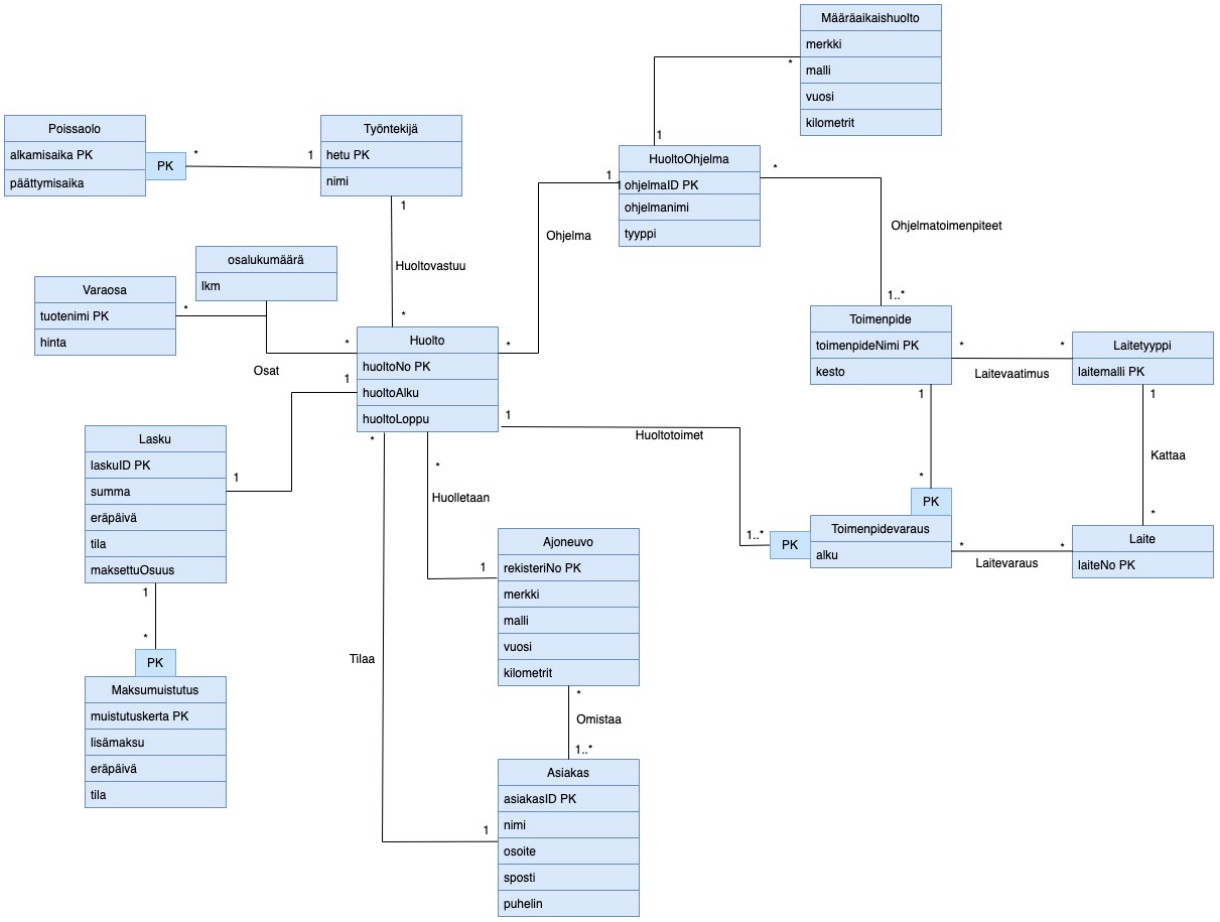
Autokorjaamon tietokanta

CS-A1150

6.5.2022

Pauli-Tapio Lahtinen
Mikael Rönnholm

UML-kaavio:



(kirjoitusvirhe toimenpidevarauksista huoltoon menevään assosiaatioon liittyen: yhdessä huollossa voi olla 1..* toimenpidevarausta *..1 sijaan)

UML-kaavion relaatiomalli ja funktionaaliset riippuvuudet:

Relaatio	Funktionaaliset riippuvuudet
Huolto(huoltoNo, huoltoAlku, huoltoLoppu, ohjelmaID, rekisteriNo, tilaajaID, työntekijäHetu)	huoltoNo → huoltoAlku huoltoLoppu ohjelmaID rekisteriNo tilaajaID työntekijäHetu rekisteriNo huoltoAlku → huoltoNo huoltoLoppu ohjelmaID tilaajaID työntekijäHetu
Lasku(laskuID, summa, eräpäivä, tila, maksettuOsuus, huoltoNo)	laskuID → summa eräpäivä tila maksettuOsuus huoltoNo huoltoNo → laskuID summa eräpäivä tila maksettuOsuus
Maksumuistutus(muistutuskerta, laskuID, lisämaksu, eräpäivä, tila)	muistutuskerta laskuID → lisämaksu, eräpäivä, tila
Varaosa(tuotenimi, hinta)	tuotenimi → hinta
Osat(tuotenimi, huoltoNo, lkm)	tuotekoodi huoltoNo → lkm
Ajoneuvo(rekisteriNo, merkki, malli, vuosi)	rekisteriNo → merkki malli vuosi
Asiakas(asiakasID, nimi, osoite, sposti, puhelin)	asiakasID → nimi osoite sposti puhelin
AutonOmistaja(asiakasID, rekisteriNo)	ei funktionaalisia riippuvuuksia
Työntekijä(hetu, nimi)	hetu → nimi
Poissaolo(alkamisaika, päättymisaika, hetu)	ei funktionaalisia riippuvuuksia
Määräaikaishuolto(merkki, malli, vuosi, kilometrit, ohjelmaID)	merkki malli vuosi kilometrit → ohjelmaID
Ohjelmatoimenpiteet(ohjelmaID, toimenpideNimi)	ei funktionaalisia riippuvuuksia
Toimenpide(toimenpideNimi, kesto)	toimenpideNimi → kesto
Toimenpidevaraus(toimenpideNimi, huoltoNo, alku)	toimenpideNimi huoltoNo → alku
Laitetyyppi(laitemalli)	ei funktionaalisia riippuvuuksia
Laite(laiteNo, laitemalli)	laiteNo → laitemalli
Laitevaraus(toimenpideNimi, laiteNo, huoltoNo)	ei funktionaalisia riippuvuuksia
Laitevaatimus(toimenpideNimi, laitemalli)	ei funktionaalisia riippuvuuksia
HuoltoOhjelmat(ohjelmaID, ohjelmanimi, tyyppi)	ohjelmaID → ohjelmanimi tyyppi ohjelmanimi → ohjelmaID tyyppi

