Lappeenrannan teknillinen yliopisto
Tietotekniikan koulutusohjelma
CT60A4303 - Tietokantojen perusteet
Antti Knutas

Tietokantojen perusteet

Harjoitustyö – Tietokannan suunnittelu

23.7.2019
Tomi Enberg - 0518456
tomi.enberg@student.lut.fi
Markus Strandman - 0521605
markus.strandman@student.lut.fi
Aarne Savolainen - 0521838
aarne.savolainen@student.lut.fi

SISÄLLYSLUETTELO

1 MÄÄRITYS

Kurssin harjoitustyössä kehitämme sovellusta, jonka avulla voi arvostella yliopiston eri ravintoloiden ruokia. Lisäämme sovellukseen myös muita yliopistoja ja muiden yliopistojen ravintoloita, jotta sovelluksesta saataisiin yleiskäyttöisempi useamman yliopiston ravintoloiden avuksi.

FoodReview sovellukseen toteutetaan eri yliopistojen, arvosteluiden ja käyttäjien tietojen tallentamiseksi tietokanta. Tietokannan avulla tiedot voidaan tallentaa ja niitä voidaan käyttää helposti. Käyttäjät ovat yliopistojen opiskelijoita, joten voidaan sovelluksen tietokannan tulevan käyttäjäkunnan kasvaessa noin tuhansien opiskelijoiden laajuiseksi. Tietokanta ei kuitenkaan toistaiseksi sisällä enempää kuin kolme yliopistoa ja viisi ravintolaa. Tämän vuoksi voidaan ajatella tietokantaa kevyenä.

Tietokannassa pidetään yllä 7 erilaista taulua. Tärkeimpinä tietoina ovat ruokien nimet ja niiden saamat arvostelut. Lisäksi tärkeää on tietää, mihin käyttäjään annettu arvostelu liittyy.

1.1 Käyttäjät

Tietokantaa käyttävät niin ravintoloiden ylläpidosta vastaavat henkilöt kuin yliopiston opiskelijat. Sovelluksen käyttäjänä, opiskelijalla on mahdollisuus tarkastella tulevan 4 viikon ruokalistoja ravintoloista etukäteen. Käyttäjä pystyy antamaan arvosteluja mille tahansa ruualle, kunhan ruoka on tarjolla kyseisenä päivänä. Lisäksi käyttäjä pystyy näkemään tietyn ravintolan tarjoileman ruuan saamien arvostelun keskiarvon. Käyttäjä pystyy myös muokkaamaan tai poistamaan aikaisemman arvostelunsa.

Ylläpitäjällä, joita voidaan tarvittaessa luoda useampia, on oikeus luoda ja poistaa ravintoloita sekä ruokia. Ylläpitäjä pystyy myös poistamaan arvosteluja, joita ylläpitäjä pitää epäasiallisina tai muuten turhina. Ylläpitäjä kykenee myös hallitsemaan kaikkia sovelluksen käyttäjiä. Tähän sisältyy ylläpitäjän oikeuksien myöntäminen ja poistaminen. Ylläpitäjän ominaisuudet eivät näy tavallisille käyttäjille ja arvostelut -välilehti muuttuu sen mukaan, onko kirjautunut käyttäjä ylläpitäjä vai ei. Muuten kaikki muut ominaisuudet ovat samat niin tavallisille käyttäjille kuin ylläpitäjillekin.

1.2 Tietokantakyselyt

Sovellus toimii tietokannan varassa, joten lähes kaikki sovelluksen käsittelemä tieto on siirrettävä tietokannasta ohjelman käyttöön tai päinvastoin. Erilaisia siirrettäviä tietoja ovat niin käyttäjien nimet, yliopistojen sijainnit ja arvostelujen päivämäärät. Näiden tietojen hakemiseen tehdään seuraavat kyselyt:

1) Hae kaikkien yliopistojen tiedot
2) Hae kaikkien ravintoloiden tiedot ja osoitteet sekä lisäksi vierasavainyhteydet yliopistoihin
3) Hae kaikkien ruokien tiedot ja lisäksi vierasavainyhteydet ravintoloihin
4) Hae kaikkien tietylle ruualle annettujen arvosteluiden tiedot ja lisäksi vierasavainyhteydet ravintoloihin ja käyttäjiin.
5) Hae tietyn käyttäjän tekemät kaikki arvostelut
6) Hae tietyn käyttäjän kaikki tiedot (sis. suolan ja salatun salasanan) kirjautumista varten
7) Tarkista ennen ohjelman lataamista, onko käyttäjä, jolla kirjaudutaan ylläpitäjä vai ei
8) Hae kaikki ruuat ja järjestä ne päivämäärän mukaan
9) Hae kaikki tietyn kokin tekemien ruokien arvostelut.

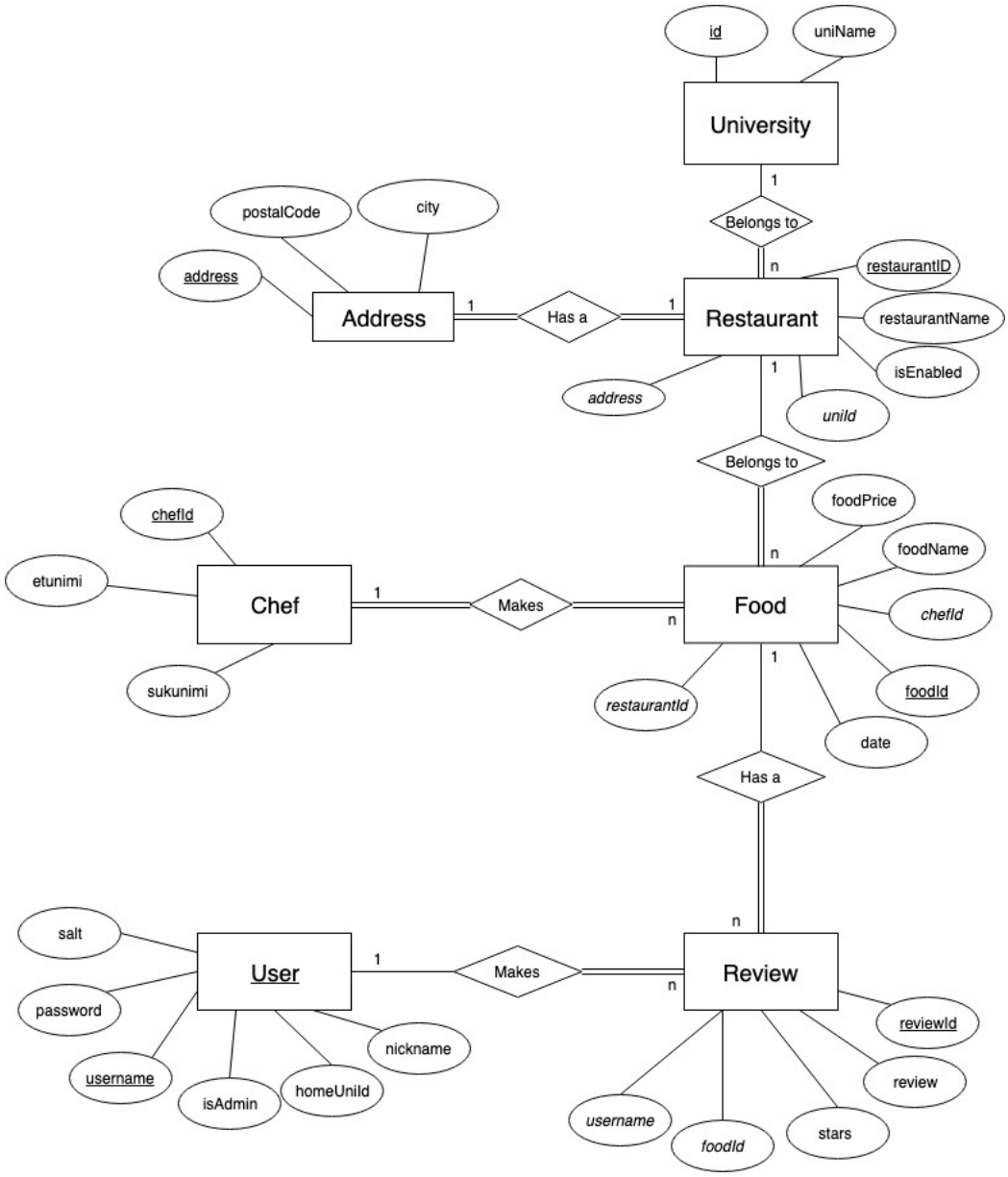
2 MALLINNUS

2.1 Käsitemalli

Lähdimme rakentamaan ensiksi luonnosta tietokannan rakenteesta paperille. Tähän osallistuivat kaikki ryhmän jäsenet yhdessä. Päädyimme tietokantaratkaisuun, joka pyrkii kohdistamaan tiedot jokaisen yksikön kohdalta järkeväksi kokonaisuudeksi. Rakensimme useamman luonnoksen ja karsimme aluksi suuren osan kohdetyyppien attribuuteista pois. Myöhemmin kuitenkin huomasimme erilaisia ominaisuuksia, joita täytyi käyttöliittymän lisäksi päivittää myös tietokantaan. Näistä merkittävin on ravintolan toiminnan ilmaisemiseen liittyvä attribuutti, ”isEnabled”.

Lähes jokaisen kohdetyypin pääavaimet ovat id -nimisiä attribuutteja. Tietokanta hoitaa jokaisen pääavaimen merkitsemisen itsenäisesti. Tästä ainoana poikkeuksena toimii username -attribuutti, joka yksilöi puolestaan jokaisen käyttäjän. Käyttäjänimen käyttäjä voi valita itse luodessaan käyttäjää.

