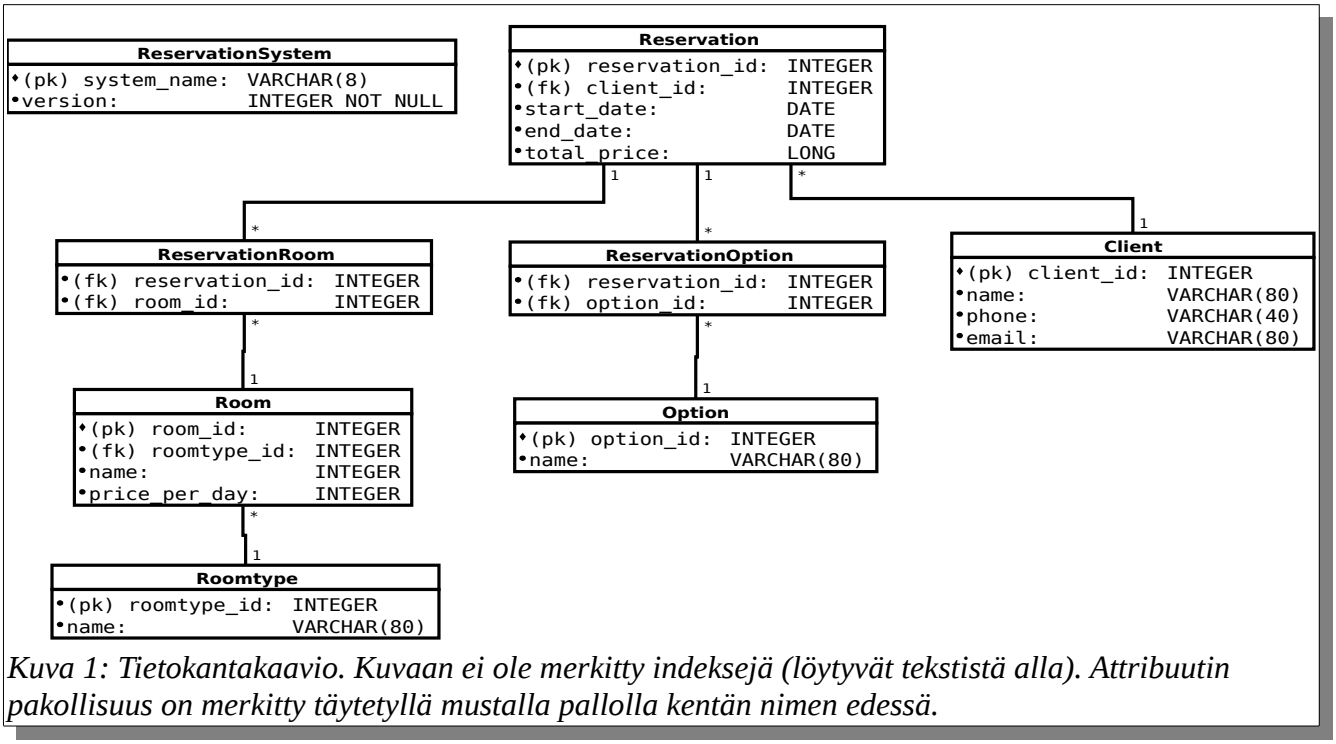
TIKAPE OHJELMOINTIPROJEKTI

HOTELLIHUONEIDEN VARAUSJÄRJESTELMÄ

Tietokanta



Kuva 1: Tietokantakaavio. Kuvaan ei ole merkitty indeksejä (löytyvät tekstistä alla). Attribuutin pakollisuus on merkitty täytetyllä mustalla pallolla kentän nimen edessä.

ReservationSystem -taulu on ohjelmaa itseään varten, se sisältää tiedon ohjelman eri komponenttien versionumeroista. Tällä hetkellä käytössä on yksi: nimeltään "database", ja sen versionumero on 1. Ohjelman käynnistyessä tarkistetaan tietokannan versionumero, mikäli se puuttuu tai on 0, tietokanta alustetaan.

Room -taulussa on kaikki hotellihuoneet. *Name* -kenttä on huonenumero. Kenttä *price_per_day* pitää sisällään huoneen päiväkohtaisen hinnan euroina. Huoneiden tyypit on listattu *Roomtype* -taulussa.

Asiakkaat on listattu *Client* -taulussa. Kahden asiakkaan katsotaan olevan eriä, jos mikä tahansa tiedoista (nimi, puhelinnumero, sähköposti) eroaa. Kyseiselle taululle on luotu uniikki indeksi näiden kenttien perusteella, jolla varmistetaan ettei käyttäjistä luoda duplikaatteja, ja optimoidaan käyttäjien hakua.