Toteutus

Tietokannan UML-kaaviota luonnostellessa huomasimme, että mahdollisuuksia toteutuksille on erittäin monta. Panostimme selkeyteen ja yksinkertaisuuteen niin oman työskentelyn kuin valmiin tietokannan käsittelynkin helpottamiseksi. Käymme seuraavaksi läpi tietokannastamme löytyviä ongelmien ratkaisuja ja sitä, miten olemme niihin ratkaisuihin päätyneet.

Lähdimme tekemään toteutusta itse huoltotapahtumien ympärille, sillä ne ovat yrityksen toiminnankin kannalta keskiössä. Jokainen huolto yksilöidään omalla numerollaan. Lisäksi huollolle merkitään aloitus- ja lopetusaika, huollon tyyppi(määräaikaishuolto, korjaus, molemmat), huollossa olevan auton rekisterinumero, huollon tilanneen asiakkaan ID ja huollosta vastaavan työntekijän ID.

Lisäsimme alunperin huolto-luokan aliluokiksi korjauksen ja määräaikaishuollon, mutta tämä osoittautui ongelmalliseksi tilanteissa, joissa määräaikaishuollon yhteydessä tarvitaan muitakin korjaustoimenpiteitä. Huollosta meneekin nyt suora assosiaatio toimenpidevaraukseen, joka ottaa avainattribuutikseen toimenpiteen nimen. Huolto koostuu toimenpidevarauksista, joista jokainen liittyy omaan toimenpiteeseensä. Määräaikaishuoltoon liittyvät toimenpiteet pystytään tarkistamaan erikseen Määräaikaistoimenpiteet-assosiaatiosta tehdyn relaation avulla.

Esimerkki: Mitä toimenpiteitä kuuluu vm. 2008 Skoda Fabian 100tkm määräaikaishuoltoon ja mitä laitteita toimenpiteisiin vaaditaan?

```
SELECT toimenpideNimi, laitemalli
FROM Määräaikaistoimenpiteet NATURAL JOIN Laitetyyppi
WHERE merkki = Skoda AND
        malli = Fabia AND
        vuosi = 2008 AND
        kilometrit = 100000
```

Toimenpidevarauksiin pystytään liittämään laitteita, joiden tarpeellisuus selviää toimenpideluokasta Laitevaatimus-relaation avulla. Liittäminen tapahtuu Laitevaraus-relaatiolla, joka ottaa attribuuteikseen toimenpiteen nimen, laitenumeron ja huoltonumeron. Näistä kaikki ovat avainattribuutteja, jotta laitevaraus pystytään yksilöimään.

Laitevaraus-relaation luomisen jälkeen pystytään selvitämään, mikä laite on mihinkin aikaan sekä toimenpiteeseen ja huoltoon liittyen käytössä. Kun tietty laite on yhdistetty tiettyyn toimenpidevaraukseen, pystytään tietokannan käsittely ohjaamaan siten, ettei toimenpidevaraukseen liitettyä laitetta pysty varaamaan toiseen toimenpiteeseen, joka ajoittuu samalle ajalle.

Koska jokainen laite liittyy tasan yhteen laitetyyppiin, voidaan laitevarausten ja toimenpidevarausten avulla selvittää, mihin laitetyyppiin kuuluvat laitteet ovat milloinkin vapaina. Tämän tiedon pohjalta voidaan laitteita lisätä toimenpidevaraukseen. Päädyimme liittämään laitetyypin suoraan toimenpiteeseen ja laitteen toimenpiteen varaukseen, jotta tiettyyn toimenpiteeseen liittyvä laitetyyppi pystytään erottamaan varaukseen kuuluvasta laitteesta.

Huoltoon tarvittavia osia pystytään lisäämään Osat-assosiaatiosta tehdyn relaation avulla. Osat relaatio yhdistää huoltoID:n ja varaosan tuotekoodin, jonka avulla saadaan puolestaan selville tuotteen nimi ja hinta. Osat-relaatio pitää kirjaa myös käytettyjen varaosien lukumäärästä. Koska toimenpiteiden tarve uusille varaosille vaihtelee samojenkin toimenpiteiden kesken, koimme järkevämmäksi yhdistää käytetyt varaosat suoraan huoltoon, eikä toimenpiteeseen. Tosiasiassa huoltamolle olisi hyötyä myös osasta, jolla pystyttäisiin yhdistämään varaosat tiettyihin ajoneuvomalleihin ja mahdollisesti jopa toimenpiteisiin, mutta jätämme tämän toiminnon toteuttamatta tietokannan yksinkertaistamiseksi.

