

CS-A1150  
2023

Tietokannat

Harjoitustyö, osa 1 (UML-mallinnus ja relaatiomalli)

Harjoitustyö, osa 2 (SQL)

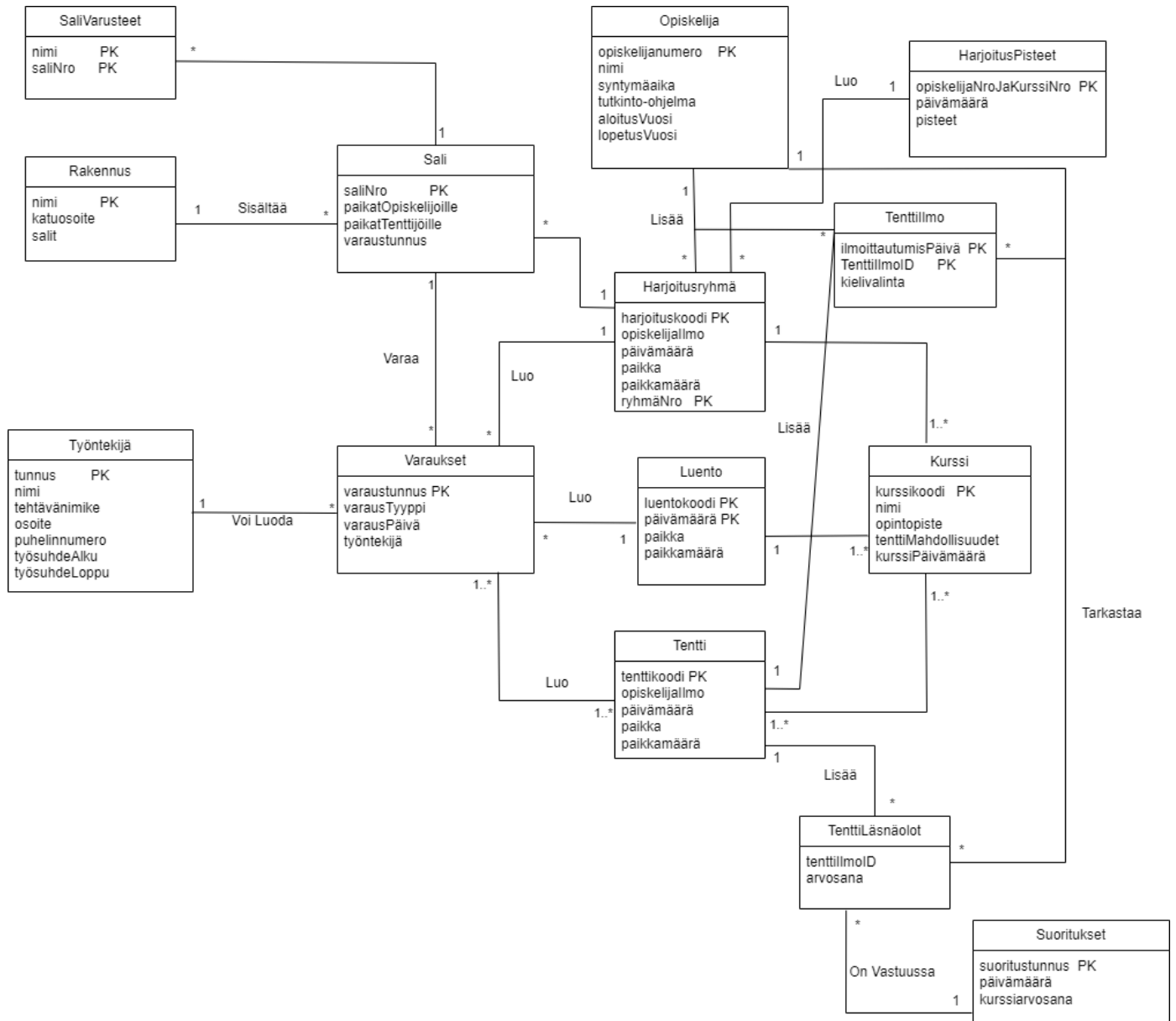
Ryhmän jäsenet:

Abdi-Ra'uf Salah [abdi-rauf.salah@aalto.fi](mailto:abdi-rauf.salah@aalto.fi)

Mahathir Mohammad Kabir [mahathir.mohammadkabar@aalto.fi](mailto:mahathir.mohammadkabar@aalto.fi)

Daud Abucar [daud.abucar@aalto.fi](mailto:daud.abucar@aalto.fi)

## 1. UML-kaavio



## 2. UML-kaaviota vastaavat relaatiokaaviot

Opiskelija(opiskelijanumero, nimi, syntymäaika, tutkinto-ohjelma, aloitusVuosi, lopetusVuosi)

Työntekijä(tunnus, nimi, tehtävänimike, osoite, puhelinnumero, työsuhdeAlku, työsuhdeLoppu)

Rakennus(nimi, katuosoite, salit)

Sali(saliNro, paikatOpiskelijoille, paikatTenttijöille, varaustunnus)

Varaukset(varaustunnus, varausTyyppi, varausPäivä, työntekijä)

Tentti(kurssikoodi, opiskelijaNro, päivämäärä, paikka, paikkamäärä)

Luento(kurssikoodi, päivämäärä, paikka, paikkamäärä)

Harjoitusryhmä(kurssikoodi, ryhmäNro, opiskelijaNro, päivämäärä, paikka, paikkamäärä)

TenttiIlmo(ilmoittautumisPäivä, kurssikoodi, opiskelijaNro, kielivalinta)

Kurssi(kurssikoodi, nimi, opintopiste, tenttiMahdollisuudet, kurssiPäivämäärä)

TenttiLäsnäolot(kurssiNro, opiskelijaNro, arvosana)

Suoritukset(opiskelijanumero, päivämäärä, kurssiarvosana, kurssiNro)

HarjoitusPisteet(opiskelijanumero, kurssikoodi, päivämäärä, pisteet)

### 3. Selostus

UML-mallimme on tätä muotoa, sillä ideoitii että työntekijä luo varauksia erilaisille saleille ja tarkoituksille kuten harjoitusryhmille, luennoille tai tenteille. Salit sisältävät Rakennus-relaatioon, josta saamme osoitteen tarvittaessa. Salissa olevat varusteet kuten videotykit, tietokoneet yms. ovat saliVarusteet-relaatioissa, jossa saliNro yhdistää varusteet sekä Sali-relaatiot.

Varausta luodessa luodaan samalla se, mihin käyttötärpeeseen kyseinen varaus on, onko se tenttiä, harjoitusryhmää vai luentoa varten. Kun käyttötärve tiedetään, voimme luoda monikon johonkin näistä relaatioista (harjoitusryhmään, luentoon, tenttiin). Käyttötärpeen avulla voimme yhdistää Sali-relaatiosta attribuutin paikatOpiskelijoille harjoitusryhmä- sekä luentorelaatioiden paikkamäärä-attribuutteihin. Sama pätee paikatTenttijöille attribuuttiin relaatioissa Sali ja paikkamäärä-attribuutti Tentti-relaatioissa. Varausta luodessa luodaan myös tietokantaan Varaukset-relaatioon uusi varaus-monikko, jossa on yksikäsitteinen varaustunnus. Tämän avulla me voimme tarkistaa historiatietoa, eli selvittää, mille harjoitusryhmälle sali oli varattu esimerkiksi viime vuonna määrättyyn aikaan. Luodessa Harjoitusryhmä-relaation monikon, lisäämme attribuutteihin päivämäärän, jonka avulla harjoitusryhmien kokoontumisajat voivat vaihdella.