Käsitemallissamme käytetään paljon minimikardinaalisuutta 1. Se tarkoittaa sitä, että tietokantatoteutukseen täytyy lisätä vierasavainyhteyksiin viite-eheyksistä riippuva poistaminen (ON DELETE CASCADE).



Kuva 1: Käsitemalli

University:

Kohdetyyppi sisältää yliopiston nimen ja sen lisäksi yksilöivän pääavaimen, yliopiston id:n. Molemmat arvot ovat lisäksi pakollisia. Yliopisto- kohdetyypillä ei ole muita vierasavainyhteyksiä.

Address:

Kohdetyyppi sisältää id:n lisäksi katuosoitteen, postinumeron ja kaupungin. Jokainen arvo on pakollinen, id:n toimiessa pääavaimena kohdetyypillä.

Restaurant:

Restaurant -kohdetyyppi sisältää useita attribuutteja, joista muutamat ovat vierasavainyhteyksiä. Ravintolan omat attribuutit ovat ravintolan nimi, id ja toiminnallisuuden merkintä. Id toimii pääavaimena, ja muutkin attribuutit ovat pakollisia. Lisäksi isEnabled -attribuutti on tietotyyppiä Boolean, joten tietokantaa toteuttaessa täytyy tarkistaa, että arvo on joko 1 tai 0.
AddressId on ensimmäinen vierasavain ravintola-kohdetyypille. AddressId viittaa siis suoraan Address -taulun attribuuttiin addressId. AddressId:n tulee myös olla merkitty pakolliseksi, koska ravintola tulee aina tarvitsemaan osoitteen.
UniversityId on toinen vierasavain ravintolalle. Nimensä mukaisesti, universityId viittaa University -kohteen arvoon universityId. Kuten addressId:n kohdalla, tämän arvon tulee olla merkitty pakolliseksi, jotta jokainen ravintola liittyy johonkin yliopistoon.

Chef:
Kokki -kohdetyyppi sisältää kokin nimen, eriteltynä etu- ja sukunimeen. Lisäksi jokainen kokki on yksilöllisesti tunnistettavissa chefId -attribuutin avulla. Kokkia lisättäessä, jokaisen arvon tulee olla määritetty.

