varten melko yhtäläisiksi, vaikka esimerkiksi Microsoft Fabriciin työnkulkujen luonti osoittautui alkuun melko työlääksi. Tuotteiden pohjalla olevat teknologiat ja käyttölogiikka erosivat kuitenkin melko paljon, ja esimerkiksi taustalla olevien teknologioiden teorian ja itse palvelujen käytön opettelu vei osan työosuudesta, vaikka muutamat palvelut olivatkin opinnäytetyön tekijälle ennestään jo töiden kautta tutuksi tulleita. Yllätyksiltäkään ei aivan kokonaan

vältytty vaan esimerkiksi testauksen jälkeen vasta ilmeni, että taulujen pohjalla olevan puhdistus ei onnistunut täysin kaikista palveluista ilman erillistä putsausta.

Palvelujen määrää rajaamalla olisi voitu syväluodata tiettyä tekniikkaa tai palvelua tarkemmalla tasolla, ja samalla itse käytännön osuuteen olisi voinut panostaa vielä enemmän käyttöönottojen ja ympäristöjen erinäisten kommervenkkien vähentyessä. Vastakohtana fokus olisi voinut siirtyä ehkä liiaksikin yhteen tai muutamaan tuotteeseen tai teknologiaan, jolloin työn soveltamisalue olisi rajautunut.

7.3 Yhteenveto ja jatkokehitysideat

Käytetyt palvelut ja teknologiat vastaavat nykypäivän tietovarastoinnin haasteisiin joustavalla tavalla ja tullevat näin uskoakseni pysymään markkinassa vielä jatkossa yhtä lailla. Teknologian kehittyessä ripeäasti, etenkin pilvimaailman palveluissa, erilaisia ratkaisuja julkaistaan ratkaisemaan erilaisia haasteita. Palveluista esimerkiksi Microsoft Fabric on vielä tuoreehko lisäys palveluvalikoimaan ja on mielenkiintoista nähdä, miten se tulee kaikki yhdessä palvelussa - ratkaisuna, muuttamaan datan käsittelyn kenttää.

Käytännön osuudessa läpikäytiin pitkälti tietokantoihin liittyviä tavanomaisia työnkulkuja ja hyödyntämistarpeita. Tästä voisi jatkaa enemmän esimerkiksi Data Lakehouse -teknologian toteutuksiin, ja tuoda laajemmin esimerkiksi raakadatan prosessoinnin näkökulmaa, mutta samalla myös datan loppukäyttäjän tarpeita esimerkiksi tekoälystä ja -malleista raporttien kehittämiseen. Medallion -arkkitehtuurin käytön vaikutuksia olisi mielenkiintoista nähdä Lakehouse-toteutuksen pohjana käytännön tasolla ja vertailla tätä muun tyyppisiin toteutuksiin.

Lähteet

Anttila, J. 2023. Databricks ja Lakehouse -arkkitehtuuri. Viitattu 19.10.2024. Saatavissa https://isletgroup.fi/2023/04/25/databricks-ja-lakehouse-arkkitehtuuri/

Atlan. 2023. Data Lake vs Data Swamp: Differences & Cautionary Steps Viitattu 5.4.2024. Saatavissa https://atlan.com/data-lake-vs-data-swamp/

Chu, D. 2024. Understanding the Difference Between Data Orchestration and ETL. Blogi. Viitattu 29.3.2024. Saatavissa https://www.secoda.co/blog/data-orchestration-vs-etl

Databricks 2022. The Good and the Bad of Databricks Lakehouse Platform. Viitattu 15.5.2024. Saatavissa https://www.databricks.com/blog/2022/05/19/day-in-the-life-of-a-customer-success-engineer.html

Databricks 2024a. What is Delta Lake? Viitattu 12.5.2024. Saatavissa https://docs.databricks.com/en/delta/index.html

Databricks 2024b. What is the medallion lakehouse architecture? Viitattu 31.5.2024. Saatavissa https://docs.databricks.com/en/lakehouse/medallion.html

Databricks. 2024c. Introduction to Data Lakes. Viitattu 5.4.2024. Saatavissa https://www.databricks.com/discover/data-lakes

Databricks. 2024d. What is a Data Lakehouse? Viitattu 15.5.2024. Saatavissa https://www.databricks.com/glossary/data-lakehouse

Databricks. 2024e. What is a DataFrame? Viitattu 19.10.2024. Saatavissa https://www.databricks.com/glossary/what-are-dataframes

Databricks. 2024f. What is a medallion architecture? Viitattu 31.5.2024. Saatavissa https://www.databricks.com/glossary/medallion-architecture

du Mortier, G. 2021. What Are Facts and Dimensions in a Data Warehouse? Blogi. Viitattu 14.6.2024. Saatavissa https://vertabelo.com/blog/facts-dimensions-data-warehouse/

