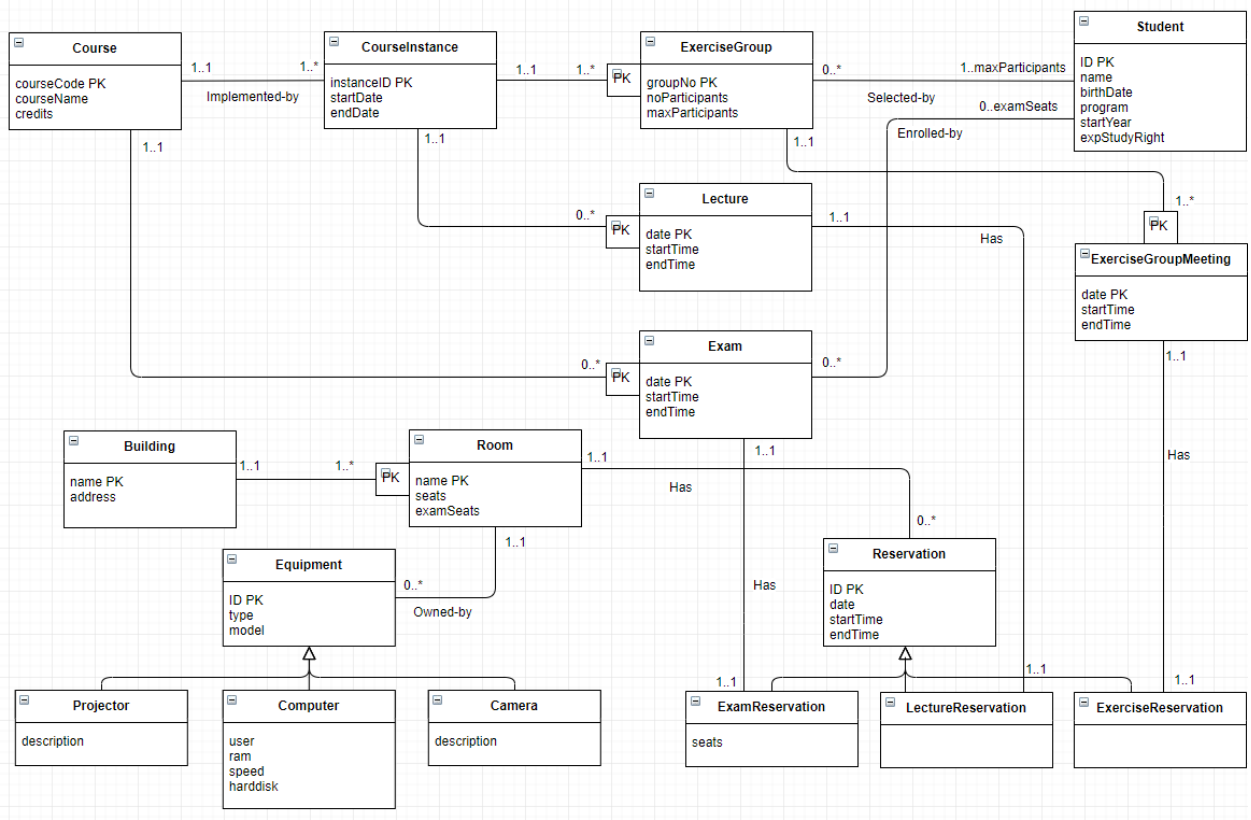
Yliopiston tietokanta

UML-kaavio, relaatiot ja toteutus SQL-kielellä

Harjoitustyö
CS-A1150 Tietokannat kesä 2020
Johanna Hakonen 48960U
Johanna.hakonen@gmail.com

UML-kaavio

Työssä suunniteltiin tietokanta, johon tallennetaan tietoja yliopiston kursseista, tiloista, tilavarauksista ja ilmoittautumisista. Suunnittelu aloitettiin UML-kaaviolla, joka on esitetty kuvassa 1.



Kuva 1. UML-kaavio.

UML-kaaviota vastaavat relaatiokaaviot

Seuraavaksi UML-kaavio muunnettiin relaatiomalliin. Relaatiokaavioista alleviivattiin avainattribuutit ja ne on esitetty alla.

Students(ID, name, birthDate, program, startYear, expStudyRight)

Courses(code, name, credits)

CourseInstance(instanceID, startDate, endDate)

ImplementedBy(instanceID, courseCode)

Lectures(instanceID, date, startTime, endTime)

ExerciseGroups(instanceID, groupNo, noParticipants, maxParticipants)

ExerciseGroupMeeting(instanceID, groupNo, date, startTime, endTime)

Exam(courseCode, date, startTime, endTime)

SelectedBy(exerciseGroupNo, instanceID, studentID)

EnrolledBy(examDate, courseCode, studentID)

Buildings(name, address)

Rooms(building, name, seats, examSeats)

HasReservation(reservationID, building, room)

ExamReservations(reservationID, date, startTime, endTime, seats)

HasExamReservation(courseCode, date, reservationID)

LectureReservations(reservationID, date, startTime, endTime)

HasLectureReservation(instanceID, date, reservationID)

ExerciseReservations(reservationID, date, startTime, endTime)

HasExerciseReservation(instanceID, groupNo, date, reservationID)

OwnedBy(equipmentID, building, room)

Equipment(ID, type, model)

Projector(ID, description)

Computer(ID, user, ram, speed, harddisk)

Camera(ID, description)

Ratkaisun kuvaus

Kursseja kuvataan UML-kaaviossa luokalla Course, jonka attribuutit ovat code, name ja credits. Avainattribuuttina toimii code. Courses-relaatiolla on vastaavat attribuutit. Sama kurssi voidaan järjestää useita kertoja joko samana lukukautena tai eri lukukausien aikana, mikä on toteutettu luokan CourseInstance avulla. Sillä on avainattribuuttina instanceID ja muina attribuutteina startDate ja endDate, kuten myös CourseInstances-relaatiolla. CourseInstance -luokka liittyy Course-luokkaan monesta yhteen -assosiaatiolla Implemented-by. Siitä tehdään ImplementedBy-relaatio, jonka avainattribuutti on monesta-puolen instanceID ja toinen attribuutti on yhteen-puolen courseCode.

Kurssilla voi olla luentoja (luokka Lecture/relaatio Lectures), harjoitusryhmiä (exerciseGroup/exerciseGroups) ja tenttejä (Exam/Exams). Luennot ja harjoitusryhmät liittyvät yhteen kurssikertaan eli luokkaan CourseInstance mutta tentit luokkaan Course. Kaikki ovat monesta yhteen -assosiaatioita ja jokaisella kurssikerralla on ainakin yksi harjoitusryhmä. Luokan Exam avainattribuutit ovat courseCode luokasta Course ja date sekä muut startTime ja endTime.

Kurssin luentoajat eivät ole välttämättä samat joka viikko, vaan ne voivat vaihdella, joten Lecture-luokan attribuutteja ovat instanceID, date, startTime ja endTime. Se saa instanceID-attribuutin CourseInstance-luokalta. Kaksi ensimmäistä riittää avainattribuuteiksi, jos oletetaan, ettei samana päivänä ole kahta luentoa.

Harjoitusryhmiä kuvataan luokalla ExerciseGroup, jonka avainattribuuteja ovat instanceID taas luokasta CourseInstance ja kurssikertakohtainen groupNo sekä tavallisina attribuutteina noParticipants ja maxParticipants. Koska sama harjoitusryhmä voi kokoontua useita kertoja eivätkä kokoontumisajat ole välttämättä samoja joka viikko, liittyy luokkaan ExerciseGroup luokka ExerciseGroupMeeting monesta yhteen -assosiaatiolla. Sen avainattribuutteja ovat instanceID luokasta CourseInstance, groupNo luokasta ExerciseGroup ja date sekä muita attribuutteja startTime ja endTime.

Opiskelijoita kuvataan luokalla Student ja relaatiolla Students, joiden atribuutteja ovat ID, name, birthDate, program, startYear ja expStudyRight. Avainattribuuttina on ID. Opiskelija ilmoittautuu kurssille ilmoittautumalla johonkin sen harjoitusryhmistä eli luokkien Student ja ExerciseGroup välillä on monesta moneen -assosiaatio Selected-by, josta tehdään relaatio SelectedBy. Avainattribuutteina ovat exerciseGroupNo, instanceID ja studentID. Opiskelijan pitää ilmoittautua erikseen sellaiseen tenttiin,

johon hän haluaa osallistua, mikä hoidetaan monesta moneen -assosiaatiolla Enrolled-by. Siitä tehdään relaatio EnrolledBy ja sen avainattribuutit ovat examDate, courseCode ja studentID.