Nyt olemme toteuttaneet kolme osaa, jotka liittyvät kurssiin, ja voimme liittää kaikki nämä relaatiot Kurssi-relaatioon käyttämällä relaatioiden yhteistä attribuuttia *kurssikoodia*. Tämä on kätevää, koska voimme katsoa samalla, mahtuuko saliin opiskelijoita. Sanotaan, että esimerkiksi luomme varauksen klo 12.00 03.03.2023 saliin A111 tenttiä varten, joten paikkoja on noin 200/2, sillä opiskelijat istuvat joka toisessa paikassa. Luodessa uusia varauksia tarkastamme minkä tyyppinen varaus on ja onko sali varattu (salivaraukset voimme tarkastaa joko Luento-, Harjoitusryhmä- tai Tentti-relaatioista, jotka sisältävät tarvittavan tiedon tarkastelua varten kurssikoodi, paikka ja päivämäärä yms.). Jos haluamme nyt luoda tälle päivälle samaan saliin minkään muun varauksen kuin tenttivarauksen, varausta ei voida luoda. Mutta jos varaus on tenttivaraus, tarkastetaan, onko kaikki 200/2 paikkaa täytetty, ja jos ei ole niin luodaan kyseinen varaus. Sama pätee toisinpäin, eli jos luomme varauksen seuraavalle päivälle klo 12 04.03.2023 saliin A1010 harjoitusryhmää tai luentoa varten, tälle tilalle tähän kellonaikaan ja päivämäärään ei voida tehdä muita varauksia.

Opiskelija voi ilmoittautua kurssille ilmoittautumalla harjoitusryhmään. Harjoitusryhmä-avaimella saadaan tiedot, ketkä opiskelijoista ovat ilmoittautuneet kurssille.

Kun opiskelija ilmoittautuu tenttiin, ilmoittautuminen välttyy TenttiIlmo-relaatiosta Tenttiläsnäolo-relaatioon, josta tietokanta tarkastaa kuinka monta kertaa opiskelija on ollut tentissä läsnä. Tarkastus toimii siten, että katsotaan, onko opiskelija ollut yli kahdessa tentissä läsnäolijana

ja on saanut hyväksytyn arvosanan tentistä, jos näin on, opiskelijan TenttiIlmo-relaatiossa oleva monikko ei lisää tietoa Tentti-relaatioon.

Järjestelmä pitää tiedon kaikista tenteistä, joihin opiskelija on ilmoittautunut ja erikseen, onko tentti suoritettu hyväksytyksi tai hylätyksi (ne missä opiskelija on ollut läsnä). Järjestelmä pitää tiedot kaikesta, mitä ohjeissa oli haluttu pitää eikä yhtään enempää.

Esimerkiksi Opiskelija xx, jonka opiskelijanumero on 0000 haluaa ilmoittautua kurssille MS-1000. Järjestelmä katsoo ensiksi TenttiIlmo-relaation kautta TenttiLäsnäolo-relaation, ja katsoo kurssikoodin ja opiskelijaNro avulla, onko kyseisellä opiskelijalla yli kaksi hyväksyttyä tenttikertaa. Kun opiskelija ei ole suorittanut kyseisen kurssin tenttiä yli kaksi kertaa hyväksytysti, niin opiskelija voi ilmoittautua harjoitusryhmään, jos siellä on tilaa. Kun harjoitusryhmässä on tilaa ja opiskelija on hyväksytty kurssille, hänelle luodaan monikko HarjoitusPisteet-relaatioon. Opiskelija ilmoittautuu itse haluamaansa tenttiin TenttiIlmo-relaation kautta.

## 4. Funktionaaliset riippuvuudet ja normaalimuotoisuus

Nyt tutkitaan relaatioiden voimassa olevia riippuvuuksia ja vasemman puolen sulkeumia.

Tutkitaan ensin **Opiskelija** relaation funktionaalista riippuvuutta:

opiskelijanumero  $\rightarrow$  nimi syntymäaika tutkinto-ohjelma aloitusVuosi lopetusVuosi

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{opiskelijanumero}\}^+ = \{\text{opiskelijanumero, nimi, syntymäaika, tutkinto-ohjelma, aloitusVuosi, lopetusVuosi}\}$

**Sulkeuma sisältää kaikki relaation Opiskelija attribuutit, joten attribuutti *opiskelijanumero* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Työntekijä** relaation funktionaalista riippuvuutta:

tunnus  $\rightarrow$  nimi tehtävänimike osoite puhelinnumero työsuhdeAlku työsuhdeLoppu

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{tunnus}\}^+ = \{\text{tunnus, nimi, tehtävänimike, osoite, puhelinnumero, työsuhdeAlku, työsuhdeLoppu}\}$

**Sulkeuma sisältää kaikki relaation Työntekijä attribuutit, joten attribuutti *tunnus* on sen avain, ja relaatio on BCNF:ssä**

Jatketaan seuraavaksi **Rakennus** relaation funktionaaliseen riippuvuuteen:

nimi  $\rightarrow$  katuosoite salit

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{nimi}\}^+ = \{\text{nimi, katuosoite, salit}\}$

**Sulkeuma sisältää kaikki relaation Rakennus attribuutit, joten attribuutti *nimi* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Sali** relaation funktionaalista riippuvuutta:

$\text{saliNro} \rightarrow \text{paikatOpiskelijoille paikatTenttijöille varaustunnus}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{salinro}\}^+ = \{\text{saliNro}, \text{paikatOpiskelijoille}, \text{paikatTenttijöille}, \text{varaustunnus}\}$

**Sulkeuma sisältää kaikki relaation Sali attribuutit, joten attribuutti *saliNro* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Varaukset** relaation funktionaalista riippuvuutta:

$\text{varaustunnus} \rightarrow \text{varausTyyppi varausPäivä työntekijä}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{varaustunnus}\}^+ = \{\text{varaustunnus}, \text{varausTyyppi}, \text{varausPäivä}, \text{työntekijä}\}$

**Sulkeuma sisältää kaikki relaation Varaukset attribuutit, joten attribuutti *varaustunnus* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Tentti** relaation funktionaalista riippuvuutta:

$\text{kurssikoodi} \rightarrow \text{opiskelijaNro päivämäärä paikka paikkamäärä}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{kurssikoodi}\}^+ = \{\text{kurssikoodi}, \text{opiskelijaNro}, \text{päivämäärä}, \text{paikka}, \text{paikkamäärä}\}$

**Sulkeuma sisältää kaikki relaation Tentti attribuutit, joten attribuutti *kurssikoodi* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Luento** relaation funktionaalista riippuvuutta:

$\text{Kurssikoodi} \rightarrow \text{päivämäärä paikka paikkamäärä}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{kurssikoodi}\}^+ = \{\text{kurssikoodi}, \text{päivämäärä}, \text{paikka}, \text{paikkamäärä}\}$

**Sulkeuma sisältää kaikki relaation Luento attribuutit, joten attribuutti *kurssikoodi* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Harjoitusryhmä** relaation funktionaalista riippuvuutta:

kurssikoodi  $\rightarrow$  ryhmäNro opiskelijaNro päivämäärä paikka paikkamäärä

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{kurssikoodi}\}+ = \{\text{kurssikoodi, ryhmäNro, opiskelijaNro, päivämäärä, paikka, paikkamäärä}\}$

