



Tietokantojen perusteet

SQL – Tietokantataulu

Taulun luominen, poistaminen

Rivien lisäys, muuttaminen, poistaminen

Yksinkertaisia kyselyjä

Tietokanta

- Tietokanta (database) on **kokoelma toisiinsa liittyviä tietoja**.
 - Tiedot kuvaavat jotakin reaalimaailman osa-alueutta, kohdealueutta (sovellusalueita).
 - Tiedot on talletettu jotakin käyttötarkoitusta varten.
- Tietokantaa käytetään **tietokannanhallintajärjestelmän** (TKHJ; database management system, DBMS) avulla.
 - Yleiskäytöinen ohjelmisto tietokannan luomiseen ja ylläpitämiseen
 - Tarjoaa tehokkaan ja hallitun ympäristön tietojen säilyttämiseen ja hyödyntämiseen samanaikaisille käyttäjille

Tietokanta ja TKHJ

- Tietokannanhallintajärjestelmän avulla voidaan

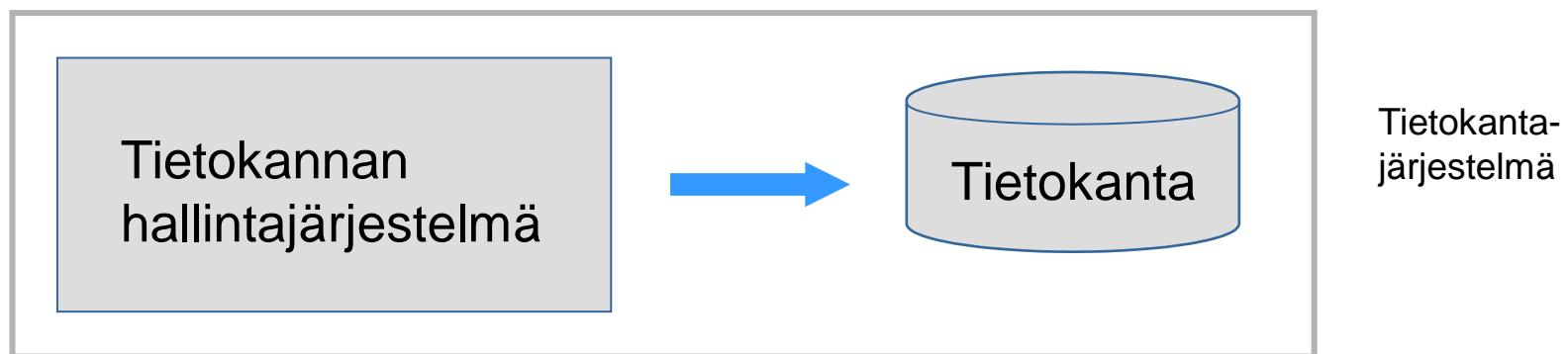
- luoda ja hallita tietokantoja

- lisätä tietoja
 - muuttaa tietoja
 - poistaa tietoja

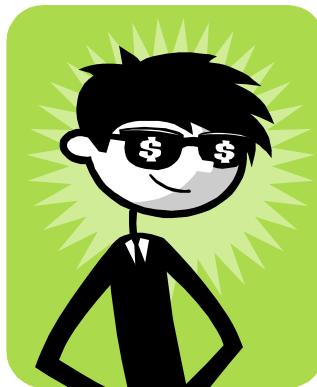
Tehdään päivityksiä

- hakea tietoja

Tehdään kyselyjä



Tietokantaa voidaan käyttää **sovellusohjelman** kautta



The screenshot shows a web browser window with the following details:

- Tab bar: TIEP3 Tietokantojen perusteet, ... × WETO: List of courses × Tietojenkäsittelytieteen kandi... × Informaatiotieteen yksikkö × +
- URL bar: i/listCourses.action
- Header: WETO
- Content:
 - Login form with fields for Username and Password, and a Submit button.
 - Text: "Links to courses you have joined are shown under "Your courses". Older courses (you are a member of) are listed under "Show old courses". Other available courses are listed under "Available courses".
 - Section: Computer Science
 - Section: Available courses

Tietokannan hallintajärjestelmä



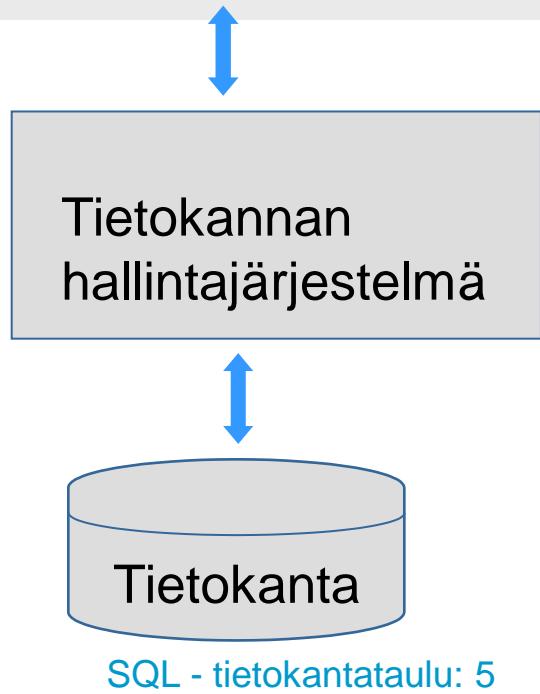
SQL - tietokantataulu: 4

Tietokantaa voidaan käyttää **komentotulkin** (komentorivikäyttöliittymän) kautta



Welcome to psql 8.1.2 (server 8.3.0), the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
\h for help with SQL commands
\? for help with psql commands
\g or terminate with semicolon to execute query
\q to quit



SQL-tietokanta

- Tällä kurssilla käsitellään SQL-tietokantoja.
- Termi SQL-tietokanta tulee näiden tietokantojen **käsittelyyn käytettävästä kielestä**, **SQL**:stä (Structured Query Language).
- SQL on nykypäivän tietokantojen käsittelyssä eniten käytetty kieli.
- SQL-tietokantoja kutsutaan myös **relaatiоракентеisiksi tietokannoiksi** tai **relaatiotietokannoiksi**.
 - Niiden taustalla oleva **tietomalli** on relaatiomalli.
 - Tietomalli on käsitteistö, jolla kuvataan tietokannan rakenne: tietojen tyyppi, tietojen väliset suhteet ja tietoja koskevat rajoitteet
 - Lisäksi tietomallit voivat sisältää operaatioita tietojen hakuun ja päivitykseen.
 - Relaatiomallia käsitellään Tietokantajärjestelmät: SQL -opintojaksolla.

SQL

- SQL:n avulla voidaan määritellä ja hallita tietokantoja sekä käsitellä niiden tietoja.
 - Data control language (DCL)
 - Data definition language (DDL)
 - Data manipulation language (DML)

SQL

- Standardoitu kieli
 - Viimeisimmät muutokset 2019
- Eri tietokannanhallintajärjestelmillä on suppeahko yhteinen standardin mukainen ydin ja lisäksi erilaisia laajennoksia.
- Tällä opintojaksolla pyritään keskittymään SQL-kielen ydinosaan.
 - Kalvojen esimerkit on tehty PostgreSQL-tietokannanhallintajärjestelmällä, joka tukee laajasti SQL:2016-standardin ydinominaisuuksia.
 - Harjoituksissa käytetään SQLite-järjestelmää, joka tukee suurinta osaa SQL:1992-standardin ominaisuuksista, noudattaa pitkälti PostgreSQL:n syntaksia ja on helppokäyttöinen.

SQL-taulu

- SQL-tietokannoissa tiedot järjestetään tauluihin.

Taulun nimi
tyontekija

Sarakkeen nimi

Rivi

Arvo

Puuttuva arvo

Sarake

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

SQL-taulu

- Rivi
 - vastaa yhtä kohdetta (objektilt. entiteettiä), jonka tietoja halutaan tauluun tallentaa
- Sarake
 - sisältää kohteeseen liittyvää tietoa
 - tieto on tiettytyyppistä
- Taulun rivien ja sarakkeiden järjestyksellä ei ole merkitystä taulun tietosisältöön
- Taulujen ja sarakkeiden nimet pyritään valitsemaan siten, että ne helpottavat riveillä olevien arvojen tulkintaa.
- Tietokantajärjestelmä sisältää
 - varsinaisia tietoja eli **dataa (ilmentymä)**
 - tietojen kuvaksen eli **metadataa (kaavio)**

Taulun luominen

- Taulu luodaan **CREATE TABLE** -lauseella, joka määrittelee
 - taulun nimen
 - taulun sarakkeiden nimet
 - Sarakkeiden nimien täytyy olla samassa taulussa toisistaan eriäviä.
 - minkä tyypistä tietoa taulun sarakkeisiin voidaan tallettaa
 - (yleensä) pääavaimen
 - mahdollisesti avaimia ja muita rajoitteita

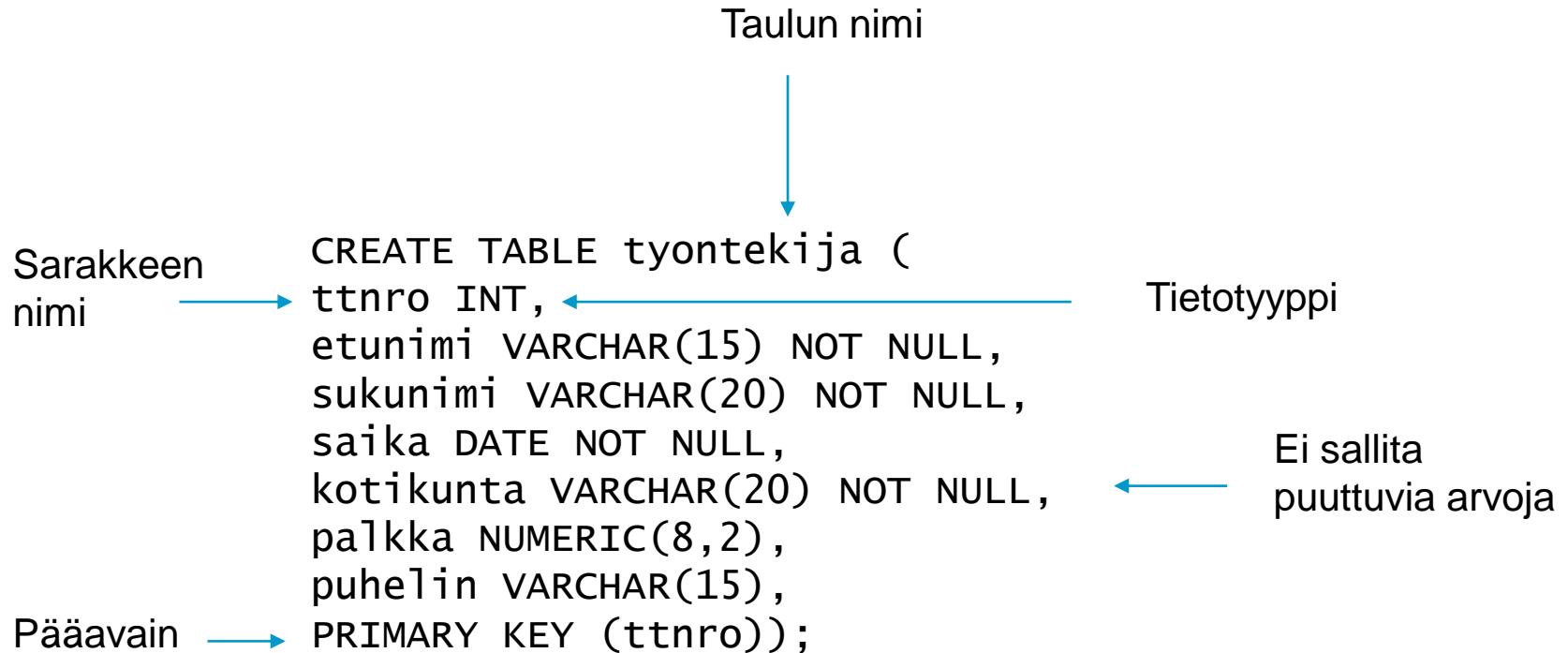
```
CREATE TABLE taulu (
    sarake tietotyyppi [rajoite]
    {, sarake tietotyyppi [rajoite]}
    [rajoite {,rajoite}]);
```

[] valinnainen osa

{ } osa toistuu 0...n kertaa

Merkit [,], {, } jätetään varsinaisesta luontilauseesta pois.

Taulun luominen



Taulun luominen

SQL ei erottele isoja ja pieniä kirjaimia toisistaan SQL:n avainsanoissa (key words, esim. CREATE ja INT) eikä taulujen ja sarakkeiden nimissä.

Kirjoitustyyli vaihtelee lähteestä riippuen. Näissä kalvoissa avainsanat on kirjoitettu isolla.

```
CREATE TABLE tyontekija (
ttnro INT,
etunimi VARCHAR(15) NOT NULL,
sukunimi VARCHAR(20) NOT NULL,
saika DATE NOT NULL,
kotikunta VARCHAR(20) NOT NULL,
palkka NUMERIC(8,2),
puhelin VARCHAR(15),
PRIMARY KEY (ttnro));
```

Komento voidaan jakaa usealle riville.

Puolipiste päättää komennon.

Useimmissa järjestelmissä taulujen ja sarakkeiden nimissä voi käyttää kirjaimia a-z. ('å', 'Å', 'ä', 'Ä', 'ö' ja 'Ö' eivät siis käy.)

Lisäksi voidaan käyttää numeroita (0-9) ja alaviivaa (_).

Nimen on alettava kirjaimella tai alaviivalla.

Taulun luominen: Tietotyyppejä

TIETOTYYPPI	KUVAUS	ESIMERKKI
INT INTEGER	kokonaisluku	ttnro INT
NUMERIC(L,S) DECIMAL(L,S)	desimaaliluku, jossa on yhteensä L numeroa, joista desimaaliosassa on S numeroa (numeric ja decimal ovat PostgreSQL:ssä ekvivalentteja.)	palkka NUMERIC(8, 2) palkka DECIMAL(8, 2)
CHAR(N) CHARACTER(N)	vakiomittainen merkkijono, pituus N merkkiä (N:ää merkkiä lyhyempien merkkijonojen loppu täytetään tyhjämerkeillä, blankoilla I. välilyönneillä)	spuoli CHAR spuoli CHAR(1) henkilotunnus CHAR(11)
VARCHAR(N) CHARACTER VARYING(N)	vaihtuvamittainen, korkeintaan N merkkiä pitkä merkkijono	etunimi VARCHAR(15)
DATE	päivämäärä	saika DATE

- Huom. SQLitella on uniikki, poikkeuksellisen joustava, dynaaminen typpijärjestelmä. Tätä ei käsitellä tarkemmin.
- Lisää tietotyypejä käydään läpi myöhemmin.

Taulun luominen: Rajoitteita

- **Primary key – pääavain (ensisijainen avain)**
 - yksilöi taulun rivit
 - voi muodostua
 - yhdestä sarakkeesta
 - sarakkeessa oltava eri arvo taulun jokaisella rivillä
 - useasta sarakkeesta
 - sarakkeissa olevien arvojen yhdistelmän oltava eri taulun jokaisella rivillä
 - puuttuvia arvoja ei sallita
 - ilmaistaan PRIMARY KEY -määreellä
PRIMARY KEY (sarake {,sarake})
 - taulussa voi olla yksi pääavain
 - tiedot löytyvät nopeasti pääavainta käyttämällä
 - tietokannanhallintajärjestelmä luo pääavainta varten erityisen hakemistorakenteen, jonka avulla tiedot löytyvät nopeasti pääavainta käyttämällä

Taulun luominen: Rajoitteita

- **Unique - avain**
 - voi muodostua
 - yhdestä sarakkeesta
 - sarakkeessa oltava eri arvo taulun jokaisella rivillä
 - useasta sarakkeesta
 - sarakkeissa olevien arvojen yhdistelmän oltava eri taulun jokaisella rivillä
 - PostgreSQL:ssä ja SQLite:ssä puuttuvat arvot sallitaan (vaihtelee järjestelmittäin)
 - ilmaistaan UNIQUE -määreellä
UNIQUE (sarake {,sarake})
 - taulussa voi olla monta avainta
 - PostgreSQL:ssä ja SQLite:ssä tietokannanhallintajärjestelmä luo avainta varten erityisen hakemistorakenteen, jonka avulla tiedot löytyvät nopeasti avainta käyttämällä.

Taulun luominen: Rajoitteita

```
CREATE TABLE osasto (
onro INT,
onimi VARCHAR(15) NOT NULL,
PRIMARY KEY (onro),
UNIQUE (onimi));
```

← Pääavain
← Avain

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

PostgreSQL:n PRIMARY KEY -määre sisältää aina NOT NULL -määreen (ei sallita puuttuvia arvoja)

SQLIten PRIMARY KEY -määre, joka koskee yhtä INT-tyyppistä saraketta, sisältää NOT NULL -määreen. Muulloin on annettava eksplisiittisesti NOT NULL -määre myös pääavainsarakkeille.

PostgreSQL:ssä ja SQLItessa UNIQUE-määre ei sisällä NOT NULL -määrettä, vaan se on annettava erikseen, jos puuttuvat arvot halutaan kielää.

Taulun luominen: Rajoitteita

- **NULL**
 - Puuttuva tieto merkitään NULL-arvolla.
 - Puhutaan NULL-arvoista tai tyhjäarvoista.
- Tieto voi puuttua sarakkeesta, koska
 - kohteella ei ole arvoa ko. sarakkeelle tai
 - kohteen arvo on tuntematon (arvoa ei tiedetä)
- Kaikki puuttuva tieto merkitään NULL-arvolla.
- **NOT NULL – ei puuttuvia arvoja**
 - Sarakkeella on aina oltava ei-tyhjä arvo
 - Kielletään tyhjäarvot taulun luontilauseessa

sarake TIETOTYYPPPI NOT NULL

Taulun luominen

- CREATE TABLE -lauseen suorituksen jälkeen tietokantajärjestelmässä on taulun määrittely (metadataa), mutta varsinaiset tiedot (data) puuttuvat.
- Taulun määrittely säilyy järjestelmässä niin kauan kunnes taulu käydään poistamassa DROP TABLE -lauseella.

Taulun rakenne

- Taulun rakenteen voi tarkistaa komentotulkissa (komentorivikäyttöliittymässä)

PostgreSQL-komennoilla

\d taulunimi

\d tyontekija

SQLite-komennoilla

.schema taulunimi

.schema tyontekija

- näyttää taulun luontilauseen

Column	Type	Modifiers
ttnro	integer	not null
etunimi	character varying(15)	not null
sukunimi	character varying(20)	not null
saika	date	not null
kotikunta	character varying(20)	not null
palkka	numeric(8,2)	
puhelin	character varying(15)	

Indexes:

"tyontekija_pkey" PRIMARY KEY, btree
(ttnro)

```
CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    PRIMARY KEY (ttnro));
```

Taulun poistaminen

- Taulun voi poistaa **DROP TABLE** -lauseella.
 - poistaa taulun määrittelyn ja varsinaisen tietosisällön

`DROP TABLE taulu;`

`DROP TABLE tyontekija;`

Taulun rakenteen muuttaminen

- Taulun rakennetta voi muuttaa **ALTER TABLE** -lauseella.
 - lisätä tai poistaa sarakkeita
 - vaihtaa sarakkeiden nimiä ja tietotyypejä
 - muuttaa taulun rajoitteita
- Mahdollisuudet muuttaa taulun rakennetta vaihtelevat tietokannanhallintajärjestelmittäin.
- ALTER TABLE -lausetta käydään läpi myöhemmin kurssilla.

Rivien lisääminen tauluun

- Rivejä (varsinaisia tietoja, dataa) lisätään tauluun **INSERT**-lauseella.
- **INSERT**-lause lisää tauluun yhden rivin.

```
INSERT INTO taulu [(sarake {, sarake} )]  
VALUES (vakioarvo {,vakioarvo});
```

```
INSERT INTO tyontekija  
VALUES (88, 'Jukka', 'Susi', '1957-11-10', 'Tampere', 5500.00, '444 1234');
```

- Merkkijonovakiot ja päivämäärät esitetään yksinkertaisissa lainausmerkkeissä (l. "hipsissa")
- Numeeriset vakiot esitetään ilman lainausmerkkejä.

Varsinaisessa datassa pienet ja isot kirjaimet erotellaan toisistaan.
'Susi' ei ole siis sama asia kuin 'SUSI' tai 'SuSi'

Rivien lisääminen

- Luettelomallia sarakkeet voidaan uudelleenkäytettäväissä lisäyslauseissa varautua taulun rakenteen muutoksiin.
 - Sarakkeiden järjestyksen vaihtumiseen
 - Uusien sarakkeiden lisäämiseen
- Lisäyslause on tässä muodossa myös informatiivisempi: se kertoo sarakkeet, joihin tietoja lisätään.

```
INSERT INTO
tyontekija(ttnro, etunimi, sukunimi, saika, kotikunta, palkka, puhelin)
VALUES (88, 'Jukka','Susi','1957-11-10','Tampere', 5500.00, '444 1234');
```

Rivien lisääminen

- Rivi, jolla on puuttuva arvo, lisätään tauluun
 - ilmaisemalla puuttuva arvo NULL-merkinnällä tai
 - luettelemalla ne sarakkeet, joiden arvo annetaan

```
INSERT INTO tyontekija  
VALUES (12, 'Pekka','Puro','1985-01-09','Tampere', 3000.00, NULL);
```

```
INSERT INTO  
tyontekija(ttnro, etunimi, sukunimi, saika, kotikunta, pa1kka)  
VALUES (12, 'Pekka','Puro','1985-01-09','Tampere', 3000.00);
```

- Tauluun lisättyt rivit säilyvät taulussa siihen asti, kunnes ne käydään poistamassa DELETE-lauseella tai koko taulu poistetaan DROP TABLE -lauseella.

Rivien lisääminen

- Tauluun lisättävien tietojen on noudatettava taulun määrittelyn yhteydessä annettuja vaatimuksia.
- SQL-komentotulkki ja tietokannanhallintajärjestelmä tarkastavat lisättävät tiedot.
- Jos lisättävät tiedot eivät vastaa taulun luontilausetta, niitä ei lisätä tietokantaan, vaan komentotulkki antaa virheilmoituksen.
- Huom. SQLItella on uniikki, poikkeuksellisen joustava, dynaaminen tyyppijärjestelmä. Yleensä tietokannanhallintajärjestelmissä tietotyyppi on sarakekohtainen, ja esimerkiksi kokonaislukusarakkeeseen ei voi tallentaa kirjaimista koostuvaa merkkijonoa, mutta SQLItessa tämä on mahdollista.
- SQLIten tyyppijärjestelmää ei käsitellä tarkemmin tällä opintojakson alla, vaan pitäydytään yleisesti käytössä olevassa sarakeperustaisessa tyyppijärjestelmässä.

Kyselyt

- Tietoja haetaan taulusta **SELECT**-lauseella.

```
SELECT sarake {, sarake}  
FROM taulu {,taulu}  
[WHERE ehto];
```

mistä sarakkeista
mistä taulusta
miltä riveiltä

- Kyselyn tuloksesta saadaan nimeton tulostaulu.
 - Jos kyselyssä ei ole WHERE-osaa, on tulostaulussa yksi rivi kutakin FROM-osan taulun (tai taulujen karteesisen tulon) riviä kohden.
 - Karteesinen tulo käsitellään myöhemmin.
- SQL on deklaratiivinen kyselykieli: kyselyssä määritellään, mitä haetaan (eikä sitä miten haetaan).

Kyselyt: SELECT-osa

- SELECT-osassa luetellaan sarakkeet, jotka halutaan tulostauluun mukaan.

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

```
SELECT etunimi, sukunimi, kotikunta  
FROM tyontekija;
```

etunimi	sukunimi	kotikunta
Jukka	Susi	Tampere
Ville	Viima	Nokia
Pekka	Puro	Tampere
Jenni	Joki	Lempäälä
Alli	Kivi	Nokia

tulostaulu

SQL - tietokantataulu: 28

Kyselyt: SELECT-osa

- * tarkoittaa "kaikki sarakkeet"

```
SELECT ttnro, etunimi, sukunimi, saika,  
       kotikunta, palkka, puhelin  
FROM tyontekija;
```

```
SELECT *  
FROM tyontekija;
```

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Kyselyt: SELECT-osa

- SQL sallii duplikaattien (ts. samojen rivien) esiintymisen sekä tauluissa että tulostauluissa.
 - Oletusarvoisesti duplikaatit säilytetään tulostauluissa.
 - Voidaan käyttää myös ALL-määreettä.
- Duplikaatit voidaan poistaa **DISTINCT**-määreellä.

```
SELECT kotikunta  
FROM tyontekija;
```

```
SELECT ALL kotikunta  
FROM tyontekija;
```

```
kotikunta  
-----  
Tampere  
Nokia  
Tampere  
Lempäälä  
Nokia
```

```
SELECT DISTINCT kotikunta  
FROM tyontekija;
```

```
kotikunta  
-----  
Lempäälä  
Nokia  
Tampere
```

Kyselyt: WHERE-osa

- Tulokseen tulevia rivejä rajataan WHERE-osassa annettavalla totuusarvoisella ehdolla.
 - Ehdossa tehdään yleensä vertailu(ja).
 - Tulokseen otetaan mukaan rivit, joille ehdon arvo on tosi (true).
 - SQL:ssä on kolme totuusarvoa
 - true - tosi
 - false - epätosi
 - unknown - tuntematon (tyhjäarvojen vuoksi)

Kyselyt: WHERE-osa

Haetaan työntekijät, joiden kotikunta on Tampere.

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE kotikunta = 'Tampere';
```

verrataan sarakkeen arvoa
merkkijonovakioon

etunimi		sukunimi
-----+-----		
Jukka		Susi
Pekka		Puro

Kyselyt: WHERE-osa

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE etunimi = kotikunta;
```

Verrataan sarakkeen arvoa
toisen sarakkeen arvoon.

- Ehdoissa käytettäviä vertailuoperaattoreita
 - = yhtä suuri kuin
 - <> eri suuri kuin
 - < pienempi kuin
 - <= pienempi tai yhtä suuri kuin
 - > suurempi kuin
 - >= suurempi tai yhtä suuri kuin
- Muita operaattoreita esitellään myöhemmin.

Kyselyt: WHERE-osa

- Operaattorit <, <=, >, >= sopivat tietotyypeille, joille järjestyksen määrääminen on järkevää

```
SELECT sukunimi, etunimi, saika  
FROM tyontekija  
WHERE saika < '1985-01-01';
```

Haetaan ennen vuotta 1985
syntyneet työntekijät

sukunimi	etunimi	saika
Susi	Jukka	1957-11-10
Viima	Ville	1975-12-08
Joki	Jenni	1961-06-20

WHERE-osa: tyhjäarvot

- Tyhjäarvot ovat keskenään eri suuria.
- Tyhjäarvon (NULL-arvon) vertailu antaa AINA arvoksi **tuntematon**.
 - Vertailuoperaattorit = ja <> EIVÄT siis sovi tyhjäarvon olemassaolon testaamiseen!!
- Tyhjäarvon olemassaoloa testataan operaattoreilla IS NULL ja IS NOT NULL
 - sarake IS NULL
 - tosi silloin, kun sarakkeessa on tyhjäarvo
 - sarake IS NOT NULL
 - tosi silloin, kun sarakkeessa on ei-tyhjä arvo

WHERE-osa: tyhjäarvot

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Kuinka tyhjäarvot käyttäytyvät?

a)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin = '444 1234';
```

ttnro

88

(1 row)

b)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin <> '444 1234';
```

ttnro

33

98

99

(3 rows)

SQL - tietokantataulu: 36

WHERE-osa: tyhjäarvot

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Kuinka tyhjäarvot käyttäytyvät?

c)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin = NULL;
```

ttnro

(0 rows)

d)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin <> NULL;
```

ttnro

(0 rows)

TYHJÄARVOJA EI TESTATA NÄIN

SQL - tietokantataulu: 37

WHERE-osa: tyhjäarvot

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Kuinka tyhjäarvot käyttäytyvät?

e)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin IS NULL;
```

ttnro

12

(1 row)

TYHJÄARVOJEN TESTAUS OK

f)

```
SELECT ttnro  
FROM tyontekija  
WHERE puhelin IS NOT NULL;
```

ttnro

88

33

98

99

(4 rows)

SQL - tietokantataulu: 38

Kyselyt: Tulosrivien järjestäminen

- Tulostaulun rivit voidaan järjestää ORDER BY -määreellä.

```
SELECT [DISTINCT] sarake {, sarake}
FROM taulu {,taulu}
[WHERE ehto]
[ORDER BY sarake [ASC|DESC] {, sarake [ASC|DESC]}]
```

| -merkki ilmaisee vaihtoehtoiset avainsanat

- ASC nouseva järjestys
 - Lajittelujärjestys on oletusarvoisesti nouseva; ASC-määrettä ei tarvitse välittämättä käyttää.
- DESC laskeva järjestys
- Lajittelu suoritetaan sisäkkäin ORDER BY -listan mukaisessa järjestyksessä.

Kyselyt: Tulosrivien järjestäminen

Järjestetään tulosrivit sukunimen mukaan laskevasti.

```
SELECT sukunimi, kotikunta  
FROM tyontekija  
ORDER BY sukunimi DESC;
```

sukunimi	kotikunta
Viima	Nokia
Susi	Tampere
Puro	Tampere
Kivi	Nokia
Joki	Lempäälä

Järjestetään tulosrivit nousevasti ensin kotikunnan mukaan ja sitten sukunimen mukaan.

```
SELECT kotikunta, sukunimi  
FROM tyontekija  
ORDER BY kotikunta, sukunimi;
```

kotikunta	sukunimi
Lempäälä	Joki
Nokia	Kivi
Nokia	Viima
Tampere	Puro
Tampere	Susi

Kyselyt: Tulosrivien järjestäminen

```
SELECT puhelin  
FROM tyontekija  
ORDER BY puhelin;
```

puhelin

```
-----  
444 1234  
444 4343  
444 4488  
444 5555
```

(5 rows)▼

```
SELECT puhelin  
FROM tyontekija  
ORDER BY puhelin DESC;
```

puhelin

```
-----  
444 5555  
444 4488  
444 4343  
444 1234  
(5 rows)
```

tyhjäarvo

Kyselyn evaluointialgoritmi

- Yksinkertaisen SQL-kyselyn rakenne

SELECT sarakeluettelo

FROM taulunnnimi

[WHERE ehto]

[ORDER BY];

Yksinkertainen kyselyn evaluointialgoritmi: kyselyn osien suoritusjärjestys

1. FROM-osa

- Kopioidaan FROM-osassa annettu taulu tulostauluksi.

2. WHERE-osa

- Poistetaan edellisen vaiheen tulostaulusta ne rivit, jotka eivät täytä ehtoa.

3. SELECT-osa

- Poistetaan kaikki sarakkeet, jotka eivät esiinny SELECT-listassa.

4. ORDER BY -osa

- Järjestetään rivit järjestystekijöiden (sarakkeiden) mukaisesti.

Tietokannanhallintajärjestelmä ei välttämättä evaluoi kyselyä tällä tavoin –

algoritmin tarkoituksesta on auttaa ymmärtämään, miten kyselyn tulos muodostuu.

Rivien muuttaminen

- Rivejä muutetaan UPDATE-lauseella.

UPDATE taulu

```
SET sarake = ilmaus {, sarake = ilmaus}
[WHERE ehto];
```

- Muutettavat rivit rajataan WHERE-osan ehdolla.
 - Jos WHERE-osan ehtoa ei anneta, päivitetään kaikkia rivejä.
- Ilmauksessa voidaan käyttää operaatioita ja funktioita.
 - Näistä lisää myöhemmin.

Rivien muuttaminen

```
UPDATE tyontekija
```

```
SET sukunimi = 'Myrsky', kotikunta = 'Orivesi'  
WHERE ttnro = 99;
```

Muutokset riville, jolla
ttnro-sarakkeen arvo
on 99.

```
UPDATE tyontekija
```

```
SET palkka = 4050.00
```

```
WHERE palkka = 4000.50;
```

2
1 Ehdon totuusarvo
lasketaan ensin.

Tämän jälkeen
päivitetään ehdossa
testatun sarakkeen
arvoa.

Rivien poistaminen

- Rivien poistaminen tapahtuu DELETE-lauseella.

```
DELETE FROM taulu  
[WHERE ehto];
```

- Poistettavat rivit rajataan WHERE-osan ehdolla.
 - Jos WHERE-osan ehtoa ei anneta, **poistetaan kaikki rivit!**
 - Taulun määrittely säilyy kuitenkin järjestelmässä.

```
DELETE FROM tyontekija;
```

```
DELETE FROM tyontekija  
WHERE ttnro > 20;
```

Esimerkki kyselyn evaluointialgoritmin käytöstä

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Minkälaisen tulostaulun alla oleva kysely tuottaa, kun sitä sovelletaan yllä annettuun tyontekija-tauluun?

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE kotikunta = 'Tampere'  
ORDER BY sukunimi;
```

Esimerkki kyselyn evaluointialgoritmin käytöstä

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

1. FROM-osa

FROM tyontekija

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

Esimerkki kyselyn evaluointialgoritmin käytöstä

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

2. WHERE-osa

WHERE kotikunta = 'Tampere'

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
12	Pekka	Puro	1985-01-09	Tampere	3000.00	

Esimerkki kyselyn evaluointialgoritmin käytöstä

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
12	Pekka	Puro	1985-01-09	Tampere	3000.00	

3. SELECT-osa

SELECT etunimi, sukunimi

etunimi	sukunimi
Jukka	Susi
Pekka	Puro

Esimerkki kyselyn evaluointialgoritmin käytöstä

etunimi	sukunimi
Jukka	Susi
Pekka	Puro

4. ORDER BY -osa

ORDER BY sukunimi

etunimi	sukunimi
Pekka	Puro
Jukka	Susi

Kuinka tietokannan taulun tila muuttuu - UPDATE

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555

UPDATE tyontekija

SET sukunimi = 'Myrsky', kotikunta = 'Orivesi'

WHERE ttnro = 99;

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Myrsky	1988-07-19	Orivesi	2500.00	444 5555

Kuinka tietokannan taulun tila muuttuu - UPDATE

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Myrsky	1988-07-19	Orivesi	2500.00	444 5555

UPDATE tyontekija

SET palkka = 4050.00

WHERE palkka = 4000.50;

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4050.00	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Myrsky	1988-07-19	Orivesi	2500.00	444 5555

Kuinka tietokannan taulun tila muuttuu - DELETE

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234
33	Ville	Viima	1975-12-08	Nokia	4050.00	444 4343
12	Pekka	Puro	1985-01-09	Tampere	3000.00	
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488
99	Alli	Myrsky	1988-07-19	Orivesi	2500.00	444 5555

```
DELETE FROM tyontekija  
WHERE ttnro > 20;
```

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin
12	Pekka	Puro	1985-01-09	Tampere	3000.00	



Tietokantojen perusteet

SQL – Useita tauluja

Taulujen väliset riippuvuudet

Viiteavain, viite-eheys

Tietokannan kaavion graafinen esitys

Useita tauluja

- SQL-tietokanta koostuu yleensä useista tauluista, joiden tiedot liittyvät toisiinsa.
- Luentojen esimerkkisovellusalueella on yrityksen työntekijöille, osastolle ja projekteille kuillekin oma taulunsa.
- Taulujen tiedot liittyvät toisiinsa:
 - Kukin työntekijä työskentelee jollakin osastolla.
 - Osa työntekijöistä toimii toisten työntekijöiden esimiehinä.
 - Osa työntekijöistä osallistuu projekteihin tiettyllä viikkotuntimäärällä.
- Taulujen tietoja liitetään toisiinsa **viiteavainrakenteen** (vierasavainrakenteen) avulla.

Viiteavain

- **Viiteavain** (foreign key) on
 - sarake tai sarakeyhdistelmä,
 - jonka kaikki arvot ovat jonkin taulun pääavaimen arvoja tai tyhjäarvoja
- Viiteavain voi viittata toiseen tauluun.

tyontekija						osasto	
ttnro	etunimi	sukunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka	Susi		1		1	Pääkonttori
33	Ville	Viima		5	88	4	Hallinto
12	Pekka	Puro		5	33	5	Tutkimus
98	Jenni	Joki		4	88		
99	Alli	Kivi		4	98		

Osastonro-sarake viittaa osasto-taulun sarakkeeseen onro. Se liittää tyontekija-taulun rivejä osasto-taulun riveihin.

Esim. Alli Kivi työskentelee Hallinto-osastolla.[SQL - useita tauluja 3](#)

Viiteavain

- Viiteavain voi viitata tauluun itseensä.
 - Liittää saman taulun rivejä toisiinsa.

tyontekija

ttnro	etunimi	sukunimi	...	osastonro	esimiesnro
88	Jukka	Susi		1	
33	Ville	Viima		5	88
12	Pekka	Puro		5	33
98	Jenni	Joki		4	88
99	Alli	Kivi		4	98

Esim.
Jukka Sudella ei
ole esimiestä.

Ville Viiman
esimies on
Jukka Susi.

Esimiesnro-sarake viittaa tyontekija-taulun
sarakkeeseen ttnro. Se liittää tyontekija-taulun
rivejä toisiinsa.

Viiteavain

- Viiteavain määritellään taulun luontilauseessa **FOREIGN KEY** - avainsanoilla
`FOREIGN KEY(sarake {,sarake}) REFERENCES taulu [(pääavainsarake {,pääavainsarake})]`
- Viiteavaimen on oltava yhteensoviva viitattavan taulun pääavaimen kanssa.
 - sarakkeiden lukumäärän ja
 - tietotyyppien
on vastattava toisiaan

Viiteavain

```
CREATE TABLE osasto (
onro INT,
onimi VARCHAR(15) NOT NULL,
PRIMARY KEY (onro),
UNIQUE (onimi));
```

Taulu, johon viitataan (jostakin toisesta taulusta), on luotava ensin.

```
CREATE TABLE tyontekija (
ttnro INT,
etunimi VARCHAR(15) NOT NULL,
sukunimi VARCHAR(20) NOT NULL,
saika DATE NOT NULL,
kotikunta VARCHAR(20) NOT NULL,
palkka NUMERIC(8,2),
puhelin VARCHAR(15),
osastonro INT NOT NULL,
esimiesnro INT,
PRIMARY KEY (ttnro),
FOREIGN KEY (osastonro) REFERENCES osasto(onro),
FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro));
```

Viiteavain

tyontekija

ttnro	etunimi	...	esimiesnro
88	Jukka		
33	Ville		88
12	Pekka		33
98	Jenni		88
99	Alli		98

projekti

pnro	pnimi
1	Tuote X
2	Tuote Y
3	Tuote Z

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

- Taulu voi toimia kahden taulun tietojen yhdistäjänä.

Osallistuu-taulu yhdistää tyontekija- ja projekti-taulujen rivejä toisiinsa.

Esim. Pekka osallistuu Tuote X -projektiin 32.5 viikkotunnin verran.

Viiteavain

```
CREATE TABLE projekti (
pnro INT,
pnimi VARCHAR(15) NOT NULL,
PRIMARY KEY (pnro),
UNIQUE (pnimi));
```

osallistuu-taulun pääavain muodostuu ttnro- ja pnro-sarakkeista.

