Tietokantojen perusteet

Relaationtietokanta ja SQL

Mitä on tietokanta ja tietokannan hallintajärjestelmä

- **Tietokanta** on järjestetty kokoelma jäsenneltyä tietoa tai dataa, joka on tyypillisesti tallennettu sähköisesti tietokonejärjestelmään.
- Tietokantaa ohjaa yleensä **tietokannan hallintajärjestelmä** (englanniksi ”database management system”, jonka lyhenne on **DBMS**).
- Yhdessä dataa ja DBMS-järjestelmää sekä niihin liittyviä sovelluksia kutsutaan tietokantajärjestelmäksi, joka usein lyhennetään vain tietokannaksi.
- Tietokantajärjestelmä on työkalu tiedon keräämiseen ja järjestämiseen
- Tietokannan hallintajärjestelmä hallitse yhtä tai useita tietokantoja ja mahdollista niiden käyttö yhdelle tai useille käyttäjille

Mitä on tietokanta ja tietokannan hallintajärjestelmä

- Tietokannan hallintajärjestelmän tarkoitus on hallita tietoja ajan mittaan ja samalla helpottaa niiden käyttöä käyttäjien toimesta.
- Tämä sisältää esimerkiksi rajoitusten määrittämisen sen varmistamiseksi, että tietokannan tiedot ovat aina oikein, ja menetelmien tarjoamista tiedoista kyselyyn eri monimutkaisuustasoilla.

“*Many databases start as a list in a word-processing program or spreadsheet. As the list grows bigger, redundancies and inconsistencies begin to appear in the data. The data becomes hard to understand in list form, and there are limited ways of searching or pulling subsets of data out for review. Once these problems start to appear, it's a good idea to transfer the data to a database created by a database management system*”

Lähde: Microsoft Database basics

Tietokantojen tyyppiä

- Hierarchical (IBM IMS, Windows registry)
- Relational (Oracle DBMS, PostgreSQL, MariaDB, MySQL, SQLite)
- Non-Relational (MongoDB, Redis)
- Object-Oriented (Realm)

Erilaisia tietokantoja

SQL (relaatio)

- Tieto tallennetaan **tauluihin**, joiden tietueet (rivit) liitetään toisiinsa avaimilla. Rivit koostuvat useista sarakkeista.
- Avaimet ovat **ainutlaatuisia**.
- Yleisin avain on ID.
- Tiedolla voidaan suorittaa matematiikan joukko-opin mukaan operaatiota.

noSQL (ei-vain-relaatio tietokannat, not only)

- Tieto järjestetään jollain muulla tavalla kuin perinteisillä tauluilla.
  - Tietokannat, jotka eivät käytä SQL-kieltä. Ei yhteistä kieltä.
  - Esimerkkejä: Olio, Avain-arvo varastot, dokumenttityypit (JSON, XML,…), graafit, joustotaulut,…
  - **DBSM**: mongoDB, dynamoDB, apache cassandra,…

Esimerkki relaatiotietokannasta

Opiskelijat

ID	NIMI	TUTKINTO	EMAIL
123	Jari	Merkonomi	jari@oma.fi
124	Jukka	Datanomi	jukka@oma.fi
125	Jaana	Datanomi	jaana@oma.fi
126	Julia	Merkonomi	julia@oma.fi

Suoritukset

Opiskelija_ID	KURSSI	ARVOSANA
123	Tk001	1
123	HTML002	3
126	Tk001	5
125	HTML001	4
124	Tk001	

Esimerkkejä ei-relaatiotietokannoista

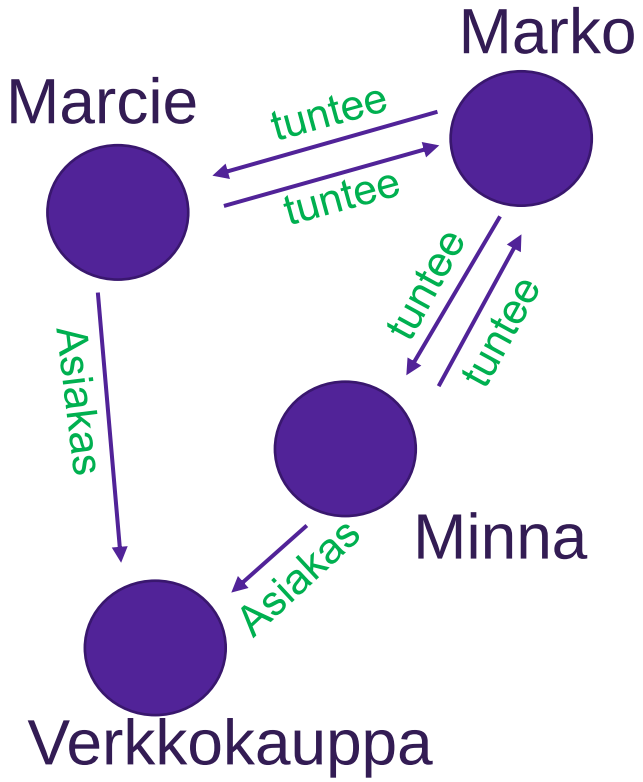
JSON

```
[{
   ”id”: 123,
   ”nimi”: ”Jukka”
},{
   ”id”: 124,
   ”nimi”: ”Jaana”,
   ”ika”: 34
},{
   ”id”: 125,
   ”nimi”: ”Jari”
}]
```

Arvo-avain

Avain	Arvo
”x100”	Merkkijono
”a900”	JSON
”lol”	BLOB
..	..

Graafi/verkosto



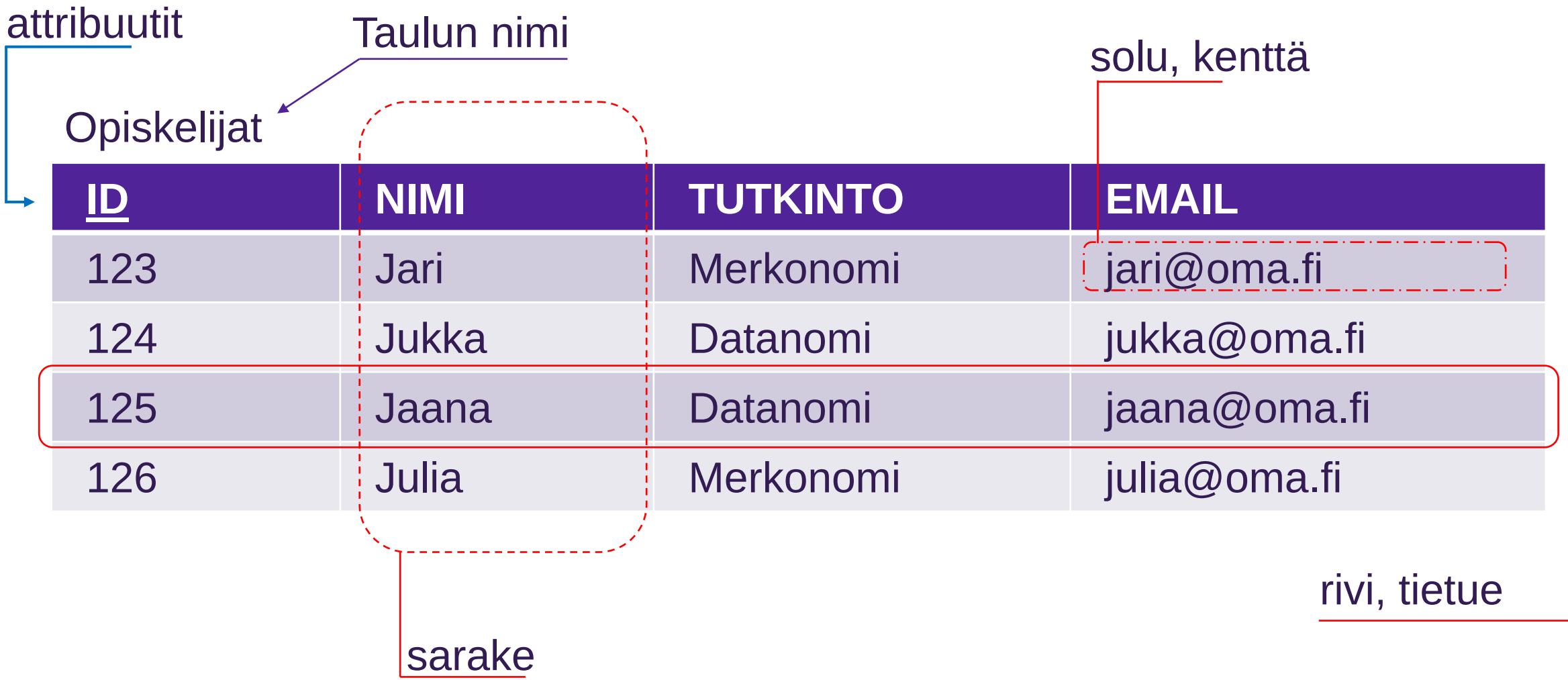
Relaatiomalli

- Ted Coddin ehdotus vuonna 1970
- Periaatteet:
  - Säilytä tietoja yksinkertaisissä tietorakenteissa
  - Käytä tietoja korkean tason kielellä
  - Fyysinen varastoinnin menetelmä jää toteutukseen
- Mitä on relaatio?
  - ”In database theory, a relation, as originally defined by E. F. Codd,[1] is a set of tuples (d1, d2, ..., dn), where each element dj is a member of Dj, a data domain.” (Wikipedia)
  - Käytännössä “relaatio” on taulu, tai tarkemmin se on taulun sisältö
- RDBMS: Relational Database Management System

Tietokannan perussanasto

- Taulu (Table): Tietojen sisältävä rakenne.
- Rivit (Row): Taulut sisältävät rivejä
- Sarake (Column): Rivit sisältävät sarakkeita
- Pääavain (Primary key): Tunnistettava sarake, jolla on mahdollista erottaa rivejä toisistaan.
- Viiteavain (Foreign key): Sarake, jolla on mahdollista tunnista rivejä toisesta taulusta.
- Tietue (Record/Tuple): Taulun rivi.
- Attribuutti/Ominaisuus (Attribute/Property): Taulun sarake.

Esimerkki yhdestä relaatiotietokannan taulusta



ID	NIMI	TUTKINTO	EMAIL
123	Jari	Merkonomi	jari@oma.fi
124	Jukka	Datanomi	jukka@oma.fi
125	Jaana	Datanomi	jaana@oma.fi
126	Julia	Merkonomi	julia@oma.fi

Korkean tason tietokantojen hallinta kieli: SQL

- SQL: Structured Query Language
- Se on IBM:n kehittämä standardoitu kyselykieli, jolla **relaatiotietokantaan** voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
- SQL-kielestä on useita murteita. Eri tietokantojen hallintajärjestelmissä on lukuisia eroja SQL-kielen toteutuksissa.
- Esimerkkejä:

```sql
SELECT id, tietue FROM taulu WHERE quux = 'xyzzy' ORDER BY id DESC;
UPDATE taulu SET kentta = 'esimerkki' WHERE id = 42;
INSERT INTO taulu (kentta,toinenkentta) VALUES ('tietoa', 5);
DELETE FROM taulu WHERE kentta = 123;
```

- Lähde: Wikipedia

Tietokannan tietojen eheys (DBMS)

ACID (Atomicity, Consistency, Isolation, Durability)

- Atomisuus, eheys, eristyneisyys ja pysyvyys

Atomicity (atomisuus) toiminta suoritetaan joko kokonaan tai ei lainkaan

Consistency (eheys) toiminta siirtyy ehjästä tilasta toiseen ehjään tilaan

Isolation (eristyneisyys) jokainen toiminto on oma tapahtumansa, eikä vaikuta muihin toimintoihin.

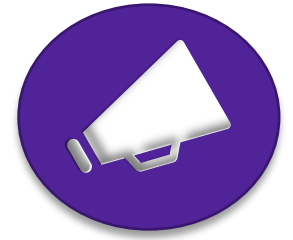
Durability (pysyvyys) toiminnon tekemisen jälkeen sen jäljet eivät voi kadota järjestelmästä.

Pääsääntöisesti DBMS huolehtii tästä, mutta on tärkeää tietää valitun DBMS:n rajoitteet ja erikoisuudet.

Käydään ensin läpi mitä tehdään, jos aloitetaan puhtaalta pöydältä.

- Jos tietokannan rakentaminen aloitetaan puhtaalta pöydältä, niin ensimmäinen vaihe on **käsiteanalyysi**, josta jatketaan **luokka-** tai **ER-kaavioon**.
- ER-kaaviosta voidaan toteuttaa **relaatiomalli**, josta tehdyille tauluille tehdään **normalisointi**, jonka jälkeen tietokannan ensimmäinen versio on valmis rakennettavaksi.

- Tässä ei oteta huomioon tietoturvaa, eikä taloudellista näkökulmaa.



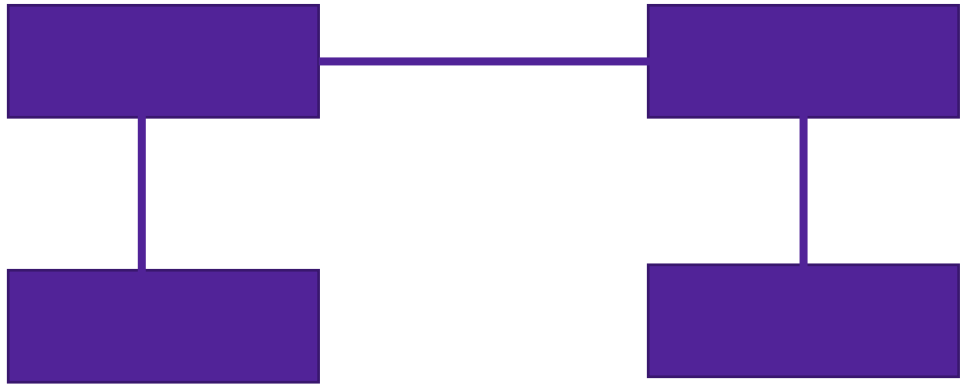
Tiedon kuvaaminen - käsiteanalyysi

Käsiteanalyysi (Conceptual Modeling)

- Prosessi, jota käytetään ongelman kielentämiseen, käsitteellistämiseen ja hajottamiseen osiin (destrukturointi). Muodollinen tapa esittää ongelma, jotta niistä keskustelu ja päätöksenteko on helpompaa.
- Lopputuloksena (tarpeesta riippuen) joko **luokka-** tai **ER-kaavio**

Käsiteanalyysin vaiheet (esimerkki)

1. Tunnista käsite-ehdokkaat
2. Tunnista käsitteiden väliset yhteydet
3. Tunnista ja määrittele osallistumisrajoitteet
4. Tunnista attribuutit/ominaisuus ja yhdistä ne käsitteisiin
5. Yleistä ja eriytä käsitteitä

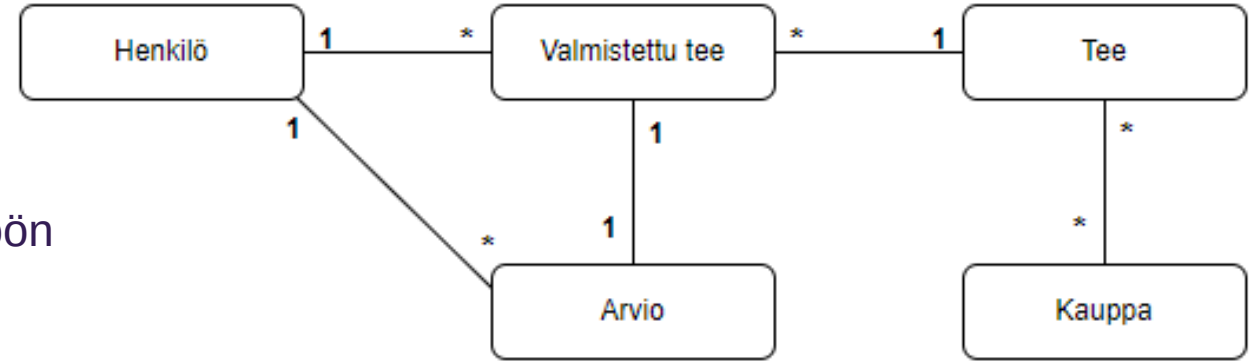


Käsitemalli/ -kaavio

Käsitemalli tehdään käsiteanalyysin pohjalta.

Malli sisältää (usein)

- Tiedon esittämiseen tarvittavat käsitteet (neliöt) ja niiden yhteydet (viivat).
  Ei tarvitse vielä sisältää attribuutteja.
- Käytetään kuvaamaan niin kutsuttua käsiterakennetta eli tietoon ja sen tarkoitukselliseen **varastoitiin** sekä käyttöön liittyviä käsitteitä ja niiden suhteita.
- Käytetään lähinnä tietokantojen suunnittelussa.
- Luokkien yhteydet merkitään 3 kategorialla:
  - monen suhde moneen, N – N, * - *, N - M
  - yhden suhde moneen ja 1 – N, 1 - *
  - yhden suhde yhteen 1 - 1



IOT AIT TAITOTALO

Tiedon kuvaaminen (ER-malli)

ER-kaavio/-malli (Entity-relationship Model)

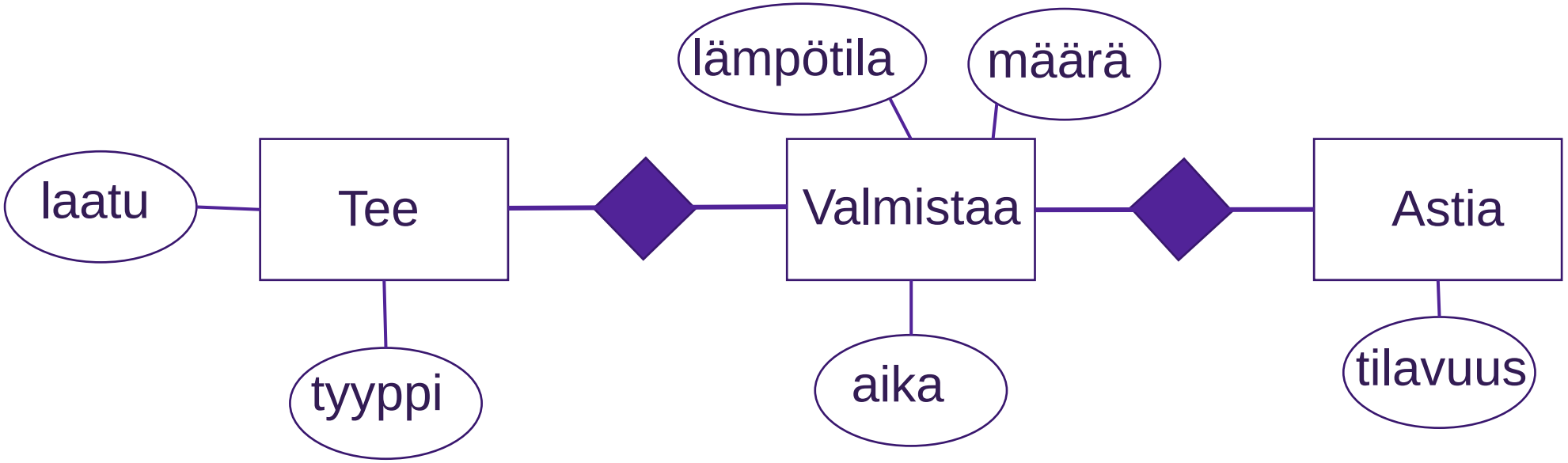
- Tiedon esittämiseen tarvittavat **käsitteet**, niiden **attribuutit** sekä väliset **suhteet**
- Kuvataan usein **käsitteet suorakulmioina**, **attribuutit ellipseinä** ja **suhteet vinoneliöinä.**
- Käytetään kuvaamaan tieto- ja toimintarakennetta. Käytetään paljon kuvaamaan tietorakenteita.
- Pääavaimet alleviivataan,
- Heikot entiteetit ympyröidään kaksoisviivalla
- Moniarvoiset attribuutit kaksoisalleviivaus
- Käsitteiden yhteydet merkitään 3 kategorialla sekä suhteella, jota kuvataan **verbillä**:
  - monen suhde moneen, N – M N:M
  - yhden suhde moneen ja 1 – N 1:N
  - yhden suhde yhteen 1 – 1 1:1
  - (yhden suhde ei yhteenkään tai moneen 1 – 0..N 1:0..N
  - Lisää erilaisia suhteita…)

Tarina ja käsite-ehdokkaiden valinta:

Teetä valmistetaan erilaisissa astioissa. Tee voidaan käyttää useaan valmistukseen. Valmistuksessa vaikuttaa astian lisäksi ainakin; teen tyyppi, käytetty aika, laatu, tilavuus, veden lämpötila ja teen määrä.

Poimitut käsite-ehdokkaat:

Tee	= teekasvista tehty rouhe, josta valmistetaan teejuoma
Astia	= astia jota käytetään teejuoman valmistukseen
Valmistaa	= yksittäinen kerta kun teestä on valmistettu tee juomaa astiassa

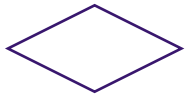
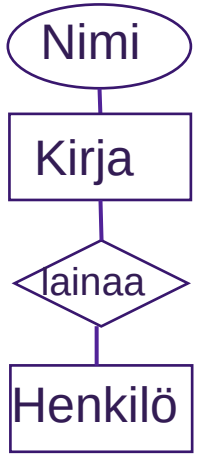


ER-mallin symbolit lyhyesti

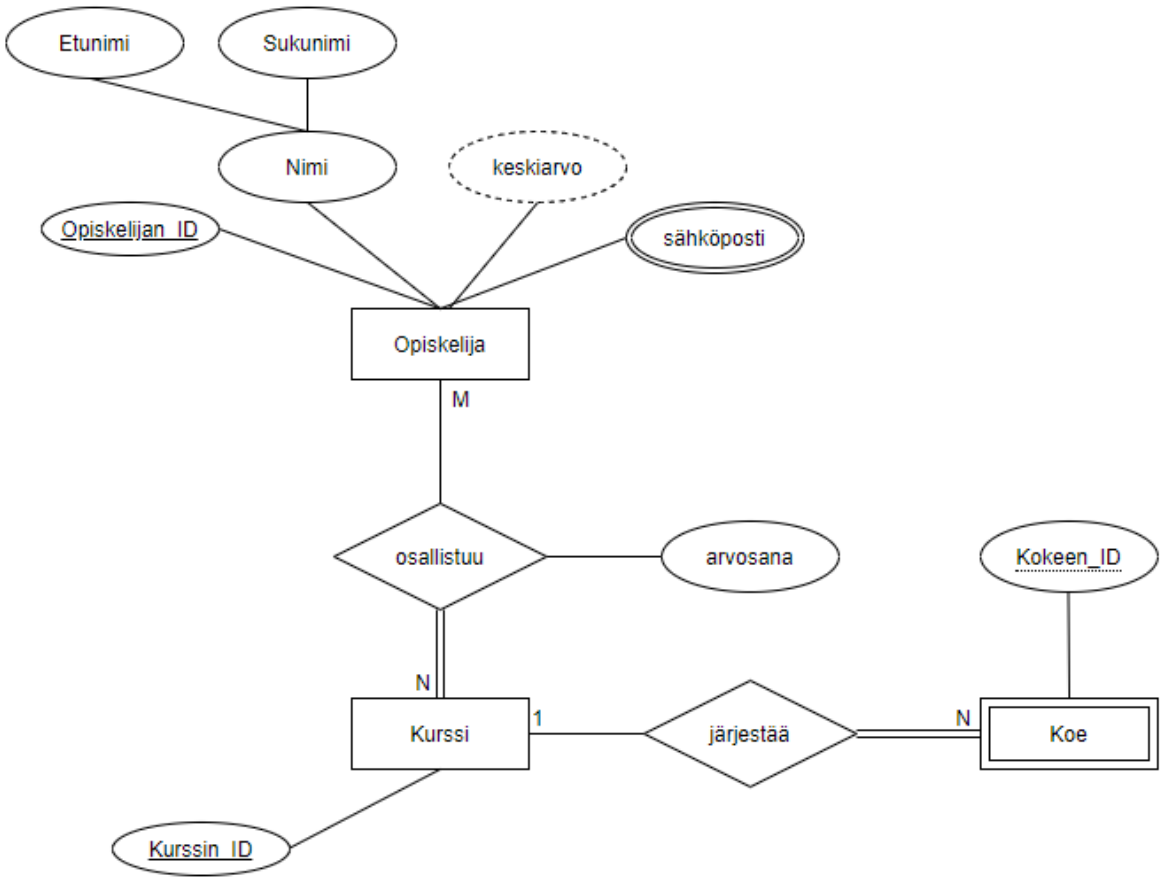
Teetä valmistetaan erilaisissa astioissa. Tee voidaan käyttää useaan valmistukseen. Valmistuksessa vaikuttaa astian lisäksi ainakin; teen tyyppi, käytetty aika, laatu, tilavuus, veden lämpötila ja teen määrä.

Harjoitus:

a) Tee entiteetit esimerkille. Ainakin kolme entiteettiä.
b) Keksi jokaiselle entiteetille ainakin kaksi hyvää arvoa.
c) Keksi entiteettien väliin järkevät suhde-timantit. Näitä kuvaa yleensä verbi.

Symboli	Nimi	Merkitys	Esimerkki
	Neliö	Entiteetin kokonaisuus	
	Ellipsi	Arvo/muuttuja	
	Timantti	Suhde	
	Viiva	Yhdistää	

Tiedon kuvaaminen (ER-kaavio)

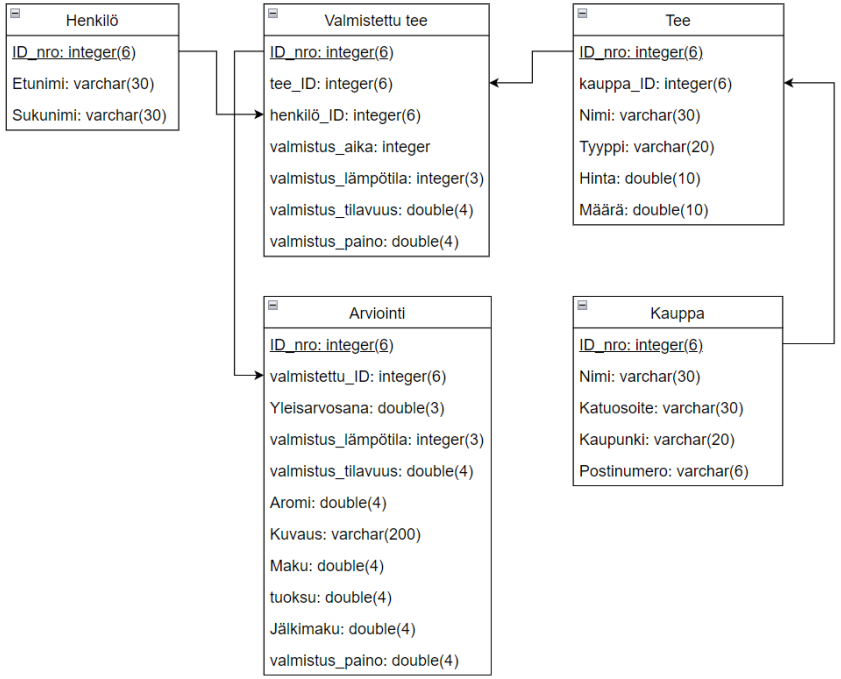


- Entiteetit: Opiskelija, Kurssi ja koe
- Koe on heikko entiteetti
- Alleviivatut attribuutit ovat avaimia
- Katkoviivainen attribuutti on johdettu arvo
- Kaksiviivainen attribuutti on moniarvoinen arvo
- N, M ja 1 osoittavat suhteita ja riippuvuuksia

Tietokannan kuvaus

- Tietokannan kuvaus tehdään valmiista tai lähes valmiista tietokannasta.

- Yksinkertainen malli sisältää taulujen nimet ja taulujen arvot.
  - Vaihtoehtoisesti sisältävät:
    - Teknisen kuvauksen taulun arvoista.
    - Esimerkkejä taulun tiedoista.



TYÖNTEKIJÄ

ID	ETUNIMI	SUKUNIMI	SUKUPUOLI	SYNTYMÄAIKA	TYÖ_ALKANUT	PALKKA	OSASTO_ID	ESIMIES_ID
123456789	Jari	Aaltonen	M	1990-02-05	2010-09-09	4 500,00 €	1	NULL
123789456	Minna	Korhonen	N	1995-05-05	2015-01-02	3 600,00 €	2	123456789
456789123	Marko	Aaltonen	M	1970-11-09	2001-03-06	3 890,00 €	3	123456789
123444555	Jaakko	Korhonen	M	1980-12-24	2018-08-09	2 800,00 €	2	123789456
123444666	Jami	Jantunen	M	1988-10-20	2019-08-09	2 800,00 €	3	456789123
123444777	Jukka	Metsä	M	1981-12-14	2018-08-09	2 880,00 €	3	123456789
123444888	Björn	Saarinen	M	1980-12-24	2018-08-09	3 100,00 €	3	123789456
123444999	Jennica	Saari	N	1996-11-20	2018-08-09	2 900,00 €	4	123789456
123444000	Mikko	Groos	M	1970-02-04	2018-08-09	3 000,00 €	4	123789456
123444111	Jane	Smith	N	1979-07-16	2020-06-09	2 850,00 €	11	456789123
123444222	Erika	Hossa	N	1985-03-11	2020-06-09	2 800,00 €	1	123456789

TYÖSKENTELEE

TYÖNTEKIJÄ_ID	PROJEKTI_ID	TUNTIA
123456789	201	10,00
123789456	201	36,50
456789123	101	10,50
456789123	102	20,00
456789123	1	5,00
123444555	101	NULL
123444666	1	10,00
123444777	2	20,00
123444666	3	15,00
123444777	3	20,00
123444888	2	37,67
123444999	401	36,00
123444000	402	29,00
123444111	110	40,00
123444222	203	25,00
123444222	105	13,00

PROJEKTI

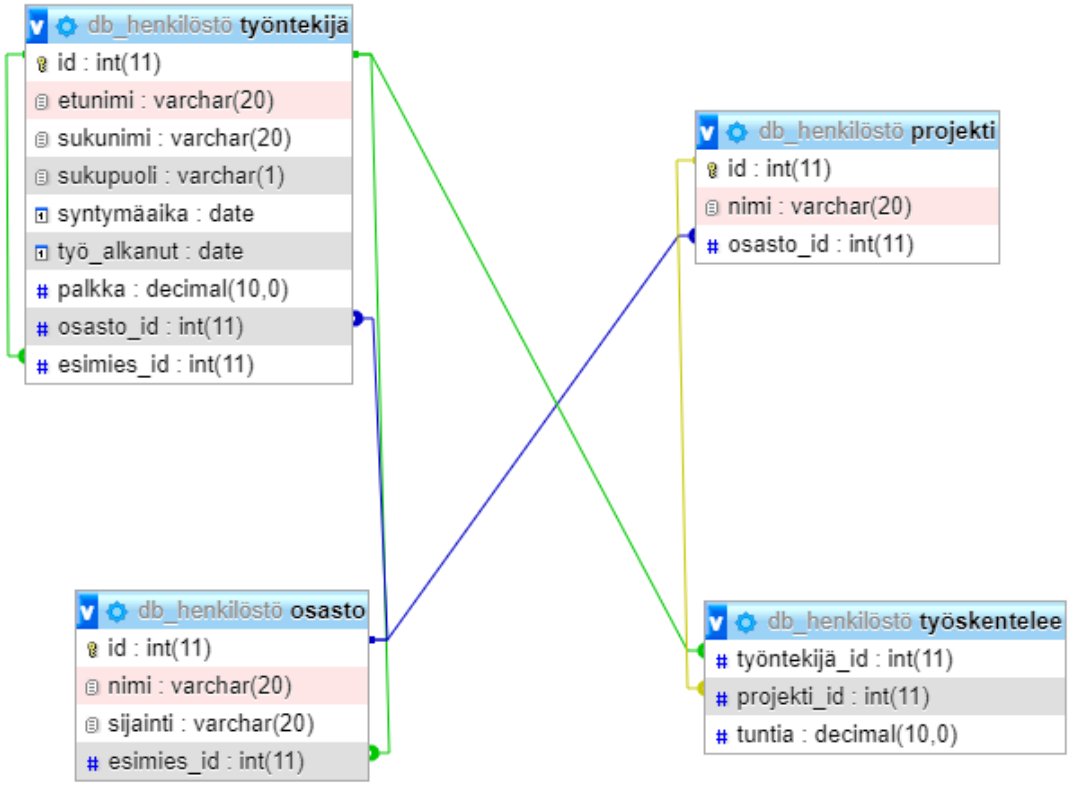
ID	NIMI	OSASTO_ID
1	TuoteX	3
2	TuoteY	3
3	TuoteZ	3
101	Automaatio	3
102	Ylläpito	3
203	Kehitys	1
201	Rekrytointi	2
105	TuoteW	1
401	Mainonta	4
402	Suunnittelu	4
50	Kehitys	3
110	Kierrätys	11

OSASTO

ID	NIMI	SIJAINTI	ESIMIES_ID
1	Kehitys	Espoo	123456789
2	Hallinto	Tampere	123789456
3	Tuotanto	Lempäälä	456789123
4	Markkinointi	Turku	123789456
11	Kierrätys	Lempäälä	456789123

IOT AT TAITOTALO

Tietokantakaavio (Tietokannan rakenne)



Kuvaa valmista tietokantaa, johon on merkitty ainakin:

- Taulujen nimet
- Taulujen sarakenimet
  - usein myös tietotyyppi
- Taulujen avaimet
- Avainten ja taulujen yhteydet
  - Usein myös taulujen suhde

Aiemmat mallit ovat tarkoitettu suunnitteluun ja sen avustamiseen. Tämä lopullisen tuotteen ”esittelyyn”.

IOT ATT TAITOTALO

Entiteettien suhteet

Relaatiotietokannoissa on kolme pääasiallista suhdetyyppiä, jotka kuvaavat, miten Entiteetit ja taulut liittyvät toisiinsa:

- One-to-one (1:1)
- One-to-many (1:N) tai Many-to-one (N:1)
- Many-to-many (N:M)

One-to-one suhde

- **Määritelmä**: Yhdellä rivillä taulussa A on täsmälleen yksi vastaava rivi taulussa B, ja päinvastoin.
- **Esimerkki**: Henkilö ja hänen passinsa.
- **Käyttö**: Harvinaisempi, käytetään usein tietoturvan tai loogisen rakenteen vuoksi.
- **Milloin käytetään one-to-one-suhdetta?**
  - Kun halutaan jakaa tietoa loogisesti eri tauluihin (esim. arkaluontoiset tiedot erilliseen tauluun).
  - Kun osa tiedoista on harvoin käytettyjä tai valinnaisia, eikä niitä haluta pitää päätaulussa.
- **Toteutus**: Viiteavain + UNIQUE tai PRIMARY KEY

One-to-many suhde

- **Määritelmä**: Yhdellä rivillä taulussa A voi olla monta vastaavaa riviä taulussa B, mutta taulun B rivillä on vain yksi vastaava rivi taulussa A.
- **Esimerkki**: Asiakas ja hänen tilauksensa.
- **Käyttö**: Yleisin suhde tietokannoissa.
- **Toteutus**: Viiteavain B-taulussa.
- **Many-to-one** on ihan sama suhde mutta toisinpäin.

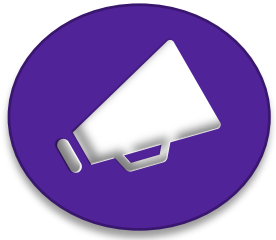
Many-to-many suhde

- **Määritelmä**: Yhdellä rivillä taulussa A voi olla monta vastaavaa riviä taulussa B, ja päinvastoin.
- **Esimerkki**: Opiskelijat ja kurssit (opiskelija voi osallistua usealle kurssille, ja kurssilla voi olla useita opiskelijoita).
- **Käyttö**: Many-to-many-suhteet ovat melko yleisiä relaatiotietokannoissa, erityisesti silloin kun mallinnetaan monimutkaisia yhteyksiä eri entiteettien välillä.
- **Toteutus**: Tarvitaan **liitostaulu** (esim. *OpiskelijaKurssi*), joka sisältää viiteavaimet molemmista tauluista.

Tietokannan normalisointi

Normalisointi voidaan tehdä keskeneräiselle tietokannalle tai jo valmiille. Helpointa tehdä esimerkkidatalla.

Youtube-video, jossa esimerkki tietojen normalisoinnista



Tietokannan normalisointi

- Normalisointia käytetään tilanteissa, joissa tietoa on jo kerätty johonkin tietokantaan tai paperille. Tietoa on siis kerätty, mutta sen säilömistä ja/tai hyödyntämistä ei ole pohdittu.
- Monivaiheinen prosessi, jossa parannetaan tietokannan ”tehokkuutta”. Ennen kaikkea tiedon ehjää tallennusta sekä tehokasta saatavuutta.
- Ongelmakohtien tunnistaminen ja korjaaminen tapahtuu vaiheittain niin kutsuttuja **normaalimuotoja** käyttämällä.

Normaalimuotoja on viisi (kuusi), joista kolme ensimmäistä ovat tärkeimmät. Silloin tietokanta katsotaan **normalisoiduksi**.

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

- Tiedon on oltava ***atomista***. Eli jokaisen attribuutin pitää olla yksinkertainen. Listat eivät ole sallittuja.
- Sarakkeen kaikkien arvojen pitää olla samaa tyyppiä
- Jokaisen rivin pitää olla uniikki

Toinen normaalimuoto (2NF)

Kolmas normaalimuoto (3NF)

Boyce-Coddiin normaalimuoto (BCNF)

Neljäs normaalimuoto (4NF)

Viides normaalimuoto (5NF)

Esimerkki: tietokannan normalisointi

1NF:

- Jokaisessa solussa **yksi** arvo
- Jokaisessa sarakkeessa vain **yksi** tietotyyppi
- Jokainen tietue oltava **yksilöitävissä** (lisää ID)

(Ei normalisoitu)

Opiskelija	Kurssit	Osoite	Opettaja	Opettajan sposti
Matti Meikäläinen	HTML1, TK2	Puistokatu 1, Espoo	Sari A., JiiPee	jiipee@oma.fi, sari@oma.fi
Maija Meikäläinen	HTML1	Kivikatu 5, Helsinki	JiiPee	jiipee@oma.fi
Niko Näkijä	TK1	Kalliokatu 13, Lempäälä	Sari A.	sari@oma.fi
Maija Meikäläinen	TK1, TK2	NULL	Sari A.	sari@oma.fi

Esimerkki: tietokannan normalisointi

1NF:

- Jokaisessa solussa yksi arvo
- Jokaisessa sarakkeessa vain yksi tietotyyppi
- Jokainen tietue oltava yksilöitävissä (**lisää ID**)

ID	Opiskelija	Kurssi	Osoite	Opettaja	Opettajan sposti
122	Matti Meikäläinen	HTML1	Puistokatu 1, Espoo	JiiPee	jiipee@oma.fi
123	Matti Meikäläinen	TK2	Puistokatu 1, Espoo	Sari A.	sari@oma.fi
124	Maija Meikäläinen	HTML1	Kivikatu 5, Helsinki	JiiPee	jiipee@oma.fi
125	Niko Näkijä	TK1	Kalliokatu 13, Lempäälä	Sari A.	sari@oma.fi
126	Maija Meikäläinen	TK2	NULL	Sari A.	sari@oma.fi
127	Maija Meikäläinen	TK1	NULL	Sari A.	sari@oma.fi

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

Toinen normaalimuoto (2NF)

Jokaisen attribuutin tulee **riippua avaimesta**

Kolmas normaalimuoto (3NF)

Boyce-Coddiin normaalimuoto (BCNF)

Neljäs normaalimuoto (4NF)

Viides normaalimuoto (5NF)

Esimerkki: tietokannan normalisointi

2NF:

- Jokaisen attribuutin tulee **riippua avaimesta**

(Ei normalisoitu)

ID	Opiskelija	Kurssi	Osoite	Opettaja	Opettajan sposti
122	Matti Meikäläinen	HTML1	Puistokatu 1, Espoo	JiiPee	jiipee@oma.fi
123	Matti Meikäläinen	TK2	Puistokatu 1, Espoo	Sari A.	sari@oma.fi
124	Maija Meikäläinen	HTML1	Kivikatu 5, Helsinki	JiiPee	jiipee@oma.fi
125	Niko Näkijä	TK1	Kalliokatu 13, Lempäälä	Sari A.	sari@oma.fi
126	Maija Meikäläinen	TK2	NULL	Sari A.	sari@oma.fi
127	Maija Meikäläinen	TK1	NULL	Sari A.	sari@oma.fi

Esimerkki: tietokannan normalisointi

2NF:

- Jokaisen attribuutin tulee **riippua avaimesta**

ID	Opiskelija	Osoite
001	Matti Meikäläinen	Puistokatu 1, Espoo
002	Maija Meikäläinen	Kivikatu 5, Helsinki
003	Niko Näkijä	Kalliokatu 13, Lempäälä
004	Maija Meikäläinen	NULL

Kurssi	Opettaja	Opettajan sposti
HTML1	JiiPee	jiipee@oma.fi
TK1	Sari A.	sari@oma.fi
TK2	Sari A.	sari@oma.fi

Vaihtoehto 1

ID	Opiskelija_ID	Kurssi
1	001	HTML1
2	001	TK2
3	002	HTML1
4	003	TK1
5	004	TK1
6	004	TK2

Esimerkki: tietokannan normalisointi

2NF:

- Jokaisen attribuutin tulee **riippua avaimesta**

ID	Opiskelija	Osoite
001	Matti Meikäläinen	Puistokatu 1, Espoo
002	Maija Meikäläinen	Kivikatu 5, Helsinki
003	Niko Näkijä	Kalliokatu 13, Lempäälä
004	Maija Meikäläinen	NULL

Kurssi	Opettaja	Opettajan sposti
HTML1	JiiPee	jiipee@oma.fi
TK1	Sari A.	sari@oma.fi
TK2	Sari A.	sari@oma.fi

Vaihtoehto 2

Opiskelija_ID	Kurssi
001	HTML1
001	TK2
002	HTML1
003	TK1
004	TK1
004	TK2

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

Toinen normaalimuoto (2NF)

Kolmas normaalimuoto (3NF)

- Taulun attribuuttien pitää olla **SELKEÄSTI** riippuvaisia avainkentästä **ja VAIN** avainkentästä
- Viimeinen normaalimuoto jota käytetään yleisesti

Boyce-Coddiin normaalimuoto (BCNF)

Neljäs normaalimuoto (4NF)

Viides normaalimuoto (5NF)

Esimerkki: tietokannan normalisointi

3NF:

Taulun attribuuttien pitää olla **SELKEÄSTI** riippuvaisia avainkentästä **ja VAIN** avainkentästä

ID	Opiskelija	Osoite
001	Matti Meikäläinen	Puistokatu 1, Espoo
002	Maija Meikäläinen	Kivikatu 5, Helsinki
003	Niko Näkijä	Kalliokatu 13, Lempäälä
004	Maija Meikäläinen	NULL

Kurssi	Opettaja	Opettajan sposti
HTML1	JiiPee	jiipee@oma.fi
TK1	Sari A.	sari@oma.fi
TK2	Sari A.	sari@oma.fi

Opiskelija_ID	Kurssi
001	HTML1
001	TK2
002	HTML1
003	TK1
004	TK1
004	TK2

(Ei normalisoitu)

Esimerkki: tietokannan normalisointi

3NF:

Taulun attribuuttien pitää olla **SELKEÄSTI** riippuvaisia avainkentästä **ja VAIN** avainkentästä

ID	Opiskelija	Osoite
001	Matti Meikäläinen	Puistokatu 1, Espoo
002	Maija Meikäläinen	Kivikatu 5, Helsinki
003	Niko Näkijä	Kalliokatu 13, Lempäälä
004	Maija Meikäläinen	NULL

Kurssi	Opettaja
HTML1	JiiPee
TK1	Sari A.
TK2	Sari A.

Opettaja	Opettajan sposti
JiiPee	jiipee@oma.fi
Sari A.	sari@oma.fi

Opiskelija_ID	Kurssi
001	HTML1
001	TK2
002	HTML1
003	TK1
004	TK1
004	TK2

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

Toinen normaalimuoto (2NF)

Kolmas normaalimuoto (3NF)

Boyce-Coddiin normaalimuoto (BCNF)

Taulujen attribuutit saavat olla riippuvaisia vain **KOKONAISESTA** avaimesta

Neljäs normaalimuoto (4NF)

Viides normaalimuoto (5NF)

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

Toinen normaalimuoto (2NF)

Kolmas normaalimuoto (3NF)

Boyce-Coddiin normaalimuoto (BCNF)

Neljäs normaalimuoto (4NF)

- Kielletään moniarvoinen riippuvuus
- Jos A määrää B:n ja C:n, niin taulu jossa on A, B ja C ei ole normaalimuodon mukainen.

Viides normaalimuoto (5NF)

Tietokannan normalisointi

Ensimmäinen normaalimuoto (1NF)

Toinen normaalimuoto (2NF)

Kolmas normaalimuoto (3NF)

Boyce-Coddiin normaalimuoto (BCNF)

Neljäs normaalimuoto (4NF)

Viides normaalimuoto (5NF)

Kieltää ei triviaalit liitosriippuvuudet, jotka eivät synny avaimista.

Indeksit

- Tietokannan indeksit ovat rakenteita, joita käytetään nopeuttamaan tietojen hakua tauluista. Ne toimivat samalla tavalla kuin kirjan hakemisto: sen sijaan että selattaisiin koko kirja läpi, voidaan siirtyä suoraan oikeaan kohtaan.
- Indeksi luodaan yhdelle tai useammalle sarakkeelle.
- Tyypillisesti PRIMARY KEY ja UNIQUE sarakkelille luodaan automaattisesti indeksiä.
- Indeksien haitat:
  - Vie levytilaa.
  - Hidastaa INSERT-, UPDATE- ja DELETE-toimintoja, koska indeksejä pitää päivittää.
  - Liian monta indeksiä voi heikentää suorituskykyä.

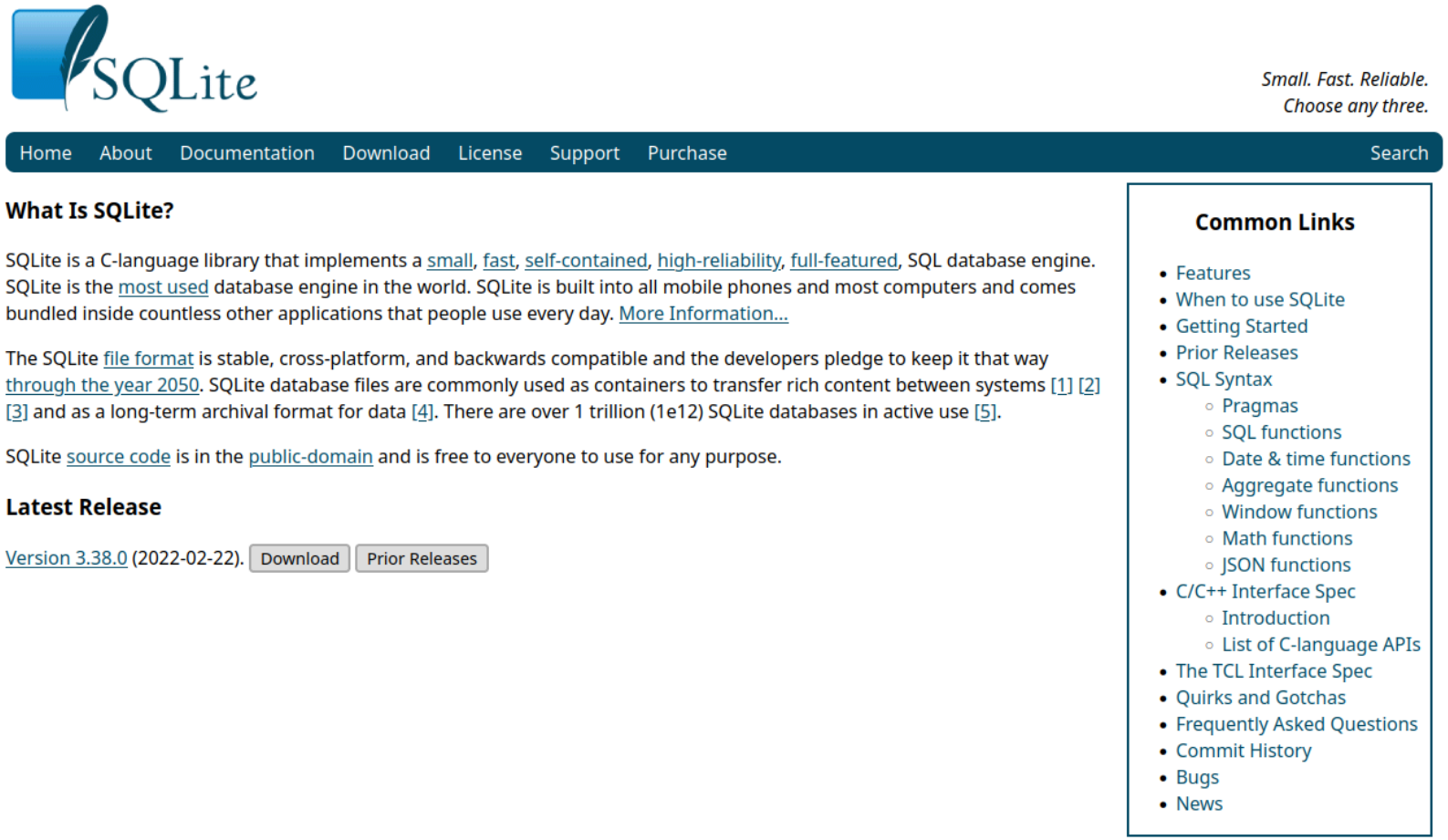
- https://en.wikipedia.org/wiki/Database_index
- https://www.essentialsql.com/what-is-a-database-index/

Indeksin esimerkki

- Henkilöistä säilytetään nimi-, ikä- ja henkilötunnus- tietoa tauluun.
- Jos jostain syystä haetaan usein henkilöitä iän perusteella, kannattaisi luoda yksi indeksi sen perusteella, joka nopeuttaisi haut huomattavasti, kun ei enää tarvitse käydä läpi koko taulun löytämään ko. henkilöitä.

SQLite

Sivusto: https://www.sqlite.org/index.html



SQLite – Mitä se on?

SQLite on tietokannan hallintajärjestelmä. Se on toteutettu pienenä C-kirjastona. SQLite tukee suurinta osaa SQL-kielen SQL-92-standardista.

Se ei ole erillinen sovellus; pikemminkin se on kirjasto, jonka ohjelmistokehittäjät upottavat sovelluksiinsa. Sellaisenaan se kuuluu sulautettujen tietokantojen perheeseen. Se on laajimmin käytetty tietokantamoottori, koska sitä käyttävät useat suosituimmat verkkoselaimet, käyttöjärjestelmät, matkapuhelimet ja muut sulautetut järjestelmät

SQLite – Miksi

Toisin kuin monet muut relaatiotietokannat, koko SQLite-järjestelmä linkitetään sitä käyttävään sovellukseen, joten erillistä ODBC-yhteyttä, tietokannanhallintaohjelmaa tai tietokantapalvelinta ei tarvita. Itse tietokanta voidaan pitää kokonaan tietokoneen muistissa, tai tallentaa yhteen tiedostoon, joka lukitaan transaktioiden ajaksi. **SQLite soveltuu tämän takia erityisesti tietokantajärjestelmän lisäämiseen sovellusohjelmiin.**

SQLite-kirjasto on public domain -ohjelmisto, joten sitä voidaan muokata ja levittää vapaasti ja se voidaan linkittää kaikkiin ohjelmistoihin ilman erillistä lupaa.

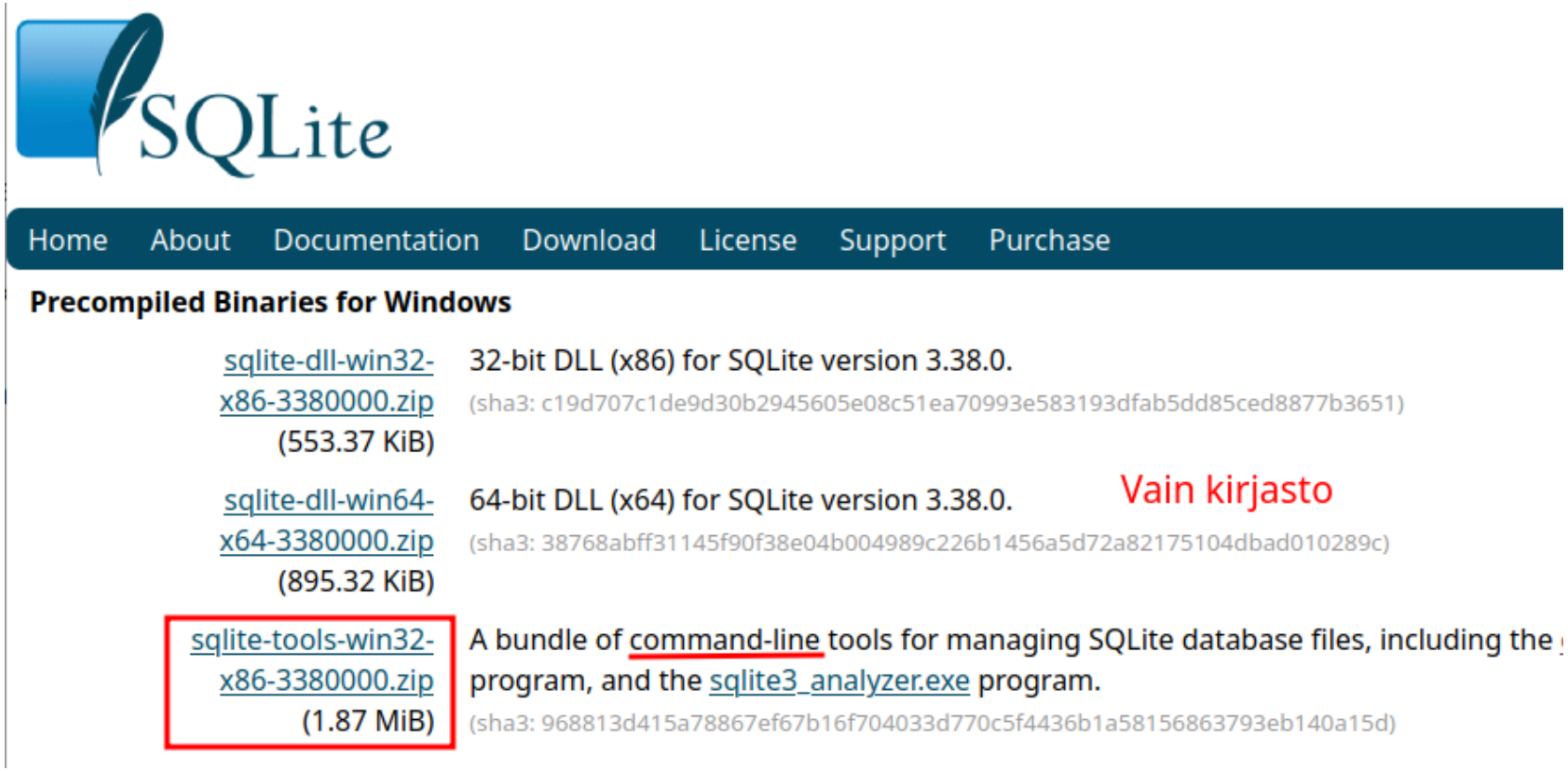
SQLite – Pitää kuitenkin huomata

SQLite sisältää sidoksia useisiin ohjelmointikieliin. Se noudattaa yleensä PostgreSQL-syntaksia, mutta ei pakota tyyppitarkistusta. Tämä tarkoittaa, että kokonaislukuna määriteltyyn sarakkeeseen voidaan esimerkiksi lisätä merkkijonoa. SQLiteä voikin kuvata dynaamisesti tyypitetyksi tietokannaksi staattisesti tyypitetyn sijaan.

SQLiten tietokannan sarakeleveyksiä ei myöskään tarvitse määrittää kiinteästi jolloin tietokanta varaa muistia vain sen verran kuin sen sisällä oleva data edellyttää. Tekstimuotoiset tietotyypit tallennetaan oletuksena UTF-8-koodattuna.

SQLite – Lataus

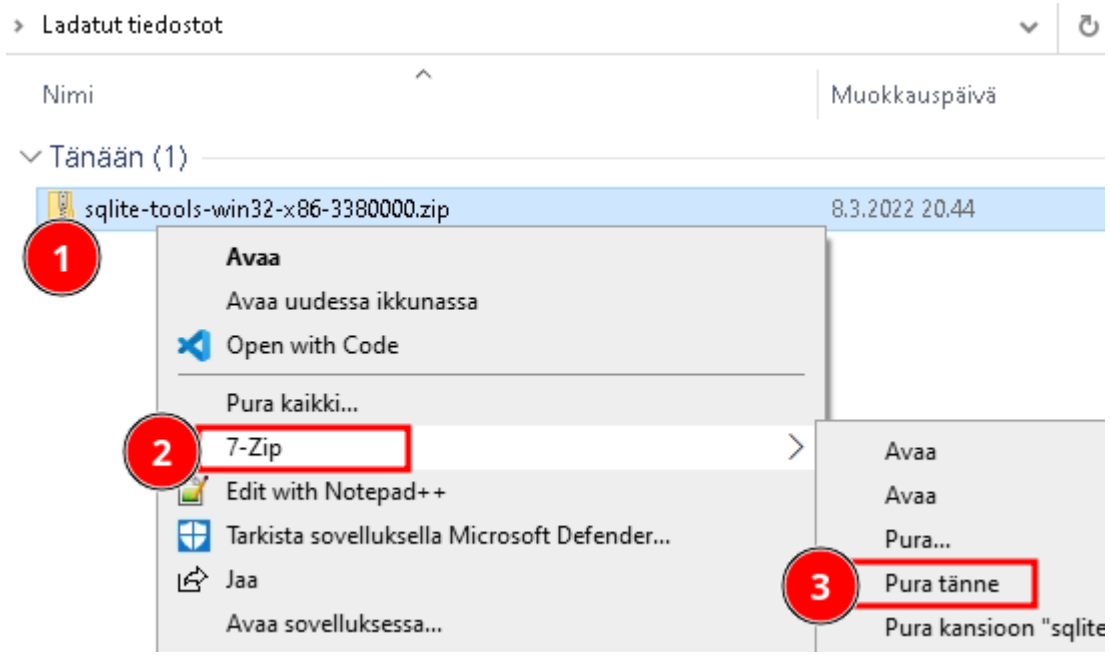
Windows:

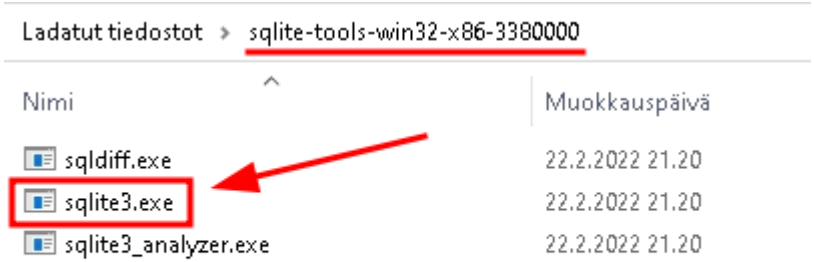


Se on myös saatavilla muille käyttöjärjestelmille kuten Linux tai MacOS

SQLite

Purkaa tiedoston. 7-Zipilla:

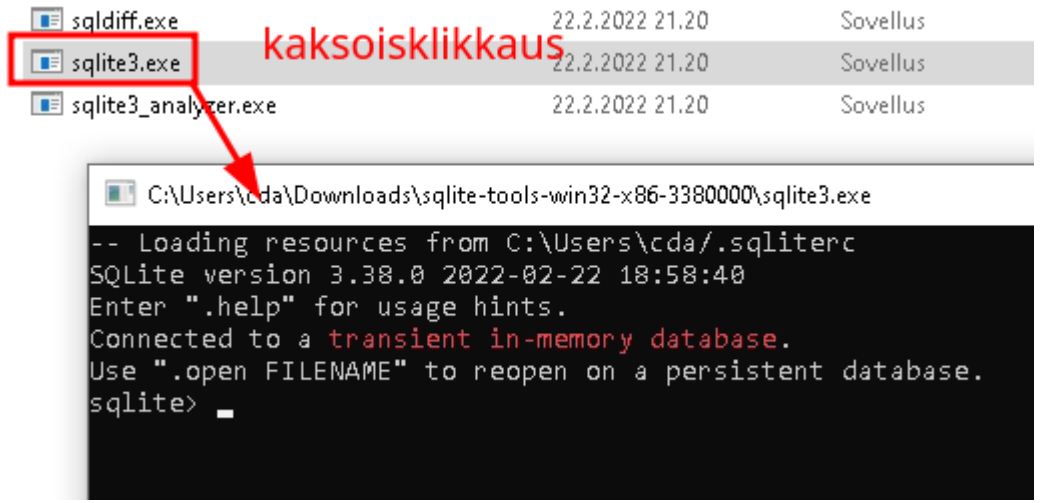




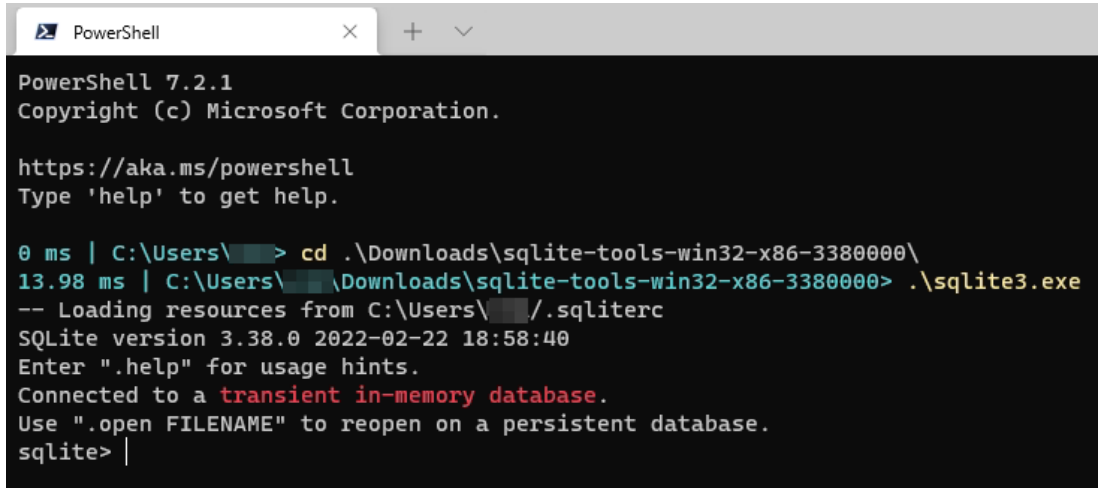
IOT AIT TAITOTALO

SQLite

Avaa SQLiteä:



tai



IOT AI TAITOTALO

SQLite

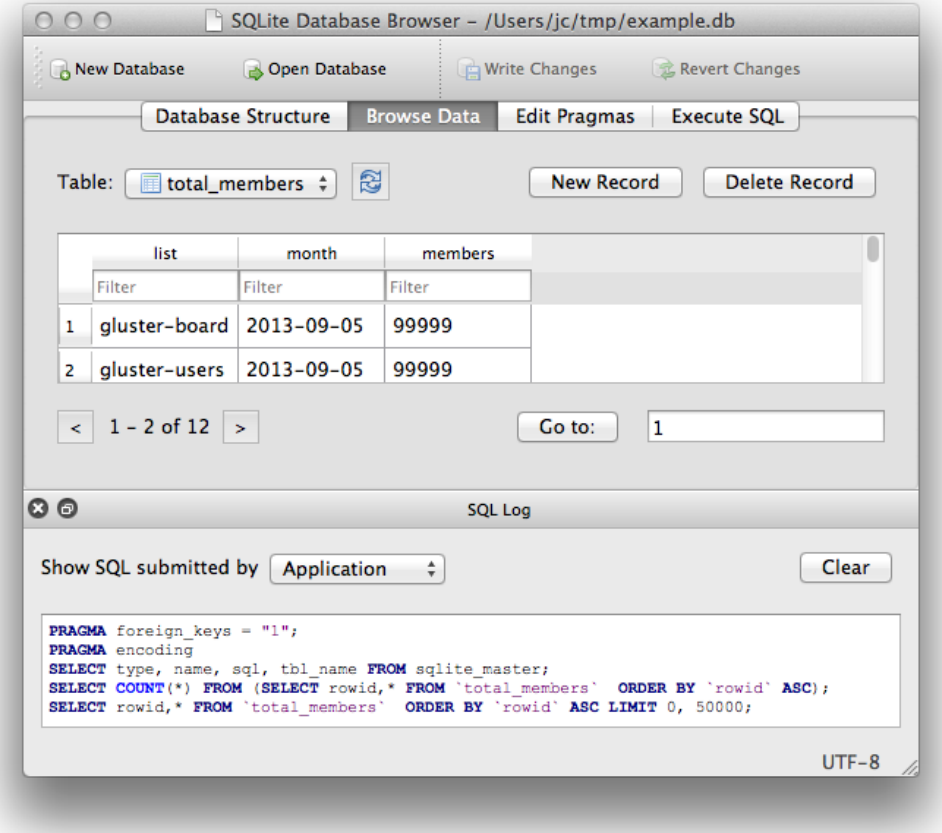
- Apua: .help
- Avata tietokantatiedostoa: .open
- Listata taulua: .tables
- Tulosta taulun järjestely: .schema *taulu*
- Aseta otsikot päälle: .headers on
- Vaihtaa tulosten muotoa: .mode box | table | json | csv | …
- Pakottaa vierasavaimien tarkistaminen: PRAGMA foreign_keys=1
- Poistua SQLitestä: .quit tai .exit

- SQLite:n asetuksen tiedosto: `~/.sqliterc`

DB Browser for SQLite

Sivusto: https://sqlitebrowser.org/

- **What it is:** *DB Browser for SQLite* (DB4S) is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.
- **What it is not:** This program is not a visual shell for the sqlite command line tool, and does not require familiarity with SQL commands.



Se löytyy myös PortableAppina! (ei tarvitse asennusta)

IOT AIT TAITOTALO

Johdatus SQL-kieleen

SQL (Structured Query Language)

- Yleiskieli tiedon varastoimiseen, käsittelyyn ja hakemiseen tietokannoista
- Kuvaava kieli, sisältää vain vähän proseduraalisia elementtejä
- Useita eri versioita eri tietokannoille ja käyttötarkoituksiin
- Koostuu käytännössä useista alikielistä: DQL, DDL, DCL , DML, SCL ja TCL

Tietokanta (DB, database)

- Kokoelma tietoja, jotka liittyvät jotenkin toisiinsa (esim. jäsenrekisteri)

DBMS (DataBase Management System, DBMS)

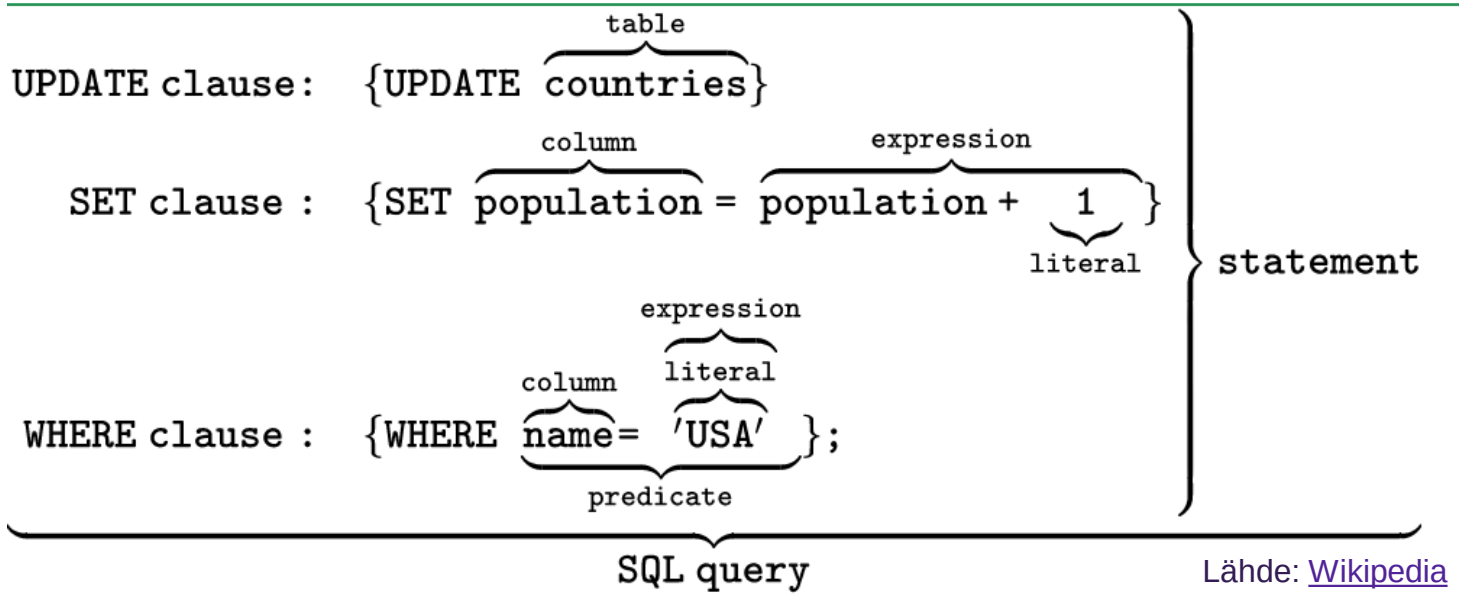
Tietokannanhallintajärjestelmä (TKHJ)

- Sovellus, joka tarjoaa pohjan tietokannan käsittelyyn ja mallintamiseen
- Rakentaa tiedon tietotauluiksi kutsutuiksi kokonaisuuksiksi, jotka muodostuvat sarakkeista ja riveistä

SQL syntaksi

- Vuoteen 1986 mennessä ANSI- ja ISO-standardiryhmät ottivat virallisesti käyttöön standardin "Database Language SQL" -kielen määritelmän. Standardin uudet versiot julkaistiin vuosina 1989, 1992, 1996, 1999, 2003, 2006, 2008, 2011, 2016 ja viimeksi 2023
- Standardin olemassaolosta huolimatta yleisesti SQL-koodi ei ole täysin siirrettävissä eri tietokantajärjestelmien välillä ilman säätöjä.
- SQL-kieli on jaettu useisiin kielielementteihin, mukaan lukien:
  - Clauses
  - Expressions
  - Predicates
  - Queries
  - Statements

SQL syntaksi



Lähde: Wikipedia

SQL syntaksi

- **Avainsanat** ovat sanoja, jotka on määritelty SQL –kielellä. Jotkut niistä ovat standardin varattuja: SELECT, UPDATE, ALTER, …
- **Tunnisteet** ovat nimiä tietokantaobjekteissa kuten taulu, sarake tai skeema. Periaatteessa tunniste ei saa olla sama kuin varattu avainsana, ellei se ole erotettu tunniste. Erotetuilla tunnisteilla tarkoitetaan lainausmerkeissä (tai heittomerkeissä) olevia tunnisteita. Ne voivat sisältää merkkejä, joita SQL-tunnisteet eivät normaalisti tue, ja ne voivat olla identtisiä varatun sanan kanssa.
- MySQL TKHJ:ssä lainausmerkkiä käytetään tekstien määritellyissä. Tunnisteiden kanssa käytetään ns. backtick –merkkiä: `

SQL syntaksi

- SQL standardi määrittelee kaksi kommenttityyppiä:
  - -- kommentti, joka jatkaa vain rivin loppuun
  - /* kommentti */, joka voi sisältää useita rivejä
- SQL kieli ei ole kirjainkokoriippuvainen, joten ei ole väliä kirjoitetaanko asiat pienillä tai isoilla kirjaimilla: SELECT = select = sELeCt
- Merkittämätön välilyönti jätetään yleensä huomitta. Tämä tarkoittaa, että:

```
SELECT * FROM persons WHERE age > 20;
```

on sama kuin:

```
SELECT *
FROM persons
Where
      age > 20
;
```

Eli puolipiste päättää lauseen. **Älä unohda sen!**

On mahdollista kuitenkin, että sellaisissa käyttöliittymissä, jossa kyselyn päättämisen on selkeä, ei tarvitse käyttää puolipistettä. Esimerkiksi DB Browser ohjelmassa.

Ensisijaiset SQL komennot

- CREATE (DATABASE | TABLE)
- INSERT INTO
- SELECT
- UPDATE
- ALTER TABLE
- DELETE FROM
- DROP (DATABASE | TABLE)
- Transaktiot: BEGIN, COMMIT, ROLLBACK
- Käyttöoikeudet: GRANT, REVOKE

SQL syntaksi

SQL-**kysely** muodostuu kolmesta osasta:

1. **Arvot** (sarakkeet), jotka esitetään.
2. **Taulu**(t), joista tietueet löytyvät.
3. **Ehto**, jonka mukaan tietueet valitaan.

SELECT	**Mitä**
FROM	**Mistä**
WHERE	**Miten**

Ehtoa ei ole pakko käyttää!

```
SELECT *
FROM taulu1
WHERE id = 123
```

```
SELECT sarake1, sarake2
FROM taulu1, taulu2
WHERE taulu1.id = taulu2.id
```

```
SELECT sarake1
FROM taulu1
```

***** on KAIKKI-valitsin. Erittäin kätevä, kun harjoitellaan ja tutkitaan hakeeko haku mitä haluttiin.

SQL syntaksi

Kun viittaat kyselyssä johonkin sarakkeeseen, niin voit käyttää sen nimeä suoraan, **JOS** kyselyssä ei ole muita tauluja, joissa on samanniminen sarake.

Jos kyselyssä on kaksi samannimistä saraketta eri tauluissa, viittaa niihin ***taulunnimi.sarake*** –tavalla.

```
SELECT *
FROM taulu1, taulu2
WHERE id = id2
```

```
SELECT *
FROM taulu1, taulu2
WHERE taulu1.id = taulu2.id
```

SQL syntaksi – WHERE vertailu – Perus operaatorit

```
SELECT    *
FROM      taulu1
WHERE     taulu1.sarake2 = 20
```

OPERAATTORI	*SELITE*
=	yhtäsuuri kuin
>	suurempi kuin
<	pienempi kuin
>=	suurempi tai yhtäsuuri kuin
<=	pienempi tai yhtäsuuri kuin
<> tai !=	erisuuri kuin

SQL syntaksi – WHERE vertailu – Lisää operaattoria

OPERAATTORI	*SELITE*
LIKE	Merkkijonon vertailu operaattori
BETWEEN … AND …	Arvojen väliin
IN	Arvo kuuluu listaan
IS NULL	Arvo on NULL
IS NOT NULL	Arvo ei ole NULL

MariaDB:n operaatorit: https://mariadb.com/kb/en/operators/
SQLiten operaatorit: https://www.tutorialspoint.com/sqlite/sqlite_operators.htm
PostgreSQL:n operaatorit: https://www.postgresql.org/docs/current/functions.html

SQL syntaksi - Laskeminen

Kaikki matemaattiset **perus**operaattorit ovat mukana SQL-kielessä

Laskeminen on helppoa. Ja laskuja voidaan käyttää tulosten syöttämiseen ja tulosten rajaamiseen.

Esimerkki (Mitä haetaan?)

WHERE 2020 – Julkaisuvuosi > 20

Operaattori	*Merkitys*
+	yhteenlasku
-	vähennyslasku
*	kertolasku
/	Jakolasku / Kokonaisluvun jakolasku
DIV (MariaDB/MySQL)	Kokonaisluvun jakolasku

MariaDB/MySQL TKHJ:ssä / -operaattori palauttaa **aina desimaaliluku**. Kokonaisluvun jakolasku tehdään DIV –operaattorilla.
Muissa TKHJ:ssä / -operaattorin tulos riippuu operandeista. Jos molemmat ovat kokonaisluvut, tulos on myös kokonaisluku. Jos kumpi niistä on desimaaliluku, tulos on myös desimaaliluku. CAST –funktio auttaa muuttamaan tietotyyppiä.

SQL syntaksi – Ehtojen yhdistäminen: AND / OR (+ NOT)

Hakujen kannalta on tärkeää voida yhdistää hakuehtoja. Koska kyseessä on **totuusarvot** (boolean), niin voimme käyttää normaaleja boolean operaattoreita.

Yleisimmät boolean operaattorit ovat AND eli ja, OR eli tai ja NOT eli ei.

Näiden lisäksi löytyy myös kehittyneemmät NAND ja NOR, jotka eivät kuulu SQL:n perussyntaksiin (vaan ne pitää toteuttaa yhdistelemällä)

Voit hakea lisää tietoa Googlaamalla
Boolen algebra - Boolean algebra

SQL syntaksi – AND / OR (+ NOT)

AND (JA)

Valinta tehdään, jos **MOLEMMAT** ehdot täyttyvät

WHERE id=ssn **AND** age > 30

OR (TAI)

Valinta tehdään, jos **JOMPIKUMPI** ehto täyttyy

WHERE id=ssn **OR** age > 30

SQL syntaksi – AND / OR (+ NOT)

NOT (EI)

Valinta tehdään, jos ehto **EI** täyty**.**

WHERE NOT id = 123

Toinen tapa tehdä tämä on…

WHERE id <> 123

Kaikkia boolean operaattoreita voi myös yhdistää.

SQL:stä puuttuvat loogiset operaattorit NAND ja NOR

SQL syntaksi – BETWEEN … AND …

BETWEEN … AND operaattoria käytetään arvoalueen määrittelemiseen. Voidaan myös aina korvata AND-operaattorilla.

Valinta tehdään, jos arvo on vertaillun arvoalueen sisällä tai vertailuarvo. (<= ja >=)

Esimerkiksi

Henkilö on syntynyt vuosien 1980 ja 1990 välissä

WHERE syntymäaika **BETWEEN** 1980 **AND** 1990

SQL syntaksi – LIKE

LIKE on merkkijonojen (string) vertailuun tarkoitettu operaatio.

Operaation merkkikokoriippuvuus riippuu TKHJ:stä.

WHERE etunimi **LIKE** ’Anssi’

Vertailtava merkkijono

- Rajataan ’ ’ –merkeillä
- % merkitsee mitä tahansa merkkejä. Eli *ei yhtään*, yhtä tai montaa tuntematonta merkkiä
- _ merkitsee **yhtä** tuntematonta merkkiä

Merkkijonoille löytyy myös **NOT LIKE** ja **STRCMP()** operaatiot.

SQLite TKHJ:ssä on olemassa **GLOB** operaattori, joka toimii samalla tavalla kuin LIKE, mutta se käyttää tutut ***** ja **?** –jokerimerkkiä.

SQL syntaksi – LIKE ESIMERKKEJÄ

1. LIKE ’a%’
2. LIKE ’%a’
3. LIKE ’%a%’

4. LIKE ’_a’
5. LIKE ’A__%’

6. LIKE ’A%o’

Mitä kukin vertailu hyväksyy?

a) anssi
b) Apuri
c) As
d) aatami
e) adalmiina
f) Ja
g) ja
h) Anssi

SQL syntaksi – Jokerimerkit (wildcards)

Jokerimerkit (wildcards) ovat merkkejä, joilla voidaan merkitä tuntemattomia merkkejä ja merkkijonoja.

Niiden avulla kyselyihin saadaan monipuolisuutta ja tehokkuutta

Wildcard = jokerimerkki
String = merkkijono
Character, **char** = merkki

Huom! LIKE hyväksy vain % ja _

Merkki	Kuvaus	Esimerkki
%	Esittää ei yhtään, yhtä tai useaa merkkiä	*hau%* löytää *hau*, *haukku* ja *hauki*
_	Esittää yhtä merkkiä	*h_t* löytää *hot*, *hat* ja *hit*
[] (REGEXP)	Esittää mitä tahansa yhtä merkkiä sulkeiden sisältä.	*h[oa]t* löytää *hot* ja *hat*, muttei *hit*
^ (REGEXP)	Esittää mitä tahansa MUUTA merkki kuin on sulkeiden sisällä.	*h[^oa]t* löytää *hit*, muttei *hot* tai *hat*
- (REGEXP)	Esittää merkkiväliä sulkeiden sisällä.	*c[a-b]t* löytää *cat* ja *cbt*

SQL syntaksi – ORDER BY

- Lisäämällä ORDER BY komennon haun loppuun, saat tuloksen järjestettyä haluttuun muotoon.
- Periaatteessa ORDER BY on aina ASCENDING järjestyksessä eli kasvavassa, mutta voit muuttaa sen laskevaan DESC määreellä (Descending).

SELECT etunimi, sukunimi
FROM opiskelijat
ORDER BY etunimi

SELECT etunimi, sukunimi
FROM opiskelijat
ORDER BY etunimi **DESC**

SQL syntaksi – ORDER BY

- Voit myös järjestää tuloksen usean arvon mukaan.

SELECT etunimi, sukunimi
FROM opiskelijat
ORDER BY sukunimi, etunimi

Nyt arvot järjestetään ensin sukunimen perusteella ja sitten etunimen

SQL syntaksi – DISTINCT

DISTINCT määreellä jätät pois toistuvat arvot

SELECT DISTINCT sukunimi
FROM opiskelijat

Saat listan sukunimiä, jotka ovat **yksilöllisiä**.

SQL syntaksi – COUNT-funktio

COUNT –funktiolla voit laskea tietueiden (eli rivien) lukumäärän.

Tämä on kätevää, jotta saat esimerkiksi tiedon kuinka moni opiskelija on suorittanut kurssin tai kuinka monta henkilöä on kurssilla.

SELECT COUNT(*)

FROM opiskelijat

WHERE kurssi = ”tietokannat”

Saat tulokseksi luvun, siitä kuinka monta opiskelijaa on tietokannat-kurssilla.

Kyselyn vaiheet

SQL kyselyn avulla tieto haetaan, järjestetään ja esitetään halutulla tavalla

1. Mitä haluat esittää? **SELECT**
2. Mistä tauluista löydät sen? **FROM**
3. Mitkä ehdot vallitsevat? **WHERE**
4. Miten tiedot järjestetään? ***GROUP BY***, **ORDER BY**

SQL syntaksi – kooste- ja skalaarifunktiot

Kooste- ja Skalaarifunktiot (aggregaattifunktiot)

- Keräävät halutut tiedot yhteen ja suorittavat niillä laskentaa
- **EIVÄT** huomio rivejä, joissa verrattava sarake on **NULL**
- **Koosteet: COUNT(), SUM() ja AVG()**
- **Skalaarit: MIN(), MAX(), ABS(), MOD(), ROUND()**

Skalaari = suure eli matemaattinen yksittäistapaus, joka voidaan esittää **mittaluvulla**

Aggregaatti = yhdistelmä, kokouma

SQL syntaksi – kooste- ja skalaarifunktiot

AVG()	lasketaan halutun sarakkeen **keskiarvon**	
MIN()	vertailee sarakkeen **pienin** arvo	
MAX()	vertailee sarakkeen **suurin** arvo	
SUM()	**lasketaan yhteen** tietyt arvot	
ABS()	lasketaan arvojen **itseisarvot**	-1 = 1, 3 = 3, -8 = 8
MOD()	esittää lukujen **jakojäännöksen**	MOD(minuutti, 60)
ROUND()	esittää **liukuluvun** halutulla tarkkuudella	(luvut, tarkkuus)

Liukuluku = desimaaliluku (suurin piirtein sama)

MOD-funktio (jakojäännös)

MOD on useissa ohjelmointikielissä käytetty komento, jolla saadaan jäljelle jakojäännös.

Mod kaava merkitään joko

luku1 % luku2	***5 % 2***
luku1 MOD luku2	***5 MOD 2***
MOD(luku1, luku2)	***MOD(5, 2)***

Jakolasku	**Jakojäännöslasku**
5/2 = 2,5	**5%2 = 1**

Mod-funktiot ovat tärkeitä mm. kun aikayksiköitä muutetaan matemaattisesti laskettaviksi.

Aloitat työpäivän kello 9.05. Työpäivän kesto on 7,67 tuntia. Mitä kello on, kun pääset töistä?

Mitä kello on 12.05 + 580 minuuttia?

SQL syntaksi – koonti

- Koonti
  - Keräävät halutut tiedot yhteen.
  - Hyödynnetään usein koostefunktioiden (sum, avg,..) kanssa
  - **GROUP BY**

- HAVING-määre
  - Käytetään samoin kuin WHERE-määrettä
  - Kohdistuu koostefunktioon

SQL syntaksi – GROUP BY 1/2

GROUP BY kokoaa koostefunktioilla kerätyn tiedon esittämistä varten. Pitää kertoa miten koostefunktion tieto ryhmitellään

```
SELECT    sarake1, SUM(sarake2)
FROM      taulu1
GROUP BY  sarake1
```

Nyt koostefunktion SUM(kesto) tiedot ryhmitellään arvo1-sarakkeen mukaan.

SQL syntaksi – GROUP BY + HAVING 2/2

HAVING määre toimii koostefunktioiden kanssa kuten WHERE-määre muissa kyselyissä.

```
SELECT    sarake1, SUM(kesto)
FROM      taulu1
GROUP BY  sarake1
HAVING    SUM(kesto) > 30
```

Nyt koostefunktion SUM(kesto) tiedot ryhmitellään arvo1-sarakkeen mukaan **JA ESITETÄÄN VAIN** tiedot, joissa SUM(kesto) on suurempi kuin 30.

SQL syntaksi – Sarakkeiden uudelleen nimeäminen

Uudelleen nimeäminen Aliaksella eli AS-komennolla

- Pyritään helppolukuisuuteen antamalle tauluille, kentille (ja funktioille) uudet nimet
- Erittäin tärkeä ominaisuus, kun kenttiä on paljon ja/tai niiden nimet ovat haastavia
- Esimerkiksi VanhaSarakeNimi **AS** UusiSarakeNimi

Sopii

Kenttien ja taulujen uudelleen nimeäminen
Apukenttien tekeminen
Laskutoimituskenttien nimeäminen
Tuloskenttien nimeäminen

Tilaaja_Kirja_Kirjasto AS kirja
’on töissä’ AS Apukenttä
COUNT(*) AS Kokonaismäärä

SQL syntaksi – Sarakkeiden/taulujen uudelleen nimeäminen (Alias)

Alias eli AS

- Usein näkee AS –nimeämistä käytettävän myös **ilman AS määrettä**.
- Useimmat SQL-ympäristöt hyväksyvät tämän

SELECT a.etunimi, b.etunimi
FROM *työntekijä a, työntekijä b* ← *Tässä käytetty*
WHERE a.id = b.esimies_id

SQL syntaksi – Sarakkeiden arvojen uudelleen nimeäminen

COALESCE

- Funktion avulla
- **NULL** arvojen huomioiminen tuloksissa
- COALESCE (SarakeNimi, ’NULL:in korvaava merkkijono’)

Harjoitus inklusiivisuus ja eksklusiivisuus

- Inklusiivisuus
  - Sisällyttäminen
  - Mainittu arvo otetaan mukana tulokseen.

  Kaikki tapaukset, jotka maksavat vähintään 50 euroa.

  = kaikki yli 50 € ja 50 €.
  >= 50

- Eksklusiivisuus
  - Poissulkeminen
  - Mainittu arvo jätetään pois tuloksesta.

  Kaikki tapaukset, jotka maksavat vähemmän kuin 50 euroa.

  = kaikki alle 50 €
  < 50

Ongelma syntyy luonnollisista kielistä, kuten suomesta, jotka eivät ole kovin tarkkoja ja täsmällisiä.

Tehtäviä

Esimerkki

Haetaan kaikki kirjat, joiden hinta on alle 30 euroa.

SELECT * FROM kirjat WHERE hinta < 30 eksklusiivinen

a) Haetaan kaikki kirjat, joiden sivumäärä on vähintään 100 sivua.
b) Haetaan kaikki CD-levyt, joissa on enintään 10 raitaa.
c) Haetaan kaikki työntekijät pois lukien henkilöt, joiden ikä on vähintään 50 vuotta.
d) Haetaan kaikki kirjat, joiden hinta korkeintaan 20 euroa. Jätetään tuloksesta pois kirjat, joiden sivumäärä on yli 500 sivua.
e) Haetaan kaikki artistit, jotka julkaisseet 3 albumia tai enemmän.

Tehtäviä

a) Haetaan kaikki kirjat, joiden sivumäärä on vähintään 100 sivua.
Select * from kirjat where sivuja >= 100 inklusiivinen

b) Haetaan kaikki CD-levyt, joissa on enintään 10 raitaa.
Select * from albumit where raita <= 10 inklusiivinen

c) Haetaan kaikki työntekijät pois lukien henkilöt, joiden ikä on vähintään 50 vuotta.
Select * from tyontekijatwhere ika < 50 eksklusiivinen

d) Haetaan kaikki kirjat, joiden hinta korkeintaan 20 euroa. Jätetään tuloksesta pois ne kirjat, joiden sivumäärä on yli 500 sivua.
Select * from kirjat where hinta <= 20 AND sivuja <= 500 ink + ink

e) Haetaan kaikki artistit, jotka julkaisseet 3 albumia tai enemmän.
Select * from artistit having count(albumi) >= 3 inklusiivinen

TAITOTALO

Tietokannat ja taulut

SQL syntaksi – tietokannan luominen

CREATE DATABASE nimi

komennolla luodaan uusi (tyhjä) tietokanta.

SQL-kielinen luonti ei sisällä muita tietoja, kuin tietokannan nimen.

SQL syntaksi – tietokannan tuhoaminen

DROP DATABASE nimi

komennolla tuhotaan tietokanta

Muuta tietoa ei tarvita kuin tuhottavan tietokannan nimi.

SQL syntaksi – taulujen luominen

```
CREATE TABLE [IF NOT EXIST] nimi (
    sarake1    tietotyyppi     [sarakkeen rajoitukset],
    sarake2    tietotyyppi     [sarakkeen rajoitukset],
    [taulun rajoitukset]
);
```

komennolla luodaan uusi (tyhjä) taulu. Komennon syntaksi voi vaihdella TKHJ:n mukaan

Tyypillisiä sarakkeen tai taulun rajoituksia:

- PRIMARY KEY
- FOREIGN KEY (vain taululle)
- UNIQUE
- NOT NULL (vain sarakkeelle)
- CHECK ehto
- DEFAULT arvo (vain sarakkeelle)

SQL syntaksi – taulun tuhoaminen

DROP TABLE nimi

komennolla tuhotaan taulu

Muuta tietoa ei tarvita kuin tuhottavan taulun nimi.

SQL – Tietotyyppiä

- SQLite: https://www.sqlite.org/datatype3.html
- MariaDB: https://mariadb.com/kb/en/data-types/
- PostgreSQL: https://www.postgresql.org/docs/current/datatype.html

SQL syntaksi – tietuen lisääminen

- INSERT INTO komennolla saadaan lisättyä uuden tietuerivin tauluun.

```
INSERT INTO taulu [(sarake1, sarake2, …)]
VALUES (arvo1, arvo2, …);
```

- Jos sarakkeita ei mainita, pitää antaa arvoa kaikille sarakkeille.
- Sarakkeet ei tarvitse olla samassa järjestyksessä, jolla ne on määritetty
- Useita rivejä voidaan lisätä yhdellä komennolla:

```
INSERT INTO taulu [(sarake1, sarake2, …)]
VALUES (arvo1, arvo2, …), (arvo3, arvo4), …;
```

SQL syntaksi – taulujen luominen - UNIQUE

- UNIQUE määrittää, että ko. sarakkeelle tai sarakkeille ei voi tulla samaa arvoa useita kertoja

```
CREATE TABLE nimi (
    sarake1 INTEGER UNIQUE,
    …
);
```

```
CREATE TABLE nimi (
    sarake1 INTEGER,
    sarake2 INTEGER,
    …
    UNIQUE (sarake1, sarake2)
);
```

NULL arvot pidetään erillisenä niiden välillä, joten saa olla useita NULL arvoa sarakkeessa.

SQL syntaksi – taulujen luominen – NOT NULL

- NOT NULL määrittää, että ko. sarakkeessa ei saa olla NULL arvoa

```
CREATE TABLE nimi (
    sarake1 INTEGER NOT NULL,
    …
);
```

Huom! SQL standardien mukaan, PRIMARY KEY pitää sisältää NOT NULL, mutta SQLite TKHJ:ssä ei seuraa sitä. Eli SQLitessä tarvitaan NOT NULL PRIMARY KEY yhteydessä.

SQL syntaksi – taulujen luominen - CHECK

- CHECK määrittää sarakkeelle tai sarakkeille ehtoa/ehtoja, jotka sarakkeen arvot pitää seurata

```
CREATE TABLE nimi (
    sarake1 INTEGER CHECK (lauseke),
    …
);
```

```
CREATE TABLE nimi (
    sarake1 INTEGER,
    sarake2 INTEGER,
    …
    CHECK (lauseke)
);
```

SQL syntaksi – taulujen luominen - DEFAULT

- DEFAULT määrittää sellainen arvo, joka annetaan sarakkeelle jos INSERT komento ei antaa mitään arvoa sille

```
CREATE TABLE nimi (
    sarake1 INTEGER DEFAULT 100,
    …
);
```

SQL syntaksi – taulujen luominen – PRIMARY KEY

- PRIMARY KEY määrittelee sarake tai sarakkeet, jotka toimivat taulun pääavaimena

```
CREATE TABLE taulu (
    sarake1 INTEGER PRIMARY KEY,
    …
);
```

```
CREATE TABLE taulu (
    sarake1 INTEGER,
    sarake2 INTEGER,
    …
    PRIMARY KEY (sarake1, sarake2)
);
```

- PRIMARY KEY sisältää periaatteessa NOT NULL ja UNIQUE rajoitukset.
- Huom! SQLite:ssä pitää lisätä NOT NULL erikseen.

SQL syntaksi – taulujen luominen - Autoincrement

- PRIMARY KEY hyväksy sellainen automaattisesti nouseva numero käsky (AUTOINCREMENT SQLite:ssä, AUTO_INCREMENT MariaDB:ssa)

```
CREATE TABLE taulu (
    sarake1 INTEGER PRIMARY KEY AUTOINCREMENT,
    …
);
```

Tällä tavalla ei tarvitse syöttää arvoa tälle sarakkeelle, koska se ottaa automaattisesti seuraava numeroarvo (aloittaa 1)

SQL syntaksi – taulujen luominen – FOREIGN KEY

- FOREIGN KEY määrittelee, että sarake on pääavain toisessa taulussa

```
CREATE TABLE taulu1 (
    sarake1 INTEGER PRIMARY KEY,
    sarake2 INTEGER,
    …
    FOREIGN KEY (sarake2) REFERENCES taulu2 (sarake3)
        [ON DELETE action]
        [ON UPDATE action]
);
```

”action” voi olla esimerkiksi: SET NULL, SET DEFAULT, RESTRICT, NO ACTION, CASCADE

FOREIGN KEY voidaan käyttää useita kertaa

Tietueen poistaminen – DELETE

DELETE komennolla poistetaan taulusta ehdon täyttävät tietuerivit.

DELETE FROM taulu
WHERE ehto

DELETE FROM opiskelijat
WHERE sukunimi=”Aaltonen”

Taulu käydään läpi, ja kaikki ehdon täyttävät tietueet poistetaan.

HUOM! Tosi tärkeä! Älä koskaan unohda laittaa WHERE –osaa!

Tietueen päivittäminen – UPDATE

UPDATE komennolla päivitetään olemassa olevan tietuerivin arvoja.

```
UPDATE      kirjailijat
SET         kirjailijat_sukunimi = ’Turunen’
                                   //Sarake johon muutos kohdistuu
WHERE       kirjailijat_tekija_id = 204
                                   //Ehto: rivi, mihin muutos kohdistuu
```

Taulu käydään läpi, ja kaikkien ehdon täyttävien tietuerivien arvo2 korvataan annetulla arvolla.

HUOM! Tosi tärkeä! Älä koskaan unohda laittaa WHERE –osaa!

Tietueen päivittiminen – UPDATE

Arvojen muuttaminen matemaattisesti.

Updatea voidaan käyttää myös lisäämään kaikkiin tietueisiin uusi kenttä matemaattisin operaation.

Tämä voi olla esimerkiksi

- Juokseva numerointi (mahdollisesti ID)
- Uusi arvo, joka on johdettu muista arvoista

UPDATE opiskelija
SET sarake1 = sarake1 + 1

UPDATE opiskelija
SET ikä = (**NOW**() - syntymäaika)