Tietokannat-projekti

Tietokanta Yliopistolle

Basant Khattab

basant.khattab@aalto.fi

Dorida Mulliqi

dorida.mulliqi@aalto.fi

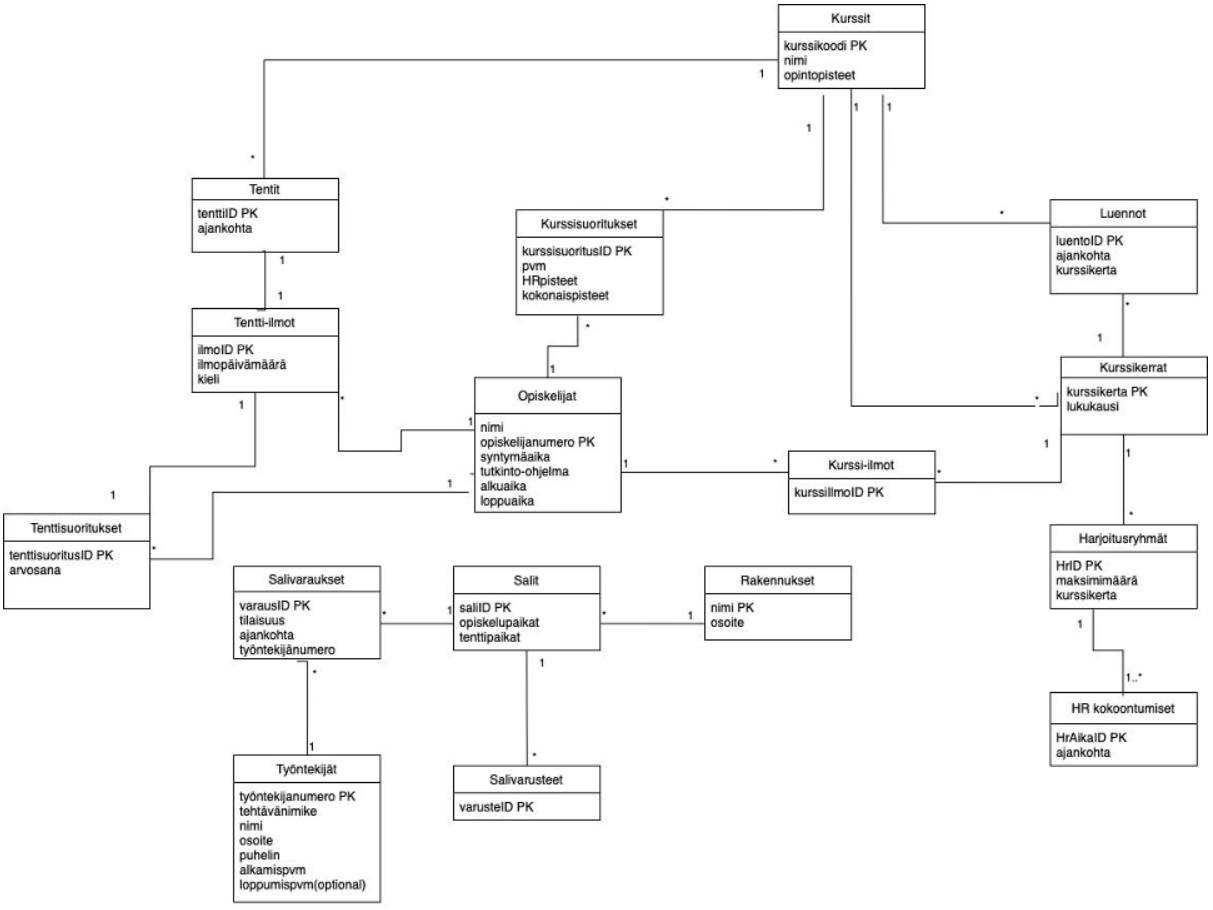
Ensimmäinen osa kokonaisuudessaan:

Tietokannat -projekti osa 1

Basant Khattab
basant.khattab@aalto.fi

Dorida Mulliqi
dorida.mulliqi@aalto.fi

UML-kaavio:



Rakenteen periaate:

Kurssit kirjataan järjestelmään kurssikoodeillaan. Monella kurssilla voi olla sama nimi, mutta jokaisella kurssilla on oma yksilökohtainen koodinsa. Samaa kurssia voidaan kuitenkin pitää eri ajankohtina. Tämä otetaan huomioon sillä, että kurssikerroille tehdään oma taulunsa. Jokaisella kurssikerralla on oma tunnuksensa, ja kurssikertaan liittyy aina se, mikä kurssi on kyseessä ja milloin. Tässä on otettu huomioon vain lukukaudet. Eli vain syksy- ja kevätmahdollisuus eikä periodeja erikseen, sillä kurssi saattaa kestää enemmän kuin yhden periodin.

Harjoitusryhmällä on jokaisella oma koodinsa. Ryhmissä on maksimimäärä paikkoja, joten, kun opiskelija ilmoittautuu kurssiin, niin kurssi-ilmon attribuutilla hrID tarkistetaan, onko harjoitusryhmäpaikkoja jäljellä. Jos ei, kurssiin ei voi ilmoittautua. Harjoitusryhmä liittyy aina tiettyyn kurssikertaan.