Kukin (ttnro, pnro)-pari voi esiintyä taulussa vain yhdellä rivillä.

```
CREATE TABLE osallistuu (
ttnro INT,
pnro INT,
tunnit NUMERIC(3,1),
PRIMARY KEY (ttnro,pnro),
FOREIGN KEY (ttnro) REFERENCES tyontekija, ← Viittauksen kohteena olevan
FOREIGN KEY (pnro) REFERENCES projekti); pääavainsarakkeen voi jättää
pois.
```

projekti

pnro	pnimi
1	Tuote X
2	Tuote Y
3	Tuote Z

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

SQL - useita tauluja 8

Viite-eheys

- Viiteavainrajoitteita ei ole välttämätöntä antaa taulua määriteltääessä, mutta ...
- kun ne on määritelty, tietokannanhallintajärjestelmä (esim. PostgreSQL) tarkistaa tietoja päivitetäessä viittausten johdonmukaisuuden.
- SQLite on poikkeus eikä valvo viite-eheyttä oletusarvoisesti, vaan viite-eheyden valvonta laitetaan päälle komennolla
`PRAGMA foreign_keys = ON;`
- Voidaan viittata vain olemassa oleviin riveihin ja arvoihin.

Viite-eheys

Alla oleva rivin lisääminen ei onnistu, koska osastotaulussa ei ole riviä, jolla onro = 10.

```
INSERT INTO tyontekija  
VALUES (102, 'Nelli', 'Naakka', '1988-09-09',  
'Toijala', 2900.00, NULL, 10, 33);
```

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

PostgreSQL-tietokannanhallintajärjestelmän antama virheilmoitus:

```
=> INSERT INTO tyontekija  
-> VALUES (102, 'Nelli', 'Naakka', '1988-09-09', 'Toijala', 2900.00, NULL, 10, 33);
```

```
ERROR: insert or update on table "tyontekija" violates  
foreign key constraint "tyontekija_osastonro_fkey"  
DETAIL: Key (osastonro)=(10) is not present in table "osasto".
```

SQLite-tietokannanhallintajärjestelmän antama virheilmoitus:

SQL error: foreign key constraint failed

Viite-eheys PostgreSQL:ssä ja SQLItessa

- Viiteavainmäärittelyn yhteydessä voidaan antaa toimintasääntö, minkä mukaan toimitaan, jos muutos- tai poisto-operaatio (UPDATE tai DELETE) uhkaa rikkoa viite-eheyden.
- Toimintasäännöt PostgreSQL:ssä ja SQLItessa
 - NO ACTION
 - oletussääntö
 - Jos viitteen kohde katoaisi operaation seurauksena, operaatio estetään (ei tehdä muutosta tai poistoa) *
 - RESTRICT
 - Jos viitteen kohde katoaa, operaatio estetään*
 - CASCADE
 - Muutokset vyörytetään viitanneisiin riveihin
 - SET NULL
 - Jos viitteen kohde katoaa, asetetaan viitannut arvo tyhjäarvoksi.
 - SET DEFAULT
 - Jos viitteen kohde katoaa, asetetaan viitannut arvo oletusarvoksi.

*NO ACTION ja RESTRICT –toimintasäännöillä on ero PostgreSQL:ssä ja SQLItessa (liittyy transaktioihin); tämä on tietokantojen jatkokurssien asioita.

Viite-eheys

```
CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    osastonro INT NOT NULL,
    esimiesnro INT,
    PRIMARY KEY (ttnro),
    FOREIGN KEY (osastonro) REFERENCES osasto(onro),
    FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro));
```

Kun viiteavain määritellään, on NO ACTION -toimintasääntö oletusarvoisesti voimassa muutos- ja poisto-operaatioille.

Viite-eheys: PostgreSQL:n ja SQLiten toimintasäännöt

NO ACTION

Table "public.tyontekija"

Column	Type	Modifiers
ttnro	integer	not null
etunimi	character varying(15)	not null
sukunimi	character varying(20)	not null
saika	date	not null
kotikunta	character varying(20)	not null
palkka	numeric(8,2)	
puhelin	character varying(15)	
osastonro	integer	not null
esimiesnro	integer	

Indexes:

"tyontekija_pkey" PRIMARY KEY, btree (ttnro)

Foreign-key constraints:

"tyontekija_esimiesnro_fkey" FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro)
"tyontekija_osastonro_fkey" FOREIGN KEY (osastonro) REFERENCES osasto(onro)

Referenced by:

TABLE "osallistuu" CONSTRAINT "osallistuu_ttnro_fkey" FOREIGN KEY (ttnro)
REFERENCES tyontekija(ttnro)
TABLE "tyontekija" CONSTRAINT "tyontekija_esimiesnro_fkey" FOREIGN KEY
(esimiesnro) REFERENCES tyontekija(ttnro)

Viite-eheys

NO ACTION

- Viittauksen kohteena olevaa arvoa ei voi muuttaa eikä riviä poistaa.
- Muita arvoja voidaan muuttaa.
- Rivi, joka ei ole viittauksen kohteena, voidaan poistaa.

NO ACTION

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

Muutosoperaatiota

```
UPDATE osasto  
SET onro = 500  
WHERE onro = 5;
```

ei suoriteta, vaan seuraaksena on tietokannanhallintajärjestelmän antama virheilmoitus:

```
=> UPDATE osasto  
-> SET onro = 500  
-> WHERE onro = 5;
```

```
ERROR: update or delete on table "osasto" violates  
foreign key constraint "tyontekija_osastonro_fkey" on table "tyontekija"  
DETAIL: Key (onro)=(5) is still referenced from table "tyontekija".
```

NO ACTION

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

Poisto-operaatiota

```
DELETE FROM osasto  
WHERE onro = 5;
```

ei suoriteta, vaan seuraaksena on tietokannanhallintajärjestelmän antama virheilmoitus:

```
=> DELETE FROM osasto  
-> WHERE onro = 5;
```

ERROR: update or delete on table "osasto" violates foreign key constraint "tyontekija_osastonro_fkey" on table "tyontekija"
DETAIL: Key (onro)=(5) is still referenced from table "tyontekija".

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

```
UPDATE osasto
SET onimi = 'Tutkimuskeskus'
WHERE onro = 5;
```

```
UPDATE tyontekija
SET osastonro = 5,
    esimiesnro = 33
WHERE ttnro = 99;
```

taulut muutosoperaatioiden jälkeen:

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimuskeskus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		5	33

NO ACTION

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus
6	Markkinointi

Muutos

```
UPDATE osasto  
SET onro = 600  
WHERE onro = 6;
```

onnistuu, koska arvo 6 ei ole viittauksen kohteena.

taulut muutosoperaation jälkeen:

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus
600	Markkinointi

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

NO ACTION

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus
6	Markkinointi

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

Poisto

```
DELETE FROM osasto  
WHERE onro = 6;
```

onnistuu, koska poistettava rivi (rivillä oleva arvo 6) ei ole viittauksen kohteena.

taulut poisto-operaation jälkeen:

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

Viite-eheys: PostgreSQL:n ja SQLiten toimintasäännöt

CASCADE

```
CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    osastonro INT NOT NULL,
    esimiesnro INT,
    PRIMARY KEY (ttnro),
    FOREIGN KEY(osastonro) REFERENCES osasto(onro)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

Oletetaan kalvoilla 21-23, että tietokannassa on vain osasto- ja tyontekija-taulut.

Viite-eheys: PostgreSQL:n ja SQLiten toimintasäännöt

CASCADE

Table "public.tyontekija"

Column	Type	Modifiers
ttnro	integer	not null
etunimi	character varying(15)	not null
sukunimi	character varying(20)	not null
saika	date	not null
kotikunta	character varying(20)	not null
palkka	numeric(8,2)	
puhelin	character varying(15)	
osastonro	integer	not null
esimiesnro	integer	

Indexes:

"tyontekija_pkey" PRIMARY KEY, btree (ttnro)

Foreign-key constraints:

"tyontekija_esimiesnro_fkey" FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro) ON UPDATE CASCADE ON DELETE CASCADE

"tyontekija_osastonro_fkey" FOREIGN KEY (osastonro) REFERENCES osasto(onro) ON UPDATE CASCADE ON DELETE CASCADE

Referenced by:

TABLE "tyontekija" CONSTRAINT "tyontekija_esimiesnro_fkey" FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro) ON UPDATE CASCADE ON DELETE CASCADE

Viite-eheys: PostgreSQL:n ja SQLiten toimintasäännöt ON UPDATE CASCADE

- Kun viittauksen kohteena olevaa arvoa muutetaan, tehdään vastaava muutos päivitettyyn riviin viitanneisiin riveihin.

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

UPDATE osasto

SET onro = 500

WHERE onro = 5;

Taulut UPDATE-lauseen suorituksen jälkeen

onro	onimi
1	Pääkonttori
4	Hallinto
500	Tutkimus

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		500	88
12	Pekka		500	33
98	Jenni		4	88
99	Alli		4	98

Viite-eheys: PostgreSQL:n ja SQLiten toimintasäännöt

ON DELETE CASCADE

- Kun viittauksen kohteena oleva rivi poistetaan, poistetaan myös siihen viitanneet rivit

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

```
DELETE FROM osasto  
WHERE onro = 5;
```

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

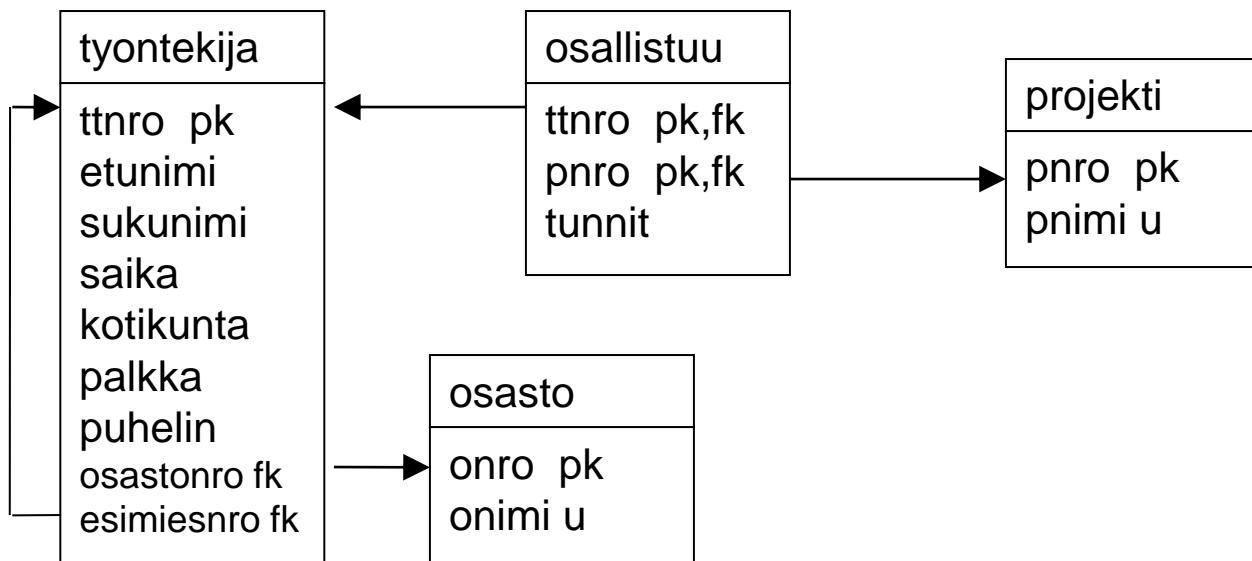
Taulut DELETE-lauseen suorituksen jälkeen

onro	onimi
1	Pääkonttori
4	Hallinto

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
98	Jenni		4	88
99	Alli		4	98

Pää- ja viiteavaimet tietokannan kaavion graafisessa esityksessä

- Eräs tietokannan kaavion graafinen esitystapa
 - taulut suorakaiteita, joissa sarakkeet lueteltu allekkain
 - pääavaimet merkitty pk-merkinnällä
 - avaimet merkitty u-merkinnällä
 - viiteavaimet merkitty fk-merkinnällä ja nuolilla: viiteavaimen kohdalta lähtee nuoli viittauksen kohteeseen



- Tietokannan kaavio on kokoelma tietokannan taulujen kaavioita.

SQL - useita tauluja 24



Tietokantojen perusteet

SQL – Useita tauluja - Kyselyt

Kyselyjä useista tauluista

Karteesinen tulo, liitosehto

Liitosehto ja valintaehdot

Liitosoperaatiot

Taulun liittäminen itseensä

Kyselyt useista tauluista

- SELECT-lauseen FROM-osassa voidaan luetella useita tauluja.
- Tulostauluna saadaan lueteltujen **taulujen rivien kaikki mahdolliset yhdistelmät** eli taulujen rivien **karteesinen tulo** eli **ristitulo**.

```
SELECT *
FROM tyontekija, osasto;
```

tyontekija

ttnro	etunimi	sukunimi	...	osastonro	esimiesnro
88	Jukka	Susi		1	
33	Ville	Viima		5	88
12	Pekka	Puro		5	33
98	Jenni	Joki		4	88
99	Alli	Kivi		4	98

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

Tulostaulun rivit ovat yhdistettyjä rivejä, joilla on kaikki sarakkeet kaikista FROM-osan tauluista. Alla on yksi tulostaulun rivi:

ttnro	etunimi	sukunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka	Susi		1		1	Pääkonttori

```
SELECT *
FROM tyontekija, osasto;
```

tyontekija-taulun
sarakkeet

osasto-taulun
sarakkeet

tyontekija-taulussa
5 riviä

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

osasto-taulussa 3
riviä

karteesisessä
tulossa $5 \times 3 = 15$
riviä

Kyselyt useista tauluista: Liitoseheto

- Karteisen tulon sijaan kyselyn tulokseksi halutaan yleensä **toisiinsa liittyviä tietoja**.
- Rajataan tulostauluun tulevia rivejä WHERE-osassa annettavalla **liitosehdolla**

Haetaan työntekijät osastoineen.

```
SELECT sukunimi, etunimi, onimi  
FROM tyontekija, osasto  
WHERE osastonro = onro;           liitoseheto
```

sukunimi	etunimi	onimi
Susi	Jukka	Pääkonttori
Viima	Ville	Tutkimus
Puro	Pekka	Tutkimus
Joki	Jenni	Hallinto
Kivi	Alli	Hallinto

FROM tyontekija, osasto
WHERE osastonro = onro

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

Kyselyt useista tauluista

Haetaan työntekijät osastoineen.

```
SELECT tyontekija.sukunimi, tyontekija.etunimi, osasto.onimi  
FROM tyontekija, osasto  
WHERE tyontekija.osastonro = osasto.onro;
```

sukunimi	etunimi	onimi
Susi	Jukka	Pääkonttori
Viima	Ville	Tutkimus
Puro	Pekka	Tutkimus
Joki	Jenni	Hallinto
Kivi	Alli	Hallinto

Taulun nimeä voidaan käyttää
sarakkeen **tarkenteena**:
taulunimi.sarakenimi

Kyselyt useista tauluista

Haetaan työntekijät osastoineen.

```
SELECT tt.sukunimi, tt.etunimi, os.onimi  
FROM tyontekija AS tt, osasto AS os  
WHERE tt.osastonro = os.onro;
```

sukunimi	etunimi	onimi
Susi	Jukka	Pääkonttori
Viima	Ville	Tutkimus
Puro	Pekka	Tutkimus
Joki	Jenni	Hallinto
Kivi	Alli	Hallinto

Taululle voidaan antaa FROM-osassa **uusi, tilapäinen nimi**.

Lyhyttä tilapäistä nimeä käyttämällä pääsee pienemmällä kirjoittamisen vaivalla, kun taulujen nimiä käytetään tarkenteina.

Kyselyt useista tauluista: Kyselyn evaluointialgoritmi

Algoritmi, jonka tarkoituksesta on auttaa ymmärtämään, miten kyselyn tulos muodostuu:

Vaihe 1. FROM-osa

- Muodostetaan FROM-osassa annettujen **taulujen karteesinen tulo eli ristitulo**, joka on tämän vaiheen tulostaulu.

Vaihe 2. WHERE-osa

- Poistetaan edellisen vaiheen tulostaulusta ne rivit, jotka eivät täytä **liitos- ja tai valintaehdoja**.

Vaihe 3. SELECT-osa

- Poistetaan edellisen vaiheen tulostaulusta sarakkeet, jotka eivät esiinny SELECT-listassa.

4. ORDER BY -osa

- Järjestetään rivit järjestystekijöiden (sarakkeiden) mukaisesti.

Tämän algoritmin tarkoituksesta on toimia ajattelun apuvälineenä.

Kyselyt useista tauluista: Liitosehdo ja valintaehdo

- Yhdistetään WHERE-osassa **liitosehdo** ja **valintaehdo AND**-operaattorilla.

Haetaan Hallinto-osastolla työskentelevät työntekijät.

```
SELECT sukunimi, etunimi, kotikunta  
FROM tyontekija, osasto  
WHERE osastonro = onro AND  
      onimi = 'Hallinto';
```

liitosehdo
valintaehdo

sukunimi	etunimi	kotikunta
Joki	Jenni	Lempäälä
Kivi	Alli	Nokia

Tässä kyselyssä WHERE-osan ehto koostuu kahdesta alkeisehdosta, jotka on yhdistetty AND-operaattorilla (JA-operaattorilla). Jotta yhdistetty ehto tuottaa totuusarvon tosi (true), on molempien alkeisehtojen tuottettava totuusarvo tosi. Loogiset operaatiot käydään läpi myöhemmin.

Esimerkki: Kyselyn evaluointi

FROM tyontekija, osasto

Vaiheen 1 tulostaulu

Sovelletaan kyselyn evaluointialgoritmia edellisen kalvon kyselyyn.

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

Esimerkki: Kyselyn evaluointi

```
WHERE osastonro = onro AND  
      onimi = 'Hallinto'
```

Vaiheen 2 tulostaulu:

karteesisesta tulosta poistettu ne rivit, jotka eivät täytä liitos- ja valintaehdoja

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto

Esimerkki: Kyselyn evaluointi

```
SELECT sukunimi, etunimi, kotikunta
```

Vaiheen 3 tulostaulu:

Poistettu kaikki sarakkeet, jotka eivät esiinny SELECT-listassa.
(Sarakkeet järjestetty SELECT-listan mukaiseen järjestykseen.)

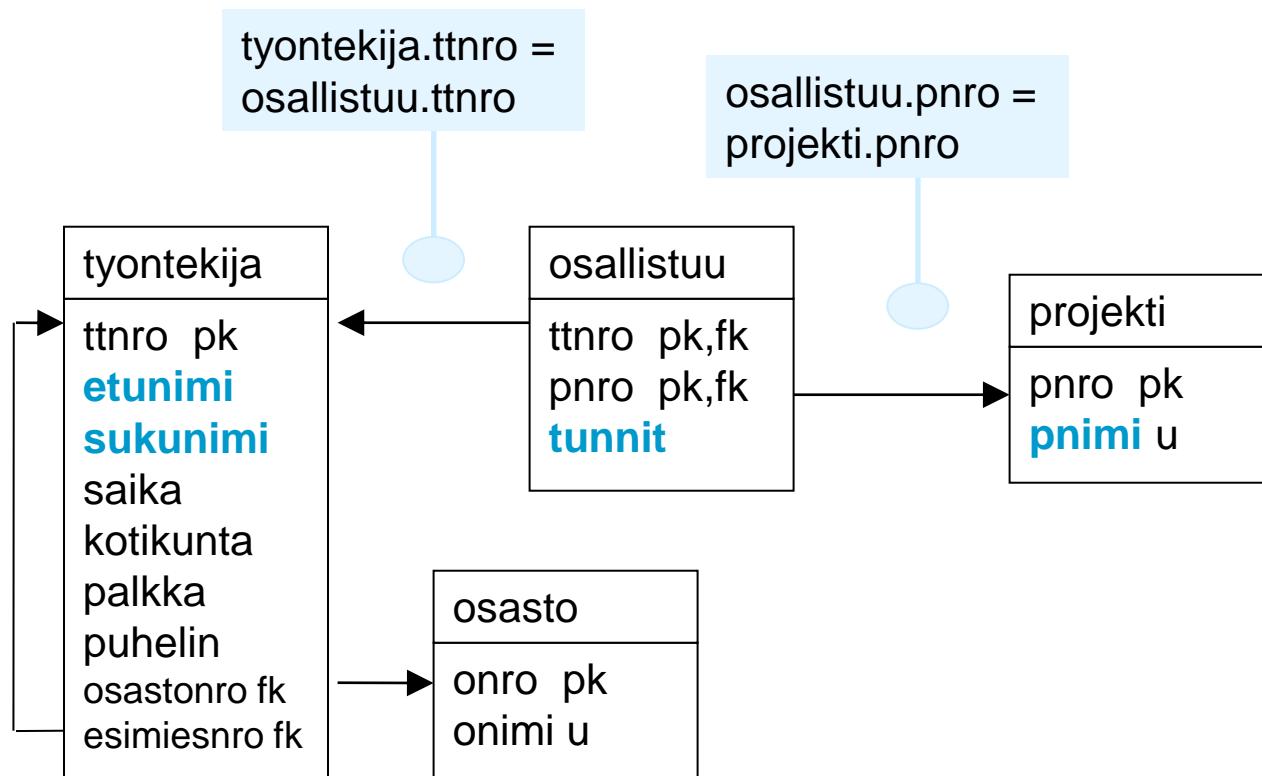
sukunimi	etunimi	kotikunta
Joki	Jenni	Lempäälä
Kivi	Alli	Nokia

Kyselyt useista tauluista

- Kyselyn rakentaminen:
 - Mistä tauluista otetaan tietoja mukaan tulostauluun?
 - Miten tiedot liitetään toisiinsa?
 - Jos FROM-osassa on annettu useita tauluja, tarvitaan yleensä liitosehto(ja) taulujen tietojen yhdistämiseksi.
 - Jos tauluja on N kappaletta, tarvitaan yleensä vähintään N-1 liitosehtoa.
 - Liitosehdossa testataan tyypillisesti pääavaimen ja viiteavaimen yhtäsuuruutta. Nämä voivat koostua useasta sarakkeesta → liitosehdo muodostuu useasta alkeisehdosta
 - Tarvitaanko jotakin taulua tietojen liittämiseksi toisiinsa?
 - Tarvitaanko jotakin taulua tulosrivien rajaamista varten?

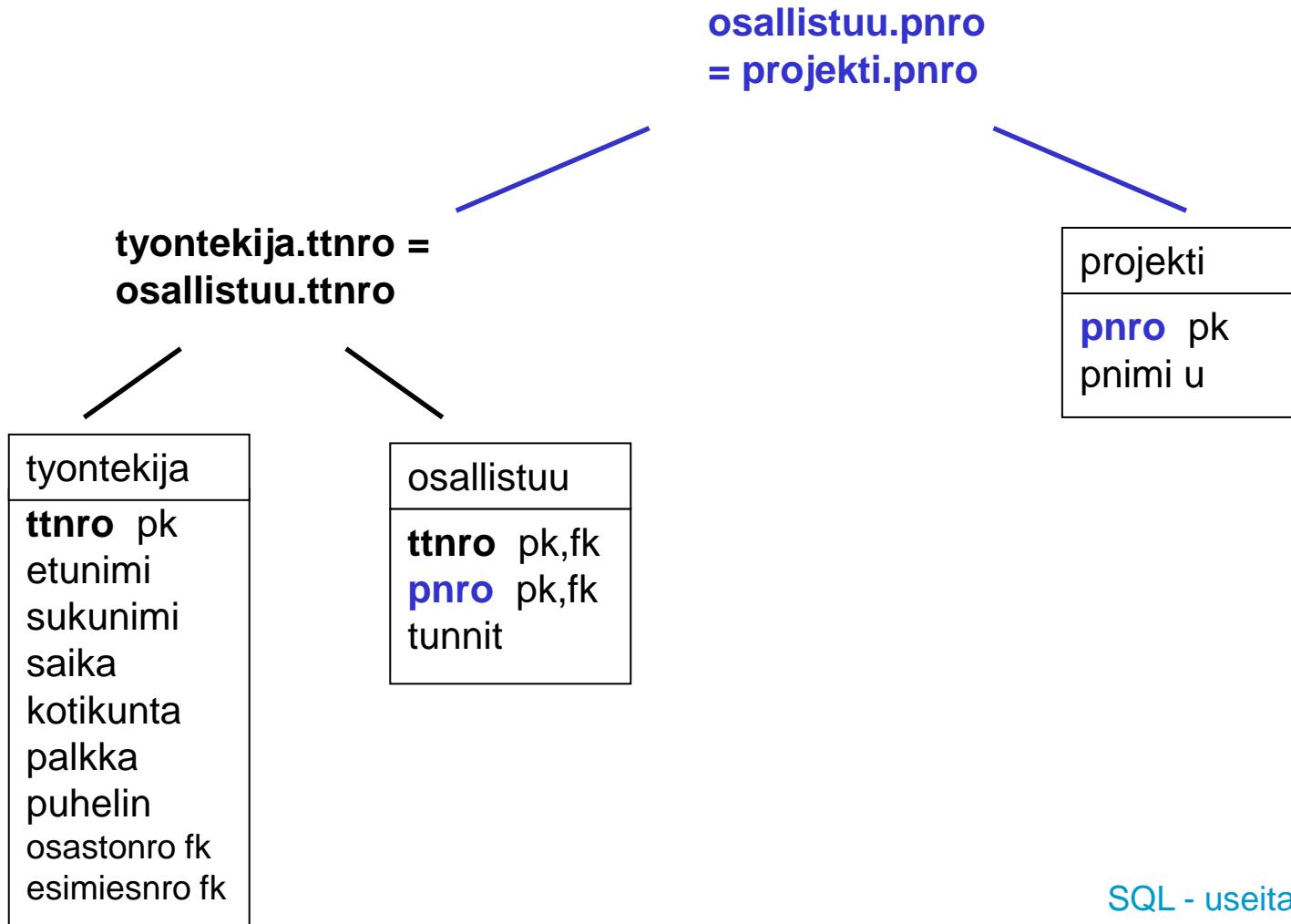
Kyselyt useista tauluista

Haetaan työntekijöiden etu- ja sukunimet, heidän projektiensa nimet ja projekteissa tehtävät tuntimääärät.



Kyselyt useista tauluista

Haetaan työntekijöiden etu- ja sukunimet, heidän projektiensa nimet ja projekteissa tehtävät tuntimäärit.



Kyselyt useista tauluista

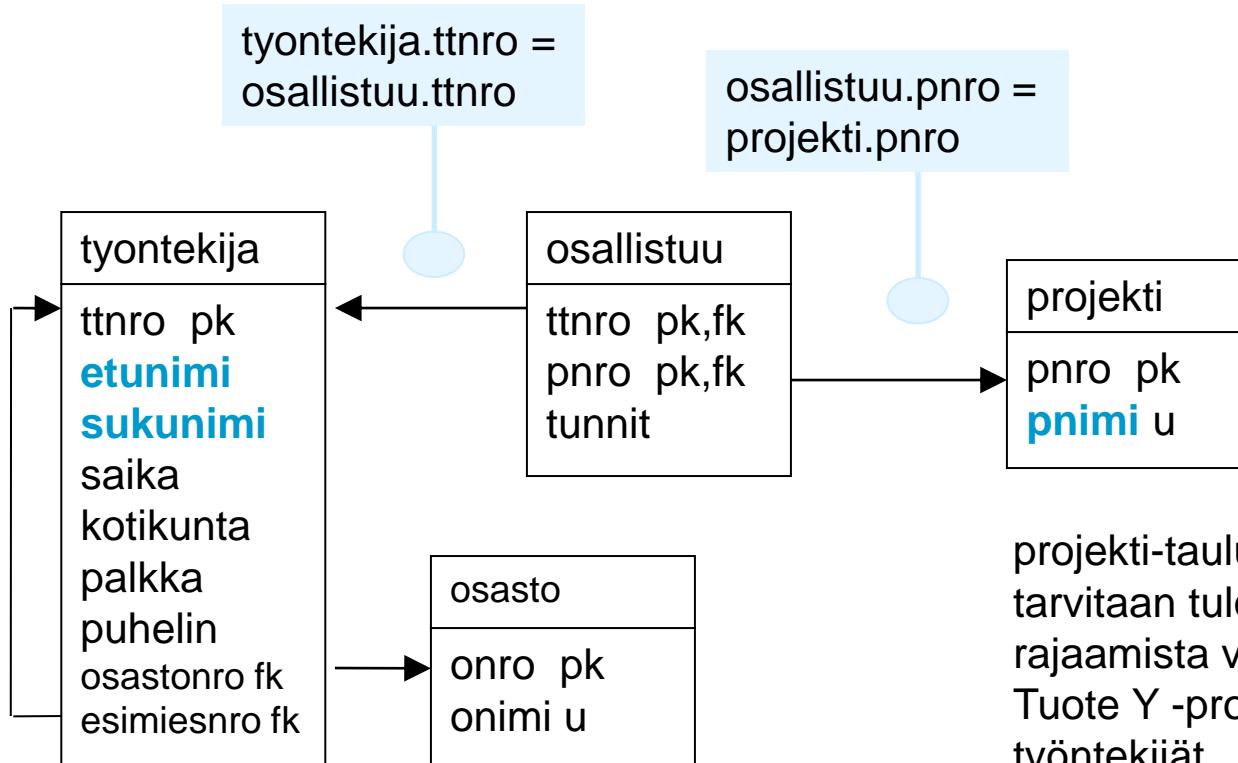
Tässä AS-sidesana on jätetty pois
annettaessa taululle uutta, tilapäistä
nimeä.

```
SELECT etunimi, sukunimi, pnimi, tunnit
FROM tyontekija tt, osallistuu o, projekti p
WHERE tt.ttnro = o.ttnro AND
      o.pnro = p.pnro;
```

etunimi	sukunimi	pnimi	tunnit
Pekka	Puro	Tuote X	32.5
Pekka	Puro	Tuote Y	7.5
Ville	Viima	Tuote Y	10.0
Ville	Viima	Tuote Z	10.0
Alli	Kivi	Tuote Z	30.0
Alli	Kivi	Tuote X	10.0
Jenni	Joki	Tuote Y	15.0

**Samannimiset sarakkeet on
erotettava toisistaan
käyttämällä taulun nimeä
tarkenteena:
taulunimi.sarakenimi**

Haetaan Tuote Y -projektiin osallistuvien työntekijöiden etu- ja sukunimet.



projekti-taulua
tarvitaan tulosrivien
rajaamista varten:
Tuote Y -projektiin
työntekijät

```

SELECT etunimi, sukunimi
FROM tyontekija tt, osallistuu o, projekt p
WHERE tt.ttnro = o.ttnro AND
      o.pnro = p.pnro AND
      pnimi = 'Tuote Y';
    
```

etunimi	sukunimi
Pekka	Puro
Ville	Viima
Jenni	Joki

Kyselyt useista tauluista: Liitosoperaatiot

- FROM-osassa käytettävät **liitosoperaatiot (JOIN)** yhdistävät kaksi taulua yhdeksi tauluksi.

```
SELECT sarake {, sarake}
  FROM taulu1
    liitostyyppi JOIN taulu2
    ON liitosehto
  [WHERE ehto]
```

- Liitosehto
 - millä perusteella kahden taulun rivejä liitetään toisiinsa
- Liitostyyppi
 - INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER
 - miten kohdellaan rivejä, joille liitosehto ei toteudu
- Huom. SQLitessa on toteutettu vain INNER JOIN ja LEFT OUTER JOIN, PostgreSQL:ssä kaikki liitostyypit.

Kyselyt useista tauluista: Liitosoperaatiot

- **INNER JOIN – liitos**
 - tuottaa samat rivit kuin karteesisen tulon ja WHERE-osan liitosehdon käyttäminen
- **OUTER JOIN - ulkoliitos**
 - **LEFT OUTER JOIN - vasen ulkoliitos**
 - liitoksen antamat rivit ja
 - myös ne vasemmanpuoleisen taulun rivit, joille liitosehdo ei toteudu.
 - **RIGHT OUTER JOIN - oikea ulkoliitos**
 - liitoksen antamat rivit ja
 - myös ne oikeanpuoleisen taulun rivit, joille liitosehdo ei toteudu.
 - **FULL OUTER JOIN - täysi ulkoliitos**
 - liitoksen antamat rivit ja
 - myös ne vasemmanpuoleisen taulun ja oikeanpuoleisen taulun rivit, joille liitosehdo ei toteudu.

Kyselyt useista tauluista: Liitosoperaatiot

taulu1

nimi	laji
Ressu	koira
Kaustinen	lintu
Wagner	sika

taulu2

nimi	sarjakuva
Kaustinen	Tenavat
Wagner	Viivi ja Wagner
Lassi	Lassi ja Leevi

```
SELECT *
FROM taulu1, taulu2
WHERE taulu1.nimi = taulu2.nimi;
```

```
SELECT *
FROM taulu1 INNER JOIN taulu2
ON taulu1.nimi = taulu2.nimi;
```

nimi	laji	nimi	sarjakuva
Kaustinen	lintu	Kaustinen	Tenavat
Wagner	sika	Wagner	Viivi ja Wagner

Kyselyt useista tauluista: Liitosoperaatiot

Haetaan Hallinto-osaston työntekijät.

```
SELECT sukunimi, etunimi, kotikunta  
FROM tyontekija INNER JOIN osasto  
    ON osastonro = onro  
WHERE onimi = 'Hallinto';
```

liitosoperaatio
liitosehto
valintaehdo

sukunimi	etunimi	kotikunta
Joki	Jenni	Lempäälä
Kivi	Alli	Nokia

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

```
FROM tyontekija INNER JOIN osasto
    ON osastonro = onro
```

ttnro	etunimi	sukunimi	kotikunta	osastonro	onro	onimi
88	Jukka	Susi	Tampere	1	1	Pääkonttori
33	Ville	Viima	Nokia	5	5	Tutkimus
12	Pekka	Puro	Tampere	5	5	Tutkimus
98	Jenni	Joki	Lempäälä	4	4	Hallinto
99	Alli	Kivi	Nokia	4	4	Hallinto

Kyselyt useista tauluista: Liitosoperaatiot

Haetaan työntekijöiden etu- ja sukunimet, heidän projektiensa nimet ja projekteissa tehtävät tuntimäärit.

Tehdään kysely liitosoperaatioiden avulla:

```
SELECT etunimi,sukunimi, pnimi, tunnit
FROM ((tyontekija tt
    INNER JOIN osallistuu o ON tt.ttnro = o.ttnro)
    INNER JOIN projekti p ON o.pnro = p.pnro);
```

```
SELECT etunimi,sukunimi, pnimi, tunnit
FROM tyontekija tt
    INNER JOIN osallistuu o ON tt.ttnro = o.ttnro
    INNER JOIN projekti p ON o.pnro = p.pnro;
```

etunimi	sukunimi	pnimi	tunnit
Pekka	Puro	Tuote X	32.5
Pekka	Puro	Tuote Y	7.5
Ville	Viima	Tuote Y	10.0
Ville	Viima	Tuote Z	10.0
Alli	Kivi	Tuote Z	30.0
Alli	Kivi	Tuote X	10.0
Jenni	Joki	Tuote Y	15.0

Liitosoperaatioita voidaan ketjuttaa: edetään sisimmistä sulusta uloimpin.

Sulut voi jättää pois. Liitokset tehdään annetussa järjestyksessä vasemmalta oikealle edeten.

tyontekija

ttnro	etunimi	...	esimiesnro
88	Jukka		
33	Ville		88
12	Pekka		33
98	Jenni		88
99	Alli		98

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

tyontekija tt INNER JOIN osallistuu o
ON tt.ttnro = o.ttnro

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia	33	2	10.0
33	Ville	Viima	Nokia	33	3	10.0
99	Alli	Kivi	Nokia	99	3	30.0
99	Alli	Kivi	Nokia	99	1	10.0
98	Jenni	Joki	Lempäälä	98	2	15.0

```

INNER JOIN
projekti p
ON o.pnro = p.pnro

```

tyontekija tt
INNER JOIN
osallistuu o
ON tt.ttnro = o.ttnro

projekti	
pnro	pnimi
1	Tuote X
2	Tuote Y
3	Tuote Z

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia	33	2	10.0
33	Ville	Viima	Nokia	33	3	10.0
99	Alli	Kivi	Nokia	99	3	30.0
99	Alli	Kivi	Nokia	99	1	10.0
98	Jenni	Joki	Lempäälä	98	2	15.0

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

```

FROM ((tyontekija tt
      INNER JOIN osallistuu o ON tt.ttnro = o.ttnro)
      INNER JOIN projekti p ON o.pnro = p.pnro)

```

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit	pnro	pnimi
12	Pekka	Puro	Tampere	12	1	32.5	1	Tuote X
12	Pekka	Puro	Tampere	12	2	7.5	2	Tuote Y
33	Ville	Viima	Nokia	33	2	10.0	2	Tuote Y
33	Ville	Viima	Nokia	33	3	10.0	3	Tuote Z
99	Alli	Kivi	Nokia	99	3	30.0	3	Tuote Z
99	Alli	Kivi	Nokia	99	1	10.0	1	Tuote X
98	Jenni	Joki	Lempäälä	98	2	15.0	2	Tuote Y

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

Kyselyt useista tauluista: Liitosoperaatiot

```
SELECT *
FROM taulu1 LEFT OUTER JOIN taulu2
    ON taulu1.nimi = taulu2.nimi;
```

nimi	laji	nimi	sarjakuva
Kaustinen	lintu	Kaustinen	Tenavat
Ressu	koira		
Wagner	sika	Wagner	Viivi ja Wagner

```
SELECT *
FROM taulu1 RIGHT OUTER JOIN taulu2
    ON taulu1.nimi = taulu2.nimi;
```

nimi	laji	nimi	sarjakuva
Kaustinen	lintu	Kaustinen	Tenavat
		Lassi	Lassi ja Leevi
Wagner	sika	Wagner	Viivi ja Wagner

Kyselyt useista tauluista: Liitosoperaatiot

```
SELECT *
FROM taulu1 FULL OUTER JOIN taulu2
    ON taulu1.nimi = taulu2.nimi;
```

nimi	laji	nimi	sarjakuva
Kaustinen	lintu	Kaustinen	Tenavat
		Lassi	Lassi ja Leevi
Ressu	koira		
Wagner	sika	Wagner	Viivi ja Wagner

- HUOM!
 - FROM-osassa käytettäviä liitosoperaatioita (JOIN) ei ole toteutettu kaikissa järjestelmissä ja niiden syntaksi voi vaihdella järjestelmittäin.
 - SQLItessa on toteutettu vain INNER JOIN ja LEFT OUTER JOIN, PostgreSQL:ssä kaikki liitostyyppit.

Kyselyt useista tauluista: Liitosoperaatiot

Haetaan kaikki työntekijät, heidän mahdolliset projektinsa ja niihin tehtävät tuntimäärit.

```
SELECT etunimi,sukunimi, pnimi, tunnit
FROM ((tyontekija tt
        LEFT OUTER JOIN osallistuu o ON tt.ttnro = o.ttnro)
        LEFT OUTER JOIN projekti p ON o.pnro = p.pnro);
```

etunimi	sukunimi	pnimi	tunnit
Pekka	Puro	Tuote X	32.5
Pekka	Puro	Tuote Y	7.5
Ville	Viima	Tuote Y	10.0
Ville	Viima	Tuote Z	10.0
Jukka	Susi		
Jenni	Joki	Tuote Y	15.0
Alli	Kivi	Tuote X	10.0
Alli	Kivi	Tuote Z	30.0

tyontekija

ttnro	etunimi	...	esimiesnro
88	Jukka		
33	Ville		88
12	Pekka		33
98	Jenni		88
99	Alli		98

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

tyontekija tt LEFT OUTER JOIN osallistuu o
ON tt.ttnro = o.ttnro

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia	33	2	10.0
33	Ville	Viima	Nokia	33	3	10.0
99	Alli	Kivi	Nokia	99	3	30.0
99	Alli	Kivi	Nokia	99	1	10.0
98	Jenni	Joki	Lempäälä	98	2	15.0
88	Jukka	Susi	Tampere			

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

SQL - useita tauluja 30

LEFT OUTER JOIN
 projekti p
 ON o.pnro = p.pnro

tyontekija tt
 LEFT OUTER JOIN
 osallistuu o
 ON tt.ttnro = o.ttnro

projekti	
pnro	pnimi
1	Tuote X
2	Tuote Y
3	Tuote Z

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia	33	2	10.0
33	Ville	Viima	Nokia	33	3	10.0
99	Alli	Kivi	Nokia	99	3	30.0
99	Alli	Kivi	Nokia	99	1	10.0
98	Jenni	Joki	Lempäälä	98	2	15.0
88	Jukka	Susi	Tampere			

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

```

FROM ((tyontekija tt
      LEFT OUTER JOIN osallistuu o ON tt.ttnro = o.ttnro)
      LEFT OUTER JOIN projekti p ON o.pnro = p.pnro)

```

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit	pnro	pnimi
12	Pekka	Puro	Tampere	12	1	32.5	1	Tuote X
12	Pekka	Puro	Tampere	12	2	7.5	2	Tuote Y
33	Ville	Viima	Nokia	33	2	10.0	2	Tuote Y
33	Ville	Viima	Nokia	33	3	10.0	3	Tuote Z
99	Alli	Kivi	Nokia	99	3	30.0	3	Tuote Z
99	Alli	Kivi	Nokia	99	1	10.0	1	Tuote X
98	Jenni	Joki	Lempäälä	98	2	15.0	2	Tuote Y
88	Jukka	Susi	Tampere					

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

Kyselyt useista tauluista: Liitosoperaatiot

Haetaan kaikki työntekijät, joiden kotikunta on Tampere, ja heidän mahdollisten projektiensa numerot.

