

|  |
| --- |
| CSC-Tieteen tietotekniikan keskus |
| **XDW** |
|  |
| Ohje |
| **Käsitemallin ER-konversio** |
| Versio 1.2 23.3.2012 |

Muutoshistoria

|  |  |  |  |
| --- | --- | --- | --- |
| **Versio** | **Pvm** | **Laatijat** | **Kuvaus** |
| 0.1 |  | KJ |  |
| 1.1 | 14.03.2011 | KJ | Poistettu kohta 2.2.6 Transformaatiot, lisätty Deprecated-ohjeistus. |
| 1.2 | 23.03.2012 | KJ | Lisätty teknisten kenttien käyttötapojen kuvaukset. Postettu kohta 2.2.5 Hierarkiat. |
| 1.3 | 30.5.2012 | TK | ER-konversiossa date->date (ei enää datetime) |
| 1.4 | 27.5.2013 | TK | URL tai URI -> nvarchar(400) (oli:200) OKM pyynnöstä |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sisällysluettelo

1 Käsitemallin rooli tietovarastoinnissa 4

1.1 Mallien keskinäinen suhde 4

1.2 Käsitemallinnus ja mallien konvertointi 5

1.3 Fyysisen tietokantarakenteen ylläpito 5

2 Konversio-ohjeet 6

2.1 Kantaobjektien nimeäminen 6

2.2 Tietokantataulun muodostaminen 10

1. Käsitemallin rooli tietovarastoinnissa
	1. Mallien keskinäinen suhde

Käsitemalli kuvaa sitä tietoa jonka tallentamisen tietokannan tulee toteuttaa. Käsitemallia vastaava tietokantamalli kuvataan perinteisesti ER-kaavioilla, joka on käsitemallia teknisempi kuvaustapa, ja josta löytyy mm teknisiä tunnistekenttiä ja muita hallinnollisia tietokenttiä, jotka eivät käsitemallissa ole relevantteja. Kutakin käsitemallia kohti voidaan luoda useita hyvinkin erirakenteisia ER-malleja, tietokannan toiminnallisista vaatimuksista riippuen. Yksinkertaisimmillaan ER-malli on suoraan käsitemallin ”näköinen” siten, että kustakin käsitteestä muodostetaan oma tietokantataulunsa, mutta esim Data Vault –toteutuksessa ER-malli on hyvinkin erinäköinen käsitemalliin verrattuna. Myös esim monikielisen tiedon tukeminen johtaa käsitemallista eriävään rakenteeseen, jos kieliversioidut tiedot viedään omiin tauluihinsa. Vastaavien teknisten rakenteiden mallintaminen käsitemalliin ei lisää käsitemallin ymmärrettävyyttä, eikä näin ollen ole mielekästä.

ER-mallista voidaan edelleen muodostaa fyysinen tietokantamalli, joka on valmistajakohtainen ER-mallin mukainen rakennekuvaus. Valmistajakohtaisia elementtejä näissä ovat mm tietotyyppien nimet sekä kannan objektien luontilauseiden syntaksi.

* 1. Käsitemallinnus ja mallien konvertointi

Tietovarastoinnissa pyritään minimoimaan manuaalisesti ylläpidettävien kohteiden lukumäärä. Käsitemallinnus tulisi suorittaa siten, että ER-malli voidaan johtaa suoraan käsitemallista, yksiselitteisiä konversiosääntöjä noudattamalla.

Käsitemallissa voi kuitenkin olla rakenteita, jotka tehokkuussyistä tapauskohtaisesti tulisi muuntaa ER-malliksi jollakin poikkeavalla tavalla (tyypillisesti hierarkiat sekä periytymisrakenteet). Näidenkin tulee perustua konversiosääntöihin, ja kyseiset tilanteet tulee merkitä malliin siten, että ne voidaan tunnistaa ja tuottaa ajatellunlainen ER-konversio. Kaiken kaikkiaan erikoistapaukset tulee kuitenkin minimoida; mitä lähempänä ER-malli on rakenteellisesti käsitemallia, sen ymmärrettävämpi ER-malli on käsitemallin lukijoille. Jos muunnos on radikaali, on syytä harkita käsitemallin muuttamista vastaamaan läheisemmin ER-mallia.

* 1. Fyysisen tietokantarakenteen ylläpito

Koska eri asiakkaat ottavat käyttöön tietokannan eri osia omien tarpeidensa ja aikataulujensa puitteissa, syntyy tarve ylläpitää tietoa siitä, missä vaiheessa asiakaskohtaiset käyttöönotot ovat tapahtuneet, ja erityisesti mihin tauluihin asiakkaat tietoja lataavat. Tämä on erityisen olennaista silloin, kun asiakas luo tietokannan omaan ympäristöönsä, ja siitä on tarkoitus tuoda yhteensopivaa tietoa mahdollisesti muuttuneeseen rakenteeseen, tai mikäli asiakas aikoo päivittää kantarakenteensa uudempaan versioon.

Koska fyysinen kantarakenne johdetaan suoraan käsitemallista, on olemassa kolme tilannetta joissa kantarakenne voi muuttua:

* käsitemallissa tehdään muutos, joka suoraan vaikuttaa siitä johdettuun taulurakenteeseen
* käsitemallissa ei tehdä muutosta, mutta konversioalgoritmia muutetaan siten, että se jostakin mallielementistä tai mallinnustavasta tuottaa tähänastisesta poikkeavan kantarakenteen
* tietotyyppien vastaavuuksia muutetaan

Käsitemalliin kohdistuvat muutokset voivat myös olla sellaisia, jotka eivät vaikuta kantarakenteeseen. Nämä muutokset ovat esim kuvaustekstien muuttuminen, kansiorakenteiden muuttuminen, käsitteen siirtyminen kansioiden välillä, attribuuttien kardinaliteetin muuttuminen esim valinnaisesta pakolliseksi (jos kantakonversiossa aina sallitaan null-arvot tietokentille) tai jonkin yhteyden siirtäminen toiseen käsitteeseen samassa käsitehierarkiassa (riippuen tavasta käsitellä kyseinen periytymishierarkia konversiossa).