Harjoitusryhmät voivat kokoontua monta kertaa eri aikoina, minkä vuoksi ne tarvitsevat oman taulunsa, joka liitetään harjoitusryhmään HrID avainattribuutin avulla.

Luennoissa pidetään kirjaa siitä, milloin (minä päivämääränä ja monelta) luento on, ja mihin kurssikertaan se liittyy. Kurssikerrasta tiedetään, mistä kurssista on kyse, joten kurssikoodin kirjaaminen olisi turhaa tiedon toistamista, joka todennäköisemmin johtaa virhetilanteisiin.

Tentit liittyvät aina tiettyyn kurssiin. Tenttejä voi yhdellä kurssilla olla useampi (kurssi- ja rästitentti). Tehtävänannon mukaan tentit liittyvät itse kurssiin eivätkä kurssikertaan. Ilmoittautumisille tehdään oma taulu, sillä niillä kuvataan ilmoittautumistapahtumaa eikä itse tenttiä tapahtumana. Tämä kirjaaminen on helpompaa, sillä ilmoittautumisia voi olla monta samaan tenttiin. Lisäksi kun halutaan esimerkiksi varata sali tenttiä varten, meillä on taulu, joka kuvastaa tenttiä tapahtumana.

Kursseihin tehdään ilmoittautuminen harjoitusryhmien kautta. Järjestelmästä voidaan tarkistaa, onko paikkoja vapaana, koska kurssi-ilmoittautumisessa viitataan harjoitusryhmään ja opiskelijaan, joka tällöin saa tai jää ilman paikkaa.

Rakennuksilla on kaikilla oma nimi, jota ei ole toisella rakennuksella. Nimi-avainattribuutilla voidaan siis salit-relaatiossa liittää suoraan tiettyyn saliin, missä rakennuksessa se sijaitsee salin muun informaation lisäksi.

Salivarausta luodessa, kirjataan tilaisuus, jota varten varaus tehdään. Tämä sen takia, että tenttiin tarvitaan joka toinen paikka salista ja harjoitusryhmään voidaan istua vierekkäin. Näin voidaan laskutoimitusten jälkeen saada esimerkiksi kaksi eri tenttiä mahtumaan samaan saliin, niin etteivät saman tentin tekijät istu vierekkäin.

Varauksen tilaisuus voi olla tentti, harjoitusryhmä, luento tai muu. (Emme olleet varmoja kuuluisiko tästä tehdä UML:ään oma taulunsa.)

Salivarusteilla on oma taulunsa, johtuen siitä, että varusteita on paljon erilaisia, ja niitä voi olla useampi yhtä salia kohden. Näin jokaisella varusteella on oma tunnuksensa ja se kuuluu aina johonkin saliin.

Työntekijöillä on samaan tapaan kuin opiskelijoilla omat numeronsa, joista heidät tunnistaa, sillä samannimisiä työntekijöitä voi olla useampi. Työntekijöillä ei välttämättä ole määräaikaista työsopimusta vaan esimerkiksi vakituinen, jolloin työsuhteen loppumisajankohtaa ei välttämättä ole olemassa.

Opiskelijoiden attribuuteissa loppuaika tarkoittaa valmistumisajankohtaa.

Suorituksista pidetään kirjaa tenttien kohdalla erikseen. Sillä opiskelija saattaa ilmoittautua tenttiin, muttei välttämättä ilmesty paikan päälle tekemään tenttiä. Koska arvosana kirjataan tässä ylös, saadaan samalla selville pääsikö opiskelija läpi eli onko suoritus hyväksytty (1-5) vai hylätty (0). Tätä kautta voidaan tarkistaa, onko opiskelijalla jo kaksi hyväksyttyä tenttisuoritusta, ja jos näin on, voidaan kieltää uudelleen samaan tenttiin ilmoittautuminen kuten tehtävänannossa pyydettiin.

Kurssisuorituksessa kirjataan koko kurssin pisteet opiskelijalle. Tässä lasketaan mukaan harjoitusryhmäpisteet, ja viitataan myös tenttisuoritukseen, josta saadaan opiskelijalle arvosana laskemalla pisteitä riippuen kurssin arviointityylistä.

Selostus jatkuu funktionaalisten riippuvuuksien kohdalla, hieman logiikan perustelulla.

Relaatiot:

1. Kurssit (kurssikoodi, nimi, op)
2. Harjoitusryhmät (HrID, maksimimäärä, kurssikerta)
3. HrKokoontumiset (HrAikaID, HrID, ajankohta)
4. Luennot (luentoID, kurssikoodi, ajankohta, kurssikerta)
5. Kurssikerrat (kurssikerta, kurssikoodi, lukukausi)
6. Tentit (tenttiID, kurssikoodi, ajankohta)
7. Tentti-ilmot (ilmoID, opiskelijanro, tenttiID, ilmopvm, kieli)
8. Kurssi-ilmot (KurssiIlmoID, opiskelijanro, HrID)
9. Opiskelijat (nimi, opiskelijanumero, syntymäaika, tutkinto-ohjelma, alkuaika, loppuaika)
10. Rakennukset (nimi, osoite)
11. Salit (saliID, opiskelupaikat, tenttipaikat,)
12. Salivaraukset (varausID, tilaisuus, ajankohta, työntekijänro)
13. Salivarusteet (saliID, varusteID)
14. Työntekijät (työntekijänro, tehtävänimike, nimi, osoite, puhelin, alkamispvm, loppumispvm(optional))
15. Tenttisuoritukset (tenttisuoritusID, opiskelijanro, tenttiID, arvosana)
16. Kurssisuoritukset (kurssisuoritusID, kurssikoodi, opiskelijanro, pvm, hr.pisteet, tenttisuoritusID, kokonaispisteet)