Esimerkki: Tiedot huoltoon 234567 liittyvistä kustannuksiin liittyvistä tekijöistä voidaan hakea esimerkiksi seuraavalla tavalla:

```
SELECT tuotekoodi, nimi, hinta, lkm, huoltoAlku, huoltoLoppu
FROM Varaosa NATURAL JOIN Osat NATURAL JOIN Huolto
WHERE huoltoNo = 234567
```

Lasku huollosta luodaan erikseen, jotta sen summa saadaan vastaamaan todellisia huollon kustannuksia. Lasku-luokkaan merkitään attribuuteiksi huolto, johon lasku liittyy, laskun summa, eräpäivä, tila(lähetetty/maksettu) ja maksettu osuus, jolla pystytään seuraamaan asiakkaan velkaa. Mietimme myös vaihtoehtoa, jossa laskun tilaa oltaisiin seurattu omalla luokallaan, mutta päädyimme kirjaamaan tilan suoraan laskuun, jossa sitä voidaan maksutapahtuman edetessä muokata esimerkiksi seuraavasti:

Esimerkki: Muutetaan laskun 123456 tila maksetuksi ja maksettu summa koko laskun summaksi(300€):

```
UPDATE Lasku
SET tila = “maksettu”, maksettuOsuus = 300,00
WHERE laskuID = 123456
```

Jos lasku on maksamatta eräpäivän mentyä, voidaan siitä lähettää maksumuistutus. Maksumuistutus ottaa avainattribuuteikseen laskuID:n ja muistutuskerran. Tällöin sama luku muistutuskerroille voi toistua useaan otteeseen, mutta tietty muistutus pystytään silti yhdistämään oikeaan laskuun. Lisäksi muistutukseen kirjataan erikseen lisämaksun määrä, muistutuksen eräpäivä ja tila.

Työntekijöiden saatavuutta seuraamme omalla luokalla, jolla pidetään kirjaa työntekijöiden poissaoloista. Poissaolo-luokka ottaa avainattribuutikseen työntekijän sosiaaliturvatunnuksen ja poissaolon alkamisajan. Poissaolon päättyminen saadaan tietää luokkaan merkittävän kesto-attribuutin avulla. Työntekijöiden saatavuutta voitaisiin seurata myös päinvastaisella luokalla, joka seuraisi työntekijöiden paikallaoloa, mutta oletimme kyseessä olevan korjaamo, jonka henkilökunta on vakituista ja pääsääntöisesti täyttä päivää tekevää. Tällöin on kevyempää seurata pelkkiä poissaoloja. Vapaita työntekijöitä halutulle ajalle voidaan etsiä esimerkiksi näin:

Esimerkki: Ketkä ovat vapaana aikavälillä 16.5.2022 klo 13:00 – 16.5.2022 klo 16:00?

```
SELECT hetu, nimi
FROM Työntekijät
WHERE hetu NOT IN
        (SELECT hetu
        FROM Poissaolo NATURAL JOIN Huolto
        WHERE
                (päättymisaika => 2022-05-16 13:00:00 AND
                alkamisaika =< 2022-05-16 16:00:00) OR
                (huoltoLoppu => 2022-05-16 13:00:00 AND
                huoltoAlku =< 2022-05-16 16:00:00)
        )
```

Asiakas- ja ajoneuvotiedot, sekä niihin liittyvät omistussuhteet ratkaisimme kahdella luokalla, sekä ne yhdistävästä omistus-assosiaatiosta tehdyllä relaatiolla. Jokaiseen huoltotapahtumaan merkitään tilaaja, eli asiakas. Huollon näkökulmasta asiakas ei liity ajoneuvoon muutoin kuin luodun monikon tilaaja-attribuutin kohdalla. Ajoneuvon omistussuhteet kirjataan Omistaa-relaation avulla. Tämä relaatio ottaa attribuuteikseen ajoneuvon rekisteritunnuksen ja asiakasID:n, eli jokaiselle ajoneuvolle luodaan niin monta monikkoa, kuin sillä on omistajia.

Asiakas-luokka pitää nimestään huolimatta kirjaa myös niistä henkilöistä, jotka eivät ole suorassa asiakassuhteessa korjaamon kanssa, mutta liittyvät tämän toimintaan huollettavien ajoneuvojen omistussuhteiden kautta. Asiakkaille kirjataan tietokantaan attribuuteiksi yksilöllinen asiakasID, nimi, osoite, sähköposti ja puhelinnumero.

BCNF-muoto, anomaliat

Tässä osiossa käydään läpi relaatioiden funktionaaliset riippuvuudet, sulkeumat ja BCNF-muotoisuus:

Relaatio on BCNF-muodossa jos sen riippuvuuksien vasemman puolen atribuuttien sulkeumat kattavat kaikki relaation attribuutit. Alla olevan listan perusteella nähdään, että kaikki relaatiot ovat jo BCNF-muodossa.

Relaatio	Funktionaaliset riippuvuudet ja sulkeumat
Huolto(huoltoNo, huoltoAlku, huoltoLoppu, ohjelmaID, rekisteriNo, tilaajaID, työntekijäHetu)	huoltoNo → huoltoAlku huoltoLoppu ohjelmaID rekisteriNo tilaajaID työntekijäHetu rekisteriNo huoltoAlku → huoltoNo huoltoLoppu ohjelmaID tilaajaID työntekijäHetu huoltoNo: huoltoNo huoltoAlku huoltoLoppu ohjelmaID rekisteriNo tilaajaID työntekijäHetu rekisteriNo huoltoAlku: rekisterNo huoltoAlku huoltoNo huoltoLoppu ohjelmaID tilaajaID työntekijäHetu
Lasku(laskuID, summa, eräpäivä, tila, maksettuOsuus, huoltoNo)	laskuID → summa eräpäivä tila maksettuOsuus huoltoNo huoltoNo → laskuID summa eräpäivä tila maksettuOsuus laskuID: laskuID summa eräpäivä tila maksettuOsuus huoltoNo

