Harjoitustyö
Tietokantaohjelmointi 2025
Dokumentaatio – vaihe 2

Ryhmä 8: Valma Haavisto, Salli Valkama, Selina Rautakoski

Tämä on tietokantaohjelmoinnin kurssille tehdyn harjoitustyön dokumentaatio, jossa käydään läpi ohjelmassa toteutetut ominaisuudet, ohjelman käyttö sekä siihen jääneet puutteet. Lisäksi käydään läpi vaiheen 1 jälkeen tehdyt korjaukset, jotka on tehty korjauskehotusten pohjalta, ja ryhmän jäsenten välinen työnjako.

Ohjelman ominaisuudet

Ohjelman toteutetut ominaisuudet on esitetty taulukossa 1 alla. Mitään ominaisuuksista ei ole toteutettu laajasti, joten siksi sarake on jätetty pois. Jokaisesta ominaisuudesta ja tapahtumasta on lisätty erikseen selostus taulukon jälkeen.

Taulukko 1. Toteutetut ominaisuudet.

Tietokantaohjelmointi 2025	
Ryhmän numero: 8	Toteutettu
Tapahtuma T1	x
Tapahtuma T2	x
Tapahtuma T3	x
Tapahtuma T4	x
Tapahtuma T5*)	x
Tapahtuma T6*)	x
Tapahtuma T7*)	x
Tapahtuma T8*)	x
Raportti R1	x
Raportti R1	x
Raportti R3*)	x
Raportti R4*)	x
Tuki samanaikaiselle käytölle	
Käyttöliittymäominaisuudet *)	x
Mahdollisia lisätoimitoja (mitä?) *)	x
Nouseva ja laskeva järjestys hakutuloksille sarakkeittain	x
Automaattinen uloskirjaus roolin vaihdon yhteydessä	x
Ostoskorin tyhjentyminen napista tai roolin vaihdossa	x

T1 Asiakas kirjautuu ja rekisteröityy järjestelmään

Asiakas kirjautuu sähköpostilla ja asettamallaan salasanalla. Sähköposti on uniikki kaikilla asiakkailla. Rekisteröitymisen yhteydessä asiakas täyttää asiakastietolomakkeen ja tiedot talletetaan tietokantaan. Jokaiselle asiakkaalle luodaan automaattisesti uniikki asiakastunnus.

T2 Lisätään yksittäisen teoksen tiedot divarin tietokantaan ja keskustietokantaan

Järjestelmään voi lisätä teoksia joko divarin tai keskusdivarin roolissa. Lisäys eroaa rooleissa niin, että BookCopy -taulun store_id -kenttä saa erilaisen arvon. Lisäys tehdään syöttämällä kirjan ISBN, jolloin järjestelmä tarkistaa onko kirjalle olemassa jo vastaava Book -taulu. Jos ei, divari lisää ensin perustiedot kirjasta ja luo Book -taulun instanssin. Tämän jälkeen luodaan varsinainen teos eli BookCopy -taulun instanssi. Mikäli ISBN -tunnusta vastaava kirja löytyy, suoritetaan ensin T3.

T3 Lisätään yksittäinen teos divarille tietokantaan, jonka teostiedot jo löytyvät tietokannasta

Kun teostiedot löytyvät tietokannasta, eli kirjaa vastaava ISBN löytyy, divari syöttää suoraan teoskappaleen lisäämiseen tarvittavat tiedot eli sisäänosto -ja myyntihinnan. Lisäämisen jälkeen siirrytään takaisin divarikohtaiseen lisäämisnäkymään ja divarille näytetään tilaukseen liittyvä statusviesti lisäämisen onnistumisesta tai epäonnistumisesta.

T4 Asiakas tekee yksittäisen kirjan tilauksen

Asiakas valitsee haluamansa kirjan hakunäkymässä napauttamalla kirjan ISBN-kenttää. Tämän jälkeen asiakas näkee mahdolliset kirjasta olemassa olevat teoskappaleet. Jos teoskappale on vapaana, asiakas voi lisätä kirjan ostoskoriin painamalla ”+” -nappia. Ostoskoriin siirrytään oikean yläkulman napista. Ostoskorinäkymässä asiakas näkee tilauksen yhteenvedon, toimituskulut ja kokonaissumman. Asiakas pystyy tekemään tilauksen napauttamalla ”Tilaa” -nappia, ja hän saa näytölle viestin tilauksen onnistumisesta.