Funktionaaliset riippuvuudet:

1. Kurssit (kurssikoodi, nimi, op)

kurssikoodi → nimi op
Sulkeuma:
{kurssikoodi}+= {kurssikoodi, nimi, op}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että kurssin kurssikoodilla löytyy myös kurssin nimi ja sen opintopistemäärä.

2. Harjoitusryhmät (HrID, maksimimäärä, kurssikerta)

HrID → maksimimäärä kurssikerta
Sulkeuma:
{HrID}+= {HrID, maksimimäärä, kurssikerta}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että harjoitusryhmän yksikäsitteisellä tunnuksella löytyy sen harjoitusryhmän maksimimäärä ja kurssikerta, johon tämä harjoitusryhmä liittyy.

3. HrKokoontumiset (HrAikaID, HrID, ajankohta)

HrAikaID → HrID ajankohta
Sulkeuma:
{HrAikaID}+= {HrAikaID, HrID, ajankohta}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että harjoitusryhmäajan yksikäsitteisellä tunnuksella löydetään harjoitusryhmän kokoontumisajat ja sen ryhmän tunnus, joka kokoontuu näinä aikoina.

4. Luennot (luentoID, ajankohta, kurssikerta)

luentoID → ajankohta kurssikerta
Sulkeuma:
{luentoID}+= {luentoID, ajankohta, kurssikerta}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että luennon yksikäsitteisellä tunnuksella löydetään ajat, jolloin luento pidetään ja kurssikerta, johon tämä luento liittyy.

5. Kurssikerrat (kurssikerta, kurssikoodi, lukukausi)

kurssikerta → kurssikoodi lukukausi
Sulkeuma:
{kurssikerta}+= {kurssikerta, kurssikoodi, lukukausi}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että kurssikerran yksikäsitteisellä tunnuksella löytyy sen kurssin kurssikoodi ja kurssikerran ajankohta.

6. Tentit (tenttiID, kurssikoodi, ajankohta)

tenttiID → kurssikoodi ajankohta
Sulkeuma:

{tenttiID}+= {tenttiID, kurssikoodi, ajankohta}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että tentin yksikäsitteisellä tunnuksella löytyy myös sen kurssin kurssikoodi, jonka tentti on kyseessä.

7. Tentti-ilmot (ilmoID, opiskelijanro, tenttiID, ilmopvm, kieli)

ilmoID → opiskelijanro tenttiID ilmopvm kieli
Sulkeuma:
{ilmoID}+= {ilmoID, opiskelijanro, tenttiID, ilmopvm, kieli}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että tentti-ilmoittautumisen yksikäsitteisellä tunnuksella löytyy ilmoittautuja, pvm, kieli ja tunnus, mihin tenttiin ollaan ilmoittauduttu.

8. Kurssi-ilmot (KurssiIlmoID, opiskelijanro, HrID, kurssikerta)

KurssiIlmoID → opiskelijanro HrID kurssikerta
Sulkeuma:
{KurssiIlmoID}+= {KurssiIlmoID, opiskelijanro, HrID, kurssikerta}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että kurssi-ilmoittautumisen yksikäsitteisellä tunnuksella löytyy ilmoittautuja, sekä sen harjoitusryhmän tunnus, johon on ilmoittauduttu.
Kurssi-ilmoittautuminen tehdään ilmoittautumalla harjoitusryhmään.

9. Opiskelijat (nimi, opiskelijanumero, syntymäaika, tutkinto-ohjelma, alkuaika, loppuaika)

opiskelijanro → nimi syntymäaika tutkinto-ohjelma alkuaika loppuaika
Sulkeuma:
{opiskelijanro}+= {opiskelijanro, nimi, syntymäaika, tutkinto-ohjelma, alkuaika, loppuaika}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.

On loogista, että opiskelijanumerolla löydetään kaikki opiskelijan tiedot.

10. Rakennukset (nimi, osoite)
nimi → osoite
Sulkeuma:
{nimi}+= {nimi, osoite}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että rakennuksen yksikäsitteisellä nimellä löydetään rakennuksen osoite.

11. Salit (saliID, opiskelupaikat, tenttipaikat,)
saliID → opiskelupaikat tenttipaikat
Sulkeuma:
{saliID}+= {saliID, opiskelupaikat, tenttipaikat, }

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että salin yksikäsitteisellä tunnuksella löydetään salin tiedot mukaan lukien sen rakennuksen nimi, jossa sali sijaitsee.

12. Salivaraukset (varausID, tilaisuus, ajankohta, työntekijänro)
varausID → tilaisuus ajankohta työntekijänro
Sulkeuma:
{varausID}+= {varausID, tilaisuus, ajankohta, työntekijänro}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että salivarauksen yksikäsitteisellä tunnuksella löydetään varauksen ajankohta, syy ja varauksen tekijä.