Escórcio, J. 2024. Spark & Databricks: Power in Data. Viitattu 19.10.2024. Saatavissa https://medium.com/@jv.escorcio/spark-databricks-power-in-data-d0b52d929b6f

Gates, Sara. 2023. Data Warehouse vs Data Lake vs Data Lakehouse: Definitions, Similarities, and Differences. Blogi. Viitattu 26.6.2024. Saatavissa https://www.montecarlodata.com/blog-data-warehouse-vs-data-lake-vs-data-lakehouse-definitions-similarities-and-differences/

Gomezm, D. 2024. Understanding Unity Catalog. Viitattu 16.11.2024. Saatavissa https://community.databricks.com/t5/technical-blog/understanding-unity-catalog/ba-p/93478

Hewlett Packard Enterprise. 2024. What is Delta Lake? Viitattu 12.5.2024. Saatavissa https://www.hpe.com/emea_europe/en/what-is/delta-lake.html

IBM a. What is ETL (extract, transform, load)? Viitattu 25.3.2024. Saatavissa https://www.ibm.com/topics/etl

IBM b. What is a data lakehouse? Viitattu 15.5.2024. Saatavissa https://www.ibm.com/topics/data-lakehouse

IBM. 2021. Keys to join the fact table with the dimension tables. Viitattu 19.6.2024. Saatavissa https://www.ibm.com/docs/en/informix-servers/12.10?topic=table-keys-join-fact-dimension-tables

Kimball, R. & Ross, M. 2013. The data warehouse toolkit. 3. uudistettu painos. John Wiley & Sons Inc.

Kutay, J. Data Warehouse vs. Data Lake vs. Data Lakehouse: An Overview of Three Cloud Data Storage Patterns. Viitattu 20.9.2024. Saatavissa https://www.striim.com/blog/data-warehouse-vs-data-lake-vs-data-lakehouse-an-overview/

Lingappa, S. 2024. Understanding Snowflake Costs: Breakdown example. Viitattu 12.10.2024. Saatavissa https://medium.com/@sanusa100/understanding-snowflake-costs-breakdown-example-8683a7efa9e6

Lukka, K. 2001. Konstruktiivinen tutkimusote. Viitattu 29.11.2024. Saatavissa https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/

Marattha, P. 2024. Databricks Pricing 101: A Comprehensive Guide (2024) Blogi. Viitattu 16.11.2024. Saatavissa https://www.chaosgenius.io/blog/databricks-pricing-guide/

Microsoft. 2022. Indexes on dedicated SQL pool tables in Azure Synapse Analytics Viitattu 27.10.2024. Saatavissa https://learn.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-index

Microsoft. 2023. Azure Synapse Analytics security white paper: Introduction. Viitattu 20.10.2024. Saatavissa https://learn.microsoft.com/en-us/azure/synapse-analytics/guidance/security-white-paper-introduction

Microsoft. 2024a. Connect to a SQL warehouse. Viitattu 19.10.2024. Saatavissa https://learn.microsoft.com/en-us/azure/databricks/compute/sql-warehouse/

Microsoft. 2024b. Integration runtime in Azure Data Factory Viitattu 20.10.2024. Saatavissa https://learn.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime

Microsoft. 2024c. What is Microsoft Fabric? Viitattu 20.10.2024. Saatavissa https://learn.microsoft.com/en-us/fabric/get-started/microsoft-fabric-overview

Microsoft. 2024d. What is data warehousing in Microsoft Fabric? Viitattu 26.10.2024. Saatavissa https://learn.microsoft.com/en-us/fabric/data-warehouse/data-warehousing

Microsoft. 2024e. Burstable capacity in Fabric Data Warehouse Viitattu 26.10.2024. Saatavissa https://learn.microsoft.com/en-us/fabric/data-warehouse/burstable-capacity

Microsoft. 2024f. Azure Synapse Analytics pricing Viitattu 20.10.2024. Saatavissa https://azure.microsoft.com/en-us/pricing/details/synapse-analytics/

Microsoft. 2024g. What is Databricks? Viitattu 19.10.2024. Saatavissa https://learn.microsoft.com/fi-fi/azure/databricks/introduction/

Morales, M. 2024. Data Lake vs. Data Warehouse vs. Data Lakehouse: Understanding the Differences. Blogi. Viitattu 20.9.2024. Saatavissa https://amplitude.com/blog/data-lake-vs-warehouse-vs-lakehouse

Myllykorpi, T. 2024. Vertaileva tutkimus tutkimusmenetelmänä. Blogi. Viitattu 29.11.2024. Saatavissa https://yhteisillatulilla.weebly.com/yhteisillauml-tulilla/vertaileva-tutkimus-tutkimusmenetelmana