**Sulkeuma sisältää kaikki relaation Harjoitusryhmä attribuutit, joten attribuutit *kurssikoodi* ja *ryhmäNro* ovat sen avaimia, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **TenttiIlmo** relaation funktionaalista riippuvuutta:

ilmoittautumisPäivä  $\rightarrow$  kurssikoodi opiskelijaNro kielivalinta

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{ilmoittautumisPäivä}\}+ = \{\text{ilmoittautumisPäivä, kurssikoodi, opiskelijaNro, kielivalinta}\}$

**Sulkeuma sisältää kaikki relaation TenttiIlmo attribuutit, joten attribuutti *ilmoittautumisPäivä* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Kurssi** relaation funktionaalista riippuvuutta:

kurssikoodi  $\rightarrow$  nimi opintopiste tenttiMahdollisuudet kurssiPäivämäärä

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{kurssikoodi}\}+ = \{\text{kurssikoodi, nimi, opintopiste, tenttiMahdollisuudet, kurssiPäivämäärä}\}$

**Sulkeuma sisältää kaikki relaation Kurssi attribuutit, joten attribuutti *kurssikoodi* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **TenttiLäsnäolo** relaation funktionaalista riippuvuutta:

kurssiNro  $\rightarrow$  opiskelijaNro arvosana

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{kurssiNro}\}+ = \{\text{kurssiNro, opiskelijaNro, arvosana}\}$



**Sulkeuma sisältää kaikki relaation TenttiLäsnäolot attribuutit, joten attribuutit *kurssiNro* ja *opiskelijaNro* ovat sen avaimia, ja relaatio on BCNF:ssä**

Tutkitaan seuraavaksi **Suoritukset** relaation funktionaalista riippuvuutta:

$\text{opiskelijanumero} \rightarrow \text{päivämäärä kurssiarvosana kurssiNro}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{opiskelijanumero}\}^+ = \{\text{opiskelijanumero, päivämäärä, kurssiarvosana, kurssiNro}\}$

**Sulkeuma sisältää kaikki relaation Suoritukset attribuutit, joten attribuutti *opiskelijanumero* on sen avain, ja relaatio on BCNF:ssä**

Tutkitaan viimeiseksi **HarjoitusPisteet** relaation funktionaalista riippuvuutta:

$\text{opiskelijanumero} \rightarrow \text{kurssikoodi päivämäärä pisteet}$

Tällöin vasemman puolen sulkeumaksi saadaan:

$\{\text{opiskelijanumero}\}^+ = \{\text{opiskelijanumero, kurssikoodi, päivämäärä, pisteet}\}$

**Sulkeuma sisältää kaikki relaation HarjoitusPisteet attribuutit, joten attribuutti *opiskelijanumero* on sen avain, ja relaatio on BCNF:ssä.**

## Ensimmäisen osan palautuksen jälkeen tehdyt muutokset

Kun luokan HarjoitusPisteet avaimen halutaan liittää toisten luokkien avainattribuutteja, loimme assosiaatioiden yhteyteen liittävä PK-laatiko. Kurssi ja kurssikerta omiksi tauluiksi, jotta voidaan kertoa, että tentti liittyy kurssiin, mutta luento ja harjoitusryhmä tiettyyn kurssikertaan.

Salit attribuutti poistettu rakennuksesta, ja sali tauluun lisättiin saliRakennus attribuutti. Jotta voimme katsoa mihin rakennukseen sali liittyy ja mitä saleja kunkin rakennus sisältää. Opiskelija ja Suoritukset taulujen välille assosiaatio (opiskelijanumero molemmissa).

Muodostimme uuden taulun työntekijöiden varauksista nimeltä **TyöntekijäVaraus** jotta voimme luoda varauksen ja työntekijän välille assosiaation.

## Ensimmäisen osan tietokannat SQL-muodossa

Työntekijä taulu:

```
CREATE TABLE Työntekijät(  
tunnus INTEGER PRIMARY KEY,  
nimi CHAR(100) NOT NULL,  
tehtävänimike TEXT,  
osoite TEXT,  
puhelinnumero INTEGER,  
työsuhdeAlku CHAR(10),  
työsuhdeLoppu CHAR(10)  
);
```

**Selitys:**

Tunnuksen tietotyyppi on Integer, koska se sisältää vain numeroita, kuten myös puhelinnumero. Tunnus toimii avaimena. Nimi koostuu vain kirjaimista eli tietotyyppiksi on valittu char(100) (ehkä TEXT olisi ollut järkevämpää sillä nyt työntekijän nimen pituus on maksimissaan 100 merkkiä pitkä). TyösuhdeAlku

ja työsuhdeloppu ovat päivämääriä, tietotyyppiä valittiin Char(10) sillä päivämäärä on 10 merkkiä pitkä, esimerkiksi '10.05.2023'. Tehtävänimikkeellä ja osoitteella ei ole määritelty pituutta, joten näiden tietotyyppi on Text.

**Varaukset taulu:**

```
CREATE TABLE Varaukset(  
    varaustunnus INTEGER PRIMARY KEY,  
    varausTyyppi CHAR(10),  
    varausPäivä CHAR(10)  
);
```

**Selitys:**

Varaus-taulu sisältää pääavaimen varaustunnuksen joka on kokonaisluku sillä tunnukset ovat kokonaislukuja, varaustyyppi char joka on 10 merkkiä pitkä sillä varaustyyppi on 'tunti', 'luento' tai 'harjoitus'. Ja lopuksi varausPäivä joka on 10 merkkiä pitkä sillä päivämäärä on muotoa '01.01.2000' aina.

Varaustunnus on myös kokonaisluku sillä se liitetään TyöntekijäVaraus taulun avulla työntekijään.

**TyöntekijäVaraus taulu:**

```
CREATE TABLE TyöntekijäVaraus(  
    id INTEGER PRIMARY KEY,  
    työntekijäID INTEGER,  
    varausID INTEGER,  
    FOREIGN KEY (työntekijäID) REFERENCES Työntekijät(tunnus),  
    FOREIGN KEY (varausID) REFERENCES Varaukset(varaustunnus)  
);
```

**Selitys:**

Tässä taulussa yhdistetään työntekijät ja varaukset attribuutit toisiinsa.

TyöntekijäVarauksessa id koostuu numeroista, täten sen tietotyyppi on INTEGER. Id toimii tässä avaimena. TyöntekijäID ja varausID kooostuu pelkästään numeroista eli INTEGER. Assosiaatioita on kaksi. Ensimmäisessä assosioidaan työntekijät relaation ja työntekijävaraus relaation välillä, jossa Toisessa yhdistetään Varaukset ja työntekijävaraus.

**Kurssi taulu:**

```
CREATE TABLE Kurssi(  
    kurssikoodi CHAR(10) PRIMARY KEY,  
    nimi TEXT,  
    opintopiste INTEGER,  
    tenttiMahdollisuus TEXT,  
    kurssiPäivämäärä TEXT  
);
```

**Selitys:**

Taulu **Kurssi** sisältää pääavaimen **kurssikoodin**, joka on määritelty CHAR(10) -tyypiksi eli siihen voi sisältyä enintään 10 merkkiä, "**nimi**" sarake, joka tallentaa kurssin nimen ja se on tietotyyppiä Text, koska se voi olla mikä tahansa merkkijono, "**opintopiste**" sarake, joka tallentaa kurssin opintopistemäärän ja se on INTEGER koska se on kokonaisluku, "**tenttiMahdollisuus**" sarake, joka tallentaa tiedon siitä, kuinka monta tenttimahdollisuutta(eri päiväämääriä) kurssilla on sillä mahdollisia päiväämääriä voi olla enemmän kuin yksi, niin se on tyyppiä TEXT ja lopuksi "**kurssiPäivämäärä**" tallentaa päivämäärän, jolloin kurssi pidetään ja tämä on tyyppiä TEXT, koska se on päivämäärä ja se on muotoa esimerkiksi '09.05.2020' aina.