13. Salivarusteet (saliID, varusteID)

varusteID → saliID
Sulkeuma:
{varusteID}+= {varusteID, saliID}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että varusteen yksikäsitteisellä tunnuksella löydetään sali, jossa varuste sijaitsee.

14. Työntekijät (työntekijänro, tehtävänimike, nimi, osoite, puhelin, alkamispvm, loppumispvm(optional)).

työntekijänro → tehtävänimike nimi osoite puhelin alkamispvm loppumispvm(optional)
Sulkeuma:
{työntekijänro}+= {työntekijänro, tehtävänimike, nimi, osoite, puhelin, alkamispvm, loppumispvm(optional)}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että työntekijän yksikäsitteisellä nimellä löydetään kaikki työntekjiän tiedot.

15. Tenttisuoritukset (tenttisuoritusID, opiskelijanro, tenttiID, arvosana)

tenttisuoritusID → opiskelijanro tenttiID arvosana
Sulkeuma:
{tenttisuoritusID}+= {tenttisuoritusID, opiskelijanro, tenttiID, arvosana}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.
On loogista, että tenttisuorituksen yksikäsitteisellä tunnuksella löydetään sen suorittaja, tentti ja saatu arvosana.

16. Kurssisuoritukset (kurssisuoritusID, kurssikoodi, opiskelijanro, pvm, hr.pisteet, tenttisuoritusID, kokonaispisteet)

kurssisuoritusID → kurssikoodi opiskelijanro pvm hyväksytyt kokonaispisteet
Sulkeuma:
{kurssisuoritusID}+= {kurssisuoritusID, kurssikoodi, opiskelijanro, pvm, hr.pisteet, tenttisuoritusID, kokonaispisteet}

On BCNF-muodossa, koska kurssikoodi on avainattribuutti, ja kaikki sen avaimen relaation muut attribuutit ovat oikealla.

On loogista, että kurssisuorituksen yksikäsitteisellä tunnuksella löydetään kaikki sen tiedot.

Jos jokaisella taululla on tällä tavalla oma, yksikäsitteinen tunnuksensa, jolla tietoja voidaan löytää, vältytään uudelleen osittamiselta ja saadaan kaikki automaattisesti BCNF-muotoon.

Kun kaikella on oma tunnuksensa, voidaan helposti erottaa kaksi muuten täysin samanlaista monikkoa, kuten esimerkiksi ilmoittautumisessa. Jos sama opiskelija ilmoittautuu useampaan tenttiin, on helpompi käsitellä niitä, kun niillä on omat tunnuskoodinsa.

Tähän tekniikkaan liittyy kuitenkin ongelmia. Mm. seuraavanlaisia anomalioita:

Anomaliat:

Tietoa toistuu, Redundanssi. Esimerkiksi opiskelijanumeroa käytetään useaan kertaan tietokannassa, sama tieto toistuu uudelleen ja uudelleen monessa kohdassa.

Poistoanomalia. Tiettyjen monikoiden poistaminen johtaa muidenkin tietojen poistumiseen. Esimerkiksi, jos jonkun opiskelijan poistaa järjestelmästä, poistuu myös sen opiskelijan ilmoittautuminen, sillä nyt opiskelijanumeroa ei enää voida yhdistää tiettyyn opiskelijaan, se on vain irtonainen, turha tieto.

Päivitysanomalia. Koska sama tieto on esitetty useaan kertaan, esimerkiksi edellä mainittu opiskelijanumero, tiedon muuttuessa, täytyy muutos tehdä erikseen kaikkiin niihin kohtiin, joissa se esiintyy.

Virheellinen tieto. Mikään ei estä virheellisen tiedon syöttämistä tietokantaan. Jolloin esimerkiksi hr-pisteet ja kokonaispisteet saattavat olla virheellisiä ja epärealistisia.

Tietokannat-projekti osa 2

Basant Khattab

basant.khattab@aalto.fi

Dorida Mulliqi

dorida.mulliqi@aalto.fi

Ensimmäisen osan palautuksen jälkeen tehdyt muutokset

1. Muutettiin Salivaraukset-relaatiota. Eli nyt on kaksi attribuuttia ajalle. Varauksen teko ajankohtaa kuvaa attribuutti “varauspvm” ja “ajankohta”-attribuutti kuvaa itse varauksen ja tilaisuuden ajankohtaa.
2. Poistettiin attribuutteja, joiden tiedot on saatavilla assosiaatioilla.
3. Poistettiin KurssiIlmot-luokka, sillä tiedot saadaan Opiskelijat ja Kurssikerrat luokkien välisellä assosiaatiolla.
4. Poistettiin Kurssisuoritukset-luokka, sillä tiedot saadaan järkevämmin Opiskelijat ja Kurssikerrat välisellä assosiaatiolla. Attribuutti harjoituspisteet tallennetaan tähän assosiaatioon.
5. Poistettiin Luennot-luokan assosiaatio Kurssit-luokkaan.