1. Konversio-ohjeet
	1. Kantaobjektien nimeäminen

Rakenteelliset kantaobjektit (taulut, kentät) nimetään pienimmän yhteisen nimittäjän mukaisesti siten, että samaa nimeämistä voidaan käyttää ainakin Oracle sekä Microsoft tietokantoja varten. Käytännössä tämä tarkoittaa että kantaobjektien nimien pituudet pitää Oracle-yhteensopivuuden mahdollistamiseksi rajata 30:een merkkiin. Taulujen nimien osalta käytetään luettavuuden maksimoimiseksi 27:n merkin maksimipituutta, jotta kyseiseen tauluun viittaaviin kenttiin voidaan lisätä liite ”\_Id” viittauksen kohteen tunnistamisen helpottamiseksi. Attribuuttikenttien nimen maksimipituus on 30 merkkiä.

Lähtökohtaisesti kaikki käsitemallin elementit nimetään käsitemallissa kuvaavasti ja mahdollisimman luonnollisesti ja/tai jotakin tiettyä nimeämiskäytäntöä hyödyntäen. Käsitemallin elementtiä vastaavan tietokannan objektin nimi johdetaan algoritmisesti suoraan käsitemallin elementin nimestä. Kantaobjektinimiä voidaan muodostaa suoraan seuraavista käsitemallissa eksplisiittisesti nimetyistä mallielementeistä: Käsitteen nimi, ominaisuuden nimi, yhteyden nimi sekä roolin nimi.

Käsitemallinnuksen osalta on sovittu, että many-to-many –yhteydet aina mallinnetaan omina käsitteinään. Mikäli käsitemallissa kuitenkin on ”sallittu” nimeämätön many-to-many –yhteys, ER-konversio tulee keskeyttää, ja puutteellisesti nimetylle yhteydelle tulee viedä käsitemalliin kuvaava nimi. Yhteys nimetään kuin jos se olisi käsitteiden välinen yhdistävä roolikäsite (esim Henkilön osoite, Henkilö tilassa).

**Nimikonversio**

Oracle konvertoi automaattisesti kaikki nimet pelkästään ISOJA KIRJAIMIA sisältäviksi, Microsoft QSL Server taas säilyttää alkuperäisen aakkoslajin. Käsitemalli->ER-mallikonversiovaiheessa ei kuitenkaan erikseen muuteta kirjaimien aakkoslajia.

Nimenmuodostamisen askeleet:

Poistetaan nimestä muut kuin alfanumeeriset merkit. *CIMO:n koodi => CIMOn koodi*

Korvataan välilyönti alaviivalla ”\_”. *Henkilön kunta => Henkilön\_kunta, CIMOn koodi => CIMOn\_koodi*

Konvertoidaan skandit ö, Ö => o, O sekä å, Å, ä, Ä => a, A. *Henkilön\_kunta => Henkilon\_kunta*





Esimerkkejä

**Nimen typistäminen**

Mikäli yllä oleva algoritmi tuottaa merkkijonon jonka pituus on suurempi kuin sallittu maksimipituus, sovelletaan alla kuvattua ohjeellista typistämissäännöstöä kunnes merkkijonon pituus on sallituissa rajoissa.

* Sanaa typistetään mahdollisimman vähän. Luettavuus on typistetyn sanan tärkein ominaisuus.
* On parempi typistää useampaa sanaa vähän kuin yhtä sanaa paljon, mikäli luettavuus kärsii.
* Yhdyssanan yksittäinen komponentti tulkitaan yksittäiseksi typistettäväksi sanaksi. Luettavuus kärsii jos yhdyssanasta poistetaan vain loppupäätä.
* Yksittäinen typistettävä sana käsitellään siten, että sen lopusta poistetaan tarvittava määrä kokonaisia tavuja ja viimeiseksi lisätään seuraavan tavun ensimmäinen kirjain. Typistettäväksi sanaksi valitaan lähtökohtaisesti ensin nimen pisin sana. Esimerkkejä:
	+ *Maantieteellinen\_postinumero* (28, max 27, koska kyseessä on tauluksi muodostettava käsite)
		- Tavutus: *Maan-tie-teel-li-nen\_pos-ti-nu-me-ro*
		- Lopputulos: *Maantieteellin\_postinumero* (26)
	+ *Tutkimusryhma\_tutkimustoiminnan\_tuloksessa* (52, max 27)
		- *Tutkryhm\_tutktoim\_tulokses* (26, sovellettu kaksi kertaa yhdyssanaohjetta)
* Viimeisenä keinona poistetaan yksittäisiä vokaaleja, vähiten merkitsevät ensin
* Typistettävän nimen sisäisten ”\_”:lla eroteltujen sanojen lukumäärä pyritään pitämään samana. ”\_”:ien lukumäärä ei kuitenkaan voi kasvaa, eli esim *Rintamamiestalo*:a ei lyhennetä *Rint\_mies\_talo*:ksi. Esimerkki:
	+ Organisaation\_rooli\_tutkimustoiminnan\_tuloksessa
		- Kaikki sanat mukana: *Org\_rool\_tutktoim\_tulokses* (26)
		- Sanoja pudotettu: *Org\_rooli\_tutktoiminnassa* (26, ei kuitenkaan paras mahdollinen muunnos, koska poikkeaa käsitteen nimeämistavasta, mutta kuitenkin kuvaava)
* Voidaan sopia (ja dokumentoida alla olevaan taulukkoon) erityismuunnoksista, joita aina priorisoidaan, esim Henkilo => Hlo, ei Henk tai Henkil.

Automaattisen ER-konversion tueksi käytetään käsitemallissa DBNameOverride-nimistä Tagged Value –objektia, johon viedään se typistetty nimi, jota käsitemallin objektista halutaan käyttää ER-mallissa suoran nimikonversioalgoritmin lopputuloksen sijasta. Tätä mahdollisuutta käytetään vain ja aina silloin kun nimen pituus ylittää sallitun rajan. Typistetty nimi viedään käsitemalliin DBNameOverride-arvoksi suoraan sille objektille johon nimenmuutos kohdistuu (käsite, attribuutti, yhteys tai rooli), josta se luetaan kun vastaavalle ER-elementille annetaan nimi.