**Harjoitusryhmä taulu:**

```
CREATE TABLE Harjoitusryhmä(  
    harjoituskoodi CHAR(10) PRIMARY KEY,  
    opiskelijaNro INT,  
    päivämäärä TEXT,  
    paikka TEXT,  
    paikkamäärä INTEGER,  
    ryhmäNro CHAR(5),  
    FOREIGN KEY (harjoituskoodi) REFERENCES Kurssi(kurssikoodi),
```

**FOREIGN KEY (opiskelijaNro) REFERENCES Opiskelija(opiskelijanumero)**

);

**Selitys:**

Tässä taulussa yhdistetään **Kurssi** ja **Opiskelija** attribuutit toisiinsa.

Taulu **Harjoitusryhmä** sisältää pääavaimen **harjoituskoodin**, joka on määritelty CHAR(10) -tyypiksi eli siihen voi sisältyä enintään 10 merkkiä. **Opiskelijanumero** ja **paikkamäärä** ovat molemmat pelkästään numeroita, joten ne ovat molemmat tyyppiä INTEGER. **Paikka** ja **päivämäärällä** ei ole tarkkaa pituutta, joten tähän on käytetty tyyppiä TEXT. Tässä taulussa on kaksi assosiaatioita, joista ensimmäisessä assosioidaan Kurssi ja Harjoitusryhmä taulut ja toisessa assosioidaan Opiskelija ja Harjoitusryhmä taulut.

**HarjoitusPisteet** taulu:

```
CREATE TABLE HarjoitusPisteet(  
    opiskelijaNro INTEGER,  
    kurssikoodi CHAR(10),  
    päivämäärä CHAR(10),  
    pisteet INTEGER,  
    PRIMARY KEY (opiskelijaNro, kurssikoodi),  
    FOREIGN KEY (opiskelijaNro) REFERENCES Opiskelija(opiskelijanumero),  
    FOREIGN KEY (kurssikoodi) REFERENCES Kurssi(kurssikoodi)  
);
```

**Selitys:**

**HarjoitusPisteet** taulussa yhdistetään **Opiskelija** ja **Kurssi** attribuutit toisiinsa. Tämä taulu sisältää **opiskelijaNro(pää avain)** ja **pisteet**, jotka ovat tyyppiä INTEGER, koska ne sisältävät numeroita. **Kurssikoodi(pää avain)** ja **päivämäärä** ovat tyyppiä Char(10), koska päivämäärä on esim muotoa '09.05.2020' eli 10 merkkiä ja **Kurssikoodi** on myös enintään 10 merkkiä. Tässä taulussa esiintyy kaksi assosiaatiota, jotka ovat HarjoitusPisteet ja Opiskelijan välillä sekä Kurssi ja HarjoituPisteet välillä.

**Luento** taulu:

```
CREATE TABLE Luento(  
    luentokoodi CHAR(10),  
    päivämäärä TEXT,
```

**paikka TEXT,**  
**paikkamäärä INTEGER,**  
**PRIMARY KEY(luentokoodi, päivämäärä),**  
**FOREIGN KEY (luentokoodi) REFERENCES Kurssi(kurssikoodi)**  
);

#### **Selitys:**

Taulussa **Luento** yhdistetään **Luento** attribuutit. **Luennolla** on assosiaatio **Kurssin** kanssa. Taululta löytyy **luentokoodi** joka on pää avain ja se on tyyppiä **Char(10)**, koska luento koodi saa olla maksimissaan 10 merkkiä pitkä. Siinä on myös **päivämäärä** ja **paikka** ,jotka ovat tyyppiä TEXT, koska se on pelkkää tekstiä. Taulussa on myös paikkamäärä, joka on tyyppiä INTEGER, koska se kertoo kuinka monta paikkaa on opiskelijoille luennolla.

**Tentti** taulu:

**CREATE TABLE Tentti(**  
**tenttikoodi CHAR(10) PRIMARY KEY,**  
**opiskelijaNro INT,**  
**päivämäärä TEXT,**  
**paikka TEXT**  
**paikkamäärä INTEGER,**  
**FOREIGN KEY (tenttikoodi) REFERENCES Kurssi(kurssikoodi),**  
**FOREIGN KEY (opiskelijaNro) REFERENCES Opiskelija(opiskelijanumero)**  
);

#### **Selitys:**

Taulussa **Tentti** yhdistetään **Kurssi** ja **Opiskelija** attribuutit toisiinsa. Taululla on **tenttikoodi**(pää avain), joka on tyyppiä CHAR(10), koska **tenttikoodi** kuten **kurssikoodikin** on enintään 10 merkkiä. **OpiskelijaNro** ja **paikkamäärä** ovat tyyppiä INTEGER, koska kyse on kokonaisluvusta ja tietystä määrästä asioista. **Päivämäärä** ja **paikka** ovat tyyppiä TEXT, koska kyseessä on pelkkää tekstiä, jolla ei ole tiettyä pituutta. Taulussa esiintyy myös kaksi assosiaatiota, joista ensimmäinen assosioidaan **Tentin** ja **Kurssin** relaatioiden välillä ja toinen tapahtuu **Tentin** ja **Opiskelijan** välillä.

**Sali** taulu:

**CREATE TABLE Sali(**  
**saliNro CHAR(10) PRIMARY KEY,**

**paikatOpiskelijoille INTEGER,**

**paikatTenttijöille INTEGER,**

**varaustunnus INTEGER,**

**saliRakennus TEXT**

);

#### **Selitys:**

**Salissa** on **saliNro**, joka on pää avain ja se on tyyppiä Char(10), koska salin numero saa olla enintään 10 merkkinen jono. **Sali** taulussa esiintyy **paikatOpiskelijoille** ja **paikatTenttijöille**, jotka ilmaisevat paikan määrän ja siksi se on tyyppiä INTEGER. Myös **varaustunnus** on samaa tyyppiä, koska se on tunnus, joka koostuu kokonaisluvuista. Lopuksi, taulussa esiintyy **saliRakennus**, joka on tyyppiä TEXT, koska sen rakennuksen nimen pituutta ei tiedetä.

**saliVarusteet** taulu:

**CREATE TABLE SaliVarusteet(**

**nimi TEXT,**

**saliNro CHAR(10),**

**PRIMARY KEY(nimi, saliNro),**

**FOREIGN KEY (saliNro) REFERENCES Sali(saliNro)**

);

#### **Selitys:**

Taulussa **saliVarusteet** pääavaimena toimii **nimi**(tyyppiä TEXT, koska varusteen nimi ja nimen pituus voi olla mikä tahansa) ja **saliNro**(tyyppiä Char(10), koska kyseessä on 10 merkkinen numero). Taulussa esiintyy myös assosiaatio, jossa assosioidaan **saliVarusteet** ja **Sali** relaatiot toisiinsa.