Uusi tietokanta skeema

1. Kurssit (kurssikoodi, nimi, op)
2. Harjoitusryhmät (HrID, maksimimäärä, kurssikerta)
3. HrKokoontumiset (HrAikaID, HrID, ajankohta)
4. Luennot (luentoID, ajankohta, kurssikerta)
5. Kurssikerrat (kurssikerta, kurssikoodi, lukukausi)
6. Tentit (tenttiID, ajankohta)
7. Tentti-ilmot (ilmoID, opiskelijanro, tenttiID, ilmopvm, kieli)
8. Opiskelijat (nimi, opiskelijanumero, syntymäaika, tutkinto-ohjelma, alkuaika, loppuaika)
9. Rakennukset (nimi, osoite)
10. Salit (saliID, opiskelupaikat, tenttipaikat, nimi)
11. Salivaraukset (varausID, saliID, tilaisuus, ajankohta, varauspvm, työntekijänro)
12. Salivarusteet(saliID,varusteID)
13. Työntekijät (työntekijänro, tehtävänimike, nimi, osoite, puhelin, alkamispvm, loppumispvm(optional))
14. Tenttisuoritukset (opiskelijanro, tenttiID, arvosana)
15. Harjoitusryhmä-ilmoittautuminen (opiskelijanro, HrID)
16. LuentoKurssikerta (luentoID, kurssikerta)

Muuntaminen SQL-muotoon

Luotiin SQLiteStudioon tietokanta nimellä Tietokannatprojekti. Käskyt sijoitimme yksi kerrallaan seuraavassa muodossa:

```
CREATE TABLE Kurssit (
  kurssikoodi CHAR(6) NOT NULL PRIMARY KEY,
  nimi VARCHAR(50),
  op INT
);

CREATE TABLE Kurssit (
  kurssikoodi varchar(10) PRIMARY KEY,
  nimi varchar(50),
  op int
);

CREATE TABLE Harjoitusryhmät (
  HrID int PRIMARY KEY,
  maksimimäärä int,
  kurssikerta varchar(10),
  FOREIGN KEY (kurssikerta) REFERENCES Kurssikerrat(kurssikerta)
);

CREATE TABLE HrKokoontumiset (
```

```
  HrAikaID int PRIMARY KEY,
  HrID int,
  ajankohta date,
  FOREIGN KEY (HrID) REFERENCES Harjoitusryhmät(HrID)
);

CREATE TABLE Luennot (
  luentoID int PRIMARY KEY,
  ajankohta datetime,
  kurssikerta varchar(10),
  FOREIGN KEY (kurssikerta) REFERENCES Kurssikerrat(kurssikerta)
);

CREATE TABLE Kurssikerrat (
  kurssikerta varchar(10) PRIMARY KEY,
  kurssikoodi varchar(10),
  lukukausi varchar(10),
  FOREIGN KEY (kurssikoodi) REFERENCES Kurssit(kurssikoodi)
);

CREATE TABLE Tentit (
  tenttiID int PRIMARY KEY,
  ajankohta datetime
```

```
);

CREATE TABLE Tenttiilmot (
  ilmoID int PRIMARY KEY,
  opiskelijanro int,
  tenttiID int,
  ilmopvm date,
  kieli varchar(10),
  FOREIGN KEY (opiskelijanro) REFERENCES Opiskelijat(opiskelijanumero),
  FOREIGN KEY (tenttiID) REFERENCES Tentit(tenttiID)
);

CREATE TABLE Opiskelijat (
  nimi varchar(50),
  opiskelijanumero int PRIMARY KEY,
  syntymäaika date,
  tutkinto-ohjelma varchar(50),
  alkuaika date,
  loppuaika date
);

CREATE TABLE Rakennukset (
```

```
  nimi varchar(50),
  osoite varchar(50),
  PRIMARY KEY (nimi, osoite)
);


CREATE TABLE Salit (
  saliID int PRIMARY KEY,
  opiskelupaikat int,
  tenttipaikat int,
  nimi varchar(50)
);


CREATE TABLE Salivaraukset (
  varausID int PRIMARY KEY,
  saliID int,
  tilaisuus varchar(50),
  ajankohta datetime,
  varauspvm date,
  työntekijänro int,
  FOREIGN KEY (saliID) REFERENCES Salit(saliID),
  FOREIGN KEY (työntekijänro) REFERENCES Työntekijät(työntekijänro)
```

```
);

CREATE TABLE Salivarusteet (
  saliID int,
  varusteID int,
  PRIMARY KEY (saliID, varusteID)
);

CREATE TABLE Työntekijät (
  työntekijänro int PRIMARY KEY,
  tehtävänimike varchar(50),
  nimi varchar(50),
  osoite varchar(50),
  puhelin varchar(20),
  alkamispvm date,
  loppumispvm date
);

CREATE TABLE Tenttisuoritukset (
  opiskelijanro int,
  tenttiID int,
```

```
  arvosana int,
  PRIMARY KEY (opiskelijanro, tenttiID),
  FOREIGN KEY (opiskelijanro) REFERENCES Opiskelijat(opiskelijanumero),
  FOREIGN KEY (tenttiID) REFERENCES Tentit(tenttiID)
);

CREATE TABLE Harjoitusryhmäilmoittautuminen (
  opiskelijanro int,
  HrID int,
  PRIMARY KEY (opiskelijanro, HrID),
  FOREIGN KEY (opiskelijanro) REFERENCES Opiskelijat(opiskelijanumero),
  FOREIGN KEY (HrID) REFERENCES Harjoitusryhmät(HrID)
);

CREATE TABLE LuentoKurssikerta (
  luentoID int,
  kurssikerta varchar(10),
  PRIMARY KEY (luentoID, kurssikerta),
  FOREIGN KEY (luentoID) REFERENCES Luennot(luentoID),
  FOREIGN KEY (kurssikerta) REFERENCES Kurssikerrat(kurssikerta)
);
```

Huom. Tietorakenteisiin tuli lopuksi muutoksia. ID:t ovat varchar-muodossa.

