1 TIETOKANNAN SUUNNITTELUPROSESSI

1.1 Tietokannan suunnittelun tavoitteet

Ensisijaisesti tietokannan suunnittelun tavoitteena on mahdollistaa suunniteltavassa järjestelmässä tarvittavat tietojen haku ja käsittelytoimenpiteet. Tämä koskee sekä järjestelmän sisäänrakennettuja, että mahdollisia ”vapaita” (ad hoc) tiedonhaku ja käsittelytoimenpiteitä. Lisäksi hyvän suunnittelun tuloksena mahdollistetaan turvalliset ja määriteltyihin tarpeisiin nähden tehokkaat tavat hakea ja käsitellä tietokannassa olevaa tietoa, tietokannan rakenne ja tietosisältö on aina ehyessä tilassa, tietokanta tukee organisaation toimintaa ja sopeutuu myös tulevaisuuden vaatimuksiin. Lisäksi voidaan kohdealueesta riippuen asettaa myös muita vaatimuksia, kuten yhteensopivuus, siirrettävyys, turvallisuus, jne.

Jotta edellä mainitut tavoitteet toteutuisivat, on suunnittelun tuloksena tultava tietokanta, joka sisältää kaikki tarvittavat tiedot ja niiden väliset suhteet sekä tarvittavat rajoitteet sallituille tiedoille ja suhteille. Tiedon toisteisuutta ja turhan monimutkaisia rakenteita välttämällä edesautetaan järjestelmän päivitettävyyttä ja useimmissa tapauksissa tästä on myös tehokkuusetua.

On myös tärkeää huomata, että tietokannan suunnittelu ei ole muusta ohjelmistotuotannosta irrallinen prosessi, vaan onnistunut tietokannan suunnittelu sisältää järjestelmän tietosisällön määrittelyn ja suunnittelun lisäksi myös järjestelmän toiminnalliset ja ei-toiminnalliset vaatimusten selvittämisen. Nämä eivät kuitenkaan kuulu tämän kurssin sisältöön, vaan kurssin opiskelijoiden oletetaan jo osaavan ne (ohjelmistotuotannon peruskurssi on tämän kurssin esitietovaatimus).

1.2 Hyvän suunnittelutapa

Hyvän suunnittelutavan tai menetelmän merkitystä on vaikea ymmärtää, jos suunnittelusta on vain vähän kokemusta. Hyvä suunnittelumenetelmä antaa tiedot ja taidot, jotka tarvitaan suunnittelu tavoitteiden mukaisen tietokannan suunnitteluun. Se sisältää järjestelmällisen joukon tekniikoita, jotka opastavat suunnittelussa vaihe vaiheelta. Siten hyvän suunnittelumenetelmän oppiminen ja seuraaminen pitää harha-askeleet ja uudelleen suunnittelun tarpeen minimissä, jolloin suunnittelija voi keskittyä olennaiseen virheiden korjaamisen sijasta ja työ on tuottavampaa.

Usein on vaikea nähdä, millaista menetelmää kokeneet suunnittelijat käyttävät missäkin vaiheessa, koska he näyttävät ”oikovan” prosessissa. Useimmiten tämä johtuu siitä, että kokeneiden suunnittelijoiden ei tarvitse tehdä kaikkia suunnittelun askelia eksplisiittisesti (näkyvästi/näkyväksi), vaan he tekevät jotkut askeleet ”päässään” – joskus jopa tietämättään. Tästä huolimatta heillä yleensä on taustalla joku suunnittelumenetelmä tai prosessi, jota he noudattavat. Minä monimutkaisemmasta ja laajemmasta kokonaisuudesta on kyse, sitä tärkeämpi on osata ja käyttää jotain hyvää suunnittelumenetelmää!

1.3 Suunnittelun vaihtoehdot

Periaatteessa tietokannan suunnitteluun on kaksi perustaltaan erilaista lähestymistapaa. Ensimmäinen tapa on suunnitella tietokanta siten, että kerätään kohdealueeseen liittyvät ominaisuudet ja ryhmitellään ne relaatioiksi ”normalisoimalla” tietokantaa. Normalisoinnista puhutaan myöhemmin lisää, mutta kysymyksessä on joukko sääntöjä, joiden perusteella tietokanta saadaan ositettua järkeviksi relaatioiksi. Tämä tapa on siis hyvin teorialähtöinen ja edustaa ns. bottom-up -ajattelutapaa, jossa edetään pienistä yksityiskohdista suurempiin kokonaisuuksiin. Lähestymistapa on käyttökelpoinen lähinnä suhteellisen pienissä tietokannoissa ja suunnittelua tehdäänkin käytännössä vain vähän pelkästään normalisoimalla.

Toinen tapa on ns. top-down -lähestymistapa, jossa ongelmaa lähestytään hahmottamalla ensin suurempia kokonaisuuksia ja niiden välisiä suhteita. Nämä kuvataan tekemällä ongelma-alueesta ensin käsitteellinen malli (puhutaan mallintamisesta). Lähestymistavassa on periaatteena myös, että aluksi tietokannan kuvaus on lopullisesta tietomallista ja alla olevasta tekniikasta riippumaton ja vasta myöhemmissä vaiheissa se sidotaan valittuun tekniikkaan. Mallintaminen on lähestymistapana maalaisjärjellä ymmärrettävämpi kuin puhtaasti normalisointiin perustuva tapa. Se toimii myös hyvin dokumenttina ja helpottaa kommunikointia. Se onkin yleisin tietokannan suunnittelutapa. Normalisointia käytetään tässäkin, mutta vain tietokannan laadun varmistajana.

1.4 Bottom-Up suunnittelu lyhyesti

Huolimatta siitä, millä tavalla tietokanta suunnitellaan, tärkeitä tietokannan suunnitteluun liittyviä kysymyksiä ovat mm:

- Mitä tietoja halutaan tallentaa ja missä muodossa?
  - millaisia tietoja – arvoalueet
  - millaisia rajoitteita
- Mitä tietoja halutaan käsitellä, millä perusteella ja missä muodossa?
  - Millaisia tiedonkäsittelytapahtumia?
- Kenen sallitaan pääsevän käsiksi mihinkin tietoihin?

- Minkälainen järjestelmä on kyseessä?
  - Käyttötarkoitus ja käyttötapa
  - käsiteltävän tiedon luonne
    - tiedonhaku, päivitykset, tiedonkeruu
  - järjestelmän arkkitehtuuri

Koska bottom-up lähestymistapaa käytetään pääsääntöisesti suhteellisen pienissä ja yksinkertaisissa tietokannoissa, suunnittelija tuntee usein kohdealueen ja tehtävän järjestelmän (esim. henkilökohtainen osoiterekisteri, tms.) hyvin ja osaa siten vastata yllä oleviin kysymyksiin suvereenisti. Joka tapauksessa tätä suunnittelutapaa käytettäessä tarvittavat tiedot saadaan yleensä jokseenkin informaalisti haastattelujen, olemassa olevien raporttien, ym. perusteella.

Suunnittelutapa on kevyt, mutta siinä keskitytään lähinnä siihen, millainen toteutettavan tietokannan rakenne on. Suunnittelutavassa toteutettavan tietomallin oletetaan siis yleensä olevan tiedossa jo suunnittelun alkuvaiheessa. Siis jos oletetaan, että ollaan suunnittelemassa relaatiotietokantaa, pohditaan, miten tietokanta jaetaan relaatioiksi, mitkä ovat relaatioiden pääavaimet ja mitkä ovat vierasavaimet.

Yksi tapa alkaa jäsentämään tietokantaan tallennettavaa tietoa relaatioiksi on ajatella erilaisia faktoja ja väittämiä, ja ryhmitellä tiedot näiden mukaan alustavasti relaatioiksi (myös muita ryhmittelyperusteita voi käyttää – mikä vain tuntuu luontevalta). Relaatiotietokantaan tallennettavat tiedot voidaan ajatella TOSINA väittäminä (tai logiikassa propositioita) kiinnostuksen kohteista. Alla olevassa esimerkkitapauksessa kiinnostuksen kohteena ovat opiskelijat:

*Opiskelijan opiskelijanumero on **12345**, nimi on **Johan Kesti**, pääaine on **Hiusfysiikka** ja opintopisteitä on **13**.*

*Opiskelijan opiskelijanumero on **23456**, nimi on **Johanna Kesti**, pääaine on **Vaahtomuotoilu** ja opintopisteitä on **123**.*

Vaihtamalla arvojen paikalle ”muuttujien” (tai paremminkin ominaisuuksien) nimiä, saadaan (logiikassa) predikaatteja:

*Opiskelijan opiskelijanumero on **(OPNUM)**, nimi on **(Etunimi, Sukunimi)**, pääaine on **(PAine)** ja opintopisteitä on **(OPisteet)**.*

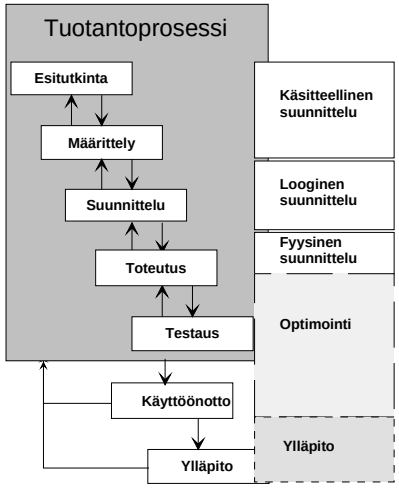
Yllä olevan mukainen predikaatti sisältää selvästikin monia opiskelijaan liittyviä tietoja ja siten voitaisiin päätyä alustavasti relaatioon Opiskelija(@OPNUM, Etunimi, Sukunimi, Paine, OPisteet), jossa opnum yksilöi kunkin opiskelijan, eli on relaation pääavain.

Antamalla muuttujille arvoja niille määritellyiltä arvoalueilta (esimerkiksi {12345, Johan Kesti, Hiusfysiikka, 13}), eli tallentamalla tauluun tietoja, saadaan jälleen tosia väittämiä.

Kun tiedot on ryhmitelty alustaviksi relaatioiksi (pää- ja vierasavaimineen), käytetään normalisointiteoriaa (tästä puhutaan myöhemmin lisää) ja lopputuloksena saadaan lopulliset relaatiot.

1.5 Kurssilla käytettävä Top-Down suunnitteluprosessi

Kuvassa 1.1 on esitetty yksi (tällä kurssilla käyetttävä) tietokannan suunnitteluprosessi korkealla tasolla. Kuvassa tietokannan suunnitteluun liittyvä osuus ohjelmistotuotannosta on otettu korostetusti esiin ja pyritty siten kuvaamaan sitä, mihin ”normaalin” ohjelmistotuotantoprosessin vaiheisiin mitkäkin tietokannan suunnittelun vaiheet liittyvät. Kuvassa käytetty vesiputousmalli on teoreettinen malli ja käytännössä ohjelmistotuotantoprosessi eroaa aina tästä mallista. Vesiputousmalli sisältää kuitenkin kaikki oleelliset ohjelmistokehityksen vaiheet ja vaikka niiden suhteet muuttuisivatkin muunlaisissa elinkaarimalleissa, tietokannan suunnittelun vaiheet liittyvät aina vastaaviin kohtiin muussa ohjelmistotuotannossa.



Kuva 1.1 *Tietokannan suunnitteluprosessi vesiputousmallin avulla esitettynä.*

Seuraavaksi käydään lyhyesti läpi suunnitteluprosessin eri vaiheet ja niiden olennaisimmat tulokset ja tulosten laatuvaatimukset.

Vaatimusten keruu ja analysointi
Ennen kuin mitään järjestelmää voidaan alkaa järkevästi suunnittelemaan, pitää toteutettava järjestelmä edes jollakin tavalla rajata sekä kerätä sille asetettavat alustavat vaatimukset ja analysoida ne. Tätä varten projekteissa on usein erillinen esitutkimusvaihe, jossa arvioidaan usein myös nykytilannetta sekä projektin eri toteuttamisvaihtoehtoja ja toteuttamiskelpoisuutta. Vaatimusten keruussa ja rajauksessa käytetään tekniikkana esimerkiksi haastatteluja. Vaatimusten keruu voidaan tehdä myös osajärjestelmittäin, esimerkiksi osastokohtaisesti (myynti ja tuotanto erikseen).

tulos: alustavat vaatimukset
laatu: vaatimusten oikeellisuus ja täsmällisyys

Käsitteellinen suunnittelu
Tässä vaiheessa on ideana luoda TKHJ:stä riippumaton esitys järjestelmän tietosisällöstä. Käsitteellisen mallin rakentaminen voidaan aloittaa useimmiten (korkean tason hahmotelmilla) jo esitutkimuksen yhteydessä ja se tarkentuu määrittelyvaiheessa. Tietosisällön määrittelyn lisäksi olennainen osa käsitteellistä suunnittelua on myös järjestelmän tietokantatapahtumien määrittely, joka kuvataan järjestelmän toiminnallisuutena (tietosisällöstä erillään).

Jos eri osajärjestelmille on kerätty vaatimukset erikseen, niille voidaan kullekin luoda oma käsitteellinen malli, jolloin kukin näistä malleista edustaa osajärjestelmään liittyvää näkymää koko tietokantaan. Tällaisesta osittamisesta on hyötyä ainakin hyvin suurissa järjestelmissä. Vaarana tässä tavassa on, että osajärjestelmien ristiriitaisuudet huomataan myöhemmässä vaiheessa kuin jos tehtäisiin alusta asti vain yhtä mallia (ja vaatimuksia). Joka tapauksessa mahdolliset erilliset mallit on yhdistettävä yhdeksi viimeistään tämän vaiheen lopussa.

tulos: käsitekaavio, TKHJ:stä riippumaton esitys tietosisällöstä
laatu: oikeellisuus, luettavuus

TKHJ:n valinta
Tietokannan hallintajärjestelmän valintaa käsitellään kurssin lopussa mikäli siihen jää aikaa.

Looginen suunnittelu
Viimeistään tässä vaiheessa valitaan tietomalli, jonka mukaista tietokantaa ollaan tekemässä. Edellisen vaiheen TKHJ riippumattoman kaavion perusteella tehdään tietosisällöstä sitten ko. tietomallin mukainen, mutta ei välttämättä mihinkään tiettyyn TKHJ:ään (tuotteeseen) tai muihin fyysisiin rajoitteisiin sidottu looginen kuvaus tietokannasta. Mikäli valittu tietomalli on relaatiomalli, normalisoidaan tietokanta tarvittaessa (normaalius tarkistetaan joka tapauksessa).

tulos: Tietomallista-riippuvainen looginen kaavio
laatu: käännöksen oikeellisuus, tehokkuus suhteessa tapahtumiin

Fyysinen suunnittelu
Viimeistään tässä vaiheessa sidotaan käytettävä tietokannan hallintajärjestelmä ja ympäristö. Edellisen vaiheen kuvauksen perusteella toteutetaan tietokanta valitulle TKHJ:lle, eli esimerkiksi relaatiotietokannan yhteydessä luodaan tarvittavat perusrelaatiot, hakemistot, näkymät jne.

tulos: TKHJ-riippuvainen fyysinen kaavio (toteutus)
laatu: tehokkuus

Tällä kurssilla ei käsitellä fyysisen suunnittelun jälkeisiä kohtia. Käsitteellistä suunnittelua lukuun ottamatta seuraavissa kohdissa oletetaan, että ollaan suunnittelemassa relaatiotietokantaa.

2 KÄSITTEELLINEN SUUNNITTELU

Tässä vaiheessa luodaan siis TKHJ:stä riippumaton käsitteellinen malli, määritellään tietojen arvoalueet ja varmistetaan, että kuvatulla tietosisällöllä voidaan tehdä järjestelmässä vaadittu käsittely ja vastata vaadittuihin kyselyihin (verrataan siis määriteltyihin tapahtumiin). Tällä kurssilla lähdetään tästä lähtien siitä, että on olemassa jonkinlainen kuvaus (vähintään alustavat vaatimukset) suunniteltavasta järjestelmästä.

2.1 Käsitteellinen mallintaminen

Käsitteellinen mallintaminen on käyty joiltakin osin (lähinnä luokkakaavion laadinta) läpi myös ohjelmistotuotannon peruskurssissa, joten luvassa on osittain asioiden kertausta. Kyseessä on kuitenkin tämän kurssin kannalta niin tärkeä asia, että mallintaminen käydään kokonaisuudessaan läpi myös tällä kurssilla.

Kun tarkkailemme maailmaa, nimeämme asioita, siis muodostamme käsitteitä, jotta voimme keskustella niistä. Kun pyrimme erottamaan käsitteitä toisistaan, huomaamme, että niillä on erilaisia ominaisuuksia (”wau – punainen tukka”) ja keskinäisiä suhteita (”toi on varmaan mun äiti”). Voimme esimerkiksi huomata, että jokin käsite koostuu joistakin toisista käsitteistä (”hei - tikkarissa on tikku”). Samoin voimme havaita, että jollain joukolla käsitteitä on samanlaisia ominaisuuksia, jolloin usein nimeämme tämän ryhmän – siis luokittelemme käsitteitä (kissa, koira, ihminen, dinosaurus,…). Käytämme näitä mekanismeja jatkuvasti kun ymmärryksemme ympäröivästä maailmasta kehittyy. Tämä heijastuu mm. siten, että myös käyttämämme luonnollinen kieli kehittyy koko ajan.

Aivan samoin käsitteellisessä mallintamisessa pyritään kuvaamaan maailmaa (kohdealuetta) käsitteiden sekä niiden ominaisuuksien ja keskinäisten suhteiden avulla. Pyritään siis kuvaamaan se, mitä kohdealueesta tiedetään. Kohdealueen tila voi muuttua ajoittain erilaisten tapahtumien vaikutuksesta. Mallintamisessa ollaan siis kiinnostuneita kohdealueeseen liittyvien tietojen lisäksi kohdealueen mahdollisista tiloista menneisyydessä, nyt ja tulevaisuudessa. Menneisyydestä ollaan kiinnostuneita mm. koska vertaamalla mennyttä ja nykyistä tilannetta, voidaan yrittää päätellä potentiaalisia tulevia muutoskohtia. Yleisemmin sanottuna, tarkoituksena on kuvata kaikki kohdealuetta kuvaavat tiedot, säännöt ja rajoitteet sekä kohdealueeseen kohdistuvat tapahtumat.

Käsitteellistä mallintamista käytetään menetelmänä tietojärjestelmien kehittämisen lisäksi mm. yrityksen toiminnan kehittämisessä (esim. liiketoimintaprosessien suunnittelu / uudelleensuunnittelu), CIM järjestelmien suunnittelussa (Computer integrated manufacturing) sekä erilaisten tuotteiden ja tuoterakenteiden kuvauksessa. Tietojärjestelmien kehittämisen yhteydessä käsitteellinen malli toimii mm. järjestelmän tietosisällön kuvauksena, dokumentointina, ja kommunikoinnin apuvälineenä.

Mallintamisessa eri käyttötarkoitukset ja käyttötilanteet vaativat erilaista tarkkuustasoa aina epäformaalista hahmottelusta täsmälliseen määrittelyyn. Joka tapauksessa tavoitteena on saada kuvattua ”yhteinen maailmankuva”, so. kohdealueella käytettävä yhteinen kieli ja yhteisesti hyväksytyt säännöt. Tätä kieltä voidaan käyttää kohdealueesta keskusteluun sekä päätelmien tekemiseen kohdealueesta. Siten esimerkiksi tietokannan toteuttajilla pitäisi olla sama ymmärrys kohdealueesta kuin esimerkiksi asiakkailla. (Määrittely on sopimus ja kaikkien osapuolien pitäisi ymmärtää sopimus samalla tavalla.)

nice to know:
Semiotiikka on tieteenala, joka tutkii toimijoiden välistä kommunikaatiota. Sen perusosia ovat syntaksi, semantiikka ja pragmatiikka. Syntaksi kertoo kuinka kielen perusosista (merkit, sanat,…) saadaan rakennettua suurempia kokonaisuuksia (merkkijonoja, lauseita…). Semantiikka tutkii kielen merkitystä – siis kielellisten ilmaisujen ja todellisuuden välistä yhteyttä. Semantiikassa tämän kurssin kannalta merkittävin periaate on nk. koostettavuuden periaate, jonka perusteella minkä tahansa kokonaisuuden merkitys saadaan koostamalla sen osien merkityksistä (mieti, miten tämä liittyy tähän kurssiin).

Käsitteellinen mallintaminen antaa siis selkeät määritelmät kohdealueen käsitteille, niiden ominaisuuksille ja suhteille sekä selkeän määrittelyn järjestelmän dynamiikalle. Dokumentoitu käsitteellinen malli auttaa havaitsemaan kohdat, joista ollaan eri mieltä ja siten päätymään yhteiseen näkemykseen kohdealueesta.

2.2 Mallintamistapoja

Mallintamiseen on olemassa lukuisia erilaisia tapoja ja tekniikoita. Niissä käytetään kuitenkin aina jotain kuvauskieltä, joka voi olla esimerkiksi luonnollinen kieli, joku formaali kieli (1. kertaluvun logiikka, OCL (object constraint language), …), tai jokin graafinen kieli. Graafisista kielistä ER-kaavioiden (Entity-Relationship – kohde-suhde) johdannaiset, kuten Chenin notaatio sekä OMT:n ja UML:n luokkakaavionotaatiot, ovat eniten käytettyjä.

ER-mallinnus (Entity-Relationship modelling) on graafinen, rakenteellinen mallinnustapa, jossa käsitteet ja niiden väliset suhteet esitetään erilaisilla laatikoilla ja niiden välisillä viivoilla. Oliomallinnus ja luokkakaaviot (OMT, UML) ovat kuvaustekniikkana ER-kaavion johdannaisia ja käytännössä melko samanlaisia kuin oliolaajennoksilla varustetut ER-kaaviot. Varsinkin UML:n luokkakaavioita käytetään myös muualla kuin käsitteellisen mallintamisen (tai tietokantojen) yhteydessä.

Perinteisesti tietokantojen kuvauksessa on käytetty chenin ER-kaavio notaatiota (tai sen laajennettua versiota), mutta nykyisin UML on saavuttamassa kuvauskielenä standardin aseman ja sitä on alettu käyttää laajamittaisesti myös tietokantojen yhteydessä. Tästä syystä tällä kurssilla käsitekaavioiden yhteydessä käsiteltäväksi kuvaustekniikaksi on valittu UML:n luokkakaaviot.

Käytännössä graafisilla kuvausmenetelmillä ei joko pysty, tai ei ole järkevää kuvata kaikkia kohdealueella olevia monimutkaisia rajoitteita. Tästä syystä graafisia kuvaustapoja täydennetään usein muilla, yleensä tekstuaalisilla kuvaustavoilla, esim. luonnollisella kielellä tai jollain rajoitekielellä (kuten OCL, object cosntraint language. UML:n tekstuaalinen rajoitekieli).

2.3 Käsitekaavio

Graafista kuvauskieltä käytettäessä mallinnuksen tärkein tulos on käsitekaavio, joka toimii mm. järjestelmän tietosisällön kuvauksena, dokumentointina ja kommunikoinnin apuvälineenä. Käsitekaavion on tarkoitus olla todellisuuden (kohdealueen) vääristymätön ja täydellinen esitys, jossa toteutusnäkökohdat jätetään huomioimatta. Kannattaa myös huomata, että aika on tärkeä reaalimaailman komponentti.

Erityisesti oliokeskeisissä mallinnuskielissä mallinnettava reaalimaailma nähdään olioina, jotka ryhmitellään **luokiksi**. On tärkeää muistaa, että luokkakaavion käyttökohde vaikuttaa kaavion tulkintaan. Käsitteellisessä mallintamisessa luokkakaavion luokat kuvaavat kohdealueella esiintyviä käsitteitä. Siten kullakin luokalla, siis käsitteellä, on kuvattavalla kohdealueella yksilöllinen nimi. Samoin ne reaalimaailman käsitteet, jotka kuuluvat (luokitellaan) samaan luokkaan pitää olla samanlaiset, eli niillä täytyy olla samat ominaisuudet. Esimerkiksi oliot “pankin-asiakkaat” ja “pankin-tilit” eivät saisi kuulua samaan luokkaan, koska niillä ei ole mitään yhteistä. Käsitekaavioissa esiintyy luokkia (ei yleensä olioita).

Ominaisuudet kuvaavat luokan tietosisältöä ja ne ovat siten aina jonkun luokan osana (ei koskaan itsenäisenä). Ominaisuuksien nimet ovat yksilöllisiä kunkin luokan sisällä. Käytännössä luokan ominaisuudet kertovat (rajaavat) millaisia ominaisuuksia luokan instansseilla voi tai pitää olla. Osa ominaisuuksista voi siis olla pakollisia ja osa vaihtoehtoisia. Tätä (pakollisuutta/vaihtoehtoisuutta) ei kuitenkaan yleensä merkitä käsitekaavioon (luettavuus), vaan erilliseen tietohakemistoon.

Luokan instanssissa annetaan ominaisuuksille jotkut arvot. Ominaisuudet taas saavat arvoja joltain arvoalueelta, (johon voi myös liittyä mielivaltaisia rajoitteita). Myöskään näitä arvoalueita ei yleensä merkitä luettavuussyistä käsitekaavioon.

Osa luokan ominaisuuksista voi olla pääteltävissä muiden ominaisuuksien perusteella. Tällainen ominaisuus kannattaa merkitä jotenkin (käytetään tässä kurssilla vaikka / -merkkiä, toinen tapa olisi merkitä tällainen ominaisuus metodina, koska kysymyksessä on yleensä jonkinlainen laskutoimitus tai koostamisfunktio).

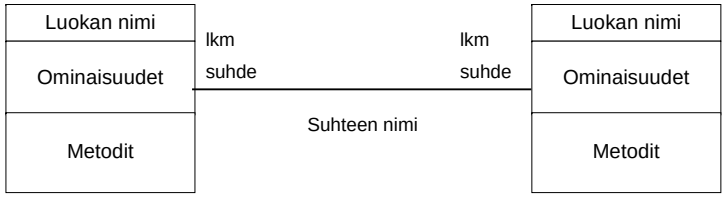
Vaikka oliot (käsitteiden instanssit) ajatellaankin automaattisesti yksilöllisiksi, on myöhempien suunnitteluvaiheiden kannalta erittäin hyödyllistä tietää, jos jotkut ominaisuudet yksilöivät olion. Osa luokan ominaisuuksista voidaan siis merkitä yksittäisen olion yksilöiviksi ”avainominaisuuksiksi” (kyseessä ei ole automaattisesti sama asia kuin relaation pääavain – sen sijaan avainkandidaattiin rinnastaminen ei ole aivan väärin). Käytetään avainominaisuuksien merkitsemiseen tällä kurssilla vaikka @ -merkkiä. Tietokantoja mallinnettaessa oliomalleissa ei kuitenkaan koskaan sisällytetä ominaisuuksiin keinotekoisia avaimia (surrogaatteja, sisäisiä tunnisteita, joilla ei ole merkitystä ongelma-alueella) koska käsitteellisessä mallintamisessa pyritään kuvaamaan reaalimaailmaa.

Luokkien välisiä riippuvuuksia kuvataan **suhteilla**. Suhde ei ole koskaan itsenäinen, vaan se liittyy yhteen tai useampaan luokkaan. Suhteita tarkennetaan lukumääräsuhteilla (kardinaliteeteilla). Huomaa, että lukumääräsuhteet ovat itse asiassa rajoitteita järjestelmään tallennettavalle tietosisällölle. Ne rajaavat (mahdollisia tai pakollisia) luokkien olioiden välisiä suhteita.

Käsitteiden, suhteiden ja ominaisuuksien lisäksi kaikkiin mallinnettaviin elementteihin voi liittyä mielivaltaisia **rajoitteita** (kohdealueen sääntöjä, ”liiketoimintasääntöjä”). Osa näistä rajoitteista merkitään yleensä käsitekaavioon (rakenteelliset rajoitteet), mutta monimutkaisemmat säännöt dokumentoidaan luettavuuden vuoksi tietohakemistoon.

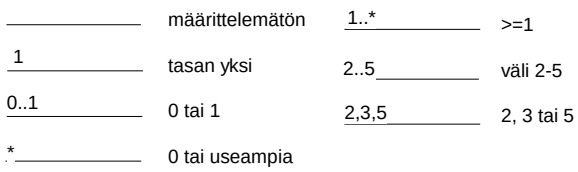
2.4 UML:n luokkakaavionotaatio

Käydään UML:n luokkakaavionotaation piirteet seuraavaksi läpi pienten esimerkkien avulla. Graafisesti luokka piirretään kuten kuvassa 1.2 on esitetty (kuvaustekniikkana UML), eli luokan symboli on laatikko, jossa on yksi tai useampia vaakaviivoin erotettua lohkoa (compartment). Ylimmässä lohkossa on *luokan nimi*, keskimmäisessä *ominaisuudet* ja alimmassa luokan *metodit*. Metodit (koko lohko) voidaan jättää käsitteellisessä mallintamisessa ja tietosisältöä kuvattaessa yleensä pois.



Kuva 1.2 Luokan graafinen esitysmuoto.

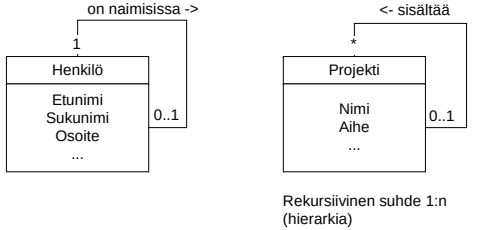
Suhteet piirretään viivoina luokkien välille ja lukumääräsuhteet merkitään (kaikkiin) suhteen päihin. Mikäli johonkin päähän ei ole määritelty lukumääräsuhdetta, se on määrittelemätön. Mikäli lukumääräsuhteen alaraja on 0, suhde on vaihtoehtoinen, mikäli se on suurempi kuin 0, suhde on pakollinen. Lukumääräsuhteista on esimerkkejä kuvassa 1.3. Suhteelle annetaan myös yleensä aina nimi ja mahdollisesti myös lukusuunta. Nimi on informatiivinen ja se auttaa kaavion tulkinnassa.



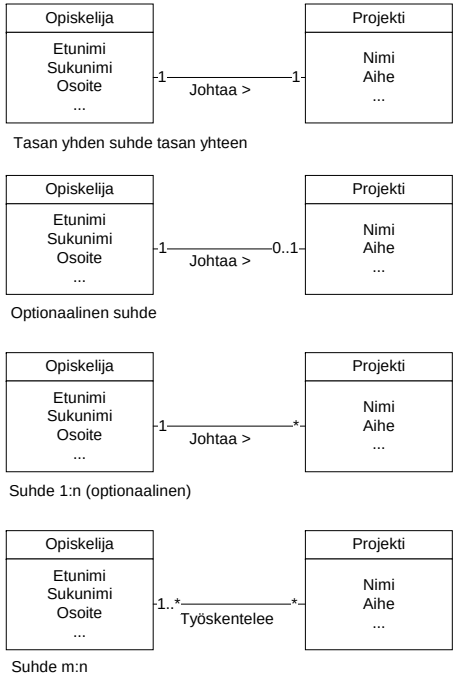
Kuva 1.3 Esimerkkejä lukumääräsuhteista.

Käsitteellisessä mallintamisessa kaikki suhteet ovat yleensä kaksi- (tai N-) suuntaisia, mutta tarvittaessa on mahdollista merkitä myös suhteen suunta (Huom! eri asia kuin lukusuunta) laittamalla suhteen päihin nuolet.

Kuvassa 1.4 on esitetty kaksi esimerkkiä **unaarisista suhteista**, eli suhteista, joihin osallistuu vain yksi luokka. Kuvassa 1.5 taas on esimerkkejä **binaarisista suhteista**, eli suhteista, joihin osallistuu kaksi luokkaa.

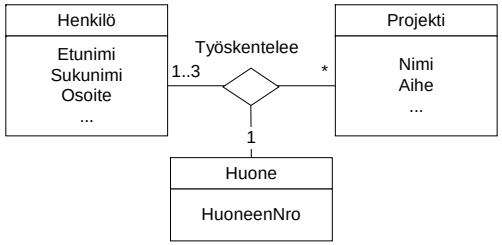


Kuva 1.4 Esimerkkejä unaarisuhteista.

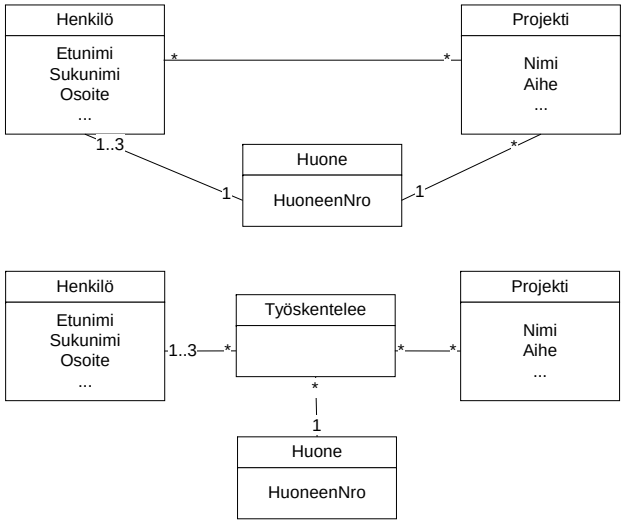


Kuva 1.5 Esimerkkejä binaarisuhteista.

Kuvassa 1.6 on esimerkki **kolmen luokan välisestä suhteesta**. Huomaa, että kolmen tai useamman luokan välinen suhde on merkitykseltään erilainen kuin samojen luokkien väliset kahden väliset suhteet (kuten esim. kuvassa 1.7).

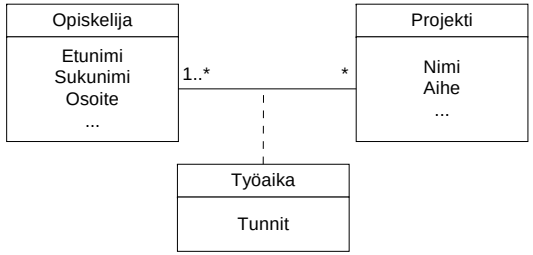


Kuva 1.6 Esimerkki kolmen luokan välisestä suhteesta.



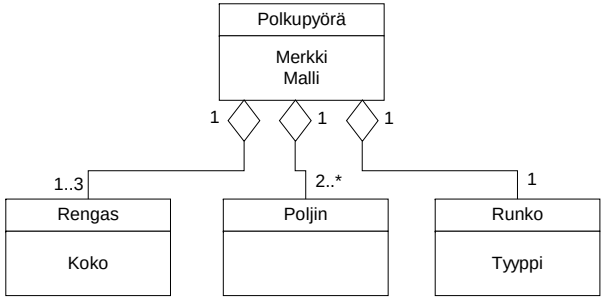
Kuva 1.7 Kuvan 1.6 suhde on eri asia kuin esim. tässä kuvatut suhteet.

Suhteeseen voi myös liittyä tietoja, kuten kuvassa 1.8. UML:ssa suhteeseen liittyvät tiedot ovat aina nk. **assosiaatioluokassa** (suhteeseen liittyvä luokka), joka on yhtä aikaa luokka ja suhde (sillä on samat ominaisuudet kuin suhteella ja luokalla). Sillä on esimerkiksi nimi ja ominaisuuksia ja siihen voi liittyä suhteita aivan samoin kuin luokkaankin.



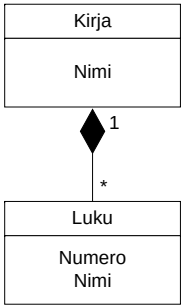
Kuva 1.8 Esimerkki suhteeseen liittyvästä luokasta.

UML:ssa on kaksi erilaista tapaa esittää käsitteen koostuminen joistain toisista käsitteistä. Ne myös tarkoittavat hieman eri asioita, joten niistä käytetään eri termejä. Käytetään termiä **koostuminen** (aggregation) sellaisessa tapauksessa, jossa sisältyvä osa on itsenäinen käsite, eikä sen elinkaarta ole sidottu sen sisältävän käsitteen elinkaareen. Koostuminen merkitään valkoisella salmiakilla, kuten kuvassa 1.9.



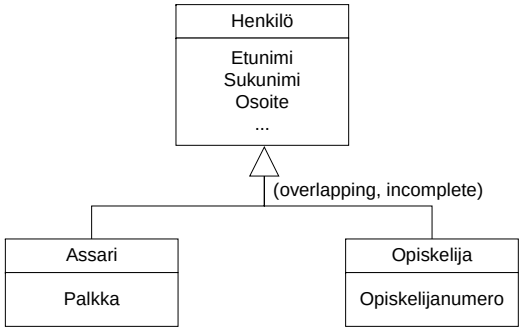
Kuva 1.9 Esimerkki koostumisesta.

Käytetään termiä **sisältyminen** sellaisissa tapauksissa, joissa koostuvan osan elinkaari on sidottu koostavan käsitteen elinkaareen. Esimerkiksi kirjan luvut sisältyvät kirjaan. Jos kirja tuhotaan, tuhoutuvat myös kirjan luvut. Toinen tulkinta on, että sisällytettävä luokka (kirjan luku) yksilöidään kohdealueella sisältyvän luokan (luku) perusteella. Sisältyminen merkitään täytetyllä (mustalla) salmiakilla, kuten kuvassa 1.10.



Kuva 1.10 Esimerkki sisältymisestä.

Periytymisellä tarkoitetaan käsitteen kaikkien piirteiden, siis ominaisuuksien, suhteiden ja menetelmien ”kopioitumista” siitä perityille luokille. Periytyminen esitetään UML:ssa kolmiolla (kuten kuvassa 1.11).



Kuva 1.11 Esimerkki periytymisestä.

Periytyvä luokka voi olla joko konkreettinen (samoin kuin muuta luokat) tai abstrakti. Abstrakti luokka on lähinnä ryhmittelyperuste, jota käytetään käsitteiden kategorisointiin tai helpottamaan kaavion tulkintaa. Abstrakti luokka merkitään näyttämällä luokan nimi kursiivilla, tai liittämällä luokkaan stereotyyppi <<Abstract>>.

Periytymiseen ei liity lukumääräsuhteita, mutta periytymiselle voidaan antaa tiettyjä rajoitteita kirjoittamalla joku tai jotkut seuraavista avainsanoista aaltosulkeissa periytymisen kohdalle.

Overlapping
Käsitteen (luokan) ilmentymä voi olla yhtä aikaa yhtä tai useampia kantaluokasta perittyjä käsitteitä. Esimerkiksi kuvassa 1.11 henkilö voi olla yhtä aikaa sekä assistentti että opiskelija.

Disjoint
Overlappingin vastakohta. Käsitteen ilmentymä voi olla yhtä aikaa vain yhtä kantaluokasta perityistä käsitteistä.

Complete
Kaikki kantaluokasta (kuvattavalla kohdealueella) periytyvät käsitteet on kuvattu tässä mallissa.

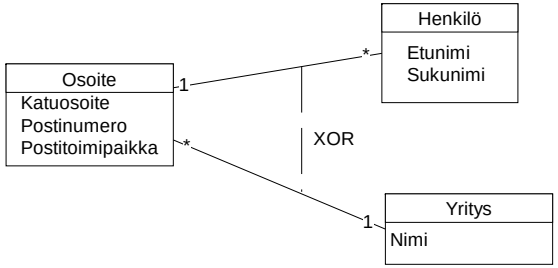
Incomplete
Kaikkia kantaluokasta periytyviä käsitteitä ei ole kuvattu (niitä ei tiedetä tai haluta kuvata tässä yhteydessä) tässä mallissa.

Aikaisemmin todettiin, että lukumääräsuhteet ovat itse asiassa rajoitteita järjestelmään tallennettavalle tietosisällölle, periytymistä voidaan rajoittaa (esim. distinct) ja jopa erilaiset suhteet (esim. koosteet) ja käsitteetkin rajaavat mallia. Koska käsitteellisessä mallintamisessa ei kuvata varsinaisesti tehtävää järjestelmää, vaan jotain tiettyä kohdealuetta, niin oikeastaan kaikki rajoitteet ovat kohdealueella (organisaatiossa, liiketoiminnassa, jne.) vallitsevia sääntöjä. Malliin kohdistuvia rajoitteita sanotaankin käsitteellisessä mallintamisessa yleisesti **liiketoimintasäännöiksi** (business rules).

UML:ssa on aikaisemmin mainittujen rajoitteiden lisäksi myös ihan erillinen rajoitteen käsite. **Rajoitteita** käytetään kuvaamaan sääntöjä, joita ei voida kuvaustekniikalla muuten mallintaa. Ne merkitään aalto-sulkujen sisään, esimerkiksi *{rajoite}* ja ne voi kuvata joko tavallisena tekstinä tai UML:n rajoitekielellä **OCL:lla** (object constraint language). Rajoite voi olla periaatteessa mitä tahansa ja kohdistua mihin tahansa mallin osiin. Rajoite kohdistetaan haluttuihin mallin osiin katkoviivalla. Pelkästään suhteisiin kohdistuvat rajoitteet merkitään piirtämällä katkoviiva ”suhteiden lävitse”. Muissa

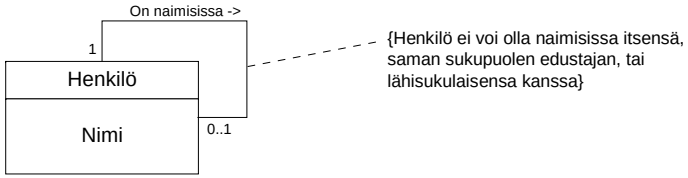
tapauksissa katkoviiva piirretään ko. rajoitteesta haluttuihin mallin osiin.

Yksi yleisesti suhteille annettavista rajoitteista on **XOR**, josta on esimerkki kuvassa 1.13. XOR rajaa kuvassa suhteita siten, että niistä vain yksi voi olla kerralla voimassa. Siis kuvassa 1.13 osoite on joko henkilön tai yrityksen osoite.



Kuva 1.13 Esimerkki XORin käytöstä.

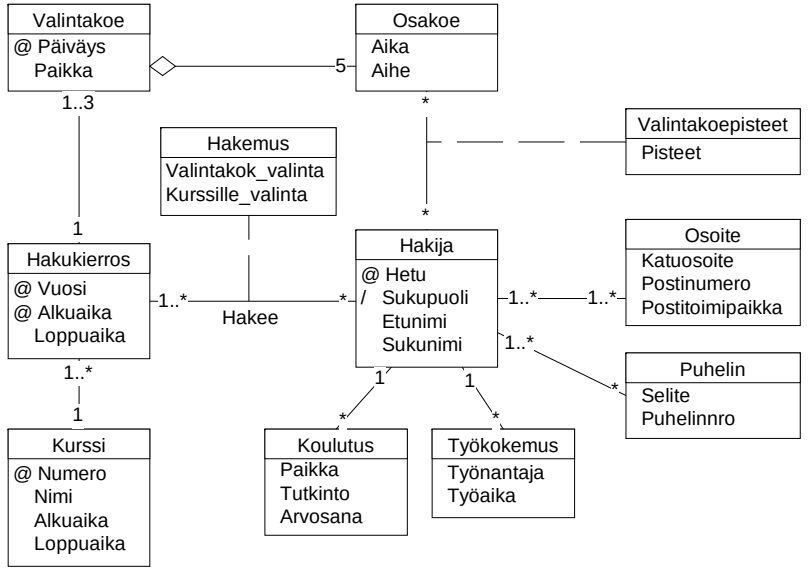
Kuvassa 3.14 on esimerkki toisenlaisesta rajoitteesta. Usein termillä liiketoimintasääntö viitataan juuri tämän tyyppisiin, ”vapaisiin”, rajoitteisiin.



Kuva 1.14 Esimerkki liiketoimintasäännöstä.

2.5 Esimerkki käsitekaaviosta

Kuvassa 1.12 on esitettynä esimerkkinä yksinkertainen oppilaitoksen oppilasvalintajärjestelmän käsitekaavio. Tulkitse kaavio.



Kuva 1.12 Esimerkki käsitekaaviosta.

2.6 Käsitteellinen mallintaminen käytännössä

Hyvä suunnittelija osaa kuvata ja ratkaista ongelmia, vaikka informaatio olisi epätäydellistä

Käsitteellinen mallintaminen ei ole ensisijaisesti ongelmanratkaisutehtävä. Keskeistä on kohdealueen ymmärtäminen ja kuvaus (dokumentointi, yhteisen kielen sopiminen, …) ja siten ongelmanratkaisuosa liittyykin ensisijaisesti siihen, miten (oikeat ja luotettavat) tiedot saadaan ja miten ne kuvataan (oikein). Haastattelut, workshopit, alustavat vaatimukset, ym. toimivat pohjana ymmärtämiselle. Kuvauksen jälkeen ymmärrys tarkentuu ja sitä voidaan validoida samoja keinoja käyttäen. Käytännössä kohdealueen analysointi on käsitteiden ja niiden välisten suhteiden etsimistä ja luokittelua sekä mm. mahdollisten kohdealueeseen kohdistuvien muutosten arviointia.tulevien

Top Down lähestymistavassa etsitään ensin t*ärkeimpiä käsitteitä (tai suurempia kokonaisuuksia) ja niiden välisiä suhteita. Näistä edetään pienempiin ja vähemmän merkittäviin kokonaisuuksiin.* Käytännössä työ alkaa nopealla hahmottelulla – etsitään aktiivisesti eri variaatioita ja keskustellaan kohdealueesta, jolloin myös ymmärrys kohdealueesta lisääntyy. Käytännössä työ on iteratiivista ja inkrementaalista.

Käsitekaavion laadinta aloitetaan yleensä siten, että pyritään ensin tunnistamaan järjestelmässä esiintyviä käsitteitä. Tässä ei kannata kursailla. Turhat käsitteet voi karsia myöhemmin pois. Tämän jälkeen haetaan käsitteiden välillä esiintyviä suhteita ja piirretään ensimmäinen versio kaaviosta, jonka jälkeen voidaan alkaa lisäämään käsitteille ominaisuuksia. Käytännössä kaavion laadinta on iterointia. Käsitteitä, suhteita ja ominaisuuksia lisätään, poistetaan ja muutellaan niin kauan, että kaavio vastaa kuvattavan kohdealueen kaikki piirteet selkeästi ja johdonmukaisesti. Kokemuksen myötä kehittyy myös näkemys siitä, mitä kannattaa mallintaa mitenkin.

Käsitteiden ”löytäminen” on luokittelua. Samaan luokkaan kuuluvilla reaalimaailman käsitteillä on samat ominaisuudet, niihin liittyy samankaltaisia tapahtumia, jne. Esimerkiksi “pankin-asiakkaat” ja “pankin-tilit” eivät kuulu samaan luokkaan. Halutaan siis luokitella aina yhden tyyppiset kohdealueen ”oliot” vain yhden nimen (luokan / käsitteen) alle (-> ei päällekkäisiä käsitteitä). Silti kaikki monimutkaisemmat kokonaisuudet vaativat yleensä käsitteen jakamisen useammaksi käsitteeksi (lasku, laskurivi, …)

Käsite on yleensä joku keskeinen tunnistettava kokonaisuus, joka voi olla:

- konkreettinen (osa, asiakas...) tai abstrakti (työnimike, vakuutusturva...)
- pysyväluontoinen
- joskus tapahtumatyyppinen (tilausrivi)
- omaa yleensä jonkinlaisen avaimen
- määrittelyssä (esim. käyttötapauksissa) usein substantiiveja.

Organisaatioissa, projekteissa, ym. olisi hyvä olla yhtenäinen politiikka käsitteiden nimeämiseen. Tällä kurssilla käytetään seuraavanlaista politiikkaa:

- Käsite on jotain, josta haluamme tallentaa tietoa.
- Nimen tulisi heijastaa tätä + ominaisuuksia (tietoja), jotka kuvaavat käsitteen
- Käytetään kohdealueen terminologiaa
- käsitteiden nimet yksikössä ja viittaavat yhteen käsitteen instanssiin
    - *Tili* – ei *tilit*
- Ei toteutusteknisiä nimiä
    - *Asiakas* mieluummin kuin *asiakastiedosto*, *asiakas tietue*, …
- Ei raporttien nimiä
    - *Tuote* mieluummin kuin *tuoteluettelo*
- Ei mielellään lyhenteitä
- Nimet yksilöllisiä

Suhteet esiintyvät määrittelyssä usein verbeinä. Suhteita etsittäessä suhteita tarkentavat osuudet on myös huomioitava, kuten:

- vaihtoehtoisuus, lukumäärä-suhteet, rajoitteet, …
- henkilö lähettää aina hakemuksen
- Opiskelija saa kurssista arvosanan jos…
- joskus, täytyy, … useita, …
- Opiskelijalle kirjataan pisteet jokaisesta osasuorituksesta erikseen.

Suhteet pitäisi nimetä siten, että kun kaaviota tulkitaan, sitä pitäisi pystyä lukemaan selkokielisinä lauseina. Tässä mielessä suhteiden nimet pitäisi ajatella ”lauseiden osia”. Kaikki suhteet olisi yleensä hyvä nimetä (osa kaavion semantiikkaa) ja on hyvä huomata, että suhteet voidaan nimetä selvyyden vuoksi molempiin suuntiin erikseen

Ominaisuuksien ”löytäminen” ja nimeäminen

Ominaisuudet antavat käsitteille merkityksen. Ne kuvaavat suoraan jotain käsitettä. Määrittelyssä ominaisuudet ovat usein substantiiveja tai adjektiiveja. (HUOM! Adjektiivi voi myös kuvata ominaisuuden

ominaisuuksia (arvoalue)). Ominaisuudet voivat myös liittyä käsitteiden välisiin suhteisiin (-> assosiaatioluokka) ja myös ominaisuuksien tarkentavat osuudet otettava huomioon. Ominaisuuksien nimeämiseen pätee samat säännöt samoin kuin käsitteillä. Nimet ovat yksilöllisiä käsitteen sisällä.

Kertauksena mallinnusprosessi
Jokaiselle muodostuu oma tapansa mallintaa, mutta seuraavaa ”prosessia” voi käyttää ainakin pohjana ensimmäisiä käsitekaavioita laadittaessa:

- tunnista luokat (älä kursaile)
- turhat pois
- tunnista suhdetyypit (älä kursaile)
- turhat pois
- piirrä ensimmäinen kaavio
- lisää ominaisuudet
- sovella periytymistä, siis yleistä yleistä...
- ryhmittele tarpeen mukaan
- testaa saantipolut

- ITEROI ITEROI ITEROI...

Yhden kohdealueen kuvaamiseen ei ole olemassa vain yhtä ainoata oikeata ratkaisua. Niinpä erilaisia vaihtoehtoja kannattaa tarkastella avoimin mielin.

Usein on mm. mahdollista mallintaa jotkin reaalimaailman käsitteet joko ominaisuutena tai luokkana. Yleisperiaatteena voidaan pitää, että jos jostakin reaalimaailman käsitteestä mallinnetaan muutakin kuin pelkkä nimi, tulisi se mallintaa luokkana.

Joskus on myös mahdollista mallintaa suhde yhtenä monijakoisena suhteena tai useana kaksijakoisena suhteena. Näissä tapauksissa valitaan jälleen paremmin (luonnollisemmin) reaalimaailmaa kuvaava tapa.

Suunnittelumalli tarkoittaa rankasti yleistettynä ratkaisua johonkin yleisesti esiintyvään ongelmaan. Tällä kurssilla ei varsinaisesti opeteta suunnittelumalleja, mutta ainakin yksi suunnittelumalli on hyvä tietää – nimittäin **kooste-malli** (composite pattern). Mallin avulla voidaan esittää hierarkioita siten, että käsitteitä ja niiden osia voidaan kaikkia käsitellä samalla tavalla. Esimerkki kooste-mallin käytöstä on kuvassa 3.15.

Kuva 3.15 Esimerkki kooste-mallista.

2.7 Käsitekaavion laatu

Aiemmin todettiin, että käsitteellisen suunnittelun tärkeimpiä laatukriteerejä ovat mallin oikeellisuus ja luettavuus. Tarkemmin sanottuna laatuvaatimuksia ovat:

- **Täydellisyys**
  Malli esittää kaikki (olennaiset) kohdealueen piirteet.
- **Oikeellisuus**
  Malli on todellisuuden (kohdealueen) vääristämätön esitys.
- **Luettavuus**
  Kaavio on luonnollinen, helposti luettava ja ymmärrettävä
- **Minimaalisuus**
  Malli ei sisällä mitään ylimääräistä
- **Riittävyys**
  Lisämääreitä ei tarvita – malli kuvaa kaiken tarvittavan riittävän tarkasti.
- **Normaalisuus**
  Malli ei sisällä turhaa toistoa. Vaatimuksiin liittyvät piirteet esitetään vain kerran

Käsitekaavion laatu varmistetaan mm. katselmoimalla käsitekaavio ja vertaamalla sitä muuhun määrittelyyn (tapahtumiin, tarvittaviin raportteihin, näyttöihin, jne.). Katselmointeja tehdään yleensä ensin mallintajien kesken sisäisesti, jolloin pyritään löytämään mahdolliset ilmeiset virheet, toisteisuudet ja tietojen puutteellisuudet. Varsinainen tulikaste on kuitenkin aina katselmointi kohdealueen asiantuntijoiden kanssa (yleensä asiakkaan edustajia ja usein tulevan järjestelmän

käyttäjiä). He ovat oikeita henkilöitä arvioimaan mm. kaavion oikeellisuutta, täydellisyyttä ja luettavuutta.

Saantipolkuanalyysi on tapa varmistaa, että mallin mukaisesta tietokannasta on mahdollista saada kaikki tarvittavat tiedot ja suorittaa kaikki tarvittavat tapahtumat. Ideana on kuvata, mitä reittiä pitkin jokin tieto saadaan käsitekaaviosta. Samalla voidaan havaita mahdollisia virheitä, kuten:

- Onko tieto saatavissa yksikäsitteisesti
- Onko saman tiedon hakemiseen useita vaihtoehtoisia polkuja (toistoa)
- Saako tietoa ylipäänsä jotain polkua pitkin
- Vaikuttaako käsitekaavio liian monimutkaiselta (Esim. ovatko hakupolut turhan pitkiä)

Tarvittavat tapahtumat saadaan toimintojen määrittelystä. Polku voidaan kuvata joko graafisesti tai tekstuaalisesti. Graafisesti polku kuvataan piirtämällä käsitekaavioon (tai sen osaan) nuolet tiedon haun mukaisesti lähtien käsitteestä, jonka tietojen perusteella haku halutaan tehdä ja päättyen käsitteeseen, jonka tietoja haussa halutaan. Tekstuaalisesti hakupolku voidaan merkitä esimerkiksi seuraavasti: *Henkilö -> Työasema -> IP numerot*

Tässä Henkilö-käsitteessä on 1:1 suhde Työasema-käsitteeseen ja edelleen IP numeroihin. Monimutkaisemmassa tapauksessa suhteita voi olla useampia kahden käsitteen välillä, kuten esim. kuvitteellisessa projektiympäristössä henkilön ja projektin välillä. Suhde voi olla joko ¨työskentelee" tai "johtaa", jolloin tämä väli kuvataan:

Henkilö ---------> Projekti
(johtaa)

Mikäli nuolen suuntaan kulkiessa suhde "laventuu" eli määräsuhde on muu kuin 1:1 tai 0:1, tulee tämä merkitä tähdellä (*), jolloin ensimmäinen esimerkki voisi näyttää seuraavalta:

Henkilö ----> Työasema -> IP numerot
(*)

Tällainen merkintä tarkoittaisi, että henkilöllä voi olla useampia työasemia, joista kullakin on yksi IP numero. Kuitenkin siten, että Henkilön ja Työaseman välillä on ainoastaan yksi suhde. Suhteeseen liittyviin luokkiin navigoidaan samoin kuin muihinkin käsitteisiin.

Seuraavassa on haettu kuvan 3.12 käsitekaaviosta muutama saantipolku:

- Lista kaikista tietylle kurssille hakeneista hakijoista:

Kurssi ---> Hakukierros ---> Hakija
(*) (*)

- Tietyn valintakokeen tietystä osakokeesta tietyt pisteet saaneiden henkilöiden puhelinnumerot:

Valintakoe ---> Osakoe ---> Valintakoepisteet ---> Hakija ---> Puhelin
(*) (*)

- Hakijan valintakoepisteet tietyltä hakukierrokselta:

Hakukierros ----> Valintakoe ----> Osakoe ----> Hakija
(*) (*) (*)

Hakija ----> Valintakoepisteet
(*)

Luettavuus & ymmärrettävyys vinkkejä:
Kaavio olisi hyvä saada mahtumaan yhdelle arkille. Monimutkaisemmat ja laajemmat ongelmat vaativat tietysti enemmän tilaa, mutta tällöin pitää olla jotkut järkevät tavat jaotella (klusteroida) mallia ja esittää osamallit eli näkymät kukin omalla arkillaan.

Kaavion layoutin olisi hyvä olla sellainen, ettei kaaviossa ole (mielellään) risteäviä viivoja, siinä ei koskaan ole viivoja käsitteiden läpi. Keskeiset asiat kannattaa laittaa keskelle ja keskeisiin tapahtumiin liittyvät asiat helposti löydettäviksi ja luettaviksi. Hierarkiat kannattaa esittää hierarkian näköisinä, koska tällöin lukija välittömästi ymmärtää mistä on kysymys. Keskeisiä käsitteitä voi myös halutessaan korostaa (koko, tyyli, väri). Yksinkertaisuus on valttia (KISS), jos vain mahdollista

2.8 Tehtävä

Yritykseen ollaan toteuttamassa uutta tietojärjestelmää. Järjestelmän on tarkoitus auttaa asiakaskontaktien ylläpidossa sekä projektien suunnittelussa ja seurannassa. Järjestelmä sisältää tiedot yrityksen työntekijöistä. Työntekijöistä on tiedossa nimen ja yhteystietojen lisäksi henkilötunnus, työntekijälle maksettava tuntipalkka sekä työntekijän ulosmyyntihinta. Yrityksen vanhassa systeemissä asiakasyrityksistä on kerättynä nimi, osoite sekä useita yhteyshenkilöitä. Henkilöillä voi olla useita puhelinnumeroita. Asiakasyritykset tilaavat yritykseltä projekteja. Projekteilla on nimi, kuvaus, perustamispäivä ja projektipäällikkö joka on yrityksen työntekijä. Projekteja halutaan suunnitella ja seurata vaiheittain. Vaiheella on nimi, arvioitu työmäärä, toteutunut työmäärä sekä alkamisajankohta.

Tilauksesta on tiedossa tilauspäivämäärä, toimituspäivämäärät vaiheittain ja hinta.

Järjestelmän pitää pystyä vastaamaan mm. seuraavanlaisiin kysymyksiin:

- Tietyssä projektissa työskentelevät työntekijät?
- Asiakasyrityksessä työskentelevän henkilön kännykkänumero?
- Projektin tuntimäärät vaiheittain?
- Projektin kustannukset vaiheittain?
- Projektissa työskentelevän henkilön kustannukset projektissa tiettynä aikavälinä?
- Projektissa työskentelevän henkilön kustannukset kaikilta projekteilta?

Piirrä edellä kuvatun järjestelmän käsitekaavio käyttäen UML:n luokkakaavionotaatiota (eri paperille). Tee saantipolkuanalyysit kuvauksessa annetuille tapahtumille.

2.9 Arvoalueet

Käsitteiden, niiden välisten suhteiden, ominaisuuksien, rajoitteiden ja mahdollisesti metodien kuvauksen lisäksi tärkeä kuvauskohde on ominaisuuksien arvoalueet. Koska arvoalueiden kuvaukset voivat olla monimutkaisiakin, ne kuvataan yleensä käsitekaaviosta erillään, jonkinlaisena listana tai taulukkona.

Myös arvoalueet pyritään kuvaamaan toteutustavasta ja toteutus-ympäristöstä riippumattomalla tavalla. Jos esimerkiksi jokin ominaisuus voi saada tietyt arvot (esim. sukupuoli on mies tai nainen), kuvataan ko. ominaisuuden arvoalue luettelemalla sallitut arvot. Arvoalueissa voidaan asettaa rajoitteita esimerkiksi sille, kuinka monta merkkiä sanassa on, kuinka suuri tai pieni luku saa olla, käytetäänkö desimaaleja, tai missä muodossa tieto pitää muuten esittää. Tässä annetaan myös yleensä eri ominaisuuksille jonkinlainen selite ja tarvittaessa ominaisuuksille voidaan antaa vapaamuotoisia rajoitteita tai vaikka laskentasääntöjä.

Tapa, jolla arvoja kuvataan, pitää aina kertoa. Yksi mahdollinen tapa merkitä arvoalueet voisi olla seuraavanlainen:

Merkintä	***Selitys***
+	*ja*
( )	*valinnainen (voi puuttua)*
{ }	*toisto (0-N kertaa)*
n{ }m	*toisto n-m kertaa*
n-m	*väli n:stä m:ään*
[ ]	*Vaihtoehtoja*
|	*Vaihtoehtojen erotin*
@	*Avainominaisuus*
***	*Selite muodossa * teksti **
/	*Automaattisesti täytettävä tai laskettava kenttä*
M	*** 8-bittisen ascii-merkistön kirjain, numero tai muu kirjoitusmerkki. **
K	*A-Ö | a-ö*
N	*0-9*
P	*** Päiväys, josta pv, kk ja vuosi selviävät yksikäsitteisesti**

Yksittäiset kirjoitettavat merkit esitetään ' -merkkien välissä. Esimerkiksi 'A'. Kirjoitettavat merkkijonot esitetään lainausmerkkien välissä. Esimerkiksi "Aasi".

Määritellään seuraavassa esimerkkinä muutamien kuvan 3.12 käsitteiden ominaisuuksien arvoalueet:

Hakija	Sukunimi + Etunimi + @Hetu + /Sukupuoli + Kotikunta * hakijan perustiedot *
Hetu	0{M}11 * Hakijan henkilötunnus. Oikeellisuus tarkistettava syötettäessä tarkistemerkin perusteella. *
Sukunimi	0{M}40 * hakijan nykyinen sukunimi *
Etunimi	0{M}40 * hakijan etunimet *
Sukupuoli	”Mies” | ”Nainen” * automaattisesti hlötunnuksen perusteella *

Osoite	Postiosoite + Postinumero + Postitoimipaikka + Puhelin
Postiosoite	0{M}30
Postinumero	0{N}5
Postitoimipaikka	0{M}30

Hakukierros	Vuosi + Alkuaika + Loppuaika * Hakukierroksen perustiedot. Alku- ja loppuaika ovat hakukierroksen tietojen hakuperuste. *
Vuosi	@NNNN * Vuosi, jolle hakukierros kohdistetaan *
Alkuaika	@P
Loppuaika	P

Tehtävä: *Määrittele kohdan 2.8 esimerkistä käsitteisiin henkilö, työntekijä ja puhelin liittyvät arvoalueet.*