Nikola. 2023. Bursting and Smoothing – Yin and Yang of the Fabric Capacity! Viitattu 26.10.2024. Saatavissa https://data-mozart.com/bursting-and-smoothing-yin-and-yang-of-the-fabric-capacity/

Oracle. 2024. What is a Data Lake? Viitattu 5.4.2024. Saatavissa https://www.oracle.com/th/big-data/data-intelligence-platform/what-is-data-lake/#data-lake-defined

Parkhouse, I. 2021. The role of Data Lakes or Staging Areas for Data Warehouse ETL Viitattu 27.10.2024. Saatavissa https://www.rittmanmead.com/blog/2021/10/the-role-of-data-lakes-or-staging-areas-for-data-warehouse-etl/

Patairya, D. 2024. Star Schema vs. Snowflake Schema Explained. Artikkeli. Viitattu 24.9.2024. Saatavissa https://builtin.com/articles/star-schema-vs-snowflake-schema

Poppy, D. 2024. Data orchestration vs. ETL: What’s the difference? Blogi. Viitattu 29.3.2024. Saatavissa https://www.getdbt.com/blog/data-orchestration-vs-etl

Power, R. 2023. What Will It Actually Take To Lead A Modern Data-First Organization? Viitattu 29.11.2024. Saatavissa https://www.forbes.com/sites/rhettpower/2023/02/12/what-will-it-actually-take-to-lead-a-modern-data-first-organization/

Qivada a. Analytics Development Accelerator (ADA). Tuotesivu. Viitattu 1.10.2024. Saatavissa https://www.qivada.com/product/

Qivada b. Customer Stories. Tuotesivu. Viitattu 1.10.2024. Saatavissa https://www.qivada.com/customer-stories/

Qivada c. Pricing. Tuotesivu. Viitattu 1.10.2024. Saatavissa https://www.qivada.com/ada-pricing/

Qlik a. Delta Lake. Viitattu 12.5.2024. Saatavissa https://www.qlik.com/us/data-lake/delta-lake

Qlik b. Cloud Data Warehouse. Viitattu 28.9.2024. Saatavissa https://www.qlik.com/us/cloud-data-migration/cloud-data-warehouse

Salo, I. 2014. Big Data ja pilvipalvelut. 1. painos. Jyväskylä: Docendo.

SAP. Mikä on tietovarasto? Viitattu 14.6.2024. Saatavissa https://www.sap.com/finland/products/technology-platform/datasphere/what-is-a-data-warehouse.html

Snowflake. 2024a. Cloud Platforms. Viitattu 12.10.2024. Saatavissa https://docs.snowflake.com/en/user-guide/intro-cloud-platforms

Snowflake. 2024b. Key Concepts. Viitattu 12.10.2024. Saatavissa https://docs.snowflake.com/en/user-guide/intro-key-concepts

Snowflake. 2024c. Data Cloud Glossary. Viitattu 12.10.2024. Saatavissa https://www.snowflake.com/data-cloud-glossary/data-warehousing/

Snowflake. 2024d. Snowflake Open Catalog. Viitattu 12.10.2024. Saatavissa https://www.snowflake.com/en/data-cloud/open-catalog/

Snowflake. 2024e. SnowSight. Viitattu 12.10.2024. Saatavissa https://docs.snowflake.com/en/user-guide/ui-snowsight

Snowflake. 2024f. Sample Data: TPC-H Viitattu 27.10.2024. Saatavissa https://docs.snowflake.com/en/user-guide/sample-data-tpch

Stryker, C. 2024. What is a data pipeline? Viitattu 22.3.2024. Saatavissa https://www.ibm.com/topics/data-pipeline

Tapionsalo, O. 2024. Databricks Lakehouse – Tietoallas ja tietovarasto samassa paketissa. Blogi. Viitattu 20.9.2024. Saatavissa https://www.epicalgroup.com/fi/blogi/databricks-lakehouse-tietoallas-ja-tietovarasto-samassa-paketissa

The Overlay Team. All You Need to Know About Modern Data Warehousing. Viitattu 29.11.2024. Saatavissa https://www.overlayanalytics.com/post/modern-data-warehousing

TPC. TPC Benchmarks Overview Viitattu 27.10.2024. Saatavissa https://www.tpc.org/information/benchmarks5.asp

Wikipedia. 2022. Tietokannan normalisointi. Viitattu 18.6.2024. Saatavissa https://fi.wikipedia.org/wiki/Tietokannan_normalisointi

Wikipedia. 2024. Denormalization. Viitattu 19.6.2024. Saatavissa https://en.wikipedia.org/wiki/Denormalization