Erityismuunnokset:

|  |  |
| --- | --- |
| Alkuperäinen | Ensisijainen muunnos |
| päivämäärä | pvm |
| ulkoinen organisaatio | ulk\_org |
| kustannuspaikka | kustpaikka, kustpaik |
| kansainvälinen | kv |









Esimerkkejä

* 1. Tietokantataulun muodostaminen
		1. Yleistä

Pääasiassa samaan tietokantatauluun kerätään käsiteltävän käsitteen kaikki attribuutit, ja viittauskentiksi sellaiset yhteydet joissa yhteyden käsiteltävästä käsitteestä katsottuna toinen pää on kardinaliteetiltaan pieni (0..1 tai 1). Näiden käsitemallista johdettujen tietojen lisäksi kullekin taululle määritellään seuraava vakiojoukko hallinnallisia kenttiä:

|  |  |  |
| --- | --- | --- |
| Kenttä | Tietotyyppi | Kuvaus |
| Id | longint | Tiedon tekninen tunniste, Primary key. |
| Avain | nvarchar | Yhteen kenttään koottu tiedon luonnollinen avain, koostettu tiedon ominaisuuksista ja yhteyksistä. |
| Omistaja | nvarchar | Kuvaa tiedon omistavaa organisaatiota, esim HY, Aalto. Tämä tunniste sovittava toteutuksen alussa. |
| Kasite | nvarchar | Tiedon käsitemallin mukaisen käsitteen nimi selkokielisenä. Tämän tiedon avulla tietorivi voidaan linkittää esim käsiterekisterin tyyppiseen sovellukseen ja saada metatietoa käsitteestä. |
| Luotu | datetime | Aikaleima jolloin rivi on luotu tietovarastoon. |
| Päivitetty | datetime | Aikaleima jolloin tietovarastossa ennestään olleen rivin tietoja on päivitetty tietovarastoon. Uuden rivin kohdalla sama kuin Luotu-tieto. |
| Havaittu | datetime | Aikaleima, jolloin rivin yksilö on viimeksi havaittu lähdemateriaalissa. Tätä kenttää voi käyttää esim tiedon poistamiseen tietovarastosta. |
| Tietolahde | nvarchar | Kuvaa järjestelmää tai muuta tietolähdettä josta rivillä oleva tieto on peräisin. Samalle riville on voitu koostaa tietoa useasta lähteestä. Tällöin mainitaan kaikki lähteet. |
| Kommentti | nvarchar | Kommentti joka kuvaa rivillä olevaa mahdollista ongelmaa tiedon laadun suhteen. Esim jos riville siirtyy päivämäärätietoa joka on syötetty käsin lähtöjärjestelmään, eikä varmuudella pystytä sanomaan onko konversio mennyt oikein, tilanne kuvataan tähän kenttään. Muuten kenttä jätetään tyhjäksi. |

Kenttien käyttö kuvattu tarkemmin omissa kappaleissaan.

Periytymistilanteessa taulujen muodostus on tapauskohtaista. Oletusarvoisesti käsitehierarkia käsitellään siten, että koko hierarkia yhdistetään yhdeksi tauluksi, jonka nimi johdetaan käsitehierarkian juurikäsitteen nimestä. Tälle menettelylle löytyy kuitenkin poikkeuksia, ja ne kuvataan jäljempänä.

* + 1. Teknisten kenttien käyttöperiaatteita

#### Id

Id-kenttä on taulutasolla rivin yksilöivä juokseva numero (SQL Serverissä IDENTITY-kentäksi määritelty BIGINT), joka ei asettamisen jälkeen muutu. Id-kenttä on aina tekninen pääavain, Primary key, jolloin taulujen väliset viittaukset kohdistuvat aina siihen.

SQL Server ei millään yksinkertaisella tavalla tue tietokantatasolla uniikkia juoksevaa numerointia (vrt Oraclen sekvenssi). Mikäli muissa yhteyksissä tiedosta halutaan käyttää tietokantatasolla uniikkia yksilöivää avainta, se voidaan muodostaa käsitteen nimestä sekä varsinaisesta Id-kentän arvosta, esim Henkilö.1231731 tai Opinto-oikeus.6561265.

Muiden taulujen vierasavainkentät on määritelty siten, että ne eivät koskaan voi olla tyhjiä, ja tästä syystä kussakin taulussa, johon tehdään viittaus jostakin toisesta taulusta, tulee olla ns -1 –rivi, jonka Id-arvo on -1. Mikäli siis vierasavainkentän sisältö olisi loogisesti tyhjä (tieto puuttuu), sen arvoksi viedään perinteisen null-arvon sijasta -1. Tämä mahdollistaa taulujen välisen outer joinia tehokkaamman inner join -liitoksen tekemisen ilman että loogisesti puuttuva tieto rajaa kyselyn tietojoukkoa liikaa. Tällöin tulosjoukkoon mukaan tulleet -1 –viittaus-Id:llä oleva tieto voidaan raportointivälineessä käsitellä esim nimeämällä ne ”Tuntematon” tai ”Tieto puuttuu”, tiedon käyttötarkoituksesta riippuen.

-1 –rivin lisäksi joissakin tapauksissa voidaan haluta käyttää myös ns -2 –riviä. Sen toimintaperiaate on sama kuin -1 –rivin, mutta se kuvaa virhetilannetta, eli tilannetta, jossa viitattava tieto puuttuu, vaikka se on käsitemallissa määritelty pakolliseksi.

  



-1 ja -2 –kenttien käyttäytyminen

Yllä olevassa kuvassa henkilön 125 äidinkielestä tai asiointikielestä ei ole tietoa lähtöjärjestelmässä. Tietojen lataus toteutetaan siten, että käsitemallin määrittelemä äidinkielen pakollisuus tuottaa Aidinkieli\_Id:ksi arvon -2, kun taas puuttuva asiontikielitieto on sallittu tilanne, jolloin Asiointikieli\_Id:ksi viedään -1. Tiedon käytön kannalta tilanne on kuitenkin täysin sama, eli henkilöä ei voida yhdistää kieleen.

Tietovarastoprojektin alussa tulee päättää käytetäänkö molempia puuttuvan tiedon indikaattoria, vai vain -1:tä. Käytännössä -1 on riittävä tieto, ja -2:n käyttö monimutkaistaa tietojen latauksia ja käyttöä sekä luo ylimääräisen ylläpitoriippuvuuden käsitemalliin. Mahdolliset -2 –tilanteet voidaan hoitaa myös Kommentti-kentällä (kts alempana).

#### Avain

Avain-kenttään viedään tiedon luonnollinen avain, mikäli sellainen on olemassa. Luonnollinen avain on merkkijonopohjainen kooste taulun varsinaisista tietokentistä (käsitemallista lähtöisin olevat attribuutti- ja viittauskentät).

Avaintiedon tarkoitus on yksilöidä tieto lähtöjärjestelmäriippumattomasti. Käsitekohtainen avaintieto selviää ladattavan tiedon analyysin lopputuloksena. Periaate on, että avaintieto voidaan aina muodostaa kustakin lähtöjärjestelmästä tietoa ladattaessa, ja sitä voidaan verrata suoraan tietovarastossa olevaan tietoon ja päätellä luotettavasti onko käsiteltävä yksilö jo tietovarastossa vai ei. Avaintietona tai sen komponenttina ei tule käyttää lähtöjärjestelmässä olevaa tiedon teknistä tunnistetta tai surrogaattiavainta, koska se on aina lähtöjärjestelmäkohtainen, ja samalle yksilölle ei näin ollen voida taata samaa avaintietoa mistään muusta järjestelmästä poimittuna.

Avaintieto ei saa muuttua sen antamisen jälkeen. Mikäli avaintieto koostuu useasta komponentista, komponenttien keskinäinen järjestys ei saa muuttua.

Jotkin lataustyökalut edellyttävät että kohdetaulussa on yksi avaintietokenttä, ns Business key –kenttä (liiketoiminta-avain).

Esimerkki

Avaintieto voi koostua käsitteen yhdestä tai useammasta ominaisuudesta. Lisäksi siihen voi sisältyä muiden käsitteiden avaintietoja kokonaisuudessaan. Avaintietojen komponentit erotellaan toisistaan jollakin tunnistettavalla erotinmerkillä, tyypillisesti |:llä tai ¤:lla jotka eivät normaalisti esiinny osana käsitteen tallennetavia ominaisuutietoja.



Yhdistelmäavain

Yllä olevassa esimerkissä kunkin neljän käsitteen avaimet voidaan määritellä esim näin:

|  |  |
| --- | --- |
| Käsite | Avain |
| Puhelinnumero | Numero: yksittäinen puhelinnumero yksilöi itse itsensä. Esimerkki +358 09 123 4567 |
| Henkilö | Henkilötunnus (piilossa, kts myös alempana). Kullakin henkilöllä oletetaan tässä olevan uniikki henkilötunnus. Esimerkki 010203-1234 |
| Puhelinnumeron rooli | Koodi. Koodistossa huolehdittu koodin yksiselitteisyydestä. Esimerkki: 1 = Ensisijainen numero |
| Henkilön puhelinnumero | Yhdistelmäavain koostuen Henkilön, Puhelinnumeron sekä Puhelinnumeron roolin avaimista sekä Alkamispäivämäärästä. Esimerkki 010203-1234|+358 09 123 4567|1|1.1.2010| (elementtien järjestys ja muoto olennainen). |

Samalla käsitteellä voi kuitenkin käytännössä lähtöjärjestelmästä riippuen olla useampi luonnollinen avain. Käsitemallia laadittaessa kuvataan *tietotarve* mallinnushetkellä tiedossa olevien tarpeiden mukaisesti, eikä voida mitenkään edellyttää että tulevaisuudessa latauksien lähteenä toimivissa tietojärjestelmissä haluttu tieto on saatavilla käsitemallin tarkkuudella. Näin ollen ei voida myöskään edellyttää että kaikkien käsitteiden avaintietojen kaikki komponentit ovat saatavilla. Avaintiedon aukoton määritteleminen käsitemallitasolla ei siis aina ole mahdollista.

Esimerkki ongelmatilanteesta

Käsitemallissa on määritelty henkilön avaintiedoksi Henkilötunnus, joka kiistatta jokaiselta tietovaraston piiriin kuuluvalta henkilöltä löytyy. Muita henkilön ominaisuuksia ovat nimitiedot, henkilönumero ja sähköpostiosoite, joista kaksi viimeistä myöskin yksinään yksilöivät henkilön. Nimitiedot eivät sovellu henkilön tunnistamistarkoitukseen.

Henkilö halutaan ladata järjestelmistä A, B ja C.

* A:ssa henkilöllä on nimitiedot, sähköpostiosoite ja henkilönumero
* B:ssä henkilöllä on nimitiedot, sähköpostiosoite ja henkilötunnus
* C:ssä henkilöllä on nimitiedot ja henkilönumero

Tästä seuraa:

* Vain B-järjestelmästä voidaan ladata henkilöt suoraan tietovarastoon. Kaikki henkilöt voidaan ladata.
* A-järjestelmästä voidaan ladata vain ne henkilöt, jotka samanaikaisesti löytyvät myös B-järjestelmän poimintamateriaalista, yhdistämällä tiedot latausalueella käyttäen sähköpostiosoitetta yhdistävänä tekijänä. Näiden henkilöiden osalta käytännössä ainoastaan tietovarastossa oleva Henkilönumerotieto päivittyy, muut tiedot löytyvät sieltä jo B-latauksen ansiosta.
* C-järjestelmän henkilöille saadaan henkilötunnus vain niille henkilöille jotka löytyvät sekä A:sta (C.Henkilönumero -> A.Henkilönumero) että B:stä (A.Sähköpostiosoite -> B.Sähköpostiosoite)

A:lle ja C:lle uniikit henkilöt siis jäävät tässä tilanteessa lataamatta. Vastaavassa tilanteessa ei siis voida käyttää yhtä ainoaa Avain-kenttää tiedon luotettavaan tunnistamiseen, vaan latauskohtaisesti käytetään sitä yksilöivää tietoa joka on saatavilla:

* A:sta ladattaessa voidaan käyttää joko sähköpostiosoitetta tai henkilönumeroa
* B:stä ladattaessa voidaan käyttää joko sähköpostiosoitetta tai henkilötunnusta
* C:stä ladattaessa käytetään henkilönumeroa

Tällä tavalla henkilötieto voidaan kohdistaa oikein lähtöjärjestelmästä riippumatta; käytännössä siis eri lähtöjärjestelmistä tulevat tiedot täydentävät tietovarastossa jo olemassa olevaa henkilöriviä silloin kun sama henkilö esiintyy useassa tietojärjestelmässä.

#### Omistaja

Omistaja-kentän käyttö on ajankohtainen silloin kun samaan tauluun tuodaan useamman organisaation omistamaa tietoa (miniasiakasympäristö). Kullekin organisaatiolle kuuluva tieto merkitään ko organisaation omistamaksi jollakin sovitulla arvolla (esim merkkijonoilla ”HY”, ”OY”, ”Aalto”).

Sellaiselle tiedolle, jonka halutaan olevan kaikille tietovarastoa käyttäville organisaatioille yhteinen, voidaan Omistaja-tieto jättää tyhjäksi. Muussa tapauksessa kenttä yksilöi aina tiedon omistajaorganisaation, ja hyödyntämisrajapinnan toteutuksessa tulee huolehtia siitä, että tieto näkyy vain kyseiselle organisaatiolle. Omistajaton tieto saa näkyä kaikille.

Kun käytetään Omistaja-tietoa, tämä tulee huomioida myös tiedon latauksessa avaintiedon ohella. Omistajatieto ja Avain-tieto (tai muu yksilöivä tieto) yhdessä siis muodostavat sen tiedon, jota latausvaiheessa käytetään olemassa olevan rivin tunnistamiseen.

Mikäli tietovarasto toteutetaan pelkästään oman organisaation käyttöön, Omistaja-kenttää ei tarvitse huomioida.

Esimerkki – koodistot

Osa koodistoista on sellaisia, joiden arvojoukko on siinä määrin staattinen tai säädelty, että niiden sisällössä ei ole asiakaskohtaista variaatiota. Tällainen koodisto voidaan toteuttaa moniasiakasympäristöönkin ilman Omistaja-tietoa. Sellaiset koodistot joissa on asiakaskohtaista variaatiota voidaan moniasiakasympäristöön tallentaa Omistaja-tiedolla. Nämä kaksi metodia voidaan myös yhdistää, eli tallentaa koodiston yhteinen/staattinen osa ilman Omistaja-tietoa ja asiakaskohtainen variaatio Omistaja-tiedolla, tai jopa käyttää omistajatonta tietoa oletusarvoina, mikäli omistajakohtaista tietoa tietyllä koodilla ei löydy.

Nämä valinnat vaikuttavat olennaisesti sekä tiedon latauksien että hyödyntämisrajapinnan toteutustapoihin, ja niiden vaikutukset tulee arvioida tarkoin ennen tietovarastoprojektin aloittamista.

 

Asiakaskohtainen sekä yhteinen sukupuolikoodistotaulu

Esimerkki – Henkilötaulu

Myös muut kuin koodistotaulut voidaan toteuttaa yhteisenä tietona, mikäli koetaan että sillä saavutetaan jotakin olennaista hyötyä. Sellaiset käsitteet jotka kuvaavat reaalimailman ylesluontoisia ilmiöitä (henkilö, rakennus, osoite jne) ovat tämän kaltaisia potentiaalisia yhteisiä tietoja, ja mikäli tällaisen esiintymistä eri asiakasorganisaatioissa halutaan seurata vaivattomammin, käsitteen lataukset voidaan toteuttaa ilman Omistajatietoa. Tämän kaltaisessa tilanteessa täytyy varmistua siitä, että käsitteen sisältö on kaikilta osin omistajariippumaton.

Omistajarajat ylittävät kyselyt voidaan tarvittaessa toteuttaa myös muilla keinoin, eli esim samalla henkilöllä tulisi olla jokin sama tunnistetieto riippumatta siitä minkä Omistajan järjestelmistä hänen tietonsa ladataan.



Keskitetty henkilötaulu sekä Omistajalla eroteltu Henkilötaulu, jossa sama henkilö toistuu maksimissaan kerran per omistaja

#### Kasite

Kenttä sisältää taulunimen ”selkokielisen” nimen, eli nimen, joka käsitteelle on annettu käsitemallissa ja johon kyseinen taulu perustuu.

Mikäli taulu sisältää useita periytymishierarkian käsitteitä, kenttä toimii ns diskriminaattorina, eli sen perusteella voidaan päätellä mitä periytymishierarkian käsitettä se edustaa. Taulun latauksen yhteydessä pitää tässä tapauksessa päättää, mitä periytymishierarkian käsitettä ladattava rivi edustaa.



Opetuksen toteutuksen lataus

Yllä oleva tilanne kääntyy ER-malliksi siten, että syntyy vain yksi Opetuksen\_toteutus –taulu, josta löytyy sekä Opintokokonaisuus\_Id että Opintojakso\_Id. Näistä kahdesta vierasavainkentästä vain toinen voi olla kerrallaan käytössä, koska opetuksen toteutus on joko Opintokokonaisuuden toteutus tai Opintojakson toteutus.





Nämä voidaan toteuttaa kahtena latauksena, yksi kutakin konkreettista alikäsitettä kohti. Lataus ottaa aina suoraan kantaa siihen mitä käsitettä parhaillaan ladataan. Samaan tauluun vietäessä syntyy alla olevan näköistä tietoa:



#### Luotu

Ilmaisee koska rivin on alun perin luotu tietovarastoon. Tieto ei muutu rivin luomisen jälkeen.

#### Paivitetty

Ilmaisee koska rivin sisältö on viimeksi päivitetty tietovarastoon. Kentän tietotyypin ollessa päivämäärätyyppiä, sen tulee olla niin tarkka, että se riittää erottelemaan saman päivän aikana ajettujen latauksien päivitykset.

#### Havaittu

Ilmaisee koska rivin mukainen tieto on havaittu käsiteltävässä materiaalissa, riippumatta siitä onko riviä päivitetty tietovarastoon kyseisen materiaalin pohjalta vai ei.

Kun tietovaraston lataus suoritetaan koko materiaalilla (vrt ei vain edellisen latauksen jälkeen muuttuneella), sellaiset rivit, joiden Havaittu-päivämäärää ei päivitetä kyseisen latauksen yhteydessä, ovat käytännössä lähtöjärjestelmistä poistettuja tietoja. Nämä rivit voidaan poistaa myös tietovarastosta, mikäli tämä on tietovarastolle määriteltyjen käyttötarkoituksien ja –tapojen mukaista.

Osittaisella materiaalilla suoritetun latauksen jälkeen Havaittu-päivämäärää ei suoraan voi käyttää päättelyyn onko tieto poistettavissa tietovarastosta vai ei.

Kentän sisällön on tarkoitus tukea tietovaraston tiedon eheyttä ja ajantasaisuutta, ja kentän sisällön käyttötapa ja -tarkoitus tulisi määritellä tietovarastoprojektin alussa.

Esimerkki

Lähtötieto, lataus 1.1.2012



Tietovarasto, lataus 1.1.2012



Lähtötieto, lataus 2.1.2012



Tietovarasto, lataus 2.1.2012



Lähtötieto, lataus 3.1.2012



Tietovarasto, lataus 3.1.2012



Mikäli OUT\_Henkilo 3.1.2012 sisälsi kaikkien tietovarastoinnin piiriin kuuluvat henkilöt, voitaneen päätellä, että rivi Id:llä 124 voidaan haluttaessa poistaa tietovarastosta. Mikäli kyseistä henkilöä ei enää esiinny missään lähtöjärjestelmässä, häneen ei voi näissä järjestelmissä liittyä mitään muutakaan tietoa, ja tietovarastoon aiemmin viedyn henkilöön liittyvän muunkin tiedon osalta lienee Havaittu-tieto jätetty päivittämättä.

#### Tietolahde

Kenttään kirjataan rivillä olevan tiedon lähde jollakin tarkoituksenmukaisella tarkkuudella. Mahdollisia tietolähteitä ovat esim lähtöjärjestelmän nimi, lähtöjärjestelmän nimi ja taulu (tai useampi, jos tieto on yhdistetty useasta lähteestä/järjestelmästä).

|  |
| --- |
| Esimerkki |
| Oodi *(lähtöjärjestelmä)* |
| Oodi, Personec, Henkilörekisteri *(useita lähtöjärjestelmiä yhdistetty samalle riville)* |
| Oodi.hlo *(lähtöjärjestelmä + taulu)* |
| Oodi.hlo, Personec.henkilo, Henkilörekisteri.Henkilö jossa Aktiivinen = true *(lähtöjärjestelmä + taulu + jokin rajauskriteeri)* |
| Oodi.hlo.SUKNIM, Oodi.hlo.ETUNIMI *(lähtöjärjestelmä, taulu, kenttä)* |

Kentän sisällön on tarkoitus tukea tiedon jäljitettävyyttä, ja kentän sisällön käyttötapa ja -tarkoitus tulisi määritellä tietovarastoprojektin alussa.

#### Kommentti

Kommenttikenttään viedään riviä ladattaessa tai sitä päivitettäessä tyypillisesti tiedon laatuongelmiin liityvä sanallinen kuvaus. Jos esim ladatun päivämääräkentän arvo perustuu lähtöjärjestelmään käsin vietyyn tietoon ”020305”, Kommenttikenttään voidaan viedä teksti ”Epäluotettava päivämäärätieto 020305”. Tietovaraston yhteydessä käytettäviä raportointivälineitä voidaan tällöin käyttää virheraporttien tuottamiseen, joihin poimitaan ne rivit, joiden Kommentti-kenttässä on sisältöä. Virheraportin perusteella ongelmallinen tieto tulisi mahdollisuuksien mukaan korjata lähtöjärjestelmään.

Kentän sisällön on tarkoitus tukea tiedon laadun parantamista, ja kentän sisällön käyttötapa ja -tarkoitus tulisi määritellä tietovarastoprojektin alussa.

* + 1. Historiatiedot

Termillä ”historiatieto” voidaan tarkoittaa useita eri asioita. Alla muutama tulkinta:

* Implisiittinen historiatieto: tietoa kertyy ajan myötä. Samasta tiedosta ei ole useaa versiota. Historiatieto tarkoittaa tässä tietoa asioista jotka on viety tietovarastoon ”kauan sitten”. Tieto on myös mahdollisesti poistettu lähtöjärjestelmästä, mutta se on edelleen olemassa tietovarastossa.
* Luonnollinen historiatieto: tiedon mukana tulee sitä kuvaava luotettava voimassaolotieto, joka on mallinnettu ja tallennetaan tietovarastoon tietona osaksi muuta tietoa. Voimassaoloa kuvaavilla päivämäärillä on vastine reaalimaailman tapahtumissa, esim henkilön osoite on ajalla 1.2.2003 – 31.1.2008 ollut *Kauppiaskatu 1*, ja 1.2.2008 alkaen *Urapolku 5*. Samasta tiedosta ei ole useaa versiota. Tässä tapauksessa ”historia” tallentuu henkilön asumisesta, ja tämä on tapauskohtaisesti määritelty juuri henkilön osoitetiedolle että näin pitää olla. Tiedon pitää myös olla saatavilla, sitä ei saa päätellä lataushetkestä.
* Tekninen historiatieto: samasta tiedosta on eri voimassaoloajalliset versiot. Aina kun latausmekanismi huomaa että tietovarastossa jo oleva tieto muuttuu, vanha tieto jää talteen ja uusi tieto astuu voimaan. Tästä syntyy tiedolle voimassaoloaikajaksoja, joilla ei ole mitään tekemistä reaalimaailman kanssa, vaan ajankohdat liittyvät puhtaasti tiedon lataushetkiin. Esim ETL-prosessi näki viime yönä että erään henkilön sukunimi on muuttunut, mutta lataus ei voi tietää, onko se oikeasti muuttunut vai onko pelkästään korjattu virhe. Lopputuloksena henkilön sukunimi on tietovarastossa ollut Jonsson 8.3.2012 00:05:23.0123 asti ja Johnsson 8.3.2012 00:05:23.0124 alkaen, koska se sattui olemaan se hetki jolloin tietovaraston tieto päivitettiin edellisen latauskerran jälkeen. Varsinainen kirjoitusvirhe on korjattu lähtöjärjestelmään milloin hyvänsä edellisen latauksen jälkeen (ja latauksia voidaan tehdä esim kuukausittain), mutta siitä tapahtumasta ei ole jäänyt jälkeä.
* Arkistoitu historiatieto: jostain tietystä tilanteesta syrjään pistetty tilanne, jonka alkuperäinen tallennuslogiikka voi olla mikä tahansa noista kolmesta aikaisemmasta. Esimerkkinä kantadumppi tai mittariarkiston tyyppinen ratkaisu, johon tietyin väliajoin lasketaan ja tallennetaan indikaattoriarvoja laskentahetken tilanteen mukaan.

Pääasiallinen tulkinta on tiedon versiointi, eli tekninen historiatieto, jolloin halutaan yleisesti talteen tiedon muutos jossa halutaan säilyttää tilanne ennen muutosta. Käytäntö on kuitenkin osoittanut, että suurin osa (eräässä vertailuasiakkuudessa n 90%) tietoihin kohdistuvista muutoksista on virheiden korjaamista tai puutteiden täydentämistä. Tästä seuraa se, että samansuuruinen osuus kaikesta historiatiedosta on joko virheellistä tai puutteellista, ja saman tiedon uusin versio on aina eniten oikein. Lähtökohtaisesti tietovaraston käyttäjiä ei kiinnosta se virheellinen tieto.

Ne tilanteet, joissa on olennaista tietää muutoshetki sekä tilanne ennen ja jälkeen muutoksen, perustuvat pääasiassa reaalimaailmassa tehtyihin päätöksiin ja suunniteltuihin muutosprosesseihin, esim organisaatio- ja koodistomuutokset. Nämä tilanteet tulee mallinnusvaiheessa tunnistaa ja mallintaa osaksi käsitemallia antamalla ko käsitteille sellaisia ominaisuuksia joilla versiotieto voidaan hallita, tyypillisesti Alkamis- ja Päättymispäivämäärätiedot. Nämä vaikuttavat esim avaintietojen rakenteisiin (esim aikaversioimaton koodisto => Avain = Koodi, aikaversioitu koodisto => Avain = Koodi + Alkamispäivämäärä), jolloin ”historialataus” käyttäytyy täysin samoin kuten muutkin lataukset.

* + 1. Käsitteestä taulu

Tauluiksi muodostetaan lähtökohtaisesti ne käsitteet, joille on käsitemalliin viety stereotyyppi <<Base>> (peruskäsite). Mikäli kuitenkin käsitteelle on määritelty stereotyyppi <<Deprecated>>, käsitettä ei toteuteta.

Periytymistilanne voidaan käsitellä monella tavalla. <<Base>>-stereotyypille on määritelty Enumeration-tyyppinen InheritanceStrategy-niminen Tagged Value, jolla ilmoitetaan haluttu periytymisen käsittelytapa.

<<Deprecated>>-stereotyypitettyä käsitehierarkiassa olevaa käsitettä alikäsitteineen ei toteuteta.

|  |  |
| --- | --- |
| InheritanceStrategy | Käsittely |
| <tyhjä> | Oletusarvo; kuten Consolidate all |
| All | Kukin periytymishierarkian käsite käsitellään kuten periytymishierarkiaan kuulumaton käsite, eli siitä syntyy taulu attribuutteineen ja yhteyksineen. Lisäksi periytymissuhteesta syntyy yhteys alikäsitettä vastaavasta taulusta yläkäsitettä vastaavaan tauluun. **Tällä hetkellä ei käytössä.**  |
| Consolidate all | Käsitehierarkia (periytymishierarkia) konvertoidaan lähtökohtaisesti yhdeksi tietokantatauluksi siten, että hierarkia käsitellään kuin jos käsitehierarkian ylimpään käsitteeseen olisi kerätty koko hierarkian kaikki attribuutit ja yhteydet. Konfliktitilanteissa (usea samanniminen kenttä) syntyy vain yksi kenttä. |
| Leaf classes only | Vain lehtikäsitteestä muodostetaan taulut, joihin tuodaan ylemmistä käsitteistä kaikki attribuutit ja ulos menevät yhteydet. <<Base>>-stereotyypillä on lisäksi ModelElement-tyypinen Redirect associations to –Tagged Value. Tähän merkitään (kullekin hierarkian käsitteelle johon kohdistuu ulkoapäin tuleva viittaus) se lehtitason käsite, johon käsitteeseen sisääntulevat viittaukset ohjataan. **Tällä hetkellä ei käytössä.** |
| Subclasses as boolean | Kuten Consolidate all, mutta luo jokaisesta alikäsitteestä boolean-tyyppisen kentän. Käytössä rooli-tyyppisissä Description-käsitteissä (Henkilön rooli, Organisaation rooli) |

Käsitehierarkia konvertoidaan lähtökohtaisesti yhdeksi tietokantatauluksi siten, että hierarkia käsitellään kuin jos käsitehierarkian ylimpään käsitteeseen olisi kerätty koko hierarkian kaikki attribuutit ja yhteydet.

Kukin algoritmi on mietitty ja toteutettu nyt ainoastaan tämänhetkisen tarpeen mukaisella tarkkuudella ja tämänhetkisten mallinnustapojen mukaan, eikä siis ole aukoton. All-vaihtoehtoa tai Leaf classes only –vaihtoehtoja ei ole toteutettu kokonaisuudessaan tai ollenkaan, koska niiden käyttämisestä ei ole vielä sovittu.

* + 1. Attribuutti kentäksi

Käsitteellä voi olla johdettuja ja ei-johdettuja attribuutteja. Kaikista ei-johdetuista attribuuteista luodaan tietokenttä käsitettä vastaavaan tietokantatauluun. Attribuutin ER-mallin vastaava tietotyyppi johdetaan käsitemallista seuraavan taulukon mukaisesti:

|  |  |  |
| --- | --- | --- |
| Attribuutti | Sarake (ER) | Sarake (MS SQL) |
| int | int | int |
| float | decimal(18, 6) | float(18, 6) |
| datetime | datetime | datetime |
| date | date | date |
| time | datetime | datetime |
| idstring | nvarchar(20) | nvarchar(20) |
| string | nvarchar(200) | nvarchar(200) |
| bool | bit | bit |
| url tai uri | nvarchar(400) | nvarchar(400) |
| email | nvarchar(200) | nvarchar(200) |
| text | nvarchar(4000) | nvarchar(4000) |

Attribuuttia vastaavalle kentälle sallitaan aina null-arvo, koska ei voida edellyttää, että kaikki korkeakoulut käyttävät kyseistä attribuuttia.

* + 1. Yhteys foreign key –kentäksi

Käsitemallin yhteys voi olla johdettu tai ei-johdettu. Foreign key –kenttä muodostetaan sellaisista ei-johdetuista yhteyksistä, joiden ”toisessa päässä” on muodostettavan tietokantataulun näkökulmasta kardinaliteetti 0..1 tai 1 (kaikki yhteydet ovat lähtökohtaisesti tällaisia, koska käsitemallinnuksessa ei sallita many-to-many –yhteyksiä).

Yhteyksistä, joiden molemmissa päissä on sama pieni kardinaliteetti (1 tai 0..1), ei voida automaattisesti päätellä kumman käsitteen mukaiseen tauluun kenttä luodaan. Näissä tapauksissa käsitemallissa tulee merkitä yhteyden kulkusuunta (Role.Navigable = false siihen päähän johon kenttä muodostetaan, Role.Navigable = true siihen päähän johon viitataan). Kulkusuuntaa ei anneta muissa tapauksissa.

|  |  |
| --- | --- |
| Yhteyden mallinnustapa | Konversio |
| A[0..\*] – B[0..\*] | Ei sallittu (many-to-many –yhteyksiä ei sallita) |
| A[1..\*] – B[0..\*] | Ei sallittu (many-to-many –yhteyksiä ei sallita) |
| A[1..\*] – B[1..\*] | Ei sallittu (many-to-many –yhteyksiä ei sallita) |
| A[1] – B[1] | Ei sallittu (sama pieni kardinaliteetti, suuntaa ei annettu) |
| A[0..1] – B[0..1] | Ei sallittu (sama pieni kardinaliteetti, suuntaa ei annettu) |
| A[1] –> B[1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[0..1] –> B[0..1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[0..\*] – B[0..1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[0..\*] – B[1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[1..\*] – B[0..1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[1..\*] – B[1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |
| A[0..1] – B[1] | FK-kenttä syntyy A:sta muodostettavaan tauluun |

 



Esimerkkejä

Mikäli yhteyden jompikumpi pää kohdistuu käsitehierarkiaan, se käsitellään käsitehierarkialle merkityn käsittelytavan mukaan.

Yhteydestä muodostetaan kenttä vain, mikäli yhteys viittaa käsitteeseen josta ajankohtaisessa konversiossa muodostetaan tietokantataulu, kohdan 2.2.2 mukaisesti. Kenttä jää siis kokonaan luomatta mikäli sillä ei ole kohdetta. Konkreettisesti, jos viitattava käsite ei ole merkitty <<Base>>-stereotyypillä, siihen viittaavasta yhteydestä ei synny attribuuttia. Myöskään <<Deprecated>>-tyypitettyyn käsitteeseen tai käsitehierarkian osaan viittaavaa yhteyttä ei toteuteta.

Foreign Key –kenttä ei salli null-arvoja. Null-arvon sijasta käytetään tietoa ladattaessa arvoa -1. Kaikissa tauluissa joihin viitataan tulee löytyä -1 –Id:llä oleva rivi.

**Foreign key –kentän nimeäminen**

Kentän nimen runko otetaan yhteyden vastakkaisessa päässä olevan roolin DBNameOverride-Tagged Value –kentästä, mikäli sellainen on annettu. Mikäli kyseistä arvoa ei ole, nimen runko johdetaan suoraan roolinimestä. Mikäli roolinimeä ei ole, nimirunko otetaan roolia vastaavan käsitteen DBNameOverride-kentästä. Mikäli tätä arvoa ei ole, nimen runko muodostetaan suoraan roolia vastaavan käsitteen nimestä. Runkoon lisätään pääte ”\_Id” merkitsemään viittausta.

 Esimerkkejä

Tässä tilanteessa voi syntyä konflikti (tauluun syntyisi useampi kenttä samalla nimellä) erityisesti suurissa käsitehierarkioissa, joissa hierarkian eri tasot viittaavat samoihin käsitteisiin joko samoilla roolinimillä tai kokonaan ilman roolinimeä. Näissä tilanteissa on täysin mahdollista että monistuvat kentät tarkoittavat täysin samaa asiaa, jolloin mitään olennaista ei menetetä luomalla vain yksi näistä samannimisistä kentistä. Tämä on kuitenkin viime kädessa tapauskohtaisesti ratkaistava tilanne, jossa tulee tarkastella käsitemallia ja tarpeen vaatiessa nimetä viittauksien roolit uudestaan, tai muodostaa taulurakenne jollakin toisella algoritmilla.

* + 1. Indeksit

Indekseillä nopeutetaan tietokantahakuja, jotka perustuvat tarpeisiin jotka eivät välttämättä ole tiedossa kantaa luotaessa. Näin ollen yleispätevää indeksointiohjetta ei voi antaa, vaan indeksointi tehdään aina tapauskohtaisesti.

Tiedon latausmekanismien käyttöön voidaan tietty joukko vakioindeksejä, tiedon lataustavasta riippuen. Esim Avain-kentän ollessa käytössä, sille tulee luoda indeksi. Indeksi ei kuitenkaan ole uniikki taulutasolla, mikäli samaan aikaan käytetään Omistaja-kenttää, jolloin uniikki indeksi muodostetaan näille kahdelle yhdessä. Jos taas taulu sisältää käsitehierarkian, Kasite-kentälle kannattaa luoda indeksi.

Käsitemallista johdetuista kentistä on hyvä luoda oma indeksi kullekin vierasavainkentälle, sekä koodistojen Koodi-kentälle. Yleistäen, kaikki kentät, jotka kuuluvat luonnolliseen avaimeen, voi olla hyvä indeksoida, etenkin mikäli latauksessa ei käytetä Avain-kenttää. Lisäksi kaikki Alkamis- ja Päättymispäivämäärä –kentät ovat hyviä indeksoinnin kohteita.