	huoltoNo: huoltoNo laskuID summa eräpäivä tila maksettuOsuus
Maksumuistutus(muistutuskerta, laskuID, lisämaksu, eräpäivä, tila)	muistutuskerta laskuID → lisämaksu, eräpäivä, tila muistutuskerta laskuID: muistutuskerta laskuID lisämaksu eräpäivä tila
Varaosa(tuotenimi, hinta)	tuotenimi → hinta tuotenimi: tuotenimi hinta
Osat(tuotenimi, huoltoNo, lkm)	tuotekoodi huoltoNo → lkm tuotekoodi huoltoNo: tuotekoodi huoltoNo lkm
Ajoneuvo(rekisteriNo, merkki, malli, vuosi)	rekisteriNo → merkki malli vuosi rekisteriNo: rekisteriNo merkki malli vuosi
Asiakas(asiakasID, nimi, osoite, sposti, puhelin)	asiakasID → nimi osoite sposti puhelin asiakasID: asiakasID nimi osoite puhelin
AutonOmistaja(asiakasID, rekisteriNo)	ei funktionaalisia riippuvuuksia
Työntekijä(hetu, nimi)	hetu → nimi hetu: hetu nimi
Poissaolo(alkamisaika, päättymisaika, hetu)	ei funktionaalisia riippuvuuksia
Määräaikaishuolto(merkki, malli, vuosi, kilometrit, ohjelmaID)	merkki malli vuosi kilometrit → ohjelmaID merkki malli vuosi kilometrit: ohjelmaID merkki malli vuosi kilometrit
Ohjelmatoimenpiteet(ohjelmaID, toimenpideNimi)	ei funktionaalisia riippuvuuksia
Toimenpide(toimenpideNimi, kesto)	toimenpideNimi → kesto toimenpideNimi: toimenpideNimi kesto
Toimenpidevaraus(toimenpideNimi, huoltoNo, alku)	toimenpideNimi huoltoNo → alku toimenpideNimi huoltoNo: toimenpideNimi huoltoNo alku
Laitetyyppi(laitemalli)	ei funktionaalisia riippuvuuksia
Laite(laiteNo, laitemalli)	laiteNo → laitemalli laiteNo: laiteNo laitemalli
Laitevaraus(toimenpideNimi, laiteNo, huoltoNo)	ei funktionaalisia riippuvuuksia
Laitevaatimus(toimenpideNimi, laitemalli)	ei funktionaalisia riippuvuuksia
HuoltoOhjelmat(ohjelmaID, ohjelmanimi, tyyppi)	ohjelmaID → ohjelmanimi tyyppi ohjelmanimi → ohjelmaID tyyppi ohjelmaID: ohjelmaID ohjelmanimi tyyppi ohjelmanimi: ohjelmaID ohjelmanimi tyyppi

Tietokannassa voi olla sen toimintaan vaikuttavia anomalioita BCNF muotoisuudesta huolimatta. Kolme keskeisintä anomaliatyyppiä ovat tiedon toisteisuus, päivitysanomaliat ja poistoanomaliat.

Tiedon toisteisuus tarkoittaa, että sama tieto on esitetty useamman kerran, kun sen voisi tilan säästämiseksi ja luotettavuuden parantamiseksi säilöä yhteen paikkaan. Tietokannassamme ei ole toisteisuutta, mutta sitä voisi syntyä jos esimerkiksi laskun tietoihin tallennettaisiin omistajan puhelinnumero.

Päivitysanomaliat puolestaan seuraavat tiedon toisteisuudesta. Päivittäessä tietokannan tietoja joudutaan sama tieto muuttamaan useammasta relaatiosta tai monikosta. Jos esimerkiksi autoon olisi kiinnitetty omistajan kotiosoite, joutuisi kaikkien yhden omistajan autojen osoitteen vaihtamaan, kun omistajan osoite vaihtuu. Tietokantamme relaatioissa ei ole toisista luokista sellaisia atribuutteja, jotka eivät ole sen luokan avainatribuutteja. Näin vältetään ongelmia päivitettäessä.

Poistoanomaliat taas johtuvat tietyn tiedon pysyvään tallentamiseen osoitetun luokan puuttumisesta. Tällöin tieto saattaa kadota, kun viimeinen monikko, johon se on liitetty poistetaan. Jos esimerkiksi tieto toimenpiteen kestosta sisällytettäisiin vain toimenpidevaraukseen, katoaisi tämä tieto jos kaikki tietyn tyyppiset varaukset jouduttaisiin perumaan. Tietokantamme toimii monipuolisten riippuvuuksien avulla ja tiedot haetaan tarvittaessa niiden pysyvään säilytykseen osoitetuista luokista, jolloin poistoanomalioita ei synny.

Osa 2

Muutokset ensimmäiseen palautukseen:

Ensimmäiseen palautukseen verrattuna muokkasimme hieman Huolto ja Määräaikaishuolto-luokkia sekä lisäsimme HuoltoOhjelmat luokan. Ensimmäisessä palautuksessa vain määräaikaishuoltoihin oli tallennettavissa erilaisia ohjelmia, vaikka niitä tuli olla tallennettavissa myös korjauksiin. Lisäsimme HuoltoOhjelmat-luokan, johon voidaan nyt tallentaa kaikenlaisille huolto-ohjelmille toimenpiteitä Toimenpide-luokasta. Ohjelmat identifioidaan ohjelmaID:llä. Vaihdoimme Huolto-luokkaan attribuutin “tyyppi” tilalle ohjelmaID:n, joka kertoo, mihin ohjelmaan viitataan. Lisäksi Määräaikaishuolto-luokka ottaa nyt yhdeksi attribuutikseen ohjelmaID:n, jolloin määräaikaishuollot voidaan auton mallin, merkin, vuosimallin ja kilometriluvun avulla yhdistää tiettyyn huolto-ohjelmaan.