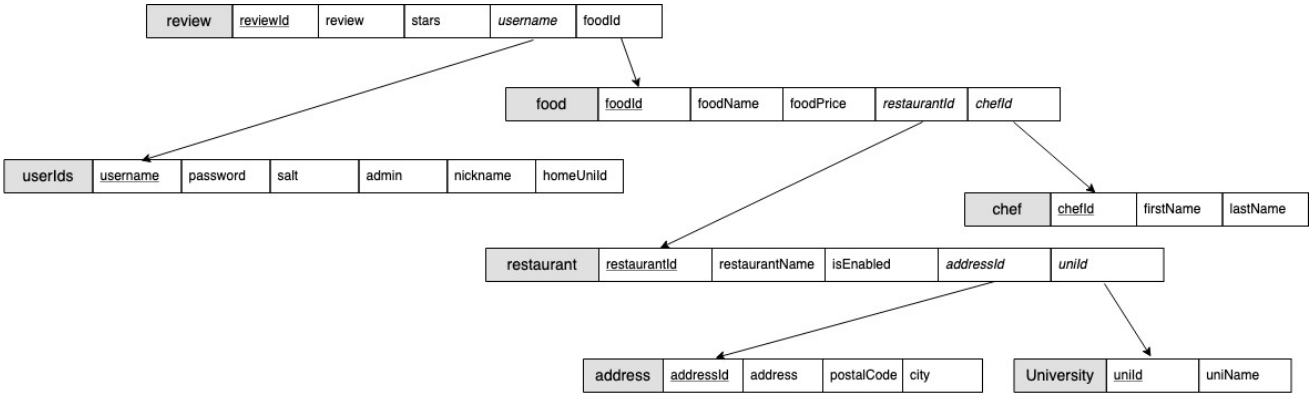
Food:
Ruoka -kohdetyypin omat attribuutit ovat ruuan id, nimi, päivämäärä ja hinta. Jokainen ruuan oma attribuutti on myös pakollinen. Vierasavainyhteyksinä tässä kohdetyypissä toimivat ravintolan id ja kokin id. Ravintolan id on pakollinen, jotta voidaan sitoa ruoka johonkin ravintolaan. Kokki tosin halutaan pitää avoimena, sillä ratkaisussamme kokkia (tai keittiömestaria) ei välttämättä tunneta.

User:
User -kohdetyyppi sisältää kaikki käyttäjät. Jokaisella käyttäjällä on käyttäjänimi, nimimerkki, suojattu salasana ja salasanan suojaamiseen käytetty ”suola”. Lisäksi käyttäjällä on attribuutti, joka kertoo käyttöjärjestelmälle, onko käyttäjä ylläpitäjä. Jokainen käyttäjän attribuuteista on pakollinen. Sen lisäksi ylläpitäjän statuksen kertova attribuutti ”isAdmin” on tietotyyppiä boolean. Tästä arvosta on tarkistettava, onko arvo 1 tai 0.

Review:
Tämä kohdetyyppi on toiminnallisuuksiltaan ja vierasavainyhteyksiltään kaikista monimutkaisin. Arvostelu -kohdetyypillä on attribuutit id, sanallinen arvio ja tähtien määrä. Jokainen näistä on jälleen pakollinen. Vierasavaimina tulevat tauluun käyttäjänimi ja arvostellun ruuan id. Molemmat

näistä attribuuteista merkitään myös pakollisiksi, jotta voidaan sitoa arvostelut oikeisiin käyttäjiin ja ruokiin.