**Rakennus** taulu:

**CREATE TABLE Rakennus(**

**nimi TEXT PRIMARY KEY,**

**katuosoite TEXT,**

**FOREIGN KEY (nimi) REFERENCES Sali(saliRakennus)**

);

#### **Selitys:**

**Rakennus** taulussa esiintyy kahta TEXT tyyppistä attribuuttia **nimi**(pää avain) ja **katusoite**. Näiden tyyppiä on määritelty näin sen vuoksi, että ne ovat pelkkää tekstiä ja niiden pituutta ei tiedetä. Taulussa esiintyvässä assosiaatiossa assosioidaan Sali ja Rakennus relaatiot.

**Opiskelija** taulu:

```
CREATE TABLE Opiskelija(  
    opiskelijanumero INTEGER PRIMARY KEY,  
    nimi TEXT,  
    syntymäaika CHAR(10),  
    tutkintoOhjelma TEXT,  
    aloitusVuosi CHAR(4),  
    lopetusVUOSI CHAR(4)  
);
```

**Selitys:**

**Opiskelijan opiskelijanumero** on pää avain ja se on tyyppiä INTEGER, koska se sisältää vain nimensä mukaan numeroita. Taulussa esiintyy **aloitusVuosi ja lopetusVuosi**, jotka ovat tyyppiä CHAR(4), sillä vuosi voi olla esim “2023” ja se on 4 merkkiä pitkä. **Syntymäaika** on tyyppiä CHAR(10), koska se on esimerkiksi muotoa “27.01.2003” ja se on 10 merkinen merkkijono. Lopuksi **nimi ja tutkintoOhjelma**, jotka ovat tyyppiä Text, koska se voi olla mikä tahansa teksti ja sen pituus ei ole määritelty.

**TenttiIlmo** taulu:

```
CREATE TABLE TenttiIlmo(  
    ilmoittautumisPaiva CHAR(10) PRIMARY KEY,  
    kurssikoodi CHAR(10) NOT NULL,  
    opiskelijaNro INTEGER,  
    kielivalinta TEXT,  
    FOREIGN KEY (kurssikoodi) REFERENCES Kurssi(kurssikoodi),  
    FOREIGN KEY (opiskelijaNro) REFERENCES Opiskelija(opiskelijanumero)  
);
```



**Selitys:** **TenttiIlmo** taulussa yhdistetään **Kurssi ja Opiskelija** attribuutit toisiinsa. Taulusta löytyy ilmoittautumisPäivä ja kurssikoodi, jotka ovat tyyppiä CHAR(10), koska päivämäärää kirjoitetaan muodossa “19.08.2009” eli 10 merkinen jono ja kurssikoodi on esimerkiksi muotoa “CS-A129A9B”. OpiskelijaNro on INTEGER, koska se sisältä vain numeroita ja numeron pituus voi vaihdella. Lopuksi kielivalinta, joka on tyyppiä TEXT, koska sen pituus voi vaihdella esim. “Suomi”, “Ruotsi” tai “Englanti”, jotka ovat kaikki eri kokoisia. Taulussa on kaksi assosiaatiota: yksi liittyy **TenttiIlmon** ja **Kurssin** välisiin relaatioihin ja toinen **TenttiIlmon** ja **Opiskelijan** välisiin suhteisiin.

**TenttiLasnaolot** taulu:

```
CREATE TABLE TenttiLasnaolot(  
    opiskelijaNro INTEGER,  
    kurssiNro CHAR(10),  
    arvosana INTEGER CHECK(arvosana >= 0 AND arvosana <= 5),  
    PRIMARY KEY(opiskelijaNro, kurssiNro),  
    FOREIGN KEY (kurssiNro) REFERENCES Kurssi(kurssikoodi),  
    FOREIGN KEY (opiskelijaNro) REFERENCES Opiskelija(opiskelijanumero)  
);
```

**Selitys:** **TenttiLasnaolot** taulussa yhdistetään **Kurssi ja Opiskelija** attribuutit toisiinsa (ehkä olisi ollut järkevämpää yhdistää **Tentti** ja **Opiskelija** taulujen attribuutit toisiinsa, sillä meillä on UML:ssä merkitty tällä tavalla). Taulusta löytyy attribuutit **opiskelijaNro** joka on tyyppiä **INTEGER** sillä meillä on **Opiskelija** taulussa määritelty esimerkiksi näin että opiskelijanumero kirjoitetaan muotoon ‘123456789’. Taulu sisältää myös **kurssiNro** attribuutin joka on tyyppiä **CHAR(10)** sillä kurssikoodi on esimerkiksi muotoa “CS-A129A9B” tai ‘A-123’. Lopuksi taulu sisältää attribuutin **arvosana** jossa arvosana on kokonaisluku välillä 0-5. **TenttiLasnaolot** taulun pää avaimiksi olemme määrittäneet **opiskelijaNro** ja **kurssiNro**.

**Suoritukset** taulu:

```
CREATE TABLE Suoritukset(  
    opiskelijanumero INTEGER PRIMARY KEY,  
    kurssikoodi CHAR(10),  
    päivämäärä CHAR(10),
```

**kurssiarvosana** INTEGER CHECK(kurssiarvosana >= 0 AND kurssiarvosana <= 5),  
**FOREIGN KEY** (kurssikoodi) **REFERENCES** Kurssi(kurssikoodi),  
**FOREIGN KEY** (opiskelijanumero) **REFERENCES** Opiskelija(opiskelijanumero)  
);

**Selitys:**

**Suoritukset** taulussa yhdistetään **Kurssi** ja **Opiskelija** attribuutit toisiinsa. Taulussa on attribuutit **opiskelijanumero** ja **kurssiarvosana** (joka on 0-5), jotka ovat tyyppiä INTEGER, koska kyseessä kokonaislukuja. Lopuksi **kurssikoodi** ja **päivämäärä** ovat tyyppiä CHAR(10) koska päivämäärä on esimerkiksi "27.01.2003" ja kurssikoodi on esim. "CS-A129A9B". Taulussa on kaksi assosiaatiota: yksi liittyy **Suorituksen** ja **Kurssin** välisiin relaatioihin ja toinen **Suorituksen** ja **Opiskelijan** välisiin suhteisiin.

## Mihin käyttöön tietokanta olisi?

Tietokannassa voidaan tehdä satoja jopa tuhansia erilaisia hakuja riippuen tietokannassa olevan datan määrästä. Nämä erilaiset haut voivat tyypillisesti koostua esimerkiksi työntekijöiden nimen ja työnimikkeen etsimistä, haku jolla saadaa ne työntekijät, jotka ovat luennoitsijoita/professoreita, haku jolla saadaan kurssikoodi, kurssinimi ja opintopistemäärä tietyllä kurssilla, kaikki kurssit joiden tentit ovat tiettynä päivänä jopa tietyssä salissa, kaikki saliin kohdistuvat varaukset ja niiden varausten tyypit ja jopa haku opiskelijoiden opiskelijanumerot harjoitusryhmissä, joissa on paikkamäärää 25.