Taulujen luontikomennot ja tietotyyppien perustelut :

Tietokannan SQL-toteutuksessa olemme luottaneet aiemmin luotuihin ja myöhemmin muokattuihin UML-kaavioon ja relaatiomalleihin. Olemme kirjoittaneet taulut auki seuraavasti:

```
CREATE TABLE Asiakas(
  asiakasID CHAR(20) PRIMARY KEY,
  nimi VARCHAR(100) NOT NULL,
  osoite VARCHAR(100),
  sposti CHAR(50),
  puhelin CHAR(20)
);
```

AsiakasID on tiedostotyyppiä CHAR(20), sillä tällöin siihen voidaan halutessa tallentaa henkilötunnus. Nimien pituudet vaihtelevat suuresti, joten VARCHAR(100). Samoin osoitteet. Sähköpostit hieman vakimittaisempia ja puhelinnumeroihin jätetty varaa pidemmille ulkomaisille numeroille.

```
CREATE TABLE Ajoneuvo(
  rekisteriNo CHAR(10) PRIMARY KEY,
  merkki VARCHAR(100),
  malli VARCHAR(100),
  vuosi INT
);
```

Rekisterinumerot lyhyitä, joten CHAR(10) riittää. Merkit ja mallit yleensä lyhyitä, mutta joskus on poikkeuksia, joten VARCHAR(100). Vuodelle INT tarkkuus hyvä.

```
CREATE TABLE Työntekijä(
  hetu CHAR(20) PRIMARY KEY,
  nimi VARCHAR(100)
);
```

Hetussa ekstratilaa erikoisuuksia varten, ei kuitenkaan ole tarvetta VARCHARille näin vähillä merkkimäärillä. Nimet voivat puolestaan olla hyvinkin erimittaisia.

```
CREATE TABLE Huolto(
  huoltoNo INT PRIMARY KEY,
  huoltoAlku CHAR(20),
  huoltoLoppu CHAR(20) CHECK (huoltoAlku <= huoltoLoppu OR huoltoLoppu=NULL),
  ohjelmaID CHAR(10) CHAR(10) REFERENCES HuoltoOhjelmat(ohjelmaID),
  rekisteriNo CHAR(15) REFERENCES Ajoneuvo(rekisteriNo),
  tilaajaID CHAR(20) REFERENCES Asiakas(asiakasID),
  työntekijäHetu CHAR(20) REFERENCES Työntekijä(hetu)
);
```

Huoltonumeroissa ajateltu, että ensimmäinen huolto on huoltoNo 1, ja niin edelleen, siksi INT. Päivämääriin ja aikaan kuluu 19 merkkiä, joten CHAR(20) on sopiva. OhjelmaID:n, rekisteriNo:n, tilaajaID:n ja työntekijäHetu:n perustelut löytyvät omista luokistaan.

```
CREATE TABLE HuoltoOhjelmat(
  ohjelmaID CHAR(10) PRIMARY KEY,
  ohjelmanimi CHAR(100) UNIQUE,
  tyyppi CHAR(17) CHECK(tyyppi IN ('korjaus', 'määräaikaishuolto'))
);
```

ohjelmaID on tietotyyppiä CHAR(10), jotta siihen voidaan halutessa tallentaa esimerkiksi valmistajat erottelevia kirjain- tai numeroyhdistelmiä. Ohjelmanimi on puolestaan CHAR(100), jotta kaikki tarpeellinen informaatio mahtuu. Ohjelmat ovat nimeltään uniikkeja asentajien työn helpottamiseksi. Tyyppiin on mahdollista tallentaa vain se, onko huolto-ohjelma korjaus vai määräaikaishuolto

```
CREATE TABLE Lasku(
 laskuID CHAR(10) PRIMARY KEY,
 summa REAL,
 eräpäivä CHAR(10),
 tila CHAR(10) DEFAULT ‘maksamatta’ CHECK(tila IN ('maksettu', 'maksamatta')),
 maksettuOsuus REAL DEFAULT 0 CHECK((maksettuOsuus < summa AND
tila='maksamatta') OR (maksettuOsuus = summa AND tila='maksettu')),
 huoltoNo INT REFERENCES Huolto(huoltoNo)
);
```

Laskun laskuID on tyyppiä CHAR(10), sillä ID voi olla jokin yhdistelmä kirjaimia ja numeroita. Summa, samoin kuin siitä maksettu osuus ilmaistaan euroissa ja senteissä, joten ne ovat reaalilukuja. Eräpäivä vie tasan 10 merkkiä, samoin tila jos se on maksamatta. huoltoNo:t on tarkoitus merkitä kasvavassa järjestyksessä, joten tähän INT on sopiva tietotyyppi.

```
CREATE TABLE Maksumuistutus (
  muistutuskerta INT,
  laskuID CHAR(10) REFERENCES Lasku(laskuID),
  lisämaksu REAL DEFAULT 5.00,
  eräpäivä CHAR(10),
  tila CHAR(10) CHECK(tila IN ('maksettu', 'maksamatta')),
  PRIMARY KEY (muistutuskerta, laskuID)
);
```

Muistutuskerta on yksiselitteinen. LaskuID puolestaan perustuu Laskuun ja on samaa tietotyyppiä. Lisämaksu ilmaistaan euroissa ja senteissä, joten se on reaaliluku. Eräpäivä ja tila voidaan perustella samoin kuin laskun vastaavat attribuutit

```
CREATE TABLE Varaosa (
  tuotenimi VARCHAR(100) PRIMARY KEY,
  hinta REAL
);
```

Tuotteen nimi voi vaihdella suuresti, joten sitä on hyödyllistä merkitä tietotyypillä VARCHAR. Hinta voidaan perustella samoin kuin laskun kohdalla.

```
CREATE TABLE Osat (
  tuotenimi CHAR(100) REFERENCES Varaosa(tuotenimi),
  huoltoNo INT REFERENCES Huolto(huoltoNo),
  lukumäärä INT,
  PRIMARY KEY (tuotenimi, huoltoNo)
);
```

Tuotenimi voidaan perustella samalla tavalla kuin aiemmin, samoin huoltoNo. Lukumäärä puolestaan ei vaadi perusteluja.

```
CREATE TABLE AutonOmistaja (
  asiakasID CHAR(20) REFERENCES Asiakas(asiakasID),
  rekisteriNo CHAR(10) REFERENCES Ajoneuvo(rekisteriNo),
  PRIMARY KEY (asiakasID, rekisteriNo)
);
```

Viite-avaimien tietotyyppien perustelut löytyvät niiden alkuperäisistä luokista

```
CREATE TABLE Poissaolo (
  alkamisaika CHAR(20),
  päättymisaika CHAR(20),
  hetu CHAR(20) REFERENCES Työntekijä(hetu),
  PRIMARY KEY (alkamisaika, hetu)
);
```