Yliopistolla on useita rakennuksia, joita kuvataan luokalla Building ja relaatiolla Buildings. Avainattribuuttina on name ja muuna attribuuttina address. Rakennuksiin liittyy opetuskäyttöön tarkoitettuja tiloja, joita kuvataan luokalla Room ja relaatiolla Rooms. Niiden attribuutteina ovat name, seats ja examSeats, joista name on avainattribuutti. Lisäksi avainattribuuttina käytetään Building-luokan avainta name nimellä building.

Salivaraukset hoidetaan yliluokalla Reservation, jolla on aliluokat ExamReservation, LectureReservation ja ExerciseReservation. Koska kaikki monikot kuuluvat johonkin aliluokkaan eikä yliluokalla Reservations ole omia monesta moneen -assosiaatioita tai monesta yhteen -assosiaatioita, joissa yliluokka on yhdestä-puolella, on toimivin tapa se, että aliluokista tehdyillä relaatioilla on sekä omat attribuutit että kaikki yliluokan attribuutit (ID, date, startTime, endTime). Tällöin ei relaatiota Reservations tarvita lainkaan. Reservation-luokka liittyy Room-luokkaan monesta yhteen -assosiaatiolla, josta tehdään relaatio HasReservation. Sen avainattribuuttina on reservationID ja muina attribuutteina building ja room.

ExamReservation-luokka liittyy Exam-luokkaan yhdestä yhteen -assosiaatiolla. Sillä on yliluokan attribuuttien lisäksi oma varattuja paikkoja kuvaava attribuutti seats. LectureReservation-luokka liittyy Lecture-luokkaan ja ExerciseReservation-luokka ExerciseGroupMeeting-luokkaan yhdestä yhteen -assosiaatioilla. Tenttejä varten tehty salivaraus voi olla pidempi kuin itse tentin kesto, joten Exam-luokan alku- ja loppuajat ovat eri kuin varauksen vastaavat.

Salien varusteita kuvataan yliluokalla Equipment, jolla on aliluokat Projector, Computer ja Camera. Muut laitteet kuvataan yliluokalla, jolloin toimivin tapa on se, että aliluokista tehdyillä relaatioilla on vain omat attribuutit ja yliluokan avainattribuutit. Equipment-luokan avainattribuuttina on ID sekä muina type ja model. Computer-luokalla on lisäksi omia attribuutteja user, ram, speed, harddisk. Projector- ja Camera-luokilla on lisäksi omana attribuuttina description. Equipment-luokka liittyy Room-luokkaan monesta yhteen -assosiaatiolla Owned-by, josta tehdään relaatio OwnedBy. Sen avainattribuutti on monesta-puolen equipmentID ja muut attribuutit yhteen-puolen building ja room.

Funktionaaliset riippuvuudet

Students-relaation ID-attribuutti yksilöi opiskelijan eli funktionaalinen riippuvuus on

ID -> name birthDate program startYear expStudyRight

Lisäksi tutkinto-ohjelma ja aloitusvuosi määräävät käsittääkseni opinto-oikeuden päättymisen:

program startYear -> expStudyRight

Courses-relaatiolla vastaavasti: code -> name credits

CourseInstances-relaatiolla: instanceID -> startDate endDate

ImplementedBy-relaatiolla: instanceID -> courseCode

Lectures-relaatiolla: instanceID date -> startTime endTime

ExerciseGroups-relaatiolla: instanceID groupNo -> noParticipants maxParticipants

ExerciseGroupMeetings-relaatiolla: instanceID groupNo date -> startTime endTime

Exams-relaatiolla: courseCode date -> startTime endTime

Buildings-relaatiolla: name -> address

Rooms-relaatiolla: building name -> seats examSeats

LectureReservations- ja ExerciseReservations-relaatiolla: reservationID date startTime -> endTime

ExamReservations-relaatiolla: reservationID date startTime -> endTime seats

HasReservation-relaatiolla: reservationID -> building room

Equipment-relaatiolla: ID -> type model

Projectors- ja Cameras-relaatiolla: ID -> description

Computers-relaatiolla: ID-> user ram speed harddisk

OwnedBy-relaatiolla: equipmentID -> room building

HasExamReservation-, HasLectureReservation-, HasExerciseReservation-, SelectedBy- ja EnrolledBy-relaatioilla ei ole funktionaalisia riippuvuuksia.

Anomaliat

Jos tutkinto-ohjelma ja aloitusvuosi määräävät opinto-oikeuden päättymisen, niitä koskeva tieto toistuu Students-relaatiossa ja kyseessä on toistoanomalia. Tällöin on hyvä tehdä oma relaatio StudyRights, jottei aiheudu myöskään poistoanomaliaa. Lisäämällä instanceID-attribuutti CourseInstance-luokkaan saatiin toisteisuutta pois. Jos kurssikerran päivämäärät muuttuvat, niitä ei tarvitse muuttaa moneen relaatiokaavioon mutta toisaalta kurssikerta tulee nyt lisätä ja poistaa monessa relaatiossa eli aiheutui päivitysanomaliaa. Tietokoneiden, videotykkien ja kameroiden hankinnat tai poistot täytyy lisätä niiden omiin relaatioihin ja Equipments-relaatioon. Varausrelaatioita yksinkertaistamaan lisättiin varausnumero mutta tietokantaa käytettäessä yhtä varausnumeroa koskevat tiedot tulee lisätä tai poistaa kolmesta relaatiosta.

Boyce-Codd-normaalimuoto

Lasketaan riippuvuuksien sulkeumat normaalimuotoisuuden selvittämiseksi ja ositetaan relaatiot Boyce-Codd-normaalimuotoon, jos ne eivät jo ole siinä.

Students-relaatio

ID -> name, ID -> birthDate, ID -> program, ID -> startYear, ID -> expStudyRight

- ⇨ X={ID}; X={ID, name, birthDate, program, startYear, expStudyRight}={ID}$^+$={A}$^+$
  Sulkeuma sisältää kaikki relaation attribuutit eli on BCNF:ssä ja ID on yliavain

program startYear -> expStudyRight

- ⇨ X={program startYear}; X={program, startYear, expStudyRight}={program startYear}$^+$={B}$^+$
  Sulkeuma ei sisällä kaikkia relaation attribuutteja eli ei ole BCNF:ssä

Jotta relaatio olisi BCNF:ssä, pitäisi kaikkien sulkeumien sisältää relaation kaikki attribuutit. {ID}$^+$ sisältää mutta {program startYear}$^+$ ei. Valitaan riippuvuuksista jälkimmäinen ja ositetaan: R1(program, startYear, expStudyRight) B:n sulkeuman mukaan ja R2(program, startYear, ID, name, birthDate) eli B ja ne

alkuperäisen relaation attribuutit, jotka eivät kuulu B:n sulkeumaan. Uusia relaatioita koskevat funktionaaliset riippuvuudet ja ositus:

R1: program startYear -> expStudyRight: $\{program\ startYear\}^+ = \{program, startYear, expStudyRight\}$

> Sulkeuma sisältää kaikki relaation R1 attribuutit eli on BCNF:ssä ja program ja startYear muodostavat yliavaimen.

R2: ID -> name, ID -> birthDate, ID -> program, ID -> startYear: $\{ID\}^+ = \{ID, name, birthDate, program, startYear\}$

> Sulkeuma sisältää kaikki relaation R2 attribuutit eli on BCNF:ssä ja ID on yliavain.

Ositus Boyce-Codd-normaalimuotoon on siis:

R1(<u>program</u>, <u>startYea</u>r, expStudyRight) ja R2(<u>ID</u>, name, birthDate, program, startYear)

Eli kuten aiemmin pohdittiin, Students kannattaa jakaa Students- ja StudyRight-relaatioihin.