T5 Asiakas tekee tilauksen, jonka paino ylittää 2000 grammaa ja lähetys joudutaan jakamaan useaan erään

Asiakkaan näkökulmasta tämä tapahtuu vastaavasti kuin yksittäisen kirjan tilaaminen. Asiakas valitsee haluamansa kirjat hakunäkymässä ja lisää ne ostoskoriin, kuten edellä. Ostoskorinäkymässä tilauksen näkyy tilauksen yhteenveto, toimituskulut ja kokonaissumma, jotka ovat laskettu useamman lähetyserän mukaan. Tilaaminen tapahtuu napauttamalla ”Tilaa” -nappia, asiakas saa viestin tilauksen onnistumisesta.

T6: Triggeri päivittää keskusdivarin automaattisesti, kun divarin omaan tietokantaan tuodaan uusi myyntikappale

Ohjelmasta ei löydy varsinaista triggeriä, mutta se kuitenkin pitää keskusdivarin ja divarin tietokannat yhtenäisinä koko ajan. Kirjoista ei siis ole kopioita keskudivarin tietokannassa, vaan teokset erotellaan store_id -kentällä divareittain. Näin voidaan pitää yllä dynaamisesti päivittyvää kuvausta molemmista tietokannoista.

T7 Divarin tietokantaan on lisätty kirjoja, joita ei löydy keskustietokannasta ja keskusdivarin tiedot joudutaan päivittämään

Kuten T6 -kohdassa todettua, keskusdivari ei kopioi kirjoja itselleen vaan käyttää suoraan versioita, jotka löytyvät ulkopuolisen divarin tietokannasta. T7 ei siis voi tapahtua tällaisessa järjestelmän toteutuksessa.

T8 Divarin data on XML-muodossa, jonka rakenne noudattaa annettua DTD:tä

Paikallinen XML-tiedosto ladataan Divari-roolin näkymässä palvelimelle. Ohjelma lukee XML-tiedoston JS-objektiksi, jonka avulla tehdään teosten ja niteiden lisäykset tietokantaan. Struktuurista puuttuvat tiedot täytetään tietokantaan null-arvoina tai default-arvoina. Käyttäjä saa ilmoituksen lisäysten onnistumisesta tai epäonnistumisesta.

R1 Anna haun tulokset, jotka toteuttavat annetut kriteerit

Hakunäkymässä löytyy erilaisia kenttiä, jotka täyttämällä asiakas voi suodattaa hakutuloksia. Mikäli kriteereihin täsmääviä kirjoja löytyy, ne näytetään asiakkaalle hakulaatikon alla. Tyhjentämällä kentät ja painamalla uudestaan hakupainiketta asiakas pääsee takaisin hakunäkymään, jossa näytetään lista kaikista tietokannassa olevista kirjoista.

R2 Ryhmittele myynnissä olevat teokset niiden luokan mukaan. Anna luokkien teosten kokonaismyyntihinta sekä keskihinta

Raportti tuotetaan taulukkomuodossa divariroolin ja keskusdivari-roolin näkymissä. SQL-kysely huomioi vain myynnissä olevat teoskappaleet, eli myydyt jätetään huomiotta tuloksesta. Divari-roolin raportissa huomioidaan vain divaritunnus 2 myynnissä olevat teoskappaleet.

R3 Listaa kaikki asiakkaat, sekä näiden viime vuonna ostamien teosten lukumäärä

Raportti tuotetaan Keskusdivari-roolin näkymässä. Tuloksessa huomioidaan edellisen vuoden aikana vahvistetut tilaukset (vahvistusaika löytyy tietokannasta). Valmis raportti latautuu selaimeen.

Nouseva ja laskeva järjestys hakutuloksille sarakkeittain

Hakunäkymässä napauttamalla sarakkeen otsikkoriviä hakutulokset on mahdollista järjestää sarakkeittain nousevaan tai laskevaan järjestykseen.

Automaattinen uloskirjaus roolin vaihdon yhteydessä

”Vaihda Rooli” -painikkeen napauttaminen haku- tai ostoskorinäkymässä kirjaa asiakkaan ulos ja tyhjentää ostoskorin.

