

# DATA.DB.210 - Projektitehtävä, Vaihe 1

## Antikvariaattijärjestelmä

Ryhmä 1

Elias Peltonen, Taisto Palo ja Roope Lindroos

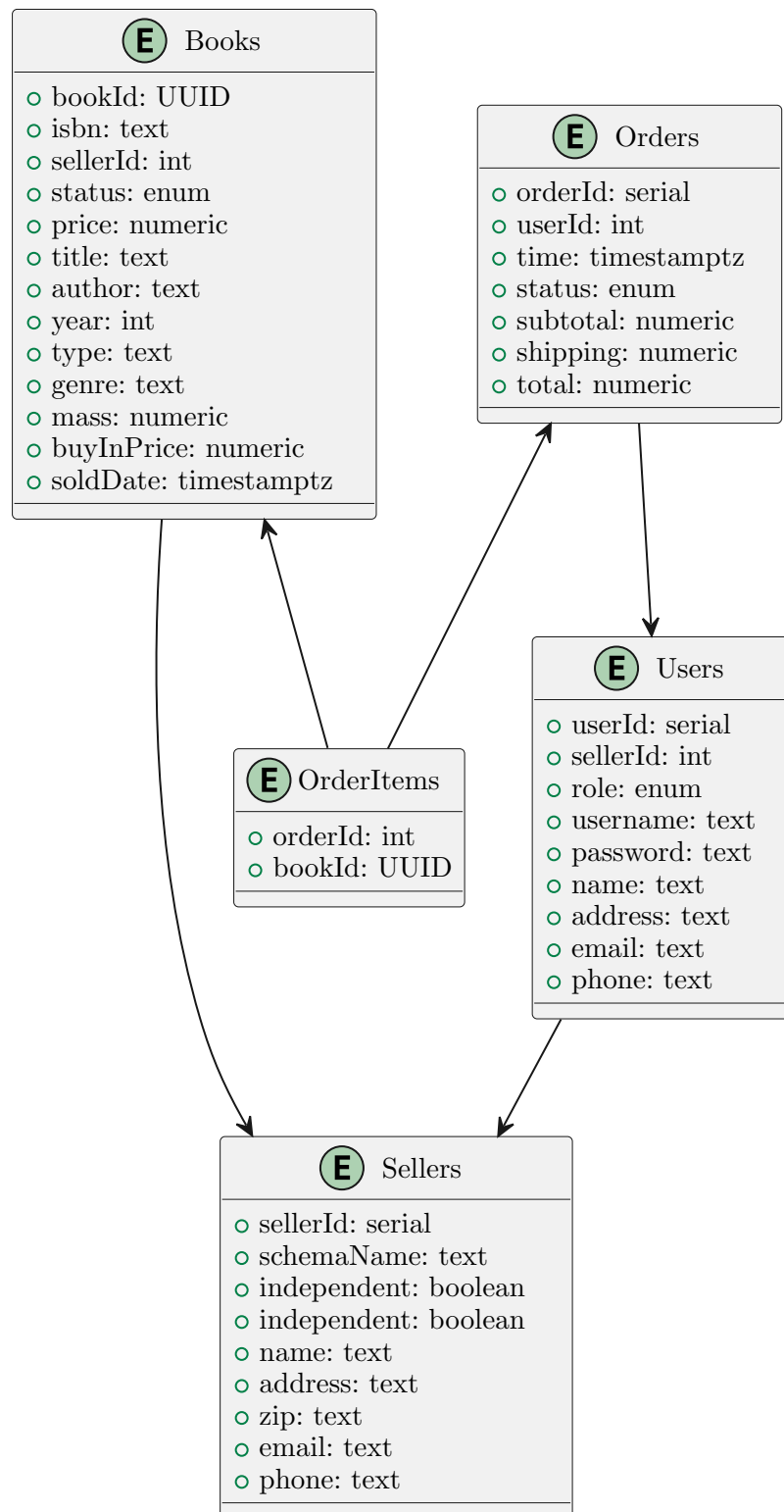
20. tammikuuta 2025

# Sisällys


<b>1</b>	<b>UML-kaaviot</b>	<b>3</b>
1.1	Keskustietokanta . . . . .	3
1.2	Yksittäisen divarin tietokanta . . . . .	4
<b>2</b>	<b>Tietokantakaavio tekstimuodossa</b>	<b>4</b>
<b>3</b>	<b>Näkymät</b>	<b>4</b>
3.1	Johdettavat tiedot . . . . .	4
3.2	Raporttien näkymät . . . . .	4
3.2.1	Raportti R1 . . . . .	4
3.2.2	Raportti R2 . . . . .	5
3.2.3	Raportti R3 . . . . .	5
3.2.4	Raportti R4 . . . . .	5
<b>4</b>	<b>Tapahtumakuvaukset</b>	<b>5</b>
4.1	Tapahtuma T1 . . . . .	5
4.1.1	Kirjautuminen . . . . .	5
4.1.2	Rekisteröityminen . . . . .	5
4.2	Tapahtuma T2 . . . . .	6
4.3	Tapahtuma T3 . . . . .	6
4.4	Tapahtuma T4 . . . . .	6
4.5	Tapahtuma T5 . . . . .	7
4.6	Tapahtuma T6 . . . . .	8
4.7	Tapahtuma T7 . . . . .	8
4.8	Tapahtuma T8 . . . . .	8
<b>5</b>	<b>SQL-luontilauseet</b>	<b>10</b>
<b>6</b>	<b>UML-kaavion muunnos tietokantakaavioksi</b>	<b>12</b>
<b>7</b>	<b>Attribuuttien arvoalueet ja rajoitukset</b>	<b>13</b>
7.1	Sellers-taulu . . . . .	13
7.2	Books-taulu . . . . .	13
7.3	Users-taulu . . . . .	13
7.4	Orders-taulu . . . . .	14
7.5	OrderItems-taulu . . . . .	14
7.6	Skeema . . . . .	14
<b>8</b>	<b>Toteutusvälineet</b>	<b>14</b>
8.1	Palvelin . . . . .	14
8.2	Käyttöliittymä . . . . .	14
8.3	Kehitysympäristö . . . . .	15