2.2 Relaatiomalli



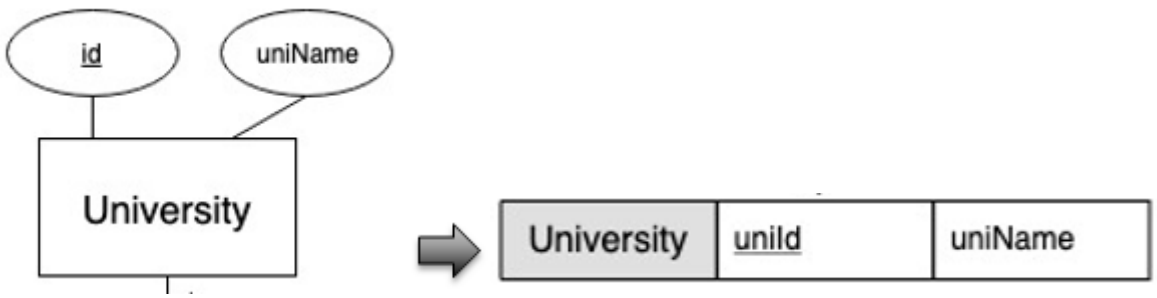
Kuva 2: Relaatiokaavat

ER-mallinnuksen jälkeen, muunsimme ER-mallin relaatiomuotoon. Lopullisessa versiossa relaatiomalli näyttää jokaisen taulun pääavaimen sekä tarvittavat vierasavainyhteydet. ER-kaaviossa olimme jo miettineet etukäteen kardinaliteettejä, vierasavainyhteyksiä ja viittauksia, joten kaavion muuntaminen ei ollut vaikeaa, eikä vaatinut merkittäviä muutoksia suunniteltuun muotoon.

Suurin osa käsitemallin kohdetyypeistä on vahvoja kohdetyyppejä. Se tarjoaa mahdollisuuden transformoida kaavio melko yksinkertaisilla säännöillä. Säännöt ja niiden numerot ovat JYU:n luentomonisteesta.

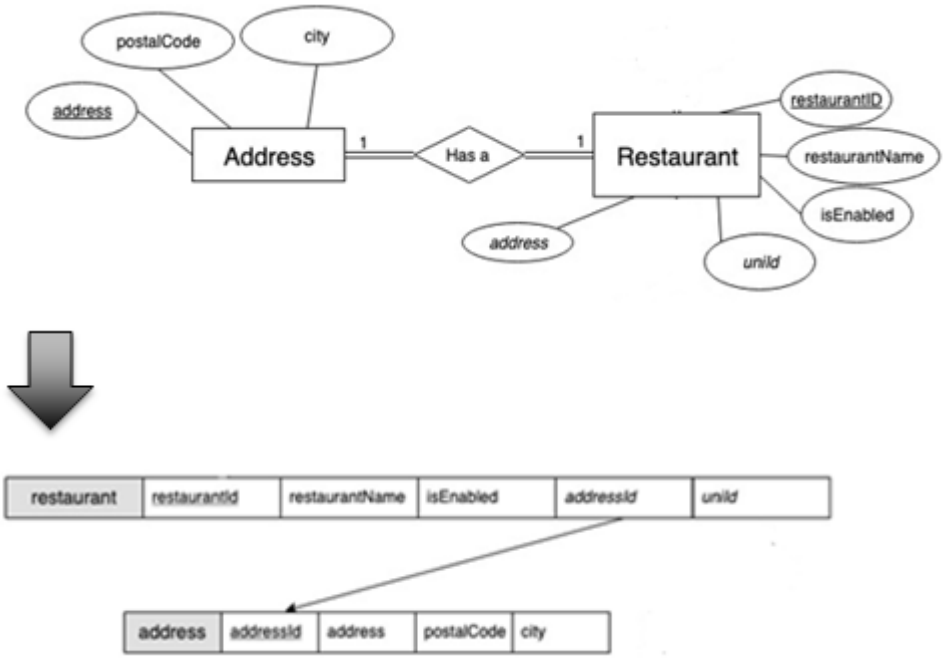
Transformointisäännöt:

Kuten kaaviosta voidaan huomata, jokainen kohdetyyppi voitiin transformoida käyttäen sääntöä numero 1; “jokaisesta vahvasta kohdetyypistä tehdään oma, ns. kohderelaatio.". Esimerkiksi:



Suhdetyypit transformoitiin käyttämällä sääntöjä 6 ja 7. Sääntöä 6, “jokaisesta binäärisestä 1:1-suhdetyypistä sijoitetaan toisen kohdetyypin avainattribuutit viiteavaimeksi toisesta kohdetyypistä

muodostettuun relaatioon", käytettiin transformoitaessa ravintolan ja osoitteen suhdetta, sillä jokaiseen ravintolaan liittyy tässä tilanteessa vain yksi osoite. Samaa sääntöä käytettiin myös luodessa ruuan ja kokin suhdetta. Esimerkiksi:



Kaikki muut suhdetyypit ovat 1:N suhdetyyppejä, jolloin voitiin käyttää kaikkiin muihin suhdetyyppeihin sääntöä numero 7; "jokaisesta binäärisestä 1:N-suhdetyypistä sijoitetaan 1:n puoleisen kohdetyypin avainattribuutit viiteavaimeksi N:n puoleisesta kohdetyypistä muodostettuun relaatioon". Esimerkiksi meidän mallissamme University-kohdetyypin pääavain sijoitetaan Restaurant-kohdetyypin vierasavaimeksi. Samoin myös Restaurant-kohdetyypistä Food-kohdetyyppiin, Chef-kohdetyypistä Food-kohdetyyppiin, Food-kohdetyypistä Review-kohdetyyppiin, sekä User-kohdetyypistä Review-kohdetyyppiin.

3 TIETOKANTATOTEUTUS

3.1 Yleistä

Transformoitua kaaviota tietokannaksi toteuttaessa huomasimme haasteita liittyen erityisesti tietotyyppeihin. SQLite 3:ssa on käytössä vain 5 tietotyyppiä. Tietotyyppejä käsiteltäessä, aloimme kuitenkin yhtenäistää tietotyyppien käyttöä ja käytimme tietokannassamme laajasti niin kokonaislukuja, tekstiä kuin liukulukujakin. Päivämääriä merkittäessä, ajattelimme aluksi tallentavamme päivämäärän eri muuttujiin, mutta lopuksi päädyimme käyttämään vain merkkijonoja. Niin saimme kaiken päivämäärätiedon yhden attribuutin alle.

Toisen haasteen kohtasimme, kun aloimme rakentaa salasanan salausta. Salasana salataan jollakin ”suolalla”, eli tietyn pituisella jonolla sattumanvaraisia tavuja. Tavujen tallentaminen tietokantaan oli aluksi haastavaa, mutta onnistuimme tallentamaan salaukseen käytetyn ”suolan” tietotyypiksi BLOB. Tämän jälkeen salasanan tarkistus ja luonti sujuivat luontaisesti.

Teimme tietokantaan kuitenkin myös merkittäviä muutoksia. Huomasimme sovellusta kehittäessämme, että käyttäjällä olisi hyvä olla jokin käyttäjänimestä eroava nimi, jota voitaisiin käyttää näyttönimenä tai nimimerkkinä. Tästä johtuen loimme ”käyttäjä” -kohdetyypille uuden attribuutin ”nickname”. Toinen merkittävä muutos kohdistui ravintolan statukseen. Ravintola sisältää suuren määrän tietoa, joten olisi järkevää, ettei kaikkea tietoa tarvitsisi luoda uudelleen, mikäli ravintola esimerkiksi remontoidaan. Tähän loimme ”isAdmin” kaltaisen boolean -arvon, jolla jokaisesta ravintolasta käy selväksi se, onko ravintola tällä hetkellä avoinna vai suljettuna. ”Restaurant” -kohdetyypille loimme lisäksi siis myös uuden arvon ”isEnabled”.

3.2 Toteutustapa

Toteutimme FoodReview -nimisen sovelluksen, joka toimii myös käyttöliittymänä tässä harjoitustyössä kuvatulle tietokannalle. Sovellus on kehitetty käyttäen Android Studio -kehitysympäristöä ja pohjautuu täysin Java -ohjelmointikielelle. SQL toteutus on tehty käyttäen Android Studion sisäänrakennettua SQLite3 toiminnallisuutta.