Ostoskorin tyhjentyminen napista tai roolin vaihdossa

Asiakas pystyy roolin vaihdon lisäksi tyhjentämään koko ostoskorin sisällön myös ilman uloskirjautumista ostoskorinäkymän ”Tyhjennä ostoskori” -napista.

Jäsenten välinen työnjako

Valma Haavisto: Book -tiedostoihin liittyvät ominaisuudet, hakunäkymä, osa dokumentaatiosta (33%)

Salli Valkama: Customer-tiedostoihin liittyvät ominaisuudet, raportteihin 2 ja 3 liittyvät ominaisuudet, XML-tiedostoihin (T8) liittyvät ominaisuudet, osa dokumentaatiosta (33 %)

Selina Rautakoski: Order -tiedostoihin liittyvät ominaisuudet, ostoskorinäkymä, osa dokumentaatiosta (33%).

Toteutuksen kuvaus

Välineet

Ohjelma on toteutettu node.js projektina. Projekti muodostaa tietokantayhteyden yliopiston tarjoaman palvelimen psql-tietokantaan ja tarjoaa REST-rajapinnan datan lisäämiselle, poistamiselle ja muokkaamiselle. Ohjelman toteutuksen apuna on käytetty tekoälyä (ChatGPT 4o, Copilot VSCode) testidatan luomisessa, virheiden etsinnässä sekä teknologioiden käytön opettelussa. Node.js ja palvelin-asiakas-mallin mukainen projektistruktuuri ei ollut entuudestaan tuttua ryhmän jäsenille.

Taulut

Toteutus hyödyntää tauluja Asiakas (Customer), Teos (Book), Teoskappale (BookCopy), Tilaus (BookOrder), Tyyppi (Type), Luokka (Category), Divari (Store), Toimitus (Shipment) ja Postikulut (ShippingRates). Taulujen luontilauseita ja tauluihin liittyviä rajoituksia voi tarkastella src/migrations-kansion tiedostosta createDefaultTables.js.

Ohjelman rakenne

Ohjelman käyttöliittymään liittyvät html-tiedostot ovat kansiossa public. Ohjelman logiikan ja tietokannan käsittelyyn liittyvät tiedostot ovat kansiossa src. Asiakas-, kirja- ja tilaustietojen (Customer, Book, Order) käsittelystä vastaavat src-kansion controller-, model- ja api-tiedostot, jotka tekevät hakuja tietokantaan, muokkaavat tietokannan dataa ja välittävät pyynnöt käyttöliittymän ja käyttäjän välillä. Index.js vastaa ohjelman käynnistämisestä ja tietokantayhteyden valmistelusta. Migrations-kansio sisältää tietokannan valmisteluun tarvittavat tiedostot, jotka luovat tarvittavat psql-taulut ja lisäävät niihin testidatan.

Lisäksi src/config-kansio sisältää tiedoston db.js, joka yhdessä .env-tiedoston kanssa huolehtii tietokantayhteyden muodostamisesta. T8 XML-ominaisuus hyödyntää tiedoston väliaikaiseen lataamiseen kansiota uploads.

Ohjelman käyttö

Ohjelman käynnistäminen

Ennen ohjelman suorittamista on asennettava lisäpaketteja, jotka kaikki pitäisi saada asennettua ajamalla komentoikkunassa ”npm install” ennen ohjelman käynnistämistä.

Ohjelma käynnistetään “node index.js” komennolla. Ohjelman käynnistyessä tarvittavat relaatiot luodaan tietokantaan ja testidata lisätään. Jos taulut ja testidata ovat jo olemassa tietokannassa, ohjelma ensin poistaa ne käynnistymisen yhteydessä ja luo ne sitten uudestaan. Ohjelma aukeaa selaimessa osoitteeseen http://tie-tkannat.it.tuni.fi:$8080.

Ohjelman aloitussivulla valitaan käyttäjärooli klikkaamalla roolien painikkeita. Roolin valitsemisen jälkeen aukeaa näkymä, jossa on saatavilla roolin mukaiset toiminnallisuudet. Takaisin roolin valintanäkymään palataan klikkaamalla painiketta “Vaihda rooli”.

