



# 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

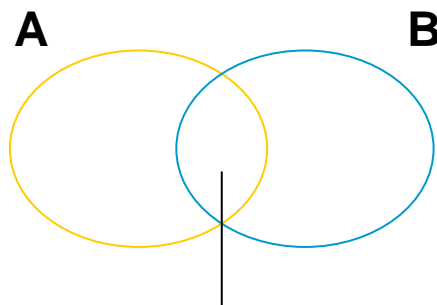
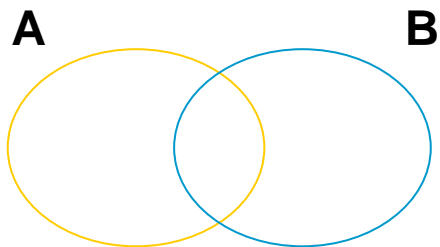
Joukon arvot lueteltu  
eksplisiittisesti



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

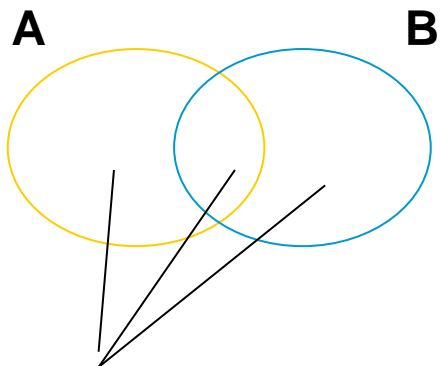
ttnro	sukunimi
33	Viima

# Joukko-operaatioita



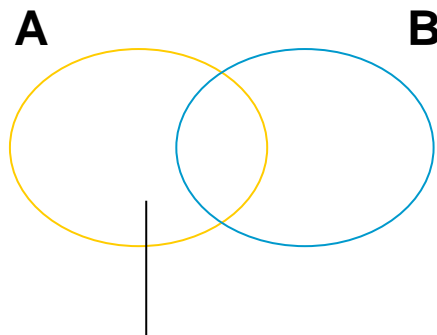
Leikkaus

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



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  $A \setminus B$

# 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

- Operaatiot 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 SQLite:ssä että PostgreSQL:ssä
  - INTERSECT ALL
    - Ei ole käytössä SQLite:ssä
  - EXCEPT ALL
    - Ei ole käytössä SQLite:ssä



# UNION ALL

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

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

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

**taulu1**

nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

**taulu2**

nimi
Bertta
Cecilia
Cecilia

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

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

Moninkertaiset esiintymät on säilytetty.

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

# INTERSECT ALL

**taulu1**

nimi
Alfred
Bertta
Bertta
Cecilia
Cecilia
Cecilia

**taulu2**

nimi
Bertta
Cecilia
Cecilia

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

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

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

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

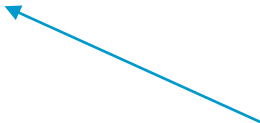
```
onimi
-----
Tutkimus
```

# Joukko-operaatioita

Haetaan niiden osastojen nimet, joilla on sijaintipaikka Tampereella muttei Nokialla. 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, tietotyyppiltään 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 -määre** sijoitetaan joukko-operaatioiden jälkeen **viimeisen SELECT-lauseen jälkeen**.
  - **ORDER BY -määrettä on käytettävä aina, kun tulostaulun rivit halutaan järjestää.** Vaikka joukko-operaatiot näyttäisivät järjestävän tulosrivit, tähän ei saa luottaa, vaan on käytettävä ORDER BY -määrettä!

```
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 suhteen. (selitys jatkuu seuraavalla sivulla)
  - Uudelleennimetyn sarakkeen käyttö toimii sekä PostgreSQL:ssä että SQLitessa.

```
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 hyödyntävissä kyselyssä ORDER BY -listassa ei voi käyttää taulujen nimiä tarkenteina.**
    - Joukko-operaatiot käsittelevät tulostauluja sarakenimineen, taulut nimineen eivät ole enää käytettävissä ORDER BY -määreessä.
  - **SQLitessa** puolestaan tämä on **mahdollista** ja **taulujen nimiä on** käytettävä **tarkenteena**, jos kyselyssä käytettävissä tauluissa on samannimisiä 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, SQLite:ssä puolestaan ei.