Tampereen yliopisto

Tietokantojen perusteet

Kertausta ja vähän uuttakin asiaa

Kalvot perustuvat kirjaan Elmasri, Navathe: Fundamentals of Database Systems.

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 suhdetyyppeinä - 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 käytettävissä 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!

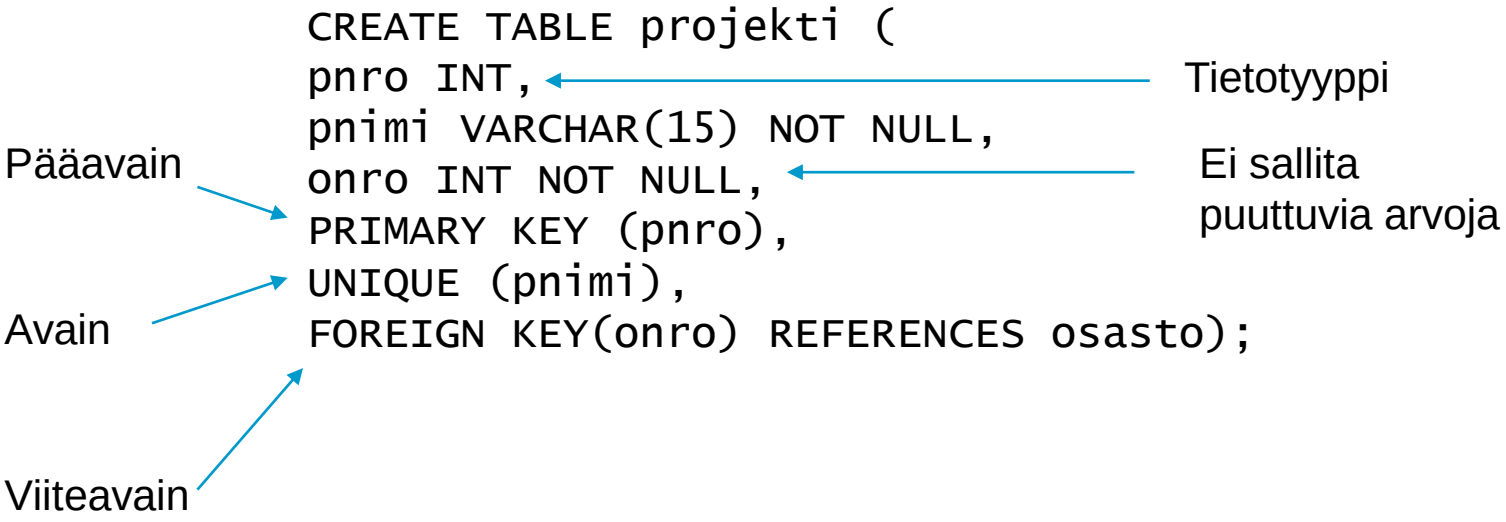
Muunnos ER-kaaviosta SQL-tietokannan kaavioksi

- 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.
- Luennoilla ja harjoituksissa käytiin 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.



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: Tietotyypeistä

- 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: Tietotyypeistä (ei tenttiin)

- Pääavaimina käytetään usein kokonaislukutyyppisiä keinotekoisia tunnuksia.

- Kokonaislukutyyppisiä avaimen arvoja voidaan generoida automaattisesti: tietokannanhallintajärjestelmä generoi avaimen arvon automaattisesti tietoja lisättäessä.
  - tietotyyppi vaihtelee järjestelmittäin
  - SQLitessa AUTOINCREMENT tai INT-tyyppinen pääavainsarake
  - PostgreSQL:ssä tämä tietotyyppi on SERIAL

SQL: Tietotyypeistä (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: Tietotyypeistä (ei tenttiin)

- Jos haluaa tehdä siirrettäviä SQL-kantoja, tulisi pitäytyä 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 rikkoa 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 seurauksena, 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, seurauksena 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 tietotyyppejä
  - 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 lempivari VARCHAR(20);
```

Sarakkeen poisto

```
ALTER TABLE taulu
DROP sarake
```

```
ALTER TABLE tyontekija
DROP lempivari;
```

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 tukee.
- 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 **liitosehto**
  - Halutaanko tulokseen vain tietyt rivit?
    - WHERE-osan valintaehdot
  - Tehdäänkö yhteenvetoja?
    - COUNT, MIN, MAX, SUM, AVG
    - Tehdäänkö yhteenvetoja ryhmittäin?
      - GROUP BY
    - Asetetaanko ryhmille vaatimuksia?
      - HAVING
  - Täytyykö tulosrivit järjestää?
    - ORDER BY
  - JA VIELÄ: kiinnitä huomiota tyhjäarvojen käsittelyyn!!!

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 rutiininsa eri tavalla määriteltyjen kyselyjen käsittelyyn.
  - 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 skalaarifunktioita (ei tenttiin)

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

||-merkkijonofunktio yhdistää merkkijonoja

| on pystyviiva 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 skalaarifunktioita (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 tietokannanhallintajärjestelmittäin (ja järjestelmän versioittain).
- Jos haluaa tehdä siirrettäviä SQL-lauseita, kannattaa funktioita käyttää ”säästeliäästi” (tai välttää niiden käyttöä).

Tietokannanhallintajärjestelmistä ja suunnitteluohjelmistoista

- 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