Toiminnot roolissa asiakas

Asiakas-roolin valitsemisen jälkeen käyttäjälle avautuu kirjautumis- ja rekisteröitymisnäkymä. Käyttäjä kirjautuu järjestelmään syöttämällä kenttiin sähköpostin ja salasanan (T1). Käyttäjä näkee virheilmoituksen, jos jompi kumpi arvoista on puuttuva tai virheellinen. Uuden asiakkaan voi rekisteröidä valitsemalla kirjautumissivulla “Rekisteröidy”, minkä jälkeen käyttäjä ohjataan täyttämään asiakastietolomake (T1). Käyttäjä näkee virheilmoituksen, jos jokin arvoista on puuttuva tai virheellinen. Onnistuneen rekisteröitymisen jälkeen käyttäjä ohjataan takaisin kirjautumissivulle.

Onnistuneen kirjautumisen jälkeen käyttäjälle avautuu kirjahaku-näkymä, jossa käyttäjä voi hakea myynnissä olevia kirjoja. Kirjoja on mahdollista suodattaa nimen, kirjailijan,

luokan ja tyypin mukaan. Hakutulokset on mahdollista järjestää sarakkeittain nousevaan tai laskevaan järjestykseen napauttamalla sarakkeen otsikkoriviä ja tämä taulukko muodostaa raportin R1. Napauttamalla kirjan ISBN-tunnusta avatuu näkymä mahdollisista kirjaan liitetyistä teoskappaleista. Teoskappale on mahdollista lisätä ostokoriin napauttamalla ”+” -nappia.

Hakunäkymästä pääsee siirtymään ostoskoriin oikean yläkulman napista. Ostoskorinäkymässä käyttäjä pystyy tarkastamaan ostoskorin sisällön, poistamaan yksittäisiä kirjoja, kirjan rivillä olevalla ”Poista” -napilla tai tyhjentämään koko ostoskorin ”Tyhjennä ostoskori” -napista vasemmalla alhaalla tai tekemään tilauksen napauttamalla ”Tilaa” -nappia. Ostoskorista pääsee myös takaisin hakunäkymään oikean yläkulman napista.

Sekä haku- että ostoskorinäkymässä on myös painike ”Vaihda Rooli”, jota napauttamalla käyttäjä kirjautuu ulos ja siirtyy takaisin aloitussivulle.

Toiminnot roolissa Divari

Divari-roolin valitsemisen jälkeen käyttäjälle avautuu näkymä, jossa divarin ylläpitäjä voi lisätä tietokantaan myytäviä kirjoja ja niteitä, sekä tuottaa raportteja. Riippuen siitä, löytyykö syötetty ISBN jo tietokannasta, divarinäkymä suorittaa tapahtuman T2 tai T3.

Näkymässä voi tuottaa raportin R2 klikkaamalla painiketta, jolloin divarin myynnissä olevat kirjat listataan näkymään. Näkymässä voi myös lisätä kirjoja ja teoskappaleita myyntiin lataamalla palvelimelle XML-tiedoston kenttää ja painiketta käyttäen (T8).

Toiminnot roolissa Keskusdivari

Keskusdivari-roolin valitsemisen jälkeen käyttäjälle avautuu näkymä, jossa keskusdivarin ylläpitäjä voi lisätä tietokantaan myytäviä kirjoja ja niteitä, sekä tuottaa raportteja. Samoin kuin divarinäkymässä, näkymä suorittaa tapahtuman T2 tai T3 riippuen annetun ISBN-tunnuksen olemassaolosta.

Näkymässä voi tuottaa raportit R2 ja R3 klikkaamalla painikkeita. Divarin myynnissä olevat kirjat listataan näkymään (R2) ja asiakkaiden aiempana vuonna ostamat kirjamäärät tuotetaan CSV-tiedostoksi, jonka käyttäjä voi ladata itselleen selaimesta (R3).