Sovellus tekee ensin ns. pohjadatan sovelluksen käytettäväksi. Tämän jälkeen pohjadatana lisätty ensimmäinen ylläpitäjä voi muokata lähes kaikkea tietoa sovelluksen sisällä.

3.3 Testauksessa huomattua

Sovellusta testatessamme huomasimme useampaan otteeseen, että muutokset, joita tehtiin tietokannan attribuutteihin tai skeemaan, vaativat tietokannan täydellisen nollaamisen. Tietokannan toimivuus voitiin testata vasta tämän jälkeen.

3.4 Ryhmän jäsenten työnjako

Sovellusta kehittäessämme, emme jakaneet vastuita aluksi lähes ollenkaan. Myöhemmin huomasimme, että työnjakoa on, ja se on tapahtunut aika lailla itsestään. Markus keskittyi pääsääntöisesti eri näkymien tekemiseen, joita Android Studiossa nimitetään activity -nimellä. Aarne keskittyi myös käyttöliittymän toteuttamiseen, mutta erilaisten ”Fragment” popup -ikkunoiden osalta. Tomi puolestaan keskittyi sovelluksen taustalla toimivaan tietokantaan. Raportti tehtiin suurilta osin yhdessä.

Suunnittelimme kuitenkin sovelluksen ulkonäköä, tietokantaa ja toteutustapaa pääsääntöisesti ryhmänä. Kesäleirin luokassa pystyimme helposti kommunikoimaan ja sopimaan asioita yhteisesti. Loimme myös whatsapp -ryhmän, jonka kautta pystyimme jatkamaan ryhmänä työskentelyä kesäleirin loppumisen jälkeen.

3.4.1 Markus

Markus oli mukana tietokannan suunnittelussa, mutta itse toteutusvaiheen hoiti pääosin Tomi. Tämän lisäksi Markus hoiti tietokannan metodien implementoinnin applikaation aktiviteetteihin.

3.4.2 Aarne

Vaikkakin Tomi loi suurimman osan tietokantaosuudesta, Aarne auttoi Tomia transformoimalla ER-mallin relaatiomuotoon, jolloin Tomi sai luotua itse tietokannan helposti. Aarne, oman varsinaisen hommansa ohella, myös korjasi tietokannan koodia Tomin ohjeistuksella, kun Tomi oli kiireinen, eikä itse päässyt koodin ääreen, sekä Aarne implementoi tietokantaa käsittelevät metodit applikaatioomme.

3.4.3 Tomi

Tomi keskittyi yhteisessä harjoitustyössä pääsääntöisesti tietokantoihin. Kun tietokanta oli yhdessä suunniteltu, Tomin oli helppo toteuttaa tietokanta .sql -tiedostoon, ja siitä ohjelmaan. Tomi teki myös databaseManager -nimisen luokan, joka hallitsee kaikkea yhteyttä käyttöliittymän ja tietokannan välillä.

4 KESKUSTELU

Tämä projekti oli suurin projekti, mitä olemme tähän mennessä tehneet, sekä tämä oli samalla meille ensimakua ohjelmistotuotantoon ryhmässä. Tietokannan sisällyttäminen ohjelmaan näytti hyvin tietokannan keskeisyyden ja tärkeyden.

Opimme työtä tehdessämme paljon ryhmässä toimimisesta ja töiden jakamisesta. Saimme myös kiinni aikataulutuksesta suurempaa projektia tehdessä.

4.1 Pisteytys

Mielestämme olemme noudattaneet tehtävänantoa tarkasti, rakentaneet koodista toimivan, laajennettavan ja ennen kaikkea noudattaneet käyttöliittymän rakentamisessa olio-ohjelmoinnin hyviä käytänteitä ja periaatteita. Rakensimme tietokannan vastaamaan sovelluksen tarpeita ja käytimme suunnittelussa ER-kaaviota ja transformointia hyväksi.

Rakensimme sovellukseen toiminnallisuuksia kaikkein laajimpien vaatimusten mukaisesti ja täytimme myös kaikki ryhmätyön tuomat lisävaatimukset. Ajattelisimme, että työ on täyden 40:n pisteen arvoinen.