Alkamisaika ja päättymisaika vievät molemmat 19 merkkiä, joten CHAR(20) on osuva ja selkeä tietotyyppi. Hetu on perusteltu Työntekijä-luokassa.

```
CREATE TABLE Määräaikaishuolto (
  merkki VARCHAR(100),
  malli VARCHAR(100),
  vuosi INT,
  kilometrit INT,
  ohjelmaID CHAR(10) REFERENCES HuoltoOhjelmat(ohjelmaID),
  PRIMARY KEY (merkki, malli, vuosi, kilometrit)
);
```

Auton merkki ja malli voivat vaihdella suuresti, joten VARCHAR on sopiva. Vuosi ja kilometrit taas voidaan merkitä kokonaisluvun tarkkuudella. Viite-avaimien tietotyyppien perustelut löytyvät niiden alkuperäisistä luokista

```
CREATE TABLE Ohjelmatoimenpiteet (
  ohjelmaID CHAR(10) REFERENCES HuoltoOhjelmat(ohjelmaID),
  toimenpideNimi CHAR(100) REFERENCES Toimenpide(toimenpideNimi)
);
```

Viite-avaimien tietotyyppien perustelut löytyvät niiden alkuperäisistä luokista

```
CREATE TABLE Toimenpide (
  toimenpideNimi CHAR(100) PRIMARY KEY,
  kesto INT NOT NULL
);
```

Kestoon riittää tietotyypiksi INT, joka mittaa kestoa minuutteina. Sekuntin tarkkuuteen ei korjaamon työskentelyssä tarvitse päästä.

```
CREATE TABLE Toimenpidevaraus (
  toimenpideNimi CHAR(100) REFERENCES Toimenpide(toimenpideNimi),
  huoltoNo INT REFERENCES Huolto(huoltoNo),
  alku CHAR(20) NOT NULL,
  PRIMARY KEY (toimenpideNimi, huoltoNo)
);
```

Perustelut löytyvät attribuuttien omista luokista.

```
CREATE TABLE Laitetyyppi(
  laitemalli CHAR(100) PRIMARY KEY
);
```

laitemallien nimien pituudet voivat usein olla pitkiä ja spesifejä, siksi CHAR(100)

```
CREATE TABLE Laite(
  laiteNo INT PRIMARY KEY,
  laitemalli CHAR(100) REFERENCES Laitetyyppi(laitemalli)
);
```

laitteiden erittelyyn ajattelimme riittävän tieto, jonka avulla pystytään erottamaan korjaamon laitteet toisistaan, siksi INT.

```
CREATE TABLE Laitevaraus(
  toimenpideNimi CHAR(100) REFERENCES Toimenpide(toimenpideNimi),
  laiteNo INT REFERENCES Laite(laiteNo),
  huoltoNo INT REFERENCES Huolto(huoltoNo),
  PRIMARY KEY (toimenpideNimi, laiteNo, huoltoNo)
);
```
Perustelut tietotyypeille attribuuttien omissa luokissa.

```
CREATE TABLE Laitevaatimus (
  toimenpideNimi CHAR(100) REFERENCES Toimenpide(toimenpideNimi),
  laitemalli CHAR(100) REFERENCES Laitetyyppi(laitemalli),
  PRIMARY KEY (toimenpideNimi, laitemalli)
);
```
Perustelut tietotyypeille attribuuttien omissa luokissa.

Hakemistot ja näkymät

Hakemistoja luotaessa on tärkeää miettiä, mitä hakuja tietokantaan tehdään useasti ja mitkä luokat ovat sellaisia, joihin tehtäessä hakuja tulee käydyksi läpi useita levysivuja. Yleisimmät haut liittyvät todennäköisesti huoltojen aikataulutukseen ja laskutukseen, joten tarkastellaan näihin tehtäviä hakemistoja. Toimenpide ja laitevarauksia sekä käytettyjä varaosia tulee luotua kaikista eniten (vuosien aikana varmasti tuhansia) , sillä niitä liittyy todennäköisesti yhteen huoltoon aina useampia, lisäksi samaan huoltoon liittyvät varaukset on luotu samoihin aikoihin, jolloin ne todennäköisemmin sijaitsevat samalla levysivulla. Tehdään siis huoltonumeron perusteella hakemistot sekä toimenpide- että laitevaraukseen.

```
CREATE INDEX ToimvarausIndex ON Toimenpidevaraus(huoltoNo);
CREATE INDEX LaitevarausIndex ON Laitevaraus(huoltoNo);
CREATE INDEX OsaIndex ON Osat(huoltoNo);
```

Näkymistä on hyötyä kun halutaan esimerkiksi yhdistellä samankaltaista tietoa useista luokista käsiteltäväksi. Tällaisia voisivat olla esimerkiksi tietyn tilaajan maksumuistutusten ja laskujen maksamattoman summan koostava näkymä. Luodaan kyseinen näkymä:

```
CREATE VIEW Kokonaislasku AS
SELECT tilaajaID, SUM(yhteissumma)
FROM Huolto NATURAL JOIN (
  SELECT COALESCE(sakko, 0) + summa-maksettuOsuus AS yhteissumma, huoltoNo
  FROM Lasku AS L LEFT OUTER JOIN (SELECT SUM(lisämaksu) AS sakko, laskuID
FROM Maksumuistutus GROUP BY laskuID) AS M ON L.laskuID=M.laskuID
  WHERE yhteissumma > 0
)
GROUP BY tilaajaID ;
```

Toinen hyödyllinen näkymä voisi olla kalenteri, joka pitää kirjaa työntekijöiden poissaoloista ja huolloista. Näkymän avulla voidaan helpommin hakea vapaana olevia työntekijöitä tiettynä ajankohtana.