Ohjelman testaaminen

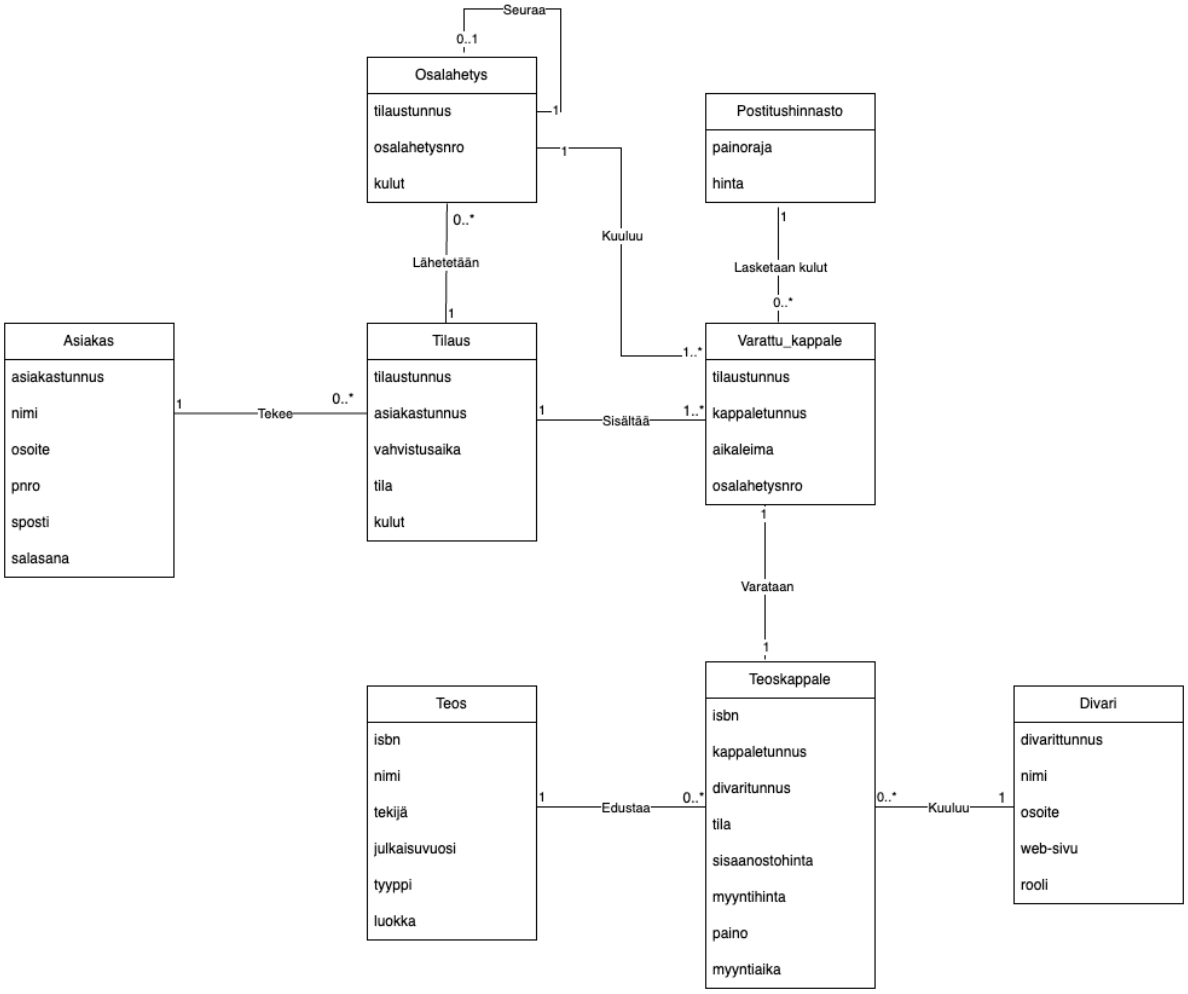
Ohjelman asiakaskirjautumista, teoskappalehakuja ja raporttien tuottamista voi kokeilla esimerkkidatalla, joka löytyy src/migrations-kansion tiedostosta insertTestData.js. Esimerkkitiedosto T8 varten (XML muuntaminen) löytyy projektikansiosta nimellä divari4testi.xml.

Ohjelmaa voi myös testata komentoriviltä curl-työkalulla. Esimerkkejä kometoihin voi katsoa api- ja controller-tiedostoista. Esimerkki: curl http://tie-tkannat.it.tuni.fi:8080/api/books/bookcopies

Ohjelma näyttää käyttäjälle suppeasti viestejä ja virheilmoituksia. Developer Tools – ikkunaan ja komentoriville tuotetut ilmoitukset antavat tarkempaa tietoa tapahtumista.