# 1 UML-kaaviot

## 1.1 Keskustietokanta



## 1.2 Yksittäisen divarin tietokanta

 Books
<ul style="list-style-type: none"><li>○ bookId: UUID</li><li>○ isbn: text</li><li>○ sellerId: int</li><li>○ status: enum</li><li>○ price: numeric</li><li>○ title: text</li><li>○ author: text</li><li>○ year: int</li><li>○ type: text</li><li>○ genre: text</li><li>○ mass: numeric</li><li>○ buyInPrice: numeric</li><li>○ soldDate: timestampz</li></ul>

## 2 Tietokantakaavio tekstimuodossa

**central.Sellers**(sellerId, schemaName, independent, name, address, zip, city, email, phone)

**central.Books**(bookId, isbn, sellerId, status, price, title, author, year, type, genre, mass, buyInPrice, soldDate)

**central.Users**(userId, role, sellerId, username, password, name, address, zip, city, email, phone)

**central.Orders**(orderId, userId, time, status, subtotal, shipping, total)

**central.OrderItems**(orderId, bookId)

**D1.Books**(bookId, isbn, price, title, author, year, type, genre, mass, buyInPrice, soldDate)

## 3 Näkymät

### 3.1 Johdettavat tiedot

- Tilauksen kokonaispaino ja jakaminen painoluokittain
- Tilauksessa olevien niteiden kokonaishinta
- Tilauksen postituksen kustannukset
- Tilauksen kokonaishinta (niteiden ja postituksen yhteishinta)

### 3.2 Raporttien näkymät

#### 3.2.1 Raportti R1

Haku kohdistuu suoraan Books-tauluun, joten haussa käytettyä tietoa ei tarvitse varsinaisesti johtaa.