CREATE VIEW Kalenteri AS
SELECT alkamisaika AS alku, loppumisaika AS loppu, hetu
FROM Poissaolo
UNION
SELECT huoltoAlku AS alku, huoltoLoppu AS loppu, työntekijäHetu AS hetu;

Kyselyitä:

Asiakkaan Nissan Qashqaille AOF-682 on suoritettu huolto ja halutaan selvittää, kuinka paljon huollon loppusummaksi tuli. On selvitettävä, mitä osia käytettiin eli ennen kaikkea, mikä käytettyjen osien hinta oli, ja kuinka paljon aikaa huollon toimenpiteisiin kului. Huoltojen numerointi toimii siten, että mitä vanhempi huoltotapahtuma, sitä pienempi huoltonumero. Haetaan tietokannasta viimeisimmän ajoneuvolle tehdyn huollon ja toimenpiteiden tiedot. Haut, jotka tulee tehdä ovat:

```
SELECT huoltoNo, rekisteriNo, toimenpideNimi, kesto
FROM Huolto NATURAL JOIN Toimenpidevaraus NATURAL JOIN Toimenpide
WHERE huoltoNo IN
  (SELECT max(huoltoNo)
  FROM Huolto
  WHERE rekisteriNo = 'AOF-682'
  );
```

	huoltoNo	rekisteriNo	toimenpideNimi	kesto
1	3	AOF-682	renkaanvaihto	10
2	3	AOF-682	tuulilasin vaihto	30
3	3	AOF-682	öljyjen vaihto	10

Tuloksesta saadaan huoltonumero, jonka perusteella voidaan vielä etsiä käytetyt osat:

```
SELECT huoltoNo, tuoteNimi, lukumäärä, hinta
FROM Osat NATURAL JOIN Varaosa
WHERE huoltoNo = 3;
```

	huoltoNo	tuoteNimi	lukumäärä	hinta
1	3	öljy	1	10
2	3	tuulilasi	1	200
3	3	talvirengas	4	40

Lukumäärien, hintojen ja toimenpiteiden kestojen perusteella huollolle saadaan hinta. Toimenpiteiden kestoista voitaisiin myös laskea pelkkä summa, mutta läpinäkyvyyden vuoksi haussamme on eritelty myös se, mitä on tehty. Käyttöliittymä tekisi molemmat haut rekisterinumeron perusteella automaattisesti.

Nissan AOF-682 huollosta on lähetetty lasku ja nyt halutaan selvittää, onko lasku maksettu ja keneen tulee ottaa yhteys laskusta:

```
SELECT LaskuID, summa, tila, eräpäivä, maksettuOsuus, nimi, puhelin, sposti
FROM Huolto NATURAL JOIN Lasku, Asiakas
WHERE rekisteriNo = 'AOF-682' AND tilaajaID = asiakasID;
```

	LaskuID	summa	tila	eräpäivä	maksettu	nimi	puhelin	sposti
1	20003	260	maksamatta	2022-05-25	0	James Bond	+44 05303401	secret.agent@MI6.uk

Korjaamon johtaja miettii, mikä on huoltamon suurempien laitteiden käyttöaste 11.5.2022 klo 9-11, sillä siihen olisi tulossa korkean profiilin asiakas, jolle kelpaa vain paras laatu. Hän on erityisen kiinnostunut siitä, mitkä laitteet ovat vapaana, jotta voi päättää, pitääkö muita huoltoja lykätä laitepulan vuoksi. Haku:

```
SELECT laitemalli, laiteNo
FROM Laite
WHERE laiteNo NOT IN
  (SELECT laiteNo
  FROM Laite NATURAL JOIN Laitevaraus, Toimenpidevaraus NATURAL JOIN Toimenpide
  WHERE Toimenpidevaraus.huoltoNo = Laitevaraus.huoltoNo AND
    (alku BETWEEN '2022-05-11 09:00:00' AND '2022-05-11 11:00:00') OR
    (datetime(julianday(alku) +kesto/(60.0*24)) BETWEEN '2022-5-11 09:00:00' AND
'2022-5-11 11:00:00') OR
    ('2022-5-11 09:00:00' BETWEEN alku AND datetime(julianday(alku) + kesto/(60.0*24)))
OR
    ('2022-5-11 11:00:00' BETWEEN alku AND datetime(julianday(alku) + kesto/(60.0*24)))
  );
```

	laitemalli	laiteNo
1	pulttipyssy	1
2	pulttipyssy	2
3	pulttipyssy	3
4	hitsauskone	5
5	pilarinosturi	7
6	tunkki	8
7	tunkki	9
8	tunkki	10
9	kompressori	13
10	kompressori	14
11	rengaskone	15

Haluttaisiin luoda huolto kesäkuun ensimmäiselle päivälle klo 12-13, mutta ei tiedetä, kuka työntekijöistä on vapaana kyseisenä päivänä. Luodaan kysely, joka palauttaa työntekijät, jotka eivät ole lomalla tai varattuna muihin huoltoihin kyseisenä aikana.

```
SELECT nimi
FROM Työntekijä
WHERE nimi NOT IN
  (SELECT nimi
  FROM Työntekijä as T, Huolto as H
  WHERE T.hetu=H.työntekijäHetu AND
     (
     (huoltoAlku BETWEEN '2022-06-01 12:00:00' AND '2022-06-01 13:00:00')
     OR
     (huoltoLoppu BETWEEN '2022-06-01 12:00:00' AND '2022-06-01 13:00:00')
     OR
     ('2022-06-01 12:00:00' BETWEEN huoltoAlku AND huoltoLoppu)
     OR
     ('2022-06-01 13:00:00' BETWEEN huoltoAlku AND huoltoLoppu)
  ))
AND nimi NOT IN
  (SELECT nimi
  FROM Työntekijä NATURAL JOIN Poissaolo
  WHERE '2022-06-01' BETWEEN alkamisaika AND päättymisaika);
```

	nimi
1	Veera Immonen
2	Tommi Työläs

Asiakas on tuonut autonsa renkaanvaihtoon, yksi vaadittu vaihe toimenpiteen suorittamiseksi on selvittää mitä laitteita kyseisen toimenpiteen suorittamiseen vaaditaan:

```
SELECT laitemalli
FROM Laitevaatimus
WHERE toimenpideNimi='renkaanvaihto';
```

	laitemalli
1	kompressori
2	tunkki

Toimitusjohtaja valmistelee neljännesvuosikatsausta ja tahtoo selvittää, montako korjausta ja yksittäistä toimenpidettä korjaamolla on tehty 2. neljänneksellä. Lisäksi toimitusjohtaja tahtoo tietää mikä on yleisimmin huollettu automerkki / automerkit, sillä hänen opiskelija-tyttärensä tekee yliopistolla tutkimusta siitä, miten automerkit korreloivat onnettomuuksia.