Nyt luodaan vaikka näkymä joka yhdistää Kurssi, Harjoitusryhmä ja Opiskelija taulut, niin että se palauttaa kurssinimen, opintopisteet ja niiden opiskelijoiden nimet, jotka ovat ilmoittautuneet kurssille ja harjoitusryhmien päivämäärät:

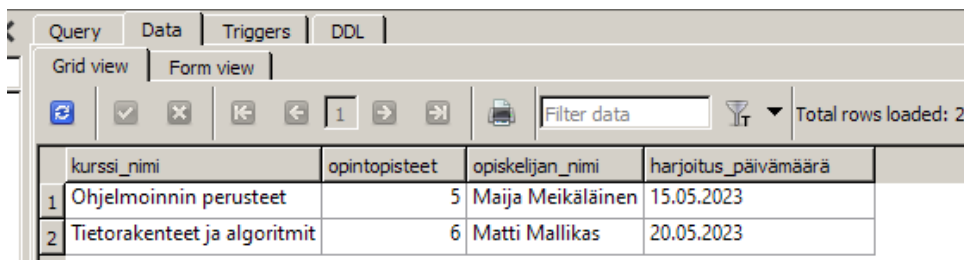
```
CREATE VIEW KurssitJaOpiskelijat AS
```

```
SELECT Kurssi.nimi AS kurssi_nimi, Kurssi.opintopiste AS opintopisteet, Opiskelija.nimi AS  
opiskelijan_nimi, Harjoitusryhmä.päivämäärä AS harjoitus_päivämäärä
```

```
FROM Kurssi
```

```
JOIN Harjoitusryhmä ON Kurssi.kurssikoodi = Harjoitusryhmä.harjoituskoodi
```

```
JOIN Opiskelija ON Harjoitusryhmä.opiskelijaNro = Opiskelija.opiskelijanumero
```



	kurssi_nimi	opintopisteet	opiskelijan_nimi	harjoitus_päivämäärä
1	Ohjelmoinnin perusteet	5	Maija Meikäläinen	15.05.2023
2	Tietorakenteet ja algoritmit	6	Matti Mallikas	20.05.2023

Nyt pistetään käyttötapaukset tulille ja näihin liittyvät SQL-kyselyt.

1. Aloitetaan vaikka kysely joka palauttaa ne varaukset joiden varaustyyppi on harjoitus:

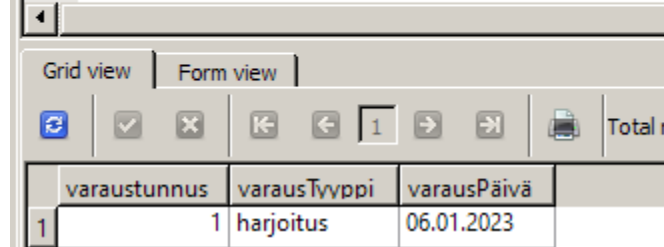
```
SELECT *
```

```
FROM Varaukset
```

WHERE varausTyyppi = 'harjoitus';

Saamme taulun muotoa:

```
227 SELECT *
228 FROM Varaukset
229 WHERE varausTyyppi = 'harjoitus';
```



	varautunnus	varausTyyppi	varausPäivä
1	1	harjoitus	06.01.2023

2. Seuraavaksi tehdään kysely joka palauttaa työntekijät, niiden varaukset ja varaus päivämäärän:

SELECT Työntekijät.nimi, Varaukset.varausTyyppi, Varaukset.varausPäivä

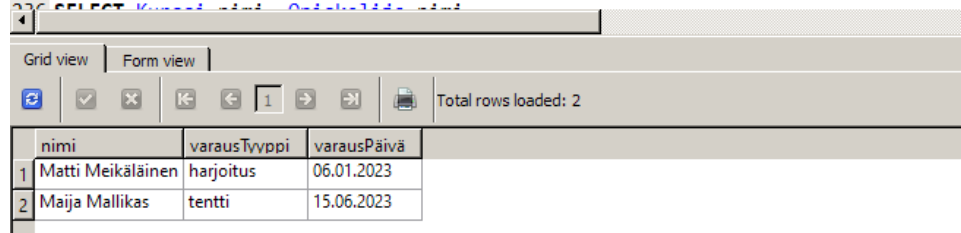
FROM Työntekijät

JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID

JOIN Varaukset ON TyöntekijäVaraus.varausID = Varaukset.varautunnus;

Kysely palauttaa taulun muotoa:

```
231 SELECT Työntekijät.nimi, Varaukset.varausTyyppi, Varaukset.varausPäivä
232 FROM Työntekijät
233 JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID
234 JOIN Varaukset ON TyöntekijäVaraus.varausID = Varaukset.varautunnus;
235
```



	nimi	varausTyyppi	varausPäivä
1	Matti Meikäläinen	harjoitus	06.01.2023
2	Maija Mallikas	tentti	15.06.2023

3. Kysely jossa haemme listan kaikista kursseista ja niiden opiskelijoista:

SELECT Kurssi.nimi, Opiskelija.nimi

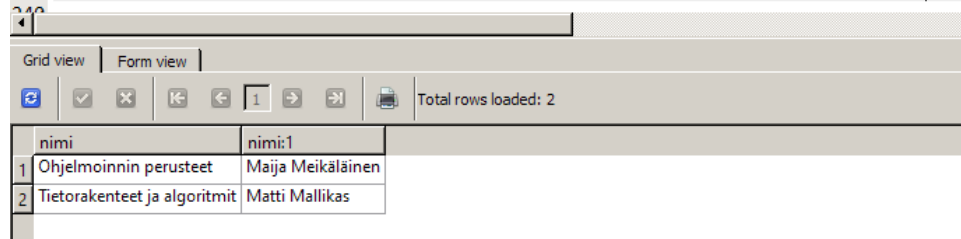
FROM Kurssi

JOIN Harjoitusryhmä ON Kurssi.kurssikoodi = Harjoitusryhmä.harjoituskoodi

JOIN Opiskelija ON Harjoitusryhmä.opiskelijaNro = Opiskelija.opiskelijanumero;

Kysely palauttaa taulun muotoa:

```
236 SELECT Kurssi.nimi, Opiskelija.nimi
237 FROM Kurssi
238 JOIN Harjoitusryhmä ON Kurssi.kurssikoodi = Harjoitusryhmä.harjoituskoodi
239 JOIN Opiskelija ON Harjoitusryhmä.opiskelijaNro = Opiskelija.opiskelijanumero;
```



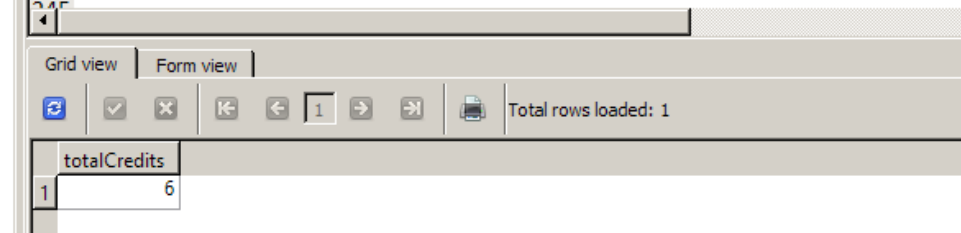
	nimi	nimi:1
1	Ohjelmoinnin perusteet	Maija Meikäläinen
2	Tietorakenteet ja algoritmit	Matti Mallikas

4. Nyt mennään vähän monimutkaisempiin kyselyihin, tehdään kysely joka laskee tietyn opiskelijan jonka opiskelijaNro attribuutti Suoritukset taulussa on 12345:

```
SELECT SUM(Kurssi.opintopiste) AS totalCredits
FROM Kurssi
JOIN Suoritukset ON Kurssi.kurssikoodi = Suoritukset.kurssikoodi
WHERE Suoritukset.opiskelijanumero = 12345;
```

Kysely palauttaa taulun muotoa:

```
241 SELECT SUM(Kurssi.opintopiste) AS totalCredits
242 FROM Kurssi
243 JOIN Suoritukset ON Kurssi.kurssikoodi = Suoritukset.kurssikoodi
244 WHERE Suoritukset.opiskelijanumero = 12345;
```



totalCredits
6

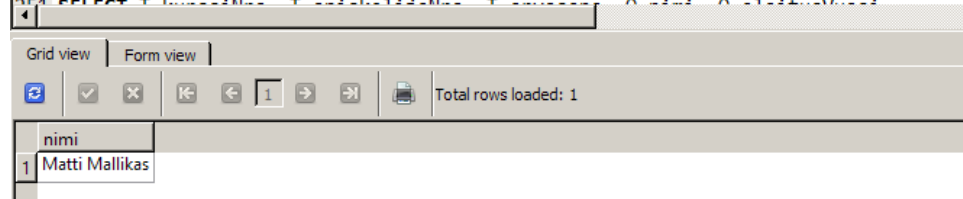
(kysely palauttaa 6 sillä suoritukset taulu sisältää yhden kurssin opintopisteet (6) tälle opiskelijalle).