```
SELECT tt.ttnro, sukunimi, etunimi, kotikunta, pnro AS projektinumero  
FROM tyontekija tt LEFT OUTER JOIN osallistuu o  
    ON tt.ttnro = o.ttnro AND kotikunta = 'Tampere';
```

Ei näin.

ttnro	sukunimi	etunimi	kotikunta	projektinumero
12	Puro	Pekka	Tampere	1
12	Puro	Pekka	Tampere	2
33	Viima	Ville	Nokia	
98	Joki	Jenni	Lempäälä	
99	Kivi	Alli	Nokia	
88	Susi	Jukka	Tampere	

```
SELECT tt.ttnro, sukunimi, etunimi, kotikunta, pnro AS projektinumero  
FROM tyontekija tt LEFT OUTER JOIN osallistuu o  
    ON tt.ttnro = o.ttnro  
WHERE kotikunta = 'Tampere';
```

**Vaan
näin.**

ttnro	sukunimi	etunimi	kotikunta	projektinumero
88	Susi	Jukka	Tampere	
12	Puro	Pekka	Tampere	1
12	Puro	Pekka	Tampere	2

Edellisen kalvon ylempi kysely, joka EI toimi halutulla tavalla

tyontekija

ttnro	etunimi	...	kotikunta	esimiesnro
88	Jukka		Tampere	
33	Ville		Nokia	88
12	Pekka		Tampere	33
98	Jenni		Lempäälä	88
99	Alli		Nokia	98

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

```
FROM tyontekija tt LEFT OUTER JOIN osallistuu o  
ON tt.ttnro = o.ttnro AND kotikunta = 'Tampere'
```

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia			
98	Jenni	Joki	Lempäälä			
99	Alli	Kivi	Nokia			
88	Jukka	Susi	Tampere			

Alempi kysely, joka toimii halutulla tavalla

tyontekija

ttnro	etunimi	...	kotikunta	esimiesnro
88	Jukka		Tampere	
33	Ville		Nokia	88
12	Pekka		Tampere	33
98	Jenni		Lempäälä	88
99	Alli		Nokia	98

osallistuu

ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

```
FROM tyontekija tt LEFT OUTER JOIN osallistuu o  
ON tt.ttnro = o.ttnro
```

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
33	Ville	Viima	Nokia	33	2	10.0
33	Ville	Viima	Nokia	33	3	10.0
99	Alli	Kivi	Nokia	99	3	30.0
99	Alli	Kivi	Nokia	99	1	10.0
98	Jenni	Joki	Lempäälä	98	2	15.0
88	Jukka	Susi	Tampere			

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

Alempi kysely, joka toimii halutulla tavalla

```
WHERE kotikunta = 'Tampere'
```

ttnro	etunimi	sukunimi	kotikunta	ttnro	pnro	tunnit
12	Pekka	Puro	Tampere	12	1	32.5
12	Pekka	Puro	Tampere	12	2	7.5
88	Jukka	Susi	Tampere			

(osa tyontekija-taulun sarakkeista jätetty pois tilan puutteen vuoksi)

Kyselyt useista tauluista: Kyselyn evaluointi

Vaihe 1. FROM-osa

- Jos FROM-osassa ei ole liitosoperaatioita,
 - muodostetaan annettujen **taulujen karteesinen tulo eli ristitulo**, joka on tämän vaiheen tulostaulu.
- Jos FROM-osassa on liitosoperaatio(ita),
 - muodostetaan liitoksen (karteesinen tulo ja liitoseheto) antama tulostaulu ja
 - lisätään tulostauluun mahdollisen ulkoliitoksen antamat "lisä rivit".
 - (Käydään läpi kaikki liitosoperaatiot vastaavalla tavalla.)

Vaihe 2. WHERE-osa

- Poistetaan edellisen vaiheen tulostaulusta ne rivit, jotka eivät täytä **liitos- ja/tai valintaehdoja**.

Vaihe 3. SELECT-osa

- Poistetaan edellisen vaiheen tulostaulusta sarakkeet, jotka eivät esiinny SELECT-listassa.

4. ORDER BY -osa

- Järjestetään rivit järjestystekijöiden (sarakkeiden) mukaisesti.

Kyselyt useista tauluista: Taulun liittäminen itseensä

- Kyselyssä voidaan liittää saman taulun rivejä toisiinsa antamalla sama taulu useaan kertaan FROM-osassa.
- Taululle on annettava tällöin FROM-osassa uusi, tilapäinen nimi.

Haetaan työntekijät ja heidän esimiehensä.

```
SELECT tt.etunimi, tt.sukunimi, em.sukunimi  
FROM tyontekija AS tt, tyontekija AS em  
WHERE tt.esimiesnro = em.ttnro;
```

Tulostaulun
sarakkeelle voidaan
antaa uusi nimi.

AS pomo

taululle uusi,
tilapäinen nimi

tt:n (työntekijän) sarakkeet

ttnro	etunimi	...	esimiesnro	ttnro	etunimi	...	esimiesnro
88	Jukka			88	Jukka		
88	Jukka			33	Ville		88
88	Jukka			12	Pekka		33
88	Jukka			98	Jenni		88
88	Jukka			99	Alli		98
33	Ville		88	88	Jukka		
33	Ville		88	33	Ville		88
33	Ville		88	12	Pekka		33
33	Ville		88	98	Jenni		88
33	Ville		88	99	Alli		98
12	Pekka		33	88	Jukka		
12	Pekka		33	33	Ville		88
12	Pekka		33	12	Pekka		33
12	Pekka		33	98	Jenni		88
12	Pekka		33	99	Alli		98
98	Jenni		88	88	Jukka		
98	Jenni		88	33	Ville		88
98	Jenni		88	12	Pekka		33
98	Jenni		88	98	Jenni		88
98	Jenni		88	99	Alli		98
99	Alli		98	88	Jukka		
99	Alli		98	33	Ville		88
99	Alli		98	12	Pekka		33
99	Alli		98	98	Jenni		88
99	Alli		98	99	Alli		98

em:n (esimiehen) sarakkeet

```
FROM tyontekija AS tt,
tyontekija AS em
WHERE tt.esimiesnro =
em.ttnro;
```

Kyselyt useista tauluista: Taulun liittäminen itseensä

```
SELECT tt.etunimi, tt.sukunimi, em.sukunimi AS pomo  
FROM tyontekija AS tt, tyontekija AS em  
WHERE tt.esimiesnro = em.ttnro;
```

etunimi	sukunimi	pomo
ville	viima	Susi
Pekka	Puro	viima
Jenni	Joki	Susi
Alli	Kivi	Joki

Kyselyt useista tauluista: Taulun liittäminen itseensä

```
SELECT em.etunimi AS esimies  
FROM tyontekija tt, tyontekija em  
WHERE tt.esimiesnro = em.ttnro AND  
tt.ttnro = 12;
```

Haetaan
työntekijän nro
12 lähin esimies.

```
esimies  
-----  
ville
```

```
SELECT eem.etunimi AS esimiehen_esimies  
FROM tyontekija tt, tyontekija em, tyontekija eem  
WHERE tt.esimiesnro = em.ttnro AND  
em.esimiesnro = eem.ttnro AND  
tt.ttnro = 12;
```

Haetaan
työntekijän nro
12 lähimän
esimiehen lähin
esimies

```
esimiehen_esimies  
-----  
Jukka
```

Edellisen kalvon alempi kysely

FROM tyontekija tt, tyontekija em, tyontekija eem

osa karteesisen tulon riveistä ja sarakkeista

tt:n (työntekijän) sarakkeet

em:n (esimiehen) sarakkeet

eem:n (esimiehen esimiehen)
sarakkeet

ttnro	etunimi	esimiesnro	ttnro	etunimi	esimiesnro	ttnro	etunimi	esimiesnro
12	Pekka	33	88	Jukka		88	Jukka	
12	Pekka	33	88	Jukka		33	Ville	88
12	Pekka	33	88	Jukka		12	Pekka	33
12	Pekka	33	88	Jukka		98	Jenni	88
12	Pekka	33	88	Jukka		99	Alli	98
12	Pekka	33	33	Ville	88	88	Jukka	
12	Pekka	33	33	Ville	88	33	Ville	88
12	Pekka	33	33	Ville	88	12	Pekka	33
12	Pekka	33	33	Ville	88	98	Jenni	88
12	Pekka	33	33	Ville	88	99	Alli	98
12	Pekka	33	12	Pekka	33	88	Jukka	
12	Pekka	33	12	Pekka	33	33	Ville	88
12	Pekka	33	12	Pekka	33	12	Pekka	33
12	Pekka	33	12	Pekka	33	98	Jenni	88
12	Pekka	33	12	Pekka	33	99	Alli	98

```
tt.esimiesnro = em.ttnro AND  
em.esimiesnro = eem.ttnro AND  
tt.ttnro = 12;
```

Kyselyt useista tauluista: Taulun liittäminen itseensä

```
SELECT tt.etunimi, tt.sukunimi, em.sukunimi AS pomo  
FROM tyontekija AS tt INNER JOIN tyontekija AS em  
    ON tt.esimiesnro = em.ttnro;
```

etunimi	sukunimi	pomo
Ville	Viima	Susi
Pekka	Puro	Viima
Jenni	Joki	Susi
Alli	Kivi	Joki

```
SELECT tt.etunimi, tt.sukunimi, em.sukunimi AS pomo  
FROM tyontekija AS tt LEFT OUTER JOIN tyontekija AS em  
    ON tt.esimiesnro = em.ttnro;
```

etunimi	sukunimi	pomo
Jukka	Susi	
Ville	Viima	Susi
Pekka	Puro	Viima
Jenni	Joki	Susi
Alli	Kivi	Joki

Haetaan kaikki työntekijät ja heidän mahdolliset esimiehensä.

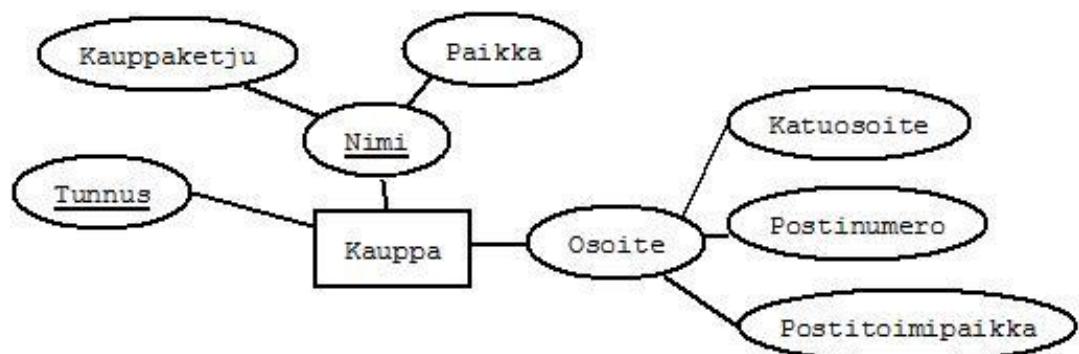
Tietokantojen perusteet

ER-kaavio ja muunnos SQL-tietokannan kaavioksi:

- Koottu attribuutti
- N-paikkaiset suhdetyypit
- Heikon entiteettityyppin hyödyntämisestä mallintamisessa
- Avaimista
- Johdetuista attribuuteista
- Johdetuista suhdetyypeistä
- Suunnitteluperiaatteita

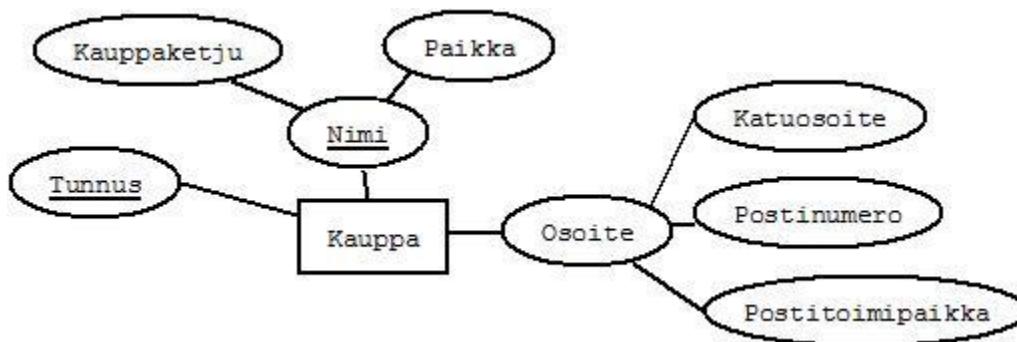
Koottu attribuutti

- Koottu (composite) attribuutti koostuu kahdesta tai useammasta yksinkertaisesta (simple, atomic) attribuutista.
 - Yksinkertaista attribuuttia käsitellään yhtenä kokonaisuutena, jota ei jaeta osiin.
 - Koottua attribuuttia voidaan käsitellä joko kokonaisuutena tai osina.
- Alla olevassa esimerkissä Kauppa-entiteettityypillä on kaksi koottua attribuuttia:
 - Nimi
 - Kauppaketju
 - Paikka
 - Osoite
 - Katuosoite
 - Postiosoite
 - Postitoimipaikka



ER_SQL: Koottu attribuutti

- Kun koottu attribuutti muunnetaan taulun sarakkeiksi, kootun attribuutin jokaiselle yksinkertaiselle attribuutille lisätään oma sarakke.



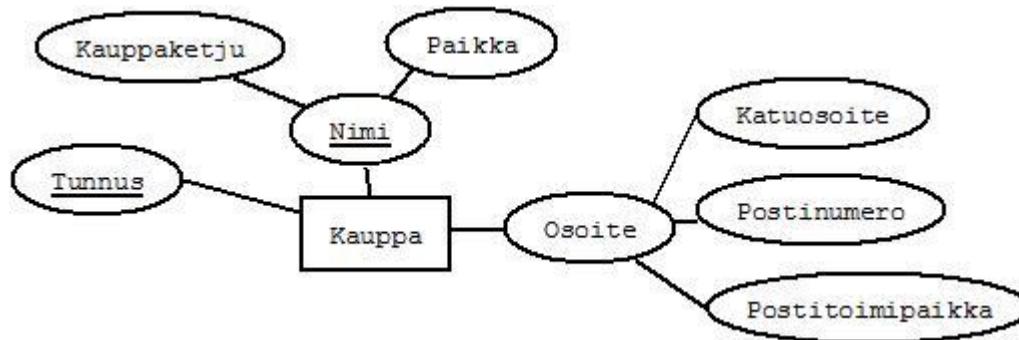
Kauppa-entiteettityypillä on kaksi tunnisteominaisuutta eli avainta. Näistä tunnus valitaan pääavaimeksi ja kauppaketju, paikka -yhdistelmästä tehdään avain.

kauppa
°tunnus PK
°kauppaketju U
°paikka U
°katuosoite
°postinumero
°postitoimipaikka

kauppa
°tunnus PK
°kauppaketju U1
°paikka U1
°katuosoite
°postinumero
°postitoimipaikka

Jos taulussa olisi useita avaimia, voitaisiin merkinnällä U#, missä # on numero, ilmaista eri avainten sarakkeet.

ER_SQL: Koottu attribuutti

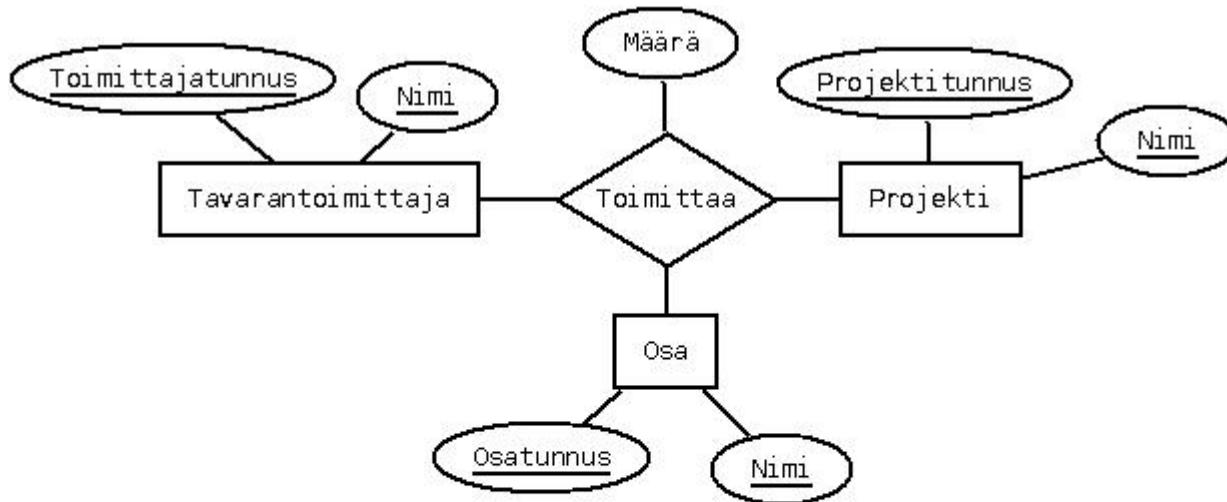


kauppa
°tunnus PK
°kauppaketju U
°paikka U
°katuosoite
°postinumero
°postitoimipaikka

kauppa

tunnus	kauppaketju	paikka	katuosoite	postinumero	postitoimipaikka
1	Ketju X	Kaleva	A-katu 22	33540	Tampere
2	Ketju Y	Kaleva	A-katu 22	33540	Tampere
3	Ketju Y	Lielahti	B-katu 1	33400	Tampere

N-paikkaiset suhdetyypit



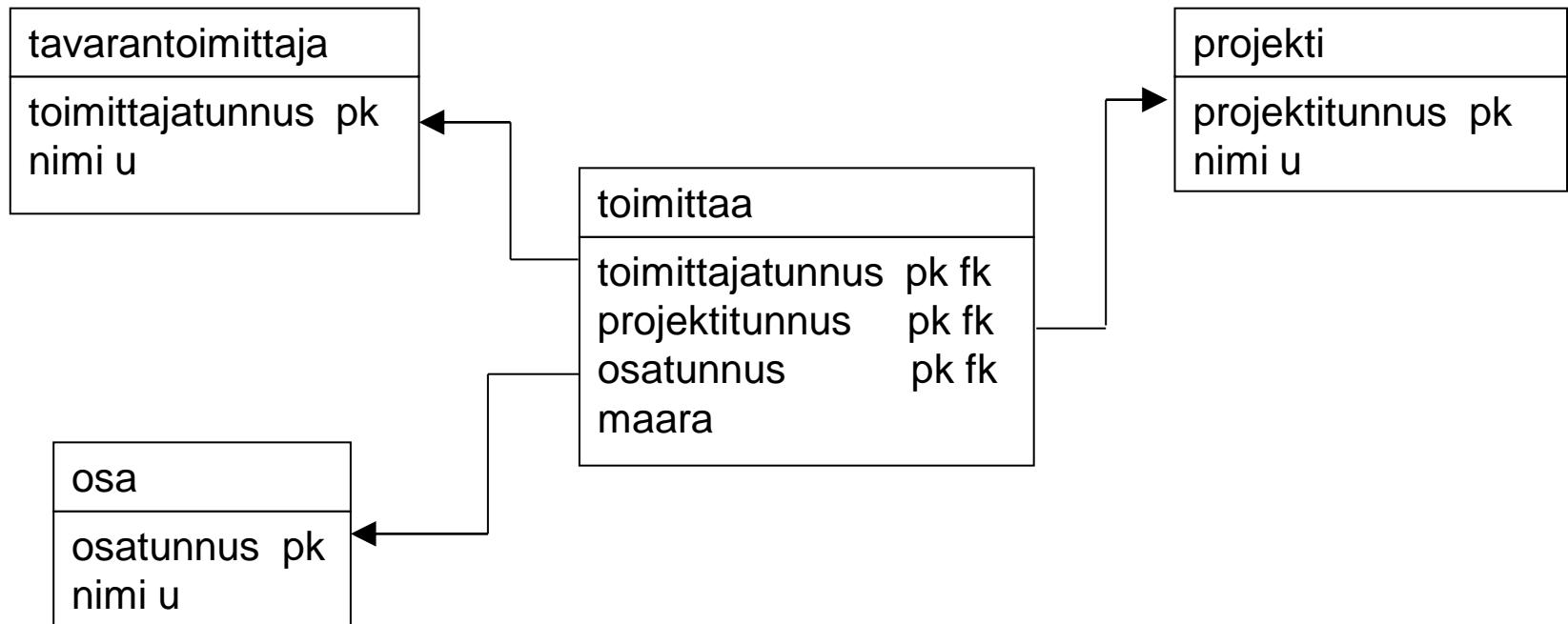
3-paikkainen (tertiääriinen) suhde: suhde yhdistää kolme entiteettiä toisiinsa.

Tietty tavarantoimittaja toimittaa tiettyä osaa tiettylle projektille

ER_SQL: N-paikkaiset suhdetyypit

- Luodaan oma taulu suhdetyyppiä varten (“suhdetaulu”).
- Taulun nimeksi tulee suhdetyyppin nimi.
- Tauluun sarakkeiksi
 - suhdetyyppiin liittyvien entiteettityyppien taulujen pääavainsarakkeet
 - joista tehdään viiteavaimet vastaaviin tauluihin
 - suhdetyypille mahdollisesti määritellyt attribuutit
- Suhdetaulun pääavaimeksi entiteettitauluilta saatujen sarakkeiden yhdistelmä.
 - PRIMARY KEY -määre

ER_SQL: N-paikkaiset suhdetyypit



ER → SQL: N-paikkaiset suhdetyypit

```
CREATE TABLE tavarantoimittaja(  
    toimittajatunnus INT,  
    nimi VARCHAR(30) NOT NULL,  
    PRIMARY KEY (toimittajatunnus),  
    UNIQUE (nimi));
```

```
CREATE TABLE osa(  
    osatunnus INT,  
    nimi VARCHAR(30) NOT NULL,  
    PRIMARY KEY (osatunnus),  
    UNIQUE (nimi));
```

```
CREATE TABLE toimittaa(  
    toimittajatunnus INT,  
    projektitunnus INT,  
    osatunnus INT,  
    maara INT NOT NULL,  
    PRIMARY KEY (toimittajatunnus, projektitunnus, osatunnus),  
    FOREIGN KEY (toimittajatunnus) REFERENCES tavarantoimittaja(toimittajatunnus),  
    FOREIGN KEY (projektitunnus) REFERENCES projekti(projektitunnus),  
    FOREIGN KEY (osatunnus) REFERENCES osa(osatunnus));
```