Huoltojen määrä:

```
SELECT COUNT(huoltoNo)
FROM Huolto
WHERE huoltoLoppu BETWEEN '2022-04-01 00:00:00' AND  '2022-06-30 23:59:00';
```

	COUNT(huoltoNo)
1	5

Toimenpiteiden määrä:

```
SELECT COUNT(*)
FROM Huolto NATURAL JOIN Toimenpidevaraus
WHERE huoltoLoppu BETWEEN '2022-04-01 00:00:00' AND  '2022-06-30 23:59:00';
```

	COUNT(*)
1	10

Eniten huollettu automerkki/merkit:

```
SELECT merkki, COUNT(merkki) as total
FROM Ajoneuvo NATURAL JOIN Huolto
GROUP BY merkki
HAVING total = (
  SELECT max(amount)
  FROM (
  SELECT merkki, COUNT(merkki) as amount
  FROM Ajoneuvo NATURAL JOIN Huolto
  GROUP BY merkki )
  );
```

	merkki	total
1	Lamborghini	3

Korjaamon tilit ovat pahasti miinuksella ja yt-neuvottelut ovat käynnistymässä, jos jostain ei saada lisää tuloja. Eräs työntekijä keksii tarkastaa ne asiakkaat, joilla on maksamattomia laskuja tai maksumuistutuksia ja listaa myös jokaiselle maksamattomien laskujen ja muistutusten summan. Sen jälkeen tarkistetaan myös huollot joista ei ole lähetetty/luotu vielä laskua ollenkaan.

Asiakkaat joilla maksamattomia laskuja tai muistutuksia ja maksamattomat summat:

```
SELECT tilaajaID, SUM(yhteissumma)
FROM Huolto NATURAL JOIN (
  SELECT COALESCE(sakko, 0) + summa-maksettuOsuus AS yhteissumma, huoltoNo
  FROM Lasku AS L LEFT OUTER JOIN (SELECT SUM(lisämaksu) AS sakko, laskuID
FROM Maksumuistutus GROUP BY laskuID) AS M ON L.laskuID=M.laskuID
  WHERE yhteissumma > 0
)
GROUP BY tilaajaID ;
```

	tilaajaID	SUM(yhteissumma)
1	7	370

!!! COALESCE funktio muuttaa sakkojen yhteissumman nollaksi tapauksessa, jossa sen arvo on NULL eli laskuun ei liity maksumuistutuksia !!!

Huollot, joista ei ole lähtenyt vielä laskua:

```
SELECT huoltoNo
FROM Huolto
WHERE huoltoNo NOT IN (
  SELECT huoltoNo
  FROM Lasku);
```

	huoltoNo
1	1
2	2
3	4
4	12

Huomataan, että huollosta numero 1 ei ole lähtenyt vielä laskua ollenkaan. Työntekijä aikoo luoda laskun ja tahtoo tätä varten selvittää huollon kokonaishinnan tehtyjen toimenpiteiden keston ja käytettyjen varaosien summana.
Tämä onnistuu seuraavalla kyselyllä:

```
SELECT Y AS 'Varaosien hinta', ROUND(S/60.0*75 , 2) AS 'Työt (75€/tunti)',
ROUND(S/60.0*75+Y , 2) AS Yhteensä
FROM (
  SELECT SUM(lukumäärä*hinta) AS Y, huoltoNo
  FROM Osat NATURAL JOIN Varaosa
  WHERE huoltoNo=1
  ) NATURAL JOIN (
  SELECT huoltoNo, SUM(kesto) as S
  FROM Toimenpide NATURAL JOIN Toimenpidevaraus
  WHERE huoltoNo=1);
```

	Varaosien hinta	Työt (75€/tunti)	Yhteensä
1	460	87.5	547.5

Työvuorovastaava suunnittelee kesän työvuoroja ja tahtoo tätä varten selvittää milloin Antti Herlin on lomalla:

```
SELECT alkamisaika, päättymisaika
FROM Poissaolo, Työntekijä
WHERE Poissaolo.hetu=Työntekijä.hetu AND Työntekijä.nimi='Antti Herlin';
```

	alkamisaika	päättymisaika
1	2022-08-01	2022-08-14

!!! Tietokannassa voi vielä toistaiseksi hakea työntekijöitä yksiselitteisesti nimellä, sillä kahta samannimistä henkilöä ei ole palkattu. !!!

Erääseen huoltoon tarvitaan jarrupaloja, joten työntekijä haluaa selvittää jarrupalojen hinnan, jotta voi antaa asiakkaalle hinta-arvion korjauksesta:

```
SELECT hinta
FROM Varaosa
WHERE tuotenimi='jarrupala';
```

	hinta
1	25

Työntekijä on tekemässä uutta varaosatilausta ja tahtoo selvittää, kuinka paljon mitäkin varaosaa on kulunut toukokuun aikana:

```
SELECT tuotenimi, SUM(lukumäärä) AS Total
FROM Osat
WHERE huoltoNo IN (
  SELECT huoltoNo
  FROM Huolto
  WHERE DATE(huoltoLoppu) BETWEEN '2022-05-01' AND '2022-05-31')
GROUP BY tuotenimi
```

	tuotenimi	Total
1	alatukivarsi	1
2	jakopäähihna	1
3	jarrulevy	4
4	jarrupala	8
5	kesärengas	1
6	talvirengas	4
7	tuulilasi	1
8	öljy	2