Reservation -taulussa on listattu kaikki varaukset. Kentässä *total_price* on laskettu kyseisen varauksen kokonaissumma, vaikka sen saisi selville myös hakemalla huonekohtaiset hinnat ja laskemalla. Tämä on tehty jotta ohjelmasta saatiin yksinkertaisempi, ja samalla myös hieman nopeampi. Samoista syistä kenttään *option_count* on lisävarusteiden lukumäärä.

Varaukseen liittyvät huoneet on listattu liitostaulussa *ReservationRoom*, ja varaukseen liittyvät lisävarusteet liitostaulussa *ReservationOption*.

Lisävarusteiden tiedot (nimi) löytyvät *Option* -taulusta. Kyseiselle taululle on luotu uniikki indeksi *name* -kentän perusteella, jolla varmistetaan ettei lisävarusteita luoda duplikaatteja, ja optimoidaan niiden hakua.

SQL-lauseet

Järjestelmän tietokantakomponentin versionumeron haku:

```
SELECT version FROM ReservationSystem WHERE system_name = 'database';
```

Tietokannan alustus:

```
BEGIN WORK;
DROP TABLE ReservationSystem IF EXISTS;
CREATE TABLE ReservationSystem ( system_name VARCHAR(8) PRIMARY KEY, version
INTEGER NOT NULL );
DROP TABLE Roomtype IF EXISTS;
CREATE TABLE Roomtype (
                  roomtype_id    INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
                  name           VARCHAR(80) NOT NULL
                );
DROP TABLE Room IF EXISTS;
CREATE TABLE Room (
                  room_id        INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
                  roomtype_id    INTEGER NOT NULL,
                  name           INTEGER NOT NULL,
                  price_per_day  INTEGER NOT NULL,
                  FOREIGN KEY(roomtype_id) REFERENCES Roomtype(roomtype_id)
                );
DROP TABLE Client IF EXISTS;
CREATE TABLE Client (
                  client_id      INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
                  name           VARCHAR(80) NOT NULL,
                  phone          VARCHAR(40) NOT NULL,
                  email          VARCHAR(80) NOT NULL
                );
CREATE UNIQUE INDEX idx_client_name_phone_email ON Client(name, phone, email);
DROP TABLE Reservation IF EXISTS;
CREATE TABLE Reservation (
                  reservation_id INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
                  client_id      INTEGER NOT NULL,
                  start_date     DATE NOT NULL,
                  end_date       DATE NOT NULL,
                  total_price    LONG NOT NULL,
                  option_count   INTEGER NOT NULL,
                  FOREIGN KEY (client_id) REFERENCES Client(client_id)
                );
DROP TABLE Option IF EXISTS;
CREATE TABLE Option (
                  option_id      INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
                  name           VARCHAR(80) NOT NULL UNIQUE
                );
CREATE UNIQUE INDEX idx_option_name ON Option(name);
DROP TABLE ReservationRoom IF EXISTS;
CREATE TABLE ReservationRoom (
                reservation_id INTEGER NOT NULL,
                room_id        INTEGER NOT NULL,
                FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id),
                FOREIGN KEY (room_id)        REFERENCES Room(room_id)
              );
DROP TABLE ReservationOption IF EXISTS;
CREATE TABLE ReservationOption (
```

```
              reservation_id INTEGER NOT NULL,
              option_id      INTEGER NOT NULL,
              FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id),
              FOREIGN KEY (option_id)      REFERENCES Option(option_id)
            );
INSERT INTO ReservationSystem ( system_name, version ) VALUES ( 'database', 1 );
COMMIT WORK;
```

Huonelistauksen haku:

Haetaan kaikki huoneet.

```
SELECT room_id, Roomtype.name AS type, Room.name AS name, price_per_day FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id;
```

Vapaiden huoneiden haku ilman parametreja:

Tällä kyselyllä haetaan vapaat huoneet ajanjaksolle [PVM1,PVM2].

```
SELECT room_id, Roomtype.name AS type, Room.name AS name, price_per_day FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
 WHERE Room.room_id NOT IN (
  SELECT DISTINCT ReservationRoom.room_id
    FROM ReservationRoom
    JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
   WHERE Reservation.start_date < 'PVM2'
     AND Reservation.end_date   > 'PVM1'
  )
 ORDER BY Room.price_per_day DESC, Room.name ASC;
```