```
CREATE TABLE projekti(  
    projektitunnus INT,  
    nimi VARCHAR(30) NOT NULL,  
    PRIMARY KEY (projektitunnus),  
    UNIQUE (nimi));
```

ER → SQL: N-paikkaiset suhdetyypit

tavarantoimittaja

toimittajatunnus	nimi
1	Pulttipaja
2	Maken metalli
3	Terästakomo

osa

osatunnus	nimi
1	Pultti 1
2	Pultti 2
3	Pikkuruuvi
4	Jättiruuvi

projekti

projektitunnus	nimi
1	Projekti A
2	Projekti B

toimittaa

toimittajatunnus	projektitunnus	osatunnus	maara
1	1	1	500
2	1	1	2000
3	2	4	1000

Esimerkki: Miksi tarvitaan 3-asteinen suhdetyyppi?

- On rekkoja, tuotteita ja kauppoja sekä 3-asteinen suhdetyyppi toimittaa.
 - Esimerkiksi rekka 1 toimittaa tuotetta 1 kaupalle 1 (r_1, t_1, k_1)

3-asteinen

(r_1, t_1, k_1)
 (r_2, t_1, k_2)
 (r_2, t_2, k_1)

2-asteinen

(r_1, t_1) (r_2, t_1)
 (r_2, t_2)
 (r_1, k_1) (r_2, k_1)
 (r_2, k_2)

(t_1, k_1) (t_2, k_1)
 (t_1, k_2)

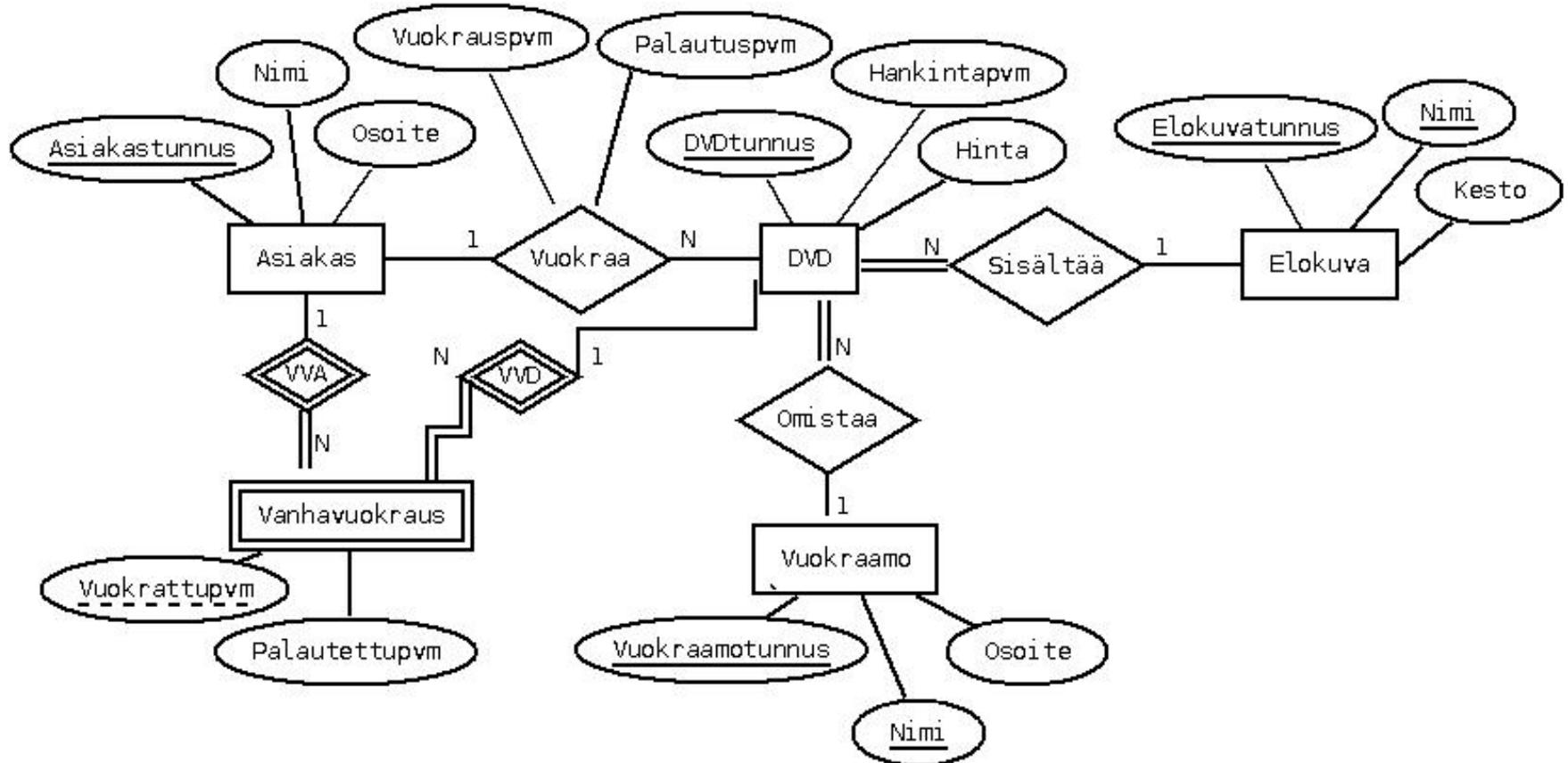
Binäärisuhteiden perusteella voitaisiin virheellisesti päätellä, että rekka 2 toimittaa tuotetta 1 kaupalle 1

$(r_2, t_1), (r_2, k_1), (t_1, k_1)$

Näin ei kuitenkaan ole – tarvitaan siis 3-asteinen suhde.

Heikon entiteettityypin hyödyntäminen mallintamisessa

- Suhdetyypeillä ei ole avainattribuutteja eikä suhdetyypin attribuutteja käytetä suhteiden tunnistamisessa.
 - Tietty suhde yksilöidään siihen osallistuvien entiteettien perusteella.
- Tietokantaan voidaan haluta tallentaa esim. "historiatietoja", esim.
 - kirjastokirjojen lainaus
 - kurssien suorittaminen (kurssiarvosanan korottaminen)
 - DVD-levyjen vuokraus
- Historiatietojen mallintamisessa voidaan käyttää heikkoa entiteettityyppiä:

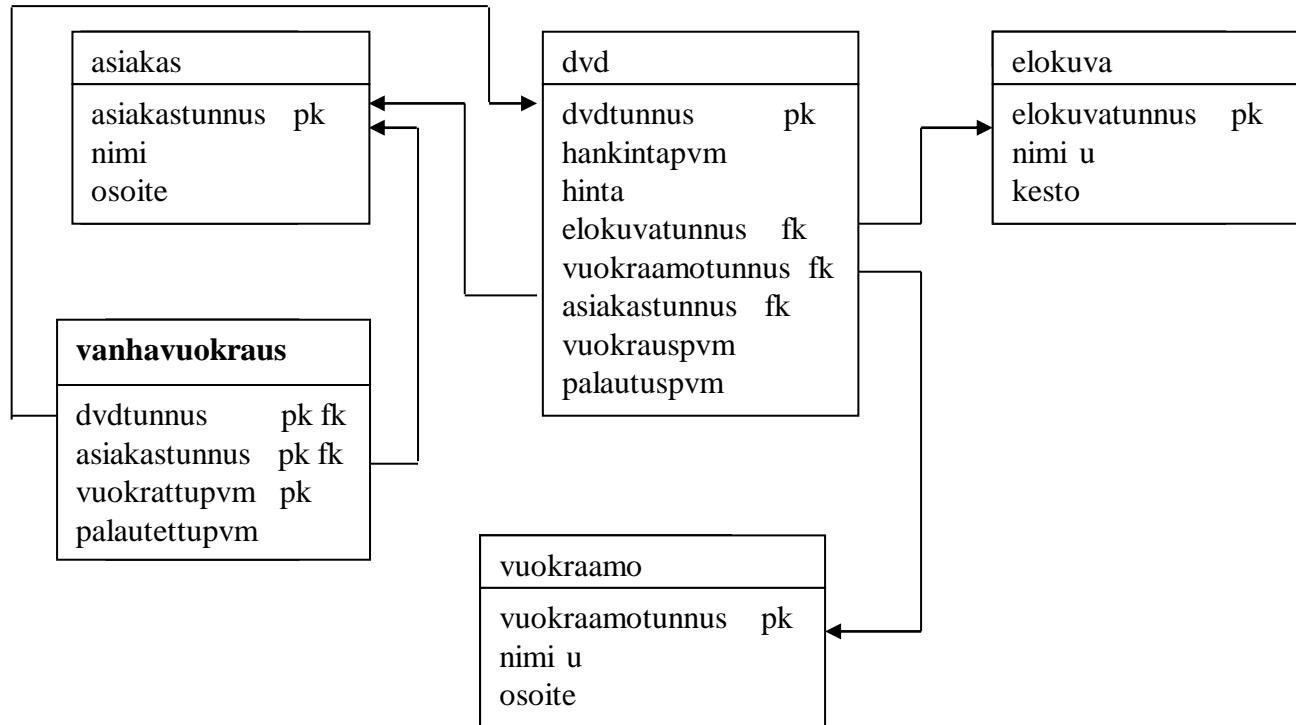


Mallinnetaan "vanhat" vuokraamiset heikkona entiteettityyppinä, jonka osittaisena avaimena on vuokrauspäivämäärä.

Entiteettityypeille, suhdetyypeille ja attribuuteille pitäisi (yleensä) löytyä luontevat, kuvaavat nimet. Suhdetyypin voi tarvittaessa nimetä myös siihen osallistuvien entiteettityyppien nimien yhdistelmällä.

- yllä olevassa kaaviossa VVA- ja VVD-nimet lyhennetty entiteettityyppien nimien yhdistelmistä.

Heikon entiteettityypin hyödyntäminen mallintamisessa



Avaimet

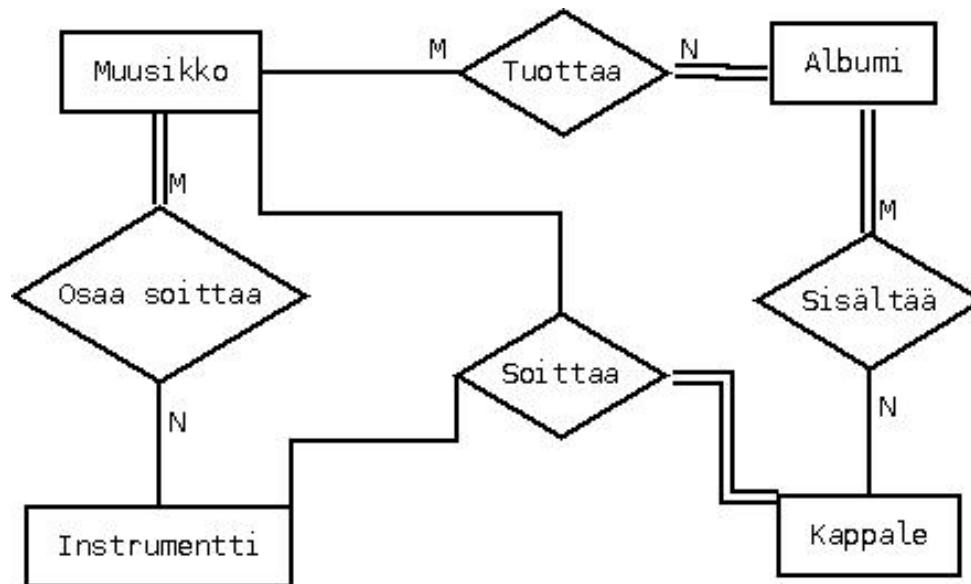
- Hyvä pääavain on attribuutti tai attribuuttiyhdistelmä, jonka arvo ei muutu (tai muuttuu hyvin harvoin).
 - Pääavaimen arvon muutos voi johtaa suureen määrään päivityksiä
- Joskus "luonnollinen" attribuutti on arvoltaan muuttumaton (tai harvoin muuttuva), esim.
 - henkilötunnus
 - valmistus- tai sarjanumero (esim. autot, kännykät)
- Usein käytetään keinotekoisia pääavaimia. Monesti nämä ovat kokonaislukutyyppisiä.
 - Esimerkiksi työntekijännumero pääavain, henkilötunnus avain
- ER-kaavion on vastattava SQL-kannan kaaviota. Siksi ER-kaavioon laitetaan mukaan myös keinotekoiset avaimet.

Johdetut attribuutit

- Johdetut attribuutit jätetään yleensä pois tietokannasta, koska niiden arvot voidaan johtaa tietokantaan tallennettavien tietojen perusteella.
 - esim. osaston työntekijöiden lukumäärä
- Joskus johdettu attribuutti halutaan tietokannan tauluun mukaan tehokkuus- ja/tai käyttömukavuussyyistä. (Ei tämän kurssin asioita)

Johdettavissa olevat suhdetyypit

- Johdettavissa olevat suhdetyypit jätetään (yleensä) ER-kaaviosta pois.



Muusikon ja albumin välille ei tarvita albumilla esiintymisestä kertovaa suhdetyyppiä, koska yhteys on johdettavissa polusta Muusikko – Soittaa – Kappale – Sisältää – Albumi

Osaa soittaa - ja Soittaa -suhdetyypit ovat molemmat tarpeellisia: Osaa soittaa -suhdetyyppi ilmaisee, mitä instrumentteja muusikko osaa soittaa. Soittaa-suhdetyyppi ilmaisee, mitä instrumentteja muusikko soittaa milläkin kappaleella.

Suunnitteluperiaatteita

- Tietokannan suunnittelun tärkeimpiä periaatteita ovat toistuvan tiedon välttäminen ja puuttuvien arvojen välttäminen.
 - Tiedon toistumiseen l. redundanssiin liittyy seuraavia ongelmia:
 - toistuva tieto vie tilaa turhaan
 - (Toistuva tietoa voidaan tosin sisällyttää tietokantaan tarkoituksellisesti tehokkuussyyistä. Ei tämä kurssin asioita.)
 - toistuvan tiedon ylläpitäminen kuluttaa turhaan resursseja ja on virhealtista
 - Puuttuihin arvoihin liittyy mm. seuraavia ongelmia:
 - mahdollinen tilan tuhlaaminen
 - puuttuvat arvot osattava huomioida kyselyissä
 - taulun rivien merkityksien tulkinta voi vaikeutua

Suunnitteluperiaatteita

- Kaikki tietokannan tiedot voitaisiin tallentaa yhteen tauluun, mutta tällainen taulu olisi ongelmallinen tietojen redundanssin ja puuttuvien arvojen vuoksi
- Pyritään suunnittelemaan "sopivan" kokoisia tauluja ja välttämään redundantteja tietoja ja puuttuvia arvoja.
- Kun ER-kaavio tehdään ja muunnetaan SQL-tietokannaksi annettujen periaatteiden mukaisesti, pitäisi tuloksena olla tietokanta, jossa **ei ole turhaa redundanssia eikä turhia puuttuvia arvoja**.
 - Joskus tietokantaan kuitenkin halutaan **hallittua redundanssia** tehokkuuden ja/tai käyttömukavuuden vuoksi.
 - Tällöin on osattava valvoa, että tietojen johdonmukaisuus säilyy ts. ettei kantaan voi tallentaa ristiriitaista tietoa.
 - (Ei tämän kurssin asioita.)



Tietokantojen perusteet

ER-mallinnus - perusteita

Tietojärjestelmän ja tietokannan suunnittelu
Käsiteellinen mallintaminen
ER-mallinnus

Tietojärjestelmän ja tietokannan suunnittelu

- Tietojärjestelmiä ja tietokantoja suunniteltaessa kartoitetaan ja analysoidaan, minkälaisia tietoja ja tietotarpeita sovellusalueella on.
 - Mistä asioista tarvitaan tietoa?
 - Miten asiat liittyvät toisiinsa? Mitä yhteyksiä asioiden välillä on?
 - Mitä tietoja asioista ja niiden välisistä yhteyksistä tarvitaan?
 - Millä tavoin tietoja jatkossa käytetään?
- Suunnittelija mallintaa sovellusalueutta sovellusalueen asiantuntijoiden antamien tietojen perusteella.
 - Sovellusalueella käytettävä ja sovellusalueutta kuvaava materiaali, asiantuntijoiden haastattelut, havainnointi

Tietojärjestelmän ja tietokannan suunnittelu

- Tietokannan tietosisällön suunnittelussa käytetään yleensä jotakin **käsitteellistä mallia**, jossa sovellusalueen tiedot kuvataan loppukäyttäjien (sovellusalueen asiantuntijoiden) käsitteitä käyttäen.
 - Puhutaan **käsitteellisestä suunnittelusta ja käsitteellisestä mallintamisesta**.
 - Tällä kurssilla käytetään ER-mallia käsitteelliseen mallintamiseen.
- Käsitteellisen suunnittelun jälkeen on vuorossa **rakenne- eli toteutustason suunnittelu (looginen suunnittelu)**.
 - Suunnitellaan esimerkiksi, mitä tietoja sijoitetaan mihinkin tauluihin.
- Viimeisenä vaiheena on vuorossa **fyysisen tason suunnittelu**.
 - Suunnitellaan esimerkiksi, mitä hakemistorakenteita käytetään hakujen nopeuttamiseksi.

Tietojärjestelmän ja tietokannan suunnittelu

- Tietokannan tietoja tarkastellaan siis eri abstraktiotasoilla erilaisia tietomalleja käyttäen.
 - **käsitetaso** (conceptual level)
 - Mitä tietoja käsitellään? Miten tiedot liittyvät toisiinsa?
 - esim. ER-malli
 - sisällön suunnittelu
 - **rakennetaso, toteutustaso** (structural level, implementation level)
 - Minkälaisina rakenteina tietoja käsitellään?
 - esim. SQL-tietokantojen taustalla oleva relatiomalli
 - SQL-komentotulkkin kautta tehtävät kyselyt ja päivitykset, ohjelointi
 - **fyysinen taso** (physical level), talletustaso
 - Minkälaisina rakenteina tiedot talletetaan levylle?
 - tietokannanhallintajärjestelmän toimittajan käyttämät käsitteet
 - esim. hakuja nopeuttavat hakemistorakenteet

Tietojärjestelmän ja tietokannan suunnittelu

- Eri tasolla olevien tietomallien avulla pystytään kuvaamaan käyttötarkoituksen kannalta olennaiset asiat.
- Tietomalli on käsitteistö, jolla kuvataan tietojen rakenne.
 - Tietojen tyyppi
 - Merkkijono, kokonaisluku, desimaaliluku, jne.
 - Tietojen väliset suhteet
 - Miten tiedot liittyvät toisiinsa
 - Tietoja koskevat rajoitteet
 - Mitkä ovat sallittuja arvoja, onko tieto pakollinen, jne.

Lisäksi tietomallit voivat sisältää operaatioita tietojen hakuun ja päivitykseen.

Tietojärjestelmän ja tietokannan suunnittelu

- Käsitteellisen, loogisen ja fyysisen suunnitellun lisäksi tarvitaan funktionaalista eli toiminnallista suunnittelua.
 - Minkälaisia sovellusohjelmia tarvitaan?
 - Minkälaisia tietokantatapahtumia sovellusohjelmissa tarvitaan?
 - Tapahtuma voi koostua esim. tietojen hausta ja päivityksestä.
 - Tietokantatapahtumiin liittyviä asioita opiskellaan TIETA7 Tietokantaohjelmointi -opintojaksoilla.
 - Minkälaisia SELECT-, INSERT-, UPDATE- ja DELETE-lauseita tarvitaan?

ER-malli

- ER-malli (entity-relationship model) on käsitteellisen tason tietomalli.
 - alunperin Peter Chenin vuonna 1976 esittämä
 - artikkelissa The entity-relationship model – toward a unified view of data, ACM Transactions on Database Systems (artikkeli saatavilla kurssimateriaalissa)
- Malli on yleisesti käytössä tietokannan suunnittelussa ja tietojärjestelmien suunnittelussa.
- ER-mallissa sovellusalueen tiedot kuvataan
 - entiteettinä (yksilönä) (entity),
 - entiteettien välisinä suhteina (yhteyksinä) (relationship) ja
 - edellisiin liittyvinä ominaisuuksina (attribute).
- ER-mallin perusesitysmuoto on graafinen.
 - Piirretään ER-kaavioita.

ER-malli

- ER-mallista on olemassa useita versioita. Peruskäsitteet ovat yleensä yhteiä eri versioille, mutta esitystavassa I. notaatiossa käytettävät merkinnät ja niiden merkitys vaihtelevat.
- Näissä kalvoissa käytetään kirjan Elmasri, Navathe: Fundamentals of Database Systems esitystapaa, joka vastaa suurelta osin Peter Chenin v. 1976 ehdottamaa esitystapaa.
- Kalvoilla käydään läpi ER-mallin peruskäsitteet ja esitystapa käyttämällä esimerkkisovellusalueena kuvitteellista yritystä, jonka tietojärjestelmän tietokantaan tarvitaan tietoja yrityksen työntekijöistä ja heidän huollettavistaan, yrityksen osastoista ja projekteista sekä näiden välisistä yhteyksistä. (Ks. sovellusalueen kuvaus erillisestä dokumentista.)

Entiteetti ja attribuutti

- **Entiteetti** (yksilö, kohde, tavallinen entiteetti tai vahva entiteetti, entity) on sovellusalueella olemassaoleva yksilö (olio), joka on erotettavissa muista sovellusalueen yksilöistä.
- Entiteetti voi olla konkreettinen tai abstrakti.
- Kullakin entiteetillä on **attribuutteja eli ominaisuuksia**, jotka kuvaavat sitä.
 - Esim. työntekijällä työntekijänumero, etunimi, sukunimi, syntymääika, jne.
- Kullakin entiteetillä on attribuuteille omat arvonsa.

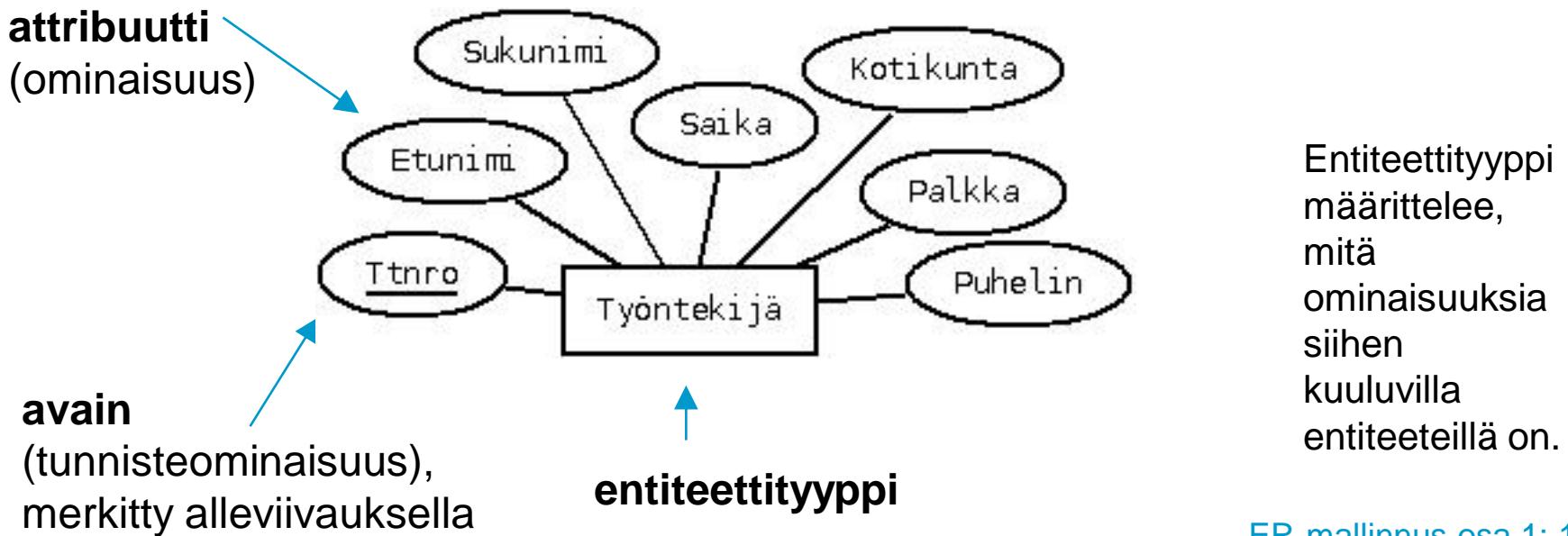


työntekijä-entiteettejä

(88, Jukka, Susi, 1957-11-10 , Tampere, 5500.00 , 444 1234)
(33, Ville, Viima, 1975-12-08 , Nokia, 4000.50 , 444 4343)
(12, Pekka, Puro, 1985-01-09, Tampere, 3000.00,)
(98, Jenni, Joki, 1961-06-20, Lempäälä, 4300.00, 444 4488)

Entiteettityyppi

- Samankaltaiset entiteetit eli entiteetit, joilla on samat ominaisuudet, kuuluvat samaan **entiteettityyppiin** (kohdetyyppiin, yksilötyyppiin, entity type).
- Entiteettityypit kuvataan suorakaiteina, attribuutit ovaaleina.



Käsite ja ilmentymä

- ER-mallissa tehdään ero käsitteiden ja niiden ilmentymien välillä.
- On olemassa yksi käsite (esim. työntekijä) ja mahdollisesti lukuisia (tai ei yhtään) sen ilmentymiä.

käsite
työntekijä



henkilö Jenni Joki –
eräs työntekijä-
käsitteen ilmentymä



(98, Jenni, Joki, 1961-06-20, Lempäälä, 4300.00, 444 4488)

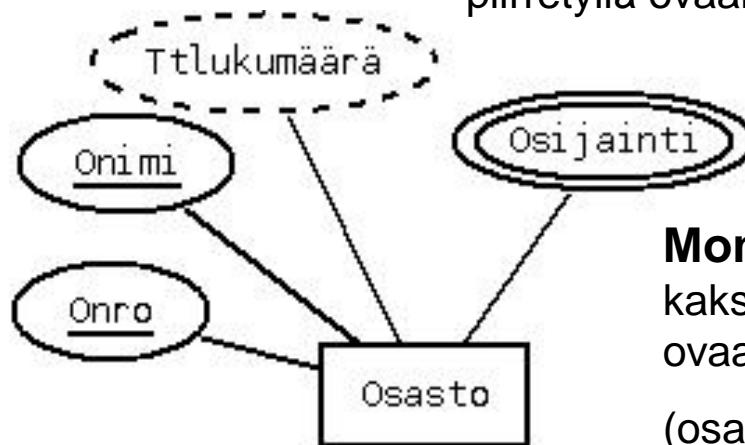
Entiteetti ja attribuutti

- Osa attribuuteista on **avaimia** eli **tunnisteominaisuksia**.
 - Avaimen arvot ovat toisistaan poikkevia ts. kullakin entiteetillä on oma, uniikki arvonsa avaimelle
 - Käytetään entiteettien tunnistamiseen.
- Avain voi muodostua **yhdestä attribuutista** tai **usean attribuutin yhdistelmästä**. Avaimeen otetaan mukaan minimimäärä attribuutteja, jotka riittävät tekemään siitä yksilöivän.
- Avaimen yksilöintikyvyn tulee olla voimassa sovellusalueen kaikissa mahdollisissa samantyyppisten entiteettien joukoissa ts. kaikissa mahdollisissa entiteettityypin ilmentymien joukoissa (ekstensioissa).
- Entiteettityypillä voi olla useita avaimia.

Attribuutit

- Attribuutti eli ominaisuus voi olla
 - yksiarvoinen – moniarvoinen (syntymääika – kielitaito)
 - tallennettava - johdettavissa oleva (syntymääika – ikä)
 - pysyvä – muuttuva (syntymääika – osoite)
 - pakollinen – valinnainen (henkilötunnus – puhelinnumero)

Yksiarvoinen tallennettava attribuutti merkitään yksinkertaisella yhtenäisellä viivalla piirretyllä ovaalilla.



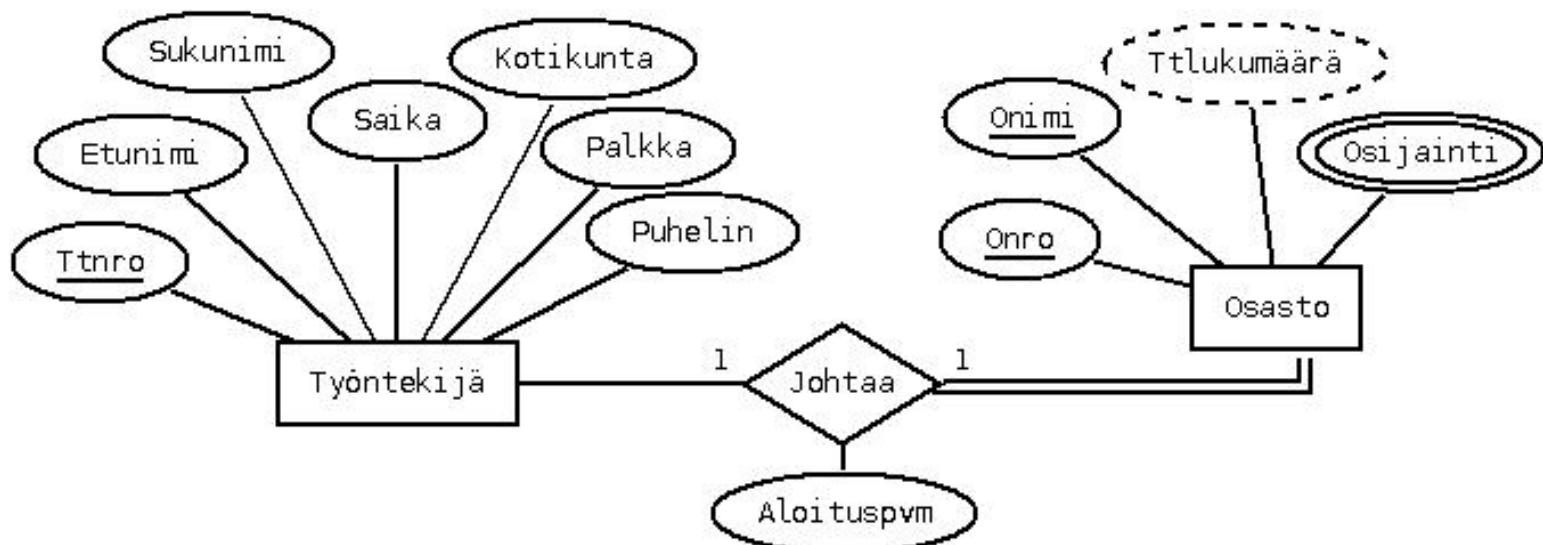
Johdettu attribuutti merkitään katkoviivalla piirretyllä ovaalilla (työntekijöiden lukumäärä)

Moniarvoinen attribuutti merkitään kaksinkertaisella viivalla piirretyllä ovaalilla.
(osastolla voi olla useita sijaintipaikkoja, toimipisteitä)

Kaksi avainta (osastot pystytään yksilöimään sekä nimen että numeron perusteella)

Suhde

- Sovellusalueen entiteettien välillä on yhteyksiä eli suhteita (relationships).
 - Esim. Jenni Joki työskentelee Hallinto-osastolla sekä johtaa sitä.
- Samankaltaiset suhteet voidaan koota **suhdetyyppiksi** (relationship type), jolla voi olla attribuutteja.
- Suhdetyypit merkitään vinoneliöillä.

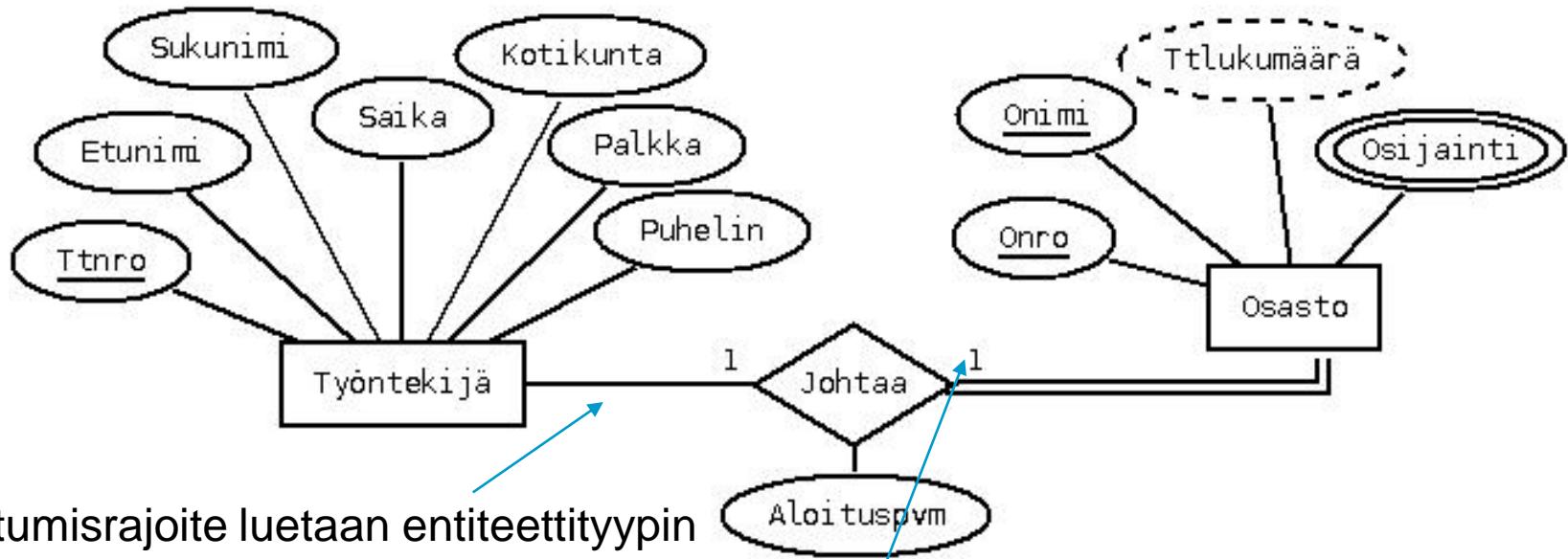


Suhdetyyppi: Rakenteelliset rajoitteet

- Sovellusalueella vallitsee suhteita koskevia rajoitteita, jotka määräävät, millä tavoin entiteetit voivat osallistua suhteisiin.
- Puhutaan **rakenteellisista rajoitteista**, joita ovat **Iukumäääräsuhteet ja osallistumisrajoitteet**
- **Lukumäääräsuhteet** (kardinaliteetit)
 - **Kuinka moneen** tietyn suhdetyypin **suhteeseen** entiteetti voi **maksimissaan** osallistua?
 - Merkitään numeroin ja kirjaimin
 - **1:1**, yhden suhde yhteen
 - **1:N**, yhden suhde moneen
 - **M:N**, monen suhde moneen
 - **Luetaan vastakkaiselta puolelta**, suhdetyypin "jälkeen" tulevasta numerosta tai kirjaimesta

Suhdetyyppi: Rakenteelliset rajoitteet

- **Osallistumisrajoitteet** (totaliteetit)
 - Onko entiteetin pakko osallistua tietyn suhdetyypin suhteeseen?
 - Merkitään viivoilla
 - **Osittainen osallistuminen** – yksinkertainen viiva _____
Entiteetin ei ole pakko osallistua tietyn suhdetyypin suhteeseen.
 - **Täydellinen osallistuminen** – kaksinkertainen viiva _____
Entiteetin on osallistuttava vähintään yhteen tietyn suhdetyypin suhteeseen. Täydellistä osallistumista kutsutaan myös **eksistentssiriippuvuudeksi**.
 - Osallistumisrajoite (eli entiteetin tietyn suhdetyypin suhteiden minimimäärä) luetaan **entiteettityypistä lähtevästä viivasta**.

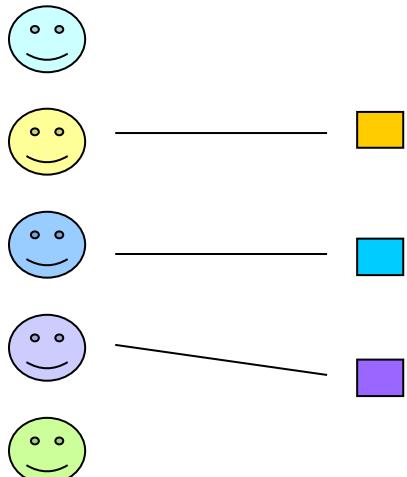


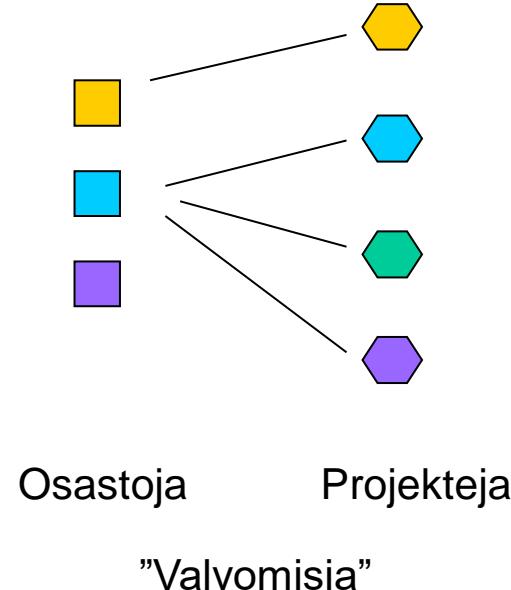
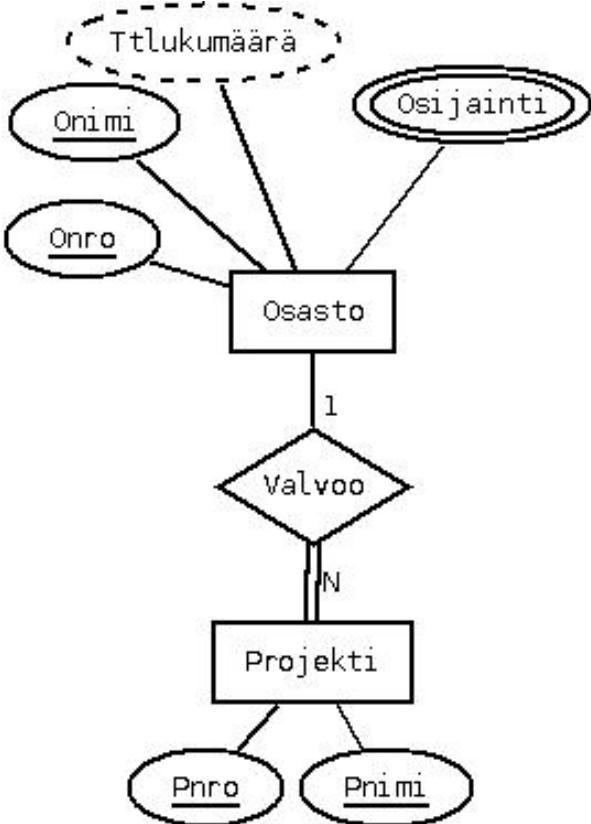
Osallistumisrajoite luetaan entiteettityypin itsensä puolelta: Työntekijöiden osallistuminen on **osittaista**, vain osa työntekijöistä johtaa osastoa (yksinkertainen viiva)

Lukumäärä luetaan entiteettityypin vastakkaiselta puolelta: työntekijä voi johtaa maksimissaan yhtä osastoa. (1)

Osastojen osallistuminen on **täydellistä**: jokaisella osastolla on oltava johtaja (kaksinkertainen viiva). Osastoa johtaa maksimissaan yksi työntekijä.

Työntekijä:Osasto, **Iukumääräsuhde 1:1**





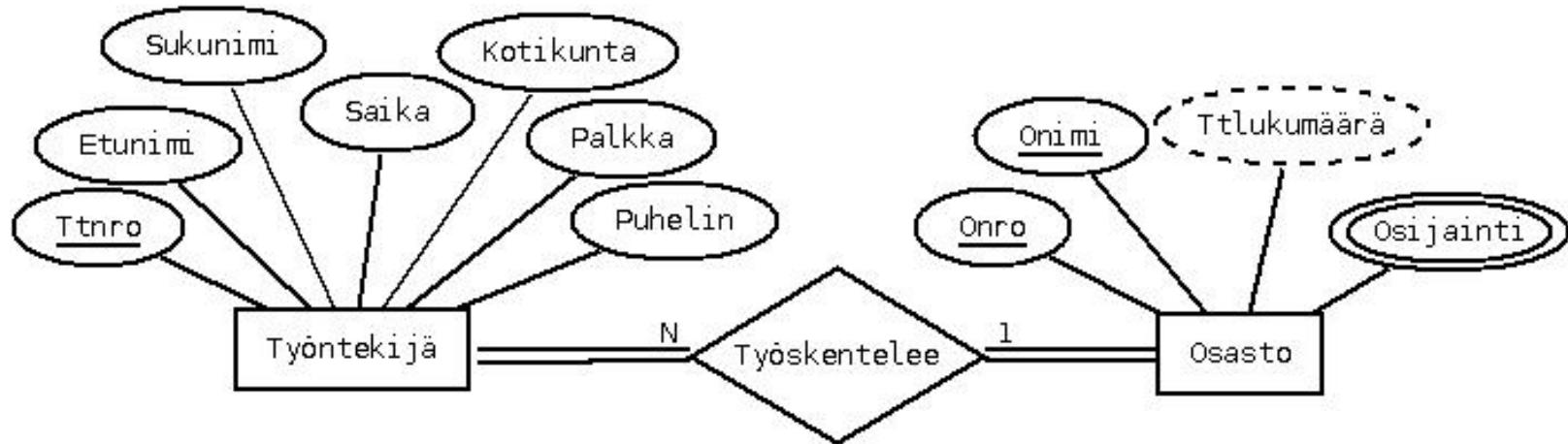
Osasto voi valvoa useaa projektia, projektia valvoo yksi osasto.

Osasto:Projekti, lukumääräsuhde 1:N

(Huom. ER-kaavion lukusuunta on tyypillisesti ylhäältä alas päin, vasemmalta oikealle.)

Osasto ei väittämättä valvo ainuttakaan projektia. Osaston osallistuminen on osittaista.

Projektia valvoo vähintään yksi osasto. Projektin osallistuminen on täydellistä.

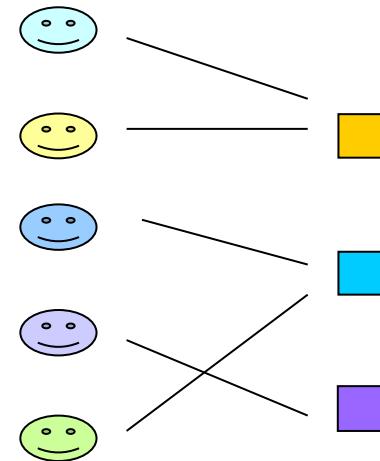


Työntekijä työskentelee enintään yhdellä osastolla. Osastolla voi työskennellä useita työntekijöitä.

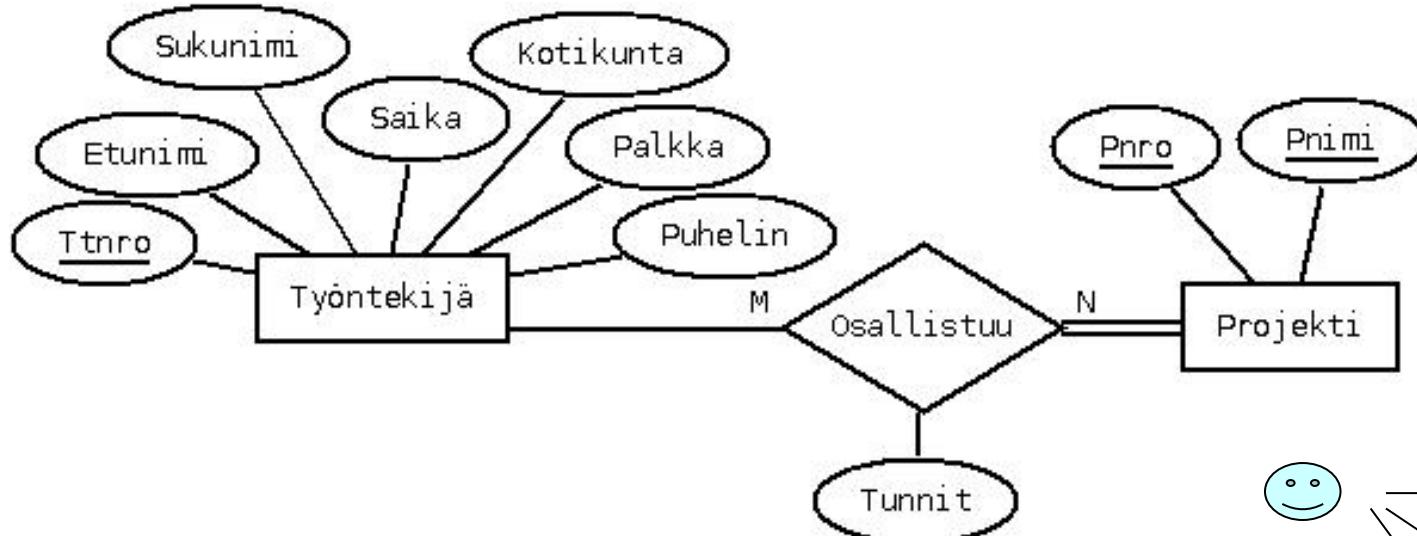
Työntekijä:Osasto, lukumääräsuhde N:1

Jokainen työntekijä työskentelee vähintään yhdellä osastolla ja jokaisella osastolla työskentelee vähintään yksi työntekijä.

Sekä työntekijän että osaston osallistuminen on täydellistä.



"Työskentelyjä"
Työntekijöitä Osastoja

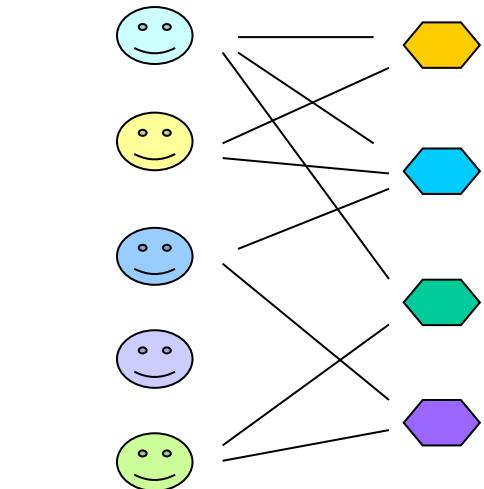


Työntekijä voi osallistua moneen projektiin.
 Projektiin voi osallistua monta työntekijää.

Työntekijä:Projekti **Iukumääräsuhde M:N**

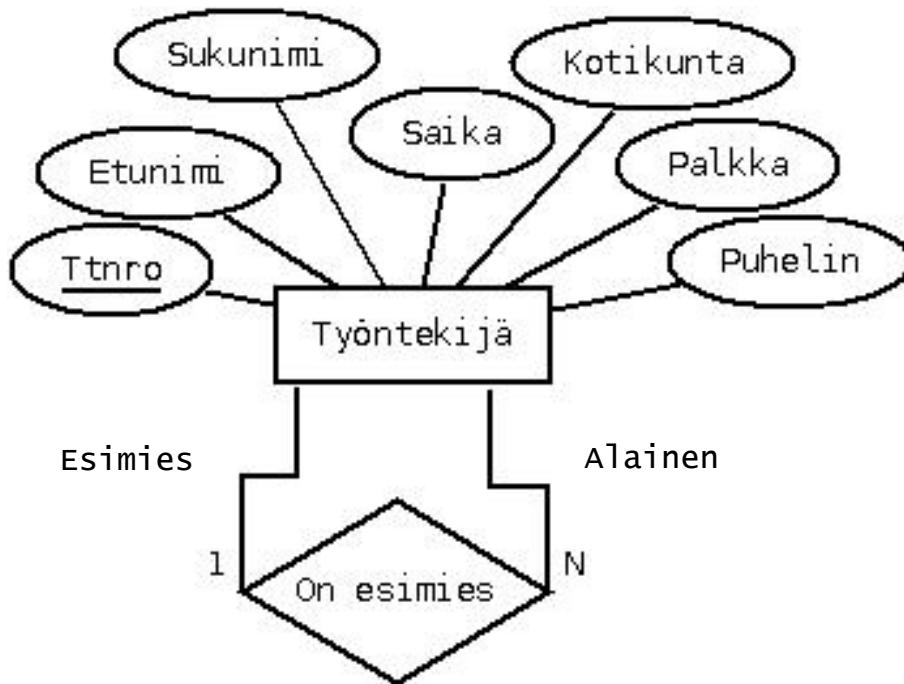
Työntekijä ei välittämättä osallistuu yhteenkään projektiin. Työntekijän osallistuminen on osittaisista.

Projektiin on osallistuttava vähintään yhden työntekijän. Projektin osallistuminen on täydellistä.



Työntekijöitä Projekteja
 "Osallistumisia"

Roolit



Osa työntekijöistä toimii esimiehinä toisille työntekijöille: osa työntekijöistä on esimiehen **roolissa**.

Osalla työntekijöistä on esimies: osa työntekijöistä on alaisen roolissa.

Esimies:Alainen 1:N

Heikko entiteettityyppi ja tunnistava suhdetyyppi

- Oletetaan, että esimerkkimme kuvitteellisessa yrityksessä halutaan tallentaa tietokantaan tietoja työntekijöiden huollettavista (lapsista).
 - nimi, sukupuoli ja syntymääika
- Huollettavista ei siis (jostakin syystä) haluta tallentaa kantaan henkilötunnuksia eikä mitään muitakaan tunnisteominaisuksia (avaimia).
 - Nimi toimii osittaisena avaimena.
- Huollettava tunnistetaan huoltajan työntekijänumerona ja huollettavan nimen perusteella.
 - Oletukset: Yhdellä työntekijällä ei voi olla useaa samannimistä huollettavaa. Jos huollettavan molemmat huoltajat ovat töissä ko. yrityksessä, vain toinen huoltajista ilmoittaa huollettavan tiedot.

(12, Pekka, ...)



(Aamu, N, 2007-01-01)
(Pekka, M, 2007-01-01)

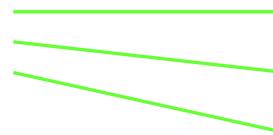
Heikko entiteettityyppi ja tunnistava suhdetyyppi

- Entiteettityyppiä, jolla ei ole omaa avainta, kutsutaan **heikoksi entiteettityypiksi** (weak entity type).
 - huollettava
- Heikolla entiteettityypillä on yleensä **osittainen avain**.
 - huollettavan nimi
- Heikon entiteetin tunnistamisessa käytetään toisen entiteettityypin entiteetiltä saatavaa avainta (toisten entiteettityyppien entiteeteiltä saatavia avaimia).
- Tätä toista entiteettityyppiä kutsutaan **tunnistavaksi** (identifying) tai **omistavaksi** (owner) entiteettityypiksi.
 - työntekijä
- Omistavia entiteettityyppejä voi olla useita.

Heikko entiteettityyppi ja tunnistava suhdetyyppi

- Tunnistaminen: omistajien avaimet + mahdollinen heikon entiteetin osittainen avain
 - työntekijän ttnro ja huollettavan nimi
- Osittaisen avaimen avulla voidaan yksilöidä tiettyyn tunnistavaan entiteettiin (tunnistaviin entiteetteihin) liittyvät heikot entiteetit.

(33, Ville, ...)



(Aamu, N, 2000-04-05)

(Taavi, M, 2002-04-10)

(Jaana, N, 2004-05-03)

(12, Pekka, ...)



(Aamu, N, 2007-01-01)

(Pekka, M, 2007-01-01)

(98, Jenni, ...)

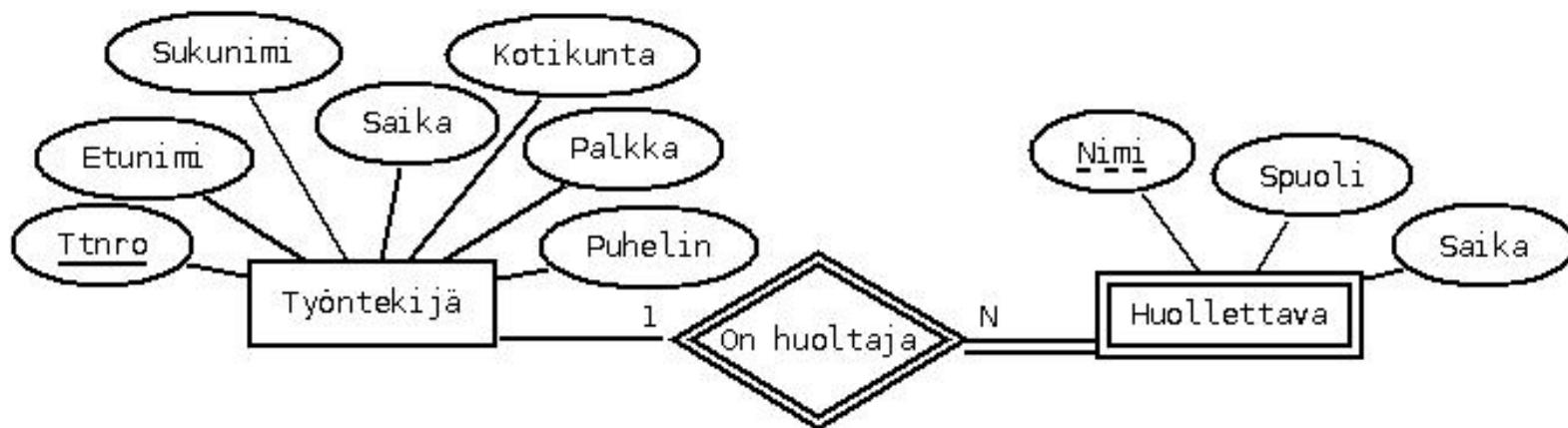


(Aapeli, M, 1991-02-28)

Heikko entiteettityyppi ja tunnistava suhdetyyppi

tunnistava
(omistava)
entiteettityyppi

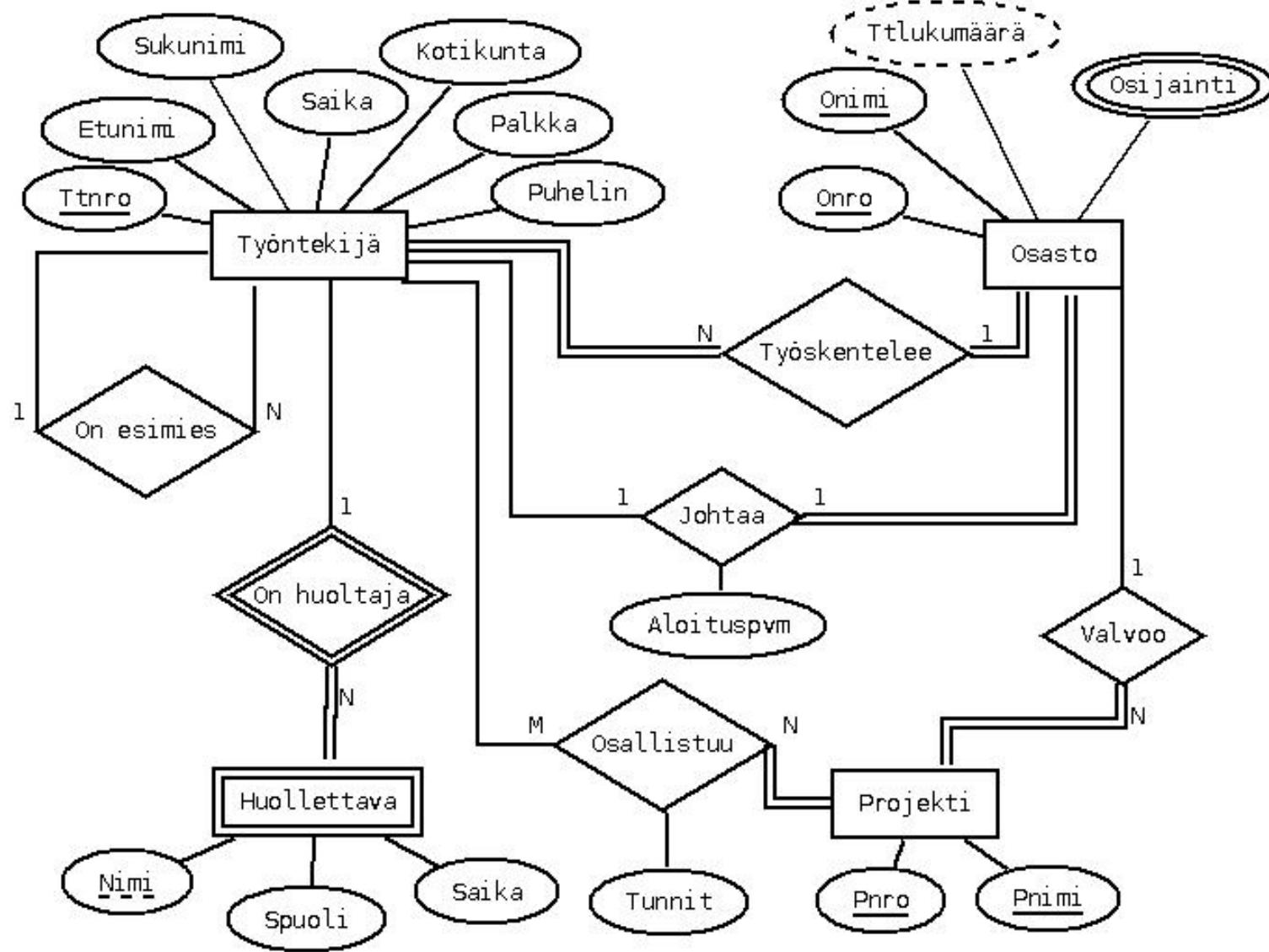
heikko
entiteettityyppi,
jolla osittainen
avain (nimi)



tunnistava
suhdetyyppi

Heikko entiteettityyppi ja tunnistava suhdetyyppi

- Heikon entiteetin on pakko osallistua tunnistavaan suhteeseen.
 - Heikon entiteettityypin osalta kyseessä on siis täydellinen osallistuminen, eksistenssiriippuvuus.
- Tunnistavien entiteettien osallistuminen on osittaisista tai täydellistä.
- Tunnistaminen voi tapahtua epäsuorasti, koska tunnistava suhde voi johtaa toiseen heikkoon entiteettiin.
 - Huom! Oletetaan, että mikään entiteetti ei ole epäsuorastikaan tunnistavassa suhteessa itsensä kanssa.
- Heikko entiteettityyppi voi osallistua suhdetyyppiin (muuhin kuin tunnistava suhdetyyppi) samalla tavalla kuin tavallisetkin entiteettityypit.



ER-kaavion lukusuunta on yleensä vasemmalta oikealle ja ylhäältä alas ER-mallinnus osa 1: 27

ER-mallinnus: Käytännön ohjeita

- ER-kaavioon otetaan mukaan ainoastaan kaikki rakennettavan tietojärjestelmän kannalta olennaiset tiedot.
- Tunnista ensin entiteettityypit ja niiden attribuutit.
 - Mistä asioista pitää tallentaa tietoa? Entiteettityypit
 - Mitä tietoja pitää tallentaa? Entiteettityyppien attribuutit
- Tunnista suhdetyypit ja niiden attribuutit
 - Miten asiat eli entiteettityypit liittyvät toisiinsa? Mitä yhteyksiä asioiden välillä on? Suhdetyypit
 - Mitä tietoja asioiden välisistä yhteyksistä pitää tallentaa? Suhdetyyppien attribuutit
- Tarkista, että entiteetti- ja suhdetyypeillä on yksikäsitteiset nimet.

ER-mallinnus: Käytännön ohjeita

- ER-kaavion tekeminen on yleensä iteratiivinen prosessi.
 - Etsitään ensin entiteettityyppi-, attribuutti- ja suhdetyyppiehdokkaita, sitten karsitaan. Etsitään uusia ehdokkaita, joita taas mahdollisesti karsitaan.
 - Attribuutit voivat muuttua entiteettityypeiksi (tai päinvastoin).
 - Attribuutit voivat muuttua suhdetyypeiksi (tai päinvastoin).
- Tekstianalyysi
 - potentiaalisia entiteettityyppejä
 - substantiivit
 - subjektit, objektit
 - potentiaalisia attribuutteja
 - substantiivit, adjektiivit
 - potentiaalisia suhdetyyppejä
 - verbit, genetiivit sekä muut ilmaukset, jotka kuvaavat kohteiden välisiä yhteyksiä
- Saman sovellusalueen voi yleensä mallintaa usealla eri tavalla.
 - Eri ratkaisuissa omat hyvät ja huonot puolensa



Tietokantojen perusteet

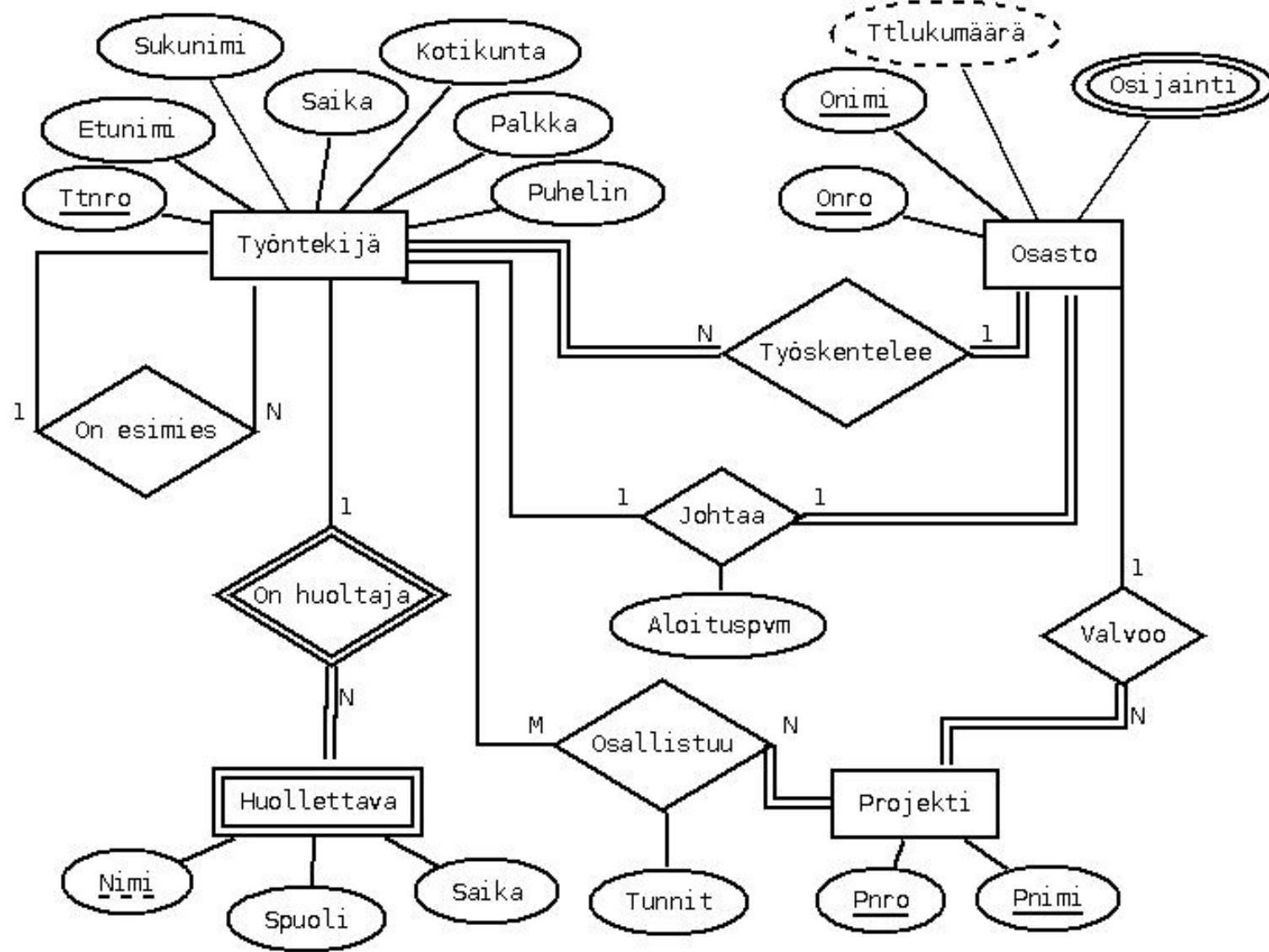
Muunnos ER-kaaviosta
SQL-tietokannan kaavioksi

Muunnos ER-kaaviosta SQL-tietokannan kaavioksi

- Muunnos ER-kaaviosta SQL-tietokannan kaavioksi tehdään tiettyjen sääntöjen mukaisesti.
- Pyritään välttämään tiedon turhaa toistoa sekä turhia tyhjäarvoja.
- Pyritään esittämään ER-kaavion rakenteellisia rajoitteita SQL-tietokannassa.
 - Kaikkia rajoitteita ei kuitenkaan pystytä esittämään aiemmin opittujen PRIMARY KEY -, UNIQUE-, FOREIGN KEY - ja NOT NULL -määreiden avulla.

ER-kaaviosta SQL-tietokannan kaavioksi

- ER-kaavio muunnetaan SQL-tietokannan kaavioksi seuraavassa järjestyksessä
 - 1 Entiteettityypit
 - 1.1 Tavalliset entiteettityypit
 - 1.2 Heikot entiteettityypit (ja tunnistavat suhdetyypit)
 - 2 Tavalliset suhdetyypit
 - 3 Moniarvoiset attribuutit



ER-kaavion lukusuunta on yleensä vasemmalta oikealle ja ylhäältä alas ER-mallinnus osa 1: 4

ER_SQL: Entiteettityypit

- Jokaiselle (tavalliselle, vahvalle) entiteettityypille luodaan oma taulu.
- Taulun nimeksi entiteettityypin nimi
- Taulun sarakkeiksi entiteettityypin attribuutit
- Taulun pääavaimeksi valitaan jokin entiteettityypin avaimista
 - PRIMARY KEY -määre pääavaimelle
 - UNIQUE-määre ja NOT NULL -määre muille avaimille

ER_SQL: Entiteettityypit

```
CREATE TABLE osasto (
    onro INT,
    onimi VARCHAR(15) NOT NULL,
    PRIMARY KEY (onro),
    UNIQUE (onimi));
```

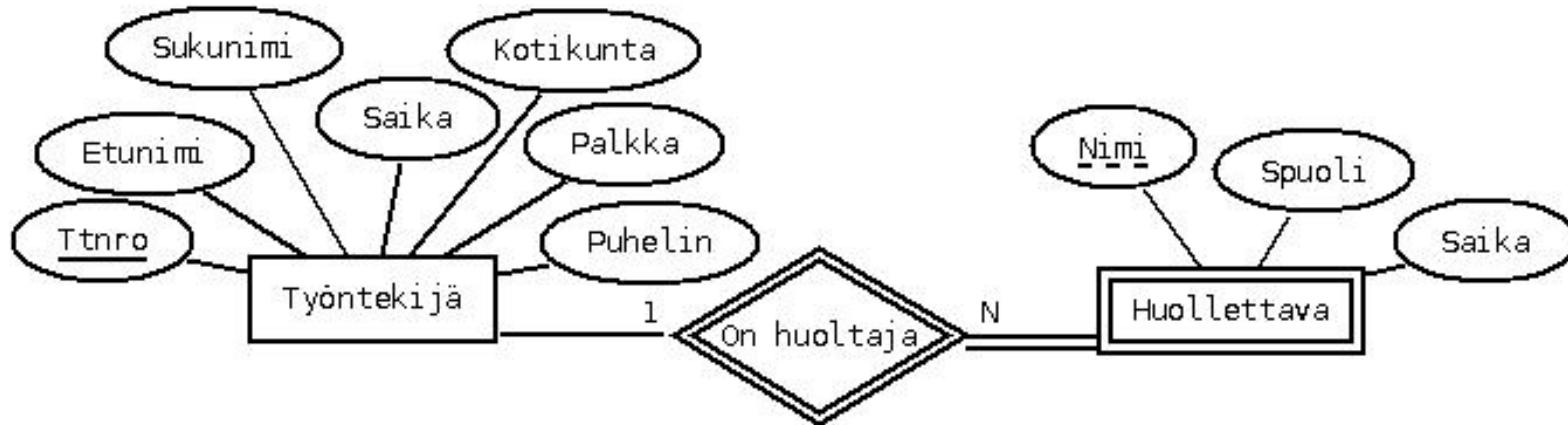
```
CREATE TABLE projekti (
    pnro INT,
    pnimi VARCHAR(15) NOT NULL,
    PRIMARY KEY (pnro),
    UNIQUE (pnimi));
```

```
CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    PRIMARY KEY (ttnro));
```

Nämä luontilauseet muuttuvat prosessin edetessä...

ER_SQL: Heikko entiteettityyppi ja tunnistava suhdetyyppi

- Luodaan heikolle entiteettityypille oma taulu.
- Taulun nimeksi heikon entiteettityypin nimi
- Taulun sarakkeiksi
 - heikon entiteettityypin attribuutit
 - tunnistavan entiteettityypin taulun pääavainsarake (sarakkeet) (tunnistavien entiteettityyppien taulujen pääavainsarakkeet)
 - josta (joista) tehdään viiteavain vastaavaan tauluun (viiteavaimet vastaaviin tauluihin)
- Taulun pääavaimeksi
 - tunnistavan entiteettityypin taululta saadun sarakkeen (saatujen sarakkeiden) (tunnistavien entiteettityyppien taululta saatujen sarakkeiden) ja
 - heikon entiteettityypin mahdollisen osittaisen avaimen yhdistelmä



tyontekija	huollettava
ttnro pk etunimi sukunimi saika kotikunta palkka puhelin	huoltajanro pk fk nimi pk spuoli saika

CREATE TABLE huollettava(

```

huoltajanro INT,
nimi VARCHAR(15),
spuoli CHAR,
saika DATE,
PRIMARY KEY (huoltajanro,nimi),
FOREIGN KEY (huoltajanro) REFERENCES tyontekija);

```

ER_SQL:
Heikko entiteettityyppi
ja tunnistava
suhdetyyppi

ER_SQL:

Heikko entiteettityyppi ja tunnistava suhdetyyppi

tyontekija

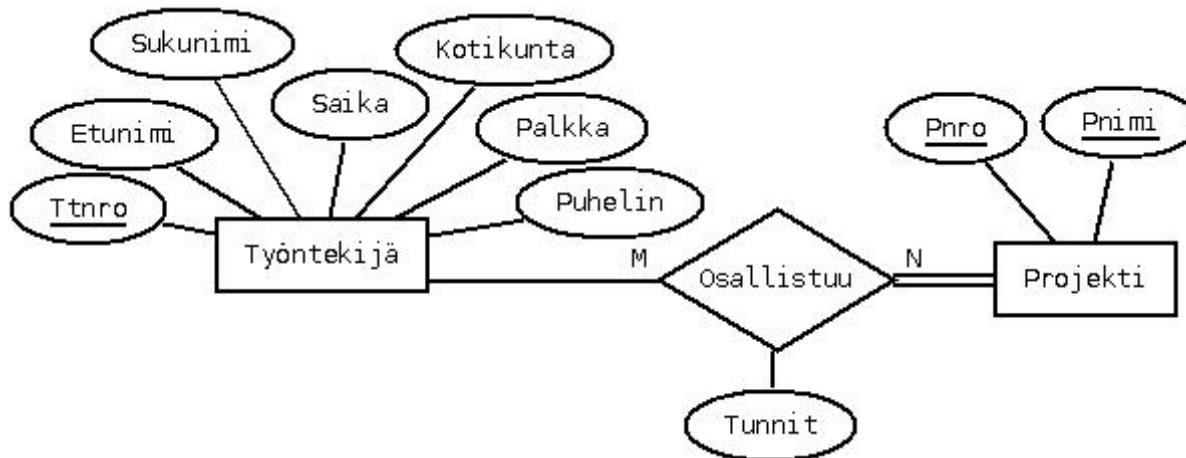
ttnro	etunimi	sukunimi	...
88	Jukka	Susi	
33	Ville	Viima	
12	Pekka	Puro	
98	Jenni	Joki	
99	Alli	Kivi	

huollettava

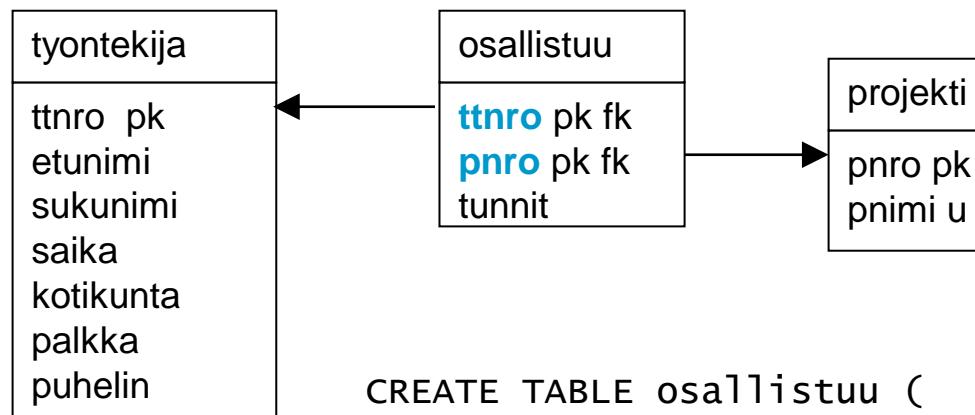
huoltajanro	nimi	spuoli	saika
33	Aamu	N	2000-04-05
33	Taavi	M	2002-04-10
33	Jaana	N	2004-05-03
98	Aapeli	M	1991-02-28
12	Aamu	N	2007-01-01
12	Pekka	M	2007-01-01

ER_SQL: M:N-suhdetyypit

- Luodaan oma taulu suhdetyyppiä varten (“suhdetaulu”).
- Taulun nimeksi tulee suhdetyyppin nimi.
- Tauluun sarakkeiksi
 - M-puolen taulun pääavainsarakkeet
 - joista tehdään viiteavain tauluun M
 - N-puolen taulun pääavainsarakkeet
 - joista tehdään viiteavain tauluun N
 - suhdetyypille mahdollisesti määritellyt attribuutit
- Suhdetaulun pääavaimeksi M- ja N-puolten taulujen pääavainsarakkeiden yhdistelmä.
 - PRIMARY KEY -määre



M:N



```
CREATE TABLE osallistuu (
  ttnro INT,
  pnro INT,
  tunnit NUMERIC(3,1),
  PRIMARY KEY (ttnro,pnro),
  FOREIGN KEY (ttnro) REFERENCES tyontekija,
  FOREIGN KEY (pnro) REFERENCES projekti);
```

tyontekija

ttnro	etunimi	...	puhelin
88	Jukka		444 1234
33	Ville		444 4343
12	Pekka		
98	Jenni		444 4488
99	Alli		444 5555

projekti

pnro	pnimi
1	Tuote X
2	Tuote Y
3	Tuote Z

osallistuu

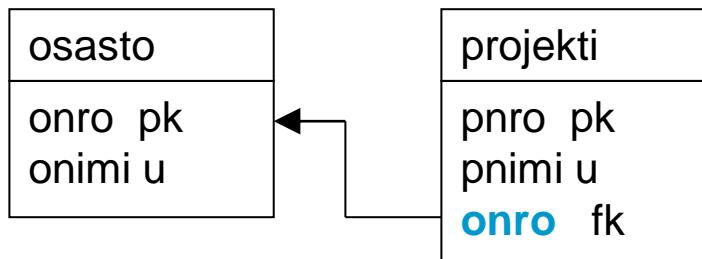
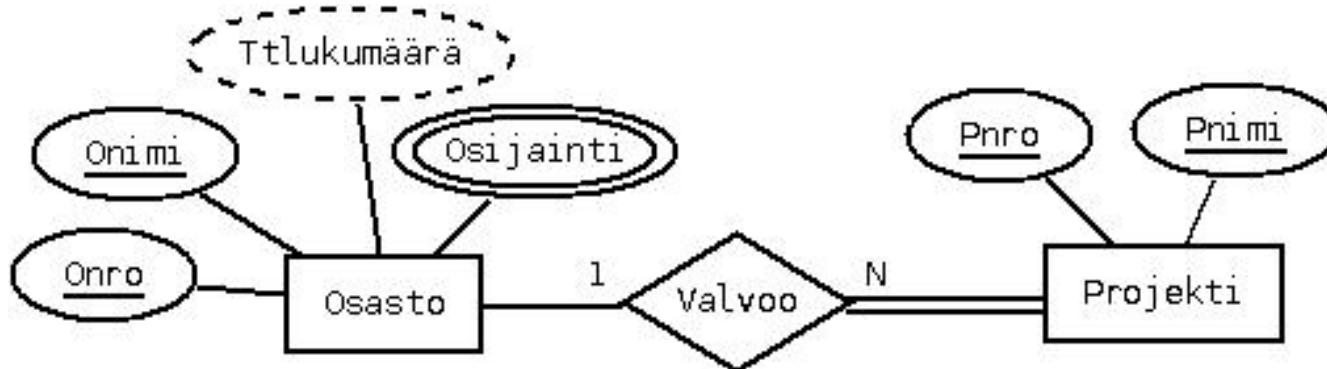
ttnro	pnro	tunnit
12	1	32.5
12	2	7.5
33	2	10.0
33	3	10.0
99	3	30.0
99	1	10.0
98	2	15.0

ER_SQL: 1:N-suhdetyypit (N:1)

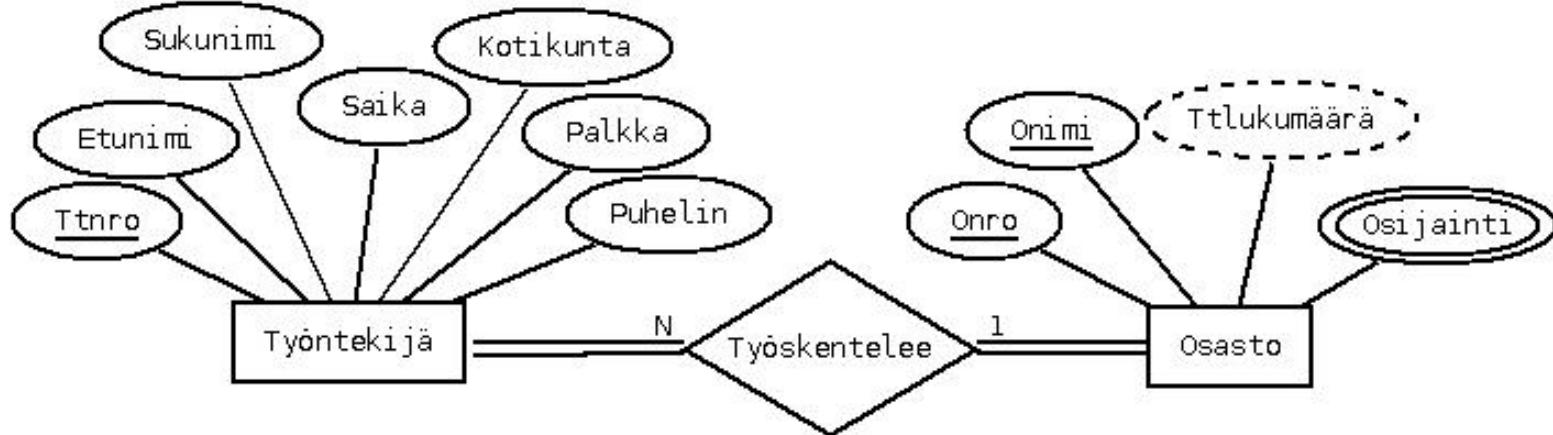
Kun **N-puolen** entiteettityyppi **osallistuu täydellisesti** suhdetyyppiin,

- lisätään N-puolen tauluun
 - 1-puolen taulun pääavainsarakkeet
 - NOT NULL -määreellä varustettuina
 - ja tehdään näistä viiteavain, joka viittaa 1-puolen tauluun
 - suhdetyypille mahdollisesti määritellyt attribuutit

1:N



```
CREATE TABLE projekti (
    pnro INT,
    pnimi VARCHAR(15) NOT NULL,
    onro INT NOT NULL,
    PRIMARY KEY (pnro),
    UNIQUE (pnimi),
    FOREIGN KEY(onro) REFERENCES osasto);
```



N:1

tyontekija
ttnro pk
etunimi
sukunimi
saika
kotikunta
palkka
puhelin
osastonro fk

osasto
onro pk
onimi u

```

CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    osastonro INT NOT NULL,
    PRIMARY KEY (ttnro),
    FOREIGN KEY (osastonro) REFERENCES osasto);
  
```

1:N

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

projekti

pnro	pnimi	onro
1	Tuote X	5
2	Tuote Y	5
3	Tuote Z	5

N:1

tyontekija

ttnro	etunimi	sukunimi	...	osastonro
88	Jukka	Susi		1
33	Ville	Viima		5
12	Pekka	Puro		5
98	Jenni	Joki		4
99	Alli	Kivi		4

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

ER_SQL: 1:N-suhdetyypit (N:1)

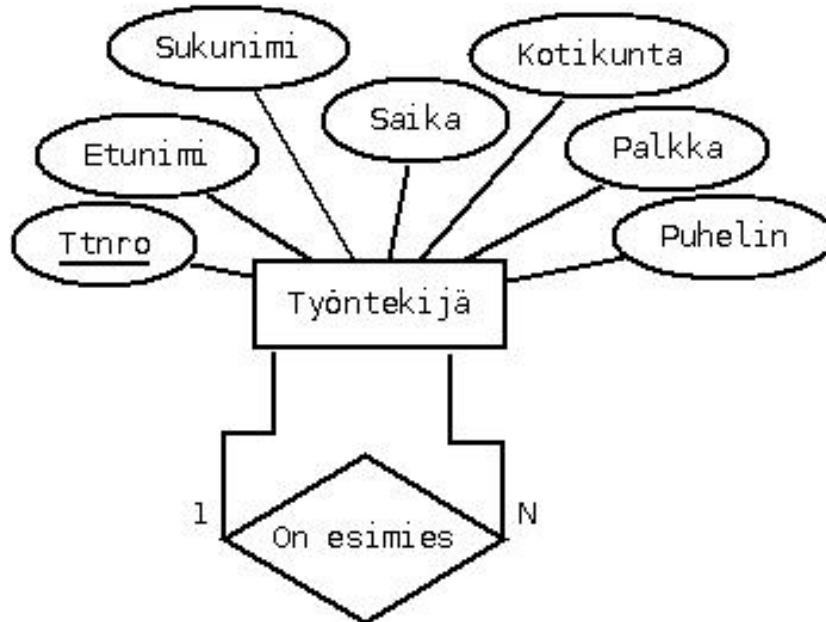
Kun **N-puolen** entiteettityyppi **osallistuu** suhdetyyppiin **osittain**,

- lisätään N-puolen tauluun
 - 1-puolen taulun pääavainsarakkeet
 - ja tehdään näistä viiteavain, joka viittaa 1-puolen tauluun
 - suhdetyypille mahdollisesti määritellyt attribuutit

tyontekija

ttnro	etunimi	sukunimi	...	osastonro	esimiesnro
88	Jukka	Susi		1	
33	Ville	Viima		5	88
12	Pekka	Puro		5	33
98	Jenni	Joki		4	88
99	Alli	Kivi		4	98

1:N



tyontekija

ttnro	pk
etunimi	
sukunimi	
saika	
kotikunta	
palkka	
puhelin	
osastonro	fk
esimiesnro	fk

CREATE TABLE tyontekija (

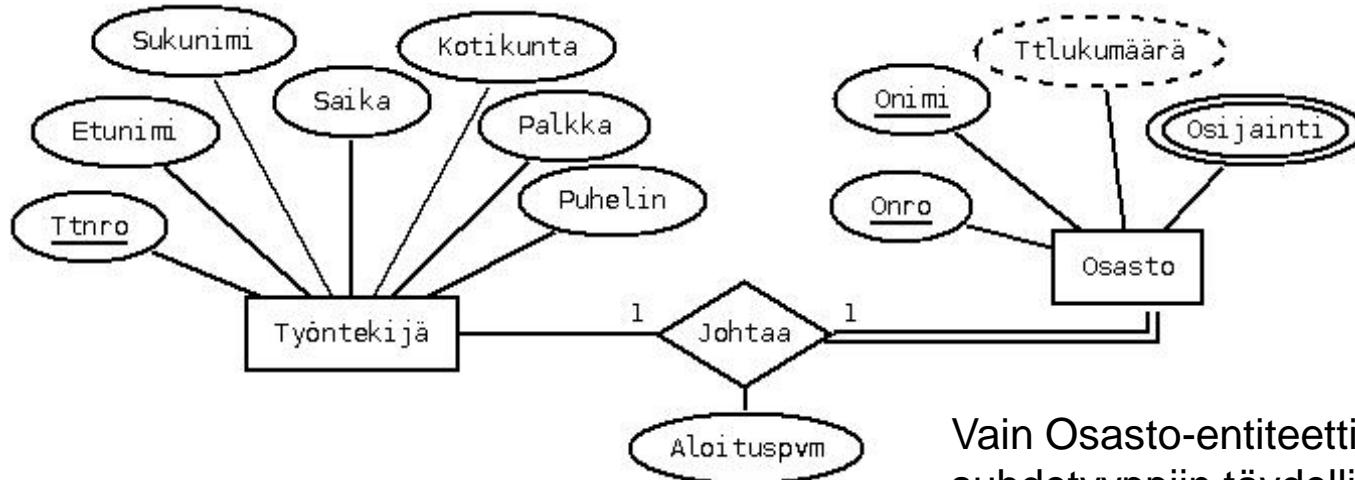
```

ttnro INT,
etunimi VARCHAR(15) NOT NULL,
sukunimi VARCHAR(20) NOT NULL,
saika DATE NOT NULL,
kotikunta VARCHAR(20) NOT NULL,
palkka NUMERIC(8,2),
puhelin VARCHAR(15),
osastonro INT NOT NULL,
esimiesnro INT,
PRIMARY KEY (ttnro),
FOREIGN KEY (osastonro) REFERENCES osasto,
FOREIGN KEY (esimiesnro) REFERENCES tyontekija);
```

ER_SQL: 1:1-suhdetyypit

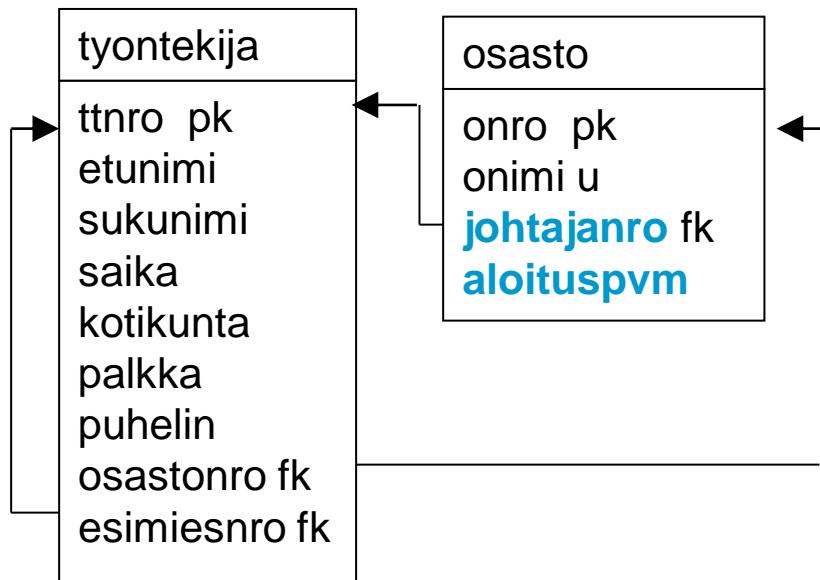
Jos **ainoastaan toinen entiteettityyppi osallistuu suhdetyyppiin täydellisesti**,

- Lisätään täydellisesti osallistuvan entiteettityyppin tauluun T
 - osittain osallistuvan entiteettityyppin taulun O pääavainsarakkeet (pääavainsarakkeet)
 - NOT NULL -määreellä varustettuina
 - ja tehdään näistä viiteavain, joka viittaa tauluun O
 - suhdetyypille mahdollisesti määritellyt attribuutit



1:1

Vain Osasto-entiteettiyyppi osallistuu suhdetyyppiin täydellisesti, joten tehdään viiteavain osasto-taulusta tyontekija-tauluun



```
CREATE TABLE osasto (
  onro INT,
  onimi VARCHAR(15) NOT NULL,
  johtajanro INT NOT NULL,
  aloituspvm DATE,
  PRIMARY KEY (onro),
  UNIQUE (onimi),
  FOREIGN KEY(johtajanro)
    REFERENCES tyontekija);
```

1:1

tyontekija

ttnro	etunimi	sukunimi	...	osastonro	esimiesnro
88	Jukka	Susi		1	
33	Ville	Viima		5	88
12	Pekka	Puro		5	33
98	Jenni	Joki		4	88
99	Alli	Kivi		4	98

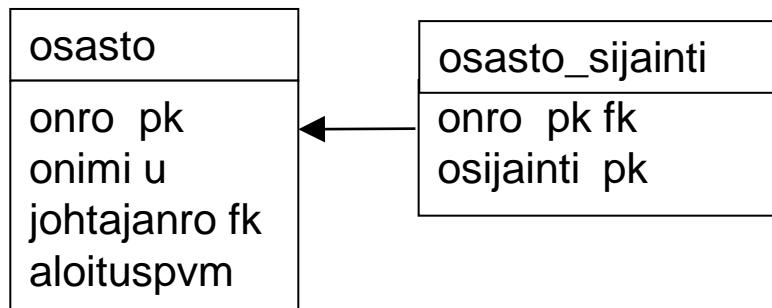
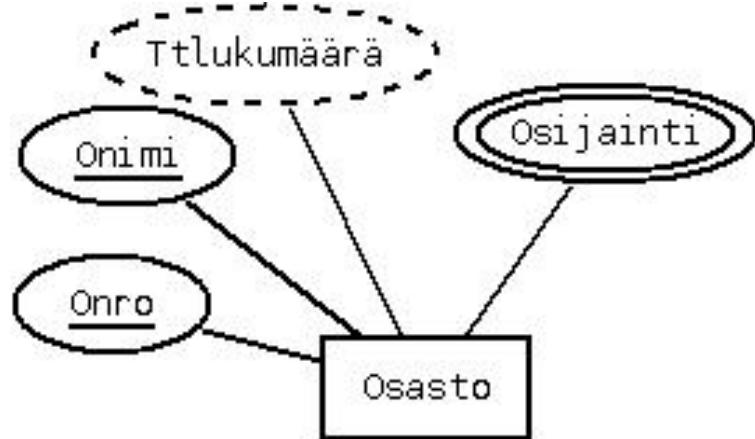
osasto

onro	onimi	johtajanro	aloituspvm
1	Pääkonttori	88	1989-06-19
4	Hallinto	98	1992-01-01
5	Tutkimus	33	2000-05-22

ER_SQL: Moniarvoiset attribuutit

- Moniarvoista attribuuttia varten tehdään oma taulu.
- Taulun nimeksi esim. yhdistelmä entiteettityypin ja moniarvoisen attribuutin nimestä.
- Taulun sarakkeiksi
 - entiteettityypin pääavainsarake (pääavainsarakkeet)
 - josta tehdään entiteettityypin tauluun viittaava viiteavain
 - moniarvoinen attribuutti
- Taulun pääavaimaksi
 - entiteettityypin taululta saadun sarakkeen (saatujen sarakkeiden) ja
 - moniarvoisen attribuutin sarakkeen yhdistelmä

ER_SQL: Moniarvoiset attribuutit



```
CREATE TABLE osasto_sijainti(
    onro INT,
    osijainti VARCHAR(15),
    PRIMARY KEY (onro, osijainti),
    FOREIGN KEY (onro) REFERENCES osasto);
```

ER_SQL: Moniarvoiset attribuutit

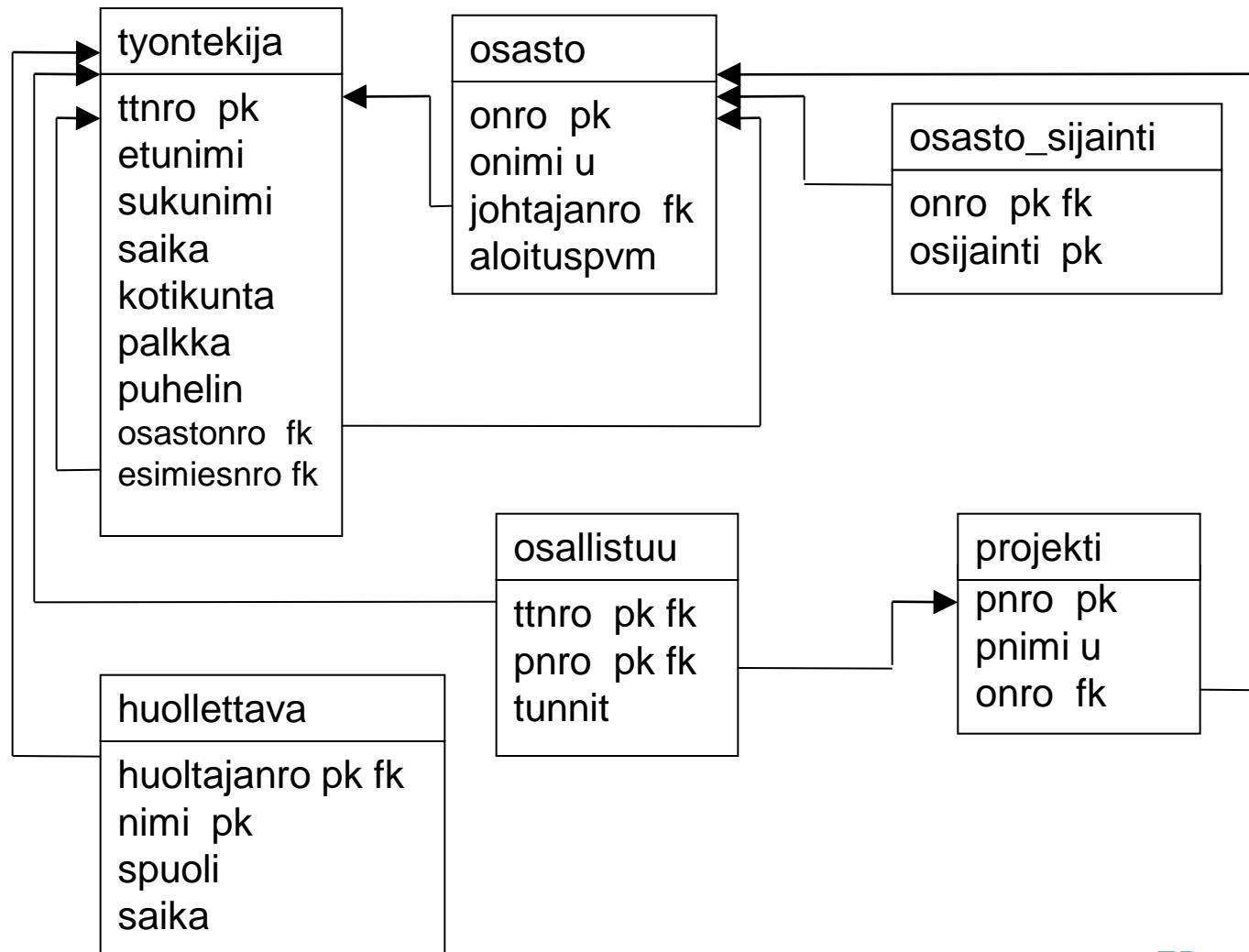
osasto

onro	onimi	johtajanro	aloituspvm
1	Pääkonttori	88	1989-06-19
4	Hallinto	98	1992-01-01
5	Tutkimus	33	2000-05-22

osasto_sijainti

onro	osijanti
1	Tampere
4	Tampere
5	Tampere
5	Lempäälä
5	Nokia

Esimerkkisovellusalueen SQL-tietokannan koko kaavio



ER-kaaviosta SQL-tietokannan kaavioksi

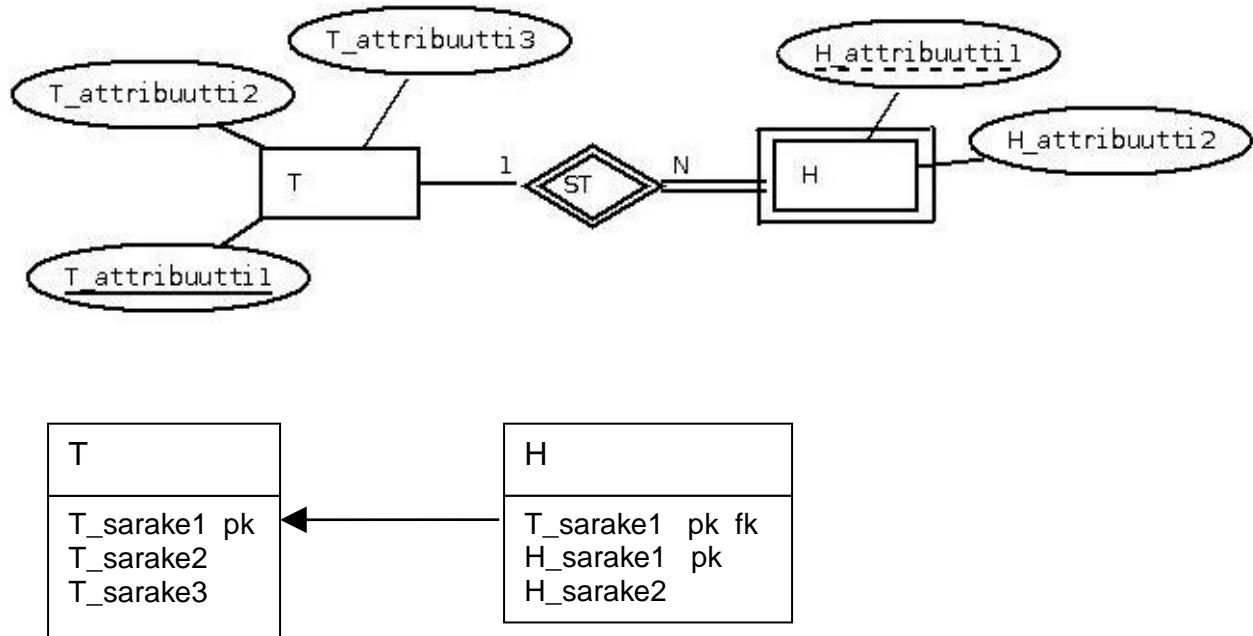
ER-kaavio	SQL-tietokannan kaavio
Entiteettityyppi	"Entiteettitaulu"
Attribuutti	Sarake
Johdettu attribuutti	Jätetään pois
Moniarvoinen attribuutti	"Attribuuttitaulu", jossa yksi viiteavain
1:1-suhdetyyppi	Viiteavain tai yhdistetty taulu tai "suhdetaulu", jossa kaksi viiteavainta
1:N-suhdetyyppi	Viiteavain tai "suhdetaulu", jossa kaksi viiteavainta
M:N-suhdetyyppi	"Suhdetaulu" ja kaksi viiteavainta
Heikko entiteettityyppi ja tunnistava suhdetyyppi	"Entiteettitaulu" ja viiteavain tunnistavaan tauluun (viiteavaimet tunnistaviin tauluihin)
Avain	PRIMARY KEY tai UNIQUE-määre

Erilaisia muunnosvaihtoehtoja esitellään myöhemmin.

ER-kaaviosta SQL-tietokannan kaavioksi

Entiteettityypeille ja heikolle entiteettityypeille tehdään omat taulunsa, joissa on oma sarake kutakin yksinkertaista attribuuttia kohti. Moniarvoisille attribuuteille tehdään omat taulunsa. Suhdetyyppit muunnetaan viiteavaimiksi ja mahdollisesti suhdetauluiksi. Viiteavainsarakkeen nimen ei tarvitse olla sama kuin viittauksen kohteena olevan sarakkeen nimen, mutta tietotyypin on oltava sama.

Heikko entiteettityyppi ja tunnistava suhdetyyppi



Luodaan heikolle entiteettityypille oma taulu.

Taulun sarakkeiksi laitetaan

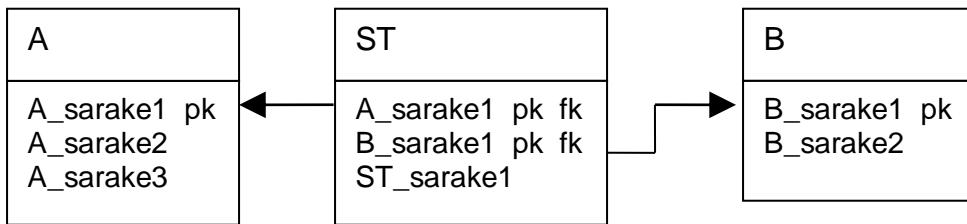
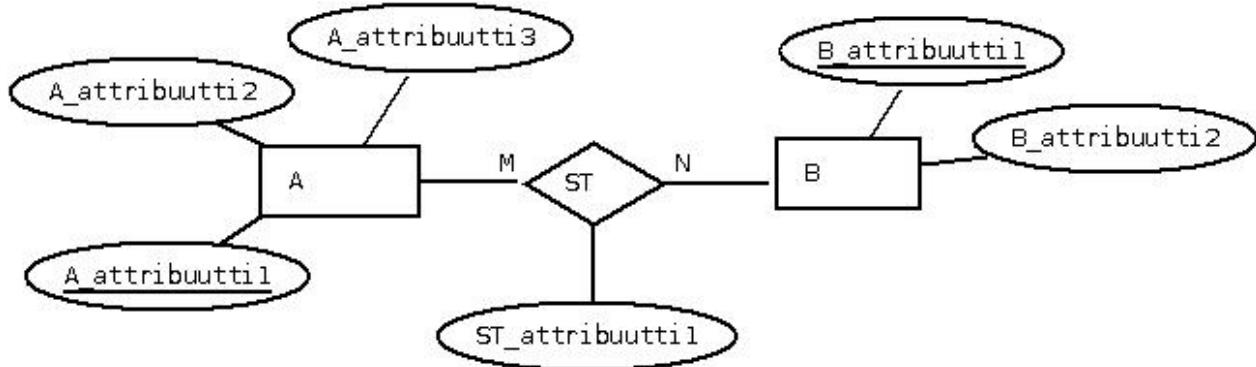
- heikon entiteettityyppin attribuutit
- tunnistavien entiteettityyppien taulujen pääavainsarakkeet, joista tehdään viiteavaimet, jotka viittaavat vastaavien taulujen pääavaimiin
 - **T_sarake1** tietotyyppi
 - FOREIGN KEY (**T_sarake1**) REFERENCES **T**

Taulun pääavaimeksi tunnistavien entiteettityyppien tauluilta saatujen sarakkeiden ja heikon entiteettityyppin mahdollisen osittaisen avaimen yhdistelmä

- PRIMARY KEY (**T_sarake1**, **H_sarake1**)

ER-kaaviosta SQL-tietokannan kaavioksi

M:N



Tehdään aina ”suhdetaulu”, jonka sarakkeiksi laitetaan

- M-puolen taulun pääavainsarake (sarakkeet), josta tehdään viiteavain, joka viittaa M-puolen taulun pääavaimiin
 - A_sarake1 tietotyyppi
 - FOREIGN KEY (A_sarake1) REFERENCES A
- N-puolen taulun pääavainsarake (sarakkeet), josta tehdään viiteavain, joka viittaa N-puolen taulun pääavaimiin
 - B_sarake1 tietotyyppi
 - FOREIGN KEY (B_sarake1) REFERENCES B
- suhdetyypin attribuutit
 - ST_sarake1

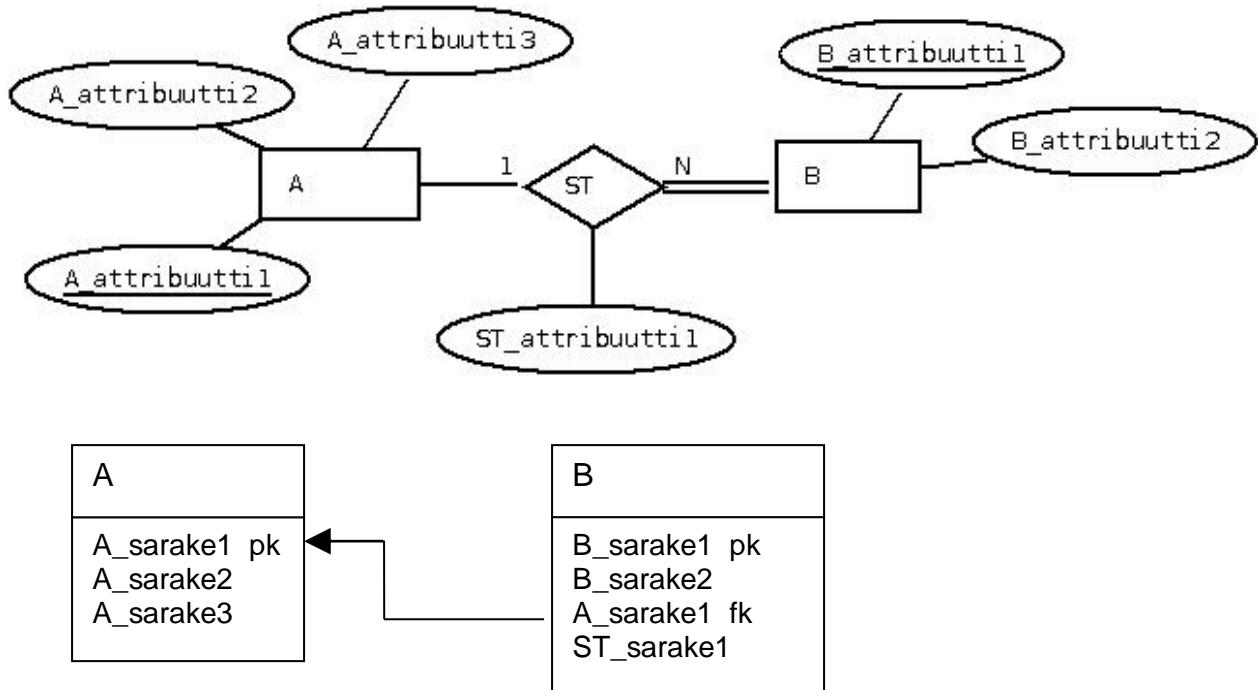
Suhdetaulun pääavaimeksi tulee M- ja N-puolten tauluilta saatujen sarakkeiden yhdistelmä

- PRIMARY KEY (A_sarake1, B_sarake1)

ER-kaaviosta SQL-tietokannan kaavioksi

1:N

Kun **N-puoli osallistuu täydellisesti** (1-puolen osallistumistavalla ei ole merkitystä)



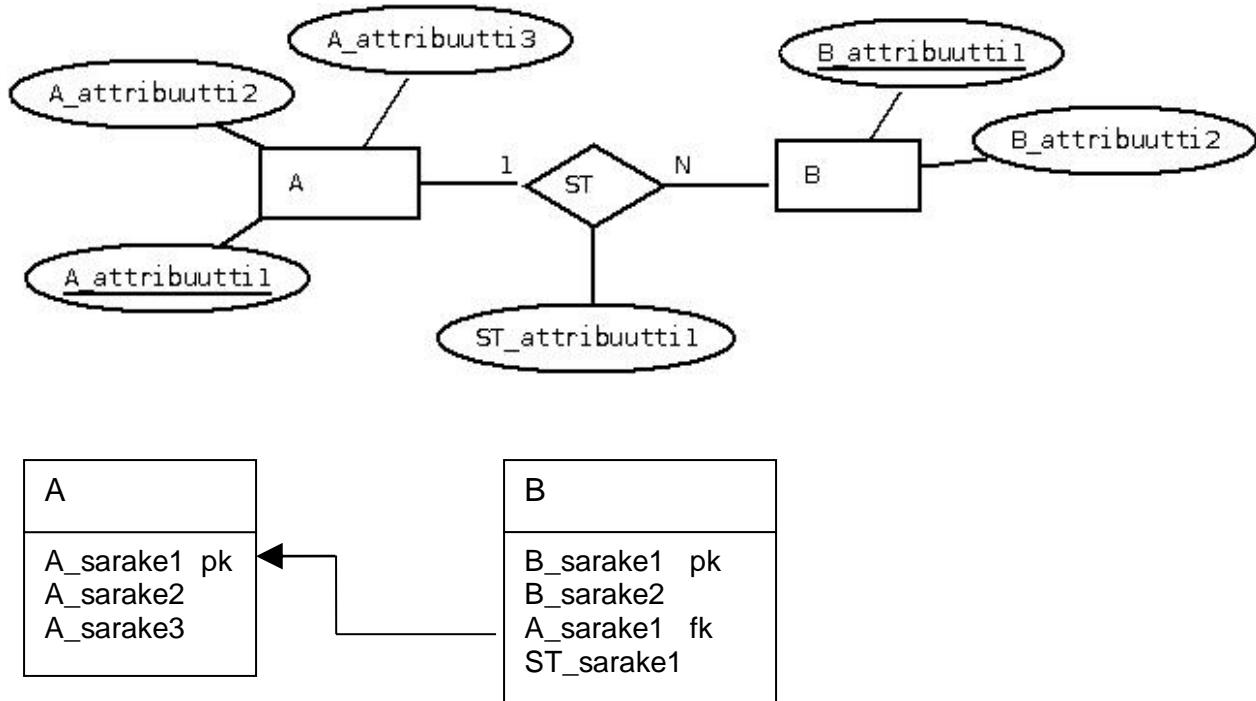
Lisätään **N-puolen tauluun** (tässä tauluun B)

- 1-puolen taulun päävainsarake (sarakkeet) **NOT NULL -määreellä** varustettuna ja tehdään siitä viiteavain, joka viittaa 1-puolen taulun päävaiimeen
 - A_sarake1 tietotyppi NOT NULL
 - FOREIGN KEY (A_sarake1) REFERENCES A
- suhdetyyppin attribuutit
 - ST_sarake1

ER-kaaviosta SQL-tietokannan kaavioksi

1:N

Kun **N-puoli osallistuu osittain** (1-puolen osallistumistavalla ei ole merkitystä)



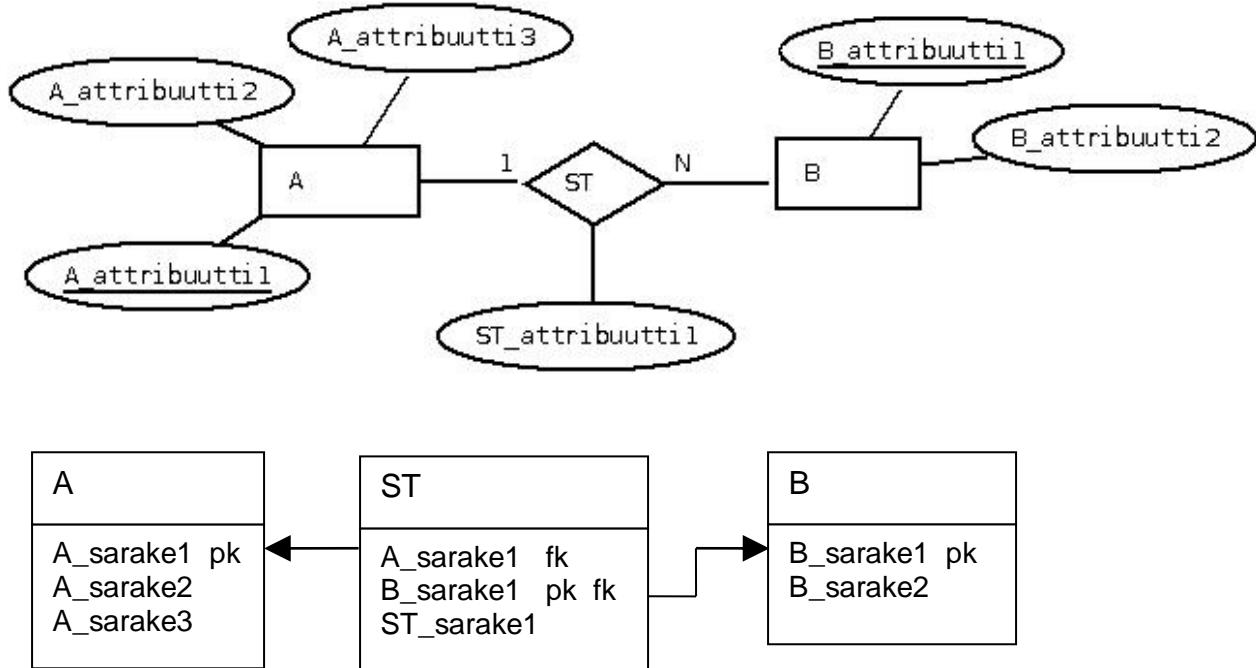
Lisätään **N-puolen tauluun** (tässä tauluun B)

- 1-puolen taulun pääavainsarake (sarakkeet) ja tehdään siitä viiteavain, joka viittaa 1-puolen taulun pääavaimseen
 - A_sarake1 tietotyyppi
 - FOREIGN KEY (A_sarake1) REFERENCES A
- suhdetyyppin attribuutit
 - ST_sarake1

ER-kaaviosta SQL-tietokannan kaavioksi

1:N

Kun **N-puoli osallistuu osittain** ja on **vain harvoja suhteita** (1-puolen osallistumistavalla ei ole merkitystä)



Voidaan tehdä oma ”suhdetaulu”. Suhdetaulun sarakkeiksi laitetaan

- 1-puolen taulun pääavainsarake (sarakkeet), josta tehdään viiteavain, joka viittaa 1-puolen taulun pääavaimeen
 - A_sarake1 tietotyyppi NOT NULL
 - FOREIGN KEY (A_sarake1) REFERENCES A
- N-puolen taulun pääavainsarake (sarakkeet), josta tehdään viiteavain, joka viittaa N-puolen taulun pääavaimeen
 - B_sarake1 tietotyyppi
 - FOREIGN KEY (B_sarake1) REFERENCES B
- suhdetyypin attribuutit
 - ST_sarake1

Suhdetaulun pääavaimeksi N-puolen taululta saatu sarake (saadut sarakkeet)

- PRIMARY KEY (B_sarake1)

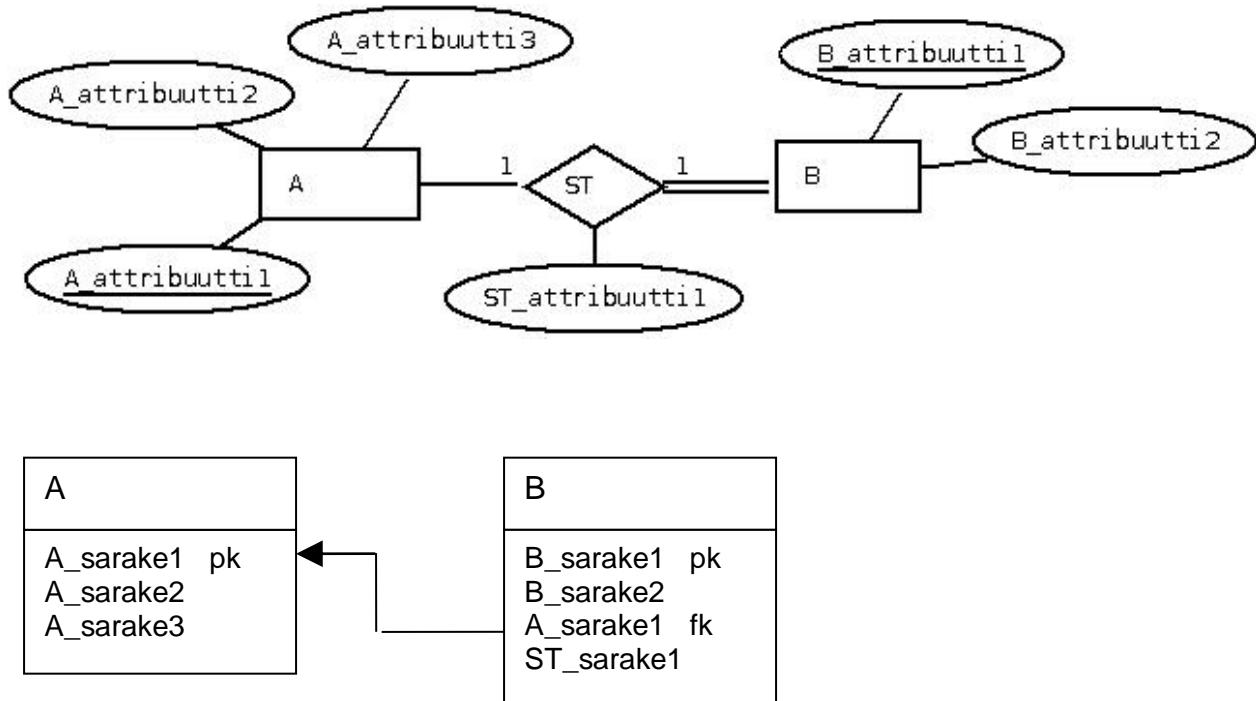
1-puolen tauluun viittaavalle sarakkeelle (viittaaville sarakkeille) annetaan NOT NULL -määre

- A_sarake1 tietotyyppi NOT NULL

ER-kaaviosta SQL-tietokannan kaavioksi

1:1

Kun vain toinen entiteettityypeistä osallistuu täydellisesti



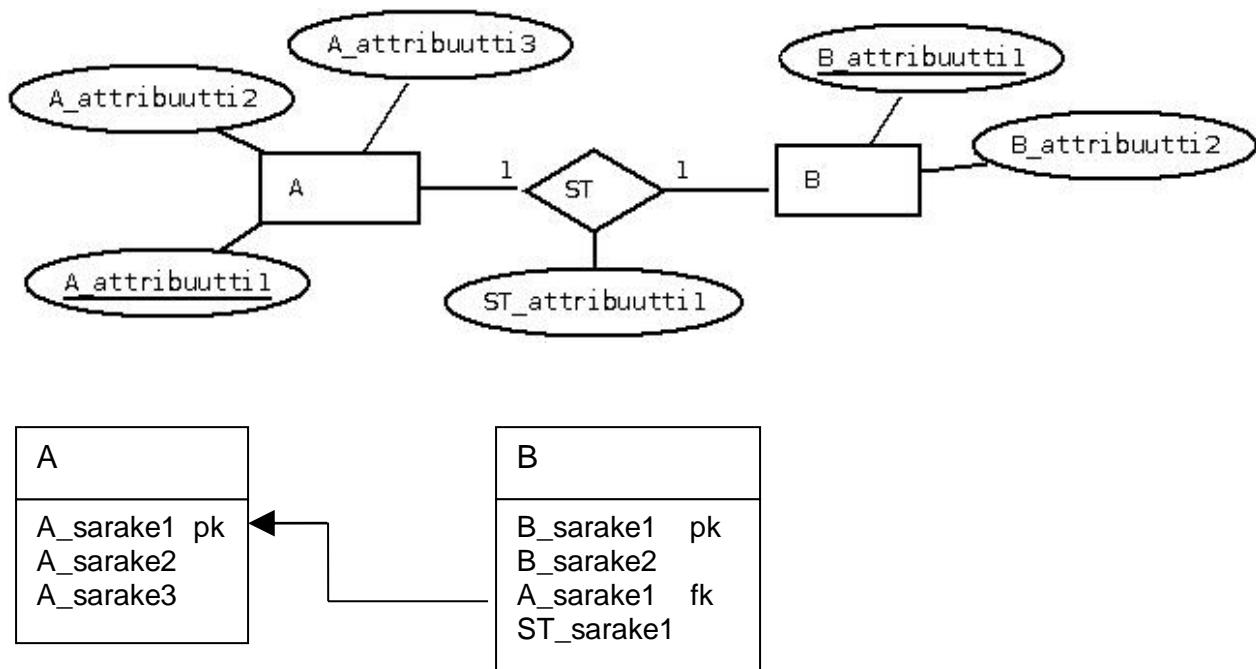
Lisätään **täydellisesti osallistuvan entiteettityypin tauluun** (tässä B)

- osittain osallistuvan entiteettityypin taulun pääavainsarake (sarakkeet) **NOT NULL - määreellä** varustettuna ja tehdään siitä viiteavain, joka viittaa osittain osallistuvan entiteettityypin taulun pääavaimeen
 - A_sarake1 tietotyppi NOT NULL
 - FOREIGN KEY (A_sarake1) REFERENCES A
- suhdetyyppin attribuutit
 - ST_sarake1

ER-kaaviosta SQL-tietokannan kaavioksi

1:1

Molemmat osallistuvat osittain

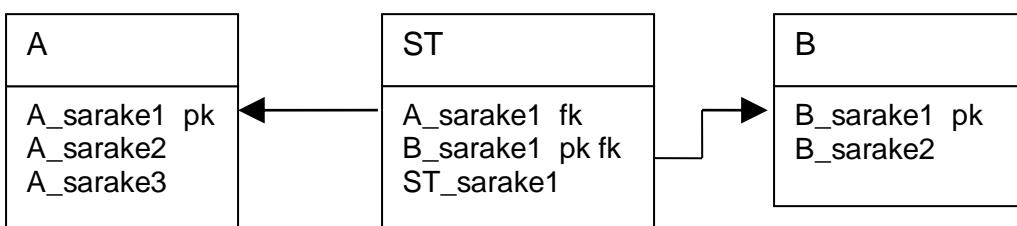


Valitaan toinen taulusta tauluksi, johon lisäykset tehdään. Tässä valittu taulu B. Tauluun B on lisätty

- taulun A pääavainsarake, josta on tehty viiteavain, joka viittaa taulun A pääavaimeen
- suhdetyypin attribuutti

(Muunnos tehdään siis samalla tavalla kuin jos toinen entiteettityyppi osallistuisi täydellisesti suhdetyyppiin, mutta NOT NULL -määre jää pois.)

Jos on vain harvoja suhteita, voidaan suhdetyypille tehdä oma taulu.

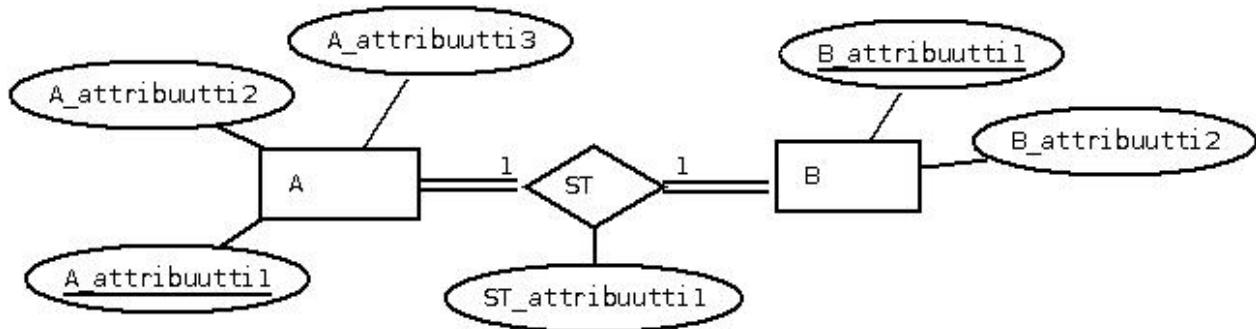


Toimitaan kuten M:N-suhdetyypin tapauksessa, mutta nyt suhdeaulun pääavain muodostuu vain toisen viiteavaimen sarakkeesta (sarakeista), ei molempien. (Pääavaimeen kuulumattomalle viiteavaimelle voidaan määritellä UNIQUE- ja NOT NULL -määreit.) (Pääavain ja avain voidaan jättää myös kokonaan määrittelemättä.)

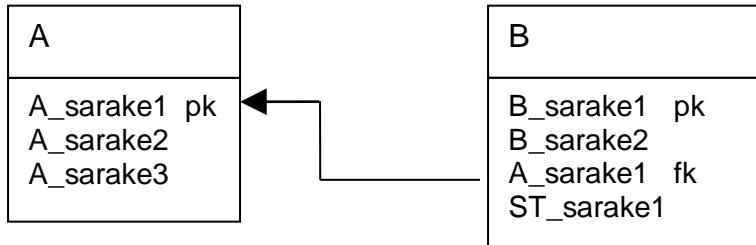
ER-kaaviosta SQL-tietokannan kaavioksi

1:1

Molemmat osallistuvat täydellisesti



Vaihtoehto A: Valitaan toinen taulusta (esim. taulu B) tauluksi, johon viiteavaimen ja suhdetyypin attribuuttien lisäykset tehdään.



Vaihtoehto B: Entiteettityypeille tehdään yhdistetty taulu.

AB
A_sarake1_pk A_sarake2 A_sarake3 B_sarake1 B_sarake2 ST_sarake1

Taulun sarakkeiksi laitetaan molempien entiteettityyppien attribuutit ja suhdetyypin attribuutit.

Jos entiteettityypeillä on eri avain, valitaan toisen entiteettityypin (esim. A) avain yhdistetyn taulun pääavaimeksi. Toisen entiteettityypin (esim. B) avaimelle voidaan antaa UNIQUE-määre ja NOT NULL -määre.

- PRIMARY KEY (A_sarake1)
- UNIQUE (B_sarake1)



Tietokantojen perusteet

SQL – Loogiset operaatiot

WHERE-osa: Loogiset operaatiot

- Kysely-, päivitys- ja poistolauseiden WHERE-osien ehtojen avulla rajataan rivejä,
 - jotka tulevat mukaan kyselyn tulostauluun
 - joita päivitetään (joiden tietoja muutetaan) tai
 - jotka poistetaan.

Toimenpiteet kohdistuvat riveihin, joilla WHERE-osan ehto saa toteutusarvon tosi.

```
SELECT sarake {, sarake}  
FROM taulu {,taulu}  
[WHERE ehto]
```

```
UPDATE taulu  
SET sarake = ilmaus {, sarake = ilmaus}  
[WHERE ehto]
```

```
DELETE FROM taulu  
[WHERE ehto]
```

WHERE-osa: Loogiset operaatiot

- WHERE-osan ehdossa voidaan käyttää AND-, OR- ja NOT-operaatioita.
 - Merkitykset vastaavat loogisia JA-, TAI- ja EI-operaatioita
- SQL:n kolme totuusarvoa käytetään loogisissa lausekkeissa kalvoilla 4-6 esitetyjen totuustaulujen mukaisesti.

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

- **A AND B** on tosi, kun sekä A että B ovat toisia

A	B	A AND B
true	true	true
true	false	false
true	unknown	unknown
false	true	false
false	false	false
false	unknown	false
unknown	true	unknown
unknown	false	false
unknown	unknown	unknown

```
SELECT sukunimi, kotikunta, palkka
FROM tyontekija
WHERE kotikunta = 'Tampere'
AND palkka > 3000;
```

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00

- A ja B ovat alkeisehtoja

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

- **A OR B** on tosi, kun
 - joko A tai B on tosi
 - sekä A että B ovat toisia

A	B	A OR B
true	true	true
true	false	true
true	unknown	true
false	true	true
false	false	false
false	unknown	unknown
unknown	true	true
unknown	false	unknown
unknown	unknown	unknown

```
SELECT sukunimi, kotikunta, palkka
FROM tyontekija
WHERE kotikunta = 'Tampere'
OR palkka > 3000;
```

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00
viima	Nokia	4000.50
Puro	Tampere	3000.00
Joki	Lempäälä	4300.00

- **A ja B** ovat alkeisehtoja

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

- **NOT A** on tosi, kun
 - kun A on epätosi

A	NOT A
true	false
false	true
unknown	unknown

```
SELECT sukunimi, kotikunta
FROM tyontekija
WHERE NOT kotikunta = 'Tampere';
```

sukunimi	kotikunta
viima	Nokia
Joki	Lempäälä
Kivi	Nokia

Loogiset operaatiot: laskentajärjestys

- Operaatiot priorisoidaan seuraavasti:
 1. NOT
 2. AND
 3. OR
 - NOT-operaation arvo lasketaan siis ensimmäisenä laskettaessa koko lausekkeen totuusarvoa.
 - Jos lausekkeessa on useita samoja operaatioita, edetään lausekkeen totuusarvoa laskettaessa vasemmalta oikealle.
- Laskentajärjestystä voidaan muuttaa suluilla.
 - Suluilla on korkein prioriteetti: sulut lasketaan ensin, sitten NOT, AND ja OR.
 - Jos lausekkeessa on sisäkkäisiä sulkuja, edetään sisimmistä suluista uloimpiin.
- Sulkuja kannattaa käyttää myös selkeyden vuoksi.

```
SELECT sukunimi, kotikunta, palkka  
FROM tyontekija  
WHERE kotikunta = 'Nokia' OR  
      kotikunta = 'Tampere' AND  
      palkka > 3000;
```

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00
Viima	Nokia	4000.50
Kivi	Nokia	2500.00

```
SELECT sukunimi, kotikunta, palkka  
FROM tyontekija  
WHERE (kotikunta = 'Nokia' OR  
       kotikunta = 'Tampere') AND  
       palkka > 3000;
```

**Sulut muuttavat
laskentajärjestystä.**

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00
Viima	Nokia	4000.50

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

```
SELECT sukunimi, kotikunta, palkka
FROM tyontekija
WHERE kotikunta = 'Nokia' OR kotikunta = 'Tampere' AND palkka > 3000;
```

kotikunta = 'Nokia' OR kotikunta = 'Tampere' AND palkka > 3000

true

false

false

OR

true

false

AND

Minkä totuusarvon WHERE-osan ehto tuottaa, kun sitä sovelletaan työntekijän ttnro 99 riviin? true

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00
Viima	Nokia	4000.50
Kivi	Nokia	2500.00

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

```
SELECT sukunimi, kotikunta, palkka
FROM tyontekija
WHERE (kotikunta = 'Nokia' OR kotikunta = 'Tampere') AND palkka > 3000;
```

(kotikunta = 'Nokia' OR kotikunta = 'Tampere') AND palkka > 3000



true

sulut,
OR

false

false

AND

false

Minkä totuusarvon WHERE-osan ehto tuottaa, kun sitä sovelletaan työntekijän ttnro 99 riviin? **false**

sukunimi	kotikunta	palkka
Susi	Tampere	5500.00
Viima	Nokia	4000.50

Loogiset operaatiot: laskentajärjestys

```
SELECT sukunimi, kotikunta, palkka, saika
FROM tyontekija
WHERE kotikunta = 'Tampere' OR
      palkka > 3000 AND
      NOT saika < '1960-01-01';
```

3. OR
2. AND
1. NOT

sukunimi	kotikunta	palkka	saika
Susi	Tampere	5500.00	1957-11-10
Viima	Nokia	4000.50	1975-12-08
Puro	Tampere	3000.00	1985-01-09
Joki	Lempäälä	4300.00	1961-06-20

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

```
SELECT sukunimi, kotikunta, palkka, saika
FROM tyontekija
WHERE kotikunta = 'Tampere' OR palkka > 3000 AND NOT saika < '1960-01-01';
```

kotikunta = 'Tampere' OR palkka > 3000 AND NOT saika < '1960-01-01'

false

false

false

NOT

AND

true

false

OR

false

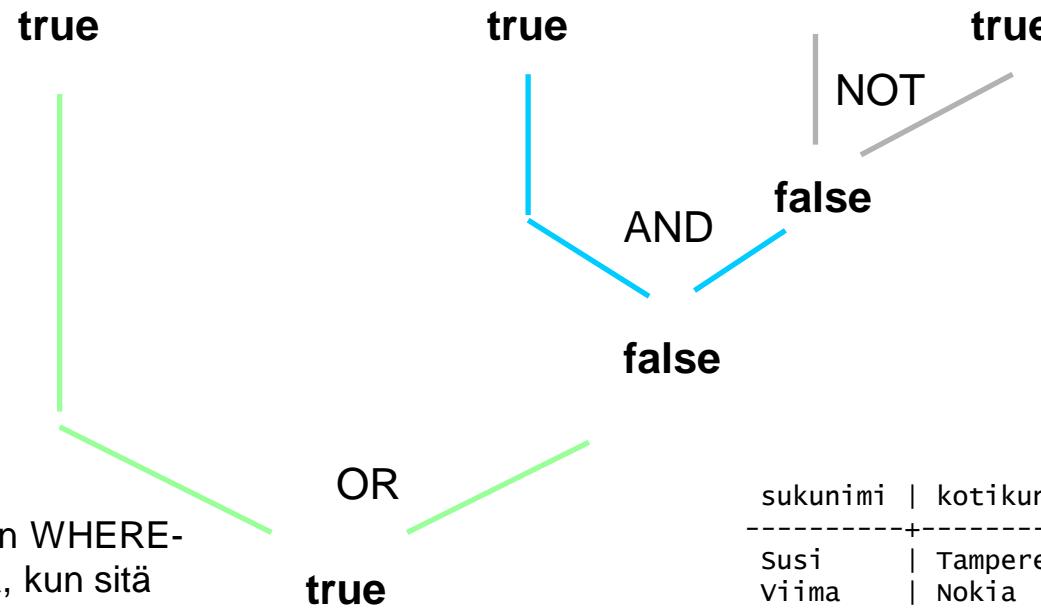
Minkä totuusarvon WHERE-osan ehto tuottaa, kun sitä sovelletaan työntekijän ttnro 99 riviin? false

sukunimi	kotikunta	palkka	saika
Susi	Tampere	5500.00	1957-11-10
Viima	Nokia	4000.50	1975-12-08
Puro	Tampere	3000.00	1985-01-09
Joki	Lempäälä	4300.00	1961-06-20

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

```
SELECT sukunimi, kotikunta, palkka, saika
FROM tyontekija
WHERE kotikunta = 'Tampere' OR palkka > 3000 AND NOT saika < '1960-01-01';
```

kotikunta = 'Tampere' OR palkka > 3000 AND NOT saika < '1960-01-01'



Minkä totuusarvon WHERE-osan ehto tuottaa, kun sitä sovelletaan työntekijän ttnro 88 riviin? true

sukunimi	kotikunta	palkka	saika
Susi	Tampere	5500.00	1957-11-10
Viima	Nokia	4000.50	1975-12-08
Puro	Tampere	3000.00	1985-01-09
Joki	Lempäälä	4300.00	1961-06-20

Loogiset operaatiot: laskentajärjestys

tyontekija

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

Haetaan Hallinto- ja Tutkimus-osastojen työntekijöiden tietoja.

```
SELECT ttnro, sukunimi, etunimi
FROM tyontekija, osasto
WHERE osastonro = onro AND
      (onimi = 'Hallinto' OR
       onimi = 'Tutkimus');
```

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

ttnro	sukunimi	etunimi
33	Viima	Ville
12	Puro	Pekka
98	Joki	Jenni
99	Kivi	Alli

FROM tyontekija, osasto

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

```
WHERE osastonro = onro AND
      (onimi = 'Hallinto' OR
       onimi = 'Tutkimus')
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

```
WHERE osastonro = onro AND  
      (onimi = 'Hallinto' OR  
       onimi = 'Tutkimus')
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
98	Jenni		4	88	4	<i>Hallinto</i>
99	Alli		4	98	4	<i>Hallinto</i>
33	Ville		5	88	5	<i>Tutkimus</i>
12	Pekka		5	33	5	<i>Tutkimus</i>

tyontekija

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

Haetaan Hallinto- ja Tutkimus-osastojen työntekijöiden tietoja.

```
SELECT ttnro, sukunimi, etunimi
FROM tyontekija, osasto
WHERE osastonro = onro AND
      onimi = 'Hallinto' OR
      onimi = 'Tutkimus';
```

ttnro	sukunimi	etunimi
88	Susi	Jukka
33	Viima	Ville
12	Puro	Pekka
98	Joki	Jenni
98	Joki	Jenni
99	Kivi	Alli
99	Kivi	Alli

(7 rows)

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

Ei näin,
tämä
ratkaisu ei
tuota
oikeaa
tulosta

FROM tyontekija, osasto

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

```
WHERE osastonro = onro AND
      onimi = 'Hallinto' OR
      onimi = 'Tutkimus'
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

Kysely
tuottaa
väärän
tuloksen

```
WHERE osastonro = onro AND  
onimi = 'Hallinto' OR  
onimi = 'Tutkimus'
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
98	Jenni		4	88	4	<i>Hallinto</i>
99	Alli		4	98	4	<i>Hallinto</i>
88	Jukka		1		5	<i>Tutkimus</i>
33	Ville		5	88	5	<i>Tutkimus</i>
12	Pekka		5	33	5	<i>Tutkimus</i>
98	Jenni		4	88	5	<i>Tutkimus</i>
99	Alli		4	98	5	<i>Tutkimus</i>

Kysely
tuottaa
väärän
tuloksen

tyontekija

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

Haetaan Hallinto- ja Tutkimus-osastojen työntekijöiden tietoja.

```
SELECT ttnro, sukunimi, etunimi
FROM tyontekija INNER JOIN osasto
    ON osastonro = onro
WHERE onimi = 'Hallinto' OR
      onimi = 'Tutkimus';
```

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

Kysely OK.

ttnro	sukunimi	etunimi
33	Viima	Ville
12	Puro	Pekka
98	Joki	Jenni
99	Kivi	Alli

```
FROM tyontekija INNER JOIN osasto
ON osastonro = onro
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
33	Ville		5	88	1	Pääkonttori
12	Pekka		5	33	1	Pääkonttori
98	Jenni		4	88	1	Pääkonttori
99	Alli		4	98	1	Pääkonttori
88	Jukka		1		4	Hallinto
33	Ville		5	88	4	Hallinto
12	Pekka		5	33	4	Hallinto
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
88	Jukka		1		5	Tutkimus
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus
98	Jenni		4	88	5	Tutkimus
99	Alli		4	98	5	Tutkimus

```
FROM tyontekija INNER JOIN osasto  
ON osastonro = onro
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
88	Jukka		1		1	Pääkonttori
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus

```
WHERE onimi = 'Hallinto' OR  
      onimi = 'Tutkimus'
```

ttnro	etunimi	...	osastonro	esimiesnro	onro	onimi
98	Jenni		4	88	4	Hallinto
99	Alli		4	98	4	Hallinto
33	Ville		5	88	5	Tutkimus
12	Pekka		5	33	5	Tutkimus



Tietokantojen perusteet

SQL – Joukko-operaatioita

Joukko-operaatioita

- SQL-taulut ovat joukkoja tai monijoukkoja.
 - Joukko: sama alkio esiintyy vain kerran
 - Taulu, jolle on määritelty pääavain, on joukko.
 - Monijoukko: sama alkio voi esiintyä useammin kuin kerran
- Kyselyissä voidaan käyttää joukko-operaatioita.
 - UNION (yhdiste eli unioni)
 - INTERSECT (leikkaus)
 - EXCEPT (erotus)
 - IN (kuuluuko arvo tai arvojen yhdistelmä joukkoon)
 - Myös moninkertaisien esiintymien poistoon käytettävä DISTINCT-määre liittyy joukkojen käsittelyyn.
 - Lisää joukko-operaatioita käydään läpi seuraavalla luentokerralla.

Joukko-operaatioita: DISTINCT

- DISTINCT-määre muuttaa tulostaulun joukoksi poistamalla siitä duplikaattirivit.
 - Huom!** DISTINCT-määre kohtelee tyhjäarvoja (NULL) yhtä suurina.

```
SELECT kotikunta  
FROM tyontekija;
```

kotikunta

Tampere
Nokia
Tampere
Lempäälä
Nokia

```
SELECT DISTINCT kotikunta  
FROM tyontekija;
```

kotikunta

Lempäälä
Nokia
Tampere

Joukko-operaatioita: IN

- IN-operaattorilla voidaan tutkia, kuuluuko arvo (tai arvojen yhdistelmä) joukkoon.
 - Arvojen yhdistelmän tutkiminen ei ole käytössä SQLItessa.

```
SELECT ttnro, sukunimi  
FROM tyontekija  
WHERE kotikunta IN ('Lempäälä', 'Parkano', 'Tampere');
```

ttnro	sukunimi
88	Susi
12	Puro
98	Joki

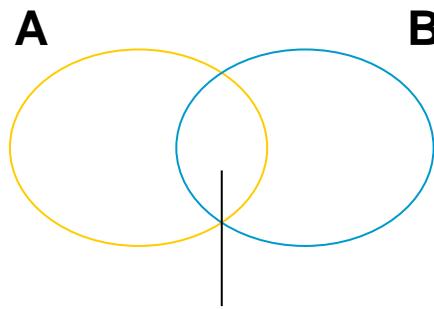
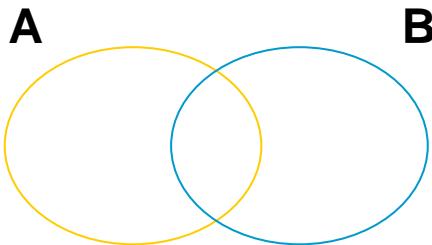
Joukon arvot lueteltu eksplisiittisesti

```
SELECT ttnro, sukunimi  
FROM tyontekija  
WHERE (sukunimi, kotikunta) IN (('Viima', 'Nokia'),  
('Viima', 'Tampere'));
```

ttnro	sukunimi
33	viima

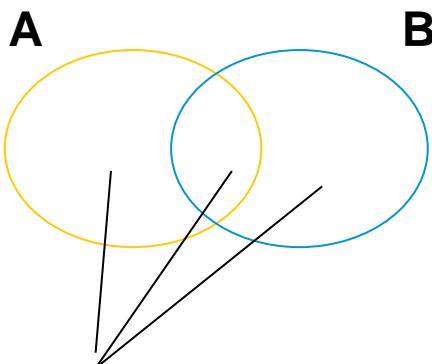
SQL joukko-operaatioita: 4

Joukko-operaatioita



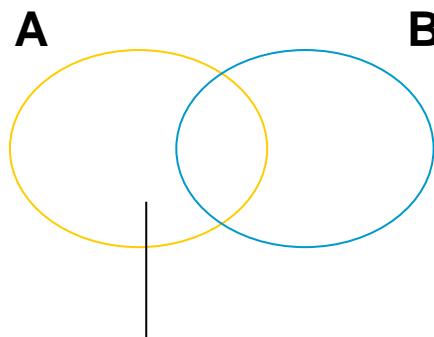
Leikkaus

$$A \cap B = \{x : x \in A \text{ ja } x \in B\}$$



Yhdiste eli unioni

$$A \cup B = \{x : x \in A \text{ tai } x \in B\}$$



Erotus

$$A - B = \{x : x \in A \text{ ja } x \notin B\}$$

Merkitään myös $A \setminus B$

SQL joukko-operaatioita: 5

Joukko-operaatioita

Olkoon joukot $A = \{2, 4, 8\}$ ja $B = \{2, 3\}$. Tällöin

$$A \cup B = \{2, 3, 4, 8\}$$

$$A \cap B = \{2\}$$

$$A - B = \{4, 8\}$$

Joukko-operaatioita: UNION, INTERSECT, EXCEPT

taulu1

nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

taulu2

nimi
Bertta
Cecilia
Cecilia

Kahden kyselyn (SELECT-lauseen) tuottamia tulostauluja (monijoukkoja) voidaan käsitellä joukko-operaatioilla.

```
SELECT nimi  
FROM taulu1  
UNION  
SELECT nimi  
FROM taulu2;
```

```
nimi  
-----  
Alfred  
Bertta  
Cecilia
```

```
SELECT nimi  
FROM taulu1  
INTERSECT  
SELECT nimi  
FROM taulu2;
```

```
nimi  
-----  
Bertta  
Cecilia
```

```
SELECT nimi  
FROM taulu1  
EXCEPT  
SELECT nimi  
FROM taulu2;
```

```
nimi  
-----  
Alfred
```

Joukko-operaatioita

- Operaatot UNION, INTERSECT ja EXCEPT poistavat duplikaattirivit.
- Joukkokäsittelyn sijasta tulokseen voidaan sisällyttää myös moninkertaiset esiintymät käyttämällä ALL-määrettä joukko-operaation jäljessä.
 - UNION ALL
 - On käytössä sekä SQLItessa että PostgreSQL:ssä
 - INTERSECT ALL
 - Ei ole käytössä SQLItessa
 - EXCEPT ALL
 - Ei ole käytössä SQLItessa

UNION ALL

```
SELECT nimi  
FROM taulu1  
UNION  
SELECT nimi  
FROM taulu2;
```

nimi

Alfred
Bertta
Cecilia

Kahden kyselyn tuloksista on muodostettu unioni.
Moninkertaiset esiintymät on poistettu.

taulu1
nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

taulu2
nimi
Bertta
Cecilia
Cecilia

Joukkokäsittelyn sijasta sisällytetään kaikki rivit tulokseen lisäämällä UNION-operaation perään ALL-määre.

```
SELECT nimi  
FROM taulu1  
UNION ALL  
SELECT nimi  
FROM taulu2;
```

nimi

Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia
Bertta
Cecilia
Cecilia

Moninkertaiset esiintymät on säilytetty.

INTERSECT ALL

taulu1

nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

taulu2

nimi
Bertta
Cecilia
Cecilia

```
SELECT nimi  
FROM taulu1  
INTERSECT  
SELECT nimi  
FROM taulu2;
```

```
SELECT nimi  
FROM taulu1  
INTERSECT ALL  
SELECT nimi  
FROM taulu2;
```

nimi

Bertta
Cecilia

nimi

Bertta
Cecilia
Cecilia

EXCEPT ALL

taulu1

nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

taulu2

nimi
Bertta
Cecilia
Cecilia

```
SELECT nimi  
FROM taulu1  
EXCEPT  
SELECT nimi  
FROM taulu2;
```

nimi

Alfred

```
SELECT nimi  
FROM taulu1  
EXCEPT ALL  
SELECT nimi  
FROM taulu2;
```

nimi

Alfred
Bertta
Cecilia

osasto

onro	onimi	johtajanro	aloituspvm
1	Pääkonttori	88	1989-06-19
4	Hallinto	98	1992-01-01
5	Tutkimus	33	2000-05-22

osasto_sijainti

onro	osijainti
1	Tampere
4	Tampere
5	Tampere
5	Lempäälä
5	Nokia

Joukko-operaatioita

Haetaan niiden osastojen nimet, joilla on sijaintipaikka Tampereella tai Nokialla.

```
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijanti='Tampere'
UNION
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijanti='Nokia';
```

onimi

Hallinto

Pääkonttori

Tutkimus

Joukko-operaatioita

Haetaan niiden osastojen nimet, joilla on sijaintipaikka Tampereella ja Nokialla.

```
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijanti='Tampere'
INTERSECT
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijanti='Nokia';
```

onimi

Tutkimus

Joukko-operaatioita

Haetaan niiden osastojen nimet, joilla on sijaintipaikka Tampereella muttei Nokiaalla. Järjestetään tulosrivit nousevaan järjestykseen.

```
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Tampere'
EXCEPT
SELECT onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Nokia'
ORDER BY onimi;
```

onimi

Hallinto
Pääkonttori

Koska onimi-niminen sarake löytyy vain osasto-taulusta, ei tarvitse käyttää taulun nimeä tarkenteena. PostgreSQL:ssä taulun nimeä ei edes voisi käyttää tarkenteena, koska kyseessä on joukko-operaation UNION-, EXCEPT- tai INTERSECT sisältävä kysely (ks. kalvo 19).

Joukko-operaatioita

- Joukko-operaatioissa käsiteltävien joukkojen on oltava "unioniyhteensopivia", jotta operaatiot voidaan suorittaa.
 - sama määrä sarakkeita, tietotyypiltään samat sarakkeet
 - sarakkeet samassa järjestyksessä
- Tulostaulun sarakkeiden nimet otetaan ensimmäisestä joukosta.

```
SELECT o.onro AS numero, onimi AS nimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Tampere'
UNION
SELECT o.onro, onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Nokia'
ORDER BY numero DESC;

numero |    nimi
-----+-----
      5 | Tutkimus
      4 | Hallinto
      1 | Pääkonttori
```

Joukko-operaatioita

- Jos tulosrivit halutaan järjestää, **ORDER BY -määrä** sijoitetaan joukko-operaatoiden jälkeen **viimeisen SELECT-lauseen jälkeen**.
 - **ORDER BY -määrättä** on **käytettävä aina, kun tulostaulun rivit halutaan järjestää**. Vaikka joukko-operaatiot näyttäisivät järjestävän tulosrivit, tähän ei saa luottaa, vaan on käytettävä ORDER BY -määrättä!

```
SELECT o.onro AS numero, onimi AS nimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Tampere'
UNION
SELECT o.onro, onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Nokia'
ORDER BY numero DESC;
```

numero	nimi
5	Tutkimus
4	Hallinto
1	Pääkonttori

Joukko-operaatioita

- Jos kyselyssä käytettävissä tauluissa on useita samannimisiä sarakkeita ja niistä jotakin halutaan käyttää ORDER BY – määreessä, voi sarakkeen nimetä uudelleen SELECT-osassa.
 - Syy: Eri järjestelmät käyttäytyvät eri tavoin ORDER BY -määreen evaluoinnin suhteeseen. (selitys jatkuu seuraavalla sivulla)
 - Uudelleennimetyt sarakkeen käyttö toimii sekä PostgreSQL:ssä että SQLite:ssa.

```
SELECT o.onro AS numero, onimi AS nimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Tampere'
UNION
SELECT o.onro, onimi
FROM osasto o, osasto_sijainti os
WHERE o.onro = os.onro AND osijainti='Nokia'
ORDER BY numero DESC;
```

numero	nimi
5	Tutkimus
4	Hallinto
1	Pääkonttori

Joukko-operaatioita

- Eri järjestelmät käyttäytyvät eri tavoin ORDER BY -määreen evaluoinnin suhteen.
 - **PostgreSQL:ssä UNION-, INTERSECT- ja EXCEPT-joukko-operaatioita hyödyntävässä kyselyssä ORDER BY -listassa ei voi käyttää taulujen nimiä tarkenteina.**
 - Joukko-operaatiot käsittelevät tulostauluja sarakenimineen, taulut nimineen eivät ole enää käytettäväissä ORDER BY -määreessä.
 - **SQLissa** puolestaan tämä on **mahdollista ja taulujen nimiä on käytettävä tarkenteena**, jos kyselyssä käytettäväissä tauluissa on samanimisiä sarakkeita ja niistä jotakin halutaan käyttää ORDER BY – määreessä.
 - Tulostaulun sarakkeen uudelleennimeäminen SELECT-osassa ja käyttäminen ORDER BY -määreessä toimii molemmissa järjestelmissä.
 - Myös tulostaulun sarakkeen järjestysnumeron (numerointi 1:stä alkaen vasemmalta oikealle) toimii molemmissa järjestelmissä.
 - ORDER BY 1

Joukko-operaatioiden ketjuttaminen

- Joukko-operaatioita voidaan ketjuttaa, esim. kyselyn
select-lause1 UNION select-lause2 UNION select-lause3
suoritusjärjestys on vasemmalta oikealle:
(select-lause1 UNION select-lause2) UNION select-lause3
- SQLItessa joukko-operaatioilla on sama prioriteetti, kun taas PostgreSQL:ssä
 - 1. INTERSECT (korkein prioriteetti)
 - 2. UNION ja EXCEPT (näillä sama prioriteetti)

Kyselyn

select-lause1 UNION select-lause2 INTERSECT select-lause3

suoritusjärjestys on siis SQLItessa

(select-lause1 UNION select-lause2) INTERSECT select-lause3

ja PostgreSQL:ssä

select-lause1 UNION (select-lause2 INTERSECT select-lause3)

Joukko-operaatioiden ketjuttaminen

- Tarkista jatkossa joukko-operaatioiden prioriteettijärjestys samoin kuin mahdollisuus käyttää sulkuja yhdistetyissä (compound) SELECT-lauseissa käyttämäsi tietokannanhallintajärjestelmän manuaalista.
 - PostgreSQL:ssä voi käyttää sulkuja, SQLitessä puolestaan ei.



Tietokantojen perusteet

SQL – Koostekyselyjä

Aggregointifunktiot

- Aggregointi-, kooste- eli yhteenvetofunktioilla tarkoitetaan operaatioita, jotka tekevät koosteita (yhteenvetuja) tietokantaan tallennetuista tiedoista.
 - COUNT - lukumäärä
 - SUM - summa
 - MAX - suurin arvo
 - MIN - pienin arvo
 - AVG - aritmeettinen keskiarvo
 - COUNT riveille, kaikentyyppisille sarakkeille
 - SUM, AVG numeerisille tietotyypeille
 - MIN, MAX tietotyypeille, joille pystytään määräämään järjestys (myös merkkijonoille, päivämääritteille)
- Aggregointifunktio(i)ta soveltava koostekysely (yhteenvetokysely) voi tuottaa **yhden tulosriven** koko taulun datan perusteella tai **useita ryhmäkohtaisia tulosrivejä**, jotka on laskettu kunkin ryhmän datan perusteella.

Aggregointifunktiot

```
SELECT AVG(palkka), SUM(palkka), COUNT(palkka)
FROM tyontekija;
```

avg	sum	count
3860.1000000000000000	19300.50	5

On muodostettu yksi tulosrivi taulun kaikkien rivien perusteella.

```
SELECT AVG(palkka) AS palkka_ka,
       SUM(palkka) AS palkkasumma,
       COUNT(palkka) AS lkm
FROM tyontekija;
```

palkka_ka	palkkasumma	lkm
3860.1000000000000000	19300.50	5

Eri järjestelmät nimeävät tulossarakkeet eri tavoin.

```
SELECT ROUND(AVG(palkka),2) AS keskipalkka
FROM tyontekija;
```

keskipalkka
3860.10

Pyöristetään keskipalkka kahden desimaalin tarkkuuteen ROUND-funktiolla.

Aggregointifunktiot

COUNT(*)

- Lasketaan rivien lukumäärä.

COUNT(sarakenimi)

- SQLite ja PostgreSQL: Lasketaan niiden ei-tyhjien arvojen (ts. muiden kuin NULL-arvojen) määrä ko. sarakkeessa.

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

```
SELECT COUNT(*)  
FROM tyontekija;
```

count

5

```
SELECT COUNT(palkka)  
FROM tyontekija;
```

count

5

```
SELECT COUNT(esimiesnro)  
FROM tyontekija;
```

count

4

Aggregointifunktiot

- Summaa, keskiarvoa, minimiä ja maksimia laskettaessa ei huomioida tyhjäarvoja.
- Jos sarakkeessa on vain tyhjäarvoja
 - COUNT(sarakenimi) antaa tulokseksi nollan (0)
 - muut funktiot antavat tulokseksi tyhjäarvon (null)
- Huom! Jatkossa tarkista aggregointifunktioiden toiminta käyttämäsi tietokannanhallintajärjestelmän manuaalista (esim. sarakkeiden nimeäminen ja COUNT(sarakenimi)-funktion käytäytyminen NULL-arvojen suhteen)

Aggregointifunktiot

- COUNT-, SUM- ja AVG-operaatioihin voidaan lisätä mukaan **DISTINCT**-määre, jolloin laskennassa käsitellään vain toisistaan poikkeavia arvoja (duplikaattiarvot eliminoidaan ennen aggregointifunktion soveltamista).

```
SELECT COUNT(DISTINCT kotikunta) AS eri_kotikuntia  
FROM tyontekija;
```

eri_kotikuntia

3

Aggregointifunktiot - Ryhmittely

- Aggregointifunktioiden yhteydessä voidaan käyttää GROUP BY ja HAVING -määreitä.
- **GROUP BY** -määreellä kerrotaan, minkä sarakkeiden suhteen tehdään ryhmittely yhteenveto-operaatioita varten.
 - Kaikki ne rivit, joilla on samat arvot annetuissa sarakkeissa, muodostavat oman ryhmänsä.
 - **Huom!** Tässä kohtaa **tyhjäarvot käsitellään samana arvona!!!!**
 - Nyt muodostetaan yksi tulosrivi yhtä ryhmää kohden.
 - Tulossarakkeissa voi olla aggregointifunktioiden lisäksi vain niitä sarakkeita, jotka esiintyvät GROUP BY -määreessä.

Aggregointifunktiot

```
SELECT kotikunta, COUNT(kotikunta) AS 1km  
FROM tyontekija  
GROUP BY kotikunta;
```

Aggregointifunktioiden tuottamien sarakkeiden lisäksi **tulossarakkeina** voi olla vain GROUP BY -määreessä annettuja sarakkeita.

ttnro	etunimi	sukunimi	saika	kotikunta	palkka	puhelin	osastonro	esimiesnro
98	Jenni	Joki	1961-06-20	Lempäälä	4300.00	444 4488	4	88

33	Ville	Viima	1975-12-08	Nokia	4000.50	444 4343	5	88
99	Alli	Kivi	1988-07-19	Nokia	2500.00	444 5555	4	98

88	Jukka	Susi	1957-11-10	Tampere	5500.00	444 1234	1	
12	Pekka	Puro	1985-01-09	Tampere	3000.00		5	33

kotikunta | 1km

-----+-----
Lempäälä | 1
Tampere | 2
Nokia | 2

Ryhmittely tehdään kotikunta-sarakkeen arvojen suhteen.

Tulostaulussa yksi rivi kullekin ryhmälle.

SQL - koostekyselyjä: 8

Aggregointifunktiot – ehtoja ryhmille

- HAVING-lauseella voidaan antaa ehtoja, jotka GROUP BY -operaatiolla tehdyn ryhmän on toteutettava, jotta ryhmä otetaan tulostauluun mukaan.
 - Ehdoissa testataan ryhmälle lasketun aggregointifunktion arvoa.
- HAVING COUNT(*) > 1 valitsee vain sellaiset GROUP BY -operaatiolla määritellyt ryhmät, joissa on enemmän kuin yksi rivi.

```
SELECT kotikunta, COUNT(kotikunta) AS 1km  
FROM tyontekija  
GROUP BY kotikunta  
HAVING COUNT(*) > 1;
```

kotikunta		1km
Tampere		2
Nokia		2

Tulokseen mukaan ryhmät, joissa vähintään 2 riviä

Kyselyn evaluointi

- SQL-kyselyn yleisrakenne
 - SELECT tulostietomäärittely
 - FROM taulut
 - [WHERE liitos- ja valintaehdot]
 - [GROUP BY ryhmittelytekijät]
 - [HAVING ryhmärajoitteet]
 - [ORDER BY järjestystekijät]
- Yksinkertainen kyselynevaluointialgoritmi: kyselyn osien suoritusjärjestys
 1. FROM-osa
 2. WHERE-osa
 3. GROUP BY -osa
 4. HAVING-osa
 5. SELECT-osa
 6. ORDER BY -osa

Huom! Tietokannanhallintajärjestelmä ei evaluoi kyselyä tällä tavoin – algoritmin tarkoituksena on auttaa ymmärtämään, miten kyselyn tulos muodostuu.

Kyselyn evaluointi

1. FROM-osa

- Jos FROM-osassa ei ole liitosoperaatioita,
 - muodostetaan annettujen **taulujen karteesinen tulo eli ristitulo**, joka on tämän vaiheen tulostaulu.
- Jos FROM-osassa on liitosoperaatio(ita),
 - muodostetaan liitoksen (karteesinen tulo ja liitosehto) antama tulostaulu ja lisätään tulostauluun mahdollisen ulkoliitoksen antamat "lisä rivit".
 - (Käydään läpi kaikki liitosoperaatiot vastaavalla tavalla.)

2. WHERE-osa

- Poistetaan edellisen vaiheen tulostaulusta ne rivit, jotka eivät täytä liitos- ja valintaehdoja.
- Huom! Jos FROM-osassa on annettu useita tauluja (ilman liitosoperaatioita), tarvitaan yleensä liitosehto(ja) taulujen tietojen yhdistämiseksi. Jos tauluja on N kappaletta, tarvitaan yleensä N-1 liitosehtoa. (Huom. Liitosehto voi koostua alkeisehdoista.)

Kyselyn evaluointi

3. GROUP BY -osa

- Jaetaan rivit ryhmiin ryhmittelytekijöiden (sarakkeiden) mukaisesti.

4. HAVING-osa

- Poistetaan ne ryhmät, jotka eivät täytä ryhmille asetettuja rajoitteita (ehtoja).

5. SELECT-osa

- Lasketaan aggregointifunktioiden arvot kullekin ryhmälle.
- Poistetaan kaikki sarakkeet, jotka eivät esiinny SELECT-listassa.
- Tuotetaan yksi tulosrivi ryhmää kohden.

6. ORDER BY -osa

- Järjestetään rivit järjestystekijöiden (sarakkeiden) mukaisesti.

Aggregointifunktiot - kyselyn evaluointi

Haetaan osaston numero, osaston nimi ja osaston valvomien projektien lukumäärä. Tulokseen otetaan mukaan ainoastaan sellaiset osastot, jotka valvovat yhtä useampaa projektia. Tulosrivit järjestetään osastojen nimien mukaan nousevaan järjestykseen.

```
SELECT o.onro, o.onimi, COUNT(*) AS projektien_1km
FROM osasto o, projekti p
WHERE o.onro = p.onro
GROUP BY o.onro, o.onimi
HAVING COUNT(*) > 1
ORDER BY o.onimi;
```

onro	onimi	projektien_1km
4	Hallinto	2
5	Tutkimus	3

Aggregointifunktiot - kyselyn evaluointi

Vaiheen 1 (FROM) tulostaulu

FROM osasto o, projekti p

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
1	Pääkonttori	88	1989-06-19	1	Tuote X	5
4	Hallinto	98	1992-01-01	1	Tuote X	5
5	Tutkimus	33	2000-05-22	1	Tuote X	5
1	Pääkonttori	88	1989-06-19	2	Tuote Y	5
4	Hallinto	98	1992-01-01	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
1	Pääkonttori	88	1989-06-19	3	Tuote Z	5
4	Hallinto	98	1992-01-01	3	Tuote Z	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
1	Pääkonttori	88	1989-06-19	10	Uudet edut	4
4	Hallinto	98	1992-01-01	10	Uudet edut	4
5	Tutkimus	33	2000-05-22	10	Uudet edut	4
1	Pääkonttori	88	1989-06-19	20	TYKY-liikunta	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4
5	Tutkimus	33	2000-05-22	20	TYKY-liikunta	4

Aggregointifunktiot - kyselyn evaluointi

Vaiheen 2 (WHERE) tulostaulu:

karteesisesta tulosta poistettu ne rivit, jotka eivät täytä liitos- ja valintaehdoja

WHERE o.onro = p.onro

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Aggregointifunktiot - kyselyn evaluointi

Vaiheen 3 (GROUP BY) tulostaulu:

rivist jaettu ryhmiin ryhmittelytekijöiden (sarakkeiden) mukaisesti
(tyhjiä rivejä on käytetty havainnollistamaan ryhmittelyä)

GROUP BY o.onro, o.onimi

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Aggregointifunktiot - kyselyn evaluointi

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Vaiheen 4 (HAVING) tulostaulu:

tulostaulusta poistetaan ne ryhmät, jotka eivät täytä ryhmille asetettuja rajoitteita
(Tässä tapauksessa ei poistettu yhtään ryhmää.)

HAVING COUNT(*) > 1

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Aggregointifunktiot - kyselyn evaluointi

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Vaiheen 5 (SELECT) tulostaulu:

Laskettu aggregointifunktioiden arvot kullekin ryhmälle.

Poistettu kaikki sarakkeet, jotka eivät esiinny SELECT-listassa.

Tuotettu yksi tulosrivi ryhmää kohden.

```
SELECT o.onro, o.onimi, COUNT(*) AS projektien_1km
```

onro	onimi	projektien_1km
5	Tutkimus	3
4	Hallinto	2

Aggregointifunktiot - kyselyn evaluointi

onro	onimi	projektien_1km
5	Tutkimus	3
4	Hallinto	2

Vaiheen 6 (ORDER BY) tulostaulu ja kyselyn lopullinen tulostaulu:

Lajiteltu tulosrivit osaston nimen mukaisesti nousevaan järjestykseen.

ORDER BY o.onimi

onro	onimi	projektien_1km
4	Hallinto	2
5	Tutkimus	3

Aggregointifunktiot – kyselyn evaluointi

Haetaan kullekin osastolle osaston numero, osaston nimi ja osaston valvomien projektien lukumäärä. Tulokseen halutaan kaikki osastot – myös sellaiset, joilla ei ole yhtään projektiä valvottavana. Tulosrivit järjestetään projektien lukumäärän mukaan nousevaan järjestykseen.

```
SELECT o.onro, o.onimi, COUNT(pnro) AS projektien_1km
FROM osasto o LEFT OUTER JOIN projekti p
    ON o.onro = p.onro
GROUP BY o.onro, o.onimi
ORDER BY projektien_1km;
```

onro	onimi	projektien_1km
1	Pääkonttori	0
4	Hallinto	2
5	Tutkimus	3

FROM

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
1	Pääkonttori	88	1989-06-19	1	Tuote X	5
4	Hallinto	98	1992-01-01	1	Tuote X	5
5	Tutkimus	33	2000-05-22	1	Tuote X	5
1	Pääkonttori	88	1989-06-19	2	Tuote Y	5
4	Hallinto	98	1992-01-01	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
1	Pääkonttori	88	1989-06-19	3	Tuote Z	5
4	Hallinto	98	1992-01-01	3	Tuote Z	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
1	Pääkonttori	88	1989-06-19	10	Uudet edut	4
4	Hallinto	98	1992-01-01	10	Uudet edut	4
5	Tutkimus	33	2000-05-22	10	Uudet edut	4
1	Pääkonttori	88	1989-06-19	20	TYKY-liikunta	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4
5	Tutkimus	33	2000-05-22	20	TYKY-liikunta	4

FROM-vaiheen tulostaulu

FROM osasto o LEFT OUTER JOIN projekti p ON o.onro = p.onro

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
1	Pääkonttori	88	1989-06-19			
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

GROUP BY -vaiheen tulostaulu:

rivist jaettu ryhmiin ryhmittelytekijöiden (sarakkeiden) mukaisesti
(tyhjiä rivejä on käytetty havainnollistamaan ryhmittelyä)

GROUP BY o.onro, o.onimi

onro	onimi	johtajanro	aloituspvm	pnro	pnimi	onro
5	Tutkimus	33	2000-05-22	1	Tuote X	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
1	Pääkonttori	88	1989-06-19			
4	Hallinto	98	1992-01-01	10	Uudet edut	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4

Aggregointifunktiot – kyselyn evaluointi

SELECT-vaiheen tulostaulu:

Laskettu aggregointifunktoiden arvot kullekin ryhmälle.

Poistettu kaikki sarakkeet, jotka eivät esiinny SELECT-listassa.

Tuotettu yksi tulosrivi ryhmää kohden.

```
SELECT o.onro, o.onimi, COUNT(pnro) AS projektien_1km
```

onro	onimi	projektien_1km
5	Tutkimus	3
1	Pääkonttori	0
4	Hallinto	2

Aggregointifunktiot – kyselyn evaluointi

ORDER BY -vaiheen tulostaulu ja kyselyn lopullinen tulostaulu:

Lajiteltu tulosrivit projektien lukumäärän mukaan nousevaan järjestykseen.

ORDER BY projektien_1km

onro	onimi	projektien_1km
1	Pääkonttori	0
4	Hallinto	2
5	Tutkimus	3



Tietokantojen perusteet

SQL: LIKE-operaattori
BETWEEN-operaattori
Aritmeettisista operaatioista

WHERE-osa: LIKE

- Merkkijonojen vertailussa voidaan käyttää **LIKE**- tai **NOT LIKE** - operaattoria.
 - Asetetaan ehtoja merkkijonojen osille.

sarake LIKE 'vakiomerkkijono'

- Vakiomerkkijono
 - yksinkertaisissa lainausmerkeissä (hipsissa)
 - %-merkki korvaa nolla tai useampia merkkejä (mikä tahansa merkkijono)
 - _-merkki korvaa yhden merkin (mikä tahansa merkki)
 - Merkkejä % ja _ voi esiintyä vertailussa käytettävässä vakiomerkkijonossa mielivaltainen määrä.

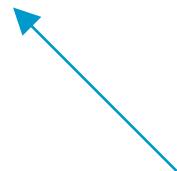
WHERE-osa: LIKE

Haetaan nimet, joiden toinen kirjain on e.

```
SELECT etunimi  
FROM tyontekija  
WHERE etunimi LIKE '_e%';
```

etunimi

Pekka
Jenni



Pekka: _ täsmäytyy merkkiin P ja % täsmäytyy merkkijonoon kka

Jenni: _ täsmäytyy merkkiin J ja % täsmäytyy merkkijonoon nni

Haetaan nimet, jotka eivät ole viisikirjaimisia.

```
SELECT etunimi  
FROM tyontekija  
WHERE etunimi NOT LIKE '_____';
```

etunimi

Alli

Viisi alaviivaa _



WHERE-osa: BETWEEN

- BETWEEN-operaattorilla voidaan tutkia, kuuluuko arvo tiettylle välille
 - NOT BETWEEN: arvo ei kuulu tiettylle välille

```
SELECT etunimi, sukunimi, ttnro  
FROM tyontekija  
WHERE ttnro BETWEEN 50 AND 99;
```

etunimi	sukunimi	ttnro
Jukka	Susi	88
Jenni	Joki	98
Alli	Kivi	99

Työntekijänumero välillä 50 - 99

```
SELECT sukunimi, etunimi  
FROM tyontekija  
WHERE sukunimi BETWEEN 'Joki' AND 'Puro';
```

sukunimi	etunimi
Puro	Pekka
Joki	Jenni
Kivi	Alli

Sukunimi välillä Joki ja Puro

Aritmeettiset operaatiot

- Aritmeettisissa ilmauksissa on käytettävissä aritmeettiset operaatiot +, -, *, / ja sulut
 - yhteen-, vähennys-, kerto- ja jakolasku
- Jos aritmeettisessa ilmauksessa on mukana tyhjäarvo, koko ilmauksen tulos on tyhjäarvo.

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE palkka + 200 >= 4500;
```

Aritmeettisen
ilmauksen
käyttäminen
ehdossa

```
UPDATE tyontekija  
SET palkka = palkka + 100;
```

Arvon muuttaminen
aritmeettisella
ilmauksella

Aritmeettiset operaatiot

```
SELECT sukunimi, palkka + 1000 AS korotettu  
FROM tyontekija;
```

sukunimi	korotettu
Susi	6500.00
Viima	5000.50
Puro	4000.00
Joki	5300.00
Kivi	3500.00

Näytetään palkat tulostaulussa tuhannella korotettuina.

Huom! tyontekija-taulun palkkasarakkeen arvot pysyvät entisellään.



Tietokantojen perusteet

Sisäkkäisistä kyselyistä

Sisäkkäisiä kyselyjä

- Alikysely on kysely, joka on upotettu toisen, "laajemman" kyselyn sisälle.
- Laajempaa kyselyä kutsutaan pääkyselyksi tai ulommaksi kyselyksi, alikyselyä kutsutaan myös sisemmäksi kyselyksi.
- Alikysely voi olla WHERE-, HAVING-, FROM- tai SELECT-osassa.
 - Tällä kurssilla käsitellään alikyselyjä, jotka ovat WHERE-, HAVING- tai FROM-osassa.

pääkysely → SELECT etunimi, sukunimi, ttnro
FROM tyontekija
WHERE ttnro IN
(SELECT huoltajanro
FROM huollettava);

alikysely

Sisäkkäisiä kyselyjä

- Alikysely voi palauttaa **yhdet** tai **useampia tulosrivejä**.
- Jos WHERE- tai HAVING-osan kysely voi palauttaa **useita rivejä**, on tällöin ulommassa kyselyssä käytettävä **joukko-operaatiota**.
 - IN
 - EXISTS
 - vertailuoperaattori ja ANY (tai SOME)
 - = ANY
 - > ANY
 - >= ANY
 - < ANY
 - <= ANY
 - <> ANY
 - vertailuoperaattori ja ALL
 - kuten ANY-määreen tapauksessa

HUOM. SQLitessa ei ole toteutettu ANY- ja ALL-avainsanoja.

SQL Sisäkkäisiä kyselyjä: 3

Alikysely WHERE-osassa

- Sisäkkäisen kyselyn evaluointistrategia, kun alikysely on WHERE-osassa:
 - Muodosta ulomman kyselyn FROM-osan antama tulostaulu kuten aiemmin (karteesinen tulo tai liitosoperaation tuottama taulu)
 - Kullekin edellisen vaiheen tulostaulun riville:
 - Testaa WHERE-osan ehto.
 - Ehdon testaamista varten (uudelleen)suorita alikysely.

Alikysely WHERE-osassa: IN

- IN-operaattorin avulla tutkitaan, kuuluuko arvo (tai arvojen yhdistelmä esim. PostgreSQL:ssä) alikyselyn tulokseen. IN palauttaa totuusarvon tosi, jos arvo kuuluu alikyselyn tulokseen.
 - Alikyselyn tulos on joukko tai monijoukko. (Nyt joukon alkioita ei luetella eksplisiittisesti.)

Kysely hakee tietoja työntekijöistä, joilla on huollettavia.

```
SELECT etunimi, sukunimi, ttnro
FROM tyontekija
WHERE ttnro IN
      (SELECT huoltajanro
       FROM huollettava);
```

etunimi	sukunimi	ttnro
Ville	Viima	33
Pekka	Puro	12
Jenni	Joki	98

Ulommassa kyselyssä tutkitaan, kuuluuko työntekijännumero (ttnro) alikyselyn tuottamaan tulostauluun.

Alikyselyn tuloksena työntekijännumerot (huoltajanro) huollettava-taulusta. Tässä alikyselyn tulos on monijoukko.

Alikysely WHERE-osassa: IN

Haetaan niiden työntekijöiden etu- ja sukunimet ja työntekijännumerot, joilla on Aamu-niminen huollettava.

```
SELECT etunimi, sukunimi, ttnro
FROM tyontekija
WHERE ttnro IN
      (SELECT huoltajanro
       FROM huollettava
       WHERE nimi = 'Aamu');
```

etunimi	sukunimi	ttnro
ville	viima	33
Pekka	Puro	12

Ulommassa kyselyssä tutkitaan, kuuluuko ttnro alikyselyn tuottamaan tulostauluun.

Alikyselyn tuloksena niiden henkilöiden työntekijännumerot, joilla on Aamu-niminen huollettava.

(huollettava-taulusta löytyvät huoltajanro-sarakkeen arvot, joille pätee, että rivillä oleva nimi = 'Aamu').

Alikysely WHERE-osassa: NOT IN

- NOT IN palauttaa totuusarvon tosi, jos arvo (tai arvojen yhdistelmä) ei kuulu alikyselyn tuottamaan tulokseen.

```
SELECT etunimi, sukunimi, ttnro  
FROM tyontekija  
WHERE ttnro NOT IN  
    (SELECT huoltajanro  
     FROM huollettava);
```

etunimi	sukunimi	ttnro
Jukka	Susi	88
Alli	Kivi	99

Kysely hakee niiden työntekijöiden tietoja, joilla ei ole huollettavia.

Alikysely WHERE-osassa: EXISTS

- EXISTS-operaattori testaa, sisältääkö alikyselyn tulos yhtään riviä. Operaattori palauttaa totuusarvon tosi, jos alikyselyn tulos ei ole tyhjä (ts. tulos sisältää yhden tai useamman rivin).

```
SELECT etunimi, sukunimi, ttnro
FROM tyontekija
WHERE EXISTS
  (SELECT *
   FROM huollettava
   WHERE tyontekija.ttnro = huollettava.huoltajanro);
```

etunimi	sukunimi	ttnro
ville	Viima	33
Pekka	Puro	12
Jenni	Joki	98

Kysely hakee niiden työntekijöiden tietoja, joilla on huollettavia.

Alikysely WHERE-osassa: EXISTS

- Aiemmissa kyselyissä (kalvot 5-7) alikysely voitiin suorittaa erillään pääkyselystä (ulomasta kyselystä).
- Nyt on kyseessä **kytketty alikysely**:
 - Alikyselyn WHERE-osan ehdossa viitataan ulommassa kyselyssä annetun taulun sarakkeeseen.
 - Alikysely on evaluoitava erikseen jokaista ulomman kyselyn FROM-osan tulostaulun riviä kohden.

```
SELECT etunimi, sukunimi, ttnro  
FROM tyontekija  
WHERE EXISTS  
(SELECT *  
  FROM huollettava  
  WHERE tyontekija.ttnro = huollettava.huoltajanro);
```



Alikyselyn tulos riippuu ulomman kyselyn "nykyisestä" rivistä: ehto
tyontekija.ttnro = huollettava.huoltajanro

Alikysely WHERE-osassa: EXISTS

Alikyselyn tulos riippuu ulomman kyselyn "nykyisestä" rivistä:
tyontekija.ttnro = **huollettava.huoltajanro**

```
SELECT etunimi,sukunimi, ttnro  
FROM tyontekija  
WHERE EXISTS  
(SELECT *  
  FROM huollettava  
 WHERE tyontekija.ttnro = huollettava.huoltajanro AND  
       nimi = 'Aamu');
```

Lisäksi mukana valintaehdot.

etunimi	sukunimi	ttnro
Ville	Viima	33
Pekka	Puro	12

Alikysely WHERE-osassa: NOT EXISTS

- NOT EXISTS palauttaa totuusarvon tosi, jos alikyselyn tulokseen ei sisällä yhtään riviä.

```
SELECT etunimi, sukunimi, ttnro  
FROM tyontekija  
WHERE NOT EXISTS  
(SELECT *  
  FROM huollettava  
 WHERE tyontekija.ttnro = huollettava.huoltajanro);
```

etunimi	sukunimi	ttnro
Jukka	Susi	88
Alli	Kivi	99

Kysely hakee niiden työntekijöiden tietoja, joilla ei ole huollettavia.

Useita alikyselyjä WHERE-osassa

- WHERE-osassa voi esiintyä useita alikyselyjä.

Kysely hakee tietoja esimiehenä toimivista työntekijöistä, jotka eivät osallistu mihinkään projektiin.

```
SELECT etunimi, sukunimi, ttnro
FROM tyontekija
WHERE ttnro IN (SELECT esimiesnro
                  FROM tyontekija)
      AND
      ttnro NOT IN
          (SELECT ttnro
             FROM osallistuu);
```

etunimi	sukunimi	ttnro
Jukka	Susi	88

Alikysely WHERE-osassa: Vertailuoperaattori ilman ANY- tai ALL-määrettä

- Kun alikysely palauttaa vain yhden tulosriven, vertailuoperaattoria voidaan käyttää ilman ANY- tai ALL-määrettä.

Haetaan niiden työntekijöiden etunimet, sukunimet ja palkat, joiden palkka on yhtä suuri kuin maksimipalkka.

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka = (SELECT MAX(palkka)  
                 FROM tyontekija);
```



Työntekijän palkan on oltava yhtä suuri kuin alikyselyn palauttaman maksimipalkkan.

etunimi	sukunimi	palkka
Jukka	Susi	5500.00

Alikysely WHERE-osassa: Vertailuoperaattori ilman ANY- tai ALL-määrettä

Haetaan niiden työntekijöiden etunimet, sukunimet ja palkat, joiden palkka on suurempi kuin minimipalkka.

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka > (SELECT MIN(palkka)  
                  FROM tyontekija);
```



Työntekijän palkan on oltava suurempi kuin alikyselyn palauttaman minimipalkkan.

etunimi	sukunimi	palkka
Jukka	Susi	5500.00
Ville	Viima	4000.50
Pekka	Puro	3000.00
Jenni	Joki	4300.00

Alikysely WHERE-osassa: Vertailuoperaattori ja ANY

- vertailuoperaattori ja ANY (SOME)
 - Joissakin järjestelmissä voidaan käyttää myös SOME-määrettä.
 - HUOM: Avainsanoja ANY ja SOME ei ole toteutettu SQLItessa.

Haetaan niiden työntekijöiden etunimet, sukunimet ja palkat, joiden palkka on suurempi kuin minimipalkka.

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka > ANY (SELECT palkka  
                      FROM tyontekija);
```



ANY: vertailuehdon on oltava **voimassa**
jonkin alikyselyn tuottaman tulostaulun
arvon suhteen

Alikysely WHERE-osassa: Vertailuoperaattori ja ANY

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka > ANY (SELECT palkka  
                      FROM tyontekija);
```

Testataan, onko palkka suurempi
kuin jokin alikyselyn tuloksen
palkoista.

Alikyselyn tuloksena
palkat tyontekija-
taulusta.

etunimi	sukunimi	palkka
Jukka	Susi	5500.00
Ville	Viima	4000.50
Pekka	Puro	3000.00
Jenni	Joki	4300.00

Alikysely WHERE-osassa: Vertailuoperaattori ja ALL

- vertailuoperaattori ja ALL

Haetaan niiden työntekijöiden etunimi, sukunimi ja palkka, joilla on korkein palkka.

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka >= ALL (SELECT palkka  
                      FROM tyontekija);
```



ALL: vertailuehdon
on oltava **voimassa**
kaikkien alikyselyn
tulostaulun **arvojen**
suhteen

Alikysely WHERE-osassa: Vertailuoperaattori ja ALL

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka >= ALL (SELECT palkka  
                      FROM tyontekija);
```

Testataan, onko palkka suurempi tai yhtä suuri kuin kaikki alikyselyn tuloksesta saatavat palkat.

Alikyselyn tuloksesta palkat tyontekijataulusta.

etunimi	sukunimi	palkka
Jukka	Susi	5500.00

Tässä kannan tilassa korkeimman palkan omaavia työntekijöitä vain yksi, mutta jossakin muussa tilassa heitä voisi olla useita

Alikysely WHERE-osassa Vertailuoperaattori ja ALL

HUOM!

Alla olevat kyselyt tuottavat esimerkkitietokannassa saman tuloksen, koska tyontekija-taulussa on jokaisella rivillä palkka-sarakkeessa ei-tyhjä arvo.

```
SELECT etunimi, sukunimi, palkka
FROM tyontekija
WHERE palkka >= ALL (SELECT palkka
                      FROM tyontekija);
```

Ylempi kysely ei toimi halutulla tavalla (haetaan suuripalkkaisin heistä, joiden palkka tiedetään), jos palkka-sarakkeessa on tyhjäarvo(ja).

Mutta sen alikyselyä voi muuttaa niin, että se (siis alikysely) palauttaa ei-tyhjäarvot ...

```
SELECT etunimi, sukunimi, palkka
FROM tyontekija
WHERE palkka = (SELECT MAX(palkka)
                  FROM tyontekija);
```

Alempi kysely sen sijaan toimii kaikissa tietokannan tiloissa halutulla tavalla.

Alikysely HAVING-osassa

- Alikyselyä voidaan hyödyntää, kun asetetaan HAVING-osassa ehtoja GROUP BY -osassa tuotetuille ryhmille.
- HAVING-osassa voi olla useita alikyselyjä.

Hae nimet ja maksimipalkat niille osastoille, joilla on ainakin yksi työntekijä, jonka palkka on yhtä suuri kuin kaikkien työntekijöiden maksimipalkka.

```
SELECT onimi, MAX(palkka) max_palkka
FROM osasto o INNER JOIN tyontekija t
    ON o.onro=t.osastonro
GROUP BY onimi
HAVING MAX(palkka) = (SELECT MAX(palkka)
                        FROM tyontekija);
```

onimi	max_palkka
Pääkonttori	5500.00

Alikysely HAVING-osassa

- SQL:ssä ei voi käyttää aggregointifunktioita sisäkkäin.
 - Esim. `max(count(...))` ei ole mahdollista.
- Tällöin voidaan ottaa avuksi HAVING-osassa oleva alikysely.
 - Huom: ALL-avainsanaa ei ole toteutettu SQLItessa.

Haetaan niiden osastojen nimet, joilla on eniten työntekijöitä.

```
SELECT onimi
FROM osasto, tyontekija
WHERE onro = osastonro
GROUP BY onimi
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                           FROM tyontekija
                           GROUP BY osastonro);
```

onimi

Hallinto
Tutkimus

Alikysely FROM-osassa

- Alikysely (tai useita alikyselyjä) voi esiintyä kyselyn FROM-osassa.
- Tällä tavalla voidaan yhdistää yksittäisillä riveillä olevia tietoja (rivikohtaisia tietoja) ja eri tavoin tuotettuja yhteenvetotietoja.
- Esimerkkikysely:

Haetaan kullekin työntekijälle

- työntekijänumero
- sukunimi
- etunimi
- palkka
- osasto, jolla työntekijä työskentelee
- osaston työntekijöiden lukumäärä
- osaston työntekijöiden minimipalkka
- osaston työntekijöiden maksimipalkka

Alikysely FROM-osassa

```
SELECT ttnro, sukunimi, etunimi, palkka, onimi,  
tyontekija_1km, min_palkka, max_palkka  
FROM tyontekija, osasto,  
(SELECT osastonro,  
COUNT(*) AS tyontekija_1km,  
MIN(palkka) AS min_palkka,  
MAX(palkka) AS max_palkka  
FROM tyontekija  
GROUP BY osastonro) AS osastotietoja  
WHERE tyontekija.osastonro = osasto.onro AND  
tyontekija.osastonro = osastotietoja.osastonro;
```

osastokohtaisia
yhteenvetotietoja, jotka on
laskettu yhden tai useamman
tyontekija-taulun rivin perusteella

ttnro	sukunimi	etunimi	palkka	onimi	tyontekija_1km	min_palkka	max_palkka
88	Susi	Jukka	5500.00	Pääkonttori	1	5500.00	5500.00
33	Viima	Ville	4000.50	Tutkimus	2	3000.00	4000.50
12	Puro	Pekka	3000.00	Tutkimus	2	3000.00	4000.50
98	Joki	Jenni	4300.00	Hallinto	2	2500.00	4300.00
99	Kivi	Alli	2500.00	Hallinto	2	2500.00	4300.00

Rivikohtaisia tietoja

Sisäkkäisiä kyselyjä: Vaihtoehto ulkoiselle liitosoperaatiolle

- Ulkoisten liitosoperaatoiden avulla voidaan yhdistää tietoja kahdesta taulusta, vaikkei kaikille riveille löytyisikään liitosehdon mukaista paria.
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN
- Tämä onnistuu myös WHERE-osan liitosehtoa, UNION-operaatiota ja sisäkkäistä kyselyä käyttämällä.

Sisäkkäisiä kyselyjä: Vaihtoehto ulkoiselle liitosoperaatiolle

Haetaan kullekin työntekijälle työntekijännumero, etunimi, sukunimi ja huollettavien lukumäärä.

```
SELECT ttnro, etunimi, sukunimi, COUNT(nimi) AS huollettavia
FROM tyontekija LEFT OUTER JOIN huollettava
    ON ttnro = huoltajanro
GROUP BY ttnro, etunimi, sukunimi
ORDER BY sukunimi, etunimi;
```

ttnro	etunimi	sukunimi	huollettavia
98	Jenni	Joki	1
99	Alli	Kivi	0
12	Pekka	Puro	2
88	Jukka	Susi	0
33	Ville	Viima	3

Sisäkkäisiä kyselyjä: Vaihtoehto ulkoiselle liitosoperaatiolle

```
SELECT ttnro, etunimi, sukunimi, COUNT(nimi) AS huollettavia  
FROM tyontekija, huollettava  
WHERE ttnro = huoltajanro  
GROUP BY ttnro, etunimi, sukunimi;
```

ttnro	etunimi	sukunimi	huollettavia
12	Pekka	Puro	2
33	Ville	Viima	3
98	Jenni	Joki	1

```
SELECT ttnro, etunimi, sukunimi, 0 AS huollettavia  
FROM tyontekija  
WHERE ttnro NOT IN  
    (SELECT huoltajanro  
     FROM huollettava);
```

ttnro	etunimi	sukunimi	huollettavia
88	Jukka	Susi	0
99	Alli	Kivi	0

Sisäkkäisiä kyselyjä: Vaihtoehto ulkoiselle liitosoperaatiolle

```
SELECT ttnro, etunimi, sukunimi, COUNT(nimi) AS huollettavia
FROM tyontekija, huollettava
WHERE ttnro = huoltajanro
GROUP BY ttnro, etunimi, sukunimi
UNION
SELECT ttnro, etunimi, sukunimi, 0 AS huollettavia
FROM tyontekija
WHERE ttnro NOT IN
      (SELECT huoltajanro
       FROM huollettava)
ORDER BY sukunimi, etunimi; ← Järjestetään tulosrivit
```

ttnro	etunimi	sukunimi	huollettavia
98	Jenni	Joki	1
99	Alli	Kivi	0
12	Pekka	Puro	2
88	Jukka	Susi	0
33	Ville	Viima	3

Sisäkkäisiä kyselyjä

- Sisäkkäinen kysely pystytään usein korvaamaan yksinkertaisella (ts. ei-sisäkkäisellä) kyselyllä.
- Sisäkkäisiä kyselyjä kannattaa käyttää ”säästeliäästi”; sisäkkäiset kyselyt tulevat helposti monimutkaisiksi niin kyselyn myöhemmän lukijan kuin tietokannanhallintajärjestelmänkin kannalta.



Tampereen yliopisto

Tietokantojen perusteet

Kertausta ja vähän uuttakin asiaa

ER-mallinnus

- ER-kaavioon otetaan mukaan ainoastaan kaikki rakennettavan tietojärjestelmän kannalta olennaiset tiedot.
 - entiteettityypit, suhdetyypit, attribuutit
 - Mistä asioista tietoa?
 - Miten asiat liittyvät toisiinsa? Mitä yhteyksiä asioiden välillä on?
 - Mitä tietoja asioista ja niiden välisistä yhteyksistä?
- Suhteet laitetaan ER-kaavioon eksplisiittisesti näkyviin suhdetyypeinä - niitä ei piiloteta attribuuteiksi.
 - ER-mallissa ei ole viiteavaimia.
- Suhdetyypeillä ei ole avainattribuutteja eikä suhdetyypin attribuutteja käytetä suhteiden tunnistamisessa.
- ER-kaavion on vastattava SQL-kannan kaaviota.

ER-mallinnus

- ER-kaavion tekeminen on usein iteratiivinen prosessi.
- Saman sovellusalueen voi yleensä mallintaa usealla eri tavalla.
 - Eri ratkaisuissa omat hyvät ja huonot puolensa
- Suunnittelija mallintaa sovellusaluetta sovellusalueen asiantuntijoiden antamien tietojen perusteella.
- Tällä kurssilla ollaan oltu sekä suunnittelijan että sovellusalueen asiantuntijan roolissa.
 - Tämän vuoksi (ja tehtäviin käytettäväissä olevan ajan vuoksi) mallintamistilanteet ovat olleet (yli)yksinkertaistettuja.
 - Todellisuudessa mallintamistilanteet eivät ole näin yksinkertaisia, mutta toisaalta silloin apuna on sovellusalueen asiantuntija, joka kertoo, mitä tietoja tietokantaan halutaan tallentaa, mitä sääntöjä tietoihin liittyy sekä missä ja miten tietoja halutaan hyödyntää.
 - Suunnittelija ei siis itse keksi kantaan tulevia tietoja!

Muunnos ER-kaaviosta SQL-tietokannan kaavioksi

- Muunnos ER-kaaviosta SQL-tietokannan kaavioksi tehdään tiettyjen sääntöjen mukaisesti.
 - Pyritään välttämään tiedon turhaa toistoa sekä turhia tyhjäarvoja.
 - Pyritään esittämään ER-kaavion rakenteellisia rajoitteita SQL-tietokannassa.
- Luennolla ja harjoituksissa käytii läpi seuraavat "perustavat" tehdä muunnos:

ER-kaavio	SQL-tietokannan kaavio
Entiteettityyppi	"Entiteettitaulu"
Attribuutti	Sarake (moniarvoiselle oma taulu + viiteavain)
1:1-suhdetyyppi	Viiteavain
1:N-suhdetyyppi	Viiteavain
M:N-suhdetyyppi	"Suhdetaulu" ja kaksi viiteavainta
Heikko entiteettityyppi ja ID-suhdetyyppi	"Entiteettitaulu" ja viiteavain tunnistavaan tauluun (viiteavaimet tunnistaviin tauluihin)

Tietoihin kohdistuvat rajoitukset

- SQL-tietokannoissa tiedot järjestetään tauluihin.
- Taulun luontilauseessa määritellään, minkälaisia tietoja tauluun voidaan tallentaa.

```
CREATE TABLE projekti (
    pnro INT,
    pnimi VARCHAR(15) NOT NULL,
    onro INT NOT NULL,
    PRIMARY KEY (pnro),
    UNIQUE (pnimi),
    FOREIGN KEY(onro) REFERENCES osasto);
```

Pääavain → Tietotyppi
Avain → Ei sallita puuttuvia arvoja
Viiteavain →

Tietoihin kohdistuvat rajoitukset (ei tenttiin)

```
CREATE TABLE projekti (
pnro INT,
pnimi VARCHAR(15) NOT NULL CHECK(pnimi <> ''),
onro INT NOT NULL,
PRIMARY KEY (pnro),
UNIQUE (pnimi),
FOREIGN KEY(onro) REFERENCES osasto);
```

CHECK-määreellä voi tarkistaa syötettäviä tai muutettavia tietoja

Tietoihin kohdistuvat rajoitukset (ei tenttiin)

- CHECK-määre voidaan sijoittaa luontilauseessa sarakemäärittelyiden jälkeen.
- Tällöin CHECK-määreessä voidaan viitata ko. taulun useisiin sarakkeisiin.

```
CREATE TABLE osasto (
onro INT,
onimi VARCHAR(15) NOT NULL,
johtajanro INT NOT NULL,
aloituspvm DATE DEFAULT '2018-01-15',
PRIMARY KEY (onro),
UNIQUE (onimi),
FOREIGN KEY(johtajanro) REFERENCES tyontekija,
CHECK (onro > 0 AND onimi <> ''));
```

Tietoihin kohdistuvat rajoitukset (ei tenttiin)

- Sarakkeen määrittelyn yhteydessä voidaan antaa sarakkeelle oletusarvo **DEFAULT**-avainsanalla.
- Oletusarvoa käytetään lisäysoperaatiossa, jossa sarakkeelle ei ole annettu arvoa.

```
CREATE TABLE osasto (
onro INT,
onimi VARCHAR(15) NOT NULL,
johtajanro INT NOT NULL,
aloituspvm DATE DEFAULT '2018-10-15',
PRIMARY KEY (onro),
UNIQUE (onimi),
FOREIGN KEY(johtajanro) REFERENCES tyontekija,
CHECK (onro > 0 AND onimi <> ''));
```

SQL: Tietotyyppiestä

- Kurssilla käytettyjä tietotyyppejä
 - CHAR(N)
 - VARCHAR(N)
 - INT
 - NUMERIC(N,D), DECIMAL(N,D)
 - DATE

Muita esim. (ei tenttiin)

- TIME - kellonaika
 - 'tt:mm:ss'
 - tunnit:minuutit:sekunnit
- TIMESTAMP – aikaleima
 - DATE ja TIME, lisäksi minimissään 6 positiota sekunnin desimaaleille
- FLOAT, REAL
 - liukulukuja
- SMALLINT, BIGINT

SQL: Tietotyyppiestä (ei tenttiin)

- Pääavaimina käytetään usein kokonaislukutyyppisiä keinotekoisia tunnuksia.
- Kokonaislukutyyppisiä avaimen arvoja voidaan generoida automaattisesti: tietokannanhallintajärjestelmä generoi avaimen arvon automaattisesti tietoja lisättäessä.
 - tietotyyppi vaihtelee järjestelmissä
 - SQLitessa AUTOINCREMENT tai INT-tyyppinen pääavainsarake
 - PostgreSQL:ssä tämä tietotyyppi on SERIAL

SQL: Tietotyyppiestä (ei tenttiin)

- Esimerkki: SERIAL

```
CREATE TABLE esimerkki(
    tunnus SERIAL,
    nimi VARCHAR(10) NOT NULL,
    PRIMARY KEY (tunnus));

INSERT INTO esimerkki(nimi) VALUES ('Lassi');
INSERT INTO esimerkki(nimi) VALUES ('Lotta');

SELECT * FROM esimerkki;
```

tunnus		nimi
-----+-----		
1		Lassi
2		Lotta

SERIAL-tyyppiseen pääavaimeen viitataan INT-tyyppisellä viiteavaimella.

SQL: Tietotyyppiestä (ei tenttiin)

- Jos haluaa tehdä siirrettäviä SQL-kantoja, tulisi pitäätyä standardin mukaisiin tietotyyppeihin.
 - Siirrettävyys: Kanta voidaan (periaatteessa) luoda samoja luontilauseita käyttäen, olipa kyseessä mikä tahansa SQL-tietokannanhallintajärjestelmä.

Viiteavain ja viite-eheys

- Taulujen tietoja liitetään toisiinsa **viiteavainrakenteen** (vierasavainrakenteen) avulla.
- **Viiteavain** on sarake tai sarakeyhdistelmä, jonka kaikki arvot ovat jonkin taulun pääavaimen arvoja tai tyhjäarvoja
 - Voidaan viitata vain olemassa oleviin riveihin (arvoihin).

Viite-eheys

- Viiteavainmäärittelyn yhteydessä voidaan antaa toimintasääntö, minkä mukaan toimitaan muutos- tai poisto-operaation uhatessa rikkoaa viite-eheyden.
 - ON UPDATE
 - ON DELETE
- Toimintasäännöt
 - **NO ACTION**
 - RESTRICT (ei tenttiin)
 - CASCADE (ei tenttiin)
 - SET NULL (ei tenttiin)
 - SET DEFAULT (ei tenttiin)
- Toimintasääntö valitaan kohdealueella vallitsevien sääntöjen perusteella.

Viite-eheys

NO ACTION

```
CREATE TABLE tyontekija (
    ttnro INT,
    etunimi VARCHAR(15) NOT NULL,
    sukunimi VARCHAR(20) NOT NULL,
    saika DATE NOT NULL,
    kotikunta VARCHAR(20) NOT NULL,
    palkka NUMERIC(8,2),
    puhelin VARCHAR(15),
    osastonro INT NOT NULL,
    esimiesnro INT,
    PRIMARY KEY (ttnro),
    FOREIGN KEY (osastonro) REFERENCES osasto(onro),
    FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro));
```

- Jos viittauksen kohde katoaisi operaation seurausena, operaatio estetään (ei siis tehdä muutosta tai poistoa)

Kun viiteavain määritellään, on NO ACTION -toimintasääntö oletusarvoisesti voimassa muutos- ja poisto-operaatioille.

Viite-eheys

NO ACTION

- Viittauksen kohteena olevaa
 - arvoa ei voi muuttaa eikä
 - riviä poistaa

Ao. lauseiden suoritus ei onnistu, seurausena virheilmoitus

```
UPDATE osasto  
SET onro = 500  
WHERE onro = 5;
```

```
DELETE FROM osasto  
WHERE onro = 5;
```

osasto

onro	onimi
1	Pääkonttori
4	Hallinto
5	Tutkimus

tyontekija

ttnro	etunimi	...	osastonro	esimiesnro
88	Jukka		1	
33	Ville		5	88
12	Pekka		5	33
98	Jenni		4	88
99	Alli		4	98

Taulun rakenteen muuttaminen (ei tenttiin)

- Taulun rakennetta voi muuttaa **ALTER TABLE** -lauseella.
 - lisätä tai poistaa sarakkeita
 - vaihtaa sarakkeiden nimiä ja tietotyypejä
 - muuttaa taulun rajoitteita
- Mahdollisuudet muuttaa taulun rakennetta vaihtelevat tietokannanhallintajärjestelmittäin.
 - PostgreSQL on joustava muutosten suhteen, SQLite puolestaan ei. Esim. :

Sarakkeen lisäys

```
ALTER TABLE taulu  
ADD sarake tietotyyppi
```

```
ALTER TABLE tyontekija  
ADD tempivari VARCHAR(20);
```

Sarakkeen poisto

```
ALTER TABLE taulu  
DROP sarake
```

```
ALTER TABLE tyontekija  
DROP tempivari;
```

Taulun rakenteen muuttaminen (ei tenttiin)

PostgreSQL: Viiteavaimen määrittely

```
ALTER TABLE tyontekija
ADD FOREIGN KEY (osastonro)
REFERENCES osasto;
```

Viiteavainmäärittelyn poisto

```
ALTER TABLE tyontekija
DROP CONSTRAINT
tyontekija_osastonro_fkey;
```

```
Table "tkp.tyontekija"
 Column | Type          | Modifiers
-----+---------------+-----------
ttnro  | integer       | not null
etunimi | character varying(15) | not null
sukunimi | character varying(20) | not null
saika   | date          | not null
kotikunta | character varying(20) | not null
palkka  | numeric(8,2)   |
puhelin | character varying(15) |
osastonro | integer       | not null
esimiesnro | integer      |

Indexes:
```

```
    "tyontekija_pkey" PRIMARY KEY, btree (ttnro)
```

Foreign-key constraints:

```
    "tyontekija_esimiesnro_fkey" FOREIGN KEY (esimiesnro) REFERENCES tyontekija(ttnro)
```

```
    "tyontekija_osastonro_fkey" FOREIGN KEY (osastonro) REFERENCES osasto(onro)
```

Referenced by:

```
    TABLE "huollettava" CONSTRAINT "huollettava_huoltajanro_fkey" FOREIGN KEY (huoltajanro) REFERENCES
tyontekija(ttnro)
```

```
    TABLE "osallistuu" CONSTRAINT "osallistuu_ttnro_fkey" FOREIGN KEY (ttnro) REFERENCES tyontekija(ttnro)
```

```
    TABLE "osasto" CONSTRAINT "osasto_johtajanro_fkey" FOREIGN KEY (johtajanro) REFERENCES tyontekija(ttnro)
```

```
    TABLE "tyontekija" CONSTRAINT "tyontekija_esimiesnro_fkey" FOREIGN KEY (esimiesnro) REFERENCES
tyontekija(ttnro)
```

Kaaviot (schema) (ei tenttiin)

- Useat relaatiotietokantajärjestelmät ja SQL tukevat kaavioiden (schema) käyttöä.
 - SQLite ei tue, mutta PostgreSQL tukkee.
- Kaavio on kokoelma tauluja.
- Tavallisesti käyttäjällä on **oletuskaavio**, jonka nimeä ei tarvitse mainita esimerkiksi taulujen luontilauseissa tai kyselyissä.
 - PostgreSQL:ssä on olemassa oletusarvoisesti public-niminen kaavio, joka on oletusarvoisesti oletuskaaviona.
- Jos halutaan viitata toisen kaavion tauluihin, voidaan käyttää pistenotaatiota:
kaavio.taulu
 - Jos taulu on luotu 'yritys'-nimiseen kaavioon, niin tyontekija-taulun sukunimi-sarakkeeseen viitataan merkinnällä
yritys.tyontekija.sukunimi

Kaavion luonti: CREATE SCHEMA (ei tenttiin)

- Kaavioita luodaan CREATE SCHEMA kaavionimi -komennolla.
- Olemassa oleva kaavio voidaan asettaa oletuskaavioksi SET SEARCH_PATH TO -komennolla.
 - Tämän jälkeen kaavion nimeä ei enää tarvitse erikseen mainita siihen viitattaessa.
 - Huom: Oletuskaavion asetus vaihtelee tietokantajärjestelmästä riippuen
- Alla oleva esimerkki luo 'yritys'-nimisen kaavio ja asettaa sen oletuskaavioksi.

```
CREATE SCHEMA yritys;  
SET SEARCH_PATH TO yritys;
```
- SHOW SEARCH_PATH -komento näyttää hakupolun.

Kaavion poisto: DROP SCHEMA (ei tenttiin)

- Kaavio poistetaan DROP SCHEMA -komennolla:

DROP SCHEMA yritys;

- Taulut on poistettava ensin

DROP SCHEMA yritys CASCADE;

- Poistaa kaavion tauluineen

SQL: Kyselyistä

- Kyselyn suunnittelu
 - Mitä tietoja? SELECT
 - Mistä tauluista tietoja tarvitaan? FROM
 - Miten tarvittavien taulujen tiedot yhdistetään?
 - **WHERE-osan liitosehdot**
 - Jos liitosoperaatio, millainen liitostyyppi? INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN + **ON liitosehdo**
 - Halutaanko tulokseen vain tietyt rivit?
 - WHERE-osan valintaehdot
 - Tehdääkö yhteenvetaja?
 - COUNT, MIN, MAX, SUM, AVG
 - Tehdääkö yhteenvetaja ryhmittäin?
 - GROUP BY
 - Asetetaanko ryhmille vaatimuksia?
 - HAVING
 - Täytyykö tulosrivit järjestää?
 - ORDER BY
 - JA VIELÄ: kiinnitä huomiota tyhjäarvojen käsitteelyyn!!!

SQL: Kyselyistä

- Merkitykseltään sama kysely on yleensä mahdollista määritellä SQL:llä usealla eri tavalla.
 - Katso kertausluennon esimerkkikalvot
- Ihannetilanteessa tietokannanhallintajärjestelmän tulisi prosessoida merkitykseltään sama kysely yhtä tehokkaasti kyselyn määrittelytavasta riippumatta.
- Käytännössä näin ei kuitenkaan ole: kullakin tietokannanhallintajärjestelmällä on omat rutuiinsa eri tavalla määriteltyjen kyselyjen käsitteilyyn.
 - Se, mikä on tehokkain tapa määritellä kysely, riippuu tietokannanhallintajärjestelmästä ja kannan tilasta.
- Tämän kurssin tietojen pohjalta: Määrittele kyselyt tavalla, jonka tunnet parhaiten hallitsevasi!
 - (Ellei sitten luennoitsija pyydä tentissä jotakin tiettyä tapaa)

Tyhjäarvot eli NULL-arvot

- Tyhjäarvot sallitaan avaimissa (UNIQUE-määre), ellei niitä ole kielletty sarakkeelle annettavalla NOT NULL -määreellä.
 - Avaimissa tyhjäarvoja kohdellaan erisuurina.
- Tyhjäarvon vertailu tuottaa totuusarvon tuntematon (unknown).
 - Tyhjäarvon olemassaolon testaamiseen käytetään IS NULL - ja IS NOT NULL -operaattoreita.
- **Huom:** DISTINCT-määre ja GROUP BY -määre käsittelevät tyhjäarvoja yhtäsuurina arvoina.

WHERE-osa

Voidaan verrata esim.

sarakkeen arvoa vakioon

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE kotikunta = 'Tampere';
```

sarakkeen arvoa toisen
sarakkeen arvoon

```
SELECT sukunimi, etunimi, onimi  
FROM tyontekija, osasto  
WHERE osastonro = onro;
```

lausekkeen arvoa vakioon

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE palkka + 200 >= 4500;
```

funktion arvoa vakioon
(ei tenttiin)

```
SELECT etunimi, sukunimi  
FROM tyontekija  
WHERE UPPER(kotikunta) = 'TAMPERE';
```

Merkkijonofunktio **UPPER** muuntaa kirjaimet isoiksi.

Joitakin skalarifunktioita (ei tenttiin)

- Skalarifunktiot palauttavat yhden arvon yhden rivin sarakkeen arvoon tai sarakkeiden arvoihin perustuen.

||-merkkijonofunktio yhdistää merkkijonoja

| on pystyiiva eli putkimerkki

```
SELECT etunimi || ' ' || sukunimi || ', ' || kotikunta  
FROM tyontekija;
```

?column?

```
-----  
Jukka Susi, Tampere  
ville viima, Nokia  
Pekka Puro, Tampere  
Jenni Joki, Lempäälä  
Alli Kivi, Nokia
```

Joitakin skalaarifunktioita (ei tenttiin)

Merkkijonofunktio **UPPER()** muuntaa kirjaimet isoiksi ja **LOWER()** pieniksi.

```
SELECT UPPER(sukunimi) AS snimi  
FROM tyontekija;
```

snimi

SUSI

VIIMA

PURO

JOKI

KIVI

Joitakin skalarifunktioita (ei tenttiin)

ROUND()-funktio pyöristää numeerisen arvon

Keskipalkka pyöristetty (keskipalkka on ensin laskettu viiden rivin perusteella ja sen jälkeen pyöristetty)

```
SELECT ROUND(AVG(palkka),2) AS keskipalkka,  
       SUM(palkka) AS palkkasumma,  
       COUNT(palkka) AS palkkalkm  
FROM tyontekija;
```

keskipalkka	palkkasumma	palkkalkm
3860.10	19300.50	5

Joitakin skalaarifunktioita (ei tenttiin)

COALESCE()-funktio korvaa NULL-arvon annetulla arvolla tai annetun sarakkeen arvolla.

```
SELECT etunimi, sukunimi,
       COALESCE(puhelin, 'ei ole')
          AS puhelin
     FROM tyontekija;
```

etunimi	sukunimi	puhelin
Jukka	Susi	444 1234
Ville	Viima	444 4343
Pekka	Puro	ei ole
Jenni	Joki	444 4488
Alli	Kivi	444 5555

```
SELECT etunimi, sukunimi,
       COALESCE(puhelin, kotikunta)
          AS yhteystieto
     FROM tyontekija;
```

etunimi	sukunimi	yhteystieto
Jukka	Susi	444 1234
Ville	Viima	444 4343
Pekka	Puro	Tampere
Jenni	Joki	444 4488
Alli	Kivi	444 5555

Joitakin skalaarifunktioita (ei tenttiin)

- Tietokannanhallintajärjestelmät tarjoavat erilaisia funktioita tietojen käsittelyyn.
- Tarjolla olevat funktiot, funktioiden nimet ja syntaksit vaihtelevat tietokannanhallintajärjestelmissä (ja järjestelmän versioittain).
- Jos haluaa tehdä siirrettäviä SQL-lauseita, kannattaa funktioita käyttää ”säästeliäästi” (tai välttää niiden käyttöä).

Tietokannanhallintajärjestelmistä ja suunnitteluoohjelmistoista

- Monet tietokannanhallintajärjestelmät tarjoavat työkaluja lomakkeiden ja graafisten käyttöliittymien tekemiseen sekä raporttien generoimiseen.
- Tietojärjestelmien ja tietokantojen suunnittelua varten on olemassa erilaisia ohjelmistoja.
- Ohjelmistoissa voi olla toimintoja, jotka generoivat yhden kaavion pohjalta toisia kaavioita.
 - esim. ER-kaaviosta SQL-kannan kaavio
 - Joskus lopputulos ok, joskus sitä joutuu muuttamaan
 - SQL-kannan graafisesta kaaviosta SQL-lauseita kannan luomiseksi

Sama kysely eri tavoin

Osastot, jotka valvovat projekteja (yhtä tai useampaa)

```
SELECT DISTINCT onimi  
FROM osasto o, projekti p  
WHERE o.onro = p.onro;
```

onimi

Hallinto
Tutkimus

```
SELECT onimi  
FROM osasto  
WHERE onro IN  
(SELECT onro  
    FROM projekti);
```

onimi

Hallinto
Tutkimus

```
SELECT onimi  
FROM osasto o  
WHERE EXISTS  
(SELECT *  
    FROM projekti  
    WHERE onro = o.onro);
```

onimi

Hallinto
Tutkimus

Sama kysely eri tavoin

Osastot, jotka eivät valvo projekteja

```
SELECT DISTINCT onimi  
FROM osasto o, projekti p  
WHERE NOT o.onro = p.onro;
```

onimi

Hallinto
Pääkonttori
Tutkimus

MIKSI TÄMÄ KYSELY EI TOIMI HALUTULLA TAVALLA?

VASTAUS:

```
SELECT *  
FROM osasto o, projekti p;
```

onro	onimi	johtaja	aloituspvm	pnro	pnimi	onro
1	Pääkonttori	88	1989-06-19	1	Tuote X	5
4	Hallinto	98	1992-01-01	1	Tuote X	5
5	Tutkimus	33	2000-05-22	1	Tuote X	5
1	Pääkonttori	88	1989-06-19	2	Tuote Y	5
4	Hallinto	98	1992-01-01	2	Tuote Y	5
5	Tutkimus	33	2000-05-22	2	Tuote Y	5
1	Pääkonttori	88	1989-06-19	3	Tuote Z	5
4	Hallinto	98	1992-01-01	3	Tuote Z	5
5	Tutkimus	33	2000-05-22	3	Tuote Z	5
1	Pääkonttori	88	1989-06-19	10	Uudet edut	4
4	Hallinto	98	1992-01-01	10	Uudet edut	4
5	Tutkimus	33	2000-05-22	10	Uudet edut	4
1	Pääkonttori	88	1989-06-19	20	TYKY-liikunta	4
4	Hallinto	98	1992-01-01	20	TYKY-liikunta	4
5	Tutkimus	33	2000-05-22	20	TYKY-liikunta	4

Sama kysely eri tavoin

Osastot, jotka eivät valvo projekteja

```
SELECT onimi  
FROM osasto  
EXCEPT  
SELECT onimi  
FROM osasto o, projekti p  
WHERE o.onro = p.onro;
```

```
SELECT onimi  
FROM osasto o LEFT OUTER JOIN  
projekti p  
ON o.onro = p.onro  
WHERE p.pnro IS NULL;
```

```
SELECT onimi  
FROM osasto  
WHERE onro NOT IN  
(SELECT onro  
FROM projekti);
```

```
SELECT onimi  
FROM osasto o  
WHERE NOT EXISTS  
(SELECT *  
FROM projekti  
WHERE onro = o.onro);
```

onimi

Pääkonttori

Muita sisäkkäisten kyselyjen käyttötapoja

```
SELECT etunimi, sukunimi, palkka  
FROM tyontekija  
WHERE palkka = (SELECT MAX(palkka)  
                 FROM tyontekija);
```

```
SELECT onimi  
FROM osasto, tyontekija  
WHERE onro = osastonro  
GROUP BY onimi  
HAVING COUNT(*) >= ALL (SELECT COUNT(*)  
                           FROM tyontekija  
                           GROUP BY osastonro);
```

```

SELECT ttnro, sukunimi, etunimi, tyontekija.kotikunta,
       kotikuntalaisia
FROM tyontekija,
     (SELECT kotikunta, COUNT(*) AS kotikuntalaisia
      FROM tyontekija
      GROUP BY kotikunta) AS kktietoja
WHERE tyontekija.kotikunta = kktietoja.kotikunta;

```

ttnro	sukunimi	etunimi	kotikunta	kotikuntalaisia
88	Susi	Jukka	Tampere	2
33	viima	Ville	Nokia	2
12	Puro	Pekka	Tampere	2
98	Joki	Jenni	Lempäälä	1
99	Kivi	Alli	Nokia	2

```

SELECT kotikunta,
       COUNT(*) AS kotikuntalaisia
FROM tyontekija
GROUP BY kotikunta

```

kotikunta	kotikuntalaisia
Tampere	2
Lempäälä	1
Nokia	2