CS-A1150 Tietokannat

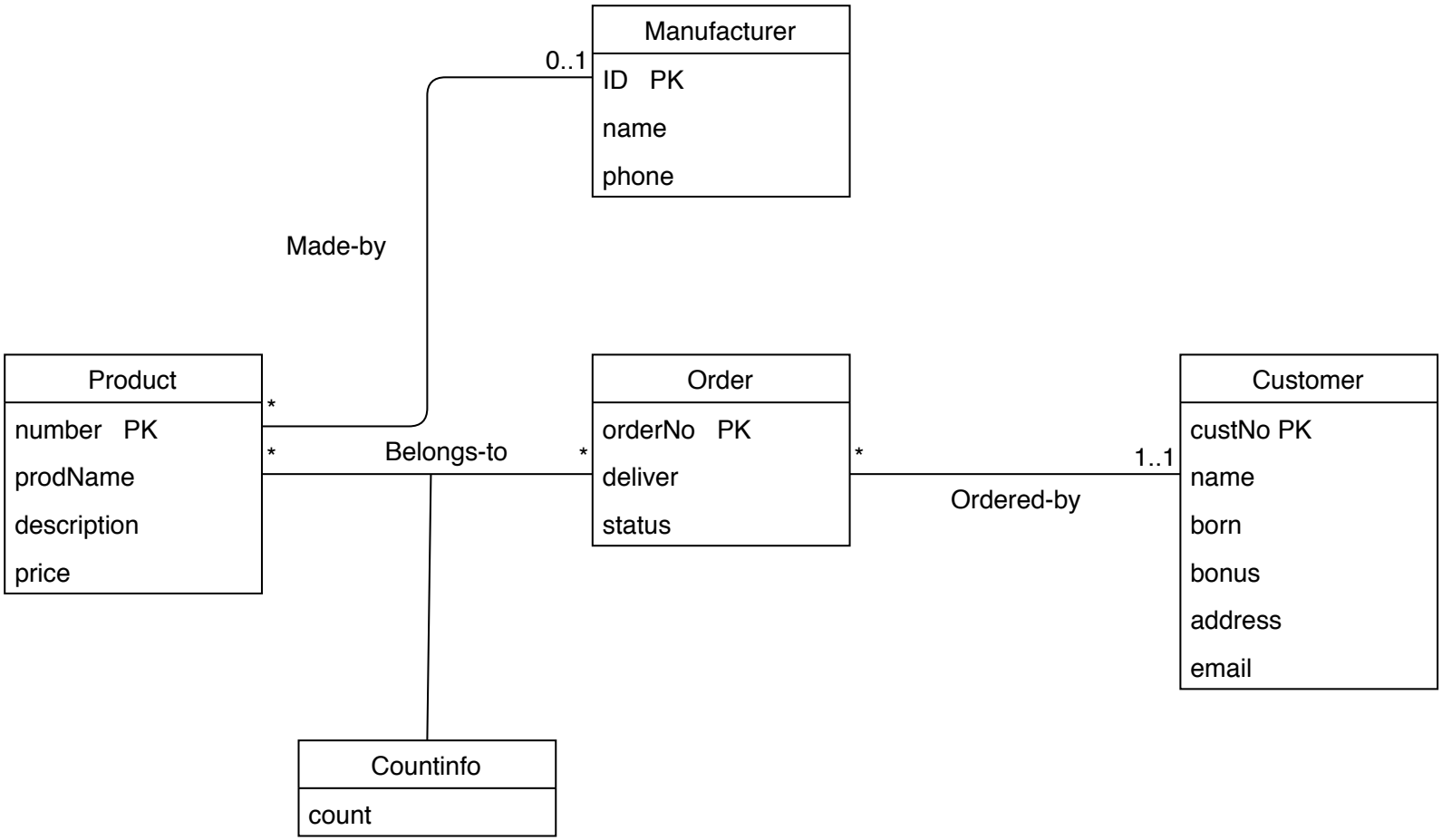
12.3.2019

Oppimistavoitteet: tämän luennon jälkeen

- Osaat muuttaa UML-kaavion relaatiomalliin. Toisin sanoen: jos sinulla on valmis UML-kaavio, niin tiedät, mitä relaatioita tietokantaan pitää määritellä ja mitä attribuutteja näillä relaatioilla pitää olla (tietokantakaavio).
- Osaat kertoa, miksi jokin tietokantakaavio on parempi kuin toinen samaa asiaa kuvaava tietokantakaavio.
- Ymmärrät joukon asioita, joita tarvitaan, kun huono tietokantakaavio muutetaan paremmaksi (tästä lisää ensi viikolla), esimerkiksi:
  - funktionaalinen riippuvuus
  - relaation avaimen suhde funktionaalisiin riippuvuuksiin
  - attribuuttien sulkeuma.

UML-kaavion muuttaminen relaatiokaavioiksi

- Mitä relaatioita määrittelisit tietokantaan tämän kaavion perusteella?



UML-kaavion muuttaminen relaatiokaavioiksi, jatkoa

- Perusperiaatteet:
  - Jokaisesta luokasta tehdään relaatio, jolla on samat attribuutit kuin itse luokalla.
  - Jokaisesta assosiaatiosta tehdään relaatio, jonka attribuutteina ovat assosiaation yhdistämien luokkien avaimet sekä mahdollisen assosiaatioluokan attribuutit.
- Erikoistapaukset (joita ei voida muuttaa suoraan em. perusperiaatteiden mukaan):
  - Luokat, joiden avaimesta osa tulee toisen luokan attribuuteista ja näiden väliset assosiaatiot pitää käsitellä eri tavalla.
  - Aliluokat pitää käsitellä eri tavalla.
  - Joskus voidaan yhdistää kaksi perussääntöjen perusteella muodostuvaa relaatiota.

Esimerkki perustapauksesta

- Verkkokaupan UML-kaavion luokista muodostetaan relaatiot

  `Customers(`<u>`custNo`</u>`, name, born, bonus, address, email)`
  `Products(`<u>`number`</u>`, name, description, price)`
  `Manufacturers(`<u>`ID`</u>`, name, phone)`
  `Orders(`<u>`orderNo`</u>`, deliver, status)`

- Kaavion assosiaatioista tehdään relaatiot

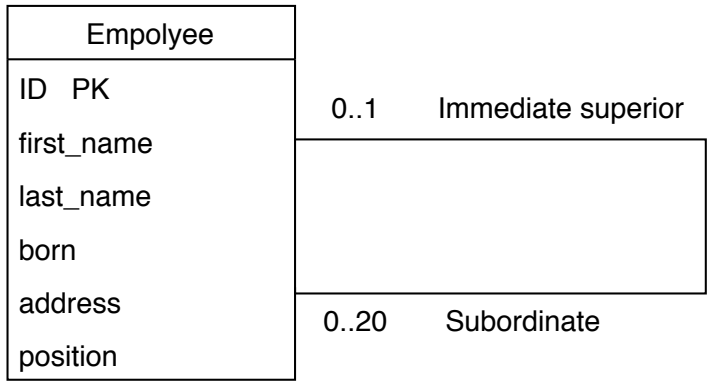
  `MadeBy(`<u>`number`</u>`, ID)`
  `BelongsTo(`<u>`orderNo`</u>`, `<u>`number`</u>`, count)`
  `OrderedBy(`<u>`orderNo`</u>`, custNo)`

Relaatioiden avaimista

- Luokasta muodostetulla relaatiolla avainattribuutit ovat samat kuin luokalla.
- Monesta moneen -assosiaatiosta muodostetulla relaatiolla avainattribuuteiksi tulevat assosiaation yhdistämien luokkien avainattribuutit.
- Monesta yhteen -assosiaatiosta muodostetulla relaatiolla avainattribuuteiksi tulevat vain monesta-puolen luokan avainattribuutit.
- Avainattribuutteja ei saa määritellä ylimääräisiä, koska avainattribuuttien pidetään huoli siitä, että relaatioon ei lisätä kahta monikkoa, jolla on samat arvot avaimelle. Jos esimerkiksi relaatiolla *MadeBy* olisi avainattribuuttina myös *ID*, voitaisiin relaatioon lisätä useita monikoita, joissa tuote on sama, mutta valmistaja eri.

Toinen esimerkki

- Sama luokka esiintyy assosiaatiossa kahteen kertaan:



  Luokan avainattribuutit tulevat relaatioon kahteen kertaan kuvaamaan kumpaakin assosiaation osapuolta.

- Assosiaatiosta muodostetaan relaatio

  `Manages(subordinateID, superiorID)`

Relaatioiden yhdistämisestä

- Ensimmäisessä esimerkissä UML-kaavion pohjalta luotiin mm. relaatiot

  `Products(`<u>`number`</u>`, name, description, price)`
  `Manufacturers(`<u>`ID`</u>`, name, phone)`
  `MadeBy(`<u>`number`</u>`, ID)`
- Relaatio *MadeBy* on syntynyt monesta yhteen -assosiaatiosta.
- Relaatiot *Products* ja *MadeBy* voidaan yhdistää yhdeksi relaatioksi lisäämällä relaatioon *Products* tieto valmistajan tunnuksesta.

  `Products(`<u>`number`</u>`, name, description, price, ID)`

  Relaatio *Manufacturers* jää entiselleen.

Relaatioden yhdistämisestä, jatkoa

- Vastaavasti voidaan yhdistää relaatiot *Orders* ja *OrderedBy*.
- Sen sijaan relaatiota *BelongsTo* ei voida yhdistää toiseen relaatioon, koska se on syntynyt monesta moneen -assosiaatiosta.
- Lopulliset relaatiot (joidenkin attribuuttien nimiä on muutettu selvyyden vuoksi):

```
Customers(custNo, name, born, bonus, address, email)
Products(number, prodName, description, price, manufID)
Manufacturers(ID, manufName, phone)
Orders(orderNo, deliver, status, custNo)
BelongsTo(orderNo, prodNo, count)
```

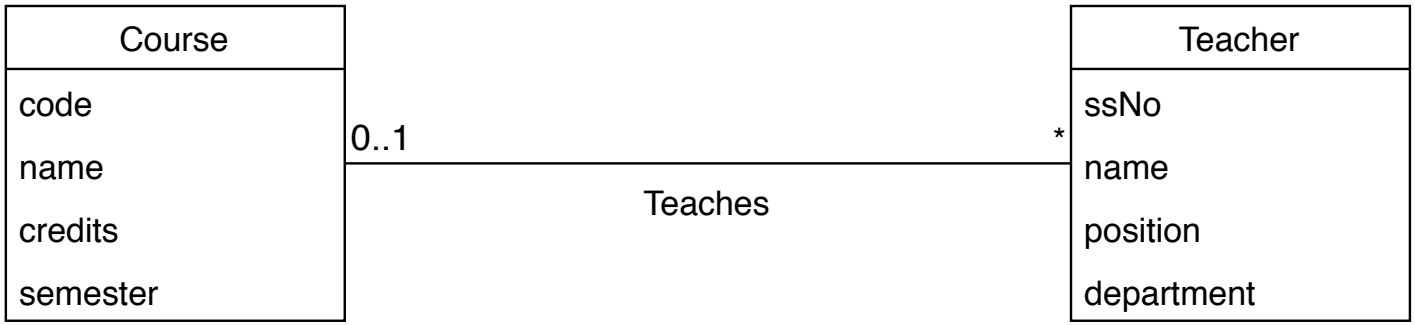
Relaatioden yhdistämisestä, jatkoa

- Jos assosiaatio on monesta moneen, relaatioita ei voida yhdistää.
- Yhdistäminen johtaisi siihen, että sama tieto toistetaan moneen kertaan.
- Jos *BelongsTo*-relaatio yhdistettäisiin *Orders*-relaatioon, tuloksena olisi relaatio

  `Orders(orderNo, deliver, status, custNo, prodNo, count)`
- Saman tilauksen toimitustapa, tila ja tilaajan asiakasnumero olisi tallennettu tietokantaan useaan kertaan.

Välitehtävä

- Mitkä seuraavan kalvon tietokantakaavioista ovat oikein, kun alla oleva UML-kaavio muutetaan relaatioiksi?



- Vastaa Presemossa `http://presemo.aalto.fi/tietokannat2019`

Välitehtävä jatkuu

- Vaihtoehdot:
  a. Teachers(ssNo, name, position, department)
     Courses(code, name, credits, semester)
     Teaches(ssNo, code)
  b. Teachers(ssNo, name, position, department)
     Courses(code, name, credits, semester)
     Teaches(code, ssNo)
  c. Teachers(ssNo, name, position, department, coursecode)
     Courses(code, name, credits, semester)
  d. Teachers(ssNo, name, position, department)
     Courses(code, name, credits, semester, teacherSsNo)

Välitehtävän ratkaisu

- UML-kaavion mukaan yksi opettaja voi opettaa korkeintaan yhtä kurssia, mutta yhdellä kurssilla voi olla useita opettajia.
  - Vaihtoehto a on oikein, koska siinä relaation `Teaches` avain on opettajan tunnus.
  - Vaihtoehto b on väärin, koska siinä relaation `Teaches` avain on on kurssikoodi, mutta tämä ei käy, koska samalla kurssilla voi olla useita opettajia (jokaista kurssi–opettaja-yhdistelmää kohti tehdään oma monikko).
  - Vaihtoehto c on oikein, koska siinä opettajan tietoihin on yhdistetty tieto siitä, mitä kurssia opettaja opettaa (opettajalla voi olla korkeintaan yksi kurssi).
  - Vaihtoehto d on väärin, koska siinä kurssin tietoihin on yhdistetty tieto opettajasta, mikä ei toimi, koska kurssilla voi olla useita opettajia.
- Oikea vastaus on siis a ja c.

Koska relaatioiden yhdistäminen on järkevää?

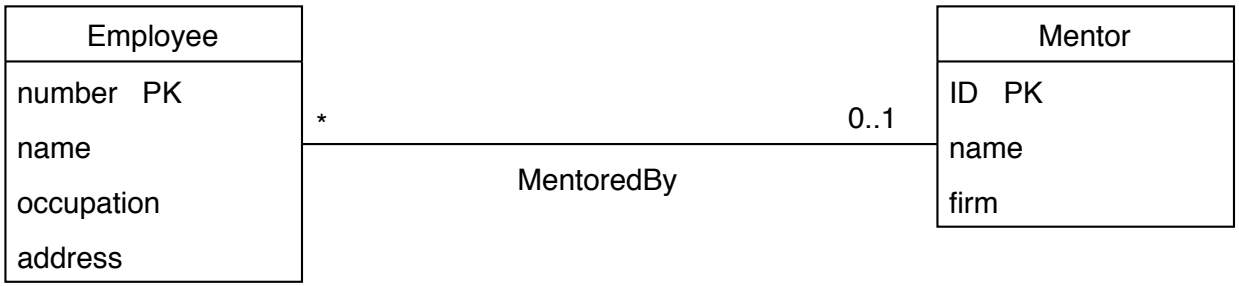
- Monesta yhteen -assosiaatiosta tehty relaatio voidaan siis yhdistää monesta-puolen relaatioon. Koska näin kannattaa tehdä?
- Relaatioiden yhdistäminen johtaa siihen, että tietokantakaaviossa on vähemmän relaatioita. Tämä yksinkertaistaa relaatiokaaviota, mikä on yleensä toivottavaa. Usein siis yhdistäminen on järkevää.
- Jos kuitenkin vain hyvin pieni osa luokan olioista liittyy assosiaation kautta toisen luokan olioon, johtaa yhdistämineen relaatioon, jossa on useimmilla monikoilla NULL-arvoja assosiaation kautta tulevien attribuuttien paikalla. Tällöin yhdistäminen ei välttämättä kannata, koska:
  - Assosiaation kautta tulevien attribuuttien tallentaminen vie ylimääräistä tilaa, jos lähes kaikilla monikoilla ko. attribuuttien arvo on NULL.
  - Jos assosiaation tietoihin tehdään paljon kyselyitä, voi olla nopeampaa tehdä kyselyt pelkästään assosiaatiota kuvaavaan relaatioon, jos sen koko on selvästi pienempi kuin luokkaa kuvaavan relaation koko. (Tämä riippuu kuitenkin siitä, mitä muita relaatioita kyselyssä tarvitaan.)

Esimerkki 1

- Lähes kaikilla tuotteilla on tieto siitä, mikä valmistaja ne on tuottanut.
- *MadeBy*-assosiaatiosta syntyneen relaation tiedot kannattaa yhdistää relaatioon *Products*, koska tietokantakaavio yksinkertaistuu eikä *Products*-relaation monikoilla juuri ole NULL-arvoja `manufID`-attribuutilla.

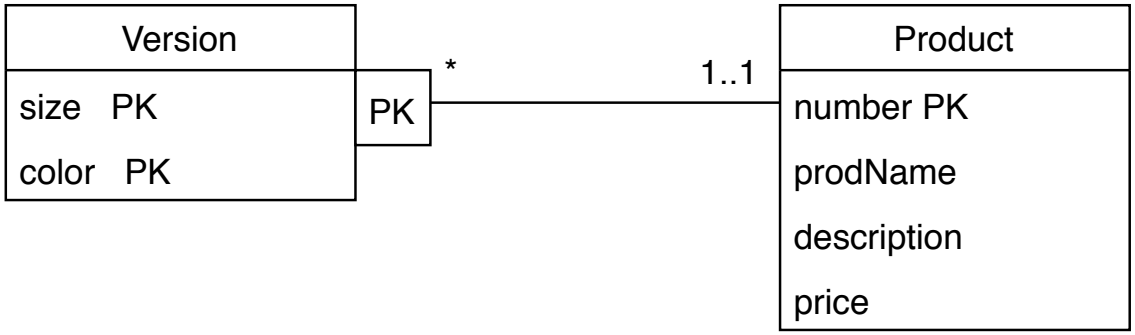
Esimerkki 2

- Oletetaan, että työpaikalla on mentorointiohjelma, jossa osalla työntekijöistä on nimetty mentori työpaikan ulkopuolelta.
- Yhdellä työntekijällä voi olla korkeintaan yksi mentori, mutta yli 90 %:lla työntekijöistä ei ole lainkaan mentoria



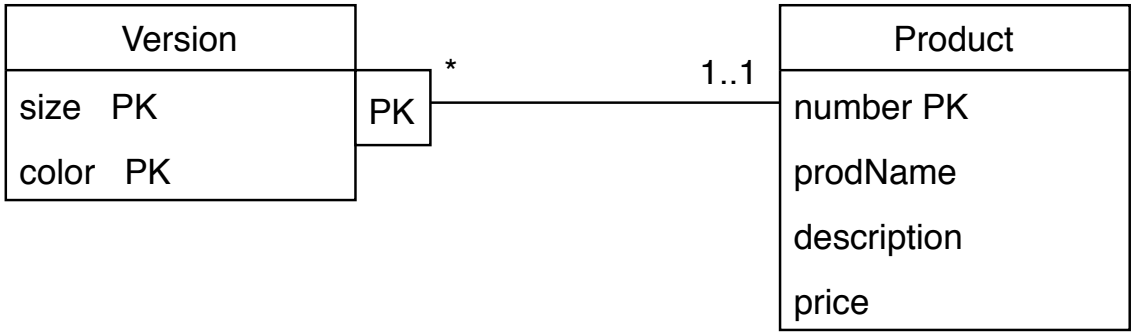
- Koska vain pienellä osalla työntekijöistä on mentori, kannattaa assosiaatiosta *MentoredBy* tehdä todennäköisesti oma relaationsa eikä yhdistää sitä relaatioon *Employees*.

Kun luokan omat attribuutit eivät riitä avaimeksi



- Luokan *Version* avainattribuuttien lisäksi relaation *Version* avainattribuutiksi otetaan luokan *Product* avainattribuutti.
- Luokkien *Version* ja *Product* välisestä assosiaatiosta ei tehdä koskaan omaa relaatiota.
- Jos luokka *Version* on osallisena jossakin toisessa assosiaatiossa, niin kyseisestä assosiaatiosta muodostetun relaation täytyy sisältää sekä luokan *Version* että luokan *Product* avainattribuutit.

Kun luokan omat attribuutit eivät riitä avaimeksi



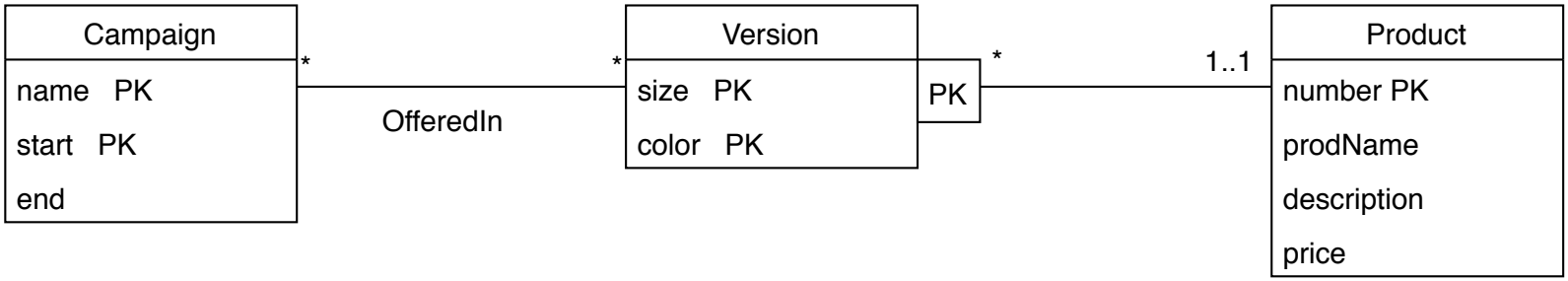
- Luokan *Version* avainattribuuttien lisäksi relaation *Version* avainattribuutiksi otetaan luokan *Product* avainattribuutti.
- Luokkien *Version* ja *Product* välisestä assosiaatiosta ei tehdä koskaan omaa relaatiota.
- Jos luokka *Version* on osallisena jossakin toisessa assosiaatiossa, niin kyseisestä assosiaatiosta muodostetun relaation täytyy sisältää sekä luokan *Version* että luokan *Product* avainattribuutit.
- Relaatiot

  `Products(number, name, description, price)`

  `Versions(prodNo, size, color)`

Luokka *Version* tavallisen assosiaation osapuolena

- Oletetaan, että verkkokaupassa joitakin tuotteen versioita on tarjolla vain määräaikaisissa kampanjoissa.

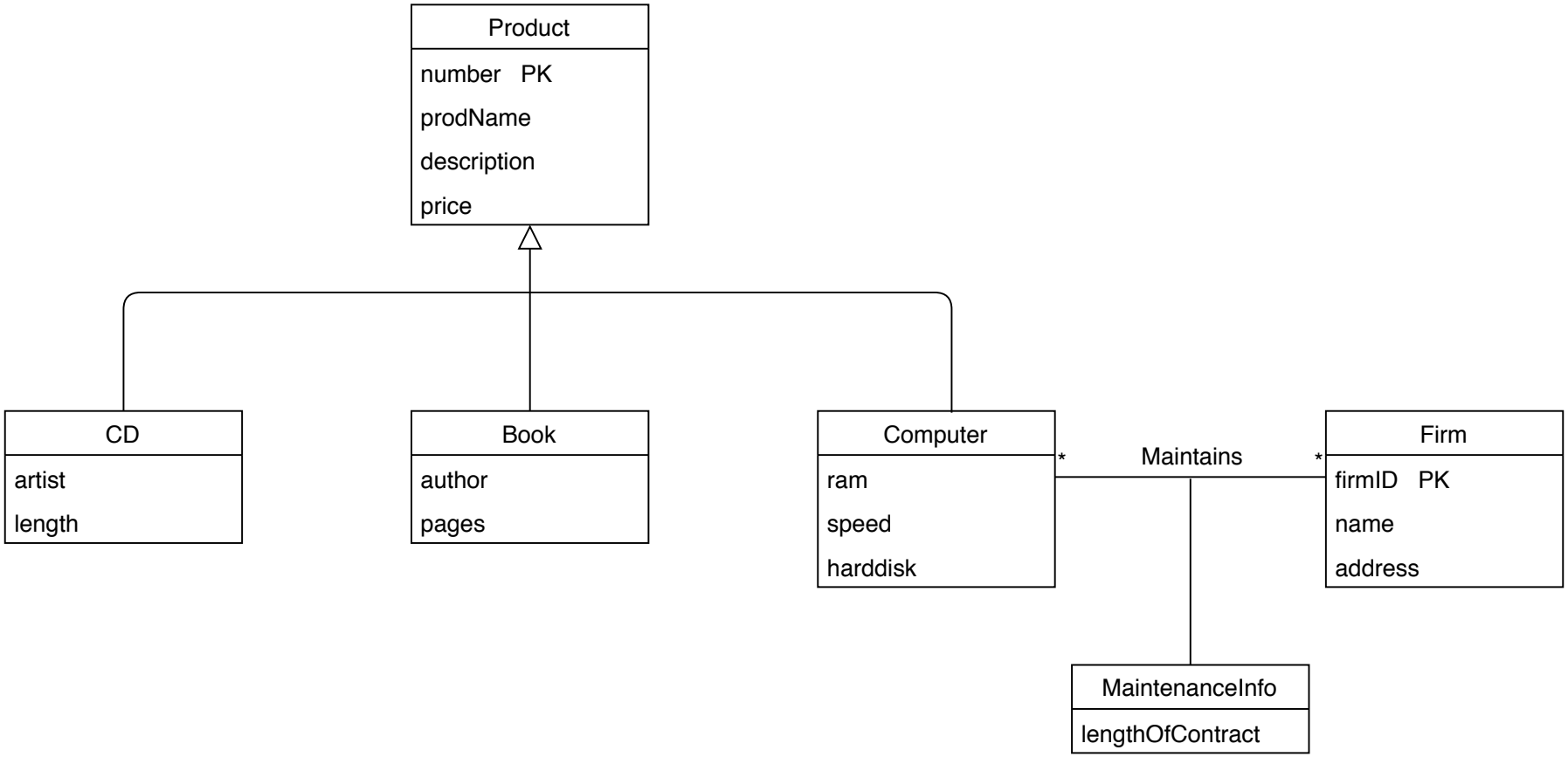


- Kun assosiaatiosta *OfferedIn* muodostetaan relaatio, otetaan relaatioon mukaan luokkien *Versions* ja *Campaigns* avainattribuuttien lisäksi myös luokan *Products* avainattribuutti, koska se tarvitaan version tunnistamiseen.
- Relaatiot

  `Products(number, name, description, price)`
  `Versions(prodNo, size, color)`
  `Campaigns(name, start, end)`
  `OfferedIn(prodNo, size, color, campaignName, start)`

Perintähierarkian esittäminen relaatioina

- Miten yliluokka ja sen aliluokat esitetään relaatioina?



Perintähierarkian esittäminen relaatioina, jatkoa

- UML-kaavio voidaan muuttaa relaatiomalliksi kolmella eri tavalla:
  1. Tee yliluokasta ja jokaisesta aliluokasta oma relaationsa. Aliluokasta tehdyn relaation attribuutteina on yliluokan avain ja aliluokan omat attribuutit.
  2. Tee yliluokasta ja jokaisesta aliluokasta oma relaationsa. Aliluokasta tehdyn relaation attribuutteiksi tulee kaikki yliluokan attribuutit ja lisäksi aliluokan omat attribuutit.
  3. Tee koko perintähierarkiasta vain yksi relaatio. Niille attribuuteille, joita tietyllä oliolla ei ole, tulee arvoksi NULL.
- Lisäksi tehdään relaatiot tavalliseen tapaan niistä assosiaatioista, joissa yli- tai aliluokka on osallisena.

Esimerkki 1: aliluokista tehdyillä relaatioilla vain omat attribuutit ja yliluokan avainattribuutit

- Kalvon 18 yliluokasta ja aliluokista muodostetaan neljä eri relaatiota:

  `Products(`<u>`number`</u>`, name, description, price)`
  `CDs(`<u>`number`</u>`, artist, length)`
  `Books(`<u>`number`</u>`, author, pages)`
  `Computers(`<u>`number`</u>`, speed, ram, harddisk)`

- Lisäksi on tarvitaan relaatiot kuvaamaan assosiaatiota *Maintains* ja luokkaa *Firm*:

  `Maintains(`<u>`number`</u>`, `<u>`firmID`</u>`, lengthOfContract)`
  `Firms(`<u>`firmID`</u>`, name, address)`

- Kutakin aliluokan oliota vastaa monikko sekä aliluokkaa kuvaavassa relaatiossa että yliluokkaa vastaavassa relaatiossa.

Esimerkki 2: aliluokista tehdyillä relaatioilla sekä omat attribuutit että kaikki yliluokan attribuutit

- Kalvon 18 aliluokista tehdyillä relaatioilla on nyt enemmän attribuutteja:

  `Products(`<u>`number`</u>`, name, description, price)`
  `CDs(`<u>`number`</u>`, name, description, price, artist, length)`
  `Books(`<u>`number`</u>`, name, description, price, author, pages)`
  `Computers(`<u>`number`</u>`, name, description, price, speed, ram, harddisk)`
- Lisäksi

  `Maintains(`<u>`number`</u>`, `<u>`firmID`</u>`, lengthOfContract)`
  `Firms(`<u>`firmID`</u>`, name, address)`
- Aliluokan tuotetta vastaa nyt monikko vain *yhdessä* relaatiossa. Relaatiossa *Products* on vain ne monikot, jotka eivät kuulu mihinkään aliluokista tehtyihin relaatioihin.
- Jos kaikki monikot kuuluvat johonkin aliluokkaan, ei relaatiota *Products* tarvita lainkaan.

Esimerkki 3: vain yksi relaatio

- Tehdään yliluokasta ja kaikista aliluokista vain yksi yhteinen relaatio.
- Kalvon 18 esimerkkitapauksessa tehtäisiin kaikkia tuotteita edustamaan yksi relaatio

  `Products(`<u>`number`</u>`, name, description, price, artist, length, author, pages, speed, ram, harddisk)`

  Esimerkiksi tuotteilla, jotka eivät ole kirjoja, attribuuttien `author` ja `pages` arvot ovat `NULL`.
- Lisäksi relaatiot

  `Maintains(`<u>`number`</u>`, `<u>`firmID`</u>`, lengthOfContract)`
  `Firms(`<u>`firmID`</u>`, name, address)`

Lähestymistapojen vertailua

- Kyselyissä, jotka koskevat kaikkia tuotteita, on yleensä edullista, jos eri relaatioita vähän.
- Kyselyissä, jotka koskevat vain esimerkiksi kirjoja, on edullisempaa, jos kirjat ovat omassa relaatiossaan.
- Jos kaikki monikot kuuluvat johonkin aliluokkaan, tapa 2 on usein toimivin.
- Tapa 3 on järkevä lähinnä silloin, jos on paljon olioita, jotka kuuluvat samanaikaisesti useaan aliluokkaan (ei aina sallittua), esimerkiksi CD:llä oleva äänikirja.

Aggregaatio ja kompositio

- Aggregaatiota ja kompositota voidaan käsitellä samalla tavalla kuin vastaavia assosiaatioita. Yleensä niistä ei tehdä omaa relaatiota, vaan relaation tiedot liitetään monesta-puolen luokasta tehtyyn relaatioon.
- Esimerkissä

  `Products(number, prodName, description, price, groupID)`
  `ProductGroups(groupID, groupname)`
- Huomautus: kalvojen 17–25 esimerkeissä relaatiolla *Products* ei ole attribuuttia `manufID`, koska esimerkeissä relaatiot on tehty vain vastaavassa UML-kaaviossa näkyvien luokkien pohjalta eikä niissä ole otettu huomioon kaavion ulkopuolisia tietoja.

Funktionaaliset riippuvuudet ja tietokannan normalisointi

- *Ongelma edelleen:* Mitä relaatioita tietokantaan pitäisi määritellä ja mitä attribuutteja näillä pitäisi olla?
- Samat tiedot voidaan esittää useilla eri tietokantakaavioilla. Jotkin niistä ovat parempia kuin toiset.
- Relaatiokaaviot voidaan muuttaa parempaan muotoon normalisoimalla, jossa tarvitaan tietoa relaatioiden attribuuttien funktionaalisista riippuvuuksista.

Esimerkki

- Kaksi eri tapaa esittää tuotteita ja niiden valmistajia.
- 1. Tiedot on tallennettu samaan relaatioon:

```
Products1(number, prodName, description, price, manufID,
          manufName, phone)
```

- 2. Tiedot on jaettu kahteen relaatioon:

```
Products(number, prodName, description, price, manufID)
Manufacturers(ID, manufName, phone)
```

Esimerkki jatkuu

Relaatioiden instanssit voisivat olla esim. seuraavat:

Relaatio Products1

number	*prodName*	*description*	*price*	*manufID*	*manufName*	*phone*
T-33441	Galaxy A5	cellphone	250.0	S123	Samsung	020-7300
S-65221	Brasserie 24	pan	33.50	F542	Fiskars	020-43910
T-33442	NX 300 Smart	camera	399.0	S123	Samsung	020-7300
T-33455	Cyber-shot	camera	463.0	L711	Sony	020-6500
R-43118	Samsung LT 24	TV	169.0	S123	Samsung	020-7300
R-27113	Sony 32 KDL	TV	347.0	L711	Sony	020-6500

Esimerkki jatkuu

Relation Products

number	*prodName*	*description*	*price*	*manufID*
T-33441	Galaxy A5	cellphone	250.0	S123
S-65221	Brasserie 24	pan	33.50	F542
T-33442	NX 300 Smart	camera	399.0	S123
T-33455	Cyber-shot	camera	463.0	L711
R-43118	Samsung LT 24	TV	169.0	S123
R-27113	Sony 32 KDL	TV	347.0	L711

Relation Manufactures

ID	*manufName*	*phone*
S123	Samsung	020-7300
L711	Sony	020-6500
F542	Fiskars	020-43910

Esimerkki: tietokantakaavioiden eroja

- Ensimmäisessä tietokantakaaviossa tiedot valmistajien nimistä ja puhelinnumeroista esiintyvät useaan kertaan, toisessa tietokantakaaviossa vain yhteen kertaan.
- Jos esim. valmistajan puhelinnumeroa muutetaan, pitää päivitys ensimmäisessä kaaviossa tehdä useaan monikkoon, toisessa vain yhteen.
- Jos ensimmäisessä kaaviossa halutaan poistaa tieto tuotteesta S-65221, häviävät samalla kaikki tiedot valmistajasta *Fiskars*. Toisessa kaaviossa valmistajan tiedot säilyvät, vaikka tieto tuotteesta poistetaan.
- Toinen tietokantakaavio on siis selvästi parempi kuin ensimmäinen.

Huonon suunnittelun aiheuttamia ongelmia

- Tietokannan huonosta rakenteesta johtuvia tietokannan käyttäytymisen poikkeavuuksia kutsutaan anomalioiksi (anomaly).
- Keskeiset anomaliat:
  - Tiedon toisteisuus (redundancy)
  - Päivitysanomaliat (update anomalies): jos sama tieto on esitetty useaan kertaan, täytyy muutokset tehdä jokaiseen esityskohtaan.
  - Poistoanomaliat (deletion anomalies): monikoiden poistolla voi olla sivuvaikutuksia. Jos esimerkiksi tuotteiden ja niiden valmistajien tiedot on yhdistetty samaan relaatioon, voi tuotteen poistaminen poistaa myös informaation valmistajan nimestä ja puhelinnumerosta.

Funktionaalisen riippuvuuden määritelmä

- Olkoon relaatiolla R attribuutit $A_1, A_2, \ldots, A_n, B, C_1, C_2, \ldots, C_k$. Attribuutit $A_1, A_2, \ldots, A_n$ määräävät funktionaalisesti attribuutin B, jos seuraava ehto on voimassa:
  - Jos kahdella relaation R monikolla on samat arvot kaikilla attribuuteilla $A_1, A_2, \ldots, A_n$, niin silloin niillä on samat arvot myös attribuutilla B.
- Merkitään

$$A_1\ A_2\ \ldots\ A_n \rightarrow B$$

- R:llä voi kuitenkin olla myös muita attribuutteja, joiden arvot voivat kahdella monikolla poiketa toisistaan, vaikka attribuuteilla $A_1, A_2, \ldots, A_n$ ja B on samat arvot.
- Funktionaalisen riippuvuuden ehdon pitää päteä kaikille *mahdollisille* R:n monikoille, ei vain kaikille tällä hetkellä R:n instanssissa oleville monikoille.
- Funktionaalisia riippuvuuksia tarkastellaan saman relaation eri attribuuttien välillä, ei eri relaatioiden attribuuttien välisiä suhteita.

Funktionaalinen riippuvuus, määritelmä yleisemmin

- Olkoon relaatiolla R attribuutit $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m, C_1, C_2, \ldots, C_k$. Attribuutit $A_1, A_2, \ldots, A_n$ määräävät funktionaalisesti attribuutit $B_1, B_2, \ldots, B_m$, jos seuraava ehto on voimassa:
  - Jos kahdella relaation R monikolla on samat arvot kaikilla attribuuteilla $A_1, A_2, \ldots, A_n$, niin silloin niillä on samat arvot myös kaikilla attribuuteilla $B_1, B_2, \ldots, B_m$.
- Merkitään

$$A_1\ A_2\ \ldots\ A_n \rightarrow B_1\ B_2\ \ldots\ B_m$$

Funktionaalinen riippuvuus, esimerkki

- Olkoon määritelty relaatio

  `Products1(number, prodName, description, price, manufID, manufName, phone)`

- Mitä funktionaalisia riippuvuuksia relaatiolla on?
- Tuotenumero yksilöi tuotteen

  $$\texttt{number} \rightarrow \texttt{prodName description price manufID manufName phone}$$

- Valmistajan tunnus yksilöi valmistajan

  $$\texttt{manufID} \rightarrow \texttt{manufName phone}$$

- Sen sijaan esim. riippuvuus

  $$\texttt{manufID} \rightarrow \texttt{number prodName}$$

  ei ole voimassa, koska samalla valmistajalla voi olla useita eri tuotteita.

Relaatioiden avaimet

- Yksi tai useampi attribuutti $\{A_1, \ldots, A_n\}$ on relaation R avain (key), jos
  1. nämä attribuutit funktionaalisesti määräävät kaikki muut relaation R attribuutit.
  2. mikään tämän attribuuttijoukon aito osajoukko ei funktionaalisesti määrää kaikkia muita relaation R attribuutteja.
- Edellisen kalvon esimerkissä attribuutti `number` on relaation `Products1` avain. Sen sijaan esim. attribuutti `manufID` ei kelpaa tämän relaation avaimeksi, koska se ei funktionaalisesti määrää kaikkia relaation attribuutteja.

Relaatioiden avaimet, jatkoa

- Tarkastellaan relaatiota `Orders1(orderNo, deliver, status, custNo, productNo, count)`. Mikä on tämän relaation avain?
- Attribuutit $\{$`orderNo`, `prodNo`$\}$ muodostavat relaation *Orders1* avaimen, koska
  1. jos kahdella monikolla on samat arvot avainattribuuttien osalta, niin niillä on samat arvot myös lopuille attribuuteille `deliver`, `status`, `custNo` ja `count`
  2. aidot osajoukot $\{$`orderNo`$\}$ ja $\{$`prodNo`$\}$ eivät määrää relaation kaikkia muita attribuutteja.
- Huom: Relaatiolla voi olla myös useampi kuin yksi mahdollinen avain. Tällöin yksi avain valitaan ensisijaiseksi avaimeksi (primary key).
- *Yliavain* (superkey) on attribuuttijoukko, joka sisältää avaimen (mutta voi sen lisäksi sisältää muitakin attribuutteja). Myös avain itse on yliavain.

Funktionaalisten riippuvuuksien päättelystä

- Funktionaaliset riippuvuudet voidaan usein esittää monella eri tavalla.
- Hyvien relaatiokaavioiden etsimisessä on usein olennaista tietää, mitkä riippuvuudet seuraavat toisistaan.
- Olkoon S ja T kaksi funktionaalisten riippuvuuksien joukkoa.
- Riippuvuuksien joukko S *seuraa* (follows) riippuvuuksista T, jos jokainen relaatioinstanssi, joka täyttää kaikki T:n riippuvuudet, täyttää myös kaikki S:n riippuvuudet.
- Riippuvuuksien joukot S ja T ovat *ekvivalentit*, jos S seuraa T:stä ja T seuraa S:stä.

Ositus/yhdistämissääntö

- Funktionaalinen riippuvuus

$$A_1\ A_2\ \ldots A_n\ \rightarrow B_1\ B_2\ \ldots\ B_m$$

on ekvivalentti seuraavan funktionaalisten riippuvuuksien joukon kanssa:

$$\begin{aligned} A_1\ A_2\ \ldots\ A_n &\rightarrow B_1 \\ A_1\ A_2\ \ldots\ A_n &\rightarrow B_2 \\ &\ldots \\ A_1\ A_2\ \ldots\ A_n &\rightarrow B_m \end{aligned}$$

- Ositussääntö (engl. splitting rule): jos funktionaalisen riippuvuuden oikealla puolella on useita attribuutteja, voidaan se jakaa useaksi riippuvuudeksi kuten yllä.
- Yhdistämissääntö (engl. combining rule): jos usealla funktionaalisella riippuvuudella on täsmälleen sama vasen puoli, voidaan ne yhdistää yhdeksi riippuvuudeksi.

Riippuvuuksien luokittelua

- Tarkastellaan riippuvuutta

$$A_1\ A_2\ \ldots\ A_n \rightarrow B_1\ B_2\ \ldots\ B_m$$

  - Riippuvuus on *triviaali* (trivial), jos kaikki B:t ovat A:iden osajoukko
  - Riippuvuus on epätriviaali (nontrivial), jos vähintään yksi B ei sisälly A:iden joukkoon.
  - Riippuvuus on täysin epätriviaali (completely nontrivial), jos mikään B ei sisälly A:iden joukkoon.
- Riippuvuuden oikealta puolelta voidaan aina poistaa sellaiset attribuutit, jotka esiintyvät myös sen vasemmalla puolella.

Transitiivisuussääntö

- Tarkastellaan relaatiota $R(A, B, C, D)$. Jos on voimassa funktionaaliset riippuvuudet $A \rightarrow B$ ja $B \rightarrow C$, niin silloin on voimassa myös riippuvuus $A \rightarrow C$.

Attribuuttien sulkeuma

- Olkoon annettu attribuuttijoukko $\{A_1, A_2, \ldots, A_n\}$ ja joukko funktionaalisia riippuvuuksia S.
- Haluamme tietää kaikki ne attribuutit, jotka jotka riippuvat funktionaalisesti joukosta $\{A_1, A_2, \ldots, A_n\}$ eli joukon $\{A_1, A_2, \ldots, A_n\}$ sulkeumaksi riippuvuusjoukon S suhteen.
- Täsmällisemmin: joukon $\{A_1, A_2, \ldots, A_n\}$ sulkeuma riippuvuusjoukon S suhteen on maksimaalinen attribuuttijoukko X siten että

$$A_1\ A_2\ \ldots\ A_n \rightarrow X$$

- Sulkeumaa merkitään

$$\{A_1, A_2, \ldots, A_n\}^+$$

Attribuuttien sulkeuman laskeminen, algoritmi

- Attribuuttien joukon $\{A_1, A_2, \ldots, A_n\}$ sulkeuman funktionaalisten riippuvuuksien joukon S suhteen voi laskea seuraavasti:
  1. Jaa tarvittaessa funktionaaliset riippuvuudet käyttämällä ositussääntöä niin, että jokaisen riippuvuuden oikealla puolella esiintyy vain yksi attribuutti.
  2. Olkoon X joukko, joka lopulta sisältää sulkeuman. Alusta X:ksi aluksi joukko $\{A_1, A_2, \ldots, A_n\}$.
  3. Etsi jokin funktionaalinen riippuvuus

     $$B_1\ B_2\ \ldots\ B_m \rightarrow C$$

     siten, että kaikki $B_1, B_2, \ldots, B_m$ sisältyvät joukkoon X, mutta C ei sisälly. Lisää C joukkoon X.
  4. Toista kohtaa kolme, kunnes X:ää ei enää voi kasvattaa.
  5. Joukko X on nyt $\{A_1, A_2, \ldots, A_n\}^+$.

Attribuuttien sulkeuman laskeminen, esimerkki

- Tarkastellaan relaatiota $R(A, B, C, D, E, F)$, jolla on riippuvuudet $B\ C \rightarrow A\ D$, $A\ B \rightarrow C$, $D \rightarrow E$ ja $C\ F \rightarrow B$. Mikä on $\{A, B\}^+$?

Miksi laskea attribuuttien sulkeuma?

- Attribuuttien sulkeuman perusteella voidaan päätellä, seuraako jokin riippuvuus $A_1\ A_2\ \ldots\ A_n \rightarrow B$ annetusta funktionaalisten riippuvuksien joukosta S:
  - Lasketaan ensin $\{A_1, A_2, \ldots, A_n\}^+$.
  - Jos B kuuluu sulkeumaan, niin $A_1\ A_2\ \ldots\ A_n \rightarrow B$ seuraa S:stä, jos B ei kuulu sulkeumaan, niin tarkasteltu riippuvuus ei seuraa joukosta S.
- Voidaan päätellä, seuraako jokin riippuvuuksien joukko toisesta riippuvuuksien joukosta, tai ovatko annetut kaksi riippuvuuksien joukkoa ekvivalentteja.
- Voidaan myös päätellä, onko annettu attribuuttijoukko relaation yliavain.