Muutokset

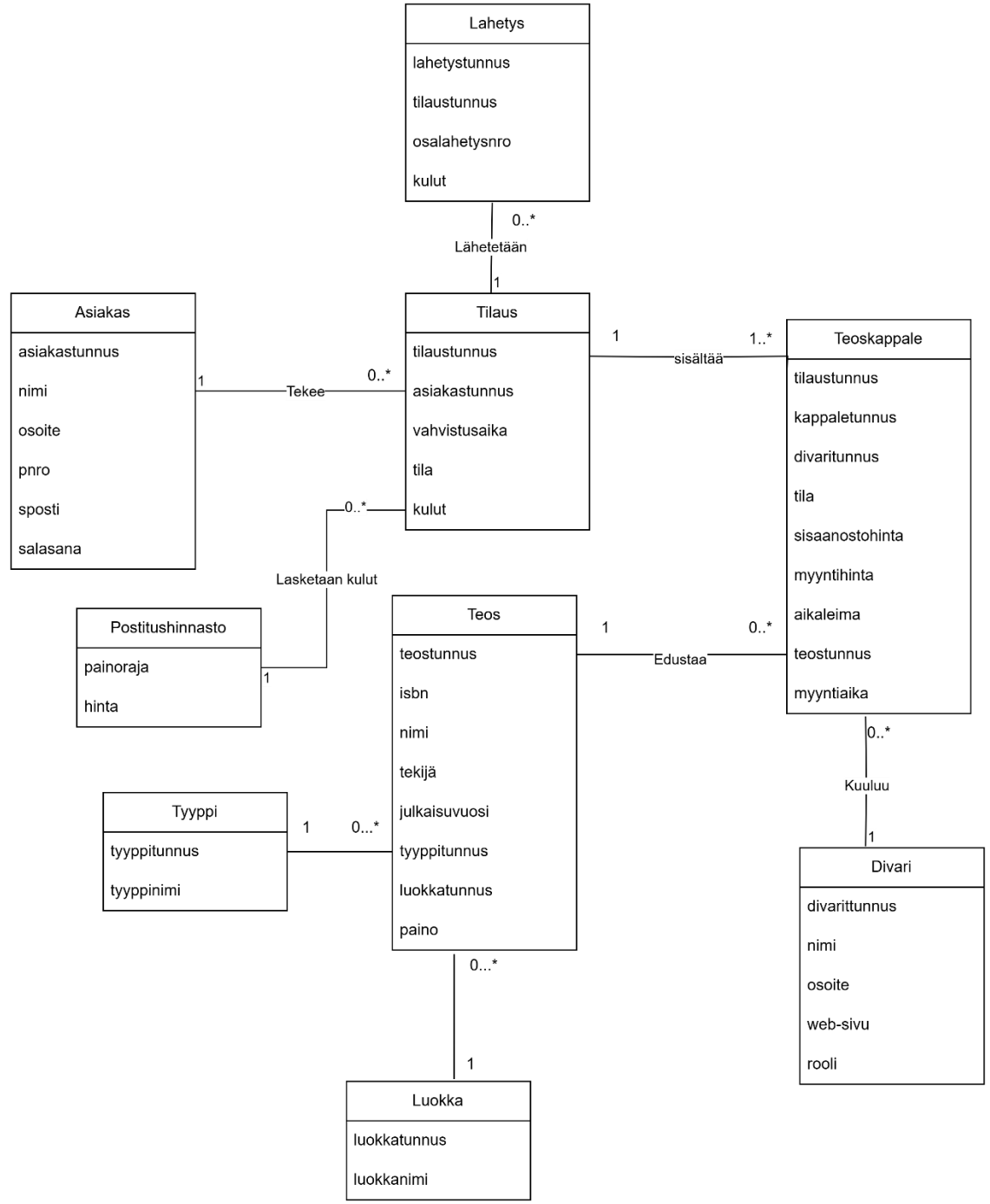
Merkittävimmät muutokset työn ensimmäiseen vaiheeseen verrattuna liittyvät piirrettyihin luokkakaavioihin. Nämä muutokset on toteutettu myös koodissa ja ne ovat vaikuttaneet merkittävästi työn lopulliseen versioon. Tässä kappaleessa käydään läpi nämä luokkakaavion muutokset sekä sen jälkeen mahdolliset toteutusvaiheessa tehdyt muutokset. Vaiheen 1 luokkakaavio on esitetty kuvana.



Kuva 1. Vaiheen 1 kaavio.

