

Santeri Kemoff

Tietokannat: SQL, NoSQL ja Graafitietokannat

Tietokannat: SQL, NoSQL ja Graafitietokannat

Santeri Kemoff
Opinnäytetyö
Kevät 2024
Tieto- ja viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka, Ohjelmisto

Tekijä(t): Santeri Kemoff

Opinnäytetyön nimi: Tietokannat: SQL, NoSQL ja Graafitietokannat

Työn ohjaaja(t): Reima Riihimäki

Työn valmistumislukukausi ja -vuosi: Kevät 2024

Sivumäärä: 30

Opinnäytetyön ensisijaisena tavoitteena oli esitellä graafitietokantoja, niiden käyttötarkoituksia, sekä hyviä ja huonoja puolia. Tämän tavoitteen täyttämiseksi esittelin myös vanhempia ja vaihtoehtoisia tietokantatyyppejä. Esittelin digitaalisten tietokantojen historiaa sekä kehitystä vuosien varrella. Lisäksi esittelin relaatio- ja graafitietokantojen käyttöä kahdella esimerkkietokannalla.

Asiasanat: Tietokanta, Graafitietokanta, NoSQL

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in information technology, Option of software

Author(s): Santeri Kemoff
Title of thesis: Databases: SQL, NoSQL and Graph databases
Supervisor(s): Reima Riihimäki
Term and year when the thesis was submitted: Spring 2024
Number of pages: 30

The primary goal of this thesis was to showcase graph databases by going over their strengths and weaknesses. To achieve this, I also had to go over the history of modern databases, and to review all the commonly used database types. I also set up two different example databases, one representing a traditional relational database, and the other one representing a graph database.

Keywords: Database, Graph database, NoSQL

KÄSITTEET

Row = Yksinkertaisesti rowlla tarkoitetaan tietokantataulussa olevaa riviä, joka sisältää yhteen aiheeseen, esimerkiksi henkilöön, liittyvää tietoa

Column = Columnilla tarkoitetaan tietokantataulun sarakkeita, eli pystyrivejä. Jokainen sarake sisältää tietoa monesta eri kohteesta.

Table = Table eli taulu on yhdistelmä rivejä ja sarakkeita. Nämä muodostavat relaatiotietokannan perustan.

Primary key = Primary keyllä tarkoitetaan tietokantataulun pääavainarvoa, eli saraketta, jota käytetään taulun linkittämiseen muihin tauluihin.

Foreign key = Foreign key on jonkun muun taulun pääavainarvo, jota käytetään sen linkittämiseen nykyiseen tauluun. Foreign keyn ja Primary keyn tulee vastata toisiaan, jotta linkitys on mahdollista.

Scheme = Skeema on tietokannan perusrakenne, ikään kuin talon piirustukset, joiden perusteella kaikki rakennetaan.

Node/Vertice = Tietopiste graafitietokannassa. Sisältää itseensä liittyvää tietoa, sekä yhdistyy muihin nodeihin linkeillä

Link/Edge = Graafitietokannassa nodeja yhdistävä tieto. Voi sisältää ylimääräistä tietoa, ja yleensä kuvaa nodejen välistä suhdetta, esimerkiksi "Omistaa".

SISÄLLYS

1	JOHDANTO.....	7
2	TIETOKANNAT.....	8
3	RELAATITietokannat ja SQL	9
3.1	Tietoturva.....	9
3.2	Työkaluja ja ohjelmistoja.....	9
4	NOSQL	10
4.1	Historiaa	10
4.2	Yleistä tietoa NoSQL-tietokannoista.....	10
4.3	NoSQL-tietokantatyypit.....	11
5	GRAAFITietokannat.....	12
5.1	Yleiset käyttötarkoitukset.....	12
5.2	JanusGraph	13
5.3	Visualisointi.....	13
6	VERTAILU.....	14
6.1	Relaatiotietokannat	14
6.2	NoSQL.....	15
6.2.1	Dokumenttitietokanta	15
6.2.2	Avain-arvotietokanta	16
6.2.3	Wide-column store.....	16
6.2.4	Graafitietokanta	16
7	RELAATITietokantaesimerkki	18
8	GRAAFITietokantaesimerkki	24
9	POHDINTA.....	30
	LÄHTEET.....	31

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on esitellä mitä ovat tietokannat, mihin niitä käytetään, sekä niiden historiaa. Lisäksi esittelen erilaisia tietokantatyyppejä, niiden vahvuuksia, heikkouksia ja käyttötarkoituksia, joissa niitä sovelletaan. Lopuksi teen kaksi esimerkkitietokantaa, joihin tallennan saman tiedon, mutta sovellettuna molempiin tietokantoihin. Esimerkkietokantatyypeiksi valitsin perinteisen relaatiotietokannan, jonka käyttöönottoon käytän Oraclen MySQL-ohjelmistoja, sekä graafitietokannan, jonka käyttöönottoon käytän JanusGraphia, Apache Cassandraa sekä ohjelmistoa, jonka olen itse aikaisemmin kirjoittanut. Valitsin nämä tietokantatyypit, koska relaatiotietokanta on moderneista tietokantatyypeistä vanhin ja graafitietokanta on käsittääkseni uusin laajalti käytönotettu tietokantatyyppi. Näillä esimerkkitietokannoilla pyrin esittelemään näiden kahden tietokantatyypin rakenne- ja käyttötarkoituseroja.

2 TIETOKANNAT

Tietokanta on järjestelmä, joka on suunniteltu tiedon järjestelmällistä tallentamista ja etsimistä varten. Tietokanta on pohjimmaltaan toimintaperiaatteeltaan kuin kirjasto, sillä sinne säilötään tietoa, yleensä ennalta määrättyssä järjestyksessä. Toki tähänkin on poikkeuksia, ja kaikki tietokannat eivät vaadi ennalta määrättyä järjestystä, mutta kerron tästä tarkemmin myöhemmissä luvuissa.

Ensimmäiset nykymuotoiset tietokannat otettiin käyttöön 1960-luvulla, mitä ennen käytettiin peräkkäin tallentavia tiedontallennusmenetelmiä, kuten magneettinauhoja. Nämä ensimmäiset tietokannat olivat navigaatiotietokantoja, joissa tieto löydettiin seuraamalla mainintoja muista tietopisteistä. (1.) Eli nyt tietoa pystyttiin tallentamaan ikään kuin kahdessa ulottuvuudessa, kun taas vanhemmissa tallennustavoissa kaikki tieto oli tallennettu peräkkäin, ilman mahdollisuutta navigoida kanta muilla tavoilla.

IBM:n Edgar Codd kehitti datan relaatiomallin vuonna 1970. Datan relaatiomalli tarkoittaa tietokantarakennetta, missä dataa säilytetään taulukoissa, jotka voivat linkittyä useisiin muihin taulukoihin. Seuraavaksi täytyi kehittää ohjelmointikieli tällaisten relaatiotietokantojen hallitsemiseen. IBM sai valmiiksi ja käyttöönotti SEQUEL-kielen vuonna 1974. SEQUEL on lyhenne sanoista Structured English Query Language, ja se lyhennettiin myöhemmin SQL:lläksi. Relational Software, nykyään Oracle, oli ensimmäinen yritys, joka tarjosi hallintajärjestelmän SQL-tietokannoille. (2.) SQL-kieli on edelleen suosittu ja laajasti käytetty kieli relaatiotietokantojen hallitsemiseen.

3 RELAATITIETOKANNAT JA SQL

Relaatiotietokannat ovat vanhin tietokantatyyppejä, joka on edelleen suosittu ja laajalti käytössä (3). Tässä luvussa käyn läpi asioita, joita on hyvä tietää relaatiotietokannoista. Relatiotietokannan käyttöönottoa ja käyttöä esittelen tarkemmin myöhemmässä luvussa.

3.1 Tietoturva

Todennäköisesti tunnetuin tietoturvariski mikä yhdistetään tietokantoihin, on SQL injection -hyökkäykset. Tällainen hyökkäys hyväksikäyttää esimerkiksi huonosti suunnitellun verkkosivun tekstikenttää ja saa tietokannan suorittamaan ei-haluttuja käskyjä. Eräs esimerkki tällaisesta hyökkäyksestä tapahtui, kun joku asensi autoonsa rekisterikilven, jossa oli käsky "DROP DATABASE Table", ja ilmeisesti onnistui poistamaan tietoja liikenteenvalvontaan liittyvästä tietokannasta (4).

3.2 Työkaluja ja ohjelmistoja

MySQL Workbench on tietokantatyökalu, jolla voi luoda ja hallita SQL Server -tietokantoja. Sillä voi muun muassa suunnitella ja muokata skeemoja, hallita tietokannan käyttäjien oikeuksia ja suorittaa kyselyitä kantaan. Se on myös ohjelmisto, jota käytän esimerkirelaatiokannan tekemiseen tätä opinnäytetyötä varten.