Avaimet

Eheysrajoitteet ja viite-eheys tarkistettiin SQLLiteStudiossa. Tarkistimme, että pääavaimet ja viiteavaimet on tietokannassa määritelty mahdollisimman hyvin.

Seuraavaksi mietimme tyypillisiä käyttötarkoituksia ja hakuja, joita tietokantaan tehdään. Ja sitä kautta päätimme luoda hakemistot. Hakemistot nopeuttavat tiedon saantia, joten yleisimpien järkevimpien hakujen mukaan seuraavat hakemistot on hyödyllistä luoda.

Hakemistot

Jotta saadaan kurssit niiden kurssikoodien mukaan, on hyödyllistä seuraava hakemisto:

1. CREATE INDEX idx_kurssit_kurssikoodi ON Kurssit (kurssikoodi);

Tietyn kurssikerran mukaan harjoitusryhmät

2. CREATE INDEX idx_hr_kurssikerta ON Harjoitusryhmät (kurssikerta);

Tietyn harjoitusryhmän kokoontuminen tiettyyn ajankohtaan

3. CREATE INDEX idx_hr_kokoontumiset ON HrKokoontumiset (HrID, ajankohta);

Luennot kurssikoodin ja ajankohdan mukaan

4. CREATE INDEX idx_luennot_kurssikoodi_ajankohta ON Luennot (kurssikoodi, ajankohta);

Kurssikerrat kurssikoodin ja lukukauden mukaan

5. CREATE INDEX idx_kurssikerrat_kurssikoodi_lukukausi ON Kurssikerrat (kurssikoodi, lukukausi);

Tentit kurssikoodin ja ajankohdan mukaan

6. CREATE INDEX idx_tentit_kurssikoodi_ajankohta ON Tentit (kurssikoodi, ajankohta);

Tentti-ilmoittautumiset opiskelijan ja tentin mukaan

7. CREATE INDEX idx_tentti_ilmot_opiskelijanro_tenttiID ON Tentti-ilmot (opiskelijanro, tenttiID);

Opiskelijat opiskelijanumeron mukaan

8. CREATE INDEX idx_opiskelijat_opiskelijanumero ON Opiskelijat (opiskelijanumero);

Rakennukset nimensä mukaan

9. CREATE INDEX idx_rakennukset_nimi ON Rakennukset (nimi);

Salit nimensä mukaan

10. CREATE INDEX idx_salit_nimi ON Salit (nimi);

Salivaraukset salin ja varauksen tilaisuuden ajankohdan mukaan

11. CREATE INDEX idx_salivaraukset_saliID_ajankohta ON Salivaraukset (saliID, ajankohta);

Salivarusteet salin ja varusteen mukaan

12. CREATE INDEX idx_salivarusteet_saliID_varusteID ON Salivarusteet (saliID, varusteID);

Työntekijät numeronsa mukaan

13. CREATE INDEX idx_työntekijät_työntekijänro ON Työntekijät (työntekijänro);

Tenttisuoritukset opiskelijan ja tentin mukaan

14. CREATE INDEX idx_tenttisuoritukset_opiskelijanro_tenttiID ON Tenttisuoritukset (opiskelijanro, tenttiID);

LuentoKurssikerrat luennon mukaan

15. CREATE INDEX idx_luentokurssikerta_luentoID ON LuentoKurssikerta (luentoID);

Harjoitusryhmäilmoittautumiset opiskelijan mukaan

16. CREATE INDEX idx_harjoitusryhmäilmoittautuminen_opiskelijanro ON HarjoitusryhmäIlmoittautuminen (opiskelijanro);

Harjoitusryhmät tunnuksensa mukaan

17. CREATE INDEX idx_harjoitusryhmät_hrID ON Harjoitusryhmät (HrID);

Näkymä

KurssiIlmoittautumiset

SELECT Kurssikerrat.kurssikoodi, Kurssikerrat.lukukausi,
COUNT(DISTINCT Opiskelijat.opiskelijanumero) AS Ilmoittautuminen

```
FROM Kurssikerrat
JOIN HrKokoontumiset ON Kurssikerrat.kurssikerta = HrKokoontumiset.kurssikerta
JOIN HarjoitusryhmäIlmoittautuminen ON HrKokoontumiset.HrID =
HarjoitusryhmäIlmoittautuminen.HrID
JOIN Opiskelijat ON HarjoitusryhmäIlmoittautuminen.opiskelijanro =
Opiskelijat.opiskelijanumero
GROUP BY Kurssikerrat.kurssikoodi, Kurssikerrat.lukukausi
```

Tässä KurssiIlmoittautumiset-näkymä. Se ottaa kokonaisuudessaan huomioon, että opiskelija ilmoittautuu kurssille liittymällä harjoitusryhmään. Lasketaan yhteen kurssille ilmoittautuneet opiskelijat. Lopuksi tulos annetaan kurssin kurssikoodin ja lukukauden mukaan. Esimerkiksi ilmoittautujamäärien muutoksia on helppo seurata tämän näkymän avulla.