Vapaiden huoneiden haku huonetyypin perusteella:

```
SELECT room_id, Roomtype.name AS type, Room.name AS name, price_per_day FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
 WHERE Room.room_id NOT IN (
  SELECT DISTINCT ReservationRoom.room_id FROM ReservationRoom
    JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
   WHERE Reservation.start_date < 'PVM2'
     AND Reservation.end_date   > 'PVM1'
  )
 AND Roomtype.name = 'TYYPPI'
 ORDER BY Room.price_per_day DESC, Room.name ASC;
```

Vapaiden huoneiden haku maksimihinnan perusteella:

```
SELECT room_id, Roomtype.name AS type, Room.name AS name, price_per_day FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
 WHERE Room.room_id NOT IN (
  SELECT DISTINCT ReservationRoom.room_id
    FROM ReservationRoom
    JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
   WHERE Reservation.start_date < 'PVM2'
     AND Reservation.end_date   > 'PVM1'
  )
 AND Room.price_per_day <= MAKSIMIHINTA
 ORDER BY Room.price_per_day DESC, Room.name ASC;
```

Vapaiden huoneiden haku huonetyypin sekä maksimihinnan perusteella:

```
SELECT room_id, Roomtype.name AS type, Room.name AS name, price_per_day
  FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
 WHERE Room.room_id NOT IN (
  SELECT DISTINCT ReservationRoom.room_id FROM ReservationRoom
    JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
   WHERE Reservation.start_date < 'PVM2'
     AND Reservation.end_date   > 'PVM1'
  )
 AND Roomtype.name = 'TYYPPI'
 AND Room.price_per_day <= MAKSIMIHINTA
 ORDER BY Room.price_per_day DESC, Room.name ASC;
```

Yhden huoneen varauksen tekeminen:

Tässä tapauksessa syntyy yksi uusi lisävaruste ja asiakas.

```
BEGIN WORK;
INSERT INTO Option ( name ) VALUES ( ? );
INSERT INTO Client ( name, phone, email ) VALUES ( ?, ?, ? );
INSERT INTO Reservation ( client_id, start_date, end_date, total_price,
option_count ) VALUES ( ?, ?, ?, ?, ? );
INSERT INTO ReservationRoom ( reservation_id, room_id ) VALUES ( ?, ? );
INSERT INTO ReservationOption ( reservation_id, option_id ) VALUES ( ?, ? );
COMMIT WORK;
```

Varauksien haku:

Haetaan kaikki varaukset.

```
SELECT *
  FROM Reservation
  JOIN Client          ON Client.client_id               = Reservation.client_id
  JOIN ReservationRoom ON ReservationRoom.reservation_id =
Reservation.reservation_id
  JOIN Room            ON Room.room_id                   = ReservationRoom.room_id
  JOIN Roomtype        ON Roomtype.roomtype_id           = Room.roomtype_id
 ORDER BY Reservation.start_date;
```

Suosituimpien lisävarusteiden haku:

Haetaan 10 suosituinta lisävarustetta.

```
SELECT Option.name, COUNT(ReservationOption.reservation_id) AS cnt
  FROM ReservationOption
  JOIN Option ON Option.option_id = ReservationOption.option_id
 GROUP BY ReservationOption.option_id
 ORDER BY cnt DESC
 LIMIT 10;
```

Parhaiden asiakkaiden haku:

Haetaan 10 parasta asiakasta.

```
SELECT Client.client_id, Client.name, Client.phone, Client.email,
       SUM(Reservation.total_price) AS value
  FROM Client
  JOIN Reservation ON Reservation.client_id = Client.client_id
 GROUP BY Client.client_id
 ORDER BY value DESC
 LIMIT 10;
```

Huonekohtaiset käyttöasteet:

Tällä kyselyllä haetaan huonekohtaiset varaukset rajattuna halutulle ajalle. Tätä listaa hyväksikäyttäen lasketaan huoneen varauspäivien lukumäärä halutun ajan sisällä. Huomaa, että laskutoimitukset tehdään Javalla.

```
SELECT ReservationRoom.room_id,
       CASE WHEN Reservation.start_date > 'PVM1'
            THEN Reservation.start_date
            ELSE 'PVM1'
       END AS start_date,
       CASE WHEN Reservation.end_date < 'PVM2'
            THEN Reservation.end_date
            ELSE 'PVM2'
       END AS end_date
  FROM ReservationRoom
  JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
 WHERE Reservation.start_date <= 'PVM2'
   AND Reservation.end_date   >= 'PVM1';
```