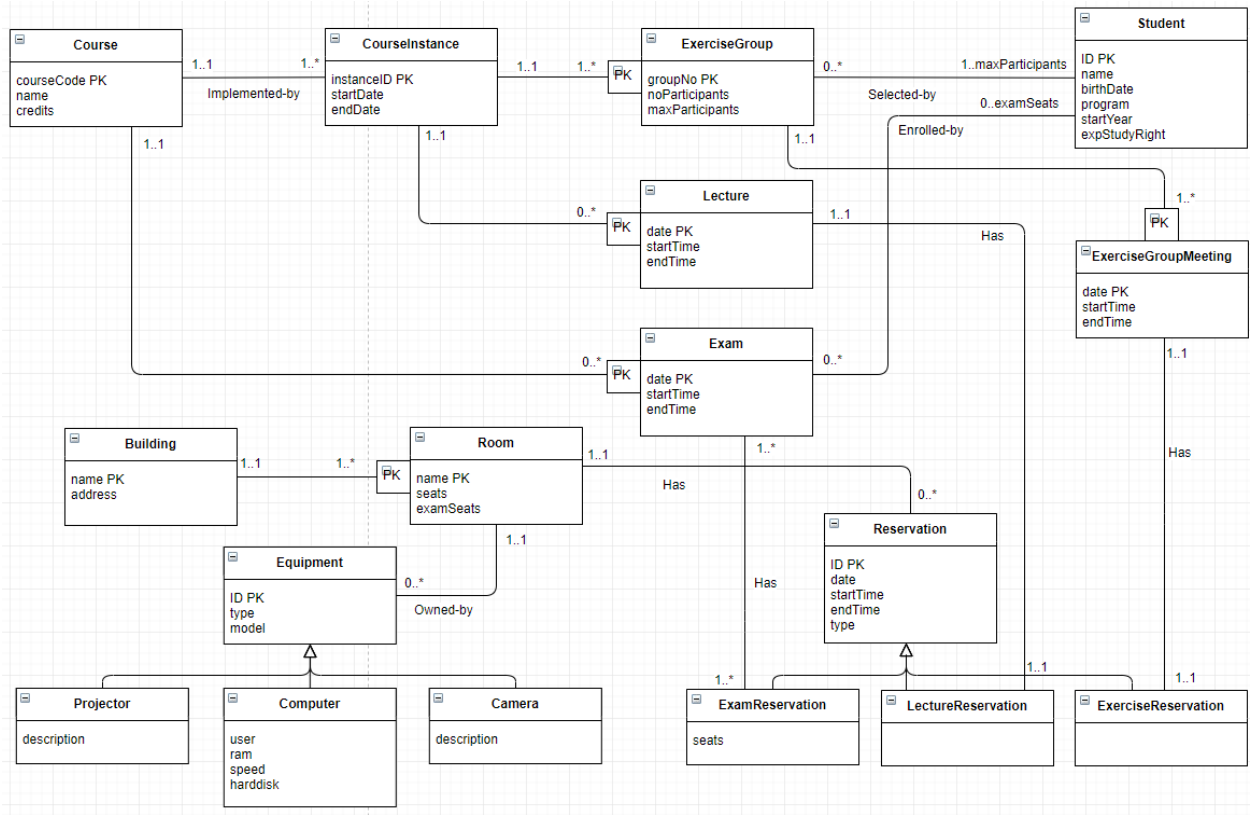
Muut relaatiot

Koska muissa relaatioissa avainattribuutti tai niiden joukko määrittää kaikki muut attribuutit alkuperäisen Students-relaation ID-attribuutin tapaan, avainattribuuttien sulkeuma sisältää kaikki relaation attribuutit eikä ei-avainriippuvuuksia ole, joten relaatiot ovat BCNF:ssä. HasExamReservation-, HasLectureReservation-, HasExerciseReservation-, SelectedBy- ja EnrolledBy-relaatioilla ei ole funktionaalisia riippuvuuksia, joten nekin ovat BCNF:ssä.

Ensimmäisen osan palautuksen jälkeen tehdyt muutokset

Korjattu UML-kaavio

Luokkien Exam ja ExamReservation välinen assosiaatio korjattiin yhdestä yhteen –tyypistä tyyppiin monesta moneen, koska salissa voi olla useita tenttejä samaan aikaan ja tentti voi vaatia useamman salin varaamisen. Lisäksi luokasta ExamReservation poistettiin attribuutti seats, koska luokan Room attribuutti examSeats riittää. SQL-hauissa verrataan tentti-ilmoittautumisia sen arvoon. Kuvassa 2 on esitetty korjattu UML-kaavio.



Kuva 2. Korjattu UML-kaavio.

Korjatut relaatiot

Opiskeluoikeuden voidaan olettaa olevan opiskelijan tiedoissa valmiiksi eikä Student-relaatiota ositeta kahdeksi relaatioksi. Sen attribuuttien program, startYear ja expStudyRight välillä ei ole funktionaalista riippuvuutta, koska opinto-oikeuden päättymisajankohtaa ei voida laskea kahden muun attribuutin arvoista. ImplementedBy-relaatio on tarpeeton ja se yhdistettiin CourseInstance-relaatioon. Lisäksi HasLectureReservation-relaatio yhdistettiin Lectures-relaatioon ja HasExerciseReservation ExerciseGroupMeeting-relaatioon. Varauksista tehtiin vain yksi yhteinen relaatio Reservations, johon yhdistetään HasReservation-relaatio. Siihen lisättiin attribuutiksi varauksen tyyppi. Korjattu lista relaatioista on esitetty alla.

Students(ID, name, birthDate, program, startYear, expStudyRight)

Courses(code, name, credits)

CourseInstances(instanceID, startDate, endDate, courseCode)

Lectures(instanceID, date, startTime, endTime, reservationID)

ExerciseGroups(instanceID, groupNo, noParticipants, maxParticipants)

ExerciseGroupMeetings(instanceID, groupNo, date, startTime, endTime, reservationID)

Exams(courseCode, date, startTime, endTime)

SelectedBy(groupNo, instanceID, studentID)

EnrolledBy(examDate, courseCode, studentID)

Buildings(name, address)

Rooms(buildingName, name, seats, examSeats)

Reservations(ID, date, startTime, endTime, roomName, buildingName, type)

HasExamReservation(courseCode, date, reservationID)

OwnedBy(equipmentID, buildingName, roomName)

Equipments(ID, type, model)

Projectors(ID, description)

Computers(ID, user, ram, speed, harddisk)

Cameras(ID, description)

Tietokanta SQL-muotoon

Luontikomennot

```
CREATE TABLE Students(
        ID TEXT PRIMARY KEY NOT NULL,
        name TEXT NOT NULL,
        birthdate TEXT NOT NULL,
        program TEXT NOT NULL,
        startYear INTEGER NOT NULL,
        expStudyRight TEXT NOT NULL
);

CREATE TABLE Courses(
        code TEXT PRIMARY KEY NOT NULL,
        name TEXT NOT NULL,
        credits REAL
);

CREATE TABLE CourseInstances(
        instanceID INTEGER PRIMARY KEY NOT NULL,
        startDate TEXT NOT NULL,
        endDate TEXT NOT NULL,
        courseCode TEXT NOT NULL,
        FOREIGN KEY (courseCode) REFERENCES Courses(code)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Buildings(
        name TEXT PRIMARY KEY NOT NULL,
        address TEXT DEFAULT "?"
);

CREATE TABLE Rooms(
        buildingName TEXT,
        name TEXT NOT NULL,
        seats INTEGER NOT NULL,
        examSeats INTEGER NOT NULL,
        PRIMARY KEY (buildingName, name)
        FOREIGN KEY (buildingName) REFERENCES Buildings(name)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

```
CREATE TABLE Reservations(
        ID INTEGER PRIMARY KEY NOT NULL,
        date TEXT NOT NULL,
        startTime TEXT NOT NULL,
        endTime TEXT NOT NULL,
        roomName TEXT,
        buildingName TEXT,
        type TEXT NOT NULL,
        FOREIGN KEY (buildingName, roomName) REFERENCES Rooms(buildingName,
                name)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Lectures(
        instanceID INTEGER,
        date TEXT NOT NULL,
        startTime TEXT NOT NULL,
        endTime TEXT NOT NULL,
        reservationID INTEGER,
        PRIMARY KEY (instanceID, date)
        FOREIGN KEY (instanceID) REFERENCES CourseInstances(instanceID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (reservationID) REFERENCES Reservations(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE ExerciseGroups(
        instanceID INTEGER,
        groupNo INTEGER NOT NULL,
        noParticipants INTEGER DEFAULT 0 CHECK (noParticipants <= maxParticipants)
NOT NULL,
        maxParticipants INTEGER NOT NULL,
        PRIMARY KEY (instanceID, groupNo)
        FOREIGN KEY (instanceID) REFERENCES CourseInstances(instanceID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

```
CREATE TABLE ExerciseGroupMeetings(
        instanceID INTEGER,
        groupNo INTEGER,
        date TEXT NOT NULL,
        startTime TEXT NOT NULL,
        endTime TEXT NOT NULL,
        reservationID INTEGER,
        PRIMARY KEY (instanceID, groupNo, date)
        FOREIGN KEY (instanceID, groupNo) REFERENCES ExerciseGroups(instanceID,
                groupNo)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (reservationID) REFERENCES Reservations(ID)
                 ON DELETE SET NULL
                 ON UPDATE CASCADE
);

CREATE TABLE Exams(
        courseCode TEXT,
        date TEXT NOT NULL,
        startTime TEXT NOT NULL,
        endTime TEXT NOT NULL,
        PRIMARY KEY (courseCode, date)
        FOREIGN KEY (courseCode) REFERENCES Courses(code)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE SelectedBy(
        groupNo INTEGER,
        instanceID INTEGER,
        studentID TEXT,
        PRIMARY KEY (groupNo, instanceID, studentID)
        FOREIGN KEY (groupNo, instanceID) REFERENCES ExerciseGroups(groupNo,
                instanceID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (studentID) REFERENCES Students(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