Käyttötapaukset

Tässä mahdolliset ja todennäköisimmät käyttötapaukset tietokannalle. Sekä SQL-muodossa, että selitettynä.

1. **Uuden kurssin lisääminen tietokantaan.**

Kun lisätään uusi kurssi tietokantaan, syötetään Kurssit-luokan attribuuttien mukaiset tiedot eli: kurssikoodi, nimi ja op.

SQL-käsky:

```
INSERT INTO Kurssit (kurssikoodi, nimi, op)
VALUES ('CS105','studio',5);
```

2. **Pyydetään kaikki rakennukset ja niiden osoitteet.**

Nähdään, missä mikäkin rakennus sijaitsee.

SQL-käsky:

```
SELECT nimi, osoite
FROM Rakennukset;
```

3. **Päivitetään tieto siitä, kuinka paljon opiskelijoita on mahdollista osallistua harjoitusryhmään. (Muutetaan harjoitusryhmän maksimiosallistujamäärä)**

Jos saadaan käyttöön esimerkiksi isompi tila tai enemmän resursseja, voidaan muuttaa harjoitusryhmän kokoa. Tai tietysti myös toiseen suuntaan.

SQL-käsky:

```
UPDATE Harjoitusryhmät
SET maksimimäärä = 25
WHERE HrID = 1;
```

4. **Tietyn harjoitusryhmän kaikki opiskelijat.**

Tätä varten tarvitaan liitos Opiskelijat ja Harjoitusryhmä-ilmoittautuminen -luokkien välille.

SQL-käsky:

```
SSELECT Opiskelijat.nimi, Opiskelijat.opiskelijanumero, Opiskelijat.tutkintoohjelma
FROM Opiskelijat
JOIN Harjoitusryhmäilmoittautuminen ON Opiskelijat.opiskelijanumero =
Harjoitusryhmäilmoittautuminen.opiskelijanro
WHERE Harjoitusryhmäilmoittautuminen.HrID = 1;
```

5. **Lista kursseista ja niiden luentojen ajankohdista.**

Tätä varten tarvitaan liitos Kurssikerrat, Opiskelijat ja Luennot -taulujen välille. Käytetään opiskelijanumeroa hyväksi.

SQL-käsky:

```
SELECT Kurssit.nimi AS Kurssi, Luennot.ajankohta AS Luentoaika
FROM Kurssit
JOIN Kurssikerrat ON Kurssit.kurssikoodi = Kurssikerrat.kurssikoodi
JOIN Luennot ON Kurssikerrat.kurssikerta = Luennot.kurssikerta
JOIN Opiskelijat ON Opiskelijat.opiskelijanumero = '123456'
WHERE Kurssikerrat.lukukausi = 'Autumn 2023';
```

6. **Päivitetään opiskelijan tietoja.**

Esimerkiksi opiskelijan nimi tai koulutusohjelma voi muuttua.

SQL-käsky:
(*=wildcard)

```
SELECT *
FROM Opiskelijat
WHERE opiskelijanumero = '123456';

UPDATE Opiskelijat
SET nimi = 'New Name'
WHERE opiskelijanumero = '123456';

UPDATE Opiskelijat SET tutkinto-ohjelma = 'New Degree'
WHERE opiskelijanumero = '123456';


SELECT *
FROM Opiskelijat
WHERE opiskelijanumero = '123456';
```

7. **Lista tietyn opiskelijan tentti-ilmoittautumisista.**

SQL-käsky:

```
SELECT *
FROM Tentti-ilmot
```

```
WHERE opiskelijanro = '123456';
```

8. **Tentin keskiarvosana**

SQL-käsky:

```
SELECT AVG(arvosana) as keskiarvo
FROM Tenttisuoritukset
WHERE tenttiID = 'T1';
```

9. **Lista tietyn ajankohdan (päivämäärän) salivarauksista.**

SQL-käsky:

```
SELECT *
FROM Salivaraukset
WHERE ajankohta = '2023-05-15';
```

10. **Kaikki tietyssä rakennuksessa pidetyt/pidettävät tentit. (tenttitilaisuudet)**

SQL-käsky:

```
SELECT * FROM Salivaraukset
JOIN Salit ON Salivaraukset.saliID = Salit.saliID
WHERE tilaisuus = 'L2' AND Salit.nimi = 'T-talo 2';
```

11. **Opiskelijat, jotka eivät ole ilmoittautuneet tiettyyn tenttiin.**

Esimerkiksi mahdollisia muistutusviestejä varten.