Huonetyyppikohtaiset käyttöasteet osa1:

Tällä kyselyllä lasketaan huoneiden lukumäärä per huonetyyppi.

```
SELECT Roomtype.name AS type, COUNT(room_id) AS cnt
  FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
 GROUP BY Room.roomtype_id;
```

Huonetyyppikohtaiset käyttöasteet osa2:

Tällä kyselyllä haetaan huonetyyppikohtaiset varaukset rajattuna halutulle ajalle. Tätä listaa hyväksikäyttäen lasketaan huonetyypin varauspäivien lukumäärä halutun ajan sisällä. Huomaa, että laskutoimitukset tehdään Javalla.

```
SELECT Roomtype.name AS type,
       CASE WHEN Reservation.start_date > 'PVM1'
            THEN Reservation.start_date
            ELSE 'PVM1'
       END AS start_date,
       CASE WHEN Reservation.end_date < 'PVM2'
            THEN Reservation.end_date
            ELSE 'PVM2'
       END AS end_date
```

```
  FROM Room
  JOIN Roomtype ON Roomtype.roomtype_id = Room.roomtype_id
  JOIN ReservationRoom ON ReservationRoom.room_id = Room.room_id
  JOIN Reservation ON Reservation.reservation_id = ReservationRoom.reservation_id
 WHERE Reservation.start_date <= 'PVM2'
   AND Reservation.end_date   >= 'PVM1';
```

Huoneen lisäys:

```
INSERT INTO Room ( roomtype_id, name, price_per_day ) VALUES ( ?, ?, ? );
```

Huonetyypin lisäys:

```
INSERT INTO Roomtype ( name ) VALUES ( ? );
```

Toteuttamatta jääneet toiminnallisuudet

Kaikki vaaditut toiminnallisuudet on toteutettu. Kaikkea ei kuitenkaan toteutettu SQL:llä, vaan osa toiminnallisuudesta, esimerkiksi prosenttilasku, on toteutettu sovelluksessa Javalla.

Ratkaisussa havaitut ongelmat

Asiakkaan yksilöinti kaikkien yhteystietojen perusteella ei ole kovin toimiva ratkaisu oikeassa käytössä. Kyseessä on myös käyttöliittymään liittyvä ongelma, koska varaustilanteessa sen hetkistä asiakasta etsittäisiin listalta ja lisättäisiin vain mikäli ei löytyisi. Tämä vaatii myös sen, että asiakkaan tietoja pitäisi pystyä muokkaamaan, ettei esimerkiksi 10 vuotta sitten hotellissa käyneen asiakkaan vanha, sittemmin vaihtunut, puhelinnumero jää yhteystiedoksi.

Huoneiden hinnat on euron tarkkuudella, todennäköisesti oikeassa järjestelmässä tarvitaan sentin tarkkuutta.

Yritys/yhdistys yms. asiakasryhmiä ei ole varsinaisesti otettu huomioon, kyseisillä ryhmillä saattaa olla useita eri varauksia tekeviä henkilöitä ja ne tulisi pystyä linkittämään yhteen tiettyyn asiakkaaseen.

Testaamisen helpottamiseksi varauksia voi luoda menneisyyteen, sitä ei pitäisi pystyä tekemään.

Tietokannassa ei tarkisteta onko varauksen päivämäärät oikein (alku ennen loppua).

Tietojen muokkausta ei vaadittu, mutta oikeassa järjestelmässä tietoja tulisi pystyä muokkaamaan. Esimerkiksi huoneita pitäisi pystyä ottamaan pois käytöstä, vaikkapa remontin tai siivouksen ajaksi. Tätä varten huonetietoihin tulisi lisätä esim. boolean -kenttä joka kertoisi, onko huone käytettävissä. Tämä ei yksistään riittäisi mikäli nämä käytöstäpoissaoloajat haluttaisiin ottaa huomioon raporteissa.

Huonenumerointia (nimeämistä) ei tarkisteta tai pakoteta syötteissä ja tulosteissa tehtäväksiannonmukaiseen malliin jossa on 3 numeroa (kerros ja huone).

Pienet kirjoitusvirheet ja isojen/pienten kirjainten eroavaisuudet syötteissä tulkitaan eriksi, jolloin saattaa syntyä useita eri instansseja samoista asioista, esimerkiksi lisävarusteet "Silitysrauta" ja "silitysrauta". Tästä on haittaa mm. raportoinnissa, mutta myös silloin kun lähdetään laajentamaan sovellusta ja lisävarusteet linkitetään oikeisiin fyysisiin esineisiin/palveluihin.