5. Kysely joka palauttaa ne opiskelijoiden nimet, jotka ovat ilmoittautuneet kurssin 'ABC123' tenttiin:

```
SELECT Opiskelija.nimi
FROM Opiskelija
JOIN TenttiIlmo ON Opiskelija.opiskelijanumero = TenttiIlmo.opiskelijaNro
WHERE TenttiIlmo.kurssikoodi = 'ABC123';
```

Kysely palauttaa taulun:

```
246 SELECT Opiskelija.nimi
247 FROM Opiskelija
248 JOIN TenttiIlmo ON Opiskelija.opiskelijanumero = TenttiIlmo.opiskelijaNro
249 WHERE TenttiIlmo.kurssikoodi = 'ABC123';
250
251 SELECT T.kurssiNro, T.opiskelijaNro, T.arvosana, O.nimi, O.aloitusVuosi
```



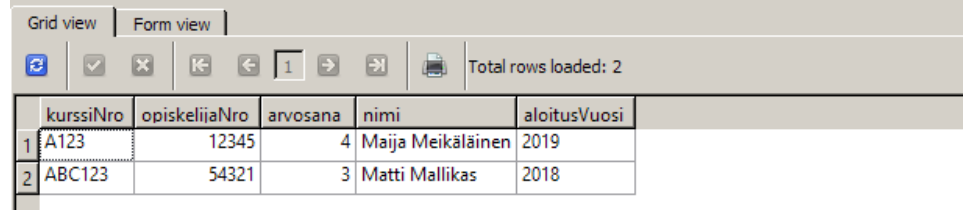
	nimi
1	Matti Mallikas

6. Kysely joka listaa kaikki tenttiläsnäöt niiden vastaavalla kurssikoodilla, opiskelijanumerolla ja arvosanalla, sekä opiskelijoiden nimi ja aloitusvuosi, jotka osallisuivat tenttiin, ja joka hakee myös ne tenttiläsnäöt, joissa on hyväksytty arvosana eli arvosana on 1-5 välillä.:

```
SELECT T.kurssiNro, T.opiskelijaNro, T.arvosana, O.nimi, O.aloitusVuosi
FROM TentiLasnaolot T, Opiskelija O
WHERE T.opiskelijaNro = O.opiskelijanumero AND T.arvosana >= 1
ORDER BY T.kurssiNro;
```

Kysely palauttaa taulun:

```
251 SELECT T.kurssiNro, T.opiskelijaNro, T.arvosana, O.nimi, O.aloitusVuosi
252 FROM TentiLasnaolot T, Opiskelija O
253 WHERE T.opiskelijaNro = O.opiskelijanumero AND T.arvosana >= 1
254 ORDER BY T.kurssiNro;
```



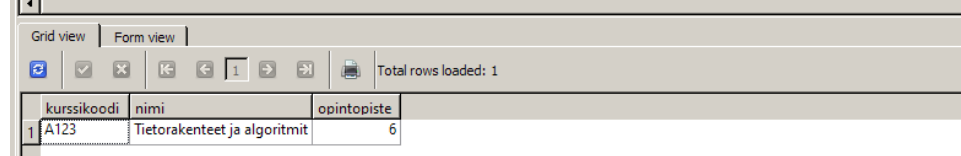
	kurssiNro	opiskelijaNro	arvosana	nimi	aloitusVuosi
1	A123	12345	4	Majja Meikalainen	2019
2	ABC123	54321	3	Matti Mallikas	2018

7. Kysely joka palauttaa tietyssä rakennuksessa olevat kaikki kurssit tässä tapauksessa rakennus joka on nimeltä 'Väre':

```
SELECT Kurssi.kurssikoodi, Kurssi.nimi, Kurssi.opintopiste
FROM Kurssi, Luento, Sali
WHERE Luento.luentokoodi = Kurssi.kurssikoodi AND Sali.saliNro = Luento.paikka AND
Sali.saliRakennus = 'Väre';
```

Kysely palauttaa taulun:

```
256 SELECT Kurssi.kurssikoodi, Kurssi.nimi, Kurssi.opintopiste
257 FROM Kurssi, Luento, Sali
258 WHERE Luento.luentokoodi = Kurssi.kurssikoodi AND Sali.saliNro = Luento.paikka AND
259 Sali.saliRakennus = 'Väre';
```



	kurssikoodi	nimi	opintopiste
1	A123	Tietorakenteet ja algoritmit	6

(joka täsmää syötetyn datan mukaan).

8. Nyt tehdään kysely joka palauttaa tentit jotka järjestetään salissa U2, ja palutetaan näiden tenttien kurssien kurssikoodi, kurssinimi, tentin päivämäärä, saliNro ja missä rakennuksessa sali sijaitsee:

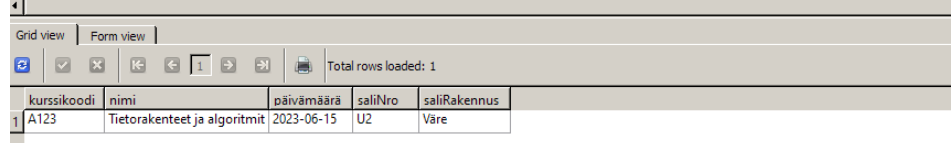
```
SELECT Kurssi.kurssikoodi, Kurssi.nimi, Tentti.päivämäärä, Sali.saliNro, Sali.saliRakennus
```

```
FROM Kurssi, Tentti, Sali
```

```
WHERE Tentti.tenttikoodi = Kurssi.kurssikoodi AND Sali.saliNro = Tentti.paikka AND
Sali.saliNro = 'U2';
```

Kysely palauttaa taulun:

```
261 SELECT Kurssi.kurssikoodi, Kurssi.nimi, Tentti.päivämäärä, Sali.saliNro, Sali.saliRakennus
262 FROM Kurssi, Tentti, Sali
263 WHERE Tentti.tenttikoodi = Kurssi.kurssikoodi AND Sali.saliNro = Tentti.paikka AND Sali.saliNro = 'U2';
```



	kurssikoodi	nimi	päivämäärä	saliNro	saliRakennus
1	A123	Tietorakenteet ja algoritmit	2023-06-15	U2	Väre

9. Yksinkertainen kysely joka hakee tietyn kurssin kaikki harjoitusryhmät:

```
SELECT *
```

```
FROM Harjoitusryhmä
```

```
WHERE harjoituskoodi = 'A123';
```

Kysely palauttaa taulun:

```

265 SELECT *
266 FROM Harjoitusryhmä
267 WHERE harjoituskoodi = 'A123';
268