SQL-käsky:

```
SELECT *
FROM Opiskelijat
WHERE opiskelijanumero
NOT IN
(SELECT opiskelijanro FROM Tenttiilmot WHERE tenttiID = 'T1');
```

12. **Ne kurssit, joilla ei ole omaa kurssikertaa vielä.**

Saadaan selville, mitkä kurssit tarvitsevat vielä paikkansa lukuvuosikalenterista.

```
SELECT *
FROM Kurssit
WHERE kurssikoodi
NOT IN
(SELECT kurssikoodi FROM Kurssikerrat);
```

13. Vapaat paikat kurssitenttiä kohden.

Saadaan selville, paljonko paikkoja on jäljellä vielä kussakin tentissä.

```
SELECT t.tenttiID, s.tenttipaikat - COUNT(DISTINCT v.varausID) AS 'seats available'
FROM Tentit t
LEFT JOIN Salivaraukset v ON v.tilaisuus = t.tenttiID
LEFT JOIN Salit s ON s.saliID = v.saliID
GROUP BY t.tenttiID;
```

14. Tietyn tentin tehneet opiskelijat ja heidän arvosanansa siitä tentistä.

Saadaan esimerkiksi opiskelijoille julkaistava pdf, josta jokainen näkee arvosanansa opiskelijanumeronsa kohdalla.

```
SELECT o.nimi, ts.arvosana
FROM Opiskelijat o, Tenttisuoritukset ts
WHERE o.opiskelijanumero = ts.opiskelijanro
AND ts.tenttiID = 'T2';
```

15. Ne opiskelijat, jotka ovat ilmoittautuneet suomenkieliseen tenttiin.

SQL-käsky:

```
SELECT Opiskelijat.nimi, Tentit.ajankohta
FROM Opiskelijat
INNER JOIN Tenttiilmot ON Opiskelijat.opiskelijanumero = Tenttiilmot.opiskelijanro
INNER JOIN Tentit ON Tenttiilmot.tenttiID = Tentit.tenttiID
WHERE Tenttiilmot.kieli = 'FI';
```

Käyttötapauksien testaamista varten loimme jokaiseen luokkaan omaa dataa suoraan SQLLiteStudioon.

Listaukset

```
INSERT INTO Kurssit (kurssikoodi, nimi, op)
VALUES ('CS105','studio',5);
SELECT nimi, osoite
FROM Rakennukset;
UPDATE Harjoitusryhmät
SET maksimimäärä = 25
WHERE HrID = 1;
SELECT Opiskelijat.nimi, Opiskelijat.opiskelijanumero, Opiskelijat.tutkintoohjelma
FROM Opiskelijat
JOIN Harjoitusryhmäilmoittautuminen ON Opiskelijat.opiskelijanumero =
Harjoitusryhmäilmoittautuminen.opiskelijanro
WHERE Harjoitusryhmäilmoittautuminen.HrID = 1;
SELECT Kurssit.nimi AS Kurssi, Luennot.ajankohta AS Luentoaika
FROM Kurssit
JOIN Kurssikerrat ON Kurssit.kurssikoodi = Kurssikerrat.kurssikoodi
JOIN Luennot ON Kurssikerrat.kurssikerta = Luennot.kurssikerta
JOIN Opiskelijat ON Opiskelijat.opiskelijanumero = '123456'
WHERE Kurssikerrat.lukukausi = 'Autumn 2023';
SELECT * FROM Opiskelijat WHERE opiskelijanumero = '123456';
UPDATE Opiskelijat SET nimi = 'New Name' WHERE opiskelijanumero = '123456';
SELECT * FROM Opiskelijat WHERE opiskelijanumero = '123456';
SELECT * FROM Tenttiilmot WHERE opiskelijanro = '123456';
SELECT AVG(arvosana) as keskiarvo
FROM Tenttisuoritukset WHERE tenttiID = 'T1';
SELECT * FROM Salivaraukset
WHERE ajankohta = '2023-05-15';
SELECT * FROM Salivaraukset
JOIN Salit ON Salivaraukset.saliID = Salit.saliID
WHERE tilaisuus = 'L2' AND Salit.nimi = 'T-talo 2';
SELECT * FROM Opiskelijat WHERE opiskelijanumero NOT IN
(SELECT opiskelijanro FROM Tenttiilmot WHERE tenttiID = 'T1');
SELECT * FROM Kurssit WHERE kurssikoodi NOT IN
(SELECT kurssikoodi FROM Kurssikerrat);
SELECT t.tenttiID, s.tenttipaikat - COUNT(DISTINCT v.varausID) AS 'seats available'
FROM Tentit t
LEFT JOIN Salivaraukset v ON v.tilaisuus = t.tenttiID
LEFT JOIN Salit s ON s.saliID = v.saliID
GROUP BY t.tenttiID;
```

```
SELECT o.nimi, ts.arvosana
FROM Opiskelijat o, Tenttisuoritukset ts
WHERE o.opiskelijanumero = ts.opiskelijanro
AND ts.tenttiID = 'T2';
SELECT Opiskelijat.nimi, Tentit.ajankohta
FROM Opiskelijat
INNER JOIN Tenttiilmot ON Opiskelijat.opiskelijanumero = Tenttiilmot.opiskelijanro
INNER JOIN Tentit ON Tenttiilmot.tenttiID = Tentit.tenttiID
WHERE Tenttiilmot.kieli = 'FI';
```