4 NOSQL

NoSQL-termillä tarkoitetaan kaikkia ei-relaatiotyyppisiä tietokantoja. NoSQL on lyhenne sanoista Not only SQL, millä viitataan siihen, että näihin kantoihin voi käyttää muitakin hakumenetelmiä kuin SQL-kieltä (5). NoSQL-tietokannat eroavat yleensä myös rakenteeltaan perinteisistä relaatiokannoista, ja niihin voi tallentaa tietoa myös muissa muodoissa perinteisten taulujen lisäksi. NoSQL-tietokannat eivät myöskään välttämättä vaadi tiukkaa skeemaa toisin kuin relaatiotietokannat.

4.1 Historiaa

NoSQL sai alkunsa 1990-luvulla, kun internetin suosion nopea kasvu toi esille relaatiokantojen rajoitteet. Perinteiset relaatiomalliin perustuvat tietokannat eivät pysyneet kasvavan tietoliikenteen perässä, eivätkä ne pystyneet käsittelemään tarpeeksi monimuotoista dataa.

NoSQL-termiä käytti ensimmäisen kerran Carlos Strozzi vuonna 1998, mutta hän puhui silloin relaatiokannasta, joka ei käyttänyt SQL-kieltä (6). Termi sai nykyisen tarkoituksensa vuonna 2009, kun Eric Evans ja Johan Oskarsson käyttivät sitä ei-relaatiokantojen kuvailuun. Alun perin NoSQL tarkoitti kirjaimellisesti sitä, että tietokannassa ei käytetty SQL-kieltä, mutta se on sittemmin muuttunut tarkoittamaan, että tietokannassa käytetään muitakin hallintamenetelmiä SQL:n lisäksi.

4.2 Yleistä tietoa NoSQL-tietokannoista

Eric Brewer julkaisi CAP-periaatteen vuonna 1999. Periaatteen mukaan jaettu tietokanta ei voi tarjota samanaikaisesti kuin kaksi kolmesta takauksesta, jotka ovat seuraavat:

- Consistency
- Availability
- Partition tolerance

Consistency tarkoittaa, että data tietokannassa pysyy yhdenmukaisena muutosten jälkeen, eli kaikki asiakasohjelmat näkevät identtisen version tietokannasta. Availability merkitsee, että tietokanta on jatkuvasti saatavilla ilman taukoja. Partition tolerance tarkoittaa, että järjestelmä voi jatkaa

toimintaa, vaikka eri osioiden välinen kommunikointi ei olisi enää luotettavaa, eli vaikka kaikki viestit eivät pääsisi perille tietokantaosioiden välillä. (6.)

4.3 NoSQL-tietokantatyypit

- Document database = Dokumenttitietokanta sisältää nimensä mukaisesti taulujen sijasta dokumentteja. Nämä dokumentit voivat olla vaikkapa JSON- tai XML-muotoisia. Dokumentit voivat olla myös kokoelmia muista dokumenteista, eli esimerkiksi dokumentissa voi olla listattuna kaikki asiakkaat, heidän tiedoissaan voi olla lista tehdyistä ostoksista, ja ne voivat sisältää tietoa kyseisestä ostoksesta. (7.)
- Key-value database = Avain-arvotietokanta on yksinkertainen NoSQL-tietokanta, joka sisältää kaksi saraketta; ensimmäisessä on avainarvo, jolla tietoa haetaan, ja toinen sisältää halutun tiedon. Esimerkkinä tästä voisi olla lista palvelimista, ja hakemalla palvelimen nimeä löytää sen IP-osoitteen. (8.)
- Wide-column store = Wide-column store tai Wide-column Database on taulupohjainen NoSQL-tietokanta, jossa sarakkeiden nimet ja formaatit voivat vaihdella rivien välillä. Tämä tarjoaa etuja perinteiseen relaatiokantaan verrattuna, kuten esimerkiksi sen, että jos yhdelle riville pitää lisätä ylimääräinen tietokenttä, vastaavaa saraketta ei tarvitse lisätä muille riveille kyseisen taulun sisällä. Tämän myötä tietokanta on rakenteeltaan paljon joustavampi, ja se on helpompi mukauttaa käyttötarkoitukseen sen muuttuessa. (9.)
- Graph database = Graafitietokannassa tieto säilytetään nodeissa/verticeissä ja linkeissä/edgeissä. Nodet ovat varsinaisia datapisteitä, jotka voivat sisältää vaikkapa asiakkaita ja tuotteita, ja linkit kertovat nodejen välisistä suhteista. Esimerkiksi asiakkaan ja tuotteen välillä voisi olla Bought-niminen linkki, joka sisältäisi tietoa ostotapahtumasta. Toinen esimerkki olisi sukupuoli, jossa vanhemman ja lapsen välillä voisi olla "hasChild"-niminen linkki. Graafitietokannat ovat tämän opinnäytetyön pääasiallinen aihe, joten kerron näistä lisää omassa luvussaan.

5 GRAAFITIEKOKANNAT

Graafitietokannat ovat NoSQL-tietokannan tyyppi, joka on kasvattanut suosiotaan viime vuosina. Niissä tietoa ei pääasiallisesti säilytetä taulukoissa, joihin tehdään kyselyitä, vaan tieto on tallennettu tietopisteisiin eli nodeihin. Nämä nodet sisältävät tietoa kyseisestä asiasta, (10) eli esimerkiksi henkilöä kuvaava node voi sisältää syntymäajan. Näiden tietopisteiden välisiä suhteita kuvataan linkeillä, eli nuolilla, jotka kertovat suhteen tyyppin. Eli esimerkiksi henkilön ja kaupungin välillä voisi olla kaupunkiin osoittava nuoli, jonka otsikko on ”Asuinpaikka”, ja sen nuolen tiedoissa voisi olla päivämäärä, jolloin henkilö on kyseiseen kaupunkiin muuttanut.

Graafitietokannat voivat myös rakentua jonkun toisen tietokannan päälle, kuten JanusGraph, jonka käyttöönottoa käsittelen myöhemmässä luvussa. JanusGraph tukee useita eri taustatietokantoja, mutta tässä työssä käytän Apache Cassandraa.

5.1 Yleiset käyttötarkoitukset

Ehkä tunnetuin graafitietokantojen käyttäjä on Facebook, joka 2000-luvun loppupuolella huomasi, että heidän käyttämästään relaatiotietokannasta alkoi muodostua pullonkaula, joka rajoitti sivuston suorituskykyä. Alun perin ongelmaa yritettiin ratkaista luomalla relaatiotietokannan päälle välimuisti, joka sisälsi avainarvopareja nopeampia hakuja varten. Tämän jälkeen kuitenkin huomattiin, että relaatiokanta ei ole optimaalinen ratkaisu palvelemaan Facebookin yleisimpiä tietokantakyselyitä, kuten tykkäysten määriä ja kaverikutsuja. Graafitietokanta sen sijaan on täydellinen ratkaisu tällaisessa tilanteessa, sillä niiden pohjimmainen toimintaperiaate on kuvata suhteita eri datapisteiden välillä. Facebook kuitenkin halusi pitää relaatiotietokannan ensisijaisena tiedon säilytysmenetelmänä, oletettavasti niiden todistetun luotettavuuden vuoksi, joten ratkaisu oli luoda relaatiotietokannan päälle graafimallinen välimuisti, joka käsittelee kaikkein yleisimmät kyselyt. (11; 12.)

Toinen yleinen käyttötarkoitus graafitietokannoille on petosten tunnistaminen ja ehkäisy. (13) Graafitietokantaa analysoimalla voi nähdä yhteyksiä vaikkapa luottokorttien ja sijaintien välillä tai löytää muita yhdistäviä tekijöitä esimerkiksi epäilyttävien ostotapahtumien välillä (14).

5.2 JanusGraph

JanusGraph on avoimen lähdekoodin graafitietokanta, jonka ensimmäinen versio julkaistiin vuonna 2017. Se ei tallenna itseensä tietokannan tietoja, vaan vaatii jonkin tuetun NoSQL-tietokannan, vaikkakin JanusGraphin saa valmiina pakettina, jossa tulee Apache Cassandra ja Elasticsearch mukana. Tätä valmista pakettia tulen myös käyttämään tämän työn esimerkkietokannassa. JanusGraph on Linux Foundation alainen projekti, ja sen tukijoihin kuuluu mm. Google ja IBM. Se on monilla suurilla yrityksillä käytössä tuotantoympäristössä, kuten Ebay:lla, Target:illa ja RedHat:illa (15).