```

	harjoituskc	opiskelijaN	päivämääri	paikka	paikkamää	ryhmäNro
1	A123	54321	20.05.2023	U2	30	H02

10. Nyt tehdään kysely josta aiemmin puhuin haetaan ne työntekijöiden nimet joiden tehtävänimike on *professori*:

SELECT nimi

FROM Työntekijät

WHERE tehtävänimike = 'Professori';

Kysely palauttaa meille taulun:

```

269 SELECT nimi
270 FROM Työntekijät
271 WHERE tehtävänimike = 'Professori';
272

```

	nimi
1	Matti Meikäläinen

11. Kysely joka hakee niiden opiskelijoiden opiskelijanumeron, kurssin kurssikoodin ja harjoituspisteet, HarjoitusPisteet taulusta. Jossa harjoituspisteet ovat yli tai yhtäsuuri kuin 20:

SELECT opiskelijaNro, kurssikoodi, pisteet

FROM HarjoitusPisteet

WHERE pisteet >= 20;

Kysely palauttaa taulun:



```

273 SELECT opiskelijaNro, kurssikoodi, pisteet
274 FROM HarjoitusPisteet
275 WHERE pisteet >= 20;
276

```

Grid view Form view			
1   Total rows loaded: 1			
	opiskelijaNro	kurssikoodi	pisteet
1	12345	ABC123	20

12. Kysely joka hakee kurssikoodin, kurssin nimen ja opintopistemäärän, niille kursseille joilla on tenttimahdollisuus päivälle '29.06.2023':

```

SELECT kurssikoodi, nimi, opintopiste
FROM Kurssi
WHERE tenttiMahdollisuus LIKE '%29.06.2023%';

```

Kysely palauttaa taulun:

```
277 SELECT kurssikoodi, nimi, opintopiste
278 FROM Kurssi
279 WHERE tenttiMahdollisuus LIKE '%29.06.2023%' ;
280
```

Grid viewForm view

☒

1

Total rows loaded: 1

	kurssikoodi	nimi	opintopiste
1	ABC123	Ohjelmoinnin perusteet	5

13. Kysely joka hakee kaikki niiden työntekijöiden nimet joilla on jonkinlainen varaus:

```

SELECT nimi FROM Työntekijät
JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID
JOIN Varaukset ON TyöntekijäVaraus.varausID = Varaukset.varaustunnus;

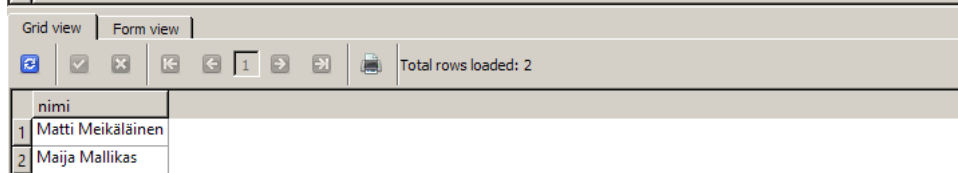
```

Kysely palauttaa:

```

281 SELECT nimi FROM Työntekijät
282 JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID
283 JOIN Varaukset ON TyöntekijäVaraus.varausID = Varaukset.varaustunnus;
284

```



14. Kysely joka hakee kaikki niiden kurssien nimet, joilla on tentti päivämäärinä '19.07.2023' ja '05.07.2023':

```

SELECT nimi, kurssikoodi
FROM Kurssi
WHERE EXISTS (
SELECT *
FROM Harjoitusryhmä
WHERE Harjoitusryhmä.harjoituskoodi = Kurssi.kurssikoodi
AND Kurssi.tenttiMahdollisuus LIKE '%05.07.2023%' AND Kurssi.tenttiMahdollisuus LIKE
'%19.07.2023%'
);


```

Kysely palauttaa taulun:

```

285 SELECT nimi, kurssikoodi
286 FROM Kurssi
287 WHERE EXISTS (
288     SELECT *
289     FROM Harjoitusryhmä
290     WHERE Harjoitusryhmä.harjoituskoodi = Kurssi.kurssikoodi
291     AND Kurssi.tenttiMahdollisuus LIKE '%05.07.2023%' AND Kurssi.tenttiMahdollisuus LIKE '%19.07.2023%'
292 );
293

```



15. Tehdään kysely joka hakee ne työntekijöiden nimet joilla on varauksia:

```

SELECT Työntekijät.nimi
FROM Työntekijät
JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID;

```

Kysely palauttaa taulun:

```

294 SELECT Työntekijät.nimi
295 FROM Työntekijät
296 JOIN TyöntekijäVaraus ON Työntekijät.tunnus = TyöntekijäVaraus.työntekijäID;
297

```

Grid view   Form view	
Total rows loaded: 2	
	nimi
1	Matti Meikäläinen
2	Maija Mallikas

16. Lopuksi tehdään kysely joka hakee työntekijöiden nimen ja tehtävänimikkeet, niille työntekijöille joiden varausID on 1 tai 2. Käytetään tässä UNIONia jotta voidaan yhdisää kaksi joukkoa:

```
SELECT nimi, tehtävänimike
```

```
FROM Työntekijät
```

```
WHERE tunnus IN (
```

```
SELECT työntekijäID
```

```
FROM TyöntekijäVaraus
```

```
WHERE varausID = 1
```

```
)
```

```
UNION
```

```
SELECT nimi, tehtävänimike
```

```
FROM Työntekijät
```

```
WHERE tunnus IN (
```

```
SELECT työntekijäID
```

```
FROM TyöntekijäVaraus
```

```
WHERE varausID = 2
```

```
);
```

Kysely palauttaa taulun:

```

298 SELECT nimi, tehtävänimike
299 FROM Työntekijät
300 WHERE tunnus IN (
301     SELECT työntekijäID
302     FROM TyöntekijäVaraus
303     WHERE varausID = 1
304 )
305 UNION
306 SELECT nimi, tehtävänimike
307 FROM Työntekijät
308 WHERE tunnus IN (
309     SELECT työntekijäID
310     FROM TyöntekijäVaraus
311     WHERE varausID = 2
312 );

```

Grid view		Form view	
		1	
		Total rows	
	nimi	tehtävänimike	
1	Maija Mallikas	Assistentti	
2	Matti Meikäläinen	Professori	

Lisätään vielä yksi kysely joka päivittää tietokannassa olevaa dataa. Tarkemmin päivitetään opiskelijan jonka opiskelijanumero on 54321, nimeä ja tutkinto-ohjelmaa:

UPDATE Opiskelija

SET nimi = 'Matti Meikäläinen', tutkintoOhjelma = 'Tietojenkäsittelytiede'

WHERE opiskelijanumero = 54321;

## Ongelmatilanteet

Jos tietokantaan lisätään dataa joka on ennen lisätty tieteenkin tulee virheitä. Työntekijän nimi ei voi olla tyhjä kun tälle lisätään dataa tietokantaan.

Yksi ongelma mikä löytyi loppuvaiheessa on se, että kun luodaan varauksia me emme tarkasta löytyykö salia ollenkaan mistään rakennuksesta. Olisi ollut järkevämpää lisätä tälle attribuutti.

Toinen ongelma löytyy varaustyypeistä. Kun luomme harjoitusryhmiä, luentoja ja tenttejä, me emme missään vaiheessa tarkasta onko näille jo tehty varausta/salivarausta. Mutta tämä ei riko tietokantaa onneksi. Tarkoituksella näin sillä suunnittelimme ensimmäisessä osassa tällaisen rakenteen.