```
CREATE TABLE EnrolledBy(
        examDate TEXT,
        courseCode TEXT,
        studentID TEXT,
        PRIMARY KEY (examDate, courseCode, studentID)
        FOREIGN KEY (studentID) REFERENCES Students(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (courseCode, examDate) REFERENCES Exams(courseCode, date)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE HasExamReservation(
        courseCode TEXT,
        date TEXT,
        reservationID INTEGER,
        PRIMARY KEY (courseCode, date, reservationID)
        FOREIGN KEY (courseCode, date) REFERENCES Exams(courseCode, date)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (reservationID) REFERENCES Reservations(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Equipments(
        ID INTEGER PRIMARY KEY NOT NULL,
        type TEXT NOT NULL,
        model TEXT NOT NULL
);

CREATE TABLE OwnedBy(
        equipmentID INTEGER PRIMARY KEY,
        buildingName TEXT,
        roomName TEXT,
        FOREIGN KEY (equipmentID) REFERENCES Equipments(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY (buildingName, roomName) REFERENCES Rooms(buildingName,
                name)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

```
CREATE TABLE Projectors(
        ID INTEGER PRIMARY KEY,
        Description TEXT NOT NULL,
        FOREIGN KEY (ID) REFERENCES Equipments(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Computers(
        ID INTEGER PRIMARY KEY,
        user TEXT NOT NULL,
        ram INTEGER NOT NULL,
        speed REAL NOT NULL,
        harddisk INTEGER NOT NULL,
        FOREIGN KEY (ID) REFERENCES Equipments(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Cameras(
        ID INTEGER PRIMARY KEY,
        Description TEXT NOT NULL,
        FOREIGN KEY (ID) REFERENCES Equipments(ID)
                ON DELETE SET NULL
                ON UPDATE CASCADE

);
```

Esimerkkidata

Esimerkkidata (DB Project create.sql) on esitetty Liitteessä 1.

Hakemistot

```
CREATE INDEX StudentStartYear ON Students(startYear);

CREATE INDEX ReservationDate ON Reservations(date);

CREATE INDEX CourseStart ON CourseInstances(startTime);

CREATE INDEX CourseEnd ON CourseInstances(endTime);

CREATE INDEX ExamDate ON Exams(date);
```

```
CREATE INDEX EnrolledByDate ON EnrolledBy(examDate);
```

Näkymät

```
CREATE VIEW ExamInstances AS
    SELECT R.buildingName, R.name, RES.date, RES.startTime, RES.endTime,
    COUNT(studentID)
    FROM Reservations AS RES, EnrolledBy AS E, Rooms AS R, HasExamReservation AS
    H
    WHERE RES.date=H.date AND H.date=E.examDate AND RES.roomName=R.name
    AND   RES.buildingName=R.buildingName AND H.courseCode=E.courseCode AND
    RES.ID=H.reservationID
    GROUP BY R.Name, R.buildingName, RES.date, RES.startTime, RES.endTime;

CREATE VIEW GroupParticipantNo AS
    SELECT code, startDate, groupNo, COUNT(studentID)
    FROM ExerciseGroups AS E, SelectedBy AS S, Courses AS C, CourseInstances AS CI
    WHERE E.instanceID=S.instanceID AND S.instanceID=CI.instanceID AND
    C.code=CI.courseCode AND E.groupNo=S.groupNo
    GROUP BY code, startDate;
```

Perustelut

Tietotyypit

SQLItessa on vain tyypit INTEGER (kokonaisluvuille), REAL (desimaaliluvuille), TEXT (merkkijonoille) ja BLOB sellaisenaan tallennettavalle binääridatalle. Koska DATE- ja TIME-tyyppejä ei ole, käytetään kaikissa tauluissa päivämäärille (”1977-11-17”) ja ajoille (”12:15:00.0”) TEXT-tyyppiä.

Students-taulun ID on muotoa ”48960U” eli tyyppiä TEXT kuten ovat myös nimi ja koulutusohjelma. startYear-attribuutti on kokonaislukuna INTEGER, koska se on opintojen aloitusvuosi. Courses-taulun code-attribuutti on muotoa ”CS-A1150” eli TEXT kuten myös kurssin nimi name. Attribuutti credits voi olla desimaaliluku eli on REAL-tyyppiä. CourseInstances-taulun instanceID-attribuutti on järjestysluku eli INTEGER. Reservations-taulun ID-attribuutti on myös järjestyslukuna INTEGER. Sen type-attribuutti saa arvon ”Exam”, ”Lecture” tai ”Exercise group meeting”, joten se on tyyppiä TEXT. ExerciseGroups-taulun

groupNo-attribuutti saa joka kurssilla numeron yhdestä ylöspäin eli se on INTEGER. Osallistujien määrät noParticipants ja maxParticipants ovat kokonaislukuja eli tyyppiä INTEGER. Attribuutille noParticipants annetaan oletusarvoksi 0, koska ryhmiä luodessa ei ole vielä ilmoittautuneita opiskelijoita. Buildings-taulun name- ja address-attribuutit ovat merkkijonoja eli TEXT-tyyppiä. Rooms-taulun name on nimenä merkkijono eli TEXT. Paikkojen määrät seats ja examSeats ovat kokonaislukuja eli INTEGER. Laitteita kuvaavissa tauluissa ID on järjestysluku eli INTEGER. Attribuutit type, model ja description ovat merkkijonoja eli TEXT. Computers-taulussa käyttäjä on ”opettaja” tai ”oppilas” eli user-attribuutti on TEXT-tyyppiä. Keskusmuistin määrä eli ram-attribuutti on megatavuina ja kovalevyn koko eli harddisc gigatavuina, joten ne ovat kokonaislukuja eli INTEGER-tyyppiä. Suorittimen nopeus speed voi olla desimaaliluku eli sen tyyppi on REAL.

Eheysehdot

Taulujen avainattribuutit määritellään luontikomentoihin ”PRIMARY KEY” -merkinnällä ja viiteavaimet merkinnällä ”FOREIGN KEY (attribute) REFERENCES Table(attribute)”. Attribuutti courseCode tauluissa CourseInstances, Exams, EnrolledBy ja HasExamReservation vastaa Courses-taulun code-attribuuttia eli on viiteavain. SelectedBy- ja EnrolledBy-taulujen studentID-attribuutti viittaa Students-taulun ID-attribuuttiin. Lectures-, ExerciseGroups-, ExerciseGroupMeetings- ja SelectedBy-taulujen instanceID viittaa CourseInstances-taulun instanceID-attribuuttiin. ExerciseGroupMeetings- ja SelectedBy-taulujen groupNo viittaa ExerciseGroups-taulun samannimiseen attribuuttiin. Lectures-, ExerciseGroupMeetings- ja HasExamReservation-taulujen reservationID viittaa Reservations-taulun ID-attribuuttiin. EnrolledBy-taulun examDate ja HasExamReservation-taulun date viittaavat Exams-taulun date-attribuuttiin. Rooms-, Reservations- ja OwnedBy-taulujen buildingName viittaa Buildings-taulun name-attribuuttiin. Reservations- ja OwnedBy-taulujen roomName viittaa Rooms-taulun name-attribuuttiin. Cameras-, Computers- ja Projectors-taulujen equipmentID viittaa Equipments-taulun ID-attribuuttiin.

Viite-eheydestä huolehditaan esimerkiksi niin, että jos tietty kurssi poistetaan Courses-taulusta mutta ei haluta poistaa sen kurssikertoja, asetetaan CourseInstances-taulun courseCoden arvoksi NULL (”ON DELETE SET NULL”) ja muutettaessa kurssin koodia päivitetään se myös CourseInstances-tauluun (”ON UPDATE CASCADE”). Triggerillä saisi asetettua vanhan kurssikoodin courseCoden arvoksi. Samoin määritellään kaikki muut viiteavaimet.

Attribuuttien arvoja rajoitetaan asettamalla kaikille muille attribuuteille NOT NULL paitsi viiteavaimille, koska niiden kohdalla käytetään nolla-arvopolitiikkaa. Lisäksi on asetettu, että ExerciseGroups-taulun noParticipants-attribuutin arvon tulee olla pienempi tai yhtä suuri kuin maxParticipants-attribuutin arvo.

Hakemistot ja näkymät

SQLite tekee automaattisesti hakemiston taulun avaimen mukaan. Lisäksi ajankohtaan liittyviä hakuja voisi nopeuttaa hakemistoilla, koska todennäköisesti päivämäärät ja opiskelijoiden aloitusvuodet on klusteroitu. Kannattaisi siis tehdä hakemisto Students-taulun startYear-attribuutin, Reservations-taulun date-attribuutin, CourseInstances-taulun startTime- ja endTime-attribuuttien mukaan, Exams-taulun date-attribuutin ja EnrolledBy-taulun examDaten mukaan.

Tenttitilaisuuksia listaava näkymä ExamInstances, josta näkyy ilmoittautuneiden opiskelijoiden määrä, voisi olla hyödyllinen etsittäessä tenttisaleja, joissa on vielä tilaa. Lisäksi harjoitusryhmien listaus GroupParticipantNo , jossa näkyy ilmoittautuneiden opiskelijoiden määrä, voisi olla tarpeellinen.

Käyttötapaukset

Selostus käyttötapauksista ja SQLiteStudio-ympäristössä ajetuista SQL-käskyistä sekä listaukset niiden tulostuksista (DB Project queries.sql) on esitetty Liitteessä 2.

Esimerkkitietueiden lisääminen

Students

```
INSERT INTO Students
VALUES("112233", "Teemu Teekkari", "1985-11-17", "TIK", 2007, "2011-08-30");

INSERT INTO Students
VALUES("123456", "Tiina Teekkari", "2000-08-18", "TUTA", 2019, "2023-08-30");

INSERT INTO Students
VALUES("218311", "Kari Kemisti", "1995-05-06", "KEM", 2015, "2023-08-30");

INSERT INTO Students
VALUES("224411", "Maija Virtanen", "1999-03-17", "AUT", 2019, "2023-08-30");

INSERT INTO Students
VALUES("433511", "Maija Virtanen", "1997-01-03", "TUTA", 2016, "2024-08-30");

INSERT INTO Students
VALUES("512442", "Kalle Rantanen", "2000-08-19", "TIK", 2019, "2023-08-30");

INSERT INTO Students
VALUES("212434", "Pekka Lampinen", "1996-12-01", "BIZ", 2015, "2019-08-30");

INSERT INTO Students
VALUES("442255", "Teemu Teekkari", "1998-06-29", "TFM", 2018, "2026-08-30");

INSERT INTO Students
VALUES("565545", "Anne Korhonen", "2000-04-25", "AUT", 2019, "2023-08-30");

INSERT INTO Students
VALUES("441232", "Minna Kalaoja", "1995-11-04", "TIK", 2014, "2018-08-30");

INSERT INTO Students
VALUES("234155", "Mari Virtanen", "1997-10-12", "TFM", 2018, "2022-08-30");

INSERT INTO Students
VALUES("553311", "Ville Vallaton", "1993-09-28", "AUT", 2015, "2023-08-30");

INSERT INTO Students
VALUES("512434", "Pirjo Pirtea", "1997-02-06", "BIZ", 2018, "2022-08-30");
```

```
INSERT INTO Students
VALUES("987202", "Karri Lappinen", "1997-03-15", "TOK", 2019, "2027-08-30");
```

Courses

```
INSERT INTO Courses
VALUES("CS-A1150", "Databases", 5);

INSERT INTO Courses
VALUES("CS-A1120", "Programming 2", 5);

INSERT INTO Courses
VALUES("CSE-A1200", "Databases", 5);
```

CourseInstances

```
INSERT INTO CourseInstances
VALUES(5347, "2019-09-01", "2019-12-16", "CSE-A1200");

INSERT INTO CourseInstances
VALUES(6679, "2020-09-01", "2020-12-16", "CSE-A1200");

INSERT INTO CourseInstances
VALUES(4567, "2018-09-01", "2018-12-16", "CSE-A1200");

INSERT INTO CourseInstances
VALUES(6398, "2020-09-01", "2020-10-26", "CS-A1150");

INSERT INTO CourseInstances
VALUES(5698, "2019-09-01", "2019-10-26", "CS-A1150");

INSERT INTO CourseInstances
VALUES(6267, "2020-10-28", "2020-12-16", "CS-A1120");
```

Buildings

```
INSERT INTO Buildings
VALUES("Kandidaattikeskus", "Otakaari 1");

INSERT INTO Buildings
VALUES("TUAS-talo", "Maarintie 8");
```

Rooms

```
INSERT INTO Rooms
VALUES("Kandidaattikeskus", "A-sali", 570, 95);

INSERT INTO Rooms
VALUES("Kandidaattikeskus", "B-sali", 320, 50);

INSERT INTO Rooms
VALUES("TUAS-talo", "AS1", 300, 50);

INSERT INTO Rooms
VALUES("TUAS-talo", "TU4", 15, 15);

INSERT INTO Rooms
VALUES("TUAS-talo", "AS5", 28, 28);
```

Reservations

```
INSERT INTO Reservations
VALUES(65980, "2020-09-10", "12:15:00.0", "14:00:00.0", "AS5", "TUAS-talo", "Exercise group meeting");

INSERT INTO Reservations
VALUES(65985, "2020-09-17", "12:15:00.0", "14:00:00.0", "AS5", "TUAS-talo", "Exercise group meeting");

INSERT INTO Reservations
VALUES(65981, "2020-09-11", "12:15:00.0", "14:00:00.0", "AS5", "TUAS-talo", "Exercise group meeting");

INSERT INTO Reservations
VALUES(55981, "2019-09-10", "12:15:00.0", "14:00:00.0", "AS5", "TUAS-talo", "Exercise group meeting");

INSERT INTO Reservations
VALUES(45981, "2018-09-10", "12:15:00.0", "14:00:00.0", "AS5", "TUAS-talo", "Exercise group meeting");

INSERT INTO Reservations
VALUES(67891, "2020-09-10", "12:15:00.0", "14:00:00.0", "TU4", "TUAS-talo", "Exercise group meeting");
```

```
INSERT INTO Reservations
VALUES(55990, "2019-09-10", "12:15:00.0", "14:00:00.0", "TU4", "TUAS-talo", "Exercise
group meeting");

INSERT INTO Reservations
VALUES(65990, "2020-09-17", "12:15:00.0", "14:00:00.0", "TU4", "TUAS-talo", "Exercise
group meeting");

INSERT INTO Reservations
VALUES(33333, "2019-09-10", "09:15:00.0", "11:00:00.0", "AS1", "TUAS-talo", "Exercise
group meeting")

INSERT INTO Reservations
VALUES(65978, "2020-09-01", "12:15:00.0", "14:00:00.0", "A-sali", "Kandidaattikeskus",
"Lecture");

INSERT INTO Reservations
VALUES(66778, "2020-09-01", "10:15:00.0", "12:00:00.0", "B-sali", "Kandidaattikeskus",
"Lecture");

INSERT INTO Reservations
VALUES(66112, "2020-10-28", "12:15:00.0", "14:00:00.0", "B-sali", "Kandidaattikeskus",
"Lecture");

INSERT INTO Reservations
VALUES(66677, "2020-10-26", "12:00:00.0", "14:15:00.0", "B-sali", "Kandidaattikeskus",
"Exam");

INSERT INTO Reservations
VALUES(55443, "2019-10-26", "12:00:00.0", "14:15:00.0", "B-sali", "Kandidaattikeskus",
"Exam");

INSERT INTO Reservations
VALUES(65432, "2020-12-16", "12:00:00.0", "16:15:00.0", "A-sali", "Kandidaattikeskus",
"Exam");

INSERT INTO Reservations
VALUES(54321, "2019-12-16", "12:00:00.0", "16:15:00.0", "A-sali", "Kandidaattikeskus",
"Exam");

INSERT INTO Reservations
VALUES(66998, "2020-12-16", "12:00:00.0", "16:15:00.0", "A-sali", "Kandidaattikeskus",
"Exam");
```

```
INSERT INTO Reservations
VALUES(44444, "2018-12-16", "12:00:00.0", "16:15:00.0", "A-sali", "Kandidaattikeskus",
"Exam");
```

Lectures

```
INSERT INTO Lectures
VALUES(6679, "2020-09-01", "12:15:00.0", "14:00:00.0", 65978);

INSERT INTO Lectures
VALUES(6398, "2020-09-01", "12:15:00.0", "14:00:00.0", 66778);

INSERT INTO Lectures
VALUES(6267, "2020-10-28", "12:15:00.0", "14:00:00.0", 66112);
```

ExerciseGroups

```
INSERT INTO ExerciseGroups
VALUES(6679, 1, 3, 28);

INSERT INTO ExerciseGroups
VALUES(6679, 2, 2, 28);

INSERT INTO ExerciseGroups
VALUES(5347, 1, 28, 28);

INSERT INTO ExerciseGroups
VALUES(4567, 1, 27, 30);

INSERT INTO ExerciseGroups
VALUES(6398, 1, 1, 15);

INSERT INTO ExerciseGroups
VALUES(5698, 1, 7, 15);

INSERT INTO ExerciseGroups
VALUES(6267, 1, 1, 15);

INSERT INTO ExerciseGroups
VALUES(6679, 10, 28, 28);
```

ExerciseGroupMeetings

```
INSERT INTO ExerciseGroupMeetings
VALUES(6679, 1, "2020-09-10", "12:15:00.0", "14:00:00.0", 65980);

INSERT INTO ExerciseGroupMeetings
VALUES(6679, 1, "2020-09-17", "12:15:00.0", "14:00:00.0", 65985);

INSERT INTO ExerciseGroupMeetings
VALUES(6679, 2, "2020-09-11", "12:15:00.0", "14:00:00.0", 65981);

INSERT INTO ExerciseGroupMeetings
VALUES(5347, 1, "2019-09-10", "12:15:00.0", "14:00:00.0", 55981);

INSERT INTO ExerciseGroupMeetings
VALUES(4567, 1, "2018-09-10", "12:15:00.0", "14:00:00.0", 45981);

INSERT INTO ExerciseGroupMeetings
VALUES(6398, 1, "2020-09-10", "12:15:00.0", "14:00:00.0", 67891);

INSERT INTO ExerciseGroupMeetings
VALUES(5698, 1, "2019-09-10", "12:15:00.0", "14:00:00.0", 55990);

INSERT INTO ExerciseGroupMeetings
VALUES(6267, 1, "2020-09-17", "12:15:00.0", "14:00:00.0", 65990);
```

Exams

```
INSERT INTO Exams
VALUES("CS-A1150", "2020-10-26", "12:15:00.0", "14:00:00.0");

INSERT INTO Exams
VALUES("CS-A1150", "2019-10-26", "12:15:00.0", "14:00:00.0");

INSERT INTO Exams
VALUES("CS-A1120", "2020-12-16", "12:15:00.0", "16:00:00.0");

INSERT INTO Exams
VALUES("CSE-A1200", "2019-12-16", "12:15:00.0", "16:00:00.0");

INSERT INTO Exams
VALUES("CSE-A1200", "2020-12-16", "12:15:00.0", "16:00:00.0");

INSERT INTO Exams
VALUES("CSE-A1200", "2018-12-16", "12:15:00.0", "16:00:00.0");
```

```
INSERT INTO Exams
VALUES("CS-A1150", "2020-12-16", "12:15:00.0", "16:00:00.0");
```

SelectedBy

```
INSERT INTO SelectedBy
VALUES(1, 6679, "987202");

INSERT INTO SelectedBy
VALUES(1, 6679, "512434");

INSERT INTO SelectedBy
VALUES(1, 6679, "234155");

INSERT INTO SelectedBy
VALUES(2, 6679, "553311");

INSERT INTO SelectedBy
VALUES(2, 6679, "565545");

INSERT INTO SelectedBy
VALUES(1, 5347, "442255");

INSERT INTO SelectedBy
VALUES(1, 4567, "212434");

INSERT INTO SelectedBy
VALUES(1, 6398, "512442");

INSERT INTO SelectedBy
VALUES(1, 5698, "433511");

INSERT INTO SelectedBy
VALUES(6267, 1, "224411");
```

EnrolledBy

```
INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "123456");

INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "224411");
```

```
INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "433511");

INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "512442");

INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "442255");

INSERT INTO EnrolledBy
VALUES("2020-10-26", "CS-A1150", "565545");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "218311");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "565545");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "234155");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "553311");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "512434");

INSERT INTO EnrolledBy
VALUES("2019-10-26", "CS-A1150", "987202");

INSERT INTO EnrolledBy
VALUES("2020-12-16", "CS-A1120", "224411");

INSERT INTO EnrolledBy
VALUES("2020-12-16", "CS-A1120", "123456");

INSERT INTO EnrolledBy
VALUES("2020-12-16", "CS-A1120", "433511");

INSERT INTO EnrolledBy
VALUES("2020-12-16", "CS-A1120", "442255");

INSERT INTO EnrolledBy
VALUES("2020-12-16", "CSE-A1200", "512442");
```

```
INSERT INTO EnrolledBy
VALUES("2020-12-16", "CSE-A1200", "565545");

INSERT INTO EnrolledBy
VALUES("2019-12-16", "CSE-A1200", "433511");

INSERT INTO EnrolledBy
VALUES("2018-12-16", "CSE-A1200", "212434");
```

HasExamReservation

```
INSERT INTO HasExamReservation
VALUES("CS-A1150", "2020-10-26", 66677);

INSERT INTO HasExamReservation
VALUES("CS-A1150", "2019-10-26", 55443);

INSERT INTO HasExamReservation
VALUES("CS-A1120", "2020-12-16", 65432);

INSERT INTO HasExamReservation
VALUES("CSE-A1200", "2019-12-16", 54321);

INSERT INTO HasExamReservation
VALUES("CSE-A1200", "2020-12-16", 66998);

INSERT INTO HasExamReservation
VALUES("CSE-A1200", "2018-12-16", 44444);
```

Equipments

```
INSERT INTO Equipments
VALUES(1, "computer", "1001");

INSERT INTO Equipments
VALUES(2, "computer", "1002");

INSERT INTO Equipments
VALUES(3, "computer", "1001");

INSERT INTO Equipments
VALUES(4, "computer", "1002");
```

```
INSERT INTO Equipments
VALUES(5, "computer", "1002");

INSERT INTO Equipments
VALUES(6, "computer", "1001");

INSERT INTO Equipments
VALUES(7, "computer", "1002");

INSERT INTO Equipments
VALUES(8, "computer", "1002");

INSERT INTO Equipments
VALUES(9, "computer", "1002");

INSERT INTO Equipments
VALUES(10, "computer", "1002");

INSERT INTO Equipments
VALUES(11, "computer", "1002");

INSERT INTO Equipments
VALUES(12, "computer", "1002");

INSERT INTO Equipments
VALUES(13, "computer", "1002");

INSERT INTO Equipments
VALUES(14, "computer", "1002");

INSERT INTO Equipments
VALUES(15, "computer", "1002");

INSERT INTO Equipments
VALUES(16, "computer", "1002");

INSERT INTO Equipments
VALUES(17, "camera", "543");


INSERT INTO Equipments
VALUES(18, "camera", "789");


INSERT INTO Equipments
VALUES(19, "camera", "543");
```

```
INSERT INTO Equipments
VALUES(20, "projector", "33");

INSERT INTO Equipments
VALUES(21, "projector", "33");

INSERT INTO Equipments
VALUES(22, "projector", "33");

INSERT INTO Equipments
VALUES(23, "projector", "99");

INSERT INTO Equipments
VALUES(24, "projector", "99");

INSERT INTO Equipments
VALUES(25, "projector", "99");

INSERT INTO Equipments
VALUES(26, "laserpointer", "plop");

INSERT INTO Equipments
VALUES(27, "laserpointer", "plip");

INSERT INTO Equipments
VALUES(28, "laserpointer", "plop");

INSERT INTO Equipments
VALUES(29, "laserpointer", "plip");

INSERT INTO Equipments
VALUES(30, "laserpointer", "plip");
```

OwnedBy

```
INSERT INTO OwnedBy
VALUES(1, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(2, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(3, "Kandidaattikeskus", "B-sali");
```

```
INSERT INTO OwnedBy
VALUES(4, "TUAS-talo", "AS1");

INSERT INTO OwnedBy
VALUES(5, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(6, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(7, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(8, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(9, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(10, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(11, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(12, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(13, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(14, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(15, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(16, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(17, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(18, "Kandidaattikeskus", "B-sali");

INSERT INTO OwnedBy
VALUES(19, "TUAS-talo", "AS1");
```

```
INSERT INTO OwnedBy
VALUES(20, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(21, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(22, "Kandidaattikeskus", "B-sali");

INSERT INTO OwnedBy
VALUES(23, "TUAS-talo", "AS1");

INSERT INTO OwnedBy
VALUES(24, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(25, "TUAS-talo", "AS5");

INSERT INTO OwnedBy
VALUES(26, "Kandidaattikeskus", "A-sali");

INSERT INTO OwnedBy
VALUES(27, "Kandidaattikeskus", "B-sali");

INSERT INTO OwnedBy
VALUES(28, "TUAS-talo", "AS1");

INSERT INTO OwnedBy
VALUES(29, "TUAS-talo", "TU4");

INSERT INTO OwnedBy
VALUES(30, "TUAS-talo", "AS5");
```

Projectors

```
INSERT INTO Projectors
VALUES(20, "projector1");

INSERT INTO Projectors
VALUES(21, "projector1");

INSERT INTO Projectors
VALUES(22, "projector1");
```

```
INSERT INTO Projectors
VALUES(23, "projector2");

INSERT INTO Projectors
VALUES(24, "projector2");

INSERT INTO Projectors
VALUES(25, "projector2");
```

Computers

```
INSERT INTO Computers
VALUES(1, "teacher", 8192, 2.7, 1000);

INSERT INTO Computers
VALUES(2, "teacher", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(3, "teacher", 8192, 2.7, 1000);

INSERT INTO Computers
VALUES(4, "teacher", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(5, "teacher", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(6, "teacher", 8192, 2.7, 1000);

INSERT INTO Computers
VALUES(7, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(8, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(9, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(10, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(11, "student", 4096, 2.7, 500);
```

```
INSERT INTO Computers
VALUES(12, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(13, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(14, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(15, "student", 4096, 2.7, 500);

INSERT INTO Computers
VALUES(16, "student", 4096, 2.7, 500);
```

Cameras

```
INSERT INTO Cameras
VALUES(17, "cam1");

INSERT INTO Cameras
VALUES(18, "cam2");

INSERT INTO Cameras
VALUES(19, "cam1");
```

Käyttötapaukset

TALLENNUKSET

Kurssien ja kurssikertojen lisäys

Lisätään uusi kurssi Sustainable Product and Service Design:

```
INSERT INTO Courses
VALUES("MUO-E8005", "Sustainable Product and Service Design", 4);
```

Lisätään uudelle kurssille kurssikerta keväälle 2021. Lisäyksessä haetaan suurin instanceID CourseInstance-taulusta, 6679, ja uuden kurssin instanceID-arvoksi tulee yhtä isompi 6680:

```
INSERT INTO CourseInstances
VALUES((SELECT MAX(instanceID) FROM CourseInstances)+1, "2021-01-08", "2021-03-16", "MUO-E8005");
```

Opiskelijoiden lisäys

Opiskelijaa lisättäessä tutkitaan ensin, mikä on seuraava vapaa opiskelijanumero eli haetaan suurin ID Students-taulusta:

```
SELECT MAX(ID)
FROM Students;
```

Tulostus:

	MAX(ID)
1	987202

Jos opiskelijoita ei vielä ole, tulos on NULL. Käyttöliittymä huolehtii tällöin ID:n arvoksi 1, muuten arvo on tulos+1. Lisätään uusi opiskelija Pilli Piipari Students-tauluun seuraavalla numerolla:

```
INSERT INTO Students
VALUES("987203", "Pilli Piipari", "2000-10-20", "TIK", 2020, "2024-08-30");
```

Yliopiston rakennusten ja salien lisäys

Lisätään uusi rakennus Väre:

```
INSERT INTO Buildings
VALUES("Väre", "Otaniementie 14");
```

Lisätään uusi Sali Väreeseen:

```
INSERT INTO Rooms
VALUES("Väre", "A123", 76, 13);
```

Luennon ja sen salivarausten lisäys

Salivaraus tehdään ennen luennon lisäystä, jotta saadaan varausnumero. Lisätään uusi salivaraus Sustainable Product and Service Design-kurssin luennolle. Ensin etsitään sali, joka on vapaa 8.1.2021 klo 12.15-14 ja jossa on paikkoja riittävästi muttei liikaa (50-100):

```
SELECT buildingName, name
FROM Rooms
WHERE seats>=50 AND seats<=100
    EXCEPT
SELECT buildingName, roomName
FROM Reservations
WHERE date="2021-01-08" AND ((startTime<="12:15:00.0" AND endTime>="12:15:00.0")
OR (startTime<="14:00:00.0" AND endTime>="14:00:00.0"));
```

Tulostus:

	buildingName	name
1	Väre	A123

Salivaraus voidaan tehdä Väre-saliin. Ensin haetaan suurin ID Reservations-taulusta eli 67891 ja uuden varauksen ID-arvoksi tulee yhtä isompi 67892:

```
SELECT MAX(ID)
FROM Reservations;
```

Jos varauksia ei vielä ole, tulos on NULL. Käyttöliittymä huolehtii tällöin ID:n arvoksi 1, muuten arvo on tulos+1. Lisätään salivaraus Reservation-tauluun:

```
INSERT INTO Reservations
VALUES(67892, "2021-01-08", "12:15:00.0", "14:00:00.0", "A123", "Väre", "Lecture");
```

Uudeksi instanceID:n arvoksi saatiin aiemmin 6680 ja uuden salivarauksen numero on siis 67892, joita käytetään luennon lisäämisessä Lectures-tauluun:

```
INSERT INTO Lectures
VALUES(6680, "2021-01-08", "12:15:00.0", "14:00:00.0", 67892);
```

Harjoitusryhmien lisäys

Lisätään uusi harjoitusryhmä kevään 2021 Sustainable Product and Service Design-kurssille. Ensin selvitetään seuraava vapaa ryhmän numero:

```
SELECT MAX(groupNo)
FROM ExerciseGroups
WHERE instanceID=6680;
```

Tulostus:

	MAX(groupNo)
1	*NULL*

Jos ryhmiä ei vielä ole, kuten tässä tapauksessa, tulos on NULL. Käyttöliittymä huolehtii tällöin groupNo:n arvoksi 1, muuten arvo on tulos+1.

```
INSERT INTO ExerciseGroups
VALUES(6680, 1, 3, 15);
```

Harjoitusryhmien aikojen ja varausten lisäys

Salivaraus täytyy tehdä ennen harjoitusryhmätapaamisen lisäämistä, jotta saadaan varausnumero. Lisätään uusi salivaraus Sustainable Product and Service Design-kurssin harjoitusryhmätapaamiselle. Ensin etsitään sali, joka on vapaa 10.1.2021 klo 12.15-14 ja jossa on paikkoja riittävästi muttei liikaa (50-100):

```
SELECT buildingName, name
FROM Rooms
WHERE seats>=50 AND seats<=100
    EXCEPT
SELECT buildingName, roomName
FROM Reservations
WHERE date="2021-01-10" AND ((startTime<="12:15:00.0" AND endTime>="12:15:00.0")
OR (startTime<="14:00:00.0" AND endTime>="14:00:00.0"));
```

Tulostus:

	buildingName	name
1	Väre	A123

Salivaraus voidaan siis tehdä Väre-saliin. Ensin haetaan suurin ID Reservations-taulusta eli 67892 ja uuden varauksen ID-arvoksi tulee yhtä isompi 67893:

```
SELECT MAX(ID)
FROM Reservations
```

```
INSERT INTO Reservations
VALUES(67893, "2021-01-10", "12:15:00.0", "14:00:00.0", "A123", "Väre", "Exercise group meeting");
```

Uudeksi instanceID:n arvoksi saatiin aiemmin 6680, groupNo:n arvoksi 1 ja uuden salivarauksen numero on siis 67893, joita käytetään harjoitusryhmätapaamisen lisäämisessä ExerciseGroupMeetings-tauluun:

```
INSERT INTO ExerciseGroupMeetings
VALUES(6680, 1, "2021-01-10", "12:15:00.0", "14:00:00.0", 67893);
```

Opiskelijan ilmoittautuminen harjoitusryhmään

Opiskelija lisätään harjoitusryhmään lisäämällä kurssin instanceID, harjoitusryhmän groupNo ja opiskelijan ID SelectedBy-tauluun, jos sinne mahtuu. Käyttöliittymä tarkistaa mahtumisen päivittämällä ExerciseGroups-taulun kyseisen harjoitusryhmän monikkoa, mikä onnistuu, jos noParticipants on pienempi kuin maxParticipants:

```
UPDATE ExerciseGroups
SET noParticipants=(SELECT noParticipants FROM ExerciseGroups WHERE groupNo=1 AND instanceID=6680)+1
WHERE groupNo=1 AND instanceID=6680;
```

Tässä tapauksessa osallistujien määrä kasvaa kolmesta neljään ja koska käsky onnistui, lisätään opiskelija:

```
INSERT INTO SelectedBy
VALUES(1, 6680, "987203");
```

Tentin ja salivarauksen lisäys

Lisätään uusi tentti Sustainable Product and Service Design-kurssille:

```
INSERT INTO Exams
VALUES("MUO-E8005", "2020-12-16", "12:15:00.0", "16:00:00.0");
```

Etsitään salit, joissa on vapaata tai tentti 16.12.2020 klo 12-16.15 ja 25 vapaata paikkaa. Kaikista tarpeeksi isoista saleista poistetaan salit, joissa on varaus, ja tuloksena saatuihin saleihin yhdistetään salit, joissa on jo tenttivaraus samaan aikaan mutta yhä 25 paikkaa vapaana. Todennäköisesti haun voisi hoitaa yksinkertaisemminkin.

```
SELECT buildingName, name
FROM Rooms
WHERE examseats>=25
      EXCEPT
SELECT buildingName, roomName
FROM Reservations
WHERE date="2020-12-16" AND ((startTime<="12:00:00.0" AND endTime>="12:00:00.0")
OR (startTime<="16:15:00.0" AND endTime>="16:15:00.0"))
      UNION
SELECT R.buildingName, R.name
FROM Reservations AS RES, EnrolledBy AS E, Rooms AS R, HasExamReservation AS H
WHERE RES.date="2020-12-16" AND RES.date=H.date AND H.date=E.examDate AND
RES.roomName=R.name AND RES.buildingName=R.buildingName AND
H.courseCode=E.courseCode AND startTime="12:00:00.0" AND endTime<="16:15:00.0" AND
RES.ID=H.reservationID
GROUP BY R.Name, R.buildingName
HAVING COUNT(studentID)<=examSeats-25;
```

Tulostus:

	buildingName	name
1	Kandidaattikeskus	A-sali
2	Kandidaattikeskus	B-sali
3	TUAS-talo	AS1
4	TUAS-talo	AS5

Näistä valitaan A-sali ja lisätään varaukset Reservations- ja HasExamReservation-tauluihin:

```
INSERT INTO Reservations
VALUES(22222, "2020-12-16", "12:00:00.0", "16:15:00.0", "A-sali", "Kandidaattikeskus",
"Exam");

INSERT INTO HasExamReservation
VALUES("MUO-E8005", "2020-12-16", 22222);
```

Tentti-ilmoittautuminen

Ilmoitetaan Pilli Piipari (987203) Sustainable Product and Service Design -kurssin tenttiin 16.12.2020.

```
INSERT INTO EnrolledBy
VALUES("2020-12-16", "MUO-E8005", "987203");
```

HAUT

Kurssi ja harjoitusryhmä, jolle sali oli varattu viime vuonna tiettyyn aikaan

Etsitään kurssi ja sen harjoitusryhmä, jolle TUAS-talon AS5-sali oli varattu 10.9.2019 klo 13:

```
SELECT code, name, groupNo
FROM Courses AS C, ExerciseGroupMeetings AS E, Reservations AS R, CourseInstances AS CI
WHERE reservationID=ID AND roomName="AS5" AND buildingName="TUAS-talo" AND R.date="2019-09-10" AND E.startTime<="13:00:00.0" AND E.endTime>="13:00:00.0" AND E.instanceID=CI.instanceID AND C.code=CI.courseCode;
```

Tulostus:

	code	name	groupNo
1	CSE-A1200	Databases	1

Kurssit tiettynä ajankohtana

Haetaan syksyn 2020 kurssien koodit ja nimet:

```
SELECT code, name
FROM Courses
WHERE code IN
   (SELECT courseCode
    FROM CourseInstances
    WHERE startDate > "2020-08-01" AND endDate < "2020-12-31");
```

Tulostus:

	code	name
1	CS-A1120	Programming 2
2	CS-A1150	Databases
3	CSE-A1200	Databases

Kurssin tentit halutulla aikavälillä

Etsitään Programming 2-kurssin syksyn 2020 tentit:

```
SELECT DISTINCT name, date
FROM Courses AS C, Exams AS E
WHERE C.code="CS-A1120" AND E.courseCode="CS-A1120" AND date>="2020-08-30"
AND date<="2020-12-31";
```

Tulostus:

	name	date
1	Programming 2	2020-12-16

Milloin haluttu kurssi on järjestetty tai se tullaan järjestämään

Etsitään CSE-A1200 Databases -kurssin järjestämisajankohdat:

```
SELECT startDate, endDate
FROM Courses AS C, CourseInstances AS CI
WHERE C.code=CI.courseCode AND name="Databases" AND C.code="CSE-A1200";
```

Tulostus:

	startDate	endDate
1	2018-09-01	2018-12-16
2	2019-09-01	2019-12-16
3	2020-09-01	2020-12-16

Tietyn kurssikerran luennot

Etsitään kurssin CSE-A1200 Databases syksyn 2020 luennot:

```
SELECT date, L.startTime, L.endTime
FROM Lectures AS L, CourseInstances AS CI
WHERE CI.startDate="2020-09-01" AND CI.endDate="2020-12-16" AND
CI.instanceID=L.instanceID AND courseCode="CSE-A1200";
```

Tulostus:

	date	startTime	endTime
1	2020-09-01	12:15:00.0	14:00:00.0

Tietyn kurssikerran harjoitusryhmät

Etsitään kurssin CSE-A1200 Databases syksyn 2020 harjoitusryhmät:

```
SELECT groupNo
FROM ExerciseGroups AS E, CourseInstances AS CI
WHERE CI.startDate="2020-09-01" AND CI.endDate="2020-12-16" AND
CI.instanceID=E.instanceID AND courseCode="CSE-A1200";
```

Tulostus:

	groupNo
1	1
2	2
3	10

Kurssin ryhmät, joissa on tilaa

Etsitään kurssin CSE-A1200 Databases syksyn 2020 harjoitusryhmät, joissa on tilaa:

```
SELECT groupNo
FROM ExerciseGroups AS E, CourseInstances AS CI
WHERE CI.startDate="2020-09-01" AND CI.endDate="2020-12-16" AND
CI.instanceID=E.instanceID AND courseCode="CSE-A1200" AND
noParticipants<maxParticipants;
```

Tulostus:

	groupNo
1	1
2	2

Koska ja missä tietty harjoitusryhmä kokoontuu

Etsitään, missä ja milloin kurssin CSE-A1200 Databases syksyn 2020 harjoitusryhmä 1 kokoontuu:

```
SELECT R.date, R.startTime, R.endTime, R.buildingName, R.roomName
FROM ExerciseGroupMeetings AS E, CourseInstances AS CI, Reservations AS R
WHERE CI.startDate="2020-09-01" AND CI.endDate="2020-12-16" AND
CI.instanceID=E.instanceID AND courseCode="CSE-A1200" AND groupNo=1 AND
E.reservationID=R.ID;
```

Tulostus:

	date	startTime	endTime	buildingName	roomName
1	2020-09-10	12:15:00.0	14:00:00.0	TUAS-talo	AS5
2	2020-09-17	12:15:00.0	14:00:00.0	TUAS-talo	AS5

Sali, jossa on vähintään haluttu määrä paikkoja ja joka on vapaa haluttuna aikana

Etsitään ensin salit, joissa on yli 50 paikkaa, ja leikataan niistä pois salit, joissa on varaus 1.9.2020 klo 12.15-14 (A-sali):

```
SELECT buildingName, name
FROM Rooms
WHERE seats>50
    EXCEPT
SELECT buildingName, roomName
FROM Reservations
WHERE date="2020-09-01" AND ((startTime<="12:15:00.0" AND endTime>="12:15:00.0")
OR (startTime<="14:00:00.0" AND endTime>="14:00:00.0"))
```

Tulostus:

	buildingName	name
1	Kandidaattikeskus	B-sali
2	TUAS-talo	AS1
3	Väre	A123

Mihin tarkoitukseen määrätty sali on varattu haluttuna aikana

Etsitään A-salin varauksen tyyppi 16.12.2020 12.15-16.

```
SELECT DISTINCT type
FROM Reservations
WHERE roomName="A-sali" AND buildingName="Kandidaattikeskus" AND date="2020-12-
16" AND ((startTime<="12:15:00.0" AND endTime>="12:15:00.0") OR
(startTime<="16:00:00.0" AND endTime>="16:00:00.0"));
```

Tulostus:

	type
1	Exam

Kaikki halutulle kurssikerralle ilmoittautuneet opiskelijat

Haetaan Databases-kurssin (CSE-A1200) kurssikerralle 6679 ilmoittautuneet opiskelijat:

```
SELECT studentID, name
FROM SelectedBy, Students
WHERE instanceID=6679 AND studentID=ID;
```

Tulostus:

	studentID	name
1	987202	Karri Lappinen
2	512434	Pirjo Pirtea
3	234155	Mari Virtanen
4	553311	Ville Vallaton
5	565545	Anne Korhonen

Kaikki haluttuun harjoitusryhmään ilmoittautuneet opiskelijat

Haetaan Databases-kurssin (CSE-A1200) kurssikerran 6679 ryhmään 1 ilmoittautuneet opiskelijat:

```
SELECT studentID, name
FROM SelectedBy, Students
WHERE instanceID=6679 AND groupNo=1 AND studentID=ID;
```

Tulostus:

	studentID	name
1	234155	Mari Virtanen
2	512434	Pirjo Pirtea
3	987202	Karri Lappinen

Kaikki haluttuun tenttiin ilmoittautuneet opiskelijat

Haetaan opiskelijat, jotka ovat ilmoittautuneet Databases-kurssin (CSE-A1200) 16.12.2020 järjestettävään tenttiin:

```
SELECT studentID, name
FROM EnrolledBy, Students
WHERE studentID=ID AND courseCode="CSE-A1200"AND examDate="2020-12-16";
```

Tulostus:

	studentID	name
1	512442	Kalle Rantanen
2	565545	Anne Korhonen

Halutun kurssin harjoitusryhmät, joissa on vielä tilaa

Etsitään syksyllä 2020 järjestettävän Programming 2-kurssin ("CS-A1120") harjoitusryhmät, joissa osallistujien määrä on vähemmän kuin maksimimäärä:

```
SELECT groupNo
FROM CourseInstances AS C, ExerciseGroups AS E
WHERE C.instanceID=E.instanceID AND startDate>"2020-08-01" AND endDate<"2020-12-
31" AND courseCode="CS-A1120" AND noParticipants<maxParticipants;
```

Tulostus:

	groupNo
1	1

Salin laitteet

Haetaan A-salin laitteiden ID:t ja tyypit:

```
SELECT ID, type
FROM Equipments
WHERE ID IN
   (SELECT equipmentID
    FROM OwnedBy
    WHERE roomName="A-sali"
    );
```

Tulostus:

	ID	type
1	1	computer
2	2	computer
3	17	camera
4	20	projector
5	21	projector
6	26	laserpointer