5.3 Visualisointi

Graafitietokantojen rakenne mahdollistaa niiden visualisoinnin esimerkiksi helposti ymmärrettävinä kuvaajina tai pistepilvinä. Esimerkiksi D3.js-kirjasto mahdollistaa datan esittämisen muun muassa sunburst-diagrammina, joka esittää tietokannan sisällön kerroksittain. Graafitietokannan sisällön voisi esittää myös kartalla, jos tarkoituksena on visualisoida yhteyksiä eri sijaintien välillä, esimerkiksi lentoreittejä. Tällainen käyttötarkoitus vaatisi vain karttapohjan, jossa on koordinaattitiedot ja että jokaiseen nodeen on tallennettu oikeat koordinaatit. Tämän jälkeen nodejen väliset linkit voivat esittää esimerkiksi edellä mainittuja lentoreittejä. Esittelen visualisointia tarkemmin myöhemmässä luvussa, kun teen esimerkkietokannan ja visualisointiratkaisun.

6 VERTAILU

Tässä luvussa käyn läpi eri tietokantatyyppeiden hyviä ja huonoja puolia sekä mahdollisia käyttökohteita kyseisille tietokantatyypeille. Vertailussa otan huomioon hieman eri tietokantatyyppeiden vaatimuksia laitteistolle, mutta pääasiassa niiden soveltuvuutta eri käyttötarkoituksiin.

6.1 Relaatiotietokannat

Relaatiotietokannat ovat hyviä tilanteissa, joissa tallennettavalla datalla on tunnettu rakenne. Yksi tällainen tilanne on esimerkiksi sosiaalisissa medioissa, missä käyttäjien profiileihin liittyy ennalta määrätty tiedot, kuten sähköposti, syntymäaika ja nimi.

Relaatiotietokanta on myös hyvä tilanteissa joissa tallennustilaa on rajallisesti, sillä hyvin suunniteltu kantarakenne pyrkii minimoimaan ja optimitilanteessa välttämään saman tiedon tallentamisen moneen kertaan.

Relaatiotietokannassa tieto on lajiteltu eri kategorioihin, mikä mahdollistaa tietyn osa-alueen tarkastelun. Esimerkiksi verkkokaupan tietokannassa voidaan säilyttää rekisteröityneet ja rekisteröitymättömät asiakkaat erillään, mikä helpottaa esimerkiksi rekisteröitymättömien asiakkaiden tietojen poistamista tietyn ajan jälkeen. Samassa tietokannassa voidaan myös säilyttää tapahtuneet ostokset erillisessä taulussa, josta ne voidaan yhdistää asiakastauluun esimerkiksi ID:n perusteella.

Relaatiotietokannat eivät sovellu hyvin tilanteisiin, joissa tallennettava tieto ei ole helposti ennalta määriteltävissä, sillä koko tietokantarakenteen eli skeeman tulisi muuttua tiedon mukana. Lisäksi tieto- ja/tai käyttäjämäärän kasvaessa relaatiotietokantojen laajentaminen voi olla ongelmallista. Vaikka yhden palvelimen skaalaaminen vertikaalisesti, eli laitteiston parantaminen esimerkiksi lisäämällä muistia ja tallennustilaa ja/tai vaihtamalla tehokkaampaan prosessoriin on mahdollista, tässä lähestymistavassa on ongelmansa. Budjetista riippuen laajentamista rajoittavaksi tekijäksi tulee joko saatavilla oleva teknologia tai hinta.

Myös tietokannan päivittäminen horisontaalisesti eli uusien palvelimien lisääminen on ongelmallista. Relaatiotietokanta ”vaatii” tiukat standardit datan eheyteen ja yhdenmukaisuuteen. Datamuutosten täytyy onnistua kokonaisuudessaan, tai koko muutos hylätään. Tämä monimutkaistuu huomattavasti, jos data on jaettu useammalle palvelimelle, jotka voivat olla sijoitettu ympäri maailmaa. Lisäksi jos taulut on jaettu useammalle palvelimelle, monimutkaiset kyselyt hidastuvat huomattavasti, kun suuria datamääriä täytyy yhdistellä palvelimien välillä. Säännöt datan jakamiseen palvelimien välillä täytyy miettiä tarkasti, koska sillä voi olla merkittävä vaikutus tietokannan suorituskykyyn. Huonosti jaettu data voi rasittaa yksittäistä palvelinta suhteettoman paljon ja siten muodostaa pullonkaulan, joka hidastaa koko tietokantaa. Relaatiotietokannan hajauttaminen vaikeuttaa myös skeeman muokkaamista, koska sen täytyy aina olla synkronisoitu eri palvelimien eli nodejen välillä. Jos skeeman synkronointi epäonnistuu, muutkaan datamuutokset eivät onnistu sen jälkeen. Tällaisissa ratkaisuisa tietokannan toimivuuden kannalta on hyvin tärkeää, että järjestelmässä on hyvät replikointi- ja palautusmekanismi. (16.)

6.2 NoSQL

NoSQL-tietokantojen yleisin etu on helpompi skaalattavuus (17) ja joustavuus (18), mutta niiden monimuotoisuuden vuoksi niistä ei voi juuri kertoa yleisiä hyviä ja huonoja puolia, vaan erityyppisillä kannoilla on omat vahvuutensa ja heikkoutensa. Käyn siis nyt tarkemmin läpi aikaisemmin listamani NoSQL-tietokantatyypit.

6.2.1 Dokumenttitietokanta

Dokumenttitietokannat ovat rakenteeltaan hyvin joustavia, joten niiden sisältämien dokumenttien sisältöä voi vapaasti muokata ilman, että koko dokumenttiin täytyy tehdä muutoksia. Lisäksi suorituskyky voi tietyissä tilanteissa olla parempi kuin relaatiotietokannoissa, sillä dokumenttitietokannassa ei tarvitse tehdä linkityksiä muihin tauluihin. Toisaalta tämä vapaus myös voi tuoda mukanaan ongelmia, sillä tietokannan rakennetta ei voi valvoa niin helposti. (19.)

6.2.2 Avain-arvotietokanta

Avain-arvotietokannan suurin vahvuus on sen yksinkertaisuus. Jokaisella avaimella on yksi arvo, ja sen tyyppillä ei ole mitään väliä. Tämä yksinkertaisuus tekee tietokannasta kevyen ja sitä myötä nopean. Tämä yksinkertaisuus myös tuo rajoituksia tietokannan käyttöön, sillä esimerkiksi sarakkeiden perusteella suodattaminen ei ole mahdollista.

6.2.3 Wide-column store

Wide-column storen etuja on tietokantataulujen rakenteen joustavuus ja yleensä hyvä suorituskyky. Ne ovat myös yleensä helposti skaalattavissa horisontaalisesti, eli uusien palvelimien lisääminen on helppoa. Tällä suorituskyvyllä on kuitenkin hintansa, sillä wide-column storet eivät yleensä pysty esittämään monimutkaisia suhteita, ja sitä myötä niihin ei myöskään yleensä voi tehdä monimutkaisia kyselyitä. Näiden ominaisuuksien vuoksi ne soveltuvat lähinnä käytettäväksi silloin, kun vaaditaan nopeaa pääsyä monimuotoiseen dataan ja sen nopeaa kirjoitusta. (6; 20.)

6.2.4 Graafitietokanta

Graafitietokantojen rakenne mahdollistaa niiden helpon visualisoinnin esimerkiksi puurakenteena, mikä helpottaa kannan sisällön analysointia. Esimerkiksi kaupan hyllyjärjestystä voisi valvoa graafitietokannalla, ja kannan visualisoimalla voi paljastua, että jollain tuotteella ei ole lainkaan hyllypaikkaa tai se on väärässä paikassa.

Graafitietokanta ei myöskään tarvitse tiukkaa skeemaa, joten sinne voi syöttää haluamaansa dataa ilman, että kannan rakenteeseen tarvitsee tehdä muutoksia. Esimerkiksi kaupan hyllyjärjestystä ja tuotteita kuvaavaan kantaan voisi lisätä noden ”Loppu”, johon voisi linkittää kaikki tuotteet, joita ei ole enää saatavilla. Sitten tuotteen ja ”Loppu”-nodejen välisen linkin voisi nimetä sen mukaan, aiotanko sitä tilata lisää. Eli jos tuote on poistunut valikoimasta, linkin nimeksi tulisi esimerkiksi ”Poistunut valikoimasta”, ja linkin tietoihin voisi laittaa syyn, ja lisää tilattavien tuotteiden linkkien tietoihin voisi laittaa vaikkapa tilattavan määrän.