Alla taas on esitetty vaiheen 2 luokkakaavio, joka on myös toteutettu koodissa. Kun sitä vertaillaan vaiheen 2 luokkakaavioon, huomataan, että siihen on tehty muutamia

muutoksia. Esimerkiksi tyyppi ja luokka esitetään vaiheessa 2 erillisinä tauluina. Postituskulut linkitetään suoraan tilaukseen, mikä helpottaa niiden laskemista tilaukselle. Lisäksi osalähetyksessä oleva seuraa -suhde on poistettu ja sen sijaan tilaukseen voi kuulua monta eri lähteystä. Merkittävä muutos oli myös selkeyttää tilauksen, osalähetyksen sekä tilauskappaleen suhdetta. Tässä kohtaa ratkaisimme asian niin, että linkitimme tilauksen suoraan teoskappaleeseen ja lisäsimme teoskappaleelle aikaleimakentän, joka kertoo varauksesta.



Kuva 2. Vaiheen 2 kaavio.

Kaavioon tehdyt muutokset helpottivat tietokannan toteuttamista koodissa sekä vähensivät päällekkäisen ja toisteisen tiedon tallentamista. Toteutusvaiheessa tehtiin myös joitakin, lähinnä kosmeettisia, muutoksia tietokantaan. Esimerkiksi kirjojen painot päätettiin tallettaa tietokantaan kilogrammoina grammojen sijaan, koska suurten lukujen tallettaminen tauluun tuotti ongelmia.

Arvio työstä

Puutteita

Toteutimme työssämme vain yhden tietokannan kaikille rooleille poiketen ohjeesta, jossa suositeltiin tekemään kopioita tietokannoista. Teimme tämän ratkaisun, koska se tuntui selkeämmältä ja helpotti koodia. Joka tapauksessa erottelemme eri roolit (yksittäinen divari, keskusdivari), Teoskappaleen taulussa (BookCopy) divaritunnuksella (store_id), niin että eri divareiden kirjoja on mahdollista tarkastella erikseen.

Teoksella on olemassa tunnus, jota käytämme koodissa ja SQL-kyselyissä teosten hakemiseen. Kuitenkin käyttöliittymässä teosta tai teoskappaletta ei pysty lisäämään ilman ISBN:ää, vaan käytämme sitä tunnistamaan teosten ja teoskappaleiden vastaavuus. Tämä siksi, että koimme nimeen perustuva lisäämisen liian alttiiksi erilaisille duplikaateille, esim. kirjoitusasusta tai -virheistä johtuen.

Keskityimme työssämme enemmän tietokantojen käyttämiseen ja pyydettyjen toiminnallisuuksien toteuttamiseen, joten tietoturva jäi vähemmälle huomiolle. Esimerkiksi käyttäjä pystyy vapaasti valita käyttäjärooliksi divari tai keskusdivari ja lisäämään kirjoja tietokantaan. Tässä olisi hyvä olla jonkunlainen kirjautuminen, mutta tämän projektin puitteissa halusimme keskittyä muihin asioihin.

Tietoturvaan liittyen myös asiakkaiden tiedot, erityisesti salasanat olisi hyvä salata paremmin. Tässä projektissa ne ovat tietokannassa sellaisenaan samasta syystä kuin edellä. Koimme tärkeämmäksi keskittyä tietokantayhteyteen ja palvelun toiminnallisuuksiin.

Yksittäisissä lisäyksissä ei ole huomioitu transaktiota, joka on osittain kieliriippuvainen ratkaisu. Tämä on huomioitu kuitenkin funktioissa, joissa on tarkistuksia olemassa oleville tiedoille.

Haasteita

Ohjelman kokonaistoiminnan hahmottaminen oli aluksi haastavaa, mikä näkyykin siinä, että muokkasimme alkuperäisiä tauluja myöhemmin kokonaiskuvan selkeydyttyä paremmin. Yhdessä keskusteleminen ja taulujen miettiminen auttoi projektin hahmottamisessa paremmin.

Pohdimme aloittaessamme koodin kirjoittamista, mitä kieltä haluamme käyttää. Koska ohjelmointikielet ovat englanniksi ja olemme myös muissa konteksteissa tottuneet kirjoittamaan koodia englanniksi, tuntui luontevalta valita englanti koodin kommentointiin ja tietokannan kieleksi. Lisäksi teknologioiden käyttö oli uutta ja niistä löytyi paremmin tietoa englanniksi. Koska kurssi on kuitenkin suomenkielinen, teimme graafisen käyttöliittymän suomen kielellä.