### 3.2.2 Raportti R2

Raportti voidaan esittää tietokantakaaviomaisena seuraavasti:

**R2**(genre, totalSoldPrice, averagePrice)

Genre (luokka) saadaan suoraan Books-tilusta. totalSoldPrice johdetaan luokkakohtaisesti kaikkien saman luokan niteille laskemalla näiden kokonaismyyntihinta summaamalla kaikkien luokan niteiden attribuutti price. averagePrice voidaan johtaa totalSoldPrice attribuutin perusteella jakamalla se samassa luokassa olevien niteiden kokonaismäärällä.

### 3.2.3 Raportti R3

Raportti voidaan esittää tietokantakaaviomaisena seuraavasti:

**R3**(usersName, boughtBooksCount)

Käyttäjänimet saadaan suoraan Users-tilusta. Teemme haun Order-tiluun ryhmitellen käyttäjänimen perusteella ja rajaten hakutulokset viime vuoden ajalta. Viime vuosi voidaan johtaa nykyisestä vuodesta vähentämällä siitä yksi (ei kiinnitetä). Asiakkaan ostamat niteet saadaan Order-tilusta ja lopuksi kaikkien niteiden summa voidaan laskea asiakaskohtaisesti. Haku on tarkoitus optimoida implementaatiovaiheessa eli tässä demonstroidaan esimerkillä, kuinka tulokset on mahdollista johtaa nykyisestä tietokannasta.

### 3.2.4 Raportti R4

Raportti R4 on jatkojalostus raporttiin R1, jolloin vakiojärjestys toteutetaan relevanssin perusteella. Tämä voidaan toteuttaa esimerkiksi vektorihaulla tai muulla vastaavalla menetelmällä.

## 4 Tapahtumakuvaukset

### 4.1 Tapahtuma T1

*Asiakas kirjautuu tai rekisteröityy*

#### 4.1.1 Kirjautuminen

1. Lue syötteenä käyttäjän antamat username ja password.
2. Lue central.Users-relaatiosta rivi, jossa username = :username.
3. Vertaile syötteenä tullutta password-arvoa tallennettuun (esim. hashattuun) password-arvoon.
4. Jos salasana on oikea, palauta tieto onnistuneesta kirjautumisesta (esim. asiakkaan userId ja rooli).

#### 4.1.2 Rekisteröityminen

1. Lue syötteenä käyttäjän rekisteröintitiedot (esim. username, password, name, address, ym.).

2. Lue central.Users-relaatiosta, onko samaa username-arvoa jo olemassa.
3. Jos ei ole, kirjoita uusi rivi central.Users-relaatioon (sis. roolin customer, hashatun salasanan ym.).
4. Palauta tieto onnistuneesta rekisteröitymisestä.

## 4.2 Tapahtuma T2

*Lisää uuden teoksen tiedot divarin D1 tietokantaan sekä keskustietokantaan*

1. Lue syötteenä kirjan tiedot (esim. title, author, isbn, year, genre, price, mass).
2. Kirjoita ensin teoksen perustiedot central.Books-relaatioon:
  - bookId generoidaan (UUID).
  - isbn, title, author, year, genre, ym. tallennetaan.
  - sellerId asetetaan vastaamaan D1:n sellerId-arvoa (haettava esim. central.Sellers-taulusta, jos ei tiedossa).
3. Kirjoita sama teos D1.Books-relaatioon käyttäen samaa bookId:tä (UUID):
  - Kopioi perustiedot (otsikko, kirjailija, hinta, paino, jne.).
  - sellerId asetetaan myös D1:n sellerId-arvoon.
4. Palauta tieto onnistuneesta tallennuksesta ja uuden teoksen tiedot.

## 4.3 Tapahtuma T3

*Lisää uusi myyntikappale teoksesta, jonka tiedot ovat jo kannassa (D2:n ylläpitäjä)*

1. Lue syötteenä tiedot, jotka yksilöivät olemassa olevan teoksen (esim. isbn tai jokin sisäinen tunniste).
2. Lue central.Books-relaatiosta onko kirjan perustiedot olemassa (vrt. isbn, title, author).
3. Jos perustiedot löytyvät, kirjoita central.Books-relaatioon uusi rivi
4. Kirjoita D2:n omaan skeemaan (esim. D2.Books) uusi rivi tai päivitys:
  - Käytä samaa bookId-tunnistetta, jos halutaan pitää yhteys keskusdivarin tietoon.
  - Aseta sellerId D2:n myyjän tunnisteeksi.
  - Aseta status = 'available'.
  - Aseta hinta, paino, ym. tiedot.

## 4.4 Tapahtuma T4

*Asiakas tekee yksittäisen kirjan tilauksen*

1. Asiakas kirjautuu/rekisteröityy (ks. T1).

2. Asiakas tekee haun (esim. hakukenttä):
  - Lue hakuehdot (otsikko, kirjailija, genre ym.).
  - Lue central.Books-relaatiosta vastaavat teokset, joilla status = 'available'.
3. Asiakas valitsee halutun kirjan ja tekee tilauksen:
  - Kirjoita uusi rivi central.Orders-relaatioon:
    - userId = asiakkaan userId.
    - status = 'pending'.
4. Kirjoita uusi rivi central.OrderItems-relaatioon (tilauksen ja kirjan välinen linkki).
  - Divari lähettää tilausvahvistuksen (sis. tilauksen hinta + toimituskulut)
  - Toimituskulujen laskenta (kirjeen painorajat) on johdettua tietoa, joten:
    - lue central.Books.mass tilatuille kirjoille
    - laske kokonaispaino
    - valitse postikulutaulukosta oikea hinta
  - kirjoita/päivitä central.Orders-relaatioon subtotal, shipping, total.
5. Asiakas maksaa tilauksen (tapahtuma maksurajapinnassa, ei välttämättä suoraa tietokantatoimintaa).
6. Divari lähettää tilauksen:
  - Päivitä central.Orders.status arvoksi 'completed'.
  - Päivitä central.Books.status arvoksi 'sold' niille teoksille, jotka kuuluvat tilaukseen.

## 4.5 Tapahtuma T5

*Asiakas tekee tilauksen, jonka paino ylittää 2000 g (tilaus jaetaan useaan erään)*

Huom: Yksinkertaistettuna laskemme postikulut ikään kuin yhden divarin teoksille.

1. Asiakas tekee tilauksen (kuten T4:n vaiheet 1-3).
2. Lue tilauksen kokonaistiedot central.OrderItems- ja central.Books-relaatioista:
  - Summataa tilattujen niteiden massat.
3. Totea, että kokonaispaino ylittää 2000 g.
  - Joudutaan jakamaan paketti useampaan erään.
  - Postikulut lasketaan esim. jakamalla teokset useampaan painoluokkaan tai sovittujen sääntöjen mukaan.
4. Kirjoita central.Orders-relaatioon lopulliset postikulut (tietueen shipping-kenttä).
5. Asiakas maksaa ja Divari lähettää (kuten T4, vaiheet 5-6).

## 4.6 Tapahtuma T6

*Toteuta triggeri, joka päivittää keskusdivarin automaattisesti, kun divarin tietokantaan tuodaan uusi myyntikappale*

1. Lue (triggerissä) uusi rivi, joka lisätään D3.Books-tauluun.
2. Trigger-funktion sisällä:
  - Tarkista, löytyvätkö teoksen perustiedot jo central.Books-relaatiosta (vrt. ISBN, tekijä, nimi).
  - Jos eivät löydy → kirjoita uusi rivi central.Books-relaatioon (luo UUID, täytä teostiedot).
  - Jos perustiedot löytyvät, tarkista onko sellerId = D3:n sellerId, ja jos ei, lisää/päivitä tallenne (riippuu toteutustavasta, halutaanko montaa riviä samaan ISBN:ään).
  - Aseta status = 'available' tai triggerin logiikan mukainen oletusarvo.
3. Tämän seurauksena keskusdivari pysyy automaattisesti synkronoituna jokaisen uuden myyntikappaleen osalta, kun se lisätään D3:n kantaan.

## 4.7 Tapahtuma T7

*Divarin tietokannassa (D1) on kirjoja, joita ei löydy keskustietokannasta - päivitä keskusdivari*

1. Lue kaikki D1.Books-rivit, joita ei löydy central.Books-relaatiosta (verrataan esim. ISBN-tunnusta, kirjan nimeä ja kirjoittajaa).
2. Jokaiselle puuttuvalle teokselle:
  - Kirjoita puuttuvat tiedot central.Books-relaatioon (luo uusi UUID, täytä perustiedot).
  - Aseta sellerId = D1:n tunniste (hae central.Sellers).
  - Aseta status, price, ym. kentät.
3. *Vaihtoehtoinen toteutus:* Jos käytössä on jokin "muuttunut/tallennettu" -lippu D1:n päässä, voidaan lukea vain ne rivit, jotka on merkitty uusiksi tai muuttuneiksi viimeisimmän synkronoinnin jälkeen.
4. Palauta tieto päivityksen onnistumisesta (esim. kirjattiin X uutta teosta keskustietokantaan).

## 4.8 Tapahtuma T8

*Divari D4:n data XML-muodossa - siirrä tiedot keskustietokantaan*

Oletetaan, että XML-data noudattaa annettua DTD-rakennetta (tehtävänannon liite 2) ja siinä on vähintään kaksi teosta, joista kummassakin kaksi nidettä.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE teokset [
  <!ELEMENT teokset (teos*)>
  <!ELEMENT teos (ttiedot,nide*)>
  <!ELEMENT ttiedot (nimi, tekija, isbn)>
  <!ELEMENT nide (hinta, paino*)>
  <!ELEMENT nimi (#PCDATA)>
  <!ELEMENT tekija (#PCDATA)>
  <!ELEMENT isbn (#PCDATA)>
  <!ELEMENT hinta (#PCDATA)>
  <!ELEMENT paino (#PCDATA)>
]>
<teokset>
  <teos>
    <ttiedot>
      <nimi>Elektran tytär</nimi>
      <tekija>Madeleine Brent</tekija>
      <isbn>9155430674</isbn>
    </ttiedot>
    <nide>
      <hinta>12.00</hinta>
      <paino>300</paino>
    </nide>
  </teos>

  <teos>
    <ttiedot>
      <nimi>Turms, kuolematon</nimi>
      <tekija>Mika Waltari</tekija>
      <isbn>9789510431122</isbn>
    </ttiedot>
    <nide>
      <hinta>15.00</hinta>
      <paino>400</paino>
    </nide>
  </teos>
</teokset>
```

1. Lue D4:n XML-aineisto (esim. tiedostosta tai rajapinnasta).
2. Parsi XML dokumentti ja erota:
  - Teoksen perustiedot (otsikko, kirjailija, ISBN, genre, julkaisu vuosi).
  - Jokaisen niteen tiedot (hinta, paino, status).

## 3. Jokaiselle teokselle:

- Tarkista, onko teos jo central.Books-relaatiossa (vertaa ISBN/otsikko).
- Jos ei ole, kirjoita uusi rivi perustiedoilla central.Books.

## 4. Sitten kullekin niteelle:

- Kirjoita tai päivitä central.Books, jos halutaan jokaiselle myyntikappaleelle oma rivi.
- Aseta sellerId = D4:n tunniste (lisää ensin D4 central.Sellers-relaatioon, jos sitä ei vielä ole).
- Aseta status, price, mass, ym. tiedot niteelle.

## 5. Mikäli D4 on täysin uusi myyjä:

- Kirjoita uusi rivi central.Sellers-relaatioon (schemaName "D4", osoitetiedot, ym.).
- Palauta tieto onnistuneesta siirrosta (esim. montako uutta teosta/nidettä lisättiin).

## 5 SQL-luontilauseet

```
-- Enable the UUID extension for globally unique IDs
```

```
CREATE EXTENSION IF NOT EXISTS "uuid-oss";
```

```
-- Initialize the Central Database
```

```
CREATE SCHEMA central;
```

```
-- Create enums
```

```
CREATE TYPE bookStatus AS ENUM('available', 'reserved', 'sold');
```

```
CREATE TYPE userRole AS ENUM('admin', 'seller', 'customer');
```

```
CREATE TYPE orderStatus AS ENUM('pending', 'completed');
```

```
-- Central Database Tables
```

```
CREATE TABLE central.Sellers (  
  sellerId SERIAL PRIMARY KEY,  
  schemaName TEXT NOT NULL,  
  independent BOOLEAN NOT NULL DEFAULT FALSE,  
  name TEXT NOT NULL,  
  address TEXT,  
  zip TEXT,  
  city TEXT,  
  email TEXT UNIQUE,  
  phone TEXT  
);
```

```
CREATE TABLE central.Books (  
    bookId UUID DEFAULT uuid_generate_v4 () PRIMARY KEY,  
    isbn TEXT NOT NULL,  
    sellerId INT REFERENCES central.Sellers (sellerId),  
    status bookStatus NOT NULL DEFAULT 'available',  
    price NUMERIC NOT NULL,  
    title TEXT NOT NULL,  
    author TEXT NOT NULL,  
    year INT,  
    type TEXT,  
    genre TEXT,  
    mass NUMERIC NOT NULL,  
    buyInPrice NUMERIC DEFAULT 0,  
    soldDate TIMESTAMPTZ  
);  
  
CREATE TABLE central.Users (  
    userId SERIAL PRIMARY KEY,  
    role userRole NOT NULL,  
    sellerId INT REFERENCES central.Sellers (sellerId),  
    username TEXT UNIQUE NOT NULL,  
    password TEXT NOT NULL, -- Must be hashed  
    name TEXT NOT NULL,  
    address TEXT,  
    zip TEXT,  
    city TEXT,  
    email TEXT UNIQUE,  
    phone TEXT  
);  
  
CREATE TABLE central.Orders (  
    orderId SERIAL PRIMARY KEY,  
    userId INT NOT NULL REFERENCES central.Users (userId),  
    time TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,  
    status orderStatus NOT NULL,  
    subtotal NUMERIC,  
    shipping NUMERIC,  
    total NUMERIC  
);  
  
CREATE TABLE central.OrderItems (  
    orderId INT NOT NULL REFERENCES central.Orders (orderId),
```

```
    bookId UUID NOT NULL REFERENCES central.Books (bookId),
    PRIMARY KEY (orderId, bookId)
);

-- Example of creating a seller schema and its tables
CREATE SCHEMA D1;

CREATE TABLE D1.Books (
    bookId UUID DEFAULT uuid_generate_v4 () PRIMARY KEY,
    isbn TEXT,
    price NUMERIC,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    year INT,
    type TEXT,
    genre TEXT,
    mass NUMERIC,
    buyInPrice NUMERIC,
    soldDate TIMESTAMPTZ
);
```

## 6 UML-kaavion muunnos tietokantakaavioksi

UML-kaavion perusteella luotiin tietokantaskeema keskustietokantaa varten (central).

Sellers-entiteetistä otettiin central.Scheman pääavaimeksi attribuutti bookId ja vierasavaimeksi attribuutti schemaName. Loput attribuutit lisättiin mukaan. Sellersillä (myyjä) tarkoitetaan yksittäistä keskustietokantaan kuuluvaa divaria.

Books-entiteetin perusteella luotiin central.Books, jonka pääavaimeksi tuli bookId ja vierasavaimeksi sellerId. Loput attribuutit lisättiin muuttamattomina mukaan. Books-taulussa jokainen rivi on yksittäinen nide, jonka yksilöi bookId (UUID). ISBN-attribuutin avulla saadaan rajattua kaikki yksittäisen teoksen niteet. Tällä rajauksella saamme laskettua teosten lukumäärät samalla säilyttäen yksittäisten niteiden yksilöinnin. Esimerkiksi samaa teosta voi olla usealla divarilla myynissä ja hinnoissa voi olla eroavaisuuksia.

Users-entiteetin perusteella luotiin central.Users, jonka pääavaimeksi valittiin userId ja vierasavaimeksi sinne valittiin sellerId. Loput attribuutit lisättiin kaavioon muuttumattomina perään. Users-taulu sisältää pääkäyttäjät, myyjät sekä asiakkaat. Käyttäjän rooli määritetään enum-tyyppin attribuutilla. Käyttäjällä *voi* olla sellerId, mikäli käyttäjän rooli on "seller".

Orders-entiteetin perusteella luotiin central.Orders, jonka pääavaimeksi valittiin orderId ja vierasavaimeksi userId. Loput entiteetin attribuutit lisättiin kaavioon mukaan. Tilaus-tauluun tallennetaan kokonaistilauksen summa kirjanpitoa ajatellen (attribuutti total) sekä pelkkien kirjojen osuun kokonaistilauksesta (attribuutti subtotal).

OrderItems-entiteetin perusteella lisättiin relaatio kirjojen ja tilausten välille, jotta tilaukseen liittyvät niteet voidaan tallentaa tietokantaan. Täten esimerkiksi voidaan tilausnumeron perusteella voidaan hakea kaikki siihen liittyvät niteet.

Central-skeeman lisäksi luotiin skeema D1 divaria varten, johon tallennetaan kyseisen divarin niteet. D1-skeemaan ei tarvitse tallentaa muuta tietoa, sillä keskusdivari huolehtii asikastietojen ja tilausten tallentamisesta.

## 7 Attribuuttien arvoalueet ja rajoitukset

### 7.1 Sellers-taulu

Myyjä-taulun pääavain on serial-tyyppinen ja myyjällä on myös tekstimuotoinen skeeman nimi, joka ei saa olla tyhjä. Mikäli myyjällä ei ole omaa tietokantaa (käyttää keskustietokantaa), skeeman nimi on "central". Attribuutti "independent" on arvoltaan "tosi", kun divarilla on oma erillinen tietokanta. Myyjän nimi ei saa olla tyhjä. Sähköpostien tulee olla uniikkeja. Muut osoite-/yhteystiedot eivät ole pakollisia ja ne ovat tekstimuotoisia.

### 7.2 Books-taulu

Kirja-taulun pääavain on UUID-tyyppinen (bookId), jotta niteiden tunnukset säilyvät yksilöivinä eri tietokantojen välillä. ISBN on pakollinen tekstimuotoinen attribuutti, joka yksilöi teokset (teoksista voi olla useampi nide). Niteellä on vierasavain sellerId, joka liittää niteen johonkin myyjään. SellerId:n avulla voidaan hakea niteitä myyjäkohtaisesti. Niteellä on attribuutti "status", joka ei saa olla tyhjä ja on oletuksena "available". Status on enum-tyyppinen ja sisältää arvot "available", "reserved" ja "sold". Niteellä on attribuutti hinta, joka ei saa olla "null" eli ilmaiseksi annettavat kirjat saavat hinnan 0. Niteellä on otsikko ja kirjailija, jotka ovat tekstimuotoisia pakollisia tietoja. Kirjalle voi lisätä myös kokonaislukumuotoisen julkaisuvuoden, mikäli se on tiedossa. Kirjan kuvausta voidaan tarkentaa lisäämällä tekstimuotoinen tyyppi ja genre. Tyyppi määrittää, onko kirja esimerkiksi sarjakuva vai romaani, ja genre määrittää kirjan luokan (esim. "jännitys"). Jokaisille kirjalle tulee lisätä myös kokonaislukumuotoinen massa (punnitse, jos et tiedä). Jokaiselle kirjalle voidaan merkitä sisäänostohinta. Mikäli sisäänostohintaa ei ole merkitty, saa se arvon 0. Niteellä on myös timestamp-tyylinen myyntipäivämäärä ja kellonaika, joka lisätään onnistuneen myyntitapahtuman yhteydessä.

### 7.3 Users-taulu

Käyttäjä-taulun pääavain on serial-muotoinen (userId). Käyttäjällä on pakollinen enum-tyyppinen rooli, joka sisältää arvot "customer", "seller" ja "admin". Käyttäjällä on vierasavain (sellerId), mikä lisätään vain käyttäjän roolin ollessa "seller". Käyttäjällä tulee olla yksilöivä tekstimuotoinen käyttäjänimi ja siihen liittyvä salasana. Salasana hajautetaan ja suolataan (hashing and salting) ennen tietokantaan tallentamista. Salasanan tallennus ei välttämättä vastaa oikean tietoturvallisen tuotantokelpoisen sovelluksen standardeja. Kirjautumistietoja tarvitaan sovelluksen toiminnan testaamisessa. Käyttäjä voi lisätä omiin tietoihinsa tekstimuotoisen nimen, osoitteen, postinumeron, kaupungin, sähköpostin ja puhelinnumeron. Näistä nimi on pakolli-

nen ja sähköpostin tulee olla yksilöivä. Käyttäjän yhteystiedot tarvitaan tilausvaiheessa, mutta tästä vastuussa on käyttöliittymä.

## 7.4 Orders-taulu

Tilaus-taulun pääavain on serial-muotoinen (orderId). Tilaustapahtumassa tallennetaan käyttäjän yksilöivä userId. Tilaustapahtuman aikaleima lisätään onnistuneen tilauksen yhteydessä. Tilauksella on enum-tyyppinen attribuutti status, joka voi saada arvot ”pending” ja ”completed”. Tietyn kuluneen ajanjakson jälkeen tilauksen statuksen ollessa edelleenkin ”pending”, varatut niteet palautetaan tilaan ”available” ja peruttu tilaus poistetaan. Asiakas voi myös itse perua tilauksen. Tilauksella on numeeriset attribuutit subtotal, shipping ja total. Subtotal käsittää tilaukseen kuuluvien niteiden kokonaishinnan, shipping postikulut ja total tilauksen kokonaishinnan.

## 7.5 OrderItems-taulu

Tilaustuotteet-taulussa lisätään relaatio tilausten ja niteiden välille, eli se toimii ns. linkkinä näiden välillä. Tilaustuote-taulussa on vain attribuutit orderId ja bookId, jotka ovat samanaikaisesti pääavaimia sekä vierasavaimia.

## 7.6 Skeema

Tietokannassa käytetään skeemaa erottamaan keskusdivari ja yksittäiset divarit toisistaan. Skeemat kuuluvat myyjille ja myyjät viittavat skeemoihin tietokantaa päivittäessä. Yksittäinen nide voidaan yksilöidä UUID:n avulla, jolloin se voidaan tallentaa turvallisesti eri skeemojen (tietokantojen) välillä.

# 8 Toteutusvälineet

Ohjelmointikielenä käytetään JavaScriptiä. Alla erittely käytetyistä työkaluista.

## 8.1 Palvelin

- **Suoritusympäristö:** Node.js
- **Verkkokehys:** Express.js
- **Tietokantamoduuli:** pg
- **Tietokanta:** PostgreSQL

## 8.2 Käyttöliittymä

- **Verkkokehys:** React
- **Tilanhallinta:** Redux

### 8.3 Kehitysympäristö

- **Käyttöliittymäkehys:** Vite
- **Ohjelmakoodin formatointityökalu:** Prettier
- **Ohjelmakoodin tarkistustyökalu:** ESLint
- **Versiohallinta:** Git