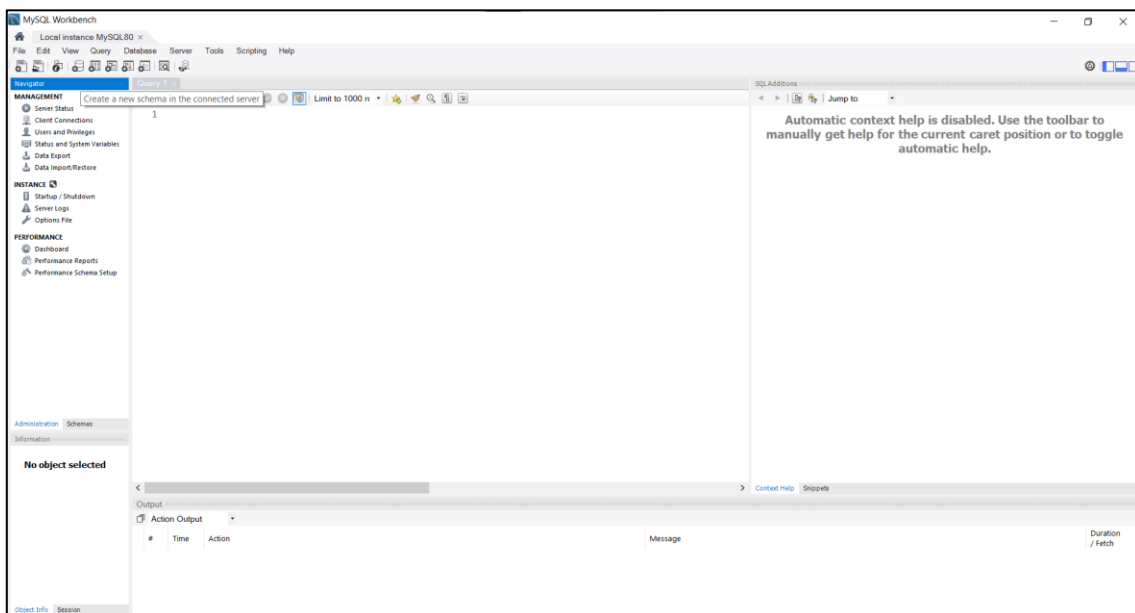
Graafitietokannoilla ei juuri ole merkittäviä huonoja puolia, mutta ne voivat joustavan ja helposti muokattavan rakenteensa vuoksi olla haastavampia käyttää. Lisäksi ne eivät tietenkään ole paras

ratkaisu kaikkiin tilanteisiin, vaan joissain tilanteissa perinteinen relaatiokanta voi olla järkevämpi ja nopeampi ratkaisu.

7 RELAATIOTIETOKANTAESIMERKKI

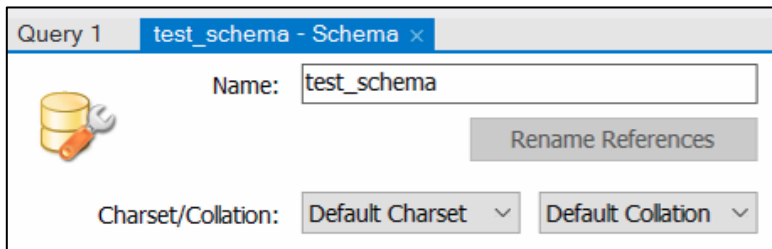
Tavoitteena on tehdä esimerkkitietokanta, johon tallennan tietokoneen ja auton osia sekä niiden hinnat. Ensimmäiseksi teen perinteisen relaatiokannan, ja sen tekemiseen käytän Oraclen MySQL-alustaa. Käytetyt työkalut tässä olivat MySQL Server, joka ylläpitää itse tietokannan, sekä MySQL Workbench, jota käytin skeeman luomiseen sekä tiedon syöttämiseen kantaan.

MySQL Serveriin tehdään uusi tietokanta, johon valitaan root-käyttäjän salasana. Tätä salasanaa käyttäen saa kaikki mahdolliset oikeudet tietokannan hallintaan. MySQL Workbenchissä luodaan uusi yhteys, jonka IP on tässä tapauksessa localhost, sillä molemmat ohjelmistot ovat ajossa samalla tietokoneella. Kun kantaan on kirjaututtu tarvittavilla oikeuksilla, eli tässä tapauksessa root-käyttäjällä, voidaan luoda uusi skeema. Skeeman luonti tapahtuu vasemmasta ylänurkasta, missä on nappi, jossa on sylinterin ja plusmerkin kuva. Kuvassa 1 näkyy tämä MySQL Workbenchin näkymä.

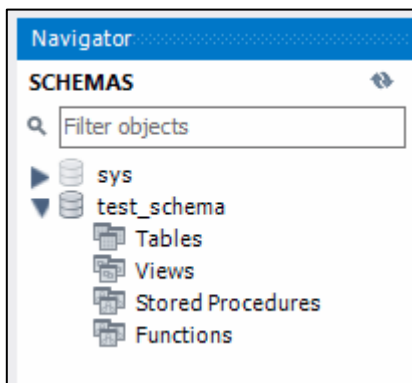


Kuva 1. MySQL Workbench

Kuvassa 2 nimeän skeeman yksinkertaisesti vain "test_schema". Workbenchin vasemmassa laidassa on kuvassa 3 näkyvä "navigator"-osio, jossa näkyy tietokannan sisältämät skeemat. Tässä tapauksessa se sisältää "sys"-, ja "test_schema"-nimiset skeemat. Niitä klikkaamalla saa näkymään esimerkiksi skeeman sisältämät taulut.

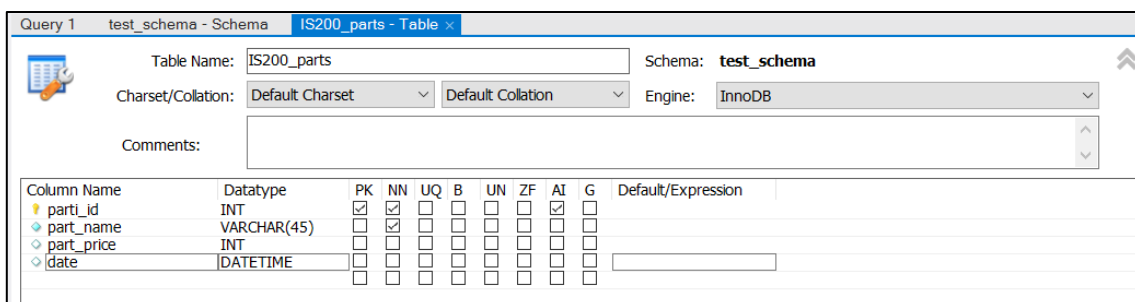


Kuva 2. Skeeman nimeäminen



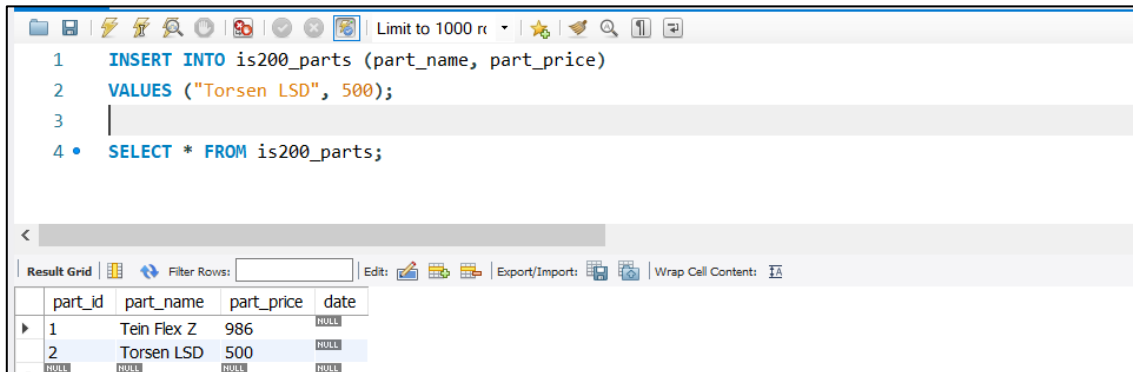
Kuva 3. Navigator-osio

Seuraavaksi luon skeemaan taulun, joka sisältää autoon ostetut osat, niiden hinnan, id:n sekä päivämäärän. Asetan part_id-sarakkeen taulun primary key:ksi eli pääavaimeksi eli sarakkeeksi, jolla kaikki tärkeät operaatiot kuten liitokset ja taulun päivittämiset tehdään. Tämä näkyy kuvassa 4.



Kuva 4. Skeeman luonti

Kuvassa 5 lisään edellä luotuun tauluun tietoa. Tämä tapahtuu INSERT INTO -komennolla, jolle määritellään haluttu taulu, halutut sarakkeet sekä lopulta syötettävä tieto. Tässä kuvassa olen jo syöttänyt kahdella käskyllä kaksi riviä tietoa, mutta ne voisi myös syöttää samalla kertaa.

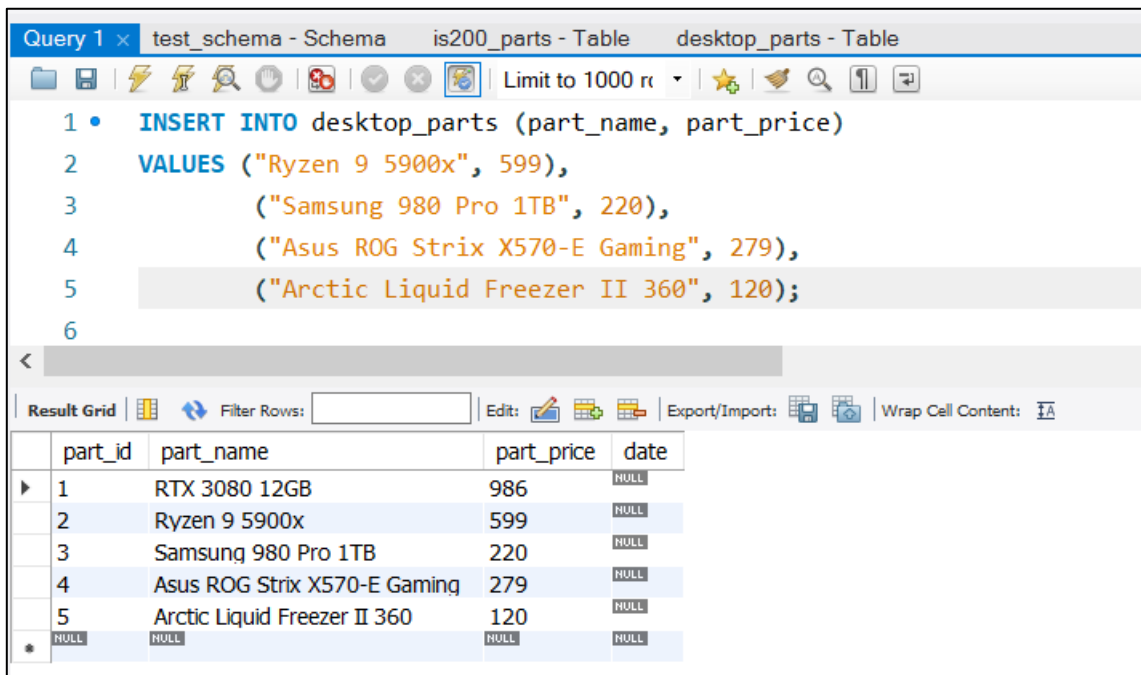


```
1 INSERT INTO is200_parts (part_name, part_price)
2 VALUES ("Torsen LSD", 500);
3
4 • SELECT * FROM is200_parts;
```

part_id	part_name	part_price	date
1	Tein Flex Z	986	NULL
2	Torsen LSD	500	NULL

Kuva 5. Tiedon syöttäminen tauluun

Seuraavaksi teen vastaavan taulun, mutta tällä kertaa pöytäkoneen osia varten. Taulun nimeksi tulee siis "desktop_parts". Kuvassa 6 syötän monta riviä tietoa kerralla, kuten äsken mainitsin olevan mahdollista.

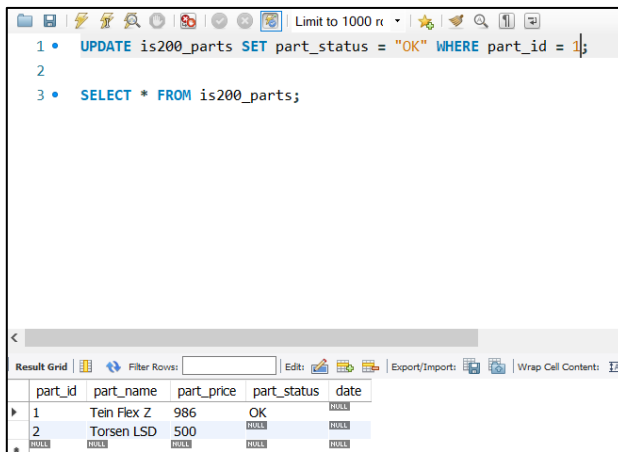


```
1 • INSERT INTO desktop_parts (part_name, part_price)
2 VALUES ("Ryzen 9 5900x", 599),
3         ("Samsung 980 Pro 1TB", 220),
4         ("Asus ROG Strix X570-E Gaming", 279),
5         ("Arctic Liquid Freezer II 360", 120);
6
```

part_id	part_name	part_price	date
1	RTX 3080 12GB	986	NULL
2	Ryzen 9 5900x	599	NULL
3	Samsung 980 Pro 1TB	220	NULL
4	Asus ROG Strix X570-E Gaming	279	NULL
5	Arctic Liquid Freezer II 360	120	NULL

Kuva 6. desktop_parts taulu

Tässä vaiheessa huomasin, että tauluissa olisi hyvä olla sarake, joka kertoo osan nykyisen tilan. Tilaa voisi kuvata esimerkiksi merkinnällä "OK" jos osa on kiinni ja toimii, "Service" jos se vaatii huoltoa, ja niin edelleen. Taulun muokkaamisen voi suorittaa esimerkiksi edellä mainitusta "Navigator"-osiosta. Kun taulua klikkaa hiiren oikealla näppäimellä, tulee esiin vaihtoehto "Alter Table". Täältä tauluun vain yksinkertaisesti lisätään uusi sarake samalla tavalla kuin sitä luodessa. Jotta tauluun voi lisätä tietoa jo olemassa oleville riveille, on käytettävä UPDATE-komentoa. Tällekin komennolle kerrotaan taulun nimi, minkä jälkeen sille kerrotaan sarake, sille asetettava arvo, ja lopuksi haluttu rivi. Tämän tietokannan oletusasetuksilla rivin ilmoittamiseen on käytettävä pääavainsaraketta, mutta jos se ei syystä tai toisesta sovi käyttötarkoitukseen, rajoituksen voi poistaa. Rivin päivittäminen näkyy kuvassa 7.



Kuva 7. Tiedon päivittäminen

Koska lisäsin tauluun sarakkeen, joka sisältää osien nykyisen tilan, voin esimerkiksi suodattaa kaikki huomiota vaativat osat listasta. Tämä tapahtuu WHERE-komennolla, ja näkyy kuvassa 8. Taulun nykyinen rakenne mahdollistaa myös esimerkiksi rahan käytön valvomista, ja voin suodattaa taulusta kaikki osat mihin on mennyt yli 500 €. Tämä näkyy kuvassa 9.

Limit to 1000 rows

```
1 • SELECT * FROM is200_parts WHERE part_status="Service";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

part_id	part_name	part_price	part_status	date
3	Eaton M62	300	Service	NULL
6	Focal K2 Power 165 KR	600	Service	NULL
NULL	NULL	NULL	NULL	NULL

Kuva 8. Tiedon suodattaminen

```
3 • SELECT * FROM is200_parts WHERE part_price>500;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

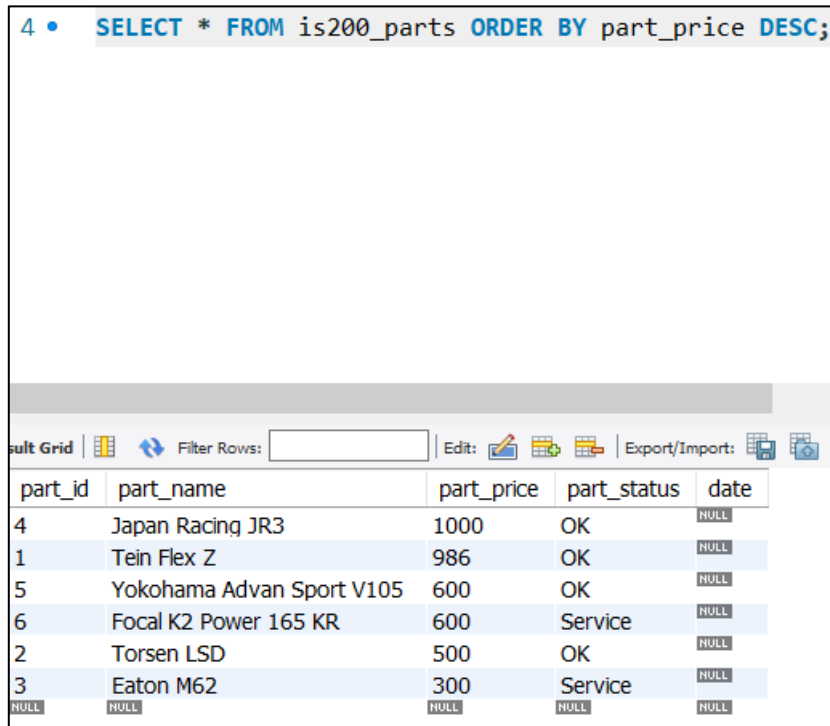
part_id	part_name	part_price	part_status	date
1	Tein Flex Z	986	OK	NULL
4	Japan Racing JR3	1000	OK	NULL
5	Yokohama Advan Sport V105	600	OK	NULL
6	Focal K2 Power 165 KR	600	Service	NULL
NULL	NULL	NULL	NULL	NULL

Kuva 9. Tiedon suodattaminen 2

Tästä voisi olla enemmän hyötyä, jos taulu sisältäisi esimerkiksi tulevat huollot ja odotettavissa olevat korjaustoimenpiteet. Tällöin taulusta voisi suodattaa kaikkein kalleimmat työt, tai jos taulussa olisi esimerkiksi tärkeyssarake, voisi sieltä suodattaa kaikkein kiireisimmät työt. SQL sallii myös

tulosten järjestämisen, joka tapahtuu ORDER BY -komennolla, minkä jälkeen annetaan järjestämiseen käytettävä sarake, ja lopuksi kuvassa 10 näkyvällä DESC-komennolla voi järjestää listan suurimmasta pienimpään.

```
4 • SELECT * FROM is200_parts ORDER BY part_price DESC;
```



The screenshot shows a database query interface. At the top, a SQL query is entered: `SELECT * FROM is200_parts ORDER BY part_price DESC;`. Below the query, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. The main area displays a table with the following data:

part_id	part_name	part_price	part_status	date
4	Japan Racing JR3	1000	OK	NULL
1	Tein Flex Z	986	OK	NULL
5	Yokohama Advan Sport V105	600	OK	NULL
6	Focal K2 Power 165 KR	600	Service	NULL
2	Torsen LSD	500	OK	NULL
3	Eaton M62	300	Service	NULL
NULL	NULL	NULL	NULL	NULL

Kuva 10. Haettavan tiedon järjestäminen

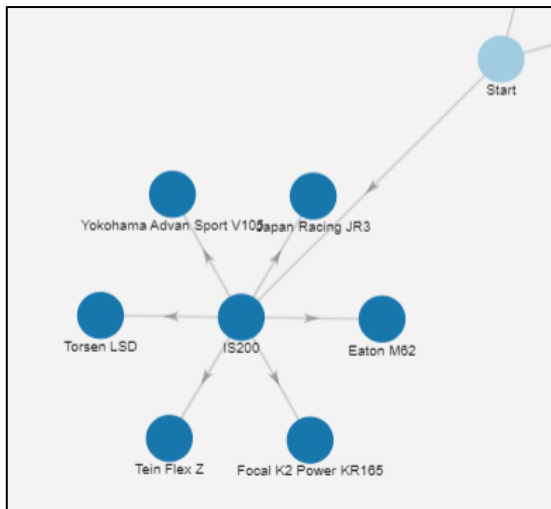
8 GRAAFITIEOKANTAESIMERKKI

Tavoitteena on tehdä esimerkkgraafitietokanta, joka sisältää samat tiedot kuin edellisessä luvussa luotu relaatiotietokanta. Tämän avulla pyrin esittelemään eroja siinä, kuinka näitä kahta eri tietokantatyyppiä käytetään sekä tuomaan esille niiden vahvuuksia ja heikkouksia. Tässä osiossa käytetyt ohjelmistot ovat osittain salattua tietoa, joten en voi niiden toimintaa esitellä kovin tarkasti. Tietokantana toimii JanusGraph, jonka talletustaustaohjelmalla toimii tässä tapauksessa Apache Cassandra.

JanusGraphin asentamiseen käytän Dockeria, sillä siihen löytyy valmis Docker-kontti, joka toimii suoraan omien ohjelmistojeni kanssa. Lisäksi käytän aikaisemmin kirjoittamaani ohjelmaa, joka lukee tietokantaan syötettävät tiedot Excel-tilukosta ja lähettää ne taustaohjelman kautta tietokantaan. Kuvassa 11 näkyy tietokantaan syötettävä taulukko, jossa kahdessa ensimmäisessä sarakkeessa ilmoitetaan luotavan noden nimi, sekä minkä nodejen välille linkki tulisi luoda. Loput sarakkeet voivat sisältää mitä tahansa tietoa, mikä sitten tallennetaan kyseiseen nodeen. Kuvassa 12 näkyy syötetyt tiedot pistepilvenä, sekä niiden välille luodut linkit.

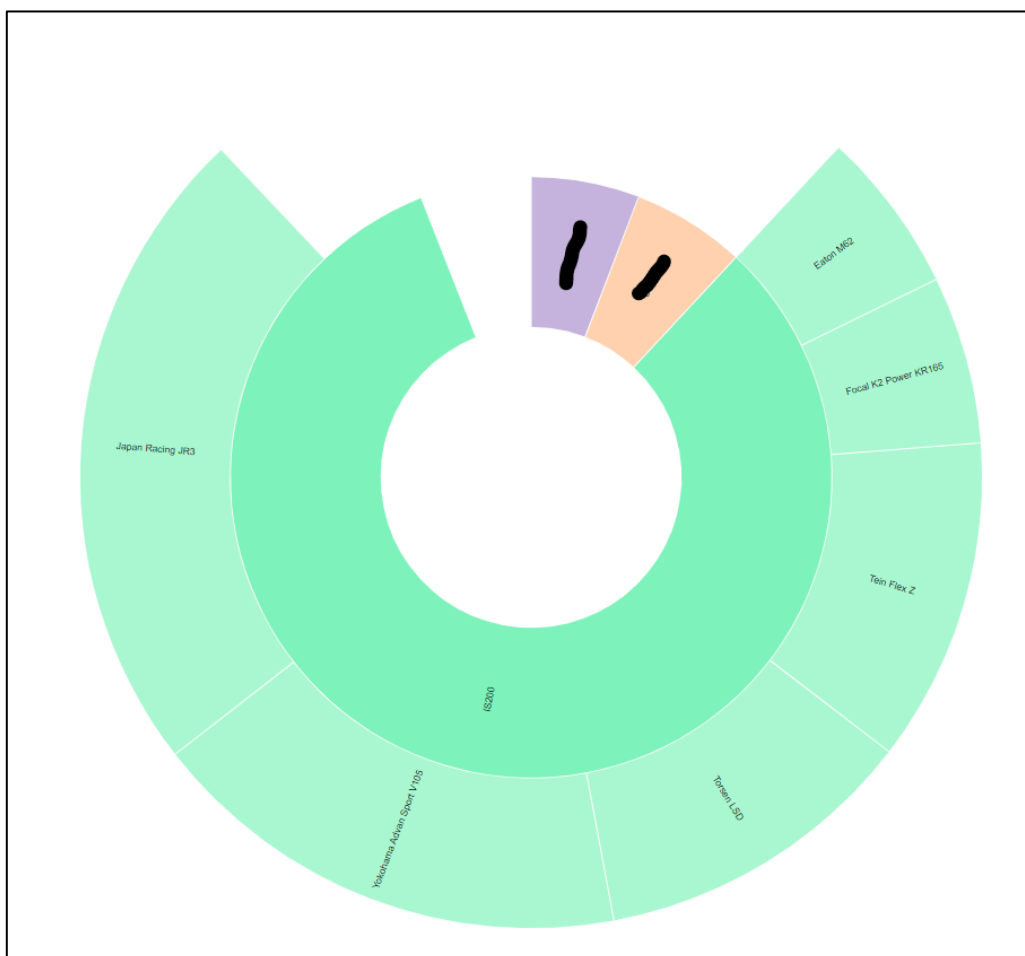
LinkTo	LinkFrom	ID	part_name	part_price	part_status
Start	IS200		IS200		
IS200	Eaton M62		Eaton M62	300	Service
IS200	Torsen LSD		Torsen LSD	500	OK
IS200	Tein Flex Z		Tein Flex Z	986	OK
IS200	Japan Racing JR3		Japan Racing JR3	1000	OK
IS200	Yokohama Advan Sport V105		Yokohama Advan Sport V105	600	OK
IS200	Focal K2 Power KR165		Focal K2 Power KR165		Service

Kuva 11. Excel-tilukko



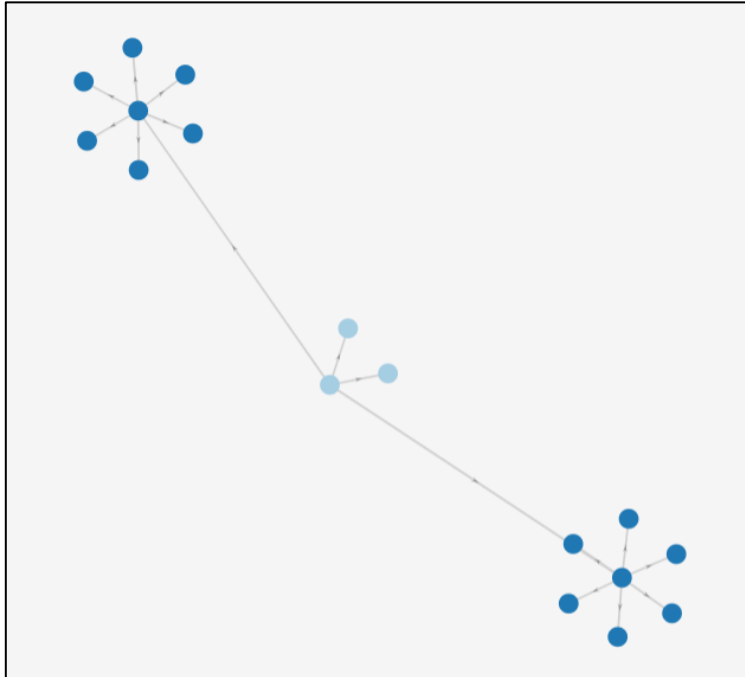
Kuva 12. Pistepilvi

Toinen tapa visualisoida graafitietokannan sisältöä on sunburst-diagrammi, josta näkee nopeasti tietokannan rakenteen sekä kuinka paljon tietoa yksittäisten tietopisteiden alla on. Kuvassa 13 näkyy edellä syötetyt tiedot.



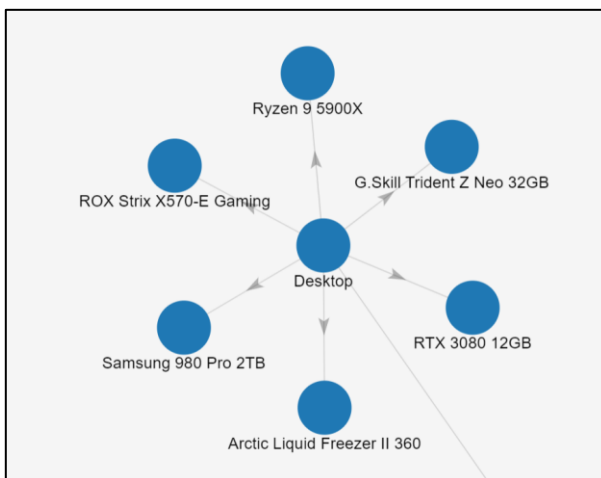
Kuva 13. Sunburst-diagrammi

Seuraavaksi lisään pöytäkoneen osalistan tietokantaan samalla menetelmällä. Kuvassa 14 näkyy pistepilvinäkymä, jossa on kaikki edellä mainitut osat. Kuvan keskellä näkyy vaaleansinisellä "Start" node, mikä toimii rakenteen ensimmäisenä tasona



Kuva 14. Pistepilvi 2

Tässä käyttöliittymässä nodejen nimet katoavat näkyvistä, kun näkymää pienentää tarpeeksi, mutta suurentamalla näkymää tietokoneen osien kohdalla myös niiden nimet tulevat esille, kuten kuvassa 15 näkyy. Lisäksi nodea klikkaamalla aukeaa luettelo, jossa näkyy kaikki kyseiseen nodeen liitetty tieto. Tämä näkyy kuvassa 16.



Kuva 15. Pistepilvi 3

Id	20704
Owner	N/A
Name	G.Skill Trident Z Neo 32GB
Uuid	d99a4611-8bcc-4aa6-b7af-86aa86a7f771
Group	N/A
CreateDate	Wed, 22 May 2024 10:03:25 GMT
LinkTo	Desktop
LinkFrom	G.Skill Trident Z Neo 32GB
ID	N/A
Part_name	G.Skill Trident Z Neo 32GB
Part_price	360
Part_status	OK
System	N/A
Modify	N/A

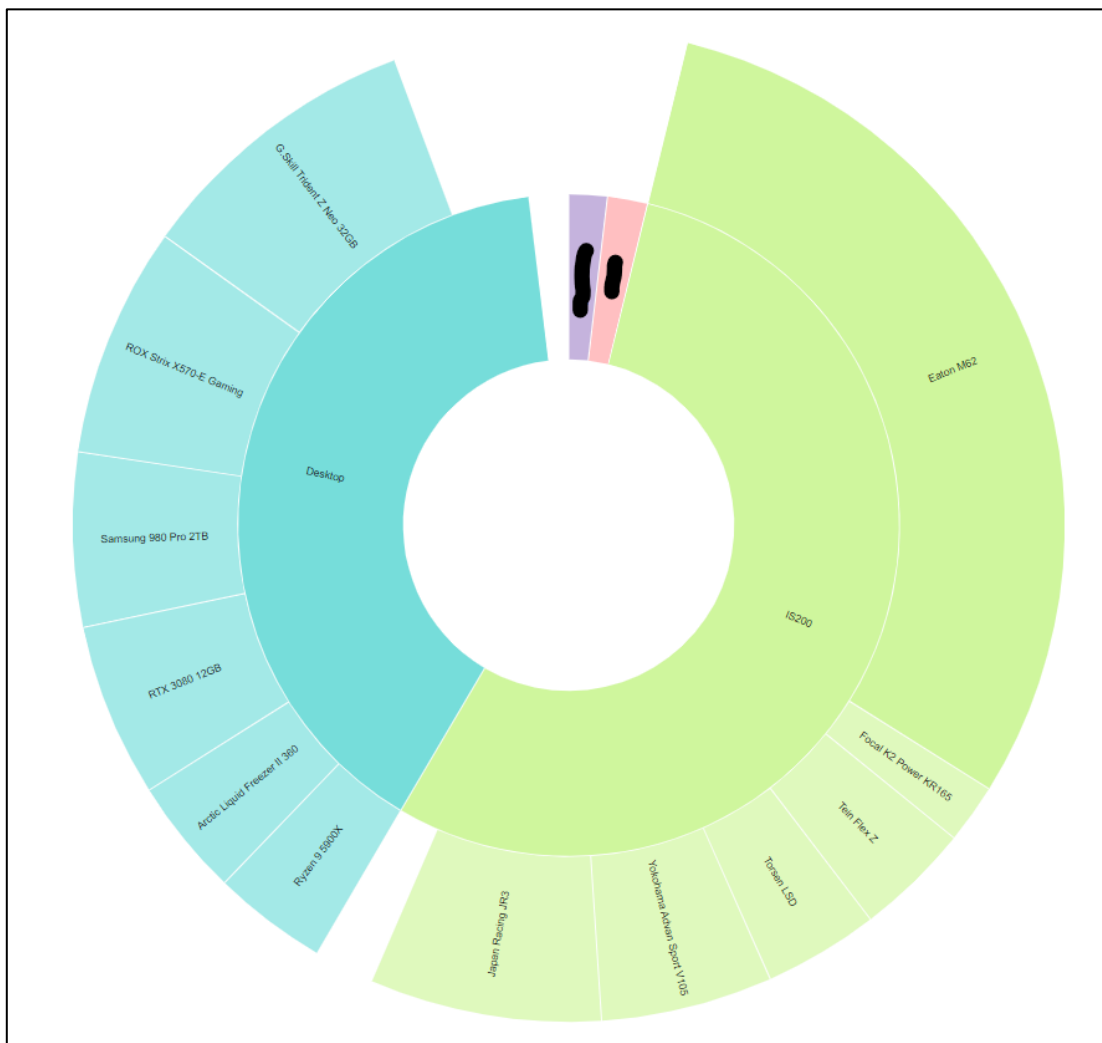
Kuva 16. Noden tiedot

Sunburst-diagrammin käyttötarkoitus alkaa tulla paremmin esille nyt, kun tietokannassa on hieman enemmän toisiinsa liittymätöntä tietoa. Kuvassa 17 näkyy diagrammi, ja auton ja tietokoneen osat on eritelty väreillä.

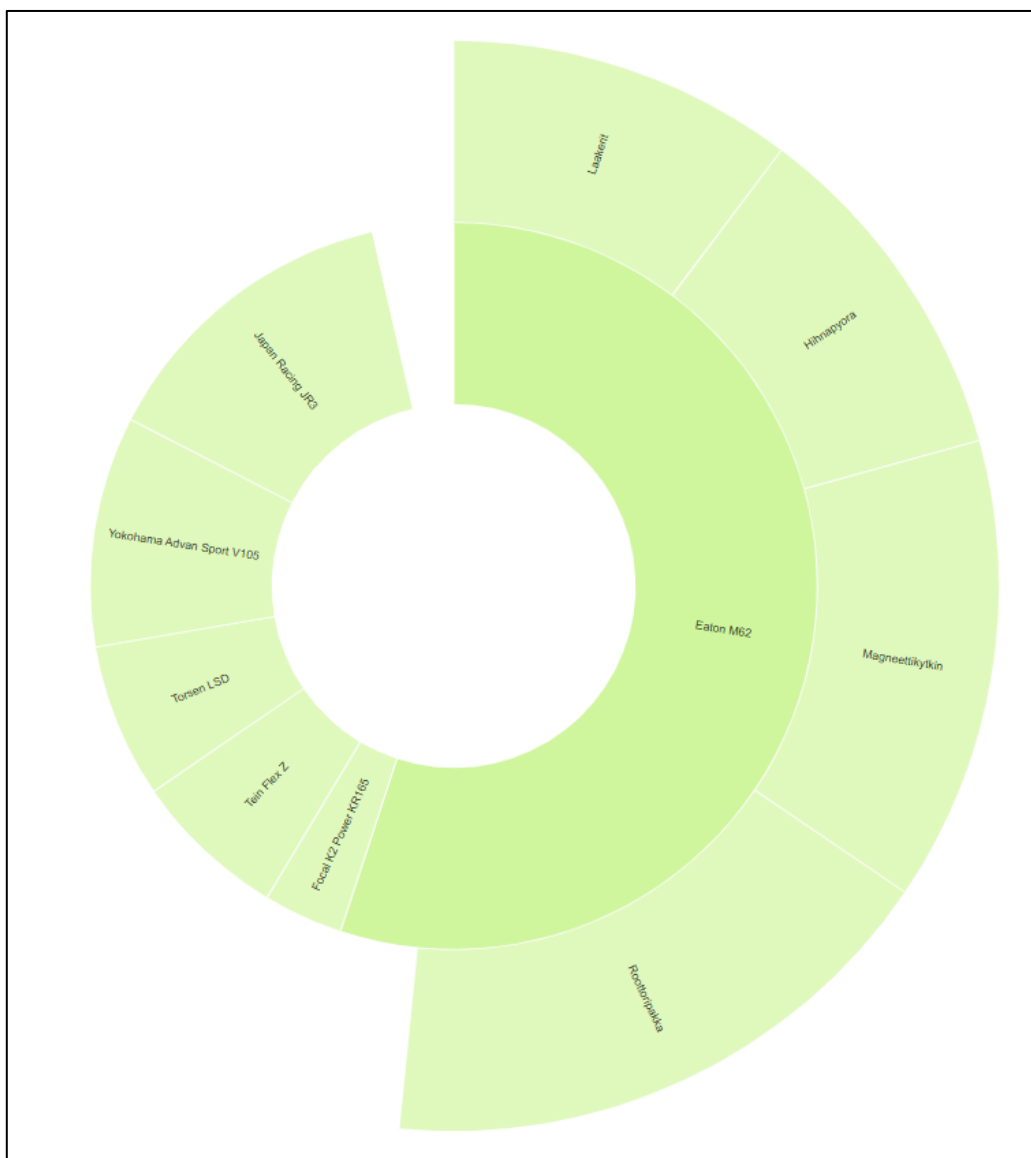


Kuva 17. Sunburst-diagrammi 2

Näin pienellä tietomäärällä sunburst-diagrammin edut eivät tule täysin esille, mutta sen toimintaperiaatteen kyllä ymmärtää. Jos tietokannassa olevan objektin alla on lisää tietoa, se näkyy diagrammissa tummempana ja suurempana sektorina. Lisäsin "Eaton M62"-noden alle luettelon sen sisältämistä komponenteista, ja tämä näkyy kuvassa 18. Kuvassa 19 näkyy, että kun jotain nodea klikkaa, sen alla oleva rakenne tulee esille. Tätä prosessia voi jatkaa niin pitkälle kuin tietorakenteessa on kerroksia.



Kuva 18. Ahtimen osat



Kuva 19. Ahtimen osat 2

9 POHDINTA

Näin pienellä tietomäärällä ei vielä pääse tutkimaan tietokantojen nopeuseroja, mutta erot tietokantojen sisällön esittämisessä ovat jo melko selkeitä. Relaatiotietokannan sisältö on helppo tuoda esille, jos tietää mitä etsii, mutta graafitietokannan sisällön helppo visualisointi mahdollistaa yhteyksien ja virheiden löytämisen datasta.

Graafitietokannan visualisointi voi auttaa myös esimerkiksi omaisuuden hallinnassa, kuten hieman esimerkissä näytin. Visualisointinäkymän voisi esimerkiksi määritellä näyttämään huoltoa vaativat osat punaisena, ja osien huollon tarpeen merkitsemisen tietokantaan voisi automatisoida esimerkiksi päivämäärän perusteella. Relaatiotietokannan päälle voisi toki myös rakentaa vastaavan hälytysjärjestelmän, mutta kokonaisuuksien ja osien välisten suhteiden esittäminen on haastavampaa.

Itse käyttäisin relaatiotietokantaa esimerkiksi videopelin pelaajien tilastojen tallentamiseen, sillä siinä tapauksessa tietokantaan syötettävä tieto on täysin ennalta arvattavissa. Tämän vuoksi kyseisen tietokannan ei tarvitse olla rakenteeltaan joustava, eikä tietoa välttämättä tarvitse visualisoida muuten kuin näyttämällä tietokantaan tallennetut arvot.

LÄHTEET

1. Lutkevich, Ben 2023, Definiton, database (DB), Hakupäivä 12.5.2024, <https://www.tech-target.com/searchdatamanagement/definition/database>
2. Amazon AWS, What is SQL (Structured Query Language?), Hakupäivä 12.5.2024, <https://aws.amazon.com/what-is/sql/>
3. Ellinwood, Justin, Comparing database types: how database types evolved to meet different needs, Hakupäivä 15.4.2024, <https://www.prisma.io/dataguide/intro/comparing-database-types>
4. Hobson, James 2014, SQL Injection fools speed traps and clears your record, Hakupäivä 23.5.2024, <https://hackaday.com/2014/04/04/sql-injection-fools-speed-traps-and-clears-your-record/>
5. MongoDB, What is NoSQL?, Hakupäivä 23.5.2024, <https://www.mongodb.com/resources/basics/databases/nosql-explained>
6. Foote, Keith D. 2018, A Brief History of Non-Relational Databases, Hakupäivä 12.5.2024, <https://www.dataversity.net/a-brief-history-of-non-relational-databases/>
7. MongoDB, What is a Document Database?, Hakupäivä 23.5.2024, <https://www.mongodb.com/resources/basics/databases/document-databases>
8. Redis, What is a Key-Value database?, Hakupäivä 12.5.2024, <https://redis.io/nosql/key-value-databases/>
9. ScyllaDB, Wide-column Database, Hakupäivä 12.5.2024, <https://www.scylladb.com/glossary/wide-column-database/>

10. Amazon AWS, What is a Graph Database? ,Hakupäivä 12.5.2024, <https://aws.amazon.com/nosql/graph/>
11. Anand, Sukhad 2021, Does Facebook really uses graph database?, Hakupäivä 14.5.2024, <https://sukhadanand.medium.com/does-facebook-really-uses-graph-database-5c3c51c6bca5>
12. Neo4J, Facebook shifts to a TAO data store, Hakupäivä 14.5.2024, <https://neo4j.com/news/how-facebook-matured-its-data-structure-and-stepped-into-the-graph-world/>
13. Saarela, Janne, Graph database use cases, Hakupäivä 14.5.2024, <https://www.profium.com/en/blog/graph-database-use-cases/>
14. Smith, Daniel Alexander, Use cases for graph databases, Hakupäivä 14.5.2024, <https://6point6.co.uk/insights/use-cases-for-graph-databases/>
15. JanusGraph, Hakupäivä 4.6.2024, <https://janusgraph.org/>
16. Ahmad, Arslan 2023, Why Is It Hard to Horizontally Scale SQL Databases?, Hakupäivä 11.5.2024, <https://www.designgurus.io/blog/horizontally-scale-sql-databases>
17. Oracle, What is NoSQL?, Hakupäivä 23.5.2024, <https://www.oracle.com/fi/database/nosql/what-is-nosql/>
18. Kelta, Zoumana 2022, NoSQL Databases: What Every Data Scientist Needs to Know, Hakupäivä 13.5.2024, <https://www.datacamp.com/blog/nosql-databases-what-every-data-scientist-needs-to-know>
19. PhoenixNAP, What is a Document Database?, Hakupäivä 23.5.2024, <https://phoenixnap.com/kb/document-database>
20. DatabaseTown, Wide Column Database (Use Cases, Example, Advantages & Disadvantages), Hakupäivä 11.5.2024, <https://databasetown.com/wide-column-database-use-cases>