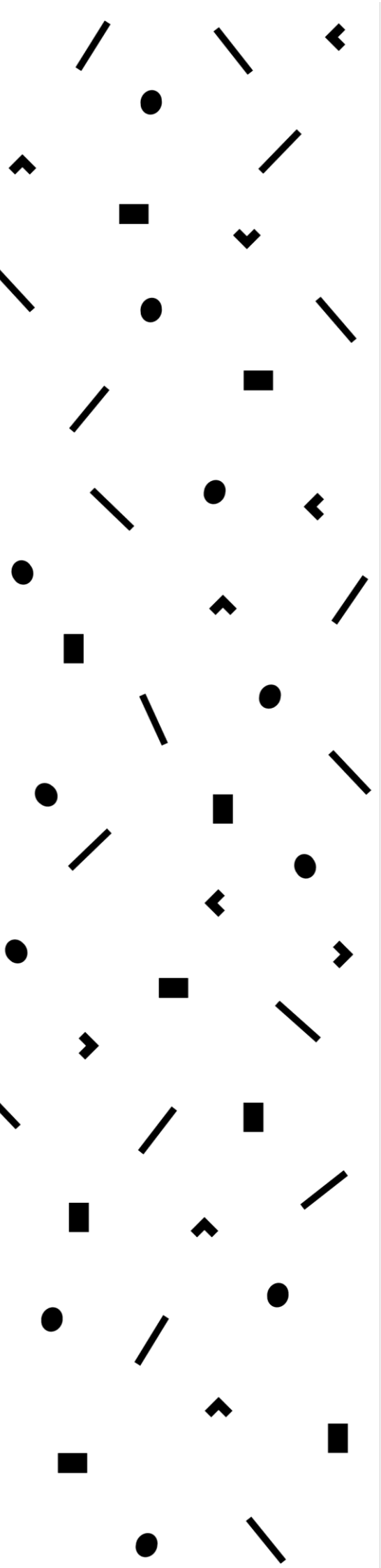


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Tietokannan suunnittelu ja toteutus Microsoft Accessilla

Iskun tarjouspyyntötietokanta

LAHDEN AMMATTIKORKEAKOULU
Tradenomi AMK
Tietojenkäsittely
Syksy 2018
Erkka Koskinen

Lahden ammattikorkeakoulu
Liiketalouden koulutusohjelma

KOSKINEN, ERKKA:

Tietokannan suunnittelu ja toteutus Microsoft Accessilla
Case: Iskun tarjouspyyntötietokanta

Tietojenkäsittelyn koulutusohjelma 51 sivua, 2 liitesivua

Syksy 2018

TIIVISTELMÄ

Opinnäytetyö tehtiin toimeksiantona Isku Interior Oy:lle. Työn tavoitteena oli kehittää prototyyppitietokanta sisäänostotiimille. Tietokantaa tultaisiin käyttämään ostotietojen seuraamiseen, ja siitä oli saatava ulos valmis tarjouspyyntö.

Tässä opinnäytetössä tutkitaan, miten relaatiotietokannan suunnittelu ja toteutus tehdään käytännössä. Aluksi kerrotaan teoriassa tietokantojen keskeisistä käsitteistä ja siitä, mitä on otettava huomioon ennen kuin tietokantaa voidaan lähteä toteuttamaan. Työssä kerrotaan myös, miten Microsoft Access -tietokannan hallintajärjestelmää käytetään.

Tutkimusosuudessa kerrotaan opinnäytetyön casena olleesta relaatiotietokantahankkeesta. Siinä käydään läpi projektin eteneminen alusta loppuun. Lisäksi tutkimuksen toteutuksen jälkeen kerrotaan, millaisia ongelmia kohdattiin ja miten ne ratkaistiin.

Lopuksi kerrotaan, mitä kannattaa ottaa huomioon relaatiotietokannan suunnittelussa ja toteuttamisessa Access-ohjelmalla. Lisäksi käydään läpi, mitä mahdollisia vaikutuksia tietokannan rakenteella on informaation käyttäytymiseen ja miten erilaiset mallit vaikuttavat päivittämiseen, hakemiseen ja lopullisen käyttämisen vaivattomuuteen.

Työn tuloksena saatiin toteutettua prototyyppitietokanta toimeksiantajayritykselle. Tietokannan käyttöliittymä ja tarjouspyynnön tulostus jäivät kuitenkin tasolle, jossa niiden käyttäminen on työlästä. Seuraava kehitysaskel olisikin näiden asioiden korjaaminen.

Asiasanat: Microsoft Access, Isku Interior Oy, tietokanta, perusavain, viiteavain

Lahti University of Applied Sciences
Faculty of Business

KOSKINEN, ERKKA:

Design and execution of database using Microsoft Access
Case: Isku request for quotation database

Bachelor's Thesis in Information technology

51 pages, 2 pages of appendices

Autumn 2018

ABSTRACT

This thesis was commissioned by Isku Interior Oy. The goal of this thesis was to create a prototype database for a team of buyers. The database was to be used for tracking information on purchases and it was meant to produce a request for quotation.

This thesis examines how a relational database is designed and executed in practice. First, the paper discusses the key concepts of databases and what kind of groundwork is required before it is possible to start implementing the database. The thesis also describes how the Access database management system is used.

The next phase describes the relational database case study. This section is focused on describing the progress, what decisions were made and why and what kind of problems were encountered.

As a result, a prototype database was created. However, user interface and the ability to print requests for quotation´s were left at a state, where their use is complicated for an average user. The next step towards improvement would be, to find a solution which would fix these issues.

Key Words: Microsoft Access, Isku Interior Oy, database, primary key, foreign key.

SANASTO

Microsoft Access: Microsoftin oma tietokannan hallintajärjestelmä.

SQL: Structure query language on ohjelmistokieli, jota tietokannat käyttävät.

DBMS: Database management system eli tietokannan hallintajärjestelmä.

Wizard: Apuohjelma, joka on suunniteltu auttamaan jonkin tehtävän suorittamisessa.

SISÄLLYS

1 JOHDANTO

Nykyään melkein kaikki tahot, jotka pitävät kirjaa suuresta määrästä tietoa, käyttävät tietokantoja. Tällainen tieto voi olla fyysisestä materiaalista, kuten tavaroista varastossa, ihmisistä tai informaatiosta, joka on olemassa paperisena versiona, mutta tallennetaan kovalevylle. Esimerkki kyseisestä informaatiosta on lisenssit. Tietokantaan tallennettu tieto on muokattavissa ja lisättävissä. (Ashe-Edmunds 2018).

Tietokantaan voidaan tehdä muutakin kuin vain säilöä tietoa. Yksinkertaisesti sitä voidaan verrata inventaarioon. Sieltä voidaan tehdä erilaisia hakuja tiedon yhtenäisyyksien paikantamiseksi. Tästä esimerkkinä päivittäistavarakaupan tietokanta, joka pitää kirjaa kaupan tuotteista. Kerran päivässä voidaan tehdä haku, joka ilmoittaa kaikki yksittäiset tuotteet, joiden parasta ennen päiväys on menossa kolmen päivän sisällä vanhaksi. Nämä tuotteet voidaan laittaa seurantaan ja alennukseen. Hauista voidaan myös tuottaa raportteja, joita yrityksen johto käyttää päätösten tekemiseen. (Ashe-Edmunds 2018).

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa tarjouspyyntötietokanta Isku Interior Oy:n ostotiimin käyttöön. Tarkoituksena oli kehittää työkalu, jolla voitaisiin helposti tehdä tarjouspyyntö esimerkiksi pdf-tiedostona. Tämän lisäksi ohjelmasta pitäisi pystyä löytämään eri hakuparametreilla muun muassa tietoa siitä, keneltä on ostettu paljon samankaltaisia tuotteita ja minkälaisista tuotteista on tehty eniten tarjouspyyntöjä. Opinnäytetyön tutkimuskysymys on, miten Microsoft Accessilla voidaan luoda tietokantasovellus, jolla seurataan ostettuja tuotteita ja josta saadaan valmis tarjouspyyntö.

Tietokannoista ja niiden suunnittelusta sekä toteutuksesta on tehty satoja opinnäytetöitä, mutta vain pieni osa niistä on keskittynyt Microsoft Accessin ympärille. Vuonna 2017 Eetu Haapala on tehnyt opinnäytetyön liittyen tutkimustietokannan luomiseen Microsoft Accessilla, mutta työssä on käyty vain pintapuolisesti läpi tietokannan luomista. Hanna Ahola on puolestaan tehnyt vuonna 2011 opinnäytetyön relaatiotietokannan

suunnittelusta ja toteutuksesta, mutta tietokannan toteutus on tehty eri järjestelmällä. Näiden perusteella tämän opinnäytetyön avulla saadaan ajantasaista tietoa Accessin käytöstä ja tietokannan luomisesta sen avulla.

Työssä on käytetty sekä elektronisia että kirjallisia lähteitä. Eniten viittauksia on tehty kirjoihin ja yleisin viittaus onkin Ari Hovin, Jouni Huotarin ja Tapio Lähdemäen kirjaan Tietokantojen suunnittelu & indeksointi (2005). Syy lähteen toistuvuuteen on sen kronologisuus ja selkeys. Kirja toimi lähes täydellisenä ohjeena tietokantojen teoriaan perusasioista, joihin tässä työssä keskitytään. Elektronisista lähteistä tärkeimmät olivat Microsoftin omat tukisivut, joissa kerrotaan Accessin käytöstä ja relaatiotietokannoista.

Tietokantoja on monia erilaisia, mutta tässä työssä on perehdytty ainoastaan relaatiomalliin, sillä työ tehtiin relaatiotietokantana. Relaatiotietokannalla tarkoitetaan tietokantaa, jossa tietoja voidaan jakaa aiheiden perusteella erillisiin tauluihin (Microsoft Corporation 2018c). Se valittiin tietokannan toteutusmuodoksi, koska se on helppokäyttöisempi kuin hierarkkiset tai verkkomalliset tietokannat (Hovi, Huotari & Lahdenmäki 2005, 7).

Aihe rajattiin pelkästään Accessin ympärille, sillä ohjelma tulisi olemaan prototyyppi, eikä sen tarvitsisi mukailla muita sovelluksia. Opinnäytetyössä kerrotaan SQL-lauseista ainoastaan kerran, koska niiden käyttö ei ole välttämätöntä Accessilla. Kielen hallitseminen tekee kuitenkin ohjelman edistyneestä käytöstä helpompaa.

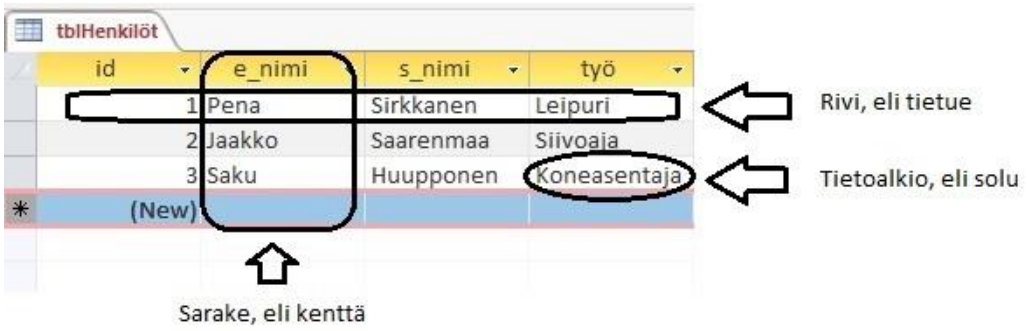
2 TIETOKANTOJEN PERUSTIETOJA

Tietokannoilla tarkoitetaan pääsääntöisesti erilaisia tallennettuja tietoja, jotka liittyvät toisiinsa loogisesti. Looginen yhteenkuuluvuus ilmenee esimerkiksi henkilöiden ja heidän kotiosoitteidensa välillä. Tietokanta, jonka tehtävä on pitää kirjaa ihmisistä, tarvitsee myös mahdollisesti tiedon heidän osoitteistaan. Tällaisia tietoja ovat esimerkiksi henkilöstörekisterit. Jokainen tietokanta on kuitenkin yksilöllinen ja niillä on oma tarkoituksensa. Näitä tietoja voidaan hallita tietokannan hallintajärjestelmillä, kuten Microsoft Accessilla (jota tässä työssä käytetään), MySQL:lla ja Oracle:lla. Tietokannan hallintajärjestelmä, eli TKHJ (Database Management System, DBMS) käyttää omaa SQL (Structured Query Language) kieltä tiedon käsittelyyn. Tietokantakielen tehtävä on lukea, päivittää ja varastoida tietoa tietokannassa. (Hovi ym. 2005, 4.)

Tietokannan hallintajärjestelmiä käytetään, koska ne takaavat informaation helpomman ja turvallisemman käsittelyn. DBMS helpottaa tiedon tallentamista ja löytämistä, sekä parantaa suorituskykyä. Jos emme käyttäisi niitä, kaikki tiedot tallennettaisiin tiedostoina, mikä hidastaisi informaation käsittelyä, sen hakemista ja veisi paljon enemmän levytilaa. Yritykselle informaatio on arvokas resurssi siinä missä henkilöstö ja materiaalitkin, koska siitä laaditaan raportteja tärkeiden päätösten tekemiseksi. (Hovi ym. 2005, 4.)

2.1 Rakenne

Tietokannan peruselementti on taulu (table). Se sisältää rivejä (row) ja sarakkeita (column). Sarakkeet ovat pystysuunnassa kulkevia elementtejä, kun taas rivit vaakasuuntaisia. Rakenteen pienin komponentti on tietoalkio (data item). Se voi sisältää yhden arvon, kuten etunimen, ID numeron tai osoitteen. (Buxton ym. 2009, 1.) Monta tietoalkiota yhdessä muodostavat rivin. Tämän rivin jokainen alkio on omalla sarakkeellaan. Kaikilla sarakkeilla on omat nimet, jotka kuvaavat niillä esiintyvää tietoa. Kuviossa 1 on yksinkertainen versio tietokannan eri osista.



KUVIO 1. Taulu ja sen osat

2.2 Yhteydet

Harva tietokanta koostuu pelkästään yhdestä taulusta. Kaiken informaation laittaminen yhteen tauluun tekisi siitä vaikeasti hallittavaa ja redundanssia, eli samaa tietoa ilmenisi useaan otteeseen. Redundanssia vältetään hyvällä rakenteella, joka saadaan aikaan suunnittelemalla tietokanta muotoon, jossa tieto esiintyy yhdessä paikassa ja on yhteydessä niihin tauluihin, joissa sitä tarvitaan. (Technopedia. 2018.)

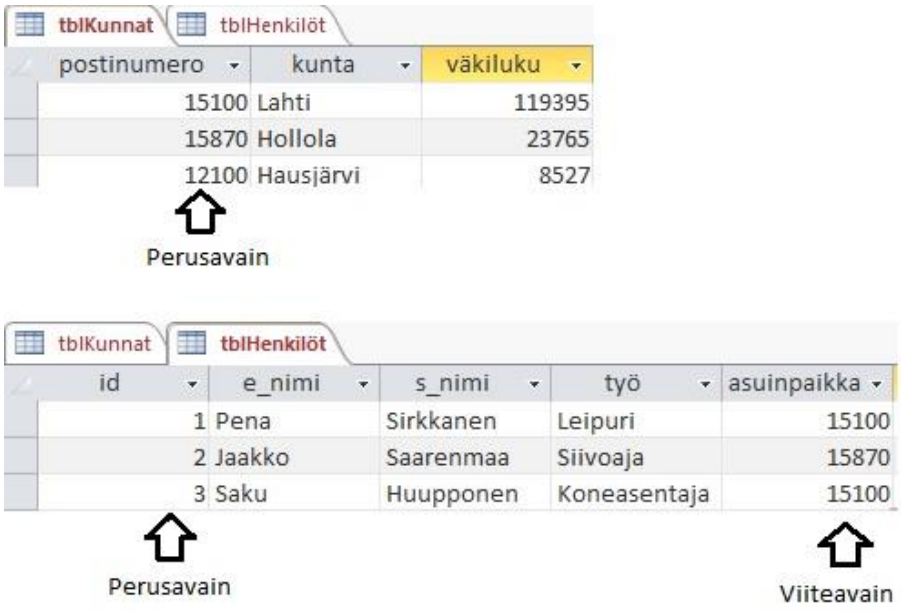
Tietokannan taulujen välisten suhteiden luomiseksi, jokaisessa taulussa on oltava perusavain/pääavain. Perusavain ja viiteavain mahdollistavat taulujen välisten suhteiden muodostumisen. (Chapple. 2018.)

2.2.1 Perus- ja viiteavaimet

Jokaiseen tauluun on syytä informaation tunnistamista varten määrittää perusavain/pääavain (primary key). Perusavain on uniikki arvo omalla sarakkeellaan eikä sen alapuolella olevilla riveillä voi olla samoja arvoja. (Hovi ym. 2005, 9.) Sen tehtävä on identifioida taulun kaikki rivit, jotta niiden suodattaminen ja yhdistäminen muihin tauluihin olisi mahdollista. Kuvion 2 taulussa voisi olla tuhansia henkilöitä ja osalla heistä voisi olla sama nimi, joten silloin sarake ”id” toimisi perusavaimena. Poikkeustapauksissa, kuten aputauluissa, perusavaimen voi kuitenkin jättää pois. Aputaululla tarkoitetaan taulua, johon tallennetaan esimerkiksi kyselyiden välituloksia (Laiho ym. 2012.) Ilman perusavainta taululle ei voi

kuitenkaan luoda lapsitaulua viite-eheytysmäärittelyin ja redundanssin tiedon kertyminen kasvaa. Lapsitaulu on taulu, jonka data on riippuvaista isätaulusta. (Hovi ym. 2005, 114-115.)

Viiteavaimilla (foreign key) yhdistetään kaksi tai useampi taulua toisiinsa. Esimerkiksi taulujen Henkilöt ja Kunnat välillä on riippuvuus. Henkilön on asuttava jossain kunnassa ja nämä tiedot on linkitettävä toisiinsa. Tällöin henkilöiden tietoja sisältävään tauluun on lisättävä viiteavaimena toimiva sarake. Tämän viiteavaimen arvo on sama, kuin siihen liitetyn taulun perusavaimen arvossa, vaikka sarake esiintyy eri nimellä. Kuviossa 2 on esimerkki viiteavainten käytöstä.



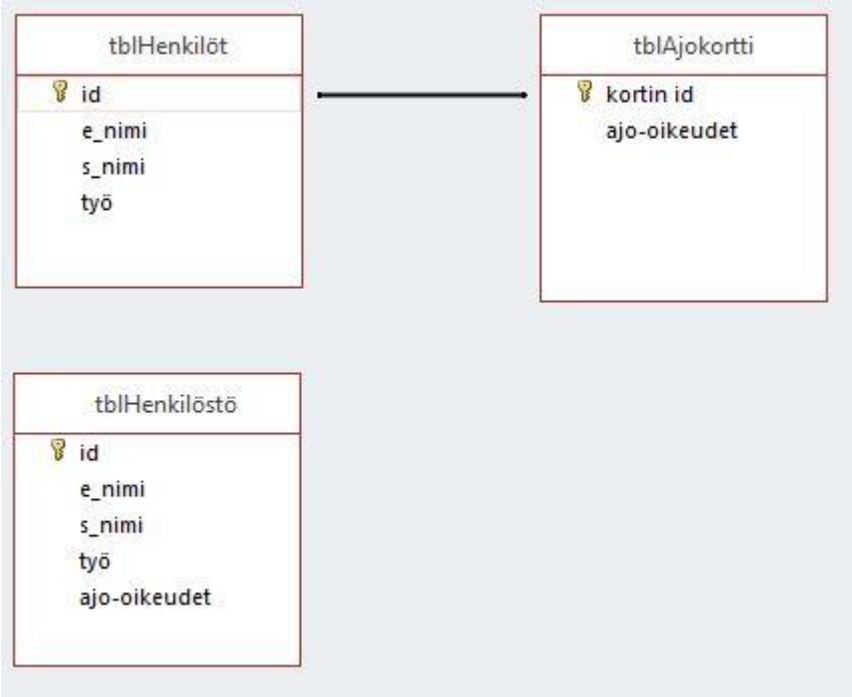
KUVIO 2. Perus, - ja viiteavaimet

2.2.2 Erilaiset yhteydet

Yhteydet taulujen välillä luodaan viite- ja perusavaimilla. Se, millainen suhde on kyseessä, vaikuttaa niiden keskinäiseen vuorovaikutukseen. Yksinkertaisimmat yhteydet ovat yhden-suhde-yhteen, yhden-suhde-moneen ja monen-suhde-moneen.

Yhden-suhde-yhteen (one-to-one) yhteyttä käytetään, kun esimerkiksi henkilön ajokortin tiedot on hajautettu omaan tauluunsa, kuten kuviossa 3

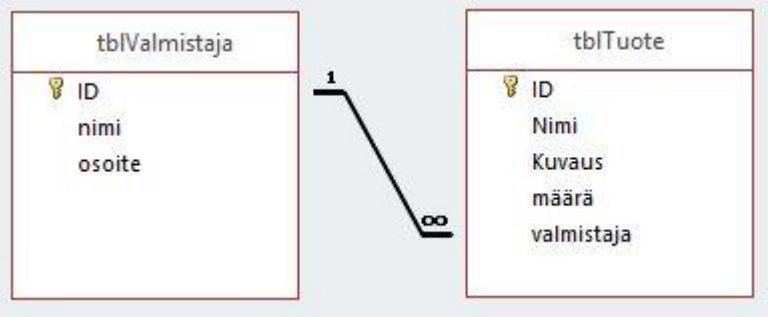
on esitetty. Henkilöllä voi olla vain yksi ajokortti ja ajokortilla voi olla vain yksi haltija. Nämä suhteet ovat harvinaisia, sillä useimmiten ne pystytään fuusioimaan yhdeksi tauluksi, mikä nopeuttaa hakujen tekoa. Vaikka tämän seurauksena tauluun jäisi tyhjiä soluja, nykyiset tietokantajärjestelmät eivät varaa turhaa levytilaa. Yhden-suhde-yhteen suhteet ovat usein lyhytnäköisen suunnittelun tulos. Näissä yhteyksissä ei yleensä ole viiteavaimia, vaan taulujen yhteys muodostuu perusavaimien välille. (Hovi ym. 2005, 67-68.) Kuviossa 3 näkyy esimerkki yhden-suhde-yhteen suhteesta ja taulusta, johon molempien tiedot on fuusioitu. Kuviosta 3 näkee myös, että Access visualisoi yhteyden pelkällä suoralla viivalla.



KUVIO 3. Yhden-suhde-yhteen

Yhden-suhde-moneen (one-to-many) on yleisin ja tärkein suhde. Sitä kutsutaan myös isä-lapsi -yhteydeksi, tai äiti-lapsi -yhteydeksi. Suhteen perusperiaate on se, että isällä voi olla monta lasta, mutta lapsella vain yksi isä. (Hovi ym. 2005, 9.) Yhteyden muodostamiseen tarvitaan viiteavain, joka sijoitetaan lapsitauluun ja joka vastaa isätaulun perusavainta (Hovi ym. 2005, 105). Käytännössä lapsitaulu voisi sisältää esimerkiksi tuotteita ja isätaulu valmistajia, kuten kuviossa 4. Kuviossa

Access esittää suhteen viivalla, jonka isätaulun päässä on 1 ja lapsitaulun päässä on ∞ merkki.



KUVIO 4. Yhden-suhde-moneen

Monen-suhde-moneen (manyt-to-many) yhteydessä myyjällä voi olla monta myytävää tuotetta ja tuotteilla voi olla monta myyjää. Suhdetta toteutettaessa taulujen väliin on luotava taulu, joka yhdistää kaksi muuta taulua. Tätä kutsutaan välikäsitteeksi eli assosiatiiviseksi käsitteeksi. (Hovi ym. 2005, 44-48.) Michael. J. Hernandez käyttää kyseisellä taululla nimitystä linkitystaulu (Hernandez 1997, 293.) Välikäsitteeseen voidaan asettaa ylimääräistä informaatiota, mikä tukee taulujen suhdetta. Esimerkiksi tuotteiden ja myyjien väliin voidaan luoda taulu, johon lisätään myynnistä kertovia ominaisuuksia. Tähän tauluun voidaan esimerkiksi lisätä myytävien tuotteiden lukumäärät. Välikäsite nimetään mahdollisimman kuvaavasti vastaamaan tapahtumaa, jota se edustaa. Tämä luo joustavuutta, sillä myyjän ei tarvitse myydä yhtään tuotetta, tai hän voi myydä kaikkia tuotteita. Sama periaate pätee tuotteisiin. Välikäsitteestä saadaan myös lista, josta nähdään, kuinka monta tuotetta on myyty ja ketkä ovat ne myyneet. Kun tiedetään tarkka myyntimäärä, voidaan pitää kirjaa tuotteiden lukumäärästä. (Hovi ym. 2005, 44-48.) Kuviosta 5 nähdään, kuinka monen-suhde-moneen koostuu oikeastaan kahdesta yhden-suhde-moneen yhteydestä. Tuotteiden ja myyjien välissä oleva myyntitaulu toimii välikäsitteenä.

KUVIO 5. Monen-suhde-moneen

2.3 Viite-eheys

Relaatiokannat perustuvat IBM:n tutkija E. F. Coddin vuonna 1970 julkaisemaan relaatiomalliin. Tietokannan eheyden suojelemiseksi hän määritteli kaksi eheyssääntöä, avaineheys (entity integrity) ja viite-eheyssäännöt (referential integrity). Tietokannan eheydellä (integrity) tarkoitetaan, että sen tiedot ovat oikein, ristiriidattomia ja vastaavat reaalimaailmaa. Eheys vaarantuu esimerkiksi, jos samoja myyjiä tallennetaan kahteen kertaan. (Hovi ym. 2005, 7-11).

Avaineheyssäännön mukaan perusavaimen arvo on pakollinen, eli se ei voi olla NULL-arvo (Hovi ym. 2005, 11). NULL ei tarkoita nollaa, vaan arvo on silloin kirjaimellisesti ei mitään (W3Schools.com 2018). Viite-eheyssäännöt kieltävät poistamasta isätaulusta tietoja, joihin viitataan lapsitaulussa. Mikäli näin tehtäisiin, lapsitauluun jäisi orporivejä. Orporivillä tarkoitetaan riviä, jolla on yhteys isäriviin, jota puolestaan ei ole olemassa (Microsoft 2018a).

Viite-eheyssääntöjä on neljä kappaletta: estosääntö (restrict), vyörytyssääntö (cascade), tyhjäyssääntö (set null) ja oletusarvosääntö (set default). Estosäännön mukaan isätaulun rivejä ei voi poistaa, mikäli niillä on yhteys lapsiriviin. Vyörytyssäännön mukaan isätaulusta poistettaessa rivi, se poistuu myös lapsitaulusta. Tyhjäyssäännön mukaan, mikäli isätaulusta poistetaan rivi, siihen yhteydessä olleet lapsirivit saavat viiteavaimen arvoksi NULL-arvon. Oletusarvosäännön mukaan, mikäli isätaulun rivi poistetaan, siihen yhteydessä olleet lapsirivit saavat ennalta määritellyn vakioarvon viiteavaimekseen. (Hovi ym. 2005, 50.)

2.4 Tietokannan suunnittelu

Tietokannan suunnittelu on tärkeimpiä ja haastavimpia tehtäviä tietokannan luontiprosessissa, ja se vie myös eniten aikaa. Suunnittelu alkaa määrittämällä tietokannan tarkoitus. Se kirjataan tehtäväselosteena mahdollisimman lyhyesti ja ytimekkäästi. Näin luodaan kiintopiste, josta voidaan varmistaa, ettei projektissa lähdetä toteuttamaan tarpeettomia tai liian monimutkaisia rakenteita. (Hernandez 1997, 81.)

Tavoitteet selvitetään asiakasyritykseen kohdistuvilla haastatteluilla. On tärkeää saada kaikkien tietokantaa käyttävien toimijoiden vaatimukset selville niin johtoportaasta, joka yleensä haluaa raportteja datasta, kuin operatiivisesta portaasta, jonka työtehtävät pyörivät datan ympärillä. Näistä vastauksista saadaan tehtävätavoitteet, jotka täytetään projektin edetessä. (Hernandez 1997, 89.)

2.4.1 Käsiteanalyysi

Käsiteanalyysillä (ER-modeling) tarkoitetaan suunnitelmaa, johon kuvataan tietokannan rakenne mahdollisimman selkeästi. Sen olemus muistuttaa ajatuskarttaa. Analyysi toimii IT-ammattilaisen ja yrityksen edustajien välisenä kommunikaatiovälineenä, mitä he työstävät ryhmätyönä. Sen tulee vastata reaalimaailmassa olevia objekteja ja tapahtumia riittävän tarkasti, jotta saadaan kuvattua tietokantaan halutut asiat. Lopputuloksena syntyy käsitemalli, jota kuvataan käsitekaaviona, eli ER-kaaviona. (Hovi ym. 2005, 32-33.) ER-kaavio on helppo piirtää paperille, mutta on olemassa suunnitteluohjelmia, joista saa tarvittaessa myös taulujen luomiseen käytettävät SQL-lauseet valmiina.

2.4.2 ER-kaavio

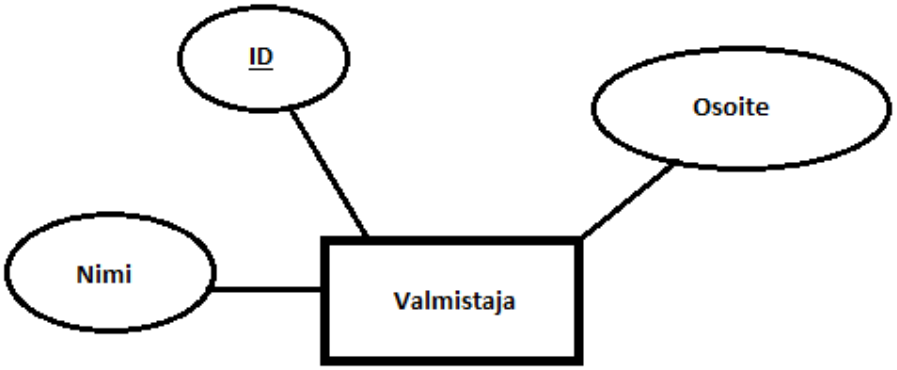
ER-kaaviossa kuvataan kolmea tekijää: tauluja, attribuutteja ja yhteyksiä. Kaavion tekemiseen on erilaisia kuvaustekniikoita eli notaatioita. Seuraavaksi esitellään Peter Chenin notaatioon perustuvaa Jeffrey D.

Ullmanin ja Widomin (1997, 128) käyttämää kuvaustekniikkaa, koska sitä käytetään myös tässä opinnäytetyössä.

Ullmannin käyttämässä kuvaustekniikassa käsite (kuvio 6) rinnastetaan tietokantaan luotavaksi tauluksi. Se siis toimii otsikkona ja taulun nimenä, joka kuvaa henkilöitä, asioita, esineitä, tai tapahtumia. ER-kaaviota mallinnettaessa pyritään keskittymään oleellisiin tietoihin, eli tietoihin, joilla on jokin syntypaikka ja ne ovat tarpeellisia. (Hovi ym. 2005, 35.) Kuviossa 6 näkyy, miten taulu on nimetty ja kuvattu nelikulmiona.

KUVIO 6. Käsite

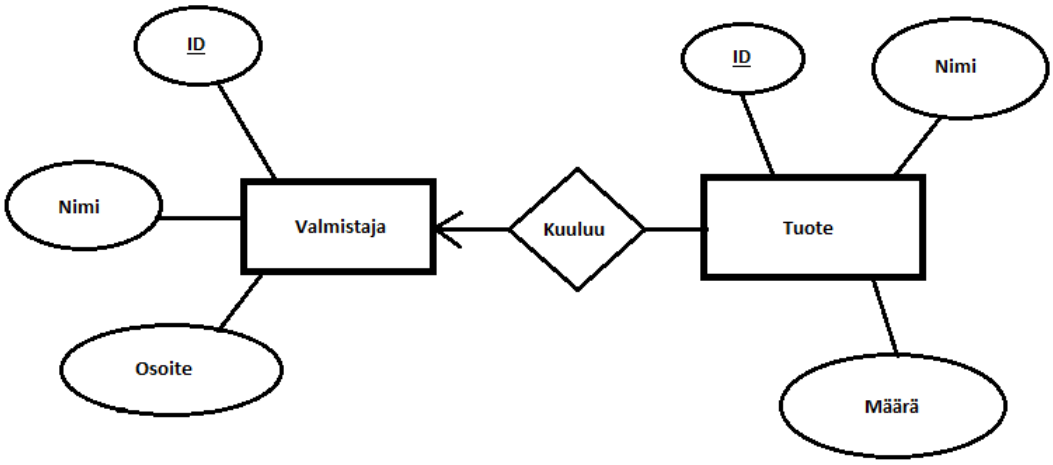
Attribuuteilla kuvataan käsitteitä, joita taulun tulee pitää sisällään. Tämä tieto on yleisesti jokin numero, kuten puhelinnumero tai lyhyt teksti, kuten nimi. ER-kaaviossa tiedot on esitetty ovaalien sisäpuolella ja niille on piirretty yhteysviiva esittämään, mihin tauluun ne kuuluvat. Taululla pitää olla jokin uniikki attribuutti, eli perusavain. (Hovi ym. 2005, 36.) Se kuvataan kaaviossa alleviivattuna. Havainnollistavana esimerkkinä: taulun käsite toimii aiheen otsikkona ja attribuutti alaotsikkona, jota verrataan sarakkeen nimeen. Kuviossa 7 on havainnollistettu, miten attribuutit sijoittuvat käsitteen ympärille.



KUVIO 7. Taulu ja sen attribuutit

Taulujen välisiä yhteyksiä kuvataan nuolilla ja viivoilla, jotka kulkevat taulusta tauluun. Niiden välillä esiintyviä suhteita voidaan kuvata vinoneliöillä. Mikäli kyseessä on monen-suhde-moneen, vinoneliö kuvastaa kolmatta taulua, jolla mahdollistetaan kyseisen suhteen luonti. Yhden-suhde-yhteen yhteyksissä nuoli osoittaa molempiin tauluihin, ja yhden-suhde-moneen yhteydessä nuoli osoittaa isätauluun.

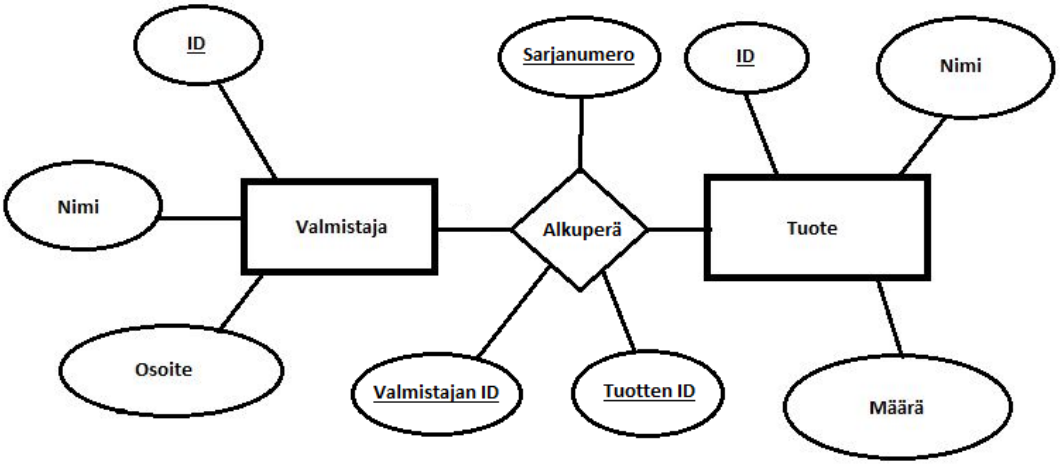
Kuviossa 8 nuoli osoittaa suhteeksi yhden suhde moneen, eli tuotteella voi olla vain yksi valmistaja ja valmistajalla voi olla monta tuotetta.



KUVIO 8. ER-malli yhden-suhde-moneen yhteydestä

Kuviossa 9 on havainnollistettu, miltä mahdollinen monen-suhde-moneen yhteys näyttäisi. Taulujen Valmistaja ja Tuote välissä on vinoneliö, josta on

muodostunut oma taulunsa. Kyseiselle taululle on annettu nimi, joka edelleen kuvastaa yhteyttä, mutta toimii myös tauluna. Kaikki taulun attribuutit ovat myös perusavaimia, joilla se yhdistetään kahteen muuhun tauluun. Sarjanumero-attribuutilla yksilöidään Alkuperä-taulun rivit. Tämä ei sinänsä ole tarpeellista näin yksinkertaisessa yhteydessä, mutta mikäli taululla Alkuperä olisi suhde vielä kolmanteen tauluun, siitä tulisi tärkeä.



KUVIO 9. ER-malli monen-suhde-moneen yhteydestä

2.4.3 Normalisointi

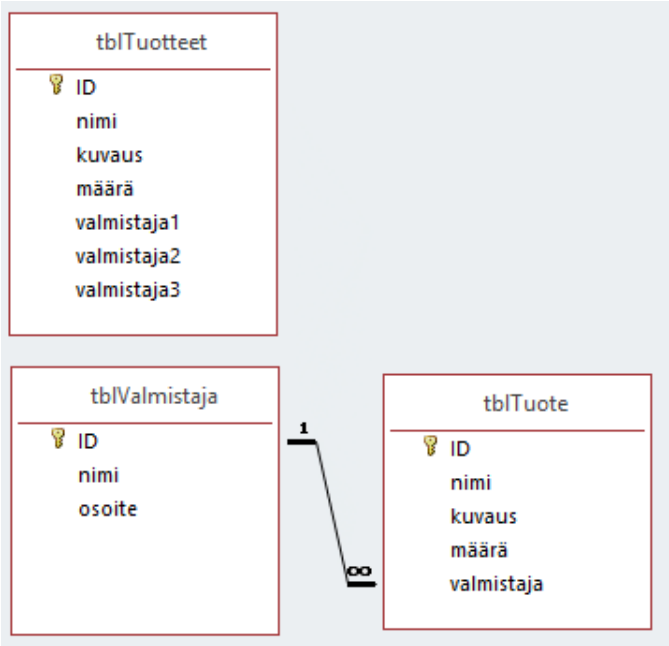
Normalisoinnilla tarkoitetaan tietokantarakenteen muuttamista parempaan muotoon, eli redundanssin minimointia, päivitystehokkuutta, yhdenmukaisuutta ja muutosjoustavuutta. (Hovi ym. 2005, 86.) Yleisesti on tärkeää miettiä, minkä tyyppinen data kuuluu minnekkin, ja mikä on sen suhde samaan tauluun tallennetun tiedon kanssa. Redundantit tiedot aiheuttavat päivitysongelmia ja vievät levytilaa. Redundanssin minimointi on tietojen toistuvuuden minimointia, mikä tarkoittaa sitä, että yksi tieto löytyy yhdestä paikasta. Jos tieto on tallennettu kahteen paikkaan, se on muutettava molemmista paikoista. Tämä vaikuttaa suoraan päivittämiseen, sillä silloin tietoja pitäisi päivittää useampaan paikkaan. (Hovi ym. 2005, 86.)

Muutosjoustavuudella tarkoitetaan mahdollisuutta laajentaa tietokantaa ilman, että se vaikuttaisi liikaa sen rakenteeseen. (Hovi ym. 2005, 86.) Tietoja, joita on tallennettu paikkaan, joihin ne eivät loogisesti kuulu,

kutsutaan epäyhtenäiseksi riippuvuussuhteeksi. Tällaiset tiedot voivat olla esimerkiksi asiakkaan tietojen tallentaminen Tilaus-tauluun, kun niiden looginen paikka olisi Asiakkaat-taulussa. (Microsoft Corporation 2017.)

Normalisoinnissa tauluille määritellään erilaisia normaalimuotoja (normal form). Yleisimmät ovat ensimmäinen, toinen ja kolmas normaalimuoto. On olemassa myös neljäs ja viides normaalimuoto, mutta niihin ei paneuduta niiden merkityksen vähäisyyden vuoksi. (Hovi ym. 2005, 86.)

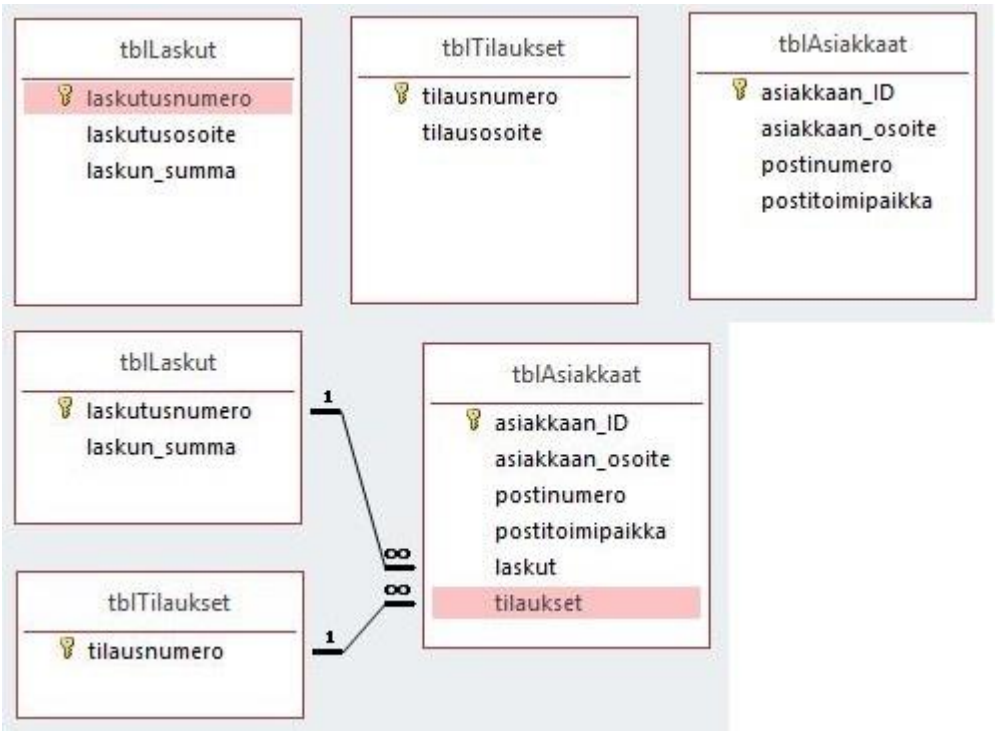
Ensimmäinen normaalimuoto toteutuu, kun yksittäisessä taulussa ei ole toistuvia ryhmiä (repeating group), ja lisäksi kaikki liittyvät tiedot on merkitty perusavaimella. Esimerkkinä tilanne, jossa halutaan jäljittää varastossa sijaitseva tuote, joka saattaa olla peräisin kahdesta eri lähteestä. Taulu voisi sisältää kentät Toimittaja1 ja Toimittaja2. Jos samaa tuotetta tilattaisiin vielä kolmannelta toimittajalta jouduttaisiin luomaan kolmas kenttä. Järkevintä olisi luoda oma taulu tavaran toimittajille, joilla olisi yhteys varaston tuotteille. (Microsoft Corporation 2017.) Kuviossa 10 taulu on normalisoitu ensimmäiseen muotoon.



KUVIO 10. Ensimmäinen normaalimuoto

Toinen normaalimuoto toteutuu, kun useita tietueita koskevien arvojen joukoille on luotu erilliset taulut, sekä taulujen välille on muodostettu suhde

viiteavaimen avulla (Microsoft Corporation 2017). Mikäli taulussa on moniosainen perusavain, eli perusavain koostuu useammasta kuin yhdestä kentästä, niin kaikkien sarakkeiden on oltava funktionaalisesti riippuvia koko perusavaimesta (Hovi ym. 2005, 91). Funktionaalinen riippuvuus (functional dependency) tarkoittaa sitä, että sarakkeen arvolla pystytään selvittämään yksittäinen arvo, joka on siihen funktionaalisesti riippuva. Esimerkkinä tästä on asiakkaan osoite. Asiakkaan osoitetta tarvitaan Tilaukset, Laskut ja Asiakkaat -tauluissa, mutta sen sijaan, että osoite tallennettaisiin kaikkiin näihin tauluihin, se kannattaa tallentaa joko Asiakkaat-tauluun tai luoda oma Osoitteet-taulu. (Microsoft Corporation 2017.) Kaikki muut taulut, jotka tarvitsevat asiakkaan osoitetta, voidaan yhdistää viiteavaimella siihen tauluun, josta tieto löytyy. Kuviossa 11 kolme alinta taulua on normalisoitu toiseen muotoon siten, että osoite esiintyy vain kerran.



KUVIO 11. Toinen normaalimuoto

Kolmas normaalimuoto toteutuu, kun taulun kaikki kentät ovat riippuvaisia pelkästään perusavaimesta. Ne eivät voi olla funktionaalisesti riippuvaisia sekä perusavaimesta, että jostain toisesta sarakkeesta. (Microsoft

Corporation 2017.) Otetaan esimerkiksi asiakkaan osoite, joka löytyy Asiakkaat-taulusta. Osoitteen lisäksi taulusta löytyy kentät Postinumero ja Postitoimipaikka. Nämä kaksi ovat riippuvaisia perusavaimen lisäksi myös toisistaan, mikä rikkoo kolmannen normaalimuodon sääntöjä. Tilanne korjataan poistamalla taulusta Postitoimipaikka sarake ja luomalla taulu, joka sisältää postinumeroita ja postitoimippaikkoja siten, että postinumero toimii kyseisen taulun perusavaimena. Kuviossa 12 taulu Asiakkaat on normalisoitu kolmanteen muotoon.

KUVIO 12. Kolmas normaalimuoto

Denormalisointi tarkoittaa normaalimuotojen sääntöjen tahallista rikkomista, jotta tiettyjen toimintojen suorittaminen helpottuu. Kun data on jakautunut useaan tauluun, se tekee hakujen luomisesta monimutkaisempaa ja hitaampaa. Tämän seurauksena on tehtävä enemmän liitoksia ja luettava enemmän tauluja, mikä saattaa myös hidastaa haun vasteaikaa. Mikäli hakujen tekemisen helppous ja nopeus ovat kriittisiä tekijöitä, on joitain tauluja denormalisoitava. (Hovi ym. 2005, 95.)

3 ACCESS JA SEN PERUSTOIMINNOT

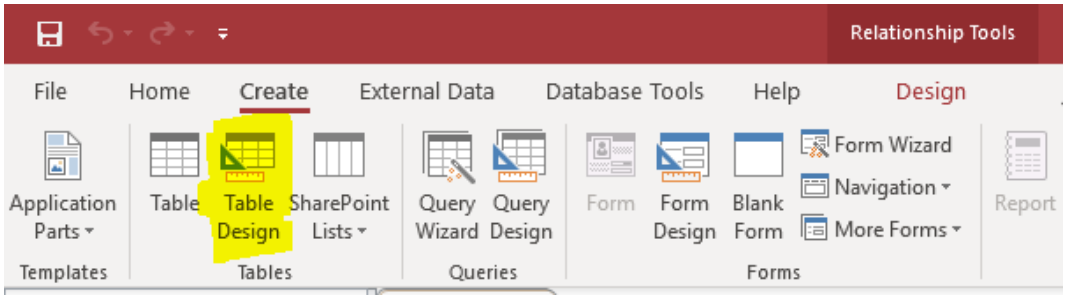
Seuraavaksi tarkastellaan, miten Microsoft Access tietokantojen käsittelyohjelmaa käytetään, sekä miten luodaan tauluja ja yhteyksiä. Access kuuluu Microsoft Office ohjelmistopakettiin. Access sopii suunnittelun tasolla tietokantojen luomiseen, ja se käy hyvin prototyyppien ja ideoiden suunnitteluun sekä visualisoimiseen. Access on suunniteltu yritysten työkaluksi erilaisten sovellusten luomiseen joko valmiiden mallien pohjalta tai kokonaan alusta. (Microsoft Office 2018b.)

Accessilla on tarpeeksi helppo luoda pieni tietokanta, jota kaksi tai useampi henkilöä voivat käyttää yhtäaikaisesti. Access ei tosin salli päällekkäistä tietojenmuokkausta samassa taulussa samaan aikaan, mutta datan tarkastelu samanaikaisesti onnistuu. Tätä varten käyttäjien on oltava samassa paikallisverkossa. Mikäli käyttäjät aikovat muokata dataa samasta paikasta samaan aikaan, voidaan valita pessimistinen lukitsemine. Se tarkoittaa mahdollisuutta valita estääkö Access toista muokkaamasta tiedostoja, ja ilmoittaako ohjelma, että toinen käyttäjä muokkaa niitä. Toinen mahdollisuus on optimistinen lukitseminen, missä käyttäjät voivat muokata tiedostoja samaan aikaan. Mikäli toinen tallentaa tietoja ensin, toinen käyttäjistä saa silloin ilmoituksen, jossa kysytään, haluaako hän hylätä työnsä vaiko jatkaa ja tallentaa tiedot edellisen käyttäjän päälle. (Sheridan 2015.)

3.1 Tietokannan ja taulujen luonti Accessilla

Accessin käyttöliittymä näyttää samalta ja toimii samalla tavalla kuin muutkin Microsoft Office ohjelmat. Sovelluksen käynnistyttyä se pyytää avaamaan valmiin tietokannan tai luomaan uuden. Mikäli käyttäjä haluaa luoda uuden, hänen on nimettävä se ja päätettävä minne se tallennetaan. Tietokannan alkava rakenne on myös mahdollista valita erilaisista valmiista pohjista.

Kun tietokanta on saatu avattua, valitaan ylävalikosta Create-sivu, josta valitaan Table Design (kuvio 13).



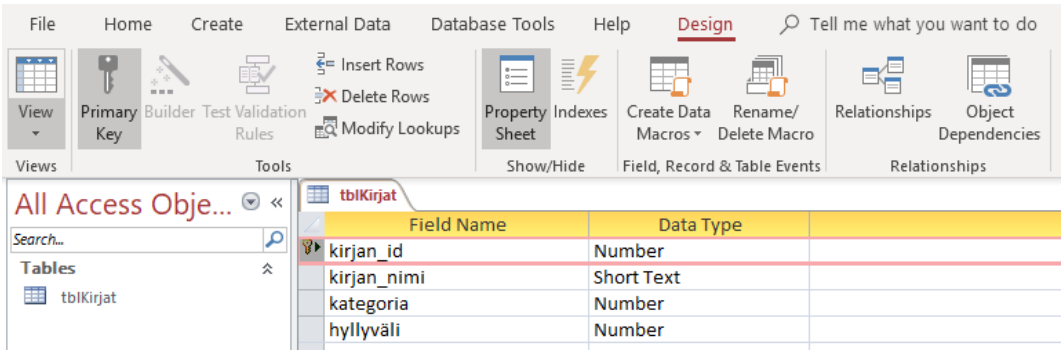
KUVIO 13. Table Design

Tämä avaa uuden taulun sekä oman työkalupalkin, josta luodaan taulun sarakkeet ja muokataan niiden ominaisuuksia (kuvio 14). Kohtaan Field Name annetaan sarakkeen nimi ja kohtaan Data Type ilmoitetaan, missä muodossa kenttä vaatii tiedon syötettävän. Kuten alla olevassa kuviossa 14 näkyy, sarake Kirjan_id on merkattu perusavaimeksi. Tämä tapahtuu valitsemalla koko rivi ja painamalla Primary Key:stä tehtäväpalkissa, tai klikkaamalla kenttää hiiren toisella näppäimellä ja valitsemalla Primary key.

On tärkeää määrittää taulun perusavain. Mikäli sitä ei ole asetettu, Access pyytää lupaa sen automaattiselle muodostamiselle tallennettaessa. Mikäli valitaan ”kyllä”, ohjelma luo uuden sarakkeen ja määrittää sen perusavaimeksi. Jos käyttäjä tallentaa tauluun dataa ja poistaa Accessin luoman sarakkeen, koko ohjelma joutuu tilaan, jossa ohjelma vaatii, ettei pääavaimen arvo voi olla null, eli ei mitään. Mikäli näin tapahtuu, ohjelman voi käynnistää uudelleen tai uudelleen luoda Accessin määrittämän kentän. Kun Access itse luo pääavaimella varustetun kentän, sen datatyyppi on automaattisesti AutoNumber, eli kun kenttään lisätään tietoa, ohjelma luo tälle sarakkeelle uuden numeron kronologisessa järjestyksessä. Täma numero on staattinen, eli sitä ei voi muuttaa. Jos taulusta poistetaan rivi, kaikki sen jälkeen tulevat rivit eivät muuta arvoaan.

Taulua tallennettaessa ohjelma pyytää nimeämään taulun. On hyvä tapa antaa taululle etuliite, kuten tbl, lyhenne sanasta table (Pittenger 2018.) Kun taulu on tallennettu, se ilmestyy vasempaan objektipalkkiin All Access Objects (kuvio 14). Palkki toimii valikkona, kun tauluja on useita. Siitä

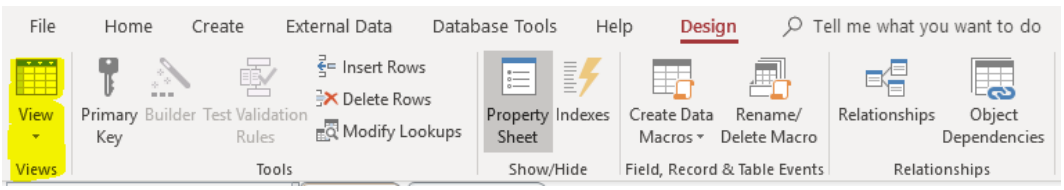
voidaan myös uudelleen nimetä, kopioida tai poistaa tauluja hiiren toisella näppäimellä.



KUVIO 14. Taulun kenttien nimeäminen ja perusavaimen määrittäminen.

Kuvion 14 kentän ”kategoria” datatyyppi on numero, koska kategorioita voi olla kymmeniä erilaisia ja niitä varten luodaan oma aputaulu. Näin Accessilla pystytään hakemaan tarvittava tieto toisesta taulusta, kun yhteys muodostetaan. Datatyyppi-valikossa on myös vaihtoehto Lookup Wizard -apuohjelma, jolla voidaan muodostaa yhteys toiseen tauluun. Jos datatyyppi on ollut väärä yhteyden muodostamiseen, Lookup Wizard vaihtaa sen oikeaan muotoon. Tämä voi johtaa kyseisessä sarakkeessa sijaitsevien tietojen häviämiseen, joten sen käytön on oltava harkittua etenkin taulussa, johon on jo tallennettu paljon tietoa.

Kun taulu on tallennettu, siihen voidaan alkaa tallentamaan dataa. Tätä varten on vaihdettava Design-näkymästä Datasheet-näkymään. Se tehdään tehtäväpalkin vasemmasta reunasta (kuvio 15). Tauluun ei kannata lisätä oikeita tietoja ennen kuin kaikki siihen liittyvät suhteet on luotu. Jos taulua on myöhemmin muutettava, tallennettu tieto voi kadota. Aputaulut, eli tytärtaulut, ovat kuitenkin poikkeus, mikäli ne sisältävät vain yhtä tietoa, kuten kirjojen kategorioita.



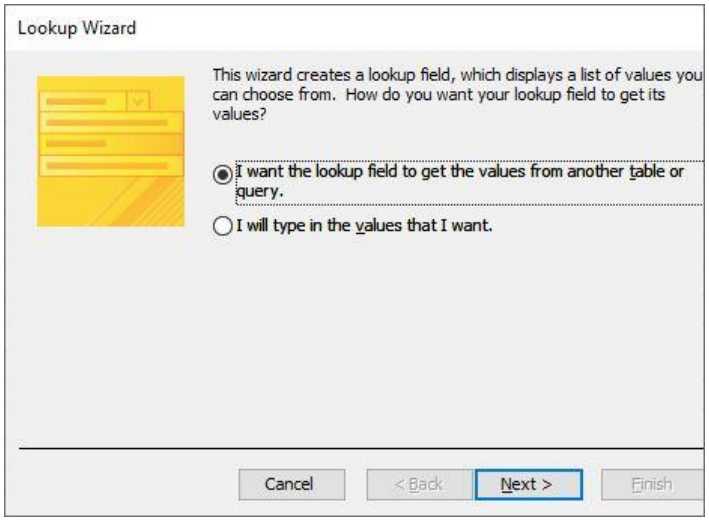
KUVIO 15. Design View ja Datasheet View

3.2 Yhteyksien luonti Accessilla

Yhteyden voi muodostaa kahdella tapaa, Lookup Wizardilla, tai Relationships-näkymästä. Edellisessä luvussa kerrotaan mistä Lookup Wizard löytyy. Seuraavaksi esitellään sarja esimerkkejä, joita varten on luotu kaksi taulua tblKirjat ja tblKategoriat, millä havainnollistetaan apuohjelman käyttöä. Kirjat-tauluun ei ole tallennettu tietoa, mutta Kategoriat-taulussa on neljä riviä.

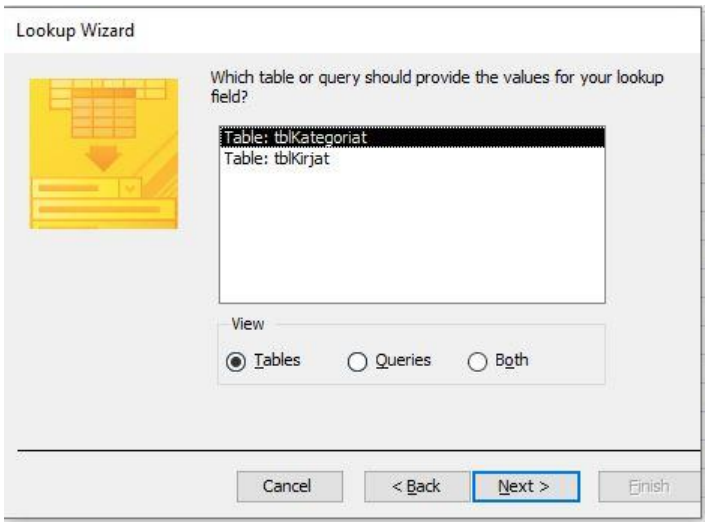
3.2.1 Yhteyksien luonti Lookup Wizardilla

Kun apuohjelma avataan (kuvio 16), ensin valitaan mistä arvoja haetaan, eli halutaanko vain luoda oma lista, vai luodaanko yhteys toiseen tauluun. Mikäli luodaan oma lista, ei synny uutta yhteyttä toiseen tauluun ja tarvittavat arvot voidaan kirjoittaa suoraan listaan. Tämä on toimiva ratkaisu, kun tiedetään ettei arvoja ole kovin montaa, eikä niitä tulla lisäämään.



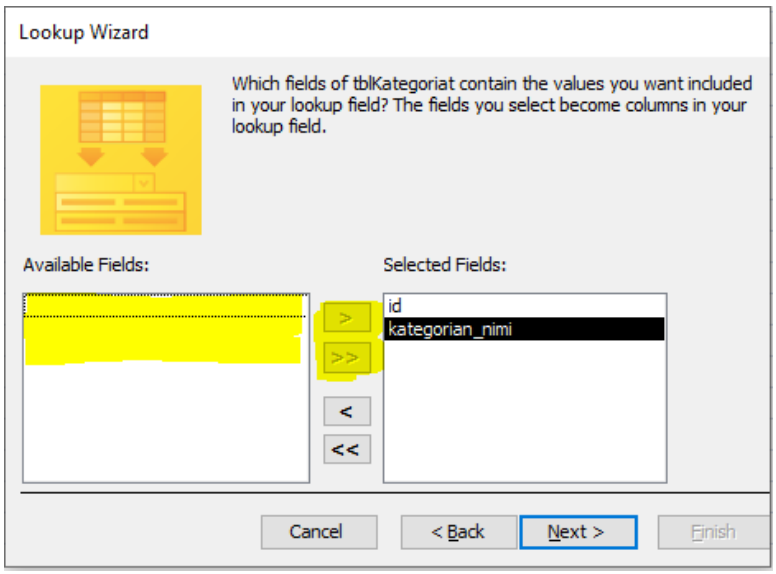
KUVIO 16. Lookup Wizard haun tapahtuessa toisesta taulusta

Seuraavaksi valitaan, mistä taulusta tieto haetaan (kuvio 17). Tässä esimerkissä se on Kategoriat taulusta.



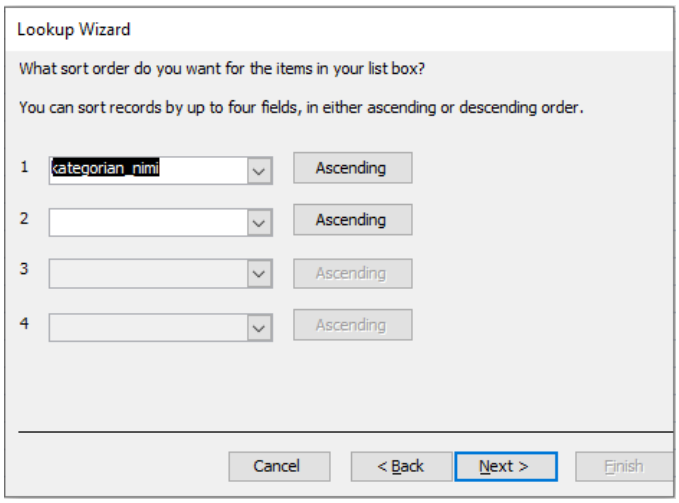
KUVIO 17. Taulun valinta

Kolmannessa kohdassa valitaan, mistä kentistä halutaan liittää tietoa. Se tapahtuu valitsemalla kenttiä Available-laatikosta ja siirtämällä ne nuolilla Selected-laatikkoon. Kuviossa 18 on valittu kaikki sarakkeet, koska taulussa niitä on vain kaksi.



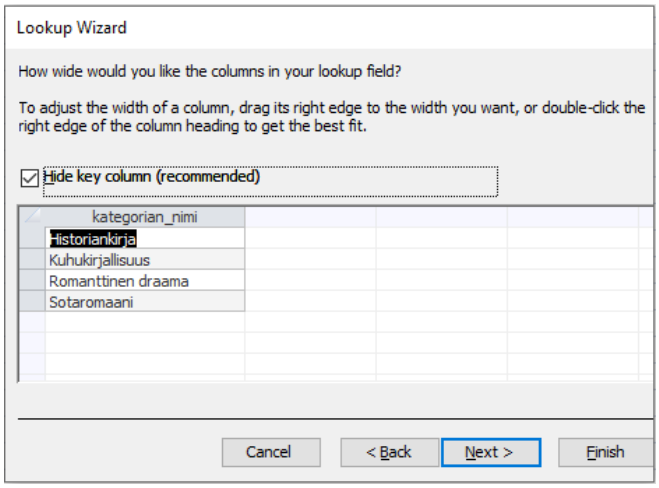
KUVIO 18. Sarakkeiden valitseminen

Neljännessä kohdassa valitaan, millaisessa järjestyksessä valittava informaatio halutaan käyttäjälle esittää. Apuohjelma antaa mahdollisuuden neljälle lajittelulle ja valinnan nousu- tai laskujohteisena. Kuviossa 19 on näkymä ikkunasta.



KUVIO 19. Järjestyksen määrittäminen

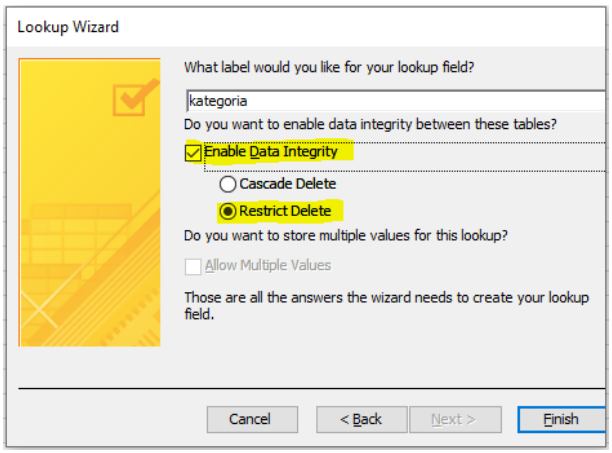
Viidennessä kohdassa määritetään listan leveys (kuvio 20). Leveyttä muutetaan nappaamalla sarakkeen oikeasta reunasta kiinni ja liikuttamalla sitä vasemmalle ja oikealle. Listasta on hyvä tehdä leveämpi kuin sen hetkinen pisin vaihtoehto, sillä aina ei voida olla varmoja lisätäänkö tauluun vielä rivi, jolla on vielä pidempi arvo. Ohjelmassa avainsarake on vakiona piilotettuna, ja mikäli siinä ole tunnistamisen kannalta tärkeää tietoa, se kannattaa myös pitää piilotettuna.



KUVIO 20. Valikon leveyden määritys

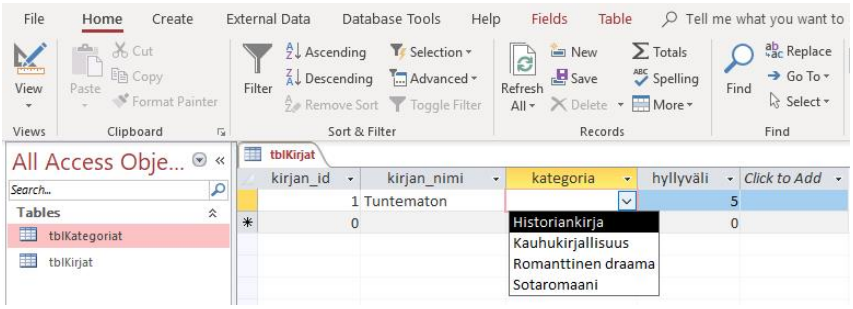
Lopuksi sarake voidaan uudelleennimetä, mikäli sille on tarvetta. Tärkeä valinta on myös, halutaanko yhteyden säilyttävän viite-eheyttä, ja toimiiko se vyörytys säännöllä (cascade), vai estosäännöllä (restrict) (kuvio 21). Tässä esimerkissä valitaan Restrict Delete, eli taulusta Kategoriat ei voida

poistaa rivejä, mikäli ne ovat kytköksissä tauluun Kirjat. Jos halutaan valita useampia arvoja yhteen soluun, Data integrity on oltava pois käytöstä.



KUVIO 21. Viite-eheys valinta

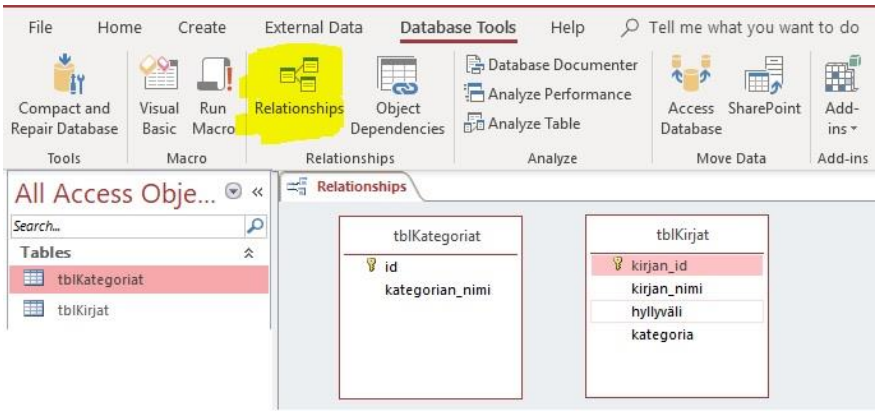
Yhteyden muodostuttua sarakkeeseen kategoriat on ilmestynyt vetovalikko, josta voidaan lisätä haluttu arvo kenttään. Apuohjelma siis muodosti yhteyden taulujen välille ja loi samalla valikon, jossa on ennalta määrättyjä arvoja. Tällä minimoidaan ihmisen tekemiä mahdollisia kirjoitusvirheitä. Valmis valikko näkyy kuviossa 22.



KUVIO 22. Lookup Wizardin luoma valikko

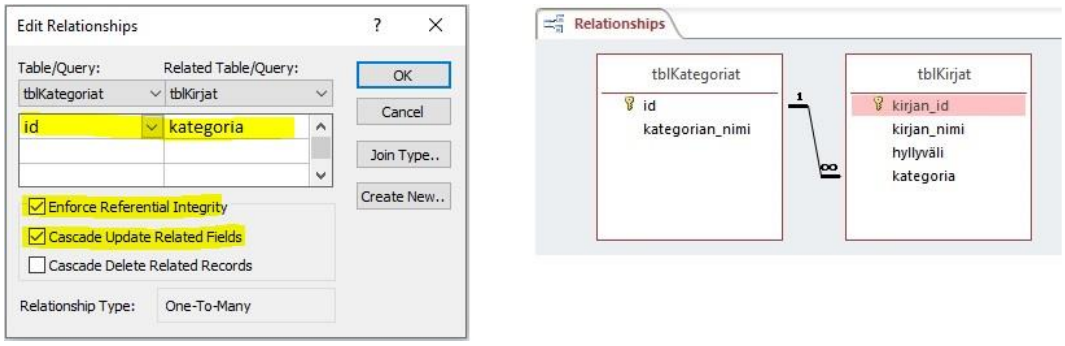
3.2.2 Yhteyksien luonti Relationships-näkymästä

Toinen tapa luoda yhteyksiä on tehdä se manuaalisesti. Työkalupalkista valitaan kategoria Database Tools, josta klikataan Relationships-painiketta. Tämä näyttää tietokannan senhetkiset taulut ja yhteydet omalla välilehdellään (kuvio 23).



KUVIO 23. Tietokannan suhteet

Saman yhteyden luomiseksi raahataan ja pudotetaan taulusta Kirjat attribuutti ”kategoria” taulun Kategoriat attribuuttiin ”id”. Näin aukeaa valikko, jossa päätetään, millainen yhteys on kyseessä. Valikossa näkyy, mitkä taulut ovat muodostamassa yhteyttä ja mihin attribuutteihin ne sidotaan. Niiden alapuolella ovat vaihtoehdot viite-eheyden säilyttämiseksi, sekä mitä viite-eheytyssääntöä yhteyden halutaan noudattavan. Vaihtoehdoista valitaan Enforce Referential Integrity, sekä Cascade Update Related Records. Näin taulusta Kategoriat ei voida poistaa rivejä, jotka ovat kytköksissä tauluun Kirjat. Painetaan Create, mikä luo yhden-suhde-moneen -yhteyden taulujen välille. Kun yhteys on luotu, se näkyy Relationships-välilehdellä taulujen välissä. Kuviossa 24 on esimerkki suhteiden määrittämisestä.



KUVIO 24. Suhteen määritys ja sen visualisointi

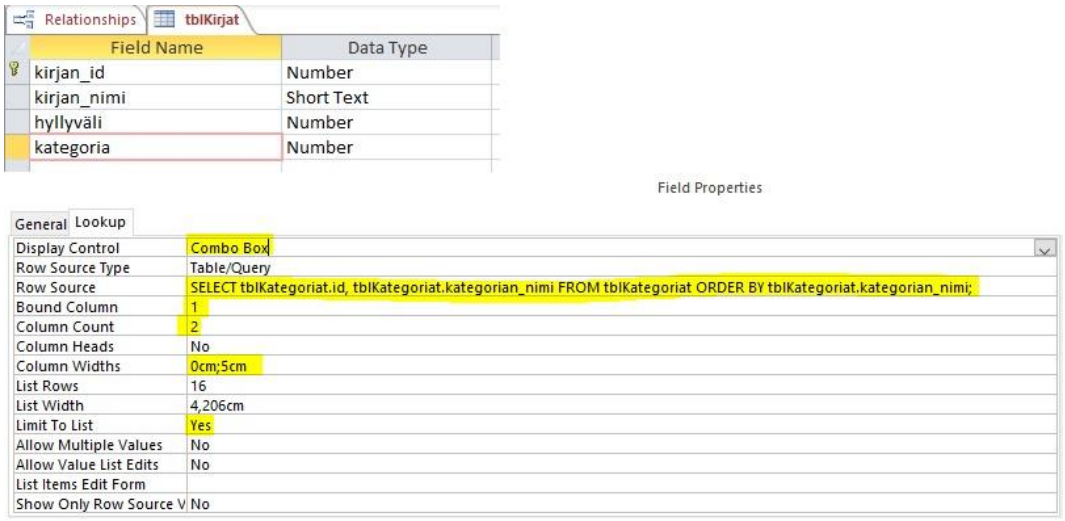
Suhde on nyt valmis, mutta taulussa Kirjat ei ole vetovalikkoa. Sarakkeelle Kategoria voidaan kirjoittaa id-arvo, joka löytyy taulusta Kategoriat. Tätä varten pitäisi tietää, mitä kategoriaa numero edustaa. Hyvässä

toteutuksessa pyritään minimoimaan pullonkaulat, joissa on mahdollisuus inhimillisen virheen syntymiselle.

Vetovalikon luomista varten on palattava taulun Kirjat Design-näkymään ja valittava Kategoria-sarake. Sivun alareunassa on Field Properties työkalu, josta valitaan Lookup-välilehti. Välilehdellä on vakiona Display Control, jossa on arvo Text Box. Tämä muutetaan Combo Box -muotoon. Se avaa listan uusia asetuksia, joita muokkaamalla saadaan aikaan vetovalikko. Seuraavaksi on listattu tärkeimmät kohdat ja miten ne vaikuttavat haun toimintaan.

- Row Sourceen kirjoitetaan SQL kysely, joka määrittää mistä, mitä ja missä järjestyksessä haku tehdään. Kyseisessä esimerkissä kysytään:
  ”SELECT tblKategoriat.id, tblKategoriat.kategorian_nimi
  FROM tblKategoriat
  ORDER BY tblKategoriat.kategorian_nimi;”
- Bound Column lukitsee valikon tiettyyn sarakkeeseen, tässä tapauksessa ensimmäiseen.
- Column Count määrittää kuinka monta saraketta otetaan huomioon, tässä tapauksessa kaksi.
- Column Widths näyttää valikon sarakkeiden leveyden kronologisessa järjestyksessä, eli jos ensimmäisen sarakkeen leveydeksi määritellään 0cm, se häviää näkyvistä. Näin tehdään tässä tapauksessa, koska ei ole tarpeellista näyttää id numeroa. Sarakkeiden leveys erotetaan toisistaan puolipisteellä. Esimerkissä kohtaan on annettu arvot; 0cm;5cm.
- Limit To List päättää, estetäänkö käyttäjää kirjoittamasta kenttään omia arvoja. Halutaan, että arvot rajoittuvat valmiisiin kategorioihin, joten rajoitetaan käyttäjän syötettä.

Kuviossa 25 on esimerkki asetuksista. Lopputuloksena on samanlainen valikko, kuin kuviossa 22.



KUVIO 25. Kategoria kentän ominaisuudet

3.3 Formien luonti

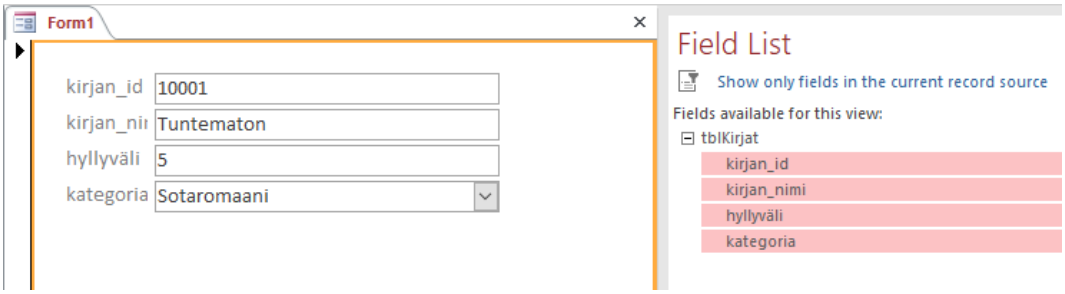
Accessin hallintajärjestelmässä on valmiina työkalu, jolla voidaan luoda käyttäjälle valmiita lomakkeita. Lomakkeet helpottavat loppukäyttäjän työtä, sillä hän ei välttämättä tiedä mitään tietokannoista tai Access-ohjelmasta. Lomakkeita voidaan tehdä rivien selailua varten, sekä uusien rivien tallentamista ja vanhojen rivien muokkaamista tai poistamista varten. Access pystyy luomaan taulusta valmiin lomakkeen, kun käyttäjä valitsee All Access Objects -kentästä sen taulun, josta lomake halutaan luoda ja painaa valikon Create kohdasta Form.

Formeihin voidaan myös lisätä makroja, jotta niillä saadaan vielä spesifimpiä ominaisuuksia. Tässä työssä käytettävät makrot ovat lähinnä yksinkertaisia ”tallenna”, ”poista” ja ”seuraava” -nappeja.

Seuraavassa esimerkissä näytetään, miten luodaan yksinkertainen lomake ilman Accessin apua, ja miten sillä selataan yksittäisiä rivejä taulussa Kirjat. Tauluun luodaan myös nappi uusien rivien tallentamista varten.

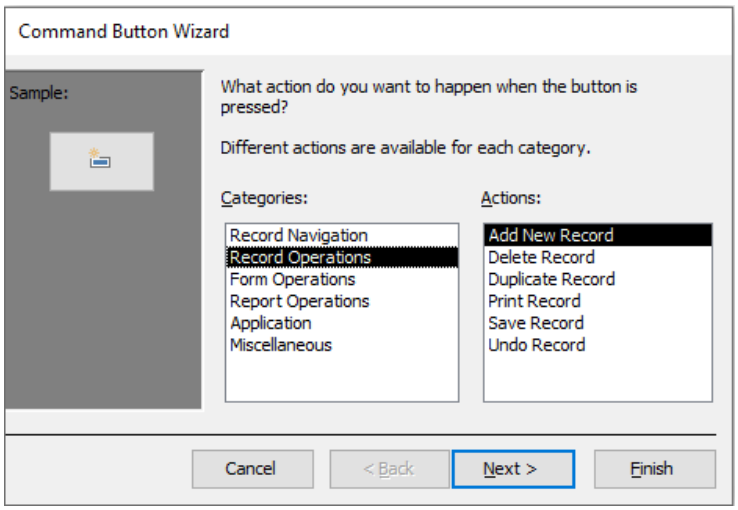
Ensimmäisenä valitaan valikosta Create kohdasta Blank Form. Tämä avaa näkymän Form Layout Tools. Oikeassa reunassa, kentässä nimeltä Field List, näkyvät tietokannan taulut. Näistä laajennetaan tblKirjat. Sen jälkeen

valitaan halutut sarakkeet ja siirretään ne klikkaamalla ja raahaamalla tyhjälle lomakkeelle (kuvio 26).



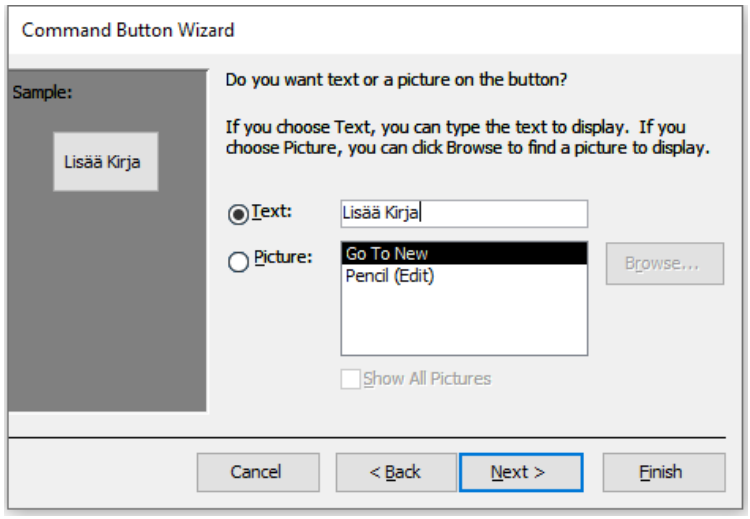
KUVIO 26. Kenttien siirtäminen lomakkeeseen

Seuraavaksi tallennetaan lomake ja nimetään se frmSelailu. Tässä etuliite frm viittaa formiin. Kuten kuviossa 26 näkyy, lomakkeen Kategoria-kohdalla on myös vetovalikko. Tämä valikko siirtyy automaattisesti formiin, kun se on kerran luotu taulujen yhteyksiä muodostettaessa. Lomake on nyt periaatteessa valmis, mutta toiminnallisuuksien lisäämiseksi valitaan valikon Design kohdasta Button, joka siirretään formiin. Tämä avaa apuohjelman, jolla voidaan määrittää mitä makro tulee tekemään. Apuohjelman ensimmäisessä ikkunassa näkyy lista eri kategorioita, jotka kuvaavat tehtävien luonnetta. Niiden oikealla puolella on lista yksittäisiä tehtäviä, jotka makro valmistuttuaan toteuttaa. Tämä ensimmäinen ikkuna on luomisen tärkein kohta, sillä siinä määritetään mitä nappi tulee tekemään. Uuden rivin luomiseksi valitaan kategorioista Record Operations ja toiminnoista Add New Record, ja painetaan seuraavaa (kuvio 27).



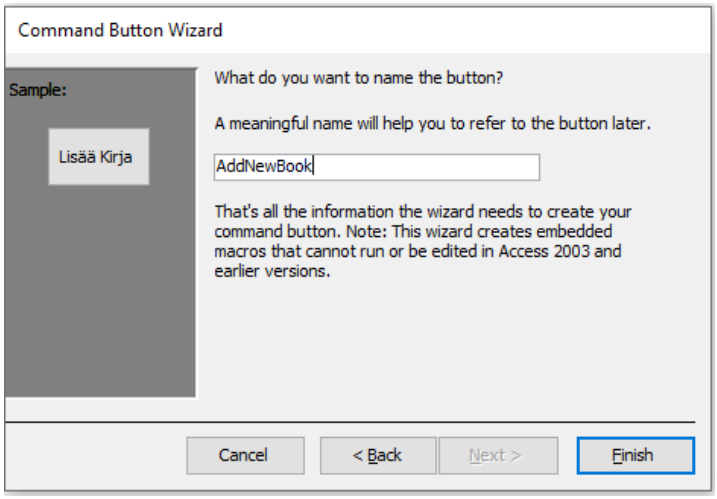
KUVIO 27. Apuohjelman näkymä

Seuraavassa näkymässä valitaan napin ulkoasu. Valittavissa on kuvia tai teksti. Valitaan teksti ja kirjoitetaan Lisää Kirja vakioarvon päälle (kuvio 28). Teksti voi olla mikä tahansa, mikä kuvaa napin toimintaa käyttäjälle selkeästi.



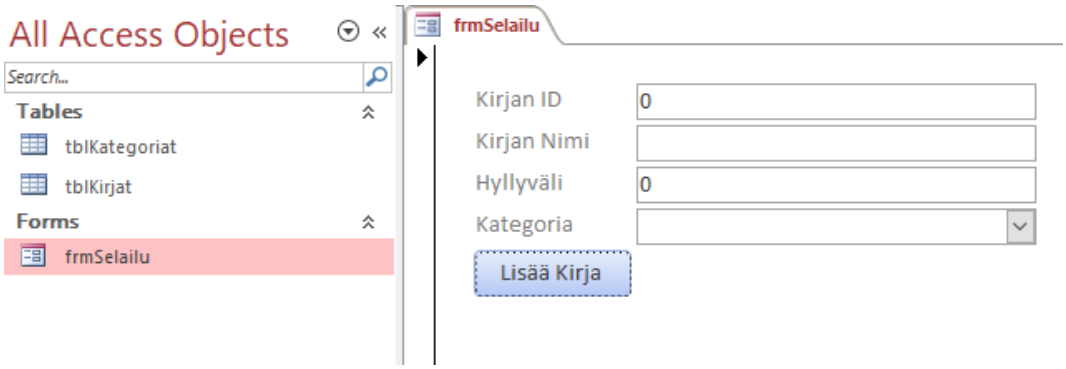
KUVIO 28. Napin ulkoasun valinta

Viimeisessä kohdassa nimetään nappi. Nappi kannattaa nimetä tarpeeksi yksilöivästi. Mikäli tulevaisuudessa luodaan makro, jonka toiminnallisuus vaikuttaa suoraan tähän makroon, löydetään se helpommin. Tässä esimerkissä nappi nimetään AddNewBook (kuvio 29).



KUVIO 29. Napin nimeäminen

Kun nappi on luotu, se näkyy lomakkeessa, ja sitä painamalla lomake näyttää tyhjän rivin (kuvio 30). Tähän voidaan nyt kirjoittaa tarvittavat tiedot kirjasta. Aina kun painetaan nappia Lisää Kirja, lomake antaa uuden tyhjän sivun. Seuraava vaihe olisi lisätä selaus, poista ja tallenna -napit, sekä tehdä siitä käyttäjäystävällisen näköinen.



KUVIO 30. Valmis lomake

Fonttien, värien ja visuaalisten efektien lisääminen ja vaihtaminen tapahtuu Form Layout Tools valikosta. Mikäli halutaan hienosäätää yksittäisten kenttien toiminnallisuutta ja ulkonäköä, avataan Design-kohdasta Property Sheet. Property Sheet sisältää kaikki mittoihin ja ulkonäköön liittyvät asetukset. Sieltä pystyy myös vaikuttamaan yksittäisten kenttien hakufunktioihin.

4 TIETOKANNAN SUUNNITTELU JA TOTEUTUS

Tässä osiossa kerrotaan, miten tietokantasovellus suunniteltiin, toteutettiin ja miksi tiettyjä ratkaisuja tehtiin. Se on jaettu viiteen osaan. Suunnitteluvaiheeseen, toteutusvaiheeseen, testaukseen ja käyttöliittymään, rakenteelliseen lisäilyyn, sekä lopullisen käyttöliittymän luomiseen. Vaikka ne ovat tässä tekstissä eroteltu, ne eivät olleet toteutettaessa selkeästi erillään. Jos työstä tekisi aikajanan, kaikki vaiheet olisivat lomittain keskenään.

Suunnittelu alkoi jo ennen kuin oli sovittu, että opinnäytetyö tehtäisiin tietokannan suunnittelusta Microsoft Accessilla. Idea tietokannan tekemiseen Iskun ostotiimille tarjouspyyntöjä varten saatiin tiimin esimieheltä. Ideoinnin jälkeen alettiin suunnitteleman yhdessä tiimin kanssa tietokannan sisältöä. Suunnittelun tueksi saatiin mallikappaleita ostajien tekemistä tarjouspyynnöistä.

4.1 Johdatus suunnitteluun ja toteutukseen

Suunnittelutyö tehtiin suurimmaksi osaksi paperille. Lisäksi suunnittelussa käytettiin apuna myös DBDesigner.net nimistä tietokatojen suunnitteluun tarkoitettua työkalua ja kuvioden tekemiseen Edward Max nimistä illustraatiotyökalua. Itse toteutus tehtiin Microsoft Access -tietokantatyökalulla. Tietokannalta haluttiin kahta pääominaisuutta, toiminnallisuutta ja sisällöllisyyttä.

Toiminnallisuuteen kuului kolme perusominaisuutta. Ensimmäinen ominaisuus oli luoda tarjouspyyntö suoraan tietokannasta. Tarkoituksena oli siis saada tarjouspyyntö tulostettua raporttina sellaisessa muodossa, jossa jokainen tuote olisi eritelty omalle rivilleen ja siinä olisi mukana Iskun määrittämät ostoehdot. Lisäksi kuvien liittämisen tarjouspyyntöön täytyi olla mahdollista joko suoraan kuvatiedostoina tai linkkeinä näihin kuviin.

Toinen ominaisuus oli mahdollisuus hakea tuotteita seuraavien kysymysten perusteella:

- Millaisia tuotteita on kysytty / ostettu, ja mihin projekteihin ne on hankittu?
- Millaiset tuotteet esiintyvät usein ja millaisia on saatu myytyä?

Tietojen perusteella olisi mahdollista tukea Iskun kehitystä ja hinnoittelua.

Kolmas ominaisuus oli hakea tietyn ominaisuuden omaavia tuotteita, kuten esimerkiksi työtuolia mustalla muoviristikolla ja käsinojilla. Näillä hakuparametreilla pitäisi löytää kaikki kyseisellä ominaisuudella varustetut tuotteet ja tiedot siitä, miltä toimittajalta niitä on pyydetty ja kuinka monta kappaletta.

Haun toteuttamista varten piti päättää mitä sisältöä tarvittaisiin ja selvittää, miten se jaettaisiin. Sisällöllisyys jaettiin kolmeen pääosaan, tarjouspyyntö-tasoon, projekti-tasoon ja tarjouspyyntörivi-tasoon. Tarjouspyyntö-taso pitäisi sisällään

- tavarantoimittajat
- projektien nimet
- päivämäärät
- tarjouksien voimassaoloajat
- toimitusajat.

Toinen taso, eli projekti-taso pitää sisällään

- kohteet
- vastuumyyjät
- asiakastyypit.

Kolmas taso, eli tarjouspyyntörivi-taso sisältää

- tuoteryhmät
- mallit
- mallin tarkenteet
- määrät
- mitat

- hinnat
- erilaiset ominaisuudet, kuten esimerkiksi
    - jalkamallit
    - jalan materiaalit
    - jalan värit
    - käsinojat kyllä/ei
    - verhoilutyypit
    - jne.

Kun tietokanta itsessään oli saatu toteutettua, eli sen taulut ja kaikki yhteydet olivat valmiina, aloitettiin testaaminen. Testaaminen tapahtui lisäämällä tauluihin tietoa ja kokeilemalla, miten yhteydet toimivat. Samalla aloitettiin käyttöliittymän luominen. Käyttöliittymällä tarkoitetaan sitä osaa, minkä loppukäyttäjä näkee ja minkä kanssa hän on tekemisissä.

4.2 Suunnitteluvaihe

Tietokannan suunnittelu oli koko tehtävän haastavin osuus, ja vaikka se esitellään kronologisesti ensimmäisenä, ei suunnittelu päättynyt missään vaiheessa tietokannan toteutuksessa. Vielä testausvaiheessakin käytiin eri skenaarioita läpi ja pohdittiin, miten joitakin asioita saataisiin tehtyä paremmin.

Suunnittelu alkoi listaamalla mitä toiminnallisuuksia tietokantaan haluttiin ja mitä tietoa varastoitaisiin. Nämä tiedot esiteltiin aiemmin johdannossa. Aluksi piirrettiin erilaisia malleja, koska tavoitteena oli saada järkevä yleiskuva koko tietokannasta. Lopuksi päätettäisiin, missä olisi parhaat ominaisuudet juuri tätä tietokantaa varten. Samalla kun tehtiin kaavioita, harjoiteltiin Access ohjelman käyttöä ja vietiin ideoita jo demotasolle. Microsoftin Accessin käyttö oli aluksi haastavaa, sillä sitä ei aiemmin käytetty. Tässä työssä esitellään vain ne luonnokset, joilla oli jokin vaikutus lopulliseen malliin. Tekstissä taulujen nimet ovat fonttia Italic selkeyttämistä varten.

4.2.1 Kokonaisuuden hahmottaminen

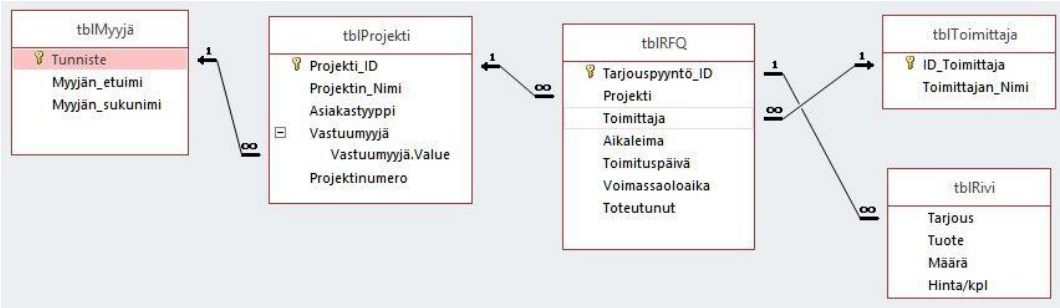
Ensimmäisissä kaavioissa (kuvio 31) yritettiin selvittää kolmea asiaa. Ensimmäisenä selvitettiin, mikä tieto olisi järkevintä sijoittaa mihinkin, kuten tarvitsisivatko myyjät kokonaan oman taulunsa, vai olisivatko ne osana projektitaulua. Toinen selvitettävä asia oli normalisointi, eli miten saada tieto mahdollisimman järkevään muotoon ja minimoida informaation toistuvuus, eli redundanssi. Kolmas pohdittava asia olivat yhteydet, eli mitkä taulut oli yhdistettävä toisiinsa, jotta tiettyjen attribuuttien hakeminen onnistuisi valmiissa tietokannassa.

Ensimmäisten luonnosten valmistuttua huomattiin kaikissa kaavioissa sama ongelma. Niissä tauluihin syntyisi rivejä, jotka sisältävät vain osan koko taulun sarakekapasiteetista. Tämä haluttiin minimoida, koska se veisi turhaa tilaa. Tietokannan käyttöönoton jälkeen siihen tallennettaisiin paljon erilaisia tuotteita, jotka kuuluisivat eri tuoteryhmiin. Esimerkiksi tuoleilla ja pöydillä on erilaisia uniikkeja ominaisuuksia, joita vain pöydillä ja tuoleilla on, kuten pöydän kansi eri variaatioineen. Tämän lisäksi se aiheuttaisi toisen ongelman. Käyttäjän syöttäessä dataa on oltava rajoituksia, eli mitä tahansa tietoa ei voi syöttää. Suurimman osa tallennettavasta informaatiosta oli siis oltava valmiiksi tehdyissä vetovalikoissa. Tämä toiminto haluttiin juuri siksi, että myöhemmin hakuja tekemällä löydettäisiin kaikki tuotteet, joissa on samoja ominaisuuksia. Mikäli loppukäyttäjä voisi itse kirjoittaa tauluihin tiedon, jäisi tilaa virheille, mikä johtaisi siihen, että haut antaisivat väärää tietoa. Tuotteiden ominaisuuksille oli siis keksittävä jokin toinen lähestymistapa.

Tässä kohtaa oltiin jo tyytyväisiä projekti-tasoon ja se pysyikin muuttumattomana loppuun asti. Taulut *projekti* ja *myyjä* eroteltiin, sekä niiden väliseksi yhteydeksi valittiin monen-suhde-moneen. Tämä johtui siitä, että myyjällä voi olla monta projektia ja projektilla voi samaan aikaan olla monta myyjää. Myös tarjouspyyntötaso oli jo selvillä. Siihen kuuluisi taulu *tarjouspyyntö* ja taulu *toimittajat*. Kuva 31 on aikainen luonnos, joten taulujen väliset suhteet eivät ole oikein, eikä kaikkia mukaan tulevia attribuutteja ole vielä päätetty.

KUVIO 31. Ensimmäinen ER-kaavio

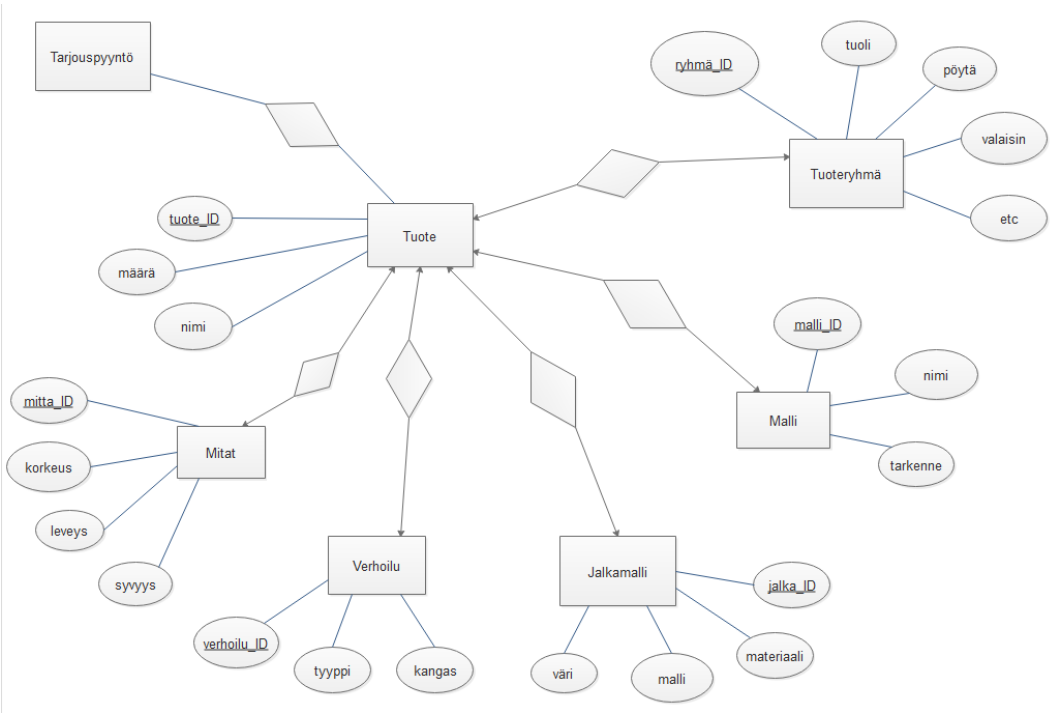
Tämän jälkeen ei enää suunniteltu koko tietokantaa, vaan vain osia siitä. Se jaettiin nyt myös suunnittelussa kolmeen aikaisemmin määriteltyyn kokonaisuuteen. Alettiin myös käyttämään lyhenteitä kahdesta viimeisestä tasosta tekstin vähentämiseksi. Tarjouspyyntö esiintyy RFQ:na, joka on lyhenne request for quotation:ista ja tarjouspyyntöriviä pelkkänä rivinä. Joissakin kaavioissa ei ole riviä laisinkaan, sillä aluksi se yritettiin korvata nimellä tuote. Lopulta tultiin kuitenkin johtopäätöksen, että molemmat nimet tarvitaan taulujen nimiksi, mutta eri rooleihin. Kuviossa 32 projektitaso ja tarjouspyyntötaso ovat valmiina.



KUVIO 32: Projekti-taso ja Tarjouspyyntö-taso

4.2.2 Tarjouspyyntörivin suunnittelu

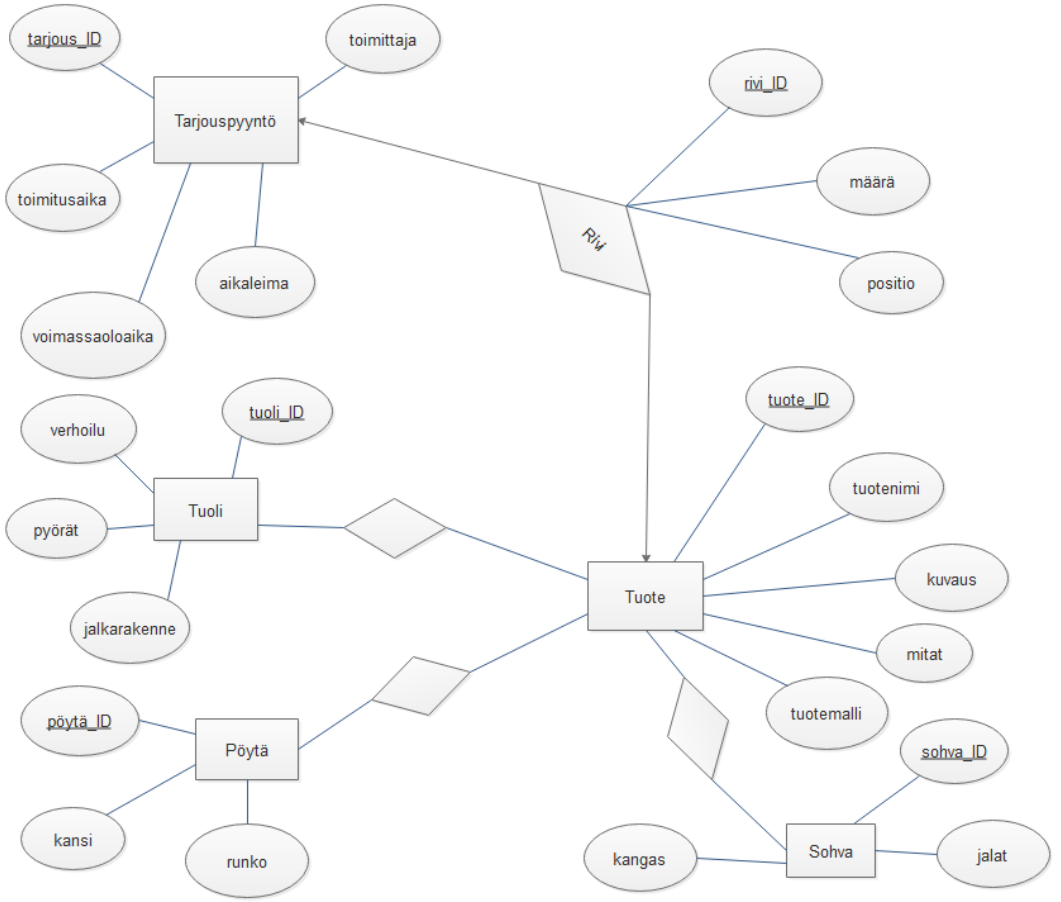
Seuraavaksi keskityttiin rivi-tasoon, sillä ominaisuudet oli pilkottava jotenkin järkevästi omiin tauluihinsa. Sen olisi tapahduttava siten, ettei syntyisi redundanssia tietoa ja ominaisuudet saataisiin valmiiksi määriteltyä käyttäjälle. Tuote hajautettiin ensin ominaisuusryhmiin, jotka sisältäisivät erilaisia ominaisuuksia (kuvio 33).



KUVIO 33. Tarjouspyyntörivin ensimmäinen er-kaavio

Sen jälkeen syntyi toinen version, jossa taulu *tuote* ja taulu *tarjouspyyntö* pitivät välissään taulua *rivi*. Tämä yksinkertaisi asioita, sillä nyt pystyttiin säätelemään tarjouspyynnöllä olevien tuotteiden määriä. Tämä johtaisi siihen, että nyt suuri määrä rivejä kasautuisi tauluun *rivi*. Sieltä voitaisiin

hallita tarjouspyynnölle kerättäviä tuotteita ja niiden määriä. Se tarkoittaisi sitä, että samoja tuotteita ei tarvitsisi perustaa aina uudestaan tarjouspyyntöä lisättäessä. Kokeiltiin myös ominaisuustyyppien hajauttamista tuotekategorioittain, mutta siitä luovuttiin nopeasti. Tuotekategorioissa oli se ongelma, että tauluun *tuote* pitäisi tehdä ylimääräisiä sarakkeita, joista vain osaa käytettäisiin. Todettiin myös paremmaksi ideaksi, että tuotekategoriat kuuluisivat omaan tauluunsa ja näin olisivat vain yksi ominaisuustyypeistä. Kuviossa 34 suhteet eivät vastaa lopullista rakennetta.



KUVIO 34. Toinen versio tarjouspyyntörivi-tasosta

Lopulta päätettiin ottaa molemmista versioista hyvältä vaikuttavat ominaisuudet. Ensimmäisestä versiosta otettiin ominaisuusryhmät ja toisesta versiosta taulujen *tarjouspyyntö* ja *tuote* välinen monen-suhde-moneen yhteys, jossa välissä oli taulu *rivi*. Lopputulokseksi kehittyi monihaarainen rakenne, jota kutsutaan lumihiutalemalliksi (Hovi ym. 2005, 139-140.)

Vaikka suunnitelma alkoi jo valmistua, tuotteiden mitat eivät sopineet oikein minnekään. Käytiin läpi kolme eri skenaariota. Ensimmäinen vaihtoehto oli, että niille tehtäisiin kokonaan oman taulu, johon olisi tallennettu eri mittoja. Tämä johtaisi siihen, että jokainen mahdollinen koko tallennettaisiin tähän tauluun ja siitä tulisi lopulta suuri ja työläs niin tietokannan ylläpitäjälle, kuin sen käyttäjällekin. Toinen vaihtoehto olisi lisätä mitat ominaisuuksiin. Esimerkiksi pöydän ominaisuuksiin kuuluisi kohta, jossa kerrotaan sen mitat. Kolmantena vaihtoehtona olisi, että tuotteen mitat kuulusivat itse tuotteeseen. Aina kun uusi tuote määriteltäisiin, sille olisi annettava myös mitat. Tämä toimisi erinomaisesti muuten, mutta samoista tuotteista on lähes aina monta eri kokoista versiota. Tämä tarkoittaisi sitä, että tuote pitäisi perustaa monta kertaa eri kokojen mukaan saamiseksi. Mielipidettä asiaan kysyttiin myös toimeksiantoyrityksestä, josta todettiin, että Iskun muiden järjestelmien mukaan, mikäli tuotteen mitat muuttuvat, niin silloin sitä pidetään järjestelmätasolla omana tuotteenaan, eikä vain erilaisena konfiguraationa. Järkevin ratkaisu olisi siis lisätä mitat tuotteen omaksi attribuutiksi. Samat tuotteet, joilla olisi eriävät koot, pitäisi vain nimetä omiksi tuotteikseen, jotta ne voitaisiin erotella mahdollisimman vaivattomasti, kun tuotteita lisättäisiin tarjouspyynnölle.

4.3 Toteutus

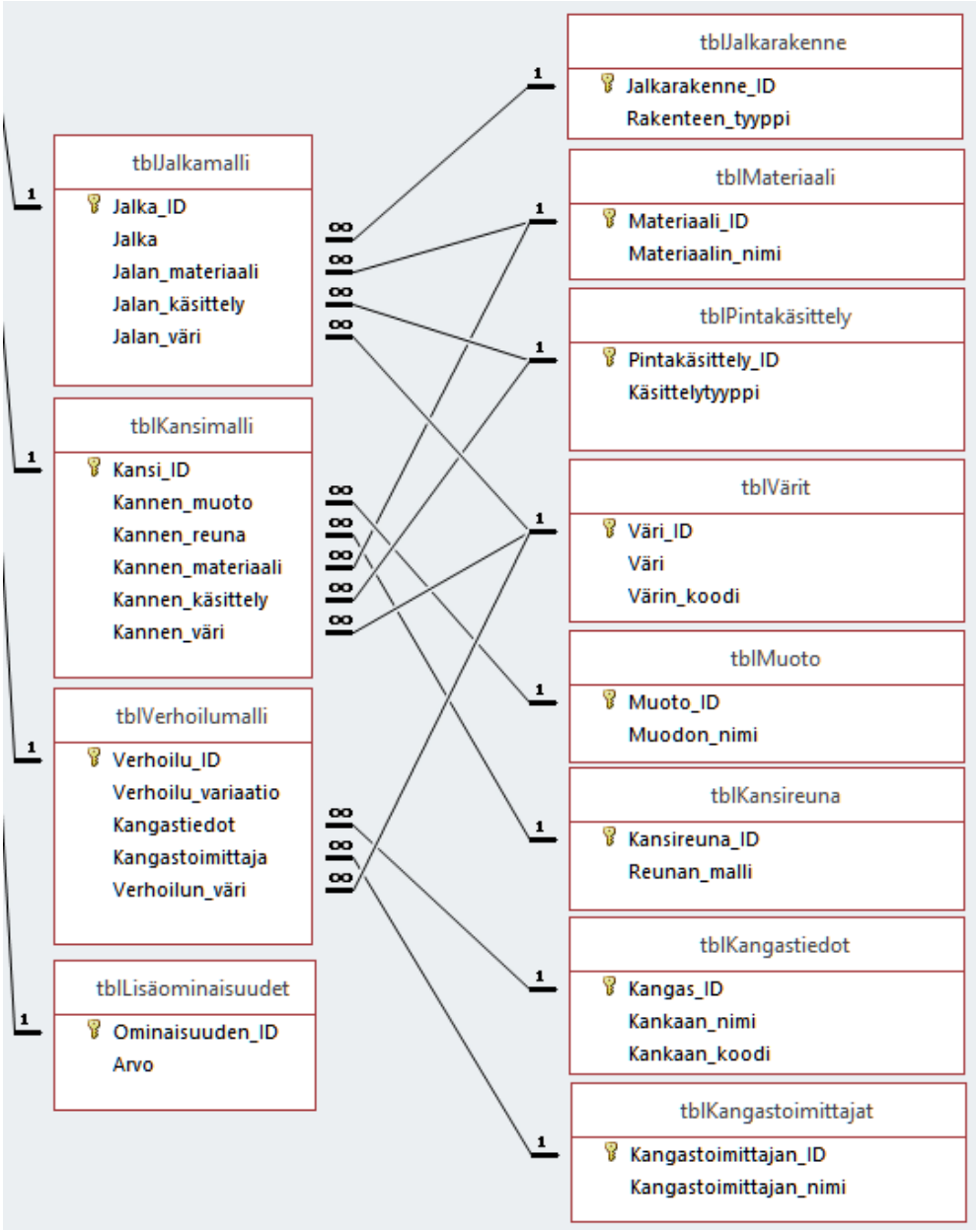
Toteuttaminen alkoi selvittämällä, miten Access tietokantatyökalua käytettiin. Samalla kun tehtiin erilaisia malleja paperille, yritettiin tuottaa niitä myös Accessiin. Alku onnistui helposti, sillä projekti-tason ja tarjouspyyntö-tason pystyi kopioimaan suoraan jo piirretyistä malleista. Suurin ongelma oli tarjouspyyntörivi-taso, koska ohjelman käyttöä ei hallittu vielä täysin. Usein lähdettiin kehittämään tietokantaa johonkin suuntaan, mutta sitten huomattiin, että yhteydet eivät toimineet niin kuin oli suunniteltu. Tajuttiin myös, että piirretyissä ER-kaaviossa oli ongelma. Kaavioissa rivi-tasolla jokainen ellipsiin nimetty ominaisuus olisi tehtävä omanaan taulunaan, mikäli niiden haluaisi toimivan vetovalikkoina. Accessissa pystyy kyllä luomaan vetovalikoita taulujen sisälle, mutta tämä

johtaa siihen, että arvojen lisääminen näihin valikkoihin hankaloituu. Arvoja pystyisi lisäämään taulun Design Viewistä, mutta loppukäyttäjä ei pääsisi siihen käsiksi. Aina kun tarvittaisiin uusi arvo, pitäisi ottaa yhteys ylläpitäjään. Samalla työ keskeytyisi ja siitä tulisi liian hidasta. Helpoin tapa lisätä listaan puuttuvia ominaisuuksia oli luoda siitä kokonaan oma taulu ja käyttää tätä taulua listan lähteenä. Näin käyttöliittymään voitaisiin vain asettaa kohta, johon käyttäjä lisää puuttuvan arvon ja päivittää lomakkeen.

Tuotteiden värit piti myös määritellä siten, että kaksiväriset tuotteet pystyttäisiin erottelemaan. Mikäli väri olisi omana ominaisuutenaan, haku ei tunnistaisi tuolia, jossa on sininen jalka, koska sinijalkaista tuolia ei olisi. Olisi vain kokonaan sininen tuoli, tai sininen ja punainen tuoli. Jos haluttaisiin hakea nimenomaan sinisellä jalkamallilla varustettuja tuoleja, pitäisi jokaiselle ominaisuudelle tai ominaisuusryhmälle pystyä myös määrittämään oma väri. Jokaisen tuotteen ominaisuudet päätettiin luokitella kolmeen ominaisuusryhmään värin ja muiden ominaisuuksien määrittelyn yksinkertaistamiseksi. Nämä ominaisuusryhmät olisivat nimeltään Jalkamalli/Runko, Kansi ja Verhoilu. Jalkamalli ja runko päätettiin yhdistää yhdeksi ryhmäksi, koska näkyvä runko on yleensä samaa materiaalia kuin jalatkin. Mikäli runko olisi eri materiaalia kuin jalkamalli, sitä ei pitäisi nähdä verhoilun alta. Tehtiin siis olettamus, että tuotteiden materiaaleilla oli väliä vain silloin kun ne ovat näkyvillä. Esimerkiksi sohvan runkoa ei näe, ja vaikka se olisi tehty koivusta, voisivat jalat silti olla saarnia. Tässä esimerkissä sohvan jalat siis valittaisiin tärkeämmäksi ominaisuudeksi. Tämä on järjestelmän heikoin kohta, mutta heikkouden minimoimiseksi lisättiin tauluun *rivi* tekstikenttä, jossa voitaisiin kuvailla tarkemmin tuotteen haluttuja ominaisuuksia. Itse taulun nimi ei viittaa runkoon, mutta sillä ei ole väliä, koska loppukäyttäjä ei tule näkemään taulun nimeä (kuvio 30).

Tässä kohtaa ominaisuusryhmät muodostivat kolme taulua *Jalkamalli, Kansimalli* ja *Verhoilumalli*. Lisänä oli myös erillinen taulu nimeltä *Lisäominaisuudet*. Sen tehtävä oli pitää sisällään erilaisia lisäominaisuuksia, jotka eivät sovi ominaisuustauluihin. Tällaisia lisäominaisuuksia olivat muun muassa käsinojat ja erilaiset tallat tuoleihin.

Koska kaikilla ryhmillä oli aina jokin ominaisuus, jonka ne jakavat, tehtiin kaikista tarvittavista ominaisuuksista oma taulu. Tämä taulu piti sisällään vain yhden tyyppistä tietoa, kuten edellä mainitun värin. Ominaisuusryhmissä oli puolestaan viiteavaimia, jotka ovat yhteyksissä tarvittaviin ominaisuustauluihin. Poikkeuksena taulu *Verhoilumalli*, jossa sarakkeella Verhoilu_variaatio ei ollut omaa taulua. Tämä johtui siitä, että sinne tallennettaisiin neljä vakioarvoa. Ominaisuusryhmät muodostivat tässä kolme kokoamisaluetta, joihin tallennettiin eri tuotteiden ominaisuuksia (kuvio 35).



KUVIO 35. Ominaisuusryhmät ja ominaisuudet

Aluksi ominaisuusryhmät olivat kaikki yhdessä taulussa, toisin kuin lopullisessa versiossa. Yksi taulu päätettiin jakaa kolmeen osaan, koska siten saatiin minimoitua tauluun jäävät tyhjät sarakkeet. Esimerkkituotteena toimii tuoli, joka ei täytä yhtäkään riviä taulusta *Kansimalli* (kuva 35). Tämän kaltaisten ratkaisujen tarkoituksena oli tuottaa mahdollisimman kompakti tietokanta, jossa tauluihin jäisi mahdollisimman vähän vain puoliksi täytettyjä rivejä ja ne veisivät vähemmän tilaa. Tämä vaikuttaa lopulta siihen, minne tietoa kasautuu ja missä muodossa. Liitteessä 1 on kuva siitä, miltä tietokanta valmiina näyttää.

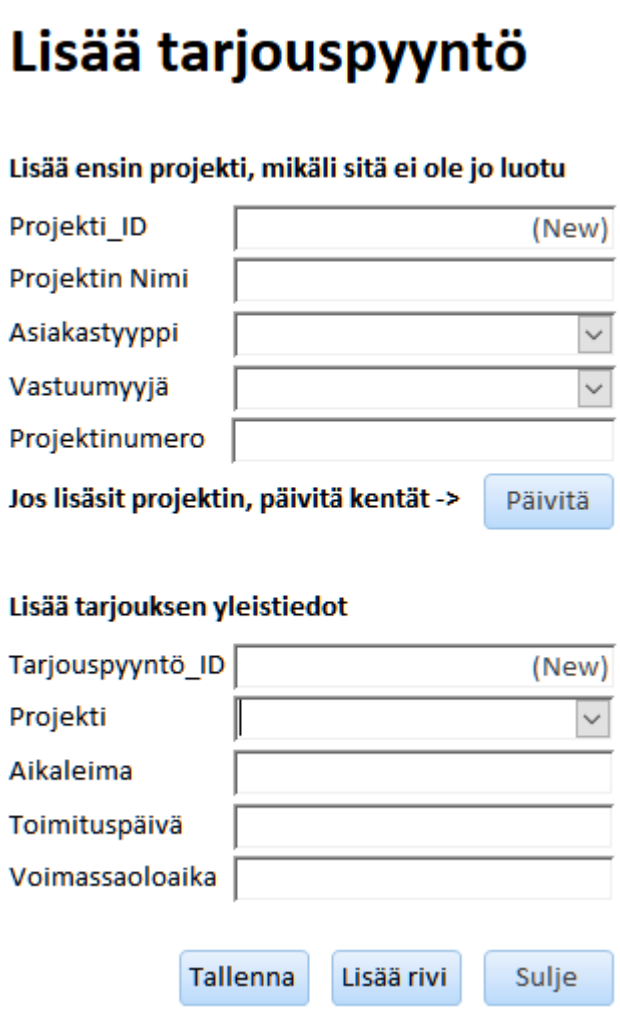
4.4 Testaus ja käyttöliittymä

Kun kaikki tasot oli saatu valmiiksi, lisättiin helposti saatavilla oleva tieto tauluihin. Näihin sisältyivät myyjät sekä tavarantoimittajat. Näitä varten oli olemassa valmiita Excel-listoja, jotka pystyttiin vain vähäisellä muokkaamisella tuomaan Accessiin. Tauluihin lisättiin myös eniten käytetyt kankaat koodeineen, sekä niiden toimittajat. Aikaisemmin oli kokeiltu yhteyksien toimivuutta kuvitteellisilla tuotteilla ja yksittäisten yhteyksien toimivuutta. Nyt testattiin koko tarjouspyynnön luomista alusta loppuun, ja aloitettiin oikeiden tuotteiden perustaminen erilaisine ominaisuuksineen. Projektien aitoudella ei ollut vielä merkitystä, koska oikeita projekteja alettaisiin lisäämään vasta, kun tietokanta olisi kokonaan valmis. Kaikki lisätyt projektit olivat siis kuvitteellisia.

Jotta käyttäjä pääsisi lisäämään tuotteita ja projekteja, tehtiin kaksi demo formia, joita he tulisivat mahdollisesti käyttämään. Formit ovat Accessin työkaluja, joilla voidaan luoda lomakkeita, valikoita ja selailla taulujen sisältämään informaatiota käyttäjäystävällisemmin.

Tarjouspyynnön tekeminen ja tuotteiden lisääminen alkavat aina projektin perustamisella. Projektille annetaan nimi, asiakastyyppi ja vastuumyyjä/myyjät. Tämän jälkeen tehdään tarjouspyyntö, joka on yhteydessä projektiin. Tarjouspyynnöllä valitaan toimittajat ja asetetaan tarjouksen voimassaoloaika, sekä toimitusviikko. Kuviossa 36 näkyvä

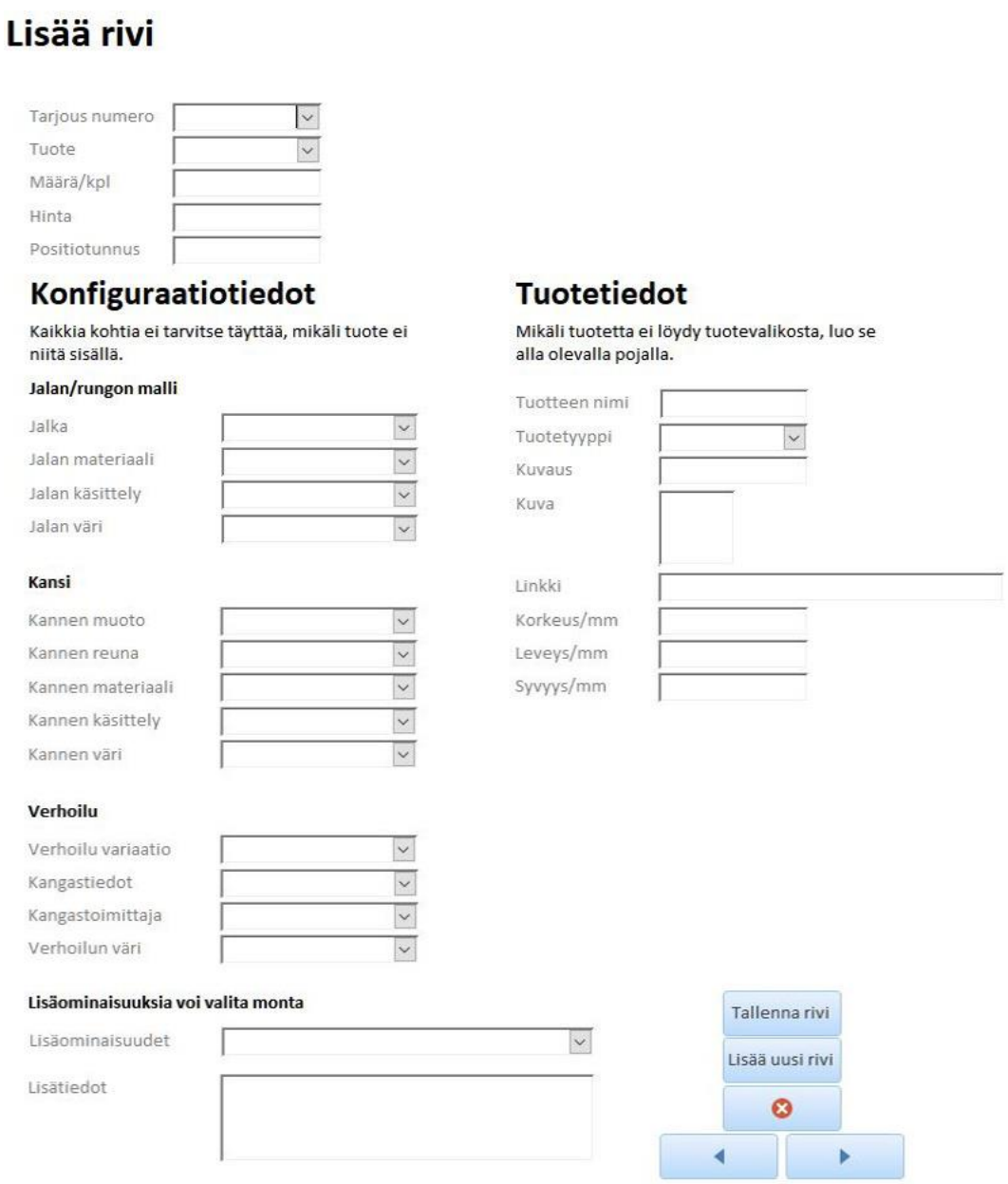
lomake osoittautui toimivaksi, joten siihen ei tehty toiminnallisia muutoksia. Päivitä-nappi on keskellä lomaketta, jotta sen täyttö tapahtuisi järjestyksessä. Mikäli uusi projekti lisätään, se päivittyy alempaan taulukkoon napista. Yleistiedolle olisi voitu tehdä oma formi, mutta kaikista lomakkeista yritettiin tehdä saman kokoisia, eli ei liian pieniä eikä liian isoja, jotta ne olisivat käyttäjälle miellyttävämpiä ja näyttäisivät virallisemmilta.



KUVIO 36. Tarjouspyynnön lisäyslomake

Kun tarjouspyynnön taustatiedot ovat valmiita, aloitetaan rivien lisääminen tarjouspyynnölle. Mikäli valikosta ei löydy oikeaa tuotetta, perustetaan se samasta kaaviosta. Kun oikea tuote on löydetty tai perustettu, sille annetaan positiotunnus, lukumäärä ja lisätään ominaisuuksia. Nämä ovat jokainen omalla paikallaan ja mikäli valikoista puuttuu haluttu arvo, se lisätään suoraan tauluun sille varatusta kentästä. Kun tuotteen

ominaisuudet on saatu lisättyä, se tallennetaan ja lisätään haluttaessa toinen rivi (kuvio 37). Yhdellä rivillä voi olla vain tietyillä ominaisuuksilla varustettu tuote.



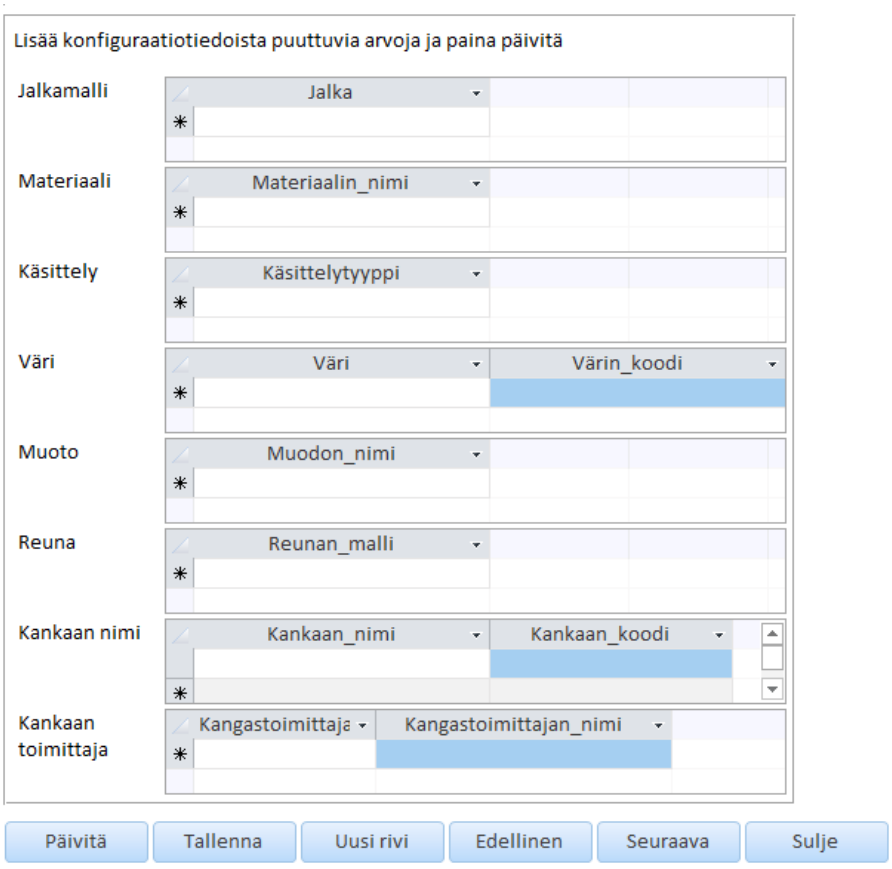
KUVIO 37. Rivinlisäyslomake

4.5 Rakenteellisia lisäyksiä

Kun tietokantasovelluksella oli toimiva käyttöliittymän alku, pidettiin demopalaveri, jossa näytettiin ostotiimille, kuinka lomakkeita käytetään, miten ohjelma rakentuu ja miten se toimii. Esitykseni jälkeen käytiin läpi,

mitä he vielä haluaisivat lisäksi, jotta se toimisi mahdollisimman tehokkaasti heidän tarpeidensa mukaan. Palaverissa tuli ilmi muutamia kohtia, jotka olisi lisättävä. Rivinlisäyslomakkeelle haluttiin mahdollisuus lisätä puuttuvia ominaisuuksia, mikäli valikosta ei löytyisi sopivaa. Tarjouspyyntö tehtäisiin lopulta raporttina, mutta sitä varten olisi kasattava kokoonpano. Tämä tarkoittaa käytännössä sitä, että kaikkia projektiin tarvittavia tuotteita ei voida kysyä samalta tavarantoimittajalta, koska he eivät välttämättä valmista kuin sohvia. Tuotteiden ja niiden ominaisuuksien valitseminen projektille olisi tällöin vasta ensimmäinen vaihe. Toisessa vaiheessa valittaisiin ne tuotteet, joista aiotaan tehdä tarjouspyyntö tietylle toimittajalle.

Tuotteiden ominaisuuksien manuaalista lisäämistä varten luotiin jokaisen ominaisuusryhmän rivinlisäyslomakkeeseen alilomake, joka päivittäisi tiedon tauluun. Alilomakkeet näkyvät alla olevassa kuviossa 38.

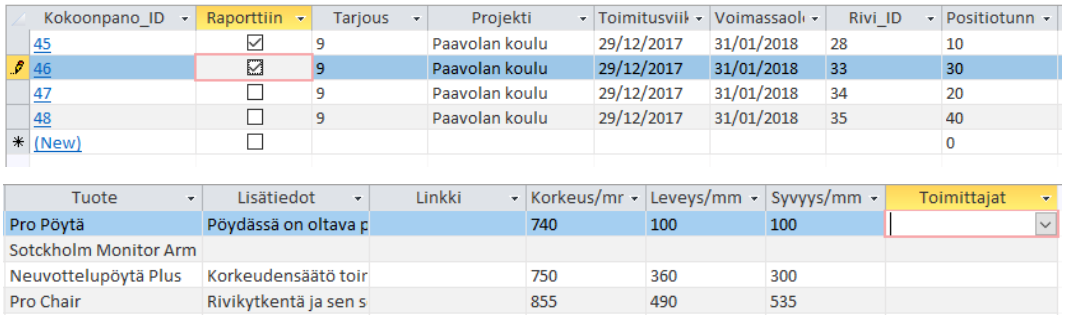


KUVIO 38. Tuotteiden ominaisuuksien manuaalinen lisääminen

Jotta oikeiden tuotteiden päätyminen haluttuun tarjouspyyntöön mahdollistuisi, oli tehtävä vielä yksi lisätaulu nimeltä RFQ*Kokoonpano.* Tämä muokkasi melkein koko tietokannan rakennetta, sillä taulussa RFQ*Kokoonpano* oli nähtävä tuotteen nimi, mille projektille se kuului, mille riville se tarjouspyynnöllä kuuluu, ja toimittaja. Päätettiin, että kokoonpanoon tulevat tiedot oli kopioitava jo valmiista tiedoista, ja näin luotiin tavallaan vain tuotelista, josta tarjouspyyntö sitten luotaisiin. Taulu *Toimittaja* jouduttiin siirtämään kiinni kokoonpanoon, koska vasta täällä valittaisiin, kenelle kyseinen RFQ lähetetään. Piti myös liittää taulu *RFQKokoonpano* tauluihin *Projekti, Rivi* ja *Tuote*, jotta oikeat arvot päivittyisivät muutettaessa myös kokoonpanossa. Liitteessä 2 on kuva siitä, miten kokoonpano muutti tietokannan rakennetta.

Kopioiminen kokoonpanoon tapahtuu erilaisia kyselyitä käyttämällä. Kyselyt piilotettiin kahden napin taakse, nämä olivat ”Suodata rivit” ja ”Tuo Rivit Kokoonpanoon”. Napeista käynnistyy kolme queryä. Ensimmäisessä haetaan tietyn tarjouspyynnön ID:llä halutun tarjouspyynnön rivit. Kun rivit on haettu, query luo uuden taulun välimuistiksi nimeltään *tbl_temp_rivi* ja tallentaa haetut arvot sinne. Tämän jälkeen rivit tuodaan välimuistista kokoonpanoon, johon ne tallentuvat. Kun prosessi toistetaan, uudet arvot kopiotuvat välimuistiin vanhojen arvojen päälle. Nyt taulussa *RFQKokoonpano* on kaikki tietyssä projektissa tarvittavat tuotteet. Kokoonpanoon lisättiin myös sarake ”Raporttiin”. Tämä on tietomuoto Yes/No, eli boolean. Se toimii valitsimena, jolla saadaan raporttiin eli lopulliseen tarjouspyyntöön tietyt tuotteet.

Kokoonpanosta tehtiin taulukkomallinen lomake (data sheet), jossa jokaisen tuotteen edessä olisi edellä mainittu valitsin kuvio 39. Huomiona, nyt jokaiselle riville on valittava erikseen tavarantoimittaja.



Kokoonpano_ID	Raporttiin	Tarjous	Projekti	Toimitusviik	Voimassaol	Rivi_ID	Positiotunn
45	☑	9	Paavolan koulu	29/12/2017	31/01/2018	28	10
46	☑	9	Paavolan koulu	29/12/2017	31/01/2018	33	30
47	☐	9	Paavolan koulu	29/12/2017	31/01/2018	34	20
48	☐	9	Paavolan koulu	29/12/2017	31/01/2018	35	40
(New)	☐						0

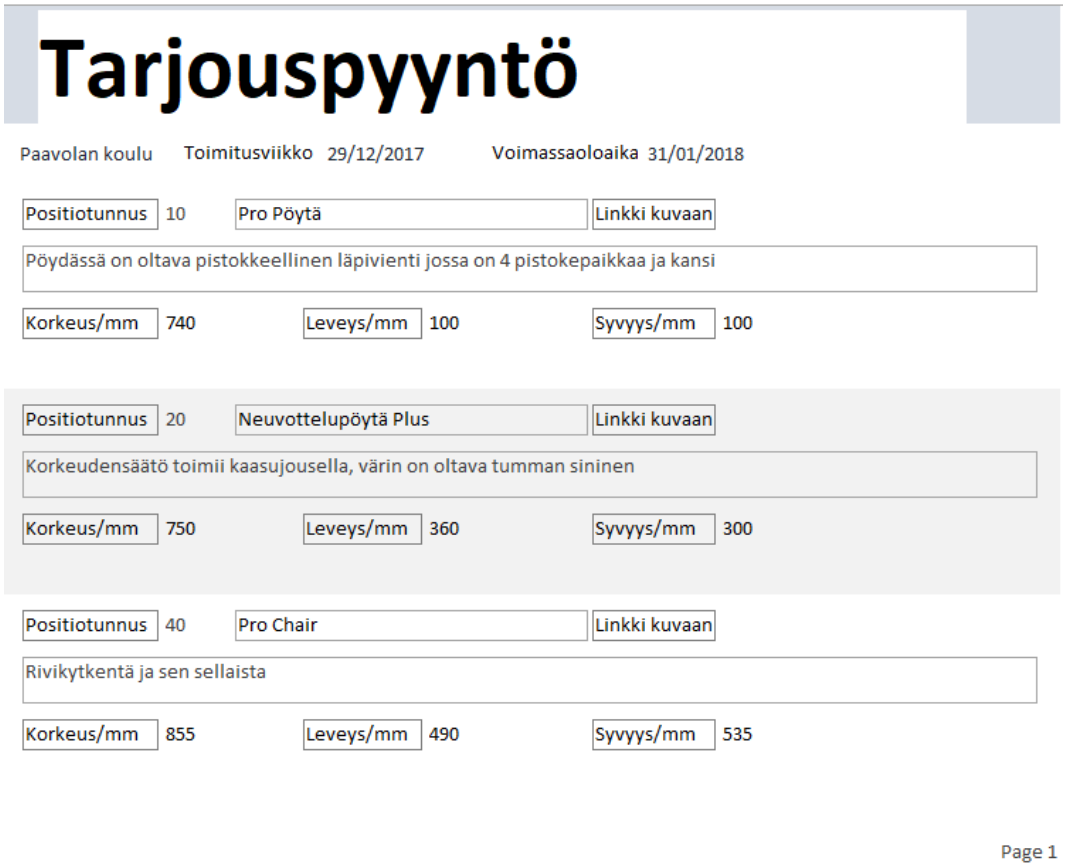
Tuote	Lisätiedot	Linkki	Korkeus/mr	Leveys/mm	Syvyys/mm	Toimittajat
Pro Pöytä	Pöydässä on oltava p		740	100	100	
Sotckholm Monitor Arm						
Neuvottelupöytä Plus	Korkeudensäätö toir		750	360	300	
Pro Chair	Rivikytkentä ja sen s		855	490	535	

KUVIO 39. KokoonpanoDS

4.6 Tarjouspyynnön raportti ja valikko

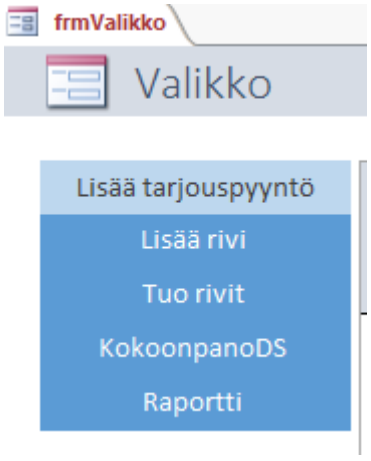
Jotta tarjouspyyntö saataisiin pdf-tiedostomuotoon, sille piti tehdä raportti. Raporteilla voidaan Accessissa kyselyitä hyväksi käyttäen luoda listoja, joissa on haluttuja arvoja. Nämä listat voidaan sitten viedä sähköpostiin ja lähettää suoraan tavarantoimittajalle. Raporttiin voidaan myös laittaa staattisia tekstipätkiä, joihin voidaan kirjoittaa vaikka yleisiä ostoehtoja. Näin niitä ei tarvitse kopioida aina erikseen sähköpostin liitteeksi. Raporttia varten tehtiin kysely, joka hakee kokoonpanosta valitut tuotteet. Se kysyy tarjouksen numeroa ja hakee sitten samalla tunnuksella olevat tuotteet, jotka ovat valittuna raporttiin.

Raportti luodaan ensin valitsemalla kokoonpanosta halutut tuotteet (kuvio 39). Tämä jälkeen painetaan valikosta ”Raportti” (kuvio 41). Se käynnistää kyselyn, johon kirjoitetaan halutun tarjouksen tarjousnumero. Kuvio 40 on esimerkkiraportista.



KUVIO 40. Tarjouspyyntöraportti

Lopuksi Formilla tehtiin vielä kaikille kaavioille ja raportille päävalikko (kuvio 41), josta niiden käyttäminen tapahtuisi mahdollisimman sujuvasti ja oikeassa järjestyksessä.



KUVIO 41. Päävalikko

4.7 Tietokannan toteutuksen lopputulos ja johtopäätökset

Tietokannalta haluttiin kahta pääominaisuutta, toiminnallisuutta ja sisällöllisyyttä. Toiminnallisuus jäi puutteelliseksi, sillä raportin laatimisesta tuli käyttäjälle työlästä. Siinä on monta kohtaa, mikä pidentää käyttöliittymän sujuvan käytön opiskeluun vaadittavaa aikaa. Ensin pitää luoda projekti ja laittaa siihen rivejä. Tämän jälkeen rivit siirretään kokoonpanoon ja kokoonpanosta lopulta raporttiin. Työntekijän pitää myös muistaa tarjousnumeroita. Yhden numeron muistaminen ei ole ongelma, sillä koko prosessi pyörii yhden numeron ympärillä. Toisaalta jos työntekijä joutuu jättämään työn kesken, aloittaa uusi ja myöhemmin palata edelliseen, muistamista tulee enemmän. Käyttäjälle olisi helpompaa muistaa tarjoukset nimien mukaan, mutta yksittäisten tarjouksien löytäminen hankaloituisi, sillä projekteilla saattaa olla samoja nimiä. Tämä luo ongelmia myös silloin, jos samaan projektiin pitää tehdä toinen tarjouspyyntö. Nämä ongelmia lukuun ottamatta tietokanta on täysin toimiva.

Sisällöllisyydeltään tietokantaan saatiin kaikki mitä haluttiin, mutta toiminnallisuuden muuttuminen rakenteellisten lisäysten jälkeen vaikutti toimittajiin. Koska taulun *tblToimittaja* yhteys oli vaihdettava taulusta *tblRFQ*, tauluun *tblRFQKokoonpano* (katso liitteet 1 ja 2). Tämä johti siihen, ettei käyttäjältä missään vaiheessa kysytä, keneltä nämä tuotteet aiotaan tilata. Nykyisessä muodossaan toimittaja pitää lisätä manuaalisesti tauluun KokoonpanoDS kohdassa, sitä varten ei ole lomaketta. Asia on korjattavissa, mutta riippuu käyttäjistä, minne he haluaisivat tämä kohdan lisättävän. Paras ratkaisu käyttäjälle olisi, jos toimittajat saisi jo valmiiseen lomakkeeseen, mutta silloin toimittajan valinta ei seuraisi prosessin toimintajärjestystä. Tämä johtuu siitä, että valmiit lomakkeet joihin käyttäjä syöttää tietoa, ovat prosessin alkupäässä, jolloin taulussa *tblRFQKokoonpano* ei vielä ole vastaavia rivejä. Loogisin ratkaisu olisi lisätä prosessin loppuun vielä yksi lomake, josta valitaan toimittaja.

Jatkotutkimuks olisi tarpeellinen, sillä edellä mainittujen asioiden parantelu ja korjaaminen olisi täydellinen lähtökohta. Mikäli tutkimus tehtäisiin

uudestaan, rakenteessa pitäisi ottaa paremmin huomioon raportin ja toimittajien seurannan vaikutukset.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Iskun Interior Oy:n Lahden konttorin ostotiimin käyttöön tietokanta, joka voitaisiin ottaa koekäyttöön. Idea opinnäytetyöhön lähti Iskun Interiorin ostotiimin vetäjältä. Tarkoituksena oli vastata tutkimuskysymykseen, eli saada tulostettua tietokannasta valmis tarjouspyyntö liitteineen. Lisäksi siitä voitaisiin hakea tarjouspyyntöjä ja tuotteita erilaisten luokitusten perusteella.

Tutkimus alkoi selvittämällä, mitä tietokanta pitäisi sisällään, eli mitkä attribuutit ja arvot ovat keskeisimmässä osassa, sekä miten nämä tiedot saataisiin tehokkaasti ja järkevästi liitettyä toisiinsa. Tämän jälkeen työ jaettiin kolmeen tasoon: projekti-, tarjouspyyntö- ja tarjouspyyntörivitasoon. Kun tasot oli suunniteltu, se luotiin Accessissa. Tutkimuksen edetessä huomattiin, että toteutus näytti usein toimivan ER-kaaviossa, mutta ei Accessissa. Tämä johti siihen, että tauluja piti suunnitella monesti uudelleen. Tätä tapahtui pääsääntöisesti tarjouspyyntörivitasolla.

Tietokannan suunnittelu tapahtui pääsääntöisesti paperilla. Sujuva Accessin käyttö mahdollisti suunnittelun myös ohjelmassa, mutta tämä tapahtui vasta kun tietokannan yleinen rakenne oli selvillä. Accessia käytettiin suunnittelussa oikeastaan siksi, että haluttiin tietää, onko ohjelmalla joitain rajoitteita. Suunnitelmaa oli myös helpompi havainnollistaa Accessia käyttäen.

Tutkimuksen tavoite saavutettiin, muttei täydellisesti. Tietokannan käyttöliittymästä olisi voitu tehdä käyttäjäystävällisempi siten, että käyttäjän ei tarvitsisi muistaa tarjouspyyntöjen ID-numeroita. Rivien tuominen raporttiin pitäisi myös pystyä toteuttamaan vaivattomammin. Käyttöliittymä toimii, mutta sen tehokas käyttäminen vaatii turhan paljon opettelua henkilöiltä, joilla on jo valmiiksi paljon työtehtäviä.

Yhteenvetona voidaan todeta, että tietokanta ei ole täysin valmis. Se toimii mallina, josta voidaan lähteä kehittämään pysyvämpää ratkaisua, mahdollisesti modulaarisempaa hallintajärjestelmää käyttäen. Jotta

nykyisestä rakenteesta saataisiin kaikki hyöty irti, olisi panostettava enemmän käyttöliittymän toiminnallisuuksiin. Tarjouspyynnön ulos saamiseksi olisi myös tehtävä parempi ratkaisu. Kun nämä ongelmat saataisiin ratkaistua, tietokannasta tulisi tehokas työkalu ostotiimin käyttöön.

LÄHTEET

Ashe-Edmunds, S. 2018. Objectives of an Inventory Database. Small Business – Chron. [viitattu 26.11.2018]. Saatavissa: https://smallbusiness.chron.com/objectives-inventory-database-64497.html

Buxton, S., Fryman, L., Güting, R, H., Halpin, T., Harrington, J, L., Inmon, W, H., Lightstone, S, S., Melton, J., Morgan, T., Nadeau, T, P., O`Neil, B., O`Neil, E., O`Neil, P., Schneider, M., Simson, G., Teorey, T, J. & Witt, G. 2009. Database Desig: know it all. United States. Morgan Kaufmann.

Chapple, M. 2018. Database Relationships. Lifewire. [viitattu 21.09.2018]. Saatavissa: https://www.lifewire.com/database-relationships-p2-1019758

Hernandez, M. 1997. Tietokannat – suunnittelu ja toteutus. Jyväskylä. IT Press.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Porvoo. WS Bookwell.

Laiho, M. Laine, H. Pelkonen, H. Rantanen, J. Tujunen, S. Virtanen, M.1997. RELAATIOTIETOKANTASANASTO. Tietotekniikan kehittämiskeskus ry. Helsinki. [viitattu 02.11.2018]. https://www.cs.helsinki.fi/relaatiosanasto/

Microsoft Corporation. 2017. Tietokannan normalisoinnin perusteiden kuvaus. [viitattu 18.10.2018]. Saatavissa: https://support.microsoft.com/fi-fi/help/283878/description-of-the-database-normalization-basics

Microsoft Corporation. 2018a. Archive Access data [viitattu 23.05.2018]. Saatavissa: https://support.office.com/en-us/article/archive-access-data-a4e5e789-ec7f-4e4c-bcde-74967be27d50

Microsoft Corporation. 2018b. Microsoft Office Access. [viitattu 07.11.2018]. Saatavissa: https://products.office.com/fi-fi/access

Microsoft Corporation. 2018c. Tietokannan suunnittelun perusteet. [viitattu 13.11.2018]. Saatavissa: https://support.office.com/fi-fi/article/tietokannan-suunnittelun-perusteet-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#bmterms

Pittenger, R. 2008. A Naming Scheme for Database Tables and Fields. Databasedev.co.uk. [viitattu 26.11.2018]. Saatavissa: http://www.databasedev.co.uk/database-naming.html

Sheridan, K. 2015. MS Access Simultaneous Editing. [viitattu 23.10.2018]. Saatavissa: https://answers.microsoft.com/en-us/office/forum/office_2010-access/ms-access-simultaneous-editing/8e2484b8-ed9e-480f-87c2-e52718ca7697

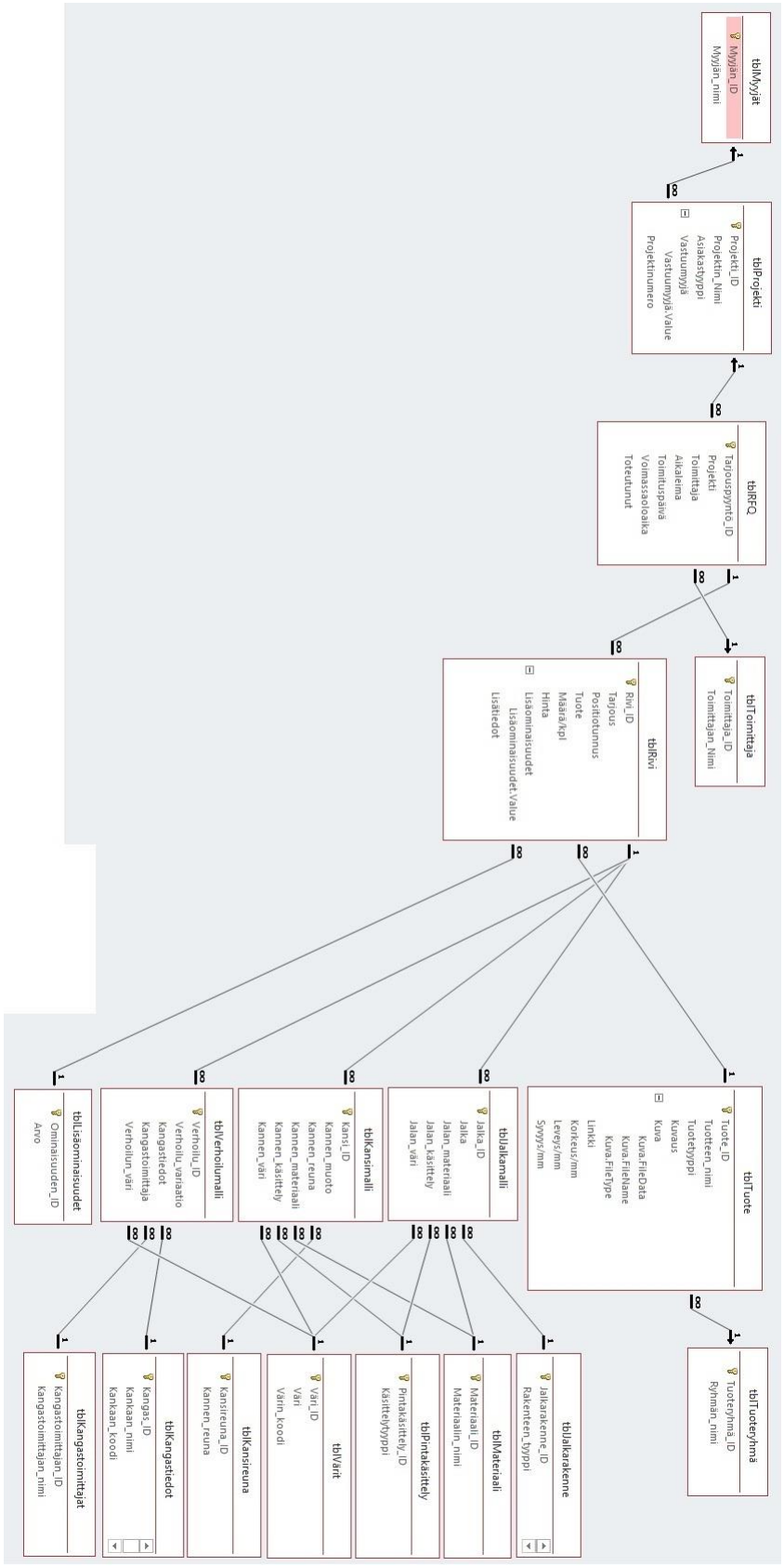
Technopedia. 2018. Data Redundancy. [viitattu 05.11.2018]. Saatavissa: https://www.techopedia.com/definition/18707/data-redundancy

Ullman, J., Widom, J. 1997. A First Course In Database Systems. Kolmas painos. Kiina. Pearson Education Asia Limited, China Machine Press.

W3schools.com. 2018. SQL NULL Values. [viitattu 11.10.2018]. Saatavissa: https://www.w3schools.com/sql/sql_null_values.asp

LIITTEET

LIITE 1 TIETOKANTA ENNEN KOKOONPANOA



LIITE 2 TIETOKANTA